Indonesian Journal of Electrical Engineering and Computer Science
Vol. 43, No. 3, September 2026, pp. 908~927
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v43.i3.pp908-927

URL-based phishing detection using XGBoost with engineered features

Jawaher Alharbi, Manal Bayousef, Hind Almisbahi
Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

Article Info

Article history:

Received Apr 11, 2025
Revised Jul 20, 2026
Accepted Aug 19, 2026

Keywords:

Cybersecurity
Machine learning
Phishing website detection
URL-based features
XGBoost model

ABSTRACT

URL-based phishing involves fake uniform resource locators (URLs) created by attackers to trick users into believing they are visiting a legitimate website and thereby steal their confidential information. While several powerful machine learning (ML) and deep learning (DL) studies exist to detect phishing, they still face limitations. Many studies rely on third-party intervention to extract features, which introduces delays that make them unsuitable for fast detection. Another limitation is that existing studies often use small datasets, and traditional features hinder models' ability to learn new phishing techniques, resulting in poor generalization. Therefore, developing new features is crucial to ensure that anti-phishing tools can keep pace with evolving phishing tactics. In addition, the existing studies do not report detection time, which is important for fast detection, and reduces methodological clarity. This paper aims to address these limitations by applying a neural network model and traditional ML classification algorithms to support browser-based phishing detection that balances high accuracy with fast detection. Our XGBoost model achieved 98% accuracy on the test set, utilizing 40 third-party-independent features. Additionally, we achieved an average response time of 0.026225 seconds and an average computation time of 1.6977×10^{-6} seconds per URL, which demonstrates competitive speed. We provided a table of features from recent studies, together with their documented sources, to support future research and analyze key URL-based features.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Jawaher Alharbi
Department of Information Technology, Faculty of Computing and Information Technology
King Abdulaziz University
Jeddah, Saudi Arabia
Email: jalharbi0086@stu.kau.edu.sa

1. INTRODUCTION

Cyber threats have become more sophisticated and complex, making their detection and analysis a constant challenge. Phishing websites are among the most prominent of these threats [1]. In 2022, several domains were registered for phishing websites impersonating the Absher platform, an electronic portal affiliated with the Saudi Ministry of Interior [2]. In 2023, financial phishing attacks accounted for 23.5% of all other attacks [3], while in 2024, credential phishing attacks increased by 703% [4]. What makes the matter even more serious is the projected rise in the annual cost of cybercrimes to $10.5 trillion by 2025 [5]. Therefore, these statistics reflect that the rapid and accurate detection of phishing is an important research challenge. In recent years, many studies have relied on ML classifiers to develop effective mechanisms for detecting phishing attacks [1]. Although these classifications have achieved high detection performance,

current studies still suffer from several limitations. A major limitation is the reliance on third-party services during feature analysis and extraction, which leads to delays in detection [6]. With the evolution of phishing attacks, traditional URL-based features, such as URL length and lexical features, are no longer sufficient to achieve effective detection [7]. However, most studies still rely on traditional features, while few have addressed advanced URL-based features, such as interplanetary file system (IPFS) decentralized phishing hosting, percent-encoding, and Punycode. In addition, previous studies often rely on a limited and outdated dataset. Therefore, the researchers in [8] pointed out the importance of training ML classifiers on a large and up-to-date dataset that includes advanced features to counter complex phishing attacks. This study aims to address these limitations by introducing a URL-based detection system using ML classifiers for phishing detection, balancing speed and detection accuracy without relying on third-party services. Additionally, it presents a dataset that includes both traditional and advanced features. Furthermore, we evaluated the effectiveness of the selected features using six traditional ML algorithms: decision tree, k-nearest neighbors (KNN), random forest (RF), adaptive boosting (AdaBoost), extreme gradient boosting (XGBoost), light gradient boosting machine (LightGBM), and a neural network model, the multi-layer perceptron (MLP). Of these models, XGBoost achieved effective results; therefore, it has been selected as the base model for our research. The main contributions in this research are as follows:

- We created a dataset compiled from recent, publicly available sources that includes URL-based features independent of third-party services. This dataset consists of 40 lexical and structural features, including three features that are not commonly used in previous studies but were identified as important and effective in our model.
- Supplementary 1, Table 1 summarizes the 73 features with their references from recent studies and describes the corresponding feature extraction methods. This provides a useful contribution to phishing detection research, as it supports researchers in identifying and applying feature information.
- Our XGBoost model achieved a balance between high accuracy and speed. It achieved 98% accuracy on the test set with an average computation time of 1.6977×10^{-6} seconds per URL. Furthermore, the model recorded an average response time of 0.026225 seconds per URL in the online setting.

The paper is organized as follows: Section 2 reviews the relevant literature. Section 3 explains the research methodology, detailing the dataset, feature extraction and selection, ML algorithms used, and evaluation criteria. Then, Section 4 presents the experimental results. Section 5 analyses the results and evaluates the performance. Additionally, this section compares the results with the existing literature. After that, the results are discussed, and future recommendations are presented in Section 6. Finally, Section 7 concludes the paper and presents future work.

2. RELATED WORK

Previous studies have explored phishing websites extensively using traditional ML and DL methods. Kocyigit *et al.* [9] focused on developing a detection system by selecting optimal features using optimized genetic algorithm feature selection (GAFS). The researchers used a dataset from a study in [10], containing 36,173 phishing and 51,316 legitimate websites. After applying GAFS, they obtained 44 optimal URL-based features. The results showed that the RF model achieved an accuracy of 92.93% on the test set, with a computation time of 0.6380 seconds. However, the researchers reported that some of the selected features are not suitable for real-time deployment due to the high processing cost. In contrast, Gupta *et al.* [11] proposed a lightweight phishing detection system compatible with internet of things (IoT) devices. Their approach relies on the lexical features of URLs without relying on third-party services. The researchers used a dataset from ISCXURL 2016 provided by the University of New Brunswick (UNB), which consists of 10,000 legitimate URLs and 9,964 phishing URLs, with only 9 lexical features extracted. The experimentation results indicate that the RF model achieved 99.57% on the test set, with a response time of 51.5 milliseconds (0.0515 seconds). The researchers indicated that future work would involve expanding the work using advanced DL algorithms. This future direction reflects the continuing interest in exploring more advanced learning approaches for phishing detection.

Similarly, Ahammad *et al.* [12] evaluated several ML classifiers and trained them on URL-based features to identify phishing URLs. They extracted 15 features from a dataset of 1,500 phishing URLs from PhishTank and 1,500 legitimate URLs from UNB. After several experiments, the LightGBM model achieved an accuracy of 86%. The researchers noted limitations in the study, namely that some URLs do not contain sufficient information in WHOIS databases, resulting in the inability to extract all the required features. They reported the need to gather newer URL-based features to improve accuracy.

