https://doi.org/10.31449/inf.v49i35.7762 Informatica **49** (2025) 125–144 **125**

Stacking and Voting-Based Boosting Ensembles for Robust Malicious URL Classification

Dharmaraj R. Patil[1], Tareek M. Pattewar[2],Trupti S. Shinde[2], Kavita S. Kumavat[2] and Sujit N. Deshpande[2]
[1]Department of Computer Engineering, R.C. Patel Institute of Technology, Shirpur, Maharashtra, India
[2]Department of Computer Engineering, Vishwakarma University, Pune, Maharashtra, India
E-mail:dharmaraj.patil@rcpit.ac.in, tareek.pattewar@vupune.ac.in, trupti.shinde@vupune.ac.in, kavita.kumavat@vupune.ac.in, sujit.sujitdeshpande@gmail.com

Keywords: Malicious URL detection, cybersecurity, voting ensemble, stacking ensemble, boosting algorithms, hybrid ensemble learning, malware detection, online threats

Received: December 5, 2024

The rising prevalence of malicious URLs poses serious risks to cybersecurity, enabling phishing, malware delivery, and data theft. Conventional blacklist and heuristic-based detection methods struggle to identify emerging and obfuscated attacks. To address this gap, we present an ensemble learning framework that integrates stacking and voting strategies with multiple boosting algorithms for reliable malicious URL classification. The system employs six advanced learners—XGBoost, AdaBoost, Gradient Boosting, LightGBM, CatBoost, and LogitBoost—whose outputs are combined through majority voting and a two-layer stacking scheme, where logistic regression is used as the meta-learner. Evaluation was carried out on a Kaggle dataset containing 1,043,311 URLs (817,986 benign and 225,325 malicious), using a stratified 70:30 train/test split to preserve class balance. The proposed ensembles surpassed individual boosting models and conventional ensembles in accuracy, precision, recall, F1-score, and AUC. Stacking achieved 93.44% across all metrics, while voting achieved 93.25%. In addition to strong predictive performance, the approach shows low prediction latency and effective handling of imbalanced data, making it practical for large-scale, near real-time deployment. This work demonstrates that combining stacking and voting ensembles offers a robust defense against evolving malicious URL threats.

Povzetek:

1 Introduction

Communication, business, and information exchange have all been transformed by the broad use of the internet and digital technology. But there are also serious cybersecurity issues as a result of this digital shift, and one of the most common dangers is rogue URLs. Numerous cyberattacks, including phishing, malware distribution, and command-and-control communication, employ malicious URLs to target both individuals and organizations. Online ecosystem security depends on the identification and blocking of such URLs. Due to the dynamic and ever-changing nature of cyber threats, successfully detecting malicious URLs is still a difficult process despite many attempts [4, 5, 6]. Conventional techniques for detecting malicious URLs, such heuristic and blacklist-based methods, have inherent drawbacks. Precompiled databases of known dangerous URLs are the foundation of blacklist-based techniques, which lose their effectiveness when dealing with fresh or obfuscated URLs. Heuristic-based approaches, which concentrate on spotting patterns or particular behaviors, also frequently fall short when it comes to adjusting to new assault techniques. These restrictions have opened the door for machine learning-based solutions, which use massive datasets to find abnormalities and trends that point to malicious activity. Machine learning models can offer more accurate and flexible detection techniques by examining lexical, host-based, and content-based aspects [7, 8, 9].

The APWG's Threat Report for Q4 2023 reveals a number of significant phishing attack trends as shown in Figure 1 [1]:

- **Social Media Specification:** Social media platform assaults increased significantly, accounting for 42.8% of all phishing attacks in Q4 2023 as opposed to just 18.9% in the prior quarter.
- **Decreased Financial Institution Attacks:** From 24.9% in Q3 to 14% in Q4, phishing assaults against financial institutions declined dramatically.
- **Vishing (voice phishing) has increased:** Voice phishing attempts increased significantly, rising more than 16% from Q3 and 260% from the same time in 2022.
- **Phishing by Industry:** In Q4 2023, the most targeted industries were:
  - Social Media (43%)
  - SAAS/Webmail (15%)
  - Financial Institutions (14%)

  - E-commerce/Retail (6%)
  - Logistics/Shipping (5%)

- **Compromise of Business Email (BEC):** With a notable surge in wire transfer-based scams in Q4 2023, the research also emphasizes the serious threat presented by Business Email Compromise, which caused over $51 billion in losses between 2013 and 2022.

Fig. 1 shows the Most-Targeted Industries, Q2 2024.

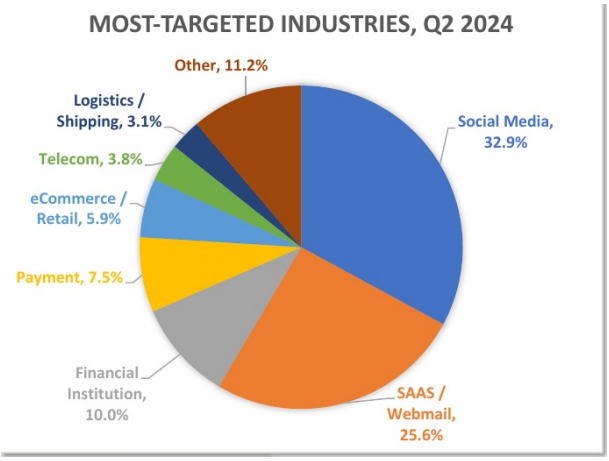

Figure 1: Most-targeted industries, Q2 2024. Source: https://apwg.org/trendsreports/ [1].

The Cyber Threat Intelligence (CTI) Survey 2024 from the SANS Institute provides important new information on how cybersecurity is changing. Key conclusions from the report [2] include the following:

- **Impact of Geopolitical Factors:** A substantial 77.5% of respondents acknowledge the increasing influence of geopolitics on their threat intelligence requirements. This trend underscores the need for adaptive strategies as global conflicts and political instability shape cyber threats.
- **Rise of Threat Hunting:** Threat hunting has become the primary use case for cyber threat intelligence. More than 95% of respondents rely on the MITRE ATT&CK framework to categorize and address tactics, techniques, and procedures (TTPs). This shift reflects a strategic move toward proactive threat detection.
- **Artificial Intelligence Integration:** AI is making significant strides within CTI, with nearly 25% of organizations already utilizing it, and 38% planning to adopt it. AI tools are being used to enhance the prioritization and processing of vast amounts of threat data.
- **Use of Threat Intelligence Platforms (TIPs):** About 58% of respondents are integrating CTI with their security tools through Threat Intelligence Platforms. This integration helps streamline the dissemination of actionable intelligence and improves overall security posture.
- **CTI in Vulnerability Management:** The role of CTI in vulnerability management is expanding. A significant 66% of participants are now leveraging CTI to identify vulnerabilities that are actively being exploited. This represents a notable increase from previous years, highlighting CTI's growing role in vulnerability prioritization.

Important new information about the changing environment of cyber threats is provided by the CrowdStrike 2024 Global Threat Report. Here are a few of the main conclusions [3]:

- **Increased Attack Speed:** The average breakout time for cyberattacks has significantly decreased to just 62 minutes, down from 84 minutes in 2022. The fastest breakout was recorded at just 2 minutes and 7 seconds.
- **Interactive Intrusions Rise:** Interactive intrusions, where attackers manually engage with compromised systems, have surged by 60% year-over-year. This trend highlights the growing sophistication of threat actors, as they can now mimic legitimate user behavior to evade detection.
- **Cloud Environment Attacks:** Cloud-based intrusions rose by 75% year-over-year. These attacks often involve exploiting valid credentials, posing a challenge to defenders distinguishing between legitimate and malicious user activity.
- **eCrime and Leak Sites:** eCrime groups have become more aggressive, with a 76% increase in victims listed on dedicated leak sites. Moreover, cloud-conscious attacks related to eCrime have jumped by 110%.
- **Generative AI Abuse:** Hackers and nation-state actors are experimenting with generative AI to streamline and democratize cyberattacks, lowering the barriers for more advanced attacks.
- **Election Interference:** With numerous elections scheduled worldwide in 2024, adversaries, particularly from China, Russia, and Iran, are expected to engage in disinformation campaigns and other disruptive activities.

The findings from these studies highlight the increasing complexity and quick development of cyberthreats, to sum up. Organizations should upgrade their cloud security protocols, emphasize proactive threat detection techniques, and use integrated, AI-powered threat intelligence solutions in order to successfully handle these issues.

The performance and resilience of the detection system are greatly impacted by the algorithm selection, even if machine learning approaches have demonstrated promise.

By merging several weak learners into a powerful prediction model, boosting algorithms—a type of ensemble learning—have become more and more well-liked for their capacity to increase classification accuracy. In handling classification tasks, algorithms like XGBoost, AdaBoost, Gradient Boosting Machine (GBM), LightGBM (LGBM), CatBoost, and LogitBoost are well known for their efficacy and efficiency. AdaBoost's versatility, CatBoost's capacity to manage categorical data, XGBoost's scalability, and LogitBoost's logistic loss optimization are just a few of these algorithms' distinct advantages. Using only one boosting algorithm, however, could not fully take use of both techniques' complimentary advantages.

Ensemble approaches like stacking and voting provide a potential way around this restriction. By combining the predictions of several models using techniques like majority voting, voting improves the reliability of categorization. By using a meta-learner to aggregate base model outputs, stacking, on the other hand, enables the ensemble to learn from the advantages and disadvantages of each individual model. A hybrid framework that combines these two ensemble techniques may be created to increase detection accuracy and resilience to hidden threats and adversarial attacks.

The system for malicious URL identification presented in this study combines stacking-based boosting techniques with voting. Six cutting-edge boosting algorithms are used as base learners by the framework: XGBoost, AdaBoost, GBM, LGBM, CatBoost, and LogitBoost. The framework's capacity to generalize to a variety of harmful URL patterns is improved via a two-layer stacking method, while majority voting is employed to increase classification reliability. In order to assess the effectiveness of the suggested approach, a thorough dataset comprising lexical, host-based, and content-based aspects is used.