Table 1. Summary of related work

Ref.	Model	Method	Dataset size	Number of features	Accuracy	Detection time (seconds)	Dependence on third parties	Limitations
[9], 2024	RF	URL-based	36,173 phishing URLs and 51,316 legitimate URLs.	44	92.93%	0.6380	×	Some selected features are not suitable for real-time deployment due to the high processing cost. Not fully reported extracted feature.
[11], 2021	RF	URL-based	9,964 phishing URLs and 10,000 legitimate URLs.	9	99.57%	0.0515	×	The dataset is old, limited, and contains a small set of traditional features.
[12], 2022	LightGBM	URL-based	1,500 phishing URLs and 1,500 legitimate URLs.	15	86%	Not mentioned	✓	Reliance on third-party services. Unspecified detection time. Low accuracy rate.
[13], 2020	RF	URL-based and content-based	6,157 phishing URLs and 4,898 legitimate URLs.	30	99.36%	4	✓	The study depends on third-party data, delays in detection, and a small dataset.
[14], 2023	LSD model with Canopy feature selection	URL-based	5,000 phishing URLs and 6,000 legitimate URLs.	33	98.12%	370.9845	Not mentioned	Not fully reported extracted feature. Detection time reported without specifying the measurement unit.
[15], 2021	GA–GradientBoost, GA–Bagging, and GA–XGBoost	URL-based and content-based	Dataset 1: 4,898 phishing URLs, 6,157 legitimate URLs. Dataset 2: 5,000 phishing URLs, 5,000 legitimate URLs, and Dataset 3: 30,647 phishing URLs, 58,000 legitimate URLs.	Dataset 1: 30, dataset 2: 48 dataset 3: 111	Dataset 1: 97.16%, dataset 2: 98.58%, dataset 3: 97.39%	Not mentioned	✓	The system suffers from high computational costs, dependence on third-party services, and unspecified detection time.
[7], 2021	RF	URL-based	Legitimate: 35,378, phishing: 9,965, malware: 11,566, and spam: 11,942.	20	99%	4.709	×	The dataset is outdated. Long detection time.
[16], 2021	RNN-GRU	URL-based	60,000 phishing URLs and 60,000 legitimate URLs.	100	99.18%	0.2	×	Limitations in processing URLs over 200 characters and a lack of URL shortening support. Long detection time.
[17], 2022	logistic regression	URL-based	30,000 phishing URLs and 30,000 legitimate URLs.	38	96.50%	Not mentioned	×	Slight decrease in overall accuracy when using login URLs.
Our study	XGBoost	URL-based	56,369 phishing URLs and 39,367 legitimate URLs.	40	98%	0.026225	×	The study relies only on URL features, limiting its effectiveness against content-based phishing attacks.

HR *et al.* [13] introduced a new browser architecture called the embedded phishing detection browser (EPDB) using an RF model. The model was trained on a dataset of 6,157 phishing URLs from PhishTank and 4,898 legitimate URLs from Millers' Miles. The EPDB system uses a rule-based extraction framework to identify 30 website features that combine URL-based and content-based features, using third-

party services. Researchers have confirmed that the proposed system detects newly created phishing pages with 99.36% accuracy, with an average response time of 4 seconds. However, the authors plan to create an updated dataset by collecting phishing URLs from users to train the models on the latest phishing sites. Maintaining up-to-date phishing datasets is essential for improving the robustness of phishing detection models. Karim *et al.* [14] proposed a hybrid model based on three main classifiers: Logistic regression, support vector classifier, and decision tree (LR+SVC+DT), with soft voting and hard voting to detect phishing websites. They also used the Canopy technique for feature selection to improve performance. To train the models, they used a dataset collected from the Kaggle platform, comprising 5,000 phishing URLs and 6,000 legitimate URLs, with 33 URL-based features. The results showed that the LSD model with Canopy feature selection achieved an accuracy of 98.12% on the test set. In addition, the researchers reported an execution time of 370.9845, which measures the time required for their model to complete a detection task. However, they did not specify the exact time unit, leaving the exact measurement unclear. The authors suggested integrating blacklist and whitelist systems with ML systems in the future, with the aim of improving the efficiency of detecting phishing websites. This future direction reflects the continuing interest in improving phishing detection efficiency.

Al-Sarem *et al.* [15] proposed the optimized stacking ensemble model using a genetic algorithm (GA) for developing phishing website detection systems. The researchers used three public datasets: the first, from the University of California, Irvine (UCI), contained 6,157 legitimate URLs and 4,898 phishing URLs, with 30 features. The second, from Mendeley data, contained 5,000 legitimate URLs and 5,000 phishing URLs, with 48 features. The third dataset, also from Mendeley data, contained 58,000 legitimate URLs and 30,647 phishing URLs, with 111 features. All datasets consist of URL-based and content-based attributes. The researchers relied on third-party services, the WHOIS database, to extract some feature information, such as domain name system (DNS), in the detection process. Based on experimental results, GA-GradientBoost, GA-Bagging, and GA-XGBoost were selected as the top three algorithms as base learners, and the support vector machine (SVM) model was adopted as an additional classifier for the system. The proposed models achieved the following results: 97.16%, 98.58%, and 97.39% for the datasets, respectively. The researchers noted that their future work focuses on creating a lightweight detection system suitable for IoT environments to protect against phishing attacks. This observation highlights the importance of achieving both computational efficiency and rapid phishing detection in resource-constrained environments.

Therefore, Raja *et al.* [7] provided a lightweight system to detect phishing websites. The authors used a dataset from the UNB dataset 2016 containing four classes: 35,378 legitimate URLs, 9,965 phishing URLs, 11,566 malware URLs, and 11,942 spam URLs. 20 URL-based features independent of third-party services were extracted. The experimental results showed that the RF model achieved an accuracy of 99% with a computation time of 4.709 seconds. The researchers pointed to the need to extract new features and expand the dataset to support sophisticated phishing detection. This recommendation highlights the importance of advanced feature extraction for detecting evolving phishing attacks. Another approach was proposed by Tang and Mahmoud [16], who presented a DL-based framework that operates independently of third-party services for detecting phishing websites in a real-time browser plugin setting. The researchers evaluated several datasets, the best of which was the KPT-12 dataset, which contains 60,000 legitimate URLs from Kaggle and 60,000 phishing URLs from PhishTank. Regarding features, they use a character-level tokenization approach, limiting each URL to 100 commonly used characters in URLs and excluding URLs longer than 200 characters to reduce computational requirements. After conducting the experiments, the results demonstrated that the RNN-GRU model achieved an accuracy of 99.18% and a response time of less than 200 milliseconds (0.2 seconds). Nevertheless, the authors reported that the model does not support short URLs and that truncating URLs at 200 characters may result in the loss of some features found in longer URLs.

Sánchez-Paniagua *et al.* [17] proposed a framework for detecting phishing URLs by representing a realistic scenario by using URLs for legitimate login pages and phishing ones instead of just URLs for homepages. To implement the proposed system, the researchers created a dataset consisting of 30,000 phishing URLs collected from PhishTank and 30,000 legitimate URLs obtained from the top million Quantcast websites. This dataset incorporates 38 lexical features and natural language processing (NLP) properties, such as word analysis and patterns. After training, the logistic regression model achieved the best performance with an accuracy of 96.50%. However, the authors highlighted some limitations, including a slight decrease in overall accuracy when using login URLs due to their high similarity between legitimate and phishing URLs. This limitation highlights the importance of effective feature engineering to improve the discriminative capability between highly similar legitimate and phishing URLs.