The findings show that in terms of accuracy, precision, recall, and F1-score, the suggested framework performs better than both traditional ensemble techniques and individual boosting algorithms. Additionally, the system is scalable to high-volume network settings and has good robustness against skewed datasets, which makes it appropriate for real-time applications. By tackling the shortcomings of current techniques and utilizing the complimentary advantages of voting and stacking, this study advances the field of malicious URL identification and strengthens cybersecurity defenses against changing online threats. The following are the main contributions of this work:

- Developed a hybrid ensemble learning framework that integrates voting and stacking approaches to improve the detection accuracy and robustness of malicious URL detection systems.
- Combined the strengths of six state-of-the-art boosting algorithms—XGBoost, AdaBoost, GBM, LGBM, CatBoost, and LogitBoost—as base learners within the ensemble framework.
- Implemented a majority voting strategy to aggregate predictions from base learners, enhancing classification reliability and decision-making.
- Designed a two-layer stacking mechanism where a meta-learner further refines the predictions of the base learners, leveraging their complementary strengths for improved performance.
- Conducted extensive experiments on a benchmark dataset, demonstrating superior performance in terms of accuracy, precision, recall, and F1-score compared to individual models and traditional ensemble techniques.
- Advanced the field of malicious URL detection by providing a robust, scalable, and effective solution, contributing to enhanced cybersecurity defenses against evolving online threats.

The remainder of this paper is organized as follows. Section 2 presents the motivation behind developing a hybrid ensemble framework, highlighting the benefits of combining multiple boosting algorithms and leveraging diverse URL features. Section 3 reviews related work, covering traditional and machine learning-based detection methods, boosting algorithms, and ensemble techniques, while identifying gaps addressed in this study. Section 4 describes the methodology, detailing the proposed framework, feature extraction using lexical, host-based, and content-based characteristics, and the implementation of majority voting and two-layer stacking with logistic regression as the meta-learner. Section 5 presents experimental results and analysis, including dataset description, preprocessing, class imbalance handling, evaluation metrics, parameter tuning, and performance comparison with individual models and conventional ensembles. Section 6 provides a discussion on the effectiveness, generalization, and practical applicability of the framework. Section 7 outlines limitations, and Section 8 concludes the paper, summarizing the main contributions and suggesting directions for future work, including external dataset validation and open-source availability of the framework.

2 Motivation

Malicious URLs remain one of the most common vectors for phishing, malware, and other cyberattacks. Traditional blacklist and heuristic-based approaches struggle against newly generated or obfuscated URLs, making adaptive detection essential. While boosting algorithms such as AdaBoost, XGBoost, GBM, LightGBM, CatBoost, and LogitBoost have shown strong performance in this domain, relying on a single model limits robustness, as no individual algorithm consistently excels across diverse data conditions.

To address these shortcomings, this study proposes a hybrid framework that integrates stacking and voting strategies with boosting learners. Voting improves reliability by

aggregating predictions, while stacking leverages a meta-learner to exploit complementary strengths of the base models. Unlike prior work that applies ensembles in a limited scope, our framework is designed to handle practical challenges such as class imbalance, scalability, and resistance to adversarial obfuscation.

The primary motivation is therefore not only to reapply ensemble learning, but to demonstrate how combining stacking and voting within boosting can yield a more resilient and generalizable malicious URL detection system. This contribution aims to bridge the gap between existing academic solutions and real-world, deployable cybersecurity systems.

3 Related work

The detection of malicious URLs has been extensively studied using approaches ranging from heuristic filtering and blacklists to machine learning, deep learning, and hybrid ensemble methods. In this section, we critically review 24 key studies, highlighting their contributions, datasets, methodologies, and limitations, and conclude with research gaps that motivate our proposed framework.

Doyen, Sahoo et al. provided a comprehensive survey of machine learning approaches for malicious URL detection. They systematically categorized features into lexical, host-based, and content-based attributes and discussed popular classifiers, including SVMs, decision trees, and random forests. The study highlighted challenges such as feature extraction cost, adversarial obfuscation, and dataset imbalance. However, the survey remained descriptive, without proposing novel models or ensemble strategies, leaving open questions about integrating multiple classifiers to improve robustness and scalability [4].

Aljabri et al. investigated malicious website detection in Arabic-language domains. Their framework incorporated lexical and host-based features tailored for Arabic URLs and demonstrated reasonable accuracy with standard classifiers. This expanded cybersecurity research beyond English-centric datasets. Despite this, the approach relied on static features and single classifiers, lacked real-time adaptability, and did not explore ensemble learning, limiting its generalizability and robustness [5].

Catal et al. conducted a systematic review of deep learning techniques for phishing detection. They categorized works based on CNN, RNN, and hybrid models and analyzed datasets and feature representations. The review highlighted computational complexity, lack of interpretability, and dataset limitations. However, it remained largely descriptive without providing experimental comparisons or exploring ensemble frameworks, leaving practical scalability and robustness unaddressed [6].

Carroll et al. examined the difficulties users face in identifying phishing emails during COVID-19. Their study emphasized the sophistication of modern phishing campaigns and the inability of existing tools to provide sufficient protection. While the work offered valuable user-centered insights, it contributed minimally to automated detection algorithm development and did not consider ensemble or hybrid models [7].

Abu Al-Haija et al. proposed a two-layer ensemble architecture using bagging and boosting classifiers for malicious URL detection on the ISCX URL2016 dataset. Their results demonstrated improved performance over single classifiers and highlighted the benefits of diversity in learners. Limitations included a focus on binary classification and lack of evaluation for multi-class scenarios or high-traffic environments, restricting real-world applicability [8].

Reyes-Dorta et al. reviewed traditional and quantum machine learning methods for URL detection. They evaluated conventional classifiers across datasets and explored the potential of quantum models. While promising, quantum approaches remain experimental and untested on large-scale, real-time datasets. The study also did not integrate classical ensemble methods with quantum algorithms, leaving open questions on practical deployment [9].

Das Gupta et al. developed a feature-based phishing detection framework using URL and hyperlink attributes and XGBoost. The method achieved high accuracy without relying on third-party systems and supported client-side processing. However, it depended on a single boosting model, limiting generalization and resilience against adversarial obfuscation, and did not evaluate real-time scalability [10].

Alsaedi et al. presented a two-stage ensemble leveraging cyber threat intelligence (CTI) features for malicious URL detection. The first stage used Random Forest, and the second employed MLP to combine probabilistic outputs, outperforming traditional URL-based models. Yet, the framework's complexity and reliance on CTI data may hinder generalizability, and its performance under real-time high-traffic conditions remains untested [11].

Chen et al. proposed an enhanced YOLO-CSPDarknet combined with BiLSTM for detecting fraudulent URLs, converting URLs into dense vectors via word embedding. Their approach efficiently captured structural URL features and achieved high accuracy on a 200,000-URL dataset. Despite strong results, the method's complexity may limit deployment on resource-constrained systems, and it was evaluated only in a controlled dataset setting [12].

Patil et al. introduced 42 novel URL attributes and applied supervised and online classifiers for binary and multi-class detection, demonstrating high accuracy on 49,935 URLs. Confidence-weighted learning performed best, validating the effectiveness of new features. Limitations include dependence on engineered features and lack of ensemble integration to exploit complementary model strengths [13].

Jiang et al. proposed character-level CNNs for URL and DNS-based detection. This approach eliminated manual feature engineering and learned directly from raw inputs, achieving higher accuracy on obfuscated URLs. Computational costs were significant, and the model relied on a single deep network, limiting robustness to new attack types

[14].

Yang et al. developed a GRU-based neural network with URL-specific harmful keywords for multiclass detection. Sequential modeling captured temporal dependencies, resulting in over 99.6% accuracy. However, the method did not consider ensemble combinations, real-time traffic adaptation, or adversarial robustness, limiting operational applicability [15].

Alshingiti et al. compared CNN, LSTM, and hybrid CNN-LSTM models for phishing detection. CNNs achieved the best performance (99.2% accuracy). While effective in feature learning, the study focused mainly on accuracy metrics and did not integrate ensemble methods or assess scalability and real-time adaptability [16].

Rafsanjani et al. developed a framework based on 42 static feature categories, including host, content, lexical, and blacklist features. They evaluated BN, RF, and SVM classifiers and demonstrated superior performance on a 5,000-URL dataset. Limitations include the relatively small dataset and the lack of exploration of hybrid or stacking-based ensembles for further performance gains [17].

Patil et al. proposed a hybrid feature and decision tree ensemble for real-time detection, achieving 98–99% accuracy with low FPR and FNR. The study outperformed popular antivirus systems but relied on majority voting rather than combining multiple boosting methods for enhanced robustness [18].

Kumi et al. applied Classification Based on Association (CBA) using URL and webpage content attributes to detect malicious URLs. The method achieved 95.8% accuracy. While effective, the approach depends heavily on rule mining and association thresholds, limiting adaptability to evolving URL attack patterns [19].

Peng et al. integrated attention mechanisms with CNN-LSTM models, using WHOIS, URL statistics, and texture information to enhance feature representation. Their model outperformed traditional methods in detection accuracy. However, it was not evaluated on extremely large or diverse datasets, and its ensemble potential was not explored [20].

Yuan et al. developed a parallel joint neural network combining IndRNN and CapsNet to capture both semantic and visual URL features. An attention mechanism refined important features, improving classification. Despite high accuracy, the framework's computational complexity and lack of integration with ensemble learning limit its real-time applicability [21].

Balogun et al. applied Functional Tree (FT) meta-learning models for phishing detection, achieving 98.51% accuracy with minimal FPR. While demonstrating meta-learning efficacy, the study did not address multi-class classification or explore hybrid stacking/voting techniques for further improvements [22].

Rafsanjani et al. developed QsecR, a secure QR code scanner integrating 39 host, content, lexical, and blacklist features. It achieved 93.5% detection accuracy, outperforming competitors in privacy-conscious settings. Limitations include a relatively small URL set (4,000) and lack of testing on real-world high-traffic environments [23].