In general, the comparison in Table 1 indicates that many current approaches have achieved high detection accuracy. However, three main research gaps can be identified: i) a limited focus on computational efficiency and practical deployment, such as reliance on third-party services or reporting relatively longer response times, ii) the evaluation of several studies is limited by the use of small-scale or outdated datasets

that primarily include traditional features, iii) limited transparency and reproducibility, particularly due to incomplete reporting of the extracted feature sets and detection time. Our methodology aims to address these limitations by balancing high speed and accuracy using the XGBoost model. This is achieved by using a recent dataset with advanced features that support the detection of sophisticated phishing attacks without relying on third-party services. Having a robust dataset and modern features can improve resilience to zero-day attacks.

3. METHOD

In this study, we reviewed recent literature that addresses the same problem of detecting phishing websites. We found that URL-based solutions are effective in detecting phishing websites [11]. Therefore, the primary objective of this research is to develop and evaluate traditional ML and neural network models to support URL-based phishing detection using engineered features while maintaining accuracy and efficiency. The following subsection provides detailed information about the sources and structure of the dataset. In this research, we used a set of features that are independent of third-party services and highly effective in detecting phishing attacks. In addition, we trained the models on a modern and realistic dataset that includes new and more complex phishing techniques. To evaluate the performance of the neural network and traditional ML models, we used Python to perform quantitative analysis and obtain statistical results for comparing the models' performance and identifying the most effective algorithm for detecting URL-based phishing attacks. The model development phase is summarized in Figure 1. This section discusses the research methodology, which comprises the following four steps: data collection; dataset pre-processing, including data cleaning, feature extraction, and feature selection; the use of neural network models and traditional ML models; and model evaluation criteria. Then, the final decision is made: legitimate or phishing.

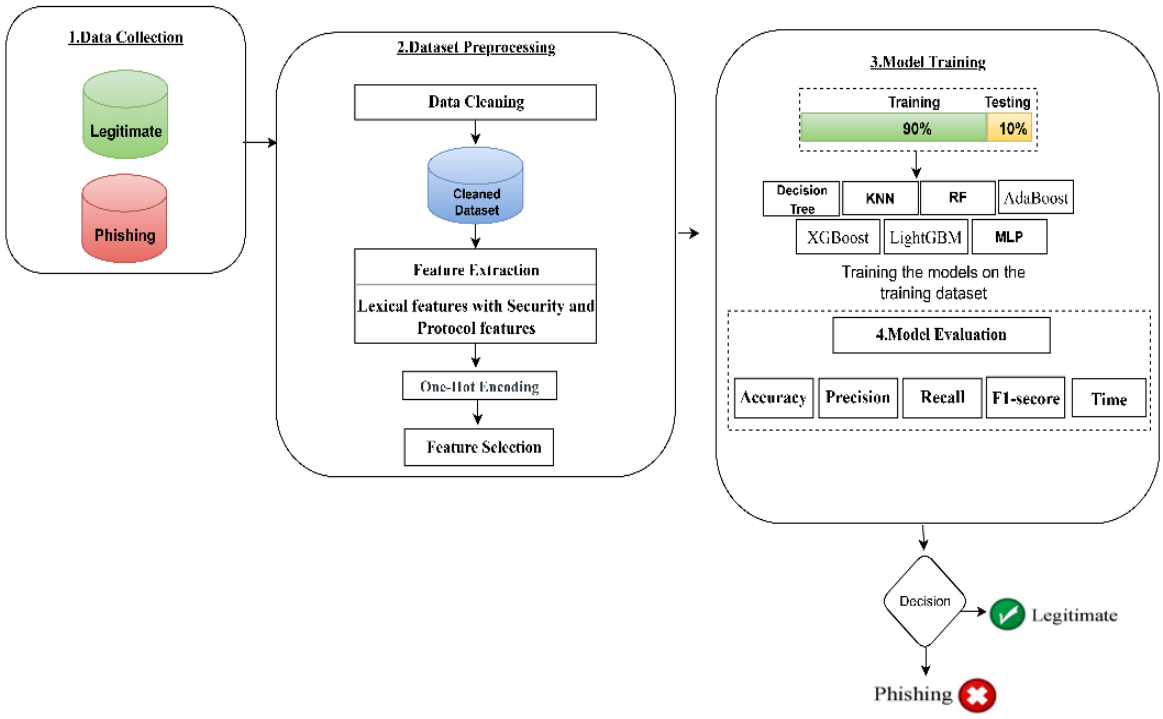


Figure 1. Overview of the proposed methodology (model development phase and output classification)

3.1. Data collection

The first step in developing ML models is to use a high-quality dataset, especially in phishing detection. We collected 72,832 phishing URLs from PhishTank, a website that updates phishing datasets hourly [18]. This dataset was accessed in May 2023. Legitimate URLs were obtained from two publicly available sources on Mendeley data [19] and Kaggle [20] platforms. Both datasets were accessed in June

2023. The Kaggle dataset contains well-known domains, while the Mendeley dataset includes less common domains and URLs representing different website levels, including login pages. This approach aimed to create a broad and unbiased dataset. To create a legitimate dataset, we employed a uniform random sampling technique on both sets, resulting in the selection of 73,637 URLs from Kaggle and 3,383 URLs from Mendeley data.

3.2. Dataset pre-processing

The pre-processing step is essential for ML models, as it prepares the data to be suitable for classification techniques. This step includes data cleaning, feature extraction, and feature selection. We explained these steps below.

3.2.1. Data cleaning

The phishing dataset required minimal pre-processing, as only the URL column was retained. In contrast, the legitimate dataset underwent additional pre-processing to extract only URLs labeled as legitimate before they were merged. Following that, we applied URL standardization to the legitimate dataset before feature extraction. For Kaggle data containing only hostnames (such as "www.youtube.com") without a protocol, we added an http:// prefix to enable URL requests and determine the actual protocol (HTTP or HTTPS) based on the server's response. This step was performed solely for feature extraction and did not alter the semantic structure of the URLs. Several preliminary experiments were conducted to determine an appropriate connection timeout. Based on these experiments, a timeout of 120 seconds was chosen to provide sufficient time for URL requests with varying server response times, especially those hosted on slower servers. Consequently, the legitimate dataset contained 76,993 URLs.

Finally, the phishing and legitimate datasets were merged. After removing duplicates and missing data, the resulting dataset contained 95,736 URLs, of which 39,367 were legitimate URLs (label 0), and 56,369 were phishing URLs (label 1). The dataset was then shuffled and split into 90% training and 10% testing using stratified sampling (random_state = 42). In our approach, we applied one-hot encoding, which converts categorical features into a numerical format that ML models can process to avoid the problem of ordinality.

In binary classification, dataset balance is particularly important when accuracy is used to assess model performance [21]. Although perfectly balanced datasets are uncommon, an imbalance ratio of approximately 10:1 is generally considered modestly imbalanced, whereas ratios exceeding 1000:1 are regarded as highly imbalanced [22]. For our dataset, after cleaning it by removing missing and duplicate values, the ratio of the majority (phishing) class to the minority (legitimate) class was approximately 1.4:1 (56,369/39,367). This ratio falls within the "marginally imbalanced" range, which means it can be considered safely balanced.