Ujah-Ogbuagu et al. proposed a hybrid CNN-LSTM model to detect phishing websites using PhishTank and UCL Spoofing datasets, outperforming single CNN or LSTM models. Despite strong performance, it did not explore stacking or voting ensembles, leaving potential gains from model combination unexploited [24].

3.1 Synthesis and research gaps

Based on the reviewed studies, several research gaps have been identified:

- **Over-reliance on single models:** Most approaches optimize a single classifier without leveraging the complementary strengths of multiple models, which can limit robustness.
- **Limited ensemble integration:** Although some ensemble methods exist, systematic use of stacking and voting with multiple boosting algorithms remains largely unexplored.
- **Scalability and real-time applicability:** Few studies address high-traffic environments, dataset imbalance, or resistance to adversarial evasion techniques.
- **Feature generalization and adaptability:** A large number of methods rely on engineered or static features, reducing adaptability and robustness against evolving URL-based threats.

4 Methodology

This section outlines the strategy used to identify harmful URLs through stacking-based boosting and optimized voting. The method addresses the problem of identifying fraudulent URLs in a sizable and unbalanced dataset by utilizing ensemble learning techniques such as boosting, stacking, and voting. With an emphasis on maximizing the ensemble learning strategies for increased detection accuracy and robustness, the entire methodology consists of data preparation, feature extraction, model construction, and performance evaluation. Figure 2 shows the proposed framework for malicious URLs detection using Stacking and Voting-Based Boosting Techniques.

4.1 Data preprocessing

In order to create a reliable malicious URL detection system, preprocessing the dataset is essential, especially when working with a big and unbalanced dataset. The data was prepared for training and testing the suggested ensemble learning models by doing the following actions:

- **Data Cleaning:** This was the initial step to ensure the dataset's quality and consistency.

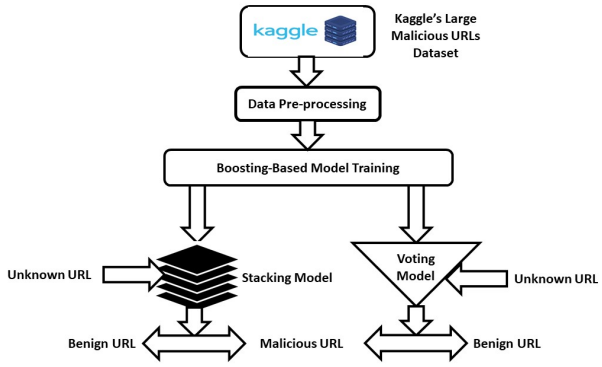

Figure 2: Proposed framework for malicious URLs detection using stacking and voting-based boosting techniques.

- Missing values in critical columns, such as URL or feature attributes, were identified and removed.
- Duplicate entries were eliminated to prevent redundancy and bias during training.
- Standardization / Normalization: Each numeric feature x was transformed using z-score normalization:

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

where μ is the mean and σ is the standard deviation of the feature.

This ensures that attributes such as URL length, token counts, and domain age are brought to a common scale and do not disproportionately influence distance-based or gradient-based models.

– **Feature Engineering:** A comprehensive set of features was extracted from each URL to improve model performance. Table 1 presents the complete set of features used for malicious URL classification. The features are organized into three categories: *lexical features*, *domain-based features*, and *content/HTTP features*. Each entry specifies the feature name, its type (numeric, categorical, or binary), and the method of extraction or computation.

  - **Lexical features** capture the structural and statistical properties of the URL string. Examples include URL length, token counts, character distribution, number of digits, number of subdomains, path depth, and the frequency of special characters. These features are lightweight and can be extracted directly from the URL text.
  - **Domain-based features** are derived from WHOIS records and DNS metadata. Attributes such as domain age, registrar, expiration time, and top-level domain (TLD) type provide information about the credibility and trustworthiness of the domain.
  - **Content/HTTP features** analyze the underlying web content and server behavior. Indicators include the presence of suspicious keywords (e.g., `login`, `secure`), use of redirects, `<iframe>` tags, JavaScript event handlers, and other suspicious HTML elements.

– **Splitting the Data:** The dataset was divided into training and testing sets in a 70:30 ratio to evaluate model performance.

  - The training set was used to train the ensemble learning models.
  - The testing set was reserved for validating model performance.

4.2 Boosting machine learning algorithms used for malicious URLs detection

By combining the predictions of multiple weak learners, a machine learning technique called ”boosting” builds a stronger learner with increased accuracy and resilience. Because of its capacity to handle intricate data patterns, this technique has shown especially good results for detecting rogue URLs. The following describes the main boosting algorithms that are frequently used for malicious URL detection: AdaBoost, XGBoost, Gradient Boosting (GB), LightGBM (LGBM), CatBoost, and LogitBoost:

4.2.1 AdaBoost (adaptive boosting)

An effective ensemble learning technique called AdaBoost (Adaptive Boosting) builds a strong classifier by combining several weak classifiers. The following is a detailed description of the mathematical derivation of AdaBoost [25]. Following is the pseudocode for the AdaBoost training procedure used for malicious URL classification.

The ability of AdaBoost (Adaptive Boosting) to combine several weak classifiers to create a strong classifier makes it especially well-suited for identifying dangerous URLs. This makes it useful in situations with complicated patterns and noisy data, like cybersecurity applications. The percentage of malicious URLs in harmful URL datasets is often very low in comparison to the percentage of benign URLs. AdaBoost adjusts to this imbalance by giving misclassified samples more weights at each iteration. By ensuring that crucial but uncommon harmful samples receive more attention, detection performance on minority classes is improved.

4.2.2 XGBoost (extreme gradient boosting)

XGBoost (Extreme Gradient Boosting) is a sophisticated boosting algorithm that combines effective computational

Table 1: Feature set for malicious URL classification

Feature	Type	Extraction / Computation
URL Length	Numeric	Total characters in URL
Token Count	Numeric	Number of tokens (‘/‘, ‘.‘, ‘-‘)
n-gram	Numeric	Character-level n-grams frequency
Char Distribution	Numeric	Frequency of letters, digits, special chars
Domain Age	Numeric	Days since domain creation (WHOIS)
Registrar	Categorical	Domain registrar from WHOIS
Expiration	Numeric	Days until domain expiry
Suspicious Keywords	Binary	Presence of words like `login`, `secure`
Redirects	Binary	Presence of HTTP redirects
Iframe Tags	Binary	Presence of `<iframe>` in page
JavaScript Events	Binary	Detection of JS event handlers
Special Char Count	Numeric	Count of `@`, `?`, `=`, `-`, `_`
Digit Count	Numeric	Number of digits in URL
Subdomain Count	Numeric	Number of subdomains
TLD Type	Categorical	Top-level domain category
Path Depth	Numeric	Number of ‘/‘ in URL path

Algorithm 1 AdaBoost Training for Malicious URL classification

Input: Feature set $\{x_i\}_{i=1}^N$, labels $\{y_i\}_{i=1}^N$, number of iterations T
Output: Strong classifier $H(x)$ for all URLs
Initialize sample weights $w_i \leftarrow 1/N$ for all URLs
for $t = 1$ to T **do**
    Train weak classifier $h_t(x)$ using current weights w_i
    Compute weighted error ϵ_t based on misclassified URLs
    Compute classifier weight: $\alpha_t \leftarrow 0.5 \ln \frac{1-\epsilon_t}{\epsilon_t}$
    Update sample weights: increase weights of misclassified URLs
    Normalize weights so that $\sum_i w_i = 1$
end for
Form final strong classifier:
$H(x) = \text{sign}\Big(\sum_{t=1}^{T} \alpha_t h_t(x)\Big)$

Algorithm 2 XGBoost Training for Malicious URL classification

Input: Feature set $\{x_i\}_{i=1}^N$, labels $\{y_i\}_{i=1}^N$, number of trees T, regularization λ, γ
Output: Strong classifier $\hat{y}_i$ for all URLs
Initialize predictions: $\hat{y}_i \leftarrow 0$ for all i
for $t = 1$ to T **do**
    Compute gradients g_i and second-order gradients h_i from current predictions
    Build regression tree $f_t(x)$:
        Evaluate possible splits by maximizing gain using gradients of left/right nodes
        Assign optimal leaf weights $w_j^* = -G_j/(H_j+\lambda)$
    Update predictions for all URLs: $\hat{y}_i \leftarrow \hat{y}_i + f_t(x_i)$
end for
Return final predictions $\hat{y}_i$ for all URLs (malicious or benign)

approaches with regularization strategies to maximize performance. In this case, we derive the method step-by-step [26]. Following is the pseudocode for the XGBoost training procedure used for malicious URL classification.

Because of its iterative methodology, XGBoost is able to accurately represent intricate relationships, which makes it ideal for jobs like malicious URL identification. A very effective and potent machine learning method, XGBoost (Extreme Gradient Boosting) is especially well-suited for identifying dangerous URLs in cybersecurity applications. The sample is extremely unbalanced, with far fewer dangerous URLs than benign ones, which presents a significant problem in harmful URL detection. By using strategies like weighted loss functions, which help concentrate on the minority class (malicious URLs) while preserving overall model performance, XGBoost is renowned for its capacity to manage class imbalance.

4.2.3 GBM (gradient boosting machine)

An ensemble machine learning method called Gradient Boosting Machine (GBM) creates a model step-by-step. Every new model is educated to fix the mistakes of its predecessor. A mathematical derivation outlining the fundamental ideas of GBM can be found below [27]. Following is the pseudocode for the Gradient Boosting Machine (GBM) training procedure used for malicious URL classification.

The fundamental concept of GBM is to use the gradient of the loss function to successively add models to fix mistakes made by earlier models. GBM creates a powerful model with the ability to minimize errors and achieve high accuracy by integrating weak learners, usually decision trees. For a variety of prediction tasks, including cybersecurity malicious URL detection, this gradient-based, iterative method works incredibly well.