3.2.2. Feature extraction

The study relied on a wide range of literature related to detecting phishing websites to identify effective URL-based features. In addition, we conducted an in-depth analysis of the phishing and legitimacy dataset to identify additional URL-based discriminatory features. As a result, we identified new phishing techniques not yet documented in the literature, discussed in Section 6, Discussion with recommendations. We intentionally excluded features that rely on third-party services such as Google index and PageRank because they are not always accessible through free APIs, making them impractical for real-time detection. We also excluded DNS and domain age features, such as the creation date and expiration date of website information obtained from the WHOIS database. The domain age feature is more helpful in detecting long-term phishing websites, but it is ineffective for zero-day attacks. This decision aims to speed up the feature extraction process and help improve the detection of zero-day attacks. Additionally, it reduces potential privacy concerns by avoiding the transmission of user data to external servers for analysis.

Accordingly, we extracted 73 features that combine lexical features with security and protocol features. Lexical features refer to the structure of URL strings, number of characters, symbols such as (@, /, -,), as well as analysis of URL components such as path and query part [23]. Moreover, lexical features can detect phishing URLs generated by AI [24]. Because AI relies on lexical features to simulate legitimate URLs. Regarding security and protocols, these features help detect phishing URLs more sophisticatedly. For example, phishing URLs use internet protocol (IP) addresses in the domain field, employ non-standard port numbers, exploit decentralized protocols such as IPFS, and leverage free services like free hosting domain (FHD) services. Combining lexical features with security and protocol features helps expand the detection range and provides broader analysis of phishing URLs.

At this stage, we identified 73 URL-based features, presented in a structured table that includes their references and an explanation of their purpose and category: categorical, count, and numerical features (see

Table 1 in Supplementary 1). These extracted features reflect the complexity of detecting new and advanced URL-based phishing attacks.

3.2.3. Feature selection

After extracting 73 features, we applied Pearson's correlation coefficient [25] to identify highly correlated features and reduce redundancy. Accordingly, features with a correlation coefficient greater than 0.8 were excluded, while the more important features were retained. However, correlation analysis revealed that the "count_http" and "count_https" features, as well as the "SDT_words_dot" and "SDT_words_slash" features, were associated with "long_url". Removing or combining these features resulted in a lower F1-score, indicating that they provide complementary predictive information. Therefore, these features were retained. Following that, we used recursive feature elimination with cross-validation (RFECV) with XGBoost as the base estimator to identify the optimal subset of features [26]. Notably, the feature selection was performed exclusively on the training set to prevent data leakage, and the selected features were then applied to both the training and test sets. Features were recursively eliminated one at a time (step = 1) and evaluated using stratified 5-fold cross-validation with accuracy as the scoring metric. To quantitatively assess the contribution of the selected features, Gain-based feature importance scores were extracted from the trained XGBoost estimator and analyzed. Consequently, 40 features were selected as the optimal subset, as illustrated in Figure 2. The most important features are marked with an asterisk (*), while the deleted features are marked with a plus (+) in the novel table (Supplementary 1, Table 1). The 40 selected features are explained below, with '#F' denoting the feature number in Table 1.

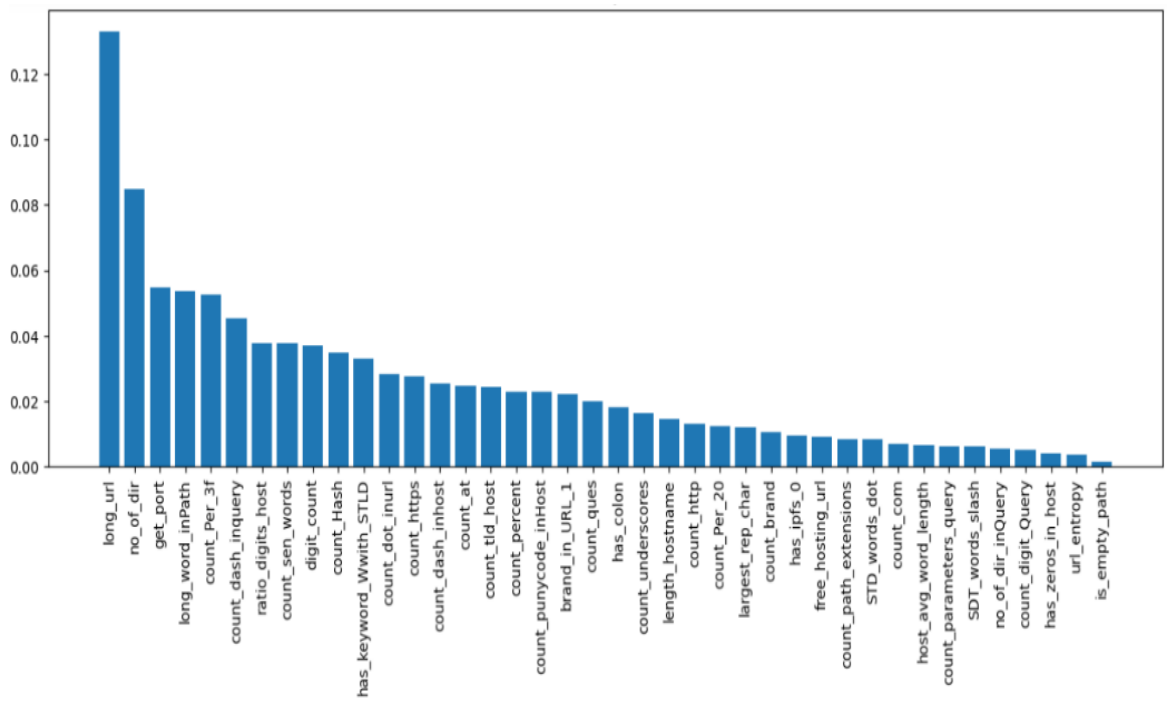


Figure 2. Gain-based feature importance obtained from the XGBoost model

Brand names and suspicious words: Phishing attackers frequently use well-known brand names and suspicious words in URLs. Feature 2F counts the frequency of brand names within URLs. As shown in Example 1 of a phishing URL from our dataset, the PayPal brand is used twice in different positions within the URL. They also use suspicious words like "signin" to entice users to visit the phishing page (signin.html). Therefore, Feature 8F counts the frequency of sensitive words in URLs. This revealed that the attackers use more than one sensitive word in a URL, while legitimate URLs use at most one. Based on our dataset, we observed that attackers use brand names at the domain level but with a fake top-level domain (TLD). Therefore, feature 55F detects the presence of brand names in a URL.

Also, feature 35F counts the number of times consecutive characters are repeated in URLs, such as 'paypal.com' to 'paypall.com'. Due to the inherent repetition of characters in URL keywords, we have

intentionally excluded protocol indicators such as HTTP, HTTPS, and WWW. This approach can support efficient detection of phishing attacks.

Top-level domain (TLD): Cybercriminals utilize TLDs to create URLs that appear legitimate. Netcraft [27] identified 50 TLDs associated with phishing attacks, for example, (.shop, .top, .cfd, and .cloud). Furthermore, they highlighted how attackers exploited new TLDs released by Google in March 2023 [28], notably the TLDs with names similar to file extensions, such as '.zip 'and '.mov'. These domains were subsequently associated with brand names and sensitive words to attract users [29]. Therefore, we developed feature 53F to detect sensitive words associated with suspicious TLDs.