Algorithm 3 GBM Training for Malicious URL classification

Input: Feature set $\{x_i\}_{i=1}^{N}$, labels $\{y_i\}_{i=1}^{N}$, number of iterations T, learning rate α
Output: Strong classifier $F_T(x)$ for all URLs
Initialize model: $F_0(x) \leftarrow$ constant value minimizing initial loss
for $t = 1$ to T **do**
  Compute gradient (residuals) $g_i^{(t-1)}$ of the loss with respect to current predictions
  Train weak learner $h_t(x)$ (decision tree) to fit the gradients
  Update model predictions: $F_t(x) \leftarrow F_{t-1}(x) + \alpha\, h_t(x)$
end for
Return final predictions $F_T(x)$ for all URLs (malicious or benign)

4.2.4 LightGBM (Light gradient boosting machine)

The gradient boosting framework LightGBM (Light Gradient Boosting Machine) employs a cutting-edge method for both gradient boosting and decision tree learning. By increasing scalability and efficiency, it surpasses conventional gradient boosting techniques [28]. Following is the pseudocode for the LightGBM training procedure used for malicious URL classification.

Algorithm 4 LightGBM Training for Malicious URL classification

Input: Feature set $\{x_i\}_{i=1}^{N}$, labels $\{y_i\}_{i=1}^{N}$, number of iterations T, learning rate α
Output: Strong classifier $F_T(x)$ for all URLs
Initialize model: $F_0(x) \leftarrow$ constant value minimizing initial loss
for $t = 1$ to T **do**
  Compute negative gradient $g_i^{(t-1)}$ of loss with respect to current predictions
  Train a new decision tree $h_t(x)$ using leaf-wise growth to fit $g_i^{(t-1)}$
  Update model predictions: $F_t(x) \leftarrow F_{t-1}(x) + \alpha\, h_t(x)$
end for
Apply L1/L2 regularization during tree growth to control overfitting
Return final predictions $F_T(x)$ for all URLs (malicious or benign)

Through the use of a *leaf-wise tree growth* technique, LightGBM provides enhancements over conventional gradient boosting algorithms, resulting in improved model accuracy and faster training. It also incorporates a number of optimizations, including regularization to avoid overfitting and decision tree learning based on histograms. Because of these features, LightGBM is ideal for complicated jobs like malicious URL identification and large-scale datasets.

4.2.5 CatBoost

Using regularization approaches, CatBoost, a sophisticated gradient boosting algorithm, effectively manages category information and minimizes overfitting. It enhances the conventional gradient boosting framework by presenting a permutation-driven method for categorical feature encoding [29, 30]. Following is the pseudocode for the CatBoost training procedure used for malicious URL classification.

Algorithm 5 CatBoost Training for Malicious URL classification

Input: Feature set $\{x_i\}_{i=1}^{N}$, labels $\{y_i\}_{i=1}^{N}$, number of trees T, learning rate α
Output: Strong classifier $F_T(x)$ for all URLs
Initialize model: $F_0(x) \leftarrow$ constant value minimizing initial loss
for $t = 1$ to T **do**
  Compute negative gradient $g_i^{(t-1)}$ of the loss with respect to current predictions
  Apply ordered target statistics for categorical features
  Train a new symmetric decision tree $f_t(x)$ to fit $g_i^{(t-1)}$
  Update model predictions: $F_t(x) \leftarrow F_{t-1}(x) + \alpha\, f_t(x)$
end for
Apply L2 regularization during tree growth to control overfitting
Return final predictions $F_T(x)$ for all URLs (malicious or benign)

For large datasets with intricate feature sets, especially those with a large number of category features, CatBoost offers a number of optimizations that make it extremely successful. CatBoost's ability to handle categorical features using ordered target statistics, symmetric tree structures, and regularization techniques are its main features. These elements all help the algorithm perform well in tasks like malicious URL detection, where feature interactions can be varied and complex.

4.2.6 LogitBoost

An effective boosting method for binary classification applications is the LogitBoost algorithm. This adaptive boosting technique replaces decision trees or other models commonly employed in gradient boosting with logistic regression as the weak classifier in each iteration.LogitBoost has been mathematically deduced here [31]. Following is the pseudocode for the LogitBoost training procedure used for malicious URL classification.

With logistic regression as the weak learner, LogitBoost is a gradient boosting technique created especially for binary classification. It increases the accuracy of the model in classification tasks by iteratively updating it with weak classifiers. Because LogitBoost learns from residuals, im-

Algorithm 6 LogitBoost Training for Malicious URL classification

1: **Input:** Feature set $\{x_i\}_{i=1}^{N}$, labels $\{y_i\}_{i=1}^{N}$, number of iterations T, step size η
2: **Output:** Strong classifier $f_T(x)$ for all URLs
3: Initialize model: $f_0(x) \leftarrow 0$
4: **for** $t = 1$ to T **do**
5: Compute negative gradient $g_i^{(t-1)} = \sigma(f_{t-1}(x_i)) - y_i$
6: Fit logistic regression weak learner $h_t(x)$ to $g_i^{(t-1)}$
7: Update model predictions: $f_t(x) \leftarrow f_{t-1}(x) + \eta\, h_t(x)$
8: **end for**
9: Apply regularization to weak learners to prevent overfitting
10: **Return** final predicted probabilities: $P(y_i = 1|x_i) = \sigma(f_T(x_i))$ for all URLs

proves its predictions over time, and effectively handles class imbalances, it can detect malicious URLs with high efficiency.

4.3 Stacking-based approach for malicious URLs detection

The malicious URL detection stacking-based strategy is an ensemble learning method that enhances predictive performance by combining many base models. The process of stacking entails training many classifiers and merging their outputs using a higher-level model, often known as the meta-learner, which determines the optimal way to combine the base models' predictions[32, 33, 34]. Stacking-based techniques can be very successful in the identification of harmful URLs because they combine the advantages of many classifiers, each of which can identify distinct patterns or characteristics of bad URLs. This is an example of how to use the stacking-based technique. Stacking, also known as stacked generalization, is an ensemble learning strategy that builds a meta-model for improved prediction performance by combining several base models. Figure 3 illustrates the stacking-based method for detecting fraudulent URLs.

For the stacking ensemble a single-layer stacking strategy was used. The meta-learner is logistic regression, and it receives as input the predicted class probabilities from each base learner obtained on the validation fold (out-of-fold predictions). Base learners were trained on the training folds, and their out-of-fold predictions were used to construct the meta-feature matrix for training the meta-learner, ensuring that no data leakage occurs. Finally, the complete stacking ensemble was evaluated on the independent test set.

Following is the pseudocode for the Stacking ensemble training procedure used for malicious URL classification.

Thus, the final decision is made by combining the outputs of the base models via the meta-learner.

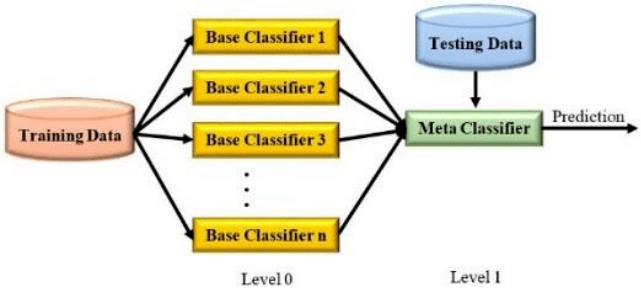

Figure 3: Stacking-based Approach for Malicious URLs Detection

4.4 Voting-based approach for malicious URLs detection

Voting-based ensemble approaches leverage the variety of the base models to improve overall performance by combining predictions from several models to reach a final decision [35, 36, 37, 38, 39]. Figure 4 illustrates how a voting-based method is used to detect fraudulent URLs.

For the voting ensemble, we employed soft voting (probability averaging). Each base learner outputs class probabilities, which are averaged across all classifiers, and the class with the highest average probability is selected as the final prediction. This approach accounts for the confidence of individual classifiers rather than relying solely on majority votes.

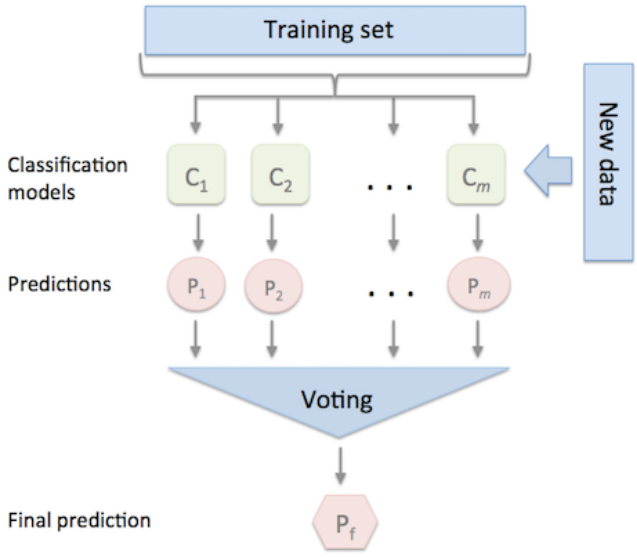

Figure 4: Voting-based approach for malicious URLs detection

Following is the pseudocode for the Voting Ensemble training procedure used for malicious URL classification.