Features 7F and 6F count the frequency of TLDs in hostnames and extensions (such as .html and .php) appearing in the path portion of a URL. Attackers use the TLD 'com' because it is frequently used in legitimate URLs. As shown in Example 1, a phishing URL from our dataset shows the frequency of the 'com' in the hostname and path portions. Therefore, Feature 3F tracks the frequency of the 'com' TLD appearing in URLs. Example 1: http://www.asaskofordham.com/paypal/www.paypal.com/signin.html.

Special characters: Phishing URLs often use symbols strategically to exploit browser behavior. Based on our analysis, phishing URLs excessively use the '@' symbol across multiple URL parts, as shown in Example 2 of a phishing URL from our dataset. As a result, feature 4F counts the number of times the '@' symbol appears in a URL.

Example 2: https://preventingoverdiagnosis.net/@@/GlobalSources/?email=abuse@chase.com.

Additionally, we examined the occurrences of symbols such as '-' (25F) in hostnames and (24F) in query, while '/' (34F), and '//' (64F) in the path, and '.' (23F), '%' (29F), '_' (22F), ':' (56F),'#' (15F), '?' (28F) across all URL parts. Furthermore, phishing URLs may employ multiple encoding techniques, such as single or double percent-encoding, to hide suspicious patterns or bypass mechanisms that rely on basic URL characteristics. In the phishing dataset, we noticed that hackers use the question mark (?) in an excessively encoded way, replacing it with '%3f' in URLs. Therefore, feature 9F counts encoded question marks in a URL. Our analysis revealed that attackers also use the non-standard directory symbol '/' within the query part, as shown in Example 3. Therefore, feature 32F counts the occurrences of the '/' symbol in the query part.

Example 3:

https://645173d69d7dda0e64b5dfe6.trk.mailchef.4dem.it/ttn.php?p=e60d44f049b0335f0f4256565b967944/804s/7cld/rs/rs/rs/rt/https://comp-assyst.com//orgbk-b/Client/connexion.html.

Query strings play a crucial role in transmitting data to the server within the URL. Therefore, feature 10F counts the parameters in the URLs' query parts. In addition, feature 26F counts a space '%20' within URLs. Based on our dataset, we found that attackers create complex URLs by using double-encoded spaces, adding the prefix '%25', which results in '%2520'. However, previous research has not widely documented this technique as a strong indicator of phishing detection. Our methodology does not determine a threshold for classifying URLs, as no specific, reliably indicated phishing behavior exists. To ensure high extraction accuracy, we excluded the colon in the protocol part, e.g., ' HTTP://' of the URL.

Punycode: The internationalized domain names (IDNs) technology enables users to register domain names in their local languages by adding the prefix 'xn--' at the beginning of the URL [30]. The researchers in [31], [32] extracted the Punycode feature, and once present, the URL is classified as phishing. Based on our dataset, we found many legitimate URLs that contain Punycode only once in the hostname, meaning that judging a URL based on the appearance of Punycode increases the rate of false positives. Conversely, attackers use multiple Punycodes (xn--) in a URL, as shown in Example 4 from our dataset, indicating an attempt to complicate the URL by encoding different parts of the hostname. As a result, in our methodology, we used feature 5F to count the number of occurrences of Punycode (xn--) within the hostname in URLs. It is worth noting that previous studies have not extracted this feature in this way. Furthermore, there is a lack of recent studies on extracting and accurately analyzing the Punycode feature to detect phishing attacks. Example 4: http://wtpfycin2-xn--ffgy3sod-xn----c1ac4bxc-xn----p1ai.translate.xn--c1ac4bxc.xn--p1ai.

Port number and protocols (HTTP and HTTPS): Security mechanisms monitor known ports, such as 80 for HTTP and 443 for HTTPS, to ensure network security. Feature 72F returns the port number in any URL part. Because the attackers exploit mixed content techniques to hide non-standard ports, such as 8081, as shown in Example 5 of a phishing URL from our dataset. Phishing URLs often contain multiple protocols in a single URL, unlike legitimate ones. As shown in Example 6, a phishing URL from our dataset, the attackers use HTTPS three times in a single URL. Therefore, features 30F and 31F measure the occurrences of HTTP and HTTPS protocols in URLs. To ensure that these features are extracted accurately, we used the complete protocol formats ('http://' and 'https://') to ensure the integrity of the result and avoid confusion.

Example 5:

https://www.google.com/url?q=http://www.customer-portal.info:8081?rid%3D92360ZX%26pcode%3DpZtNT&source=gmail-imap&ust=1664953377000000&usg=AOvVaw0vpjeHQBH1nml-ZmWqAQm2.

Example 6:

https://www.blogger.com/navbar.g?targetBlogID\x3d5373582281542081499\x26blogName\x3dcarrorosa

breas\x26publishMode\x3dPUBLISH_MODE_BLOGSPOT\x26navbarType\x3dBLUE\x26layoutType\x3d
CLASSIC\x26searchRoot\x3dhttps://carrorosabreas.blogspot.com/search\x26blogLocale\x3dpt_BR\x26v
\x3d2\x26homepageUrl\x3dhttps://carrorosabreas.blogspot.com/\x26vt\x3d2177333667468652914.

The average word length: Feature 65F calculates the average word length in the hostname part of URLs. This is a valuable feature for detecting phishing attacks. It helps detect anomalies associated with phishing websites. Standard deviation (STD): Measuring the standard deviation of variability indicates how far word lengths in URLs deviate from the average. In our research, we extracted the standard deviation in two ways. The first is through feature 66F, which measures the STD of word lengths separated by dots, such as domain and file names. The second feature is 67F, which measures the STD of word lengths separated by the slash '/' symbol, such as words in the path. As a result, both features revealed unusual numerical values and a large dispersion in phishing URLs compared to their legitimate counterparts.

URL entropy**:** The feature 70F measures the level of randomness in URL content. This feature is important because many phishing URLs contain obfuscated characters to deceive users. Digits in a URL: Feature 68F calculates the ratio of numbers to letters in a hostname, which helps detect phishing. Additionally, feature 36F measures the numeric content in URLs, as a high number of numbers in URLs indicates a phishing attempt. Based on our analysis of the phishing dataset, we created feature 33F to count the number of zeros in the hostname, while feature 38F counts the numbers in the query part.

URL length: Legitimate URLs rely on short hostnames to make them easy to use. In contrast, attackers use long hostnames to confuse their targets. Therefore, we created feature 69F, which calculates the length of the hostname to identify phishing addresses without setting a threshold. Because specifying a threshold may cause misclassification [33]. Also, using long words in the path part is a common technique in phishing attacks for obfuscation purposes. Accordingly, feature 71F measures the length of the longest word in the URL's path.

It is also important to measure the length of the entire URL. Based on our dataset analysis, we found that the longest phishing URL extends to 6097 characters. Since we rely on browser responses to legitimate URLs, the longest legitimate URL extends to 1276 characters. Therefore, we created feature 73F to calculate the length of URLs without setting a predefined classification threshold. Contrary to studies [12], [33], [34], and [35], which set different threshold values for URL lengths, the following values are used: 54, 200, 76, and 74 characters, respectively. Exceeding these values is considered phishing. These differences in criteria highlight that there is no uniform standard for evaluating URL lengths to determine whether they are phishing. Our approach helps distinguish between legitimate and phishing URLs based on their different lengths, which are classified according to the model's prediction.