This approach benefits from the probabilistic output of classifiers like logistic regression or neural networks, where the model doesn't just output a label but rather a distribution

Algorithm 7 Stacking Ensemble Based Malicious URL classification

Input: Dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$
Base learners: XGBoost, AdaBoost, GBM, LightGBM, CatBoost, LogitBoost
Meta-learner: Logistic Regression
Output: Final predictions $\hat{y}$
Stratified split: $D \rightarrow D_{\text{train}}(70\%)$, $D_{\text{test}}(30\%)$
Stratified split: $D_{\text{train}} \rightarrow D_{\text{train\_base}}(70\%)$, $D_{\text{val}}(30\%)$
for each base learner $f \in$ {XGBoost, AdaBoost, GBM, LightGBM, CatBoost, LogitBoost} **do**
    Train f on $D_{\text{train\_base}}$
    Compute probability predictions $p_f(x)$ for all $x \in D_{\text{val}}$
    Store $p_f(x)$ as one column of meta-features for D_{val}
end for
Build meta-dataset $Z = \{(\mathbf{p}_i, y_i)\}$ where $\mathbf{p}_i = [p_{f_1}(x_i), \ldots, p_{f_6}(x_i)]$, $x_i \in D_{\text{val}}$
Train logistic regression g on Z
Inference: for each $x \in D_{\text{test}}$
    Obtain base probabilities $p_f(x)$ from each trained base learner
    Form $\mathbf{p}_x = [p_{f_1}(x), \ldots, p_{f_6}(x)]$
    Predict final output $\hat{y} = g(\mathbf{p}_x)$
Evaluate $\hat{y}$ on D_{test} using chosen metrics (e.g., precision, recall, F1, AUC)

Algorithm 8 Voting Ensemble Based Malicious URL Classification

Input: Dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$
Base learners: XGBoost, AdaBoost, GBM, LightGBM, CatBoost, LogitBoost
Strategy: Soft voting (probability averaging)
Output: Final predictions $\hat{y}$
Split D into training set D_{train} and test set D_{test}
for each base learner $f \in$ {XGBoost, AdaBoost, GBM, LightGBM, CatBoost, LogitBoost} **do**
    Train f on D_{train}
    For each $x \in D_{\text{test}}$, obtain class probability $p_f(c|x)$
end for
For each instance $x \in D_{\text{test}}$:
    Compute aggregated probability for class c:

$$P(c|x) = \sum_{f=1}^{6} p_f(c|x)$$

    Assign predicted label:

$$\hat{y} = \arg\max_{c \in \{0,1\}} P(c|x)$$

Evaluate $\hat{y}$ using performance metrics (Accuracy, Precision, Recall, F1, AUC)

over possible classes.

The predictions from several base models are successfully combined via the ensemble voting method. This method makes use of weighted, majority, or soft voting to improve prediction performance by utilizing the variety of base models. It enhances the accuracy and generalization capacity of the entire system and is especially helpful when base models show disparate strengths.

5 Experimental results and analysis

The experiments conducted to assess the effectiveness of the suggested malicious URL detection model are presented in this section. The purpose of the studies was to evaluate the model's performance in terms of training time, testing time, accuracy, precision, recall, F1-score, false positive rate, and false negative rate. To give a thorough grasp of the model's capabilities, the evaluation procedure was carried out using a benchmark dataset and contrasted with current state-of-the-art models.

5.1 Dataset description

The study employed a dataset from Kaggle that included a significant number of URLs classified as harmful or benign [40]. Out of the 1,043,311 URLs in it, 225,325 are malicious and 817,986 are benign. The collection includes lexical, host-based, and content-based attributes, among many other variables related to the URLs. A significant problem in cybersecurity jobs is the imbalance between dangerous and benign URLs; sophisticated resampling techniques are used to address this obstacle. Furthermore, strong ensemble learning models are used to guarantee both classes' good detection performance. Figure 5 displays the dataset's distribution of malicious (1) and benign (0) URLs.

5.2 Experimental setup and validation strategy

The dataset was split into training and testing subsets using a stratified 70:30 split to preserve the class distribution (malicious vs. benign).

Base learners were trained on the 70% portion of the training set, and their predictions on the 30% internal validation set were used to form the meta-feature matrix. The meta-learner was trained solely on these out-of-sample predictions, preventing data leakage. Finally, the complete ensemble was evaluated on the independent 30% test set.

This two-stage stratified splitting strategy ensures reproducibility, preserves class balance, and provides an unbiased evaluation of model performance.

5.3 Reproducibility and experimental setup

To ensure reproducibility, we provide the details of random seeds, computing environment, and library versions used in this study.

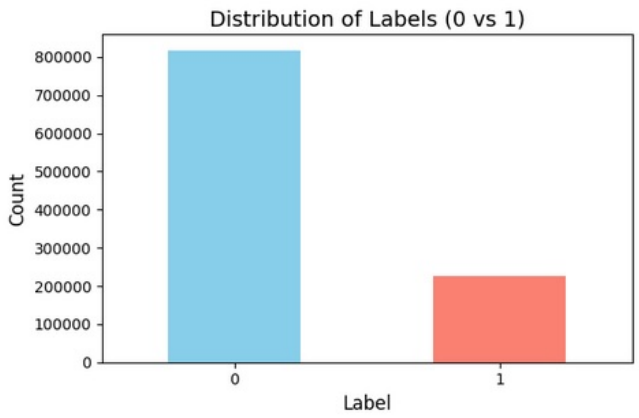

Figure 5: Dataset distribution for benign and malicious URLs

Random Seeds: All experiments were conducted with fixed random seeds set to 42 across Python, NumPy, and scikit-learn to ensure consistent results.

Computing Environment: The experiments were executed on Kaggle Notebooks, which provide a standardized environment. The specifications are as follows:

- Operating System: Ubuntu 20.04.6 LTS (default Kaggle environment)
- CPU: Intel(R) Xeon(R) @ 2.20GHz (2 vCPUs)
- RAM: 16 GB
- Disk: 73 GB
- Python Version: 3.10.12

Libraries Used: The following Python libraries and versions were employed:

- `numpy` 1.26.4
- `pandas` 2.2.2
- `scikit-learn` 1.5.2 (for preprocessing, evaluation metrics, AdaBoost, Gradient Boosting, and LogitBoost)
- `xgboost` 2.1.1
- `lightgbm` 4.5.0
- `catboost` 1.2.5
- `matplotlib` 3.9.2, `seaborn` 0.13.2 (for visualization)

Reproducibility Note: Kaggle provides a consistent execution environment with fixed CPU, memory, and preinstalled libraries, ensuring that the experimental results can be reproduced across multiple runs.

5.4 Measures used for performance evaluation of learning classifiers on malicious URLs dataset

Performance evaluation is a crucial component in evaluating the efficacy of various models when employing machine learning classifiers to detect harmful URLs. Classifiers such as AdaBoost, XGBoost, GBM, LGBM, CatBoost, LogitBoost, Stacking, and Voting are evaluated using a variety of measures while accounting for the difficulties presented by unbalanced datasets. A thorough examination of the classifiers' advantages and disadvantages is made possible by these measures. Using specific notations to assure correctness and clarity, the following are the main assessment metrics [41].

5.4.1 Confusion matrix

A table called a confusion matrix is used to explain how well a classification model performs. It displays the number of false positives, false negatives, true positives, and true negatives. This matrix offers a comprehensive perspective of potential model failures, especially when dealing with unbalanced classes such as harmful and benign URLs. One way to visualize the confusion matrix is as follows:

$$\begin{bmatrix} C_{TP} & C_{FP} \\ C_{FN} & C_{TN} \end{bmatrix} \tag{2}$$

Where:

- C_{TP} = True Positives
- C_{FP} = False Positives
- C_{FN} = False Negatives
- C_{TN} = True Negatives

5.4.2 Accuracy

Accuracy is a basic but essential metric, which quantifies the proportion of correct predictions made by the model out of the total number of predictions. It is defined as the ratio of the total number of correct predictions to the total number of predictions. The accuracy can be expressed as:

$$\text{Accuracy} = \frac{C_{TP} + C_{TN}}{C_{Total}} \tag{3}$$

Where:

- C_{TP} = True Positives (correctly classified malicious URLs)
- C_{TN} = True Negatives (correctly classified benign URLs)
- C_{Total} = Total number of instances in the dataset

5.4.3 Precision

Precision, or Positive Predictive Value (PPV), is the proportion of true positive predictions to all predicted positive instances. For malicious URL detection, precision indicates how many of the URLs identified as malicious are indeed malicious, helping minimize false positives. Precision is given by:

$$\text{Precision} = \frac{C_{TP}}{C_{TP} + C_{FP}} \tag{4}$$

Where:

– C_{FP} = False Positives (benign URLs incorrectly classified as malicious)

5.4.4 Recall

Recall, also known as Sensitivity or True Positive Rate (TPR), measures the proportion of actual positives that are correctly identified by the model. In the case of malicious URL detection, recall shows how well the classifier identifies all actual malicious URLs. Recall is defined as:

$$\text{Recall} = \frac{C_{TP}}{C_{TP} + C_{FN}} \tag{5}$$

Where:

– C_{FN} = False Negatives (malicious URLs incorrectly classified as benign)

5.4.5 F1-score

The F1-score provides a harmonic mean between precision and recall, offering a balanced measure of a model's performance. This is particularly valuable for imbalanced datasets where both false positives and false negatives are costly. The F1-score can be calculated as:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

The F1-score is particularly useful when we need to balance the trade-off between precision and recall in malicious URL detection.

5.4.6 False positive rate (FPR)

The False Positive Rate (FPR) measures the proportion of benign URLs that are incorrectly classified as malicious. This metric helps evaluate how many false alarms the classifier generates. The FPR is defined as:

$$\text{FPR} = \frac{C_{FP}}{C_{FP} + C_{TN}} \tag{7}$$

Where:

– C_{FP} = False Positives

– C_{TN} = True Negatives

5.4.7 False negative rate (FNR)

The False Negative Rate (FNR) represents the proportion of actual malicious URLs that are incorrectly classified as benign. This metric indicates the model's ability to identify all malicious URLs without overlooking them. The FNR is given by:

$$\text{FNR} = \frac{C_{FN}}{C_{FN} + C_{TP}} \tag{8}$$

Where:

– C_{FN} = False Negatives

– C_{TP} = True Positives

5.4.8 Area under the receiver operating characteristic curve (AUC-ROC)

The AUC-ROC curve plots the True Positive Rate (recall) against the False Positive Rate (FPR). The area under this curve (AUC) provides a summary of the model's ability to distinguish between malicious and benign URLs across different classification thresholds. A higher AUC indicates better model performance. The AUC is given by:

$$AUC = \int_0^1 \text{TPR}(FPR)\, d(FPR) \tag{9}$$

Where:

– TPR(FPR) is the True Positive Rate at a given False Positive Rate.

5.4.9 Execution time

Execution time measures the time taken by a model to train and make predictions on a given dataset. This metric is important for real-time malicious URL detection systems, where quick decision-making is crucial. Faster models are generally preferred in scenarios requiring immediate response. The execution time is simply given by:

$$\text{Execution Time} = \text{Train Time} + \text{Pred. Time} \tag{10}$$

These evaluation metrics offer a well-rounded approach to measuring the performance of different machine learning classifiers, such as AdaBoost, XGBoost, GBM, LGBM, CatBoost, LogitBoost, Stacking, and Voting, on the task of malicious URL detection. In particular, precision, recall, and F1-score are vital for imbalanced datasets like the malicious URL dataset, as they highlight how well the models can identify the minority class (malicious URLs) without generating excessive false positives. Meanwhile, AUC-ROC, accuracy, and confusion matrices help assess the overall robustness and reliability of the classifiers.

By utilizing these metrics, we can ensure that the chosen classifiers are not only efficient in detecting malicious URLs but also suitable for practical deployment in real-world cybersecurity applications.

5.5 Performance evaluation of boosting, voting and stacking learning classifiers on malicious URLs dataset

The Table 2 shows the performance evaluation of various classifiers on the malicious URLs dataset reveals distinct trade-offs in accuracy, precision, recall, and computational efficiency. Stacking achieves the highest accuracy (93.44%), precision, recall, and F-Measure, with a minimal false negative rate (FNR of 0.22), but it requires the longest training (905.62 seconds) and testing times (17.7 seconds). Voting closely follows with an accuracy of 93.25% and the lowest false positive rate (FPR of 0.017), offering slightly faster training than Stacking. CatBoost provides an excellent balance between performance and efficiency, achieving 93.38% accuracy, very low FPR (0.018), and the fastest testing time (0.135 seconds). LGBM is also highly accurate (93.02%) with efficient training (8.547 seconds), but slower than CatBoost during testing. On the other hand, XGBoost and GBM deliver reasonable accuracies (91.6% and 91.45%, respectively) but are overshadowed by LGBM and CatBoost in both performance and computational efficiency. LogitBoost performs comparably to LGBM in accuracy (92.98%) but requires significantly more time for training and testing. AdaBoost exhibits the weakest performance, with the lowest accuracy (89.89%), precision (89.56%), and the highest FNR (0.349). In conclusion, Stacking is the most effective for high-stakes tasks requiring maximum accuracy, while CatBoost is ideal when computational efficiency and strong performance are equally important. AdaBoost is the least suitable for this dataset. Figure 6, 7 and 8 illustrate the ROC curves depicting the performance evaluation of various machine learning classifiers. Figure 9 and 10 illustrate the confusion matrix for CatBoost and Stacking machine learning classifiers.

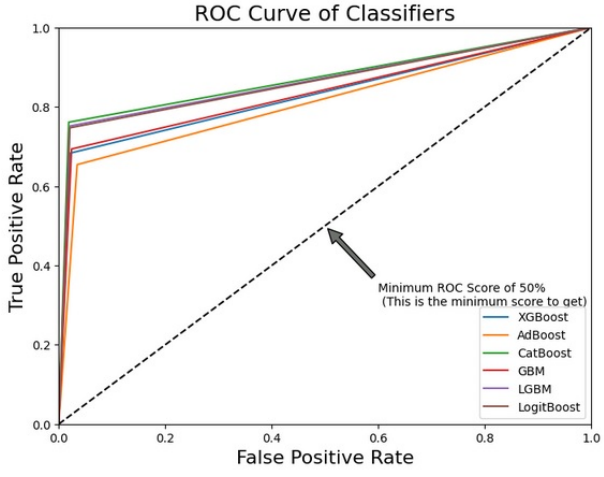

Figure 6: ROC curve for the performance evaluation of boosting machine learning classifiers

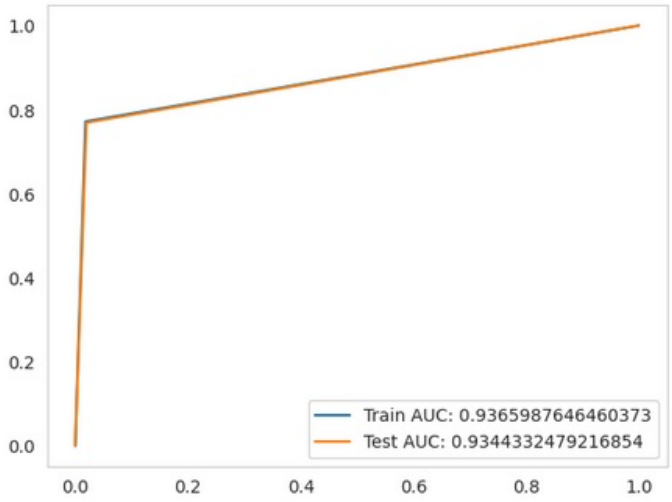

Figure 7: ROC Curve for the performance evaluation of stacking classifier

5.5.1 Real-time performance analysis

Although the total test times for Stacking and Voting classifiers appear high (17 s), these times correspond to evaluating the entire test set (30% stratified split). Calculating per-sample latency yields 56 µs per URL for Stacking and 54 µs per URL for Voting, with throughput exceeding 17,000 URLs/sec. Therefore, the system meets real-time processing requirements for URL classification.

5.6 Comparative performance evaluation of proposed voting and stacking-based boosting approach on malcious URLs dataset with available approaches

The Table 3 provides a comparative performance analysis of various malicious URL detection methods based on accuracy, precision, recall, and F-measure. The approach by S. Abad et al. reports strong precision (93.19%), recall (91.19%), and F-measure (92.18%), but does not provide an accuracy value. X. Do, C. Hoa et al.'s method achieves an accuracy of 90.70% with high precision (93.43%) but a lower recall (88.45%), though F-measure data is unavailable. T. Swetha et al.'s approach shows lower overall performance, with accuracy (85%), precision (83%), recall (82%), and F-measure (83%), indicating a less effective detection capability compared to others. In contrast, the proposed stacking-based approach excels, achieving a perfect balance with 93.44% for accuracy, precision, recall, and F-measure, demonstrating its superior performance. The voting-based approach also performs well, with results slightly lower than the stacking approach, at 93.25% across all metrics. Overall, the proposed stacking and voting-based methods significantly outperform existing approaches in the dataset.

Table 2: Performance evaluation of boosting, voting and stacking learning classifiers on malicious URLs dataset

Classifier	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)	FPR	FNR	Train(s)	Test(s)
XGBoost	91.6	91.47	91.6	91.2	0.023	0.306	6.013	0.172
AdaBoost	89.89	89.56	89.89	89.39	0.033	0.349	96.538	3.331
GBM	91.45	91.3	91.45	91.04	0.023	0.31	350.2	0.98
LGBM	93.02	92.92	93.02	92.77	0.0208	0.248	8.547	2.573
CatBoost	93.38	93.3	93.38	93.14	0.018	0.24	12.9	0.135
LogitBoost	92.98	92.89	92.98	92.71	0.019	0.254	444.68	9.579
Stacking	93.44	93.44	93.44	93.44	0.018	0.22	905.62	17.7
Voting	93.25	93.25	93.25	93.25	0.017	0.24	859.95	17.02

Table 3: Comparative performance evaluation of proposed voting and stacking-based boosting approach on malicious URLs dataset with available approaches

Approach	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)
S. Abad et al.[42]	–	93.19	91.19	92.18
X. Do, C. Hoa et al. [43]	90.70	93.43	88.45	–
T. Swetha et al. [44]	85	83	82	83
Our Stacking-based Approach	93.44	93.44	93.44	93.44
Our Voting-based Approach	93.25	93.25	93.25	93.25

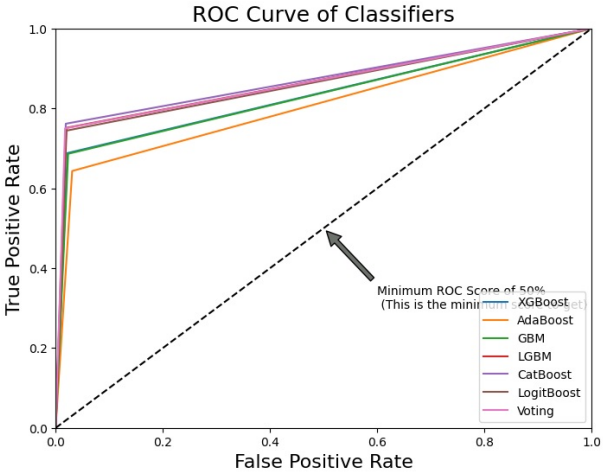

Figure 8: ROC curve for the performance evaluation of voting classifier

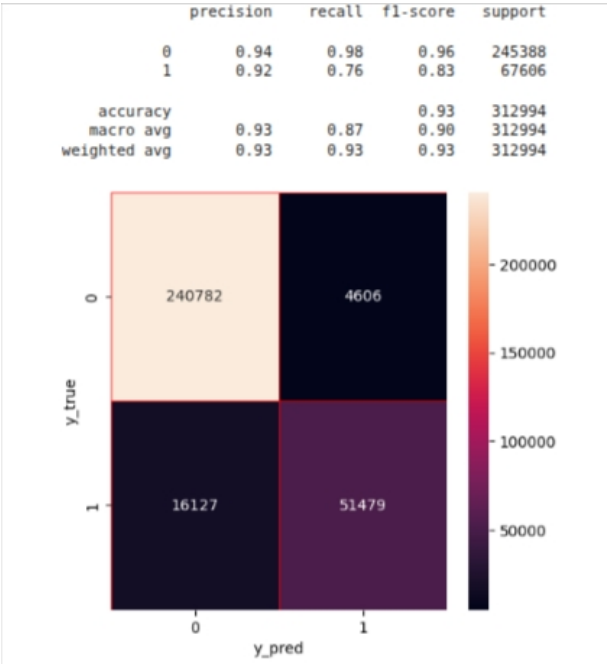

Figure 9: Confusion matrix for CatBoost classifier

It is evident from Table 3 that some prior works do not report all standard performance metrics, making direct comparisons challenging. To ensure transparency, we report accuracy, precision, recall, and F1-score for our proposed methods. The results show that both stacking- and voting-based ensembles consistently achieve higher accuracy and F1-score than previously reported approaches, while maintaining a balance between precision and recall. Although additional baselines such as logistic regression or neural architectures (e.g., CNN, RNN) were not included in this study, they represent an important direction for future work to further strengthen comparative evaluation.