IPFS: The IPFS protocol is used for decentralized file uploading, sharing, and access [36]. Phishing websites hosted on IPFS are similar to traditional phishing websites except that they use a hash value for the content instead of a domain name, making it difficult to trace the source or verify the publisher's identity. Our analysis of our dataset revealed that attackers use double encoding of IPFS-hosted webpage URLs to bypass detection systems, as illustrated in Example 7. Therefore, feature 47F detects the presence of a website hosted on IPFS and analyses its URL encoding level. Double-encoded URLs are classified as phishing, while unencoded ones are classified as suspicious; otherwise, they are classified as legitimate. The literature has not extensively analyzed this feature for detecting phishing attacks, a crucial aspect that warrants further research to mitigate phishing threats. Example 7:

https://bz907.infura-
ipfs.io/ipfs/QmWS4MuPNGmKFeVh4vGn4dESX3B7bzmPHwMbt1MhShG6pQ?gabrie%2500%251
Dv%2500%2510%2526%2523131%253B%2510Q%2526%2523135%253B%2520%2526%252314
6%253B%2526%2523139%

FHD: Free service providers provide their users with secure sockets layer (SSL) certificates and trusted TLDs such as 'com', which enables attackers to bypass detection mechanisms easily. We developed a feature 57F to detect phishing URLs hosted on FHDs and associated with sensitive words. In our research, we collected a list of 26 FHDs derived from the study [37] and our dataset. We carefully extracted FHDs to avoid false positives by including the TLDs that free hosting services provide to their users, indicating that it is indeed free hosting. For example, X10hosting offers free hosting services, such as elementfx.com, x10.mx, and x10.bz. Also, we detected 174 sensitive keywords in various languages related to free hosting, including French, Italian, English, Turkish, Indonesian, German, and others. Example 8, observed via our dataset analysis, shows a URL hiding the French phrase "payez votre facture", which means (pay your bill, in English). Furthermore, attackers use the names of government entities to trick users into revealing their credentials. Example 9, from our dataset, shows a URL impersonating the Saudi Ministry of Finance (mof.gov.sa) on free hosting 'firebaseapp.com'. Example 8: https://sites.google.com/view/pay-ez-vo-tre-facture.

Example 9: https://mof-gov.firebaseapp.com/

3.3. Applied machine learning (ML) algorithms

This section reviews the models adopted in this study. As shown in Figure 1, we evaluated seven ML classifiers. The tested algorithms included standalone classifiers, such as a decision tree [38] and KNN [39]. Additionally, we evaluated several learning methods, including ensemble learning, such as RF [40], AdaBoost [41], XGBoost [42], and LightGBM [43]. Ensemble learning techniques often outperform individual ML models by combining predictions from multiple models, thus addressing the limitations of single ML models and leveraging their strengths to improve classification performance [44]. We also trained an MLP [45] consisting of an input layer, one or more hidden layers, and an output layer. Additionally, it relies on non-linear functions, such as ReLU and Sigmoid, to process data and extract patterns. In our experiments, the models exhibit varying levels of prediction accuracy. Below, we explain the evaluation metrics.

3.3.1. Evaluation parameters

After training the models, performance metrics were applied to determine the best-performing model. The main performance metrics include accuracy, precision, recall, F1-score, and detection time. The evaluation metrics are based on a confusion matrix, which summarizes correctly and incorrectly predicted cases. The confusion matrix has four outcomes:

- True positive (TP): The total number of phishing URLs correctly classified as phishing.
- True negative (TN): The total number of legitimate URLs correctly classified as legitimate.
- False positive (FP): The total number of legitimate URLs incorrectly classified as phishing.
- False negative (FN): The total number of phishing URLs incorrectly classified as legitimate.

From these outcomes, performance metrics can be calculated to determine the best-performing model. The following are the main performance metrics:

Accuracy: Ratio of correct predictions to the total number of predictions, as in (1)

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FN+FP}} \tag{1}$$

Precision: Ratio of correctly predicted positive observations (URLs detected as phishing) to the total predicted positives (phishing URLs), as in (2)

$$\text{Precision} = \frac{TP}{\text{TP+FP}} \tag{2}$$

Recall: Ratio of correctly predicted phishing URLs to all actual phishing URLs, as in (3)

$$\text{Recall} = \frac{TP}{\text{TP+FN}} \tag{3}$$

F1-score: Harmonic mean of precision and recall balancing both recall and precision, as in (4)

$$\text{F1} = 2 * \frac{\text{Precision*Recall}}{\text{Precision+Recall}} \tag{4}$$

Detection time: The time required to classify a phishing URL is measured in milliseconds or seconds, indicating model efficiency. Receiver operating characteristic area under the curve (ROC-AUC): To evaluate the model's ability to distinguish legitimate URLs from phishing URLs across different classification thresholds.

4. EXPERIMENT SETUP AND RESULTS

This section presents the experimental setup and performance results of all evaluated models, followed by an ablation study to evaluate the effect of feature selection on the proposed XGBoost model.

4.1. Experimental setup

To optimize model performance, we used GridSearchCV with 3-fold cross-validation [46] on the training set for the traditional ML classifiers to identify optimal parameters. The optimal hyperparameters for each model were selected based on the highest cross-validation accuracy. Notably, we executed several experiments with the KNN model, where its performance depends on the number of nearest neighbors (k). Using odd numbers for (k) is helpful for binary classification datasets, as it prevents tie votes between neighbors and ensures that the majority decision is clearly defined [47]. Therefore, we tested k = 1, 3, and 5

to determine the optimal configuration. The experiments with KNN are summarized in supplementary 2, Table 2. Additionally, initial experiments indicated that tuning AdaBoost did not result in significant performance improvements; therefore, the default configuration was retained. The configuration adopted for each model is summarized in Table 2. To ensure reproducibility, a fixed random seed (random_state = 42) was applied across all applicable models.

Table 2. Hyperparameter settings for the traditional ML classifiers and the MLP model on 40 features

Model	Parameters	Value
Traditional ML		
XGBoost	learning_rate	0.3
	max_depth	14
	n_estimators	45
RF	max_depth	19
	n_estimators	33
decision tree	max_depth	10
LightGBM	max_depth	5
KNN	k	5
AdaBoost	Default	
MLP	Neural network model	
	One hidden layer	64 neurons
	Optimizer	Adam
	Learning rate	0.001
	batch size	64
	Number of epochs	36
	Output layer	Sigmoid

4.2. Results of the XGBoost model

Initial results of the XGBoost model showed strong performance across all evaluation metrics, with an accuracy of 98.35% on the training set and 98% on the test set, using the model's default settings. After tuning the hyperparameters, the prediction performance remained almost identical, with a slight decrease in computation time. The XGBoost model recorded the following results on the test set: 98% accuracy, 98.3% precision, 98% recall, and 98% F1-score, with a computation time of 0.0162 seconds on the test set and an average time of 1.6977×10^{-6} seconds per URL, as shown in Table 3. In addition, we calculated the confusion matrix for the model, as shown in Figure 3. The model correctly identified 5,510 phishing websites (TP) and 3,842 legitimate websites (TN), resulting in 95 FP and 127 FN, with corresponding false positive and false negative rates of 2.41% and 2.25%, respectively. To further evaluate the learning behavior of the XGBoost model, we generated a learning curve using stratified 5-fold cross-validation, as illustrated in Figure 4. Its implications are discussed in the analysis section.