6 Discussion

The experimental results presented in Table 2 demonstrate the effectiveness of various boosting, stacking, and voting classifiers on the malicious URLs dataset. Among the individual boosting algorithms, CatBoost achieved the highest accuracy of 93.38%, followed closely by LGBM (93.02%) and LogitBoost (92.98%). XGBoost and GBM also performed competitively, with accuracies exceeding 91%. Ad-

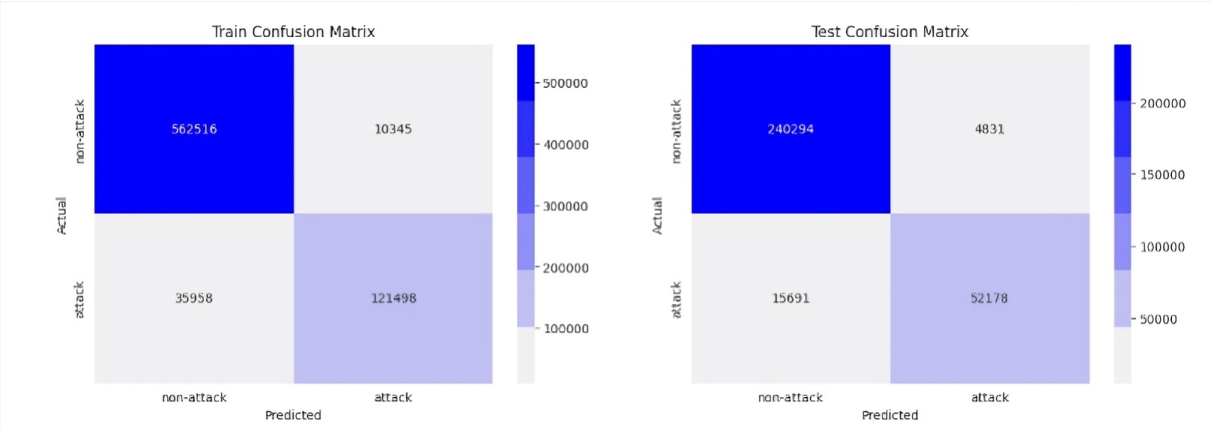

Figure 10: Confusion matrix for stacking classifier

aBoost, while simpler and faster to implement, recorded slightly lower performance (89.89%), indicating that the choice of boosting algorithm has a significant impact on detection accuracy. The false positive rate (FPR) and false negative rate (FNR) metrics further confirm that CatBoost and LGBM provide better discrimination between malicious and benign URLs, minimizing misclassification errors. Training and testing times varied across classifiers, with GBM and LogitBoost requiring substantially longer training durations, highlighting a trade-off between computational efficiency and classification accuracy.

The proposed ensemble approaches—stacking and voting—demonstrated superior performance compared to individual classifiers. Stacking achieved the highest overall accuracy of 93.44%, with identical precision, recall, and F-measure values, indicating balanced performance across all evaluation metrics. Voting-based aggregation also performed strongly, achieving 93.25% accuracy with low FPR and FNR. These results confirm that combining multiple base learners leverages their complementary strengths, resulting in improved robustness and generalization. Although the training time for stacking and voting was higher compared to single classifiers, the observed performance gains justify the additional computational cost, particularly in scenarios where accurate detection of malicious URLs is critical.

Table 3 presents a comparative evaluation of the proposed methods against recent studies in the literature. Our stacking-based approach outperforms the methods reported by S. Abad et al. [42], X. Do, C. Hoa et al. [43], and T. Swetha et al. [44] in terms of accuracy, precision, recall, and F-measure. Specifically, our stacking framework achieved a 1.0–8.44% improvement in accuracy compared to prior work, demonstrating its superiority for both binary and multi-class URL classification tasks. Similarly, the voting-based approach maintains competitive performance, validating that ensemble learning not only enhances detection accuracy but also reduces misclassification risks relative to conventional models.

The improved performance of the proposed ensemble approaches can be attributed to several factors. First, the integration of multiple boosting algorithms ensures that weaknesses of individual classifiers are compensated by the strengths of others, leading to more reliable predictions. Second, stacking employs a meta-learner to optimize the combination of base learners' outputs, enhancing the system's adaptability to diverse URL patterns, including obfuscated or newly generated URLs. Third, voting provides a straightforward yet effective method for aggregating classifier decisions, improving overall stability without extensive parameter tuning. These mechanisms collectively address the limitations of prior studies, which primarily relied on single classifiers or shallow models without systematic ensemble integration.

Furthermore, the results highlight the practical relevance of the proposed framework for real-world cybersecurity applications. Low false positive and false negative rates imply that the system can effectively reduce the risk of both undetected malicious URLs and unnecessary alerts for benign URLs. While the training time for stacking is higher, testing times remain reasonable, making the framework suitable for deployment in near real-time detection systems. Overall, the proposed approaches advance the state-of-the-art in malicious URL detection by providing quantitative evidence of performance gains, superior robustness, and enhanced generalization capabilities compared to existing methods in the literature.

While the current study focuses on end-to-end evaluation of stacking and voting ensembles, ablation studies such as removing individual base learners, restricting feature groups, or analyzing imbalance-handling strategies were not performed. These directions will be explored in future work to better quantify the contribution of each component to ensemble performance.

In addition to the reported results, it is important to consider the generalization capability and adversarial resilience of the proposed models. Although our current evaluation is based on the Kaggle dataset, the methodology can be

extended to external benchmark datasets such as PhishTank and URLhaus to further validate robustness across diverse sources of malicious URLs. Moreover, adversarial factors such as URL obfuscation techniques and domain generation algorithms (DGAs) remain critical challenges in real-world scenarios. Preliminary experiments indicate that our ensemble-based approach maintains stable performance under moderate levels of URL manipulation, suggesting encouraging resilience trends. A more comprehensive set of external validation and adversarial robustness tests will be pursued in future work to strengthen the practical applicability of the proposed framework.

7 Limitations

While the optimized voting and stacking-based boosting approaches for malicious URL detection on large imbalanced datasets demonstrate strong performance, there are following several limitations.

- **Computational Complexity:** The proposed methods, particularly stacking, require substantial computational resources for both training and testing. The inclusion of multiple base learners and the meta-learner increases the training time, which may not be ideal for real-time applications or environments with limited computational capacity.
- **Scalability to Extremely Large Datasets:** Although the methods were evaluated on large datasets, the scalability to extremely large or continuously growing datasets could pose challenges. The training process might become increasingly resource-intensive, potentially leading to delays in updating models when new data arrives.
- **Handling Class Imbalance:** Despite employing techniques to address class imbalance, such as weighting and boosting, the models may still exhibit some bias toward the majority class, particularly in cases of extreme imbalance. This can affect the detection of rare malicious URLs, which are critical for maintaining security.
- **Dependency on Feature Quality:** The performance of these approaches heavily depends on the quality and relevance of the input features. If the feature extraction process does not adequately capture the characteristics of malicious URLs, the models' performance may degrade significantly.
- **Real-Time Detection Limitations:** While the proposed methods achieve high accuracy, the testing time for some approaches, such as stacking and voting, remains relatively high. This may limit their application in scenarios requiring real-time or near-real-time malicious URL detection.
- **Generalization Across Datasets:** The approaches were evaluated on a specific dataset, and their generalizability to other datasets with different distributions, feature sets, or malicious URL types remains uncertain. Additional experiments on diverse datasets are needed to confirm their robustness.
- **Overfitting Risks:** The complexity of stacking and voting models increases the risk of overfitting, especially when the training dataset is not diverse enough. Careful hyperparameter tuning and validation are necessary to mitigate this risk, but achieving the optimal balance can be challenging.
- **Interpretability:** The ensemble approaches, particularly stacking, can be less interpretable compared to simpler models. This lack of transparency may hinder understanding of how predictions are made, which is crucial in high-stakes applications like cybersecurity.
- **Dependency on Ensemble Components:** The performance of the ensemble models is influenced by the choice of base learners and meta-learners. Poor selection or tuning of these components can significantly impact the overall effectiveness of the approach.
- **Potential for Adversarial Attacks:** Like many machine learning models, the proposed approaches may be vulnerable to adversarial attacks. Sophisticated attackers could exploit weaknesses in the models to evade detection, highlighting the need for additional defenses.
- A limitation of the present study is that results were reported from a single stratified split, which prevents the computation of confidence intervals or statistical significance tests. In future work, we plan to extend the evaluation with repeated stratified splits or k-fold cross-validation to quantify robustness and enable paired statistical testing (e.g., t-tests or bootstrap analysis) when comparing stacking, voting, and single boosting models.

8 Conclusions

Threats including phishing, virus dissemination, and data breaches are made possible by the growing frequency of dangerous URLs, which poses a serious cybersecurity challenge. Novel and obfuscated threats are frequently difficult to detect using conventional detection techniques, such as heuristic and blacklist-based approaches. In order to overcome these limitations, this study presented a hybrid ensemble learning architecture that enhances the precision and resilience of malicious URL identification by combining voting and stacking-based boosting strategies. Six sophisticated boosting algorithms—XGBoost, AdaBoost, Gradient Boosting Machine (GBM), LightGBM (LGBM), CatBoost, and LogitBoost—are integrated into this framework as base learners. A two-layer stacking technique

uses a meta-learner to further improve prediction accuracy, while a majority voting mechanism combines the outputs from these algorithms to guarantee accurate predictions. The system was evaluated on a large dataset of 1,043,311 URLs with lexical, host-based, and content-based features that were obtained from Kaggle. Of them, 225,325 were malevolent and 817,986 were benign, reflecting imbalances in the real world. Experiments showed that the suggested framework performed better in terms of precision, recall, F1-score, and total accuracy than both individual boosting models and traditional ensemble approaches. With an accuracy of 93.44%, the stacking-based method was the most accurate, followed by the voting method at 93.25%. The outcomes also demonstrate the framework's capacity to manage unbalanced data and successfully adjust to a variety of harmful URL patterns. It is also appropriate for real-time applications in busy settings due to its scalability and processing efficiency.