Table 3. Default and optimized performance of the XGBoost model using 40 features

Model	Performance metrics				
	Accuracy	Precision	Recall	F1-Score	Time (seconds)
XGBoost (Default)	Train: 98.35% Test: 98%	Train: 99% Test: 98.46%	Train: 98.18% Test: 98%	Train: 99% Test: 98%	Train: 0.08275 Test: 0.0205, Average time per URL: 2.1512×10^{-6}
XGBoost (n_estimators=45, learning_rate=0.3, max_depth=14)	Train: 99.23% Test: 98%	Train: 99.78% Test: 98.3%	Train: 99% Test: 98%	Train: 99.34% Test: 98%	Train: 0.0907 Test: 0.0162, Average time per URL: 1.6977×10^{-6}

4.3. Ablation study

We conducted an ablation study to evaluate the impact of feature selection on the proposed XGBoost model by comparing the performance obtained using all 73 extracted features and the selected 40 features, described in Section 3.2.3, Feature selection. As shown in Table 4, the 40 selected features achieved comparable classification performance with reduced prediction time.

4.4. Results of the five traditional ML models: RF, decision tree, LightGBM, KNN, and AdaBoost

This section presents a comparative evaluation of the performance of traditional ML models on the experimental dataset using 40 features. As Table 5 shows, other traditional ML classifiers (RF, decision tree, LightGBM, KNN, and AdaBoost) performed competitively compared to the XGBoost model. The accuracy

values of these models on the test set ranged between 92% and 97%, and the F1-score ranged between 93% and 98%. The RF model achieved the highest performance within this group, while the AdaBoost model showed the lowest performance across all performance metrics (accuracy, precision, recall, and F1-score). Regarding computation time, significant differences were observed between the models. However, the KNN (k = 5) model was the slowest compared to all the traditional ML models, recording an average computation time of 1.6446×10^{-4} seconds per URL on the test set. In contrast, the decision tree model achieved a processing speed with an average computation time of 5.3431×10^{-7} seconds per URL on the test set, thus outperforming the other traditional ML classifiers. In addition, the ROC-AUC was calculated on the test set for all evaluated models to provide a more comprehensive assessment of their classification performance. The comparative results are presented in Table 5.

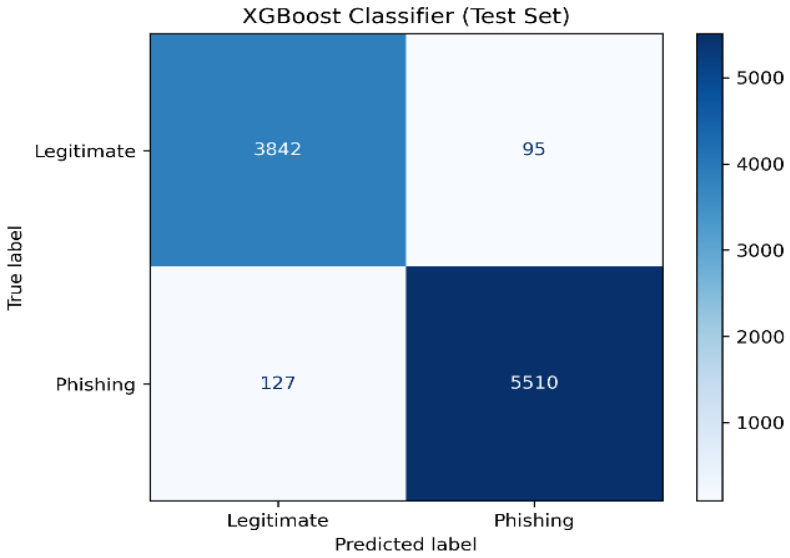


Figure 3. Confusion matrix of the XGBoost model

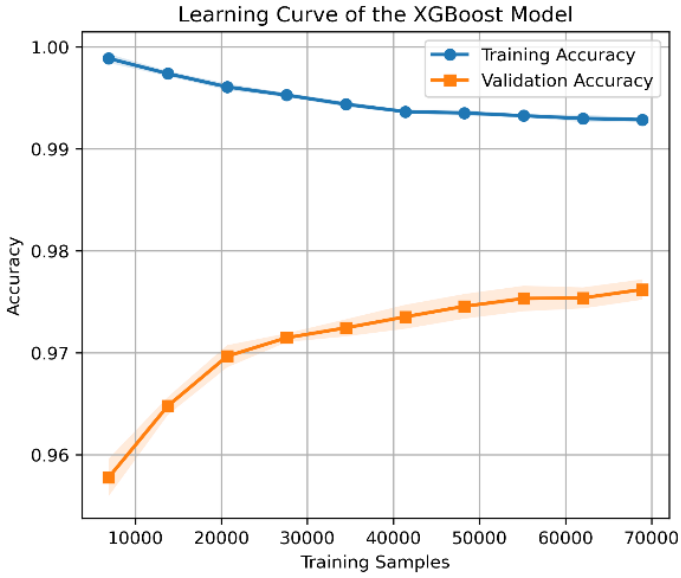


Figure 4. Learning curve of the XGBoost model

Table 4. Ablation study comparing the performance of the proposed XGBoost model using the original 73 features and the selected 40 features

	Number of features	Performance metrics				
		Accuracy	Precision	Recall	F1-Score	Time (seconds)
XGBoost model	73	Train: 99.24% Test: 98%	Train: 99.8% Test: 98.24%	Train: 99% Test: 98%	Train: 99.35% Test: 98.1%	Train: 0.1638 Test: 0.0392, Average time per URL: 4.0365×10^{-6}
	40	Train: 99.23% Test: 98%	Train: 99.78% Test: 98.3%	Train: 99% Test: 98%	Train: 99.34% Test: 98%	Train: 0.0907 Test: 0.0162, Average time per URL: 1.6977×10^{-6}

4.5. Results of the MLP model

We conducted several experiments on the MLP model to achieve strong model performance. Using the hyperparameters shown in Table 2, the model achieved the following results on the test set: an accuracy of 96.22%, a precision of 98%, a recall of 96.24%, and an F1-score of 97%, with a computation time of 5.0572 seconds and an average time of 5.281×10^{-4} seconds per URL, as illustrated in Table 5. However, the MLP model requires fine-tuning to prevent overfitting and find a balance between the number of hidden layers and the number of neurons in each layer. In our case, a single layer with 64 neurons was optimal compared to one with 1000 neurons or two hidden layers, the first with 64 neurons and the second with 32 neurons.