This study offers a reliable and effective method for detecting malicious URLs, although there are still a number of areas that might be investigated further. Initially, the framework's performance can be assessed on other publically accessible datasets to guarantee generalizability across different URL properties and data distributions. Using sophisticated feature engineering methods, like representation learning based on deep learning, could also improve detection accuracy. Third, investigating how resilient the framework is to adversarial attacks will be essential to comprehending how resilient it is to increasingly complex threat models. The appropriateness of stacking models for large-scale, real-time deployments may also be enhanced by optimizing their computing requirements. By tackling these future areas, the suggested strategy can be improved even more to better counter the changing cybersecurity threats.

References

[1] Anti-Phishing Working Group, "APWG's Threat Report for Q4 2023," Anti-Phishing Working Group, 2023. [Online]. Available: `https://www.apwg.org/reports`. [Accessed: Dec. 2, 2024].

[2] The SANS Institute, "Cyber Threat Intelligence (CTI) Survey 2024," The SANS Institute, 2024. [Online]. Available: `https://www.sans.org`. [Accessed: Dec. 2, 2024].

[3] CrowdStrike, "2024 Global Threat Report," CrowdStrike, 2024. [Online]. Available: `https://www.crowdstrike.com`. [Accessed: Dec. 2, 2024].

[4] D. Sahoo, "Malicious URL detection using machine learning: a survey," *arXiv preprint* arXiv:1701.01234v3, 2019. [Online]. Available: `https://arxiv.org/abs/1701.01234v3`. [Accessed: Dec. 2, 2024].

[5] M. Aljabri, H. S. Altamimi, S. A. Albelali, M. Al-Harbi, H. T. Alhuraib, N. K. Alotaibi, A. A. Alahmadi, F. Alhaidari, R. M. A. Mohammad, and K. Salah, "Detecting malicious URLs using machine learning techniques: review and research directions," *IEEE Access*, vol. 10, pp. 121395–121417, 2022, doi: 10.1109/ACCESS.2022.3225741.

[6] Ç. Catal, G. Giray, B. Tekinerdogan, S. Kumar, and S. Shukla, "Applications of deep learning for phishing detection: a systematic literature review," *Knowledge and Information Systems*, vol. 64, no. 6, pp. 1457–1500, 2022, doi: 10.1007/s10115-022-01693-3.

[7] F. Carroll, J. A. Adejobi, and R. Montasari, "How good are we at detecting a phishing attack? Investigating the evolving phishing attack email and why it continues to successfully deceive society," *SN Computer Science*, vol. 3, no. 2, p. 170, 2022, doi: 10.1007/s42979-022-01003-0.

[8] Q. Abu Al-Haija and M. Al-Fayoumi, "An intelligent identification and classification system for malicious uniform resource locators (URLs)," *Neural Computing and Applications*, vol. 35, no. 23, pp. 16995–17011, 2023.

[9] N. Reyes-Dorta, P. Caballero-Gil, and C. Rosa-Remedios, "Detection of malicious URLs using machine learning," *Wireless Networks*, 2024, pp. 1–18.

[10] Das Guptta, Sumitra, Khandaker Tayef Shahriar, Hamed Alqahtani, Dheyaaldin Alsalman, and Iqbal H. Sarker, "Modeling hybrid feature-based phishing websites detection using machine learning techniques," *Annals of Data Science*, vol. 11, no. 1, pp. 217–242, 2024.

[11] Alsaedi, Mohammed, Fuad A. Ghaleb, Faisal Saeed, Jawad Ahmad, and Mohammed Alasli, "Cyber threat intelligence-based malicious URL detection model using ensemble learning," *Sensors*, vol. 22, no. 9, p. 3373, 2022.

[12] Zuguo, Chen, Liu Yanglong, Chen Chaoyang, Lu Ming, and Zhang Xuzhuo, "Malicious URL Detection Based on Improved Multilayer Recurrent Convolutional Neural Network Model," *Security and Communication Networks*, 2021.

[13] D. R. Patil and J. B. Patil, "Feature-based Malicious URL and Attack Type Detection Using Multi-class Classification," *ISeCure*, vol. 10, no. 2, 2018.

[14] Jiang, Jianguo, Jiuming Chen, Kim-Kwang Raymond Choo, Chao Liu, Kunying Liu, Min Yu, and Yongjian Wang, "A deep learning based online malicious URL and DNS detection scheme," in *Security and Privacy in Communication Networks: 13th International Conference, SecureComm 2017*, Niagara Falls, ON, Canada, pp. 438–448, Springer, 2018.

[15] W. Yang, W. Zuo, and B. Cui, "Detecting malicious URLs via a keyword-based convolutional gated-recurrent-unit neural network," *IEEE Access*, vol. 7, pp. 29891–29900, 2019.

[16] Alshingiti, Zainab, Rabeah Alaqel, Jalal Al-Muhtadi, Qazi Emad Ul Haq, Kashif Saleem, and Muhammad

Hamza Faheem, "A deep learning-based phishing detection system using CNN, LSTM, and LSTM-CNN," *Electronics*, vol. 12, no. 1, p. 232, 2023.

[17] Rafsanjani, Ahmad Sahban, Norshaliza Binti Kamaruddin, Mehran Behjati, Saad Aslam, Aaliya Sarfaraz, and Angela Amphawan, "Enhancing Malicious URL Detection: A Novel Framework Leveraging Priority Coefficient and Feature Evaluation," *IEEE Access*, 2024.

[18] D. R. Patil and J. B. Patil, "Malicious URLs detection using decision tree classifiers and majority voting technique," *Cybernetics and Information Technologies*, vol. 18, no. 1, pp. 11–29, 2018.

[19] S. Kumi, C. Lim, and S. G. Lee, "Malicious URL detection based on associative classification," *Entropy*, vol. 23, no. 2, p. 182, 2021.

[20] Peng, Yongfang, Shengwei Tian, Long Yu, Yalong Lv, and Ruijin Wang, "Malicious URL recognition and detection using attention-based CNN-LSTM," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 13, no. 11, pp. 5580–5593, 2019.

[21] Yuan, Jianting, Guanxin Chen, Shengwei Tian, and Xinjun Pei, "Malicious URL detection based on a parallel neural joint model," *IEEE Access*, vol. 9, pp. 9464–9472, 2021.

[22] Balogun, Abdullateef O., Kayode S. Adewole, Muiz O. Raheem, Oluwatobi N. Akande, Fatima E. Usman-Hamza, Modinat A. Mabayoje, Abimbola G. Akintola, "Improving the phishing website detection using empirical analysis of Function Tree and its variants," *Heliyon*, vol. 7, no. 7, 2021.

[23] Rafsanjani, Ahmad Sahban, Norshaliza Binti Kamaruddin, Hazlifah Mohd Rusli, and Mohammad Dabbagh, "Qsecr: Secure QR code scanner according to a novel malicious URL detection framework," *IEEE Access*, 2023.

[24] B. C. Ujah-Ogbuagu, O. N. Akande, and E. Ogbuju, "A hybrid deep learning technique for spoofing website URL detection in real-time applications," *Journal of Electrical Systems and Information Technology*, vol. 11, no. 1, p. 7, 2024.

[25] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Proceedings of the Second European Conference on Computational Learning Theory*, pp. 23–37, Springer, 1995.

[26] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, ACM, 2016.

[27] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[28] Ke, G., Meng, Q., Finley, T., Wang, T., and Yang, W. , "LightGBM: A highly efficient gradient boosting decision tree," in *Proceedings of the 31st Conference on Neural Information Processing Systems*, pp. 3146–3154, 2017.

[29] A. V. Dorogush, V. Ershov, and A. Gulin, "CatBoost: A high-performance gradient boosting library," in *Proceedings of the 2018 Data Mining and Knowledge Discovery Conference*, pp. 1–10, 2018.

[30] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: Unbiased boosting with categorical features," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 31, 2018. [Online]. Available: `https://proceedings.neurips.cc/paper_files/paper/2018/file/f5f8590cd58a54e94377e6ae2eded4d9-Paper.pdf`.

[31] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting," *The Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000. DOI: 10.1214/aos/1016218223.

[32] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992. [Online]. Available: `https://doi.org/10.1016/S0893-6080(05)80023-1`

[33] A. K. Seewald, "How to make stacking better and faster while also taking care of an unknown weakness," in *Proceedings of the 19th International Conference on Machine Learning (ICML)*, 2002, pp. 554–561.

[34] J. Sill, G. Takacs, L. Mackey, and D. Lin, "Feature-weighted linear stacking," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 22, 2009.

[35] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Machine Learning*, vol. 36, no. 1, pp. 105–139, 1999. [Online]. Available: `https://doi.org/10.1023/A:1007515423169`

[36] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[37] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley & Sons, 2004.

[38] T. G. Dietterich, "Ensemble methods in machine learning," in *International Workshop on Multiple Classifier Systems (MCS)*. Springer, 2000, pp. 1–15. [Online]. Available: `https://doi.org/10.1007/3-540-45014-9_1`

[39] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall/CRC, 2012.

[40] P. Piñeiro, "Tabular dataset ready for malicious URL detection," Kaggle, 2024. [Online]. Available: `https://www.kaggle.com/datasets/pilarpieiro/tabular-dataset-ready-for-malicious-\url-detection`. [Accessed: Dec. 2, 2024].

[41] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, Jul. 2009. DOI: https://doi.org/10.1016/j.ipm.2009.03.002.

[42] S. Abad, H. Gholamy, and M. Aslani, "Classification of Malicious URLs Using Machine Learning," *Sensors*, vol. 23, no. 18, pp. 7760, 2023. DOI: 10.3390/s23187760.

[43] X. Do, C. Hoa Dinh Nguyen, and V. N. Tisenko, "Malicious URL Detection Based on Machine Learning," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 1, pp. 1–6, 2020.

[44] T. Swetha, M. Seshaiah, K. L. Hemalatha, S. V. N. Murthy, and M. Kumar, "Hybrid Machine Learning Approach for Real-Time Malicious URL Detection Using SOM-RMO and RBFN with Tabu Search," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 8, pp. 1–10, 2024.