Table 5. Results of all traditional ML models and the MLP model trained on 40 URL-based features

Performance metrics	Traditional ML models					Neural network model	
	XGBoost	RF	Decision tree	Decision tree	KNN (k=5)	AdaBoost	MLP
Accuracy	Train: 99.23% Test: 98%	Train: 99% Test: 97%	Train: 95.33% Test: 94.39%	Train: 96.2% Test: 96%	Train: 96% Test: 95%	Train: 92% Test: 92%	Train: 97% Test: 96.22%
Precision	Train: 99.78% Test: 98.3%	Train: 99.62% Test: 98%	Train: 96.48% Test: 96%	Train: 97.18% Test: 97%	Train: 97.3% Test: 97%	Train: 94% Test: 94%	Train: 98% Test: 98%
Recall	Train: 99% Test: 98%	Train: 99% Test: 97.32%	Train: 96% Test: 95%	Train: 96.4% Test: 96.15%	Train: 96% Test: 95%	Train: 92.43% Test: 92.31%	Train: 97% Test: 96.24%
F1-score	Train: 99.34% Test: 98%	Train: 99.1% Test: 98%	Train: 96% Test: 95.2%	Train: 97% Test: 97%	Train: 97% Test: 96%	Train: 93.13% Test: 93%	Train: 97.32% Test: 97%
ROC-AUC	99.48%	99.25%	97.48%	98.96%	97.87%	97.16%	99.12%
Time (seconds)	Train: 0.0907 Test 0.0162, Average time per URL: 1.6977×10^{-6}	Train 0.4557 Test: 0.0817, Average per URL: 8.5421×10^{-6}	Train: 0.0188 Test: 0.0051, Average per URL 5.3431×10^{-7}	Train: 0.1803 Test: 0.0411, Average per URL: 4.2929×10^{-6}	Train 29.5862 Test 1.5745, Average per URL: 1.6446×10^{-4}	Train 0.7803 Test 0.1057, Average per URL: 1.1043×10^{-5}	Train 110.4328 Test 5.0572, Average per URL: 5.281×10^{-4}

5. RESULTS ANALYSIS AND SYSTEM EVALUATION

Here, we analyze the performance results of the models summarized in Table 5. Then, we evaluate the response time of the proposed model to measure the speed of the model in detecting phishing URLs.

5.1. Results analysis

Based on Table 5, the results show that the decision tree model achieved the lowest computation time for classification. In contrast, the MLP model achieved the highest computation time compared to traditional ML models. However, the XGBoost model achieved the best overall performance by providing an effective balance between classification accuracy and computational efficiency. It achieved a test accuracy of 98% while maintaining a low computation time of 1.6977×10^{-6} seconds per URL. Furthermore, the learning curve shown in Figure 4 demonstrates a small gap between training and testing accuracy, indicating good generalization with minimal risks of overfitting. Therefore, the confusion matrix is consistent with the high testing accuracy and F1-score achieved by the proposed model, showing only a limited number of false-positive and false-negative predictions. The ROC analysis supports these results, with the proposed XGBoost

achieving the highest ROC-AUC value among all evaluated models (see Table 5), confirming its strong ability to distinguish between phishing and legitimate URLs. In addition, the ablation study shown in Table 4 demonstrates that the selected 40 features retained the most discriminative information while improving computational efficiency. We summarized the performance of the models on the test set using 40 URL-based features in Figure 5, while Figure 6 compares the computation time of these models. We excluded the KNN and MLP models from Figure 6 because they have the longest computation time. As a result, the XGBoost model is suitable for handling large datasets with numerous features, enabling it to classify phishing attacks accurately and efficiently.

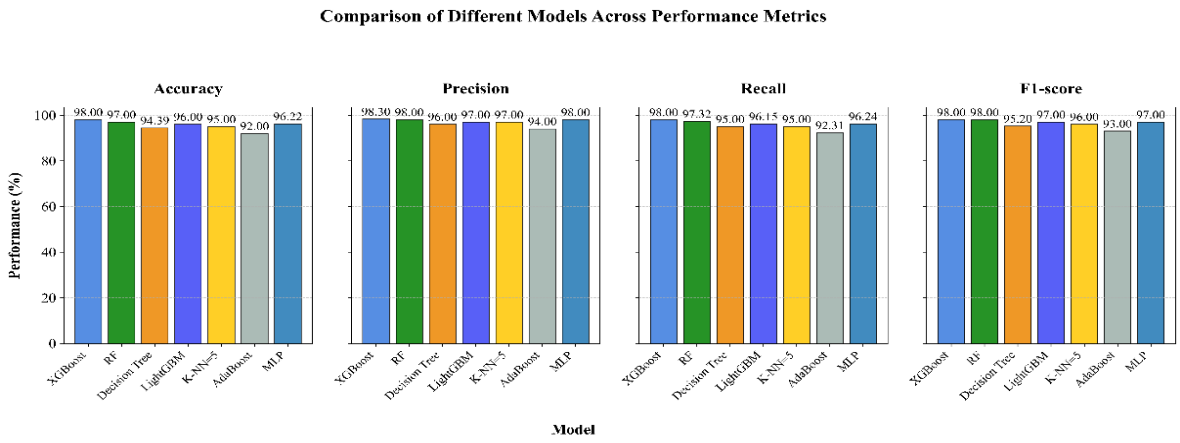


Figure 5. Performance of traditional ML models and MLP model trained on 40 features, with results calculated from the test set

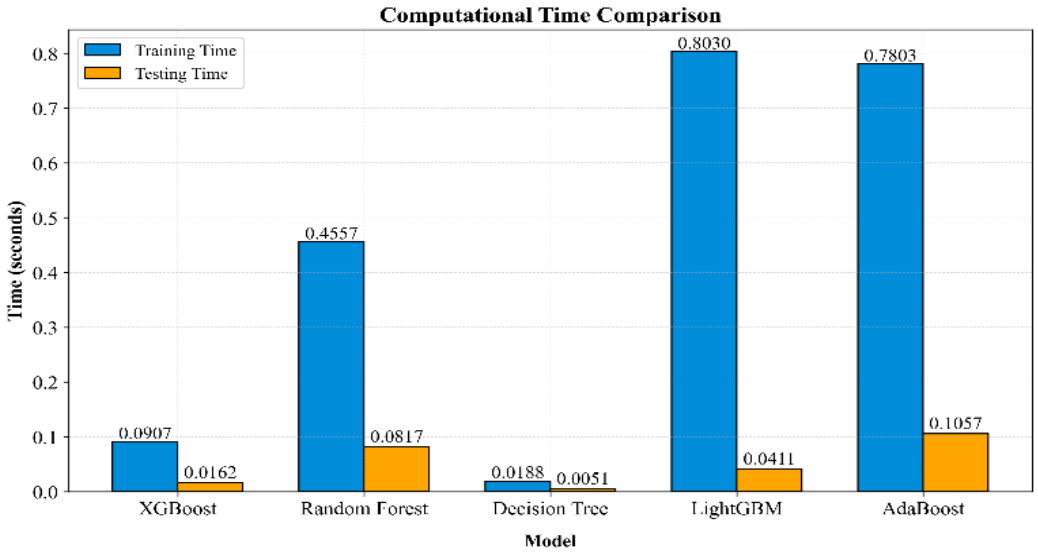


Figure 6. Comparison of training and testing computation times for ML models on 40 features

5.2. System evaluation: Response time

After analyzing the results and demonstrating the XGBoost model's classification power, we evaluated it based on its response speed in detecting phishing URLs in an online setting. To conduct this testing, we randomly collected 200 recent, real-world legitimate URLs from search engines such as Google and active phishing URLs from PhishTank. This experiment was performed on a device consisting of an Intel Core i7-8565U processor running at up to 1.99 GHz and 16 GB of RAM on Windows 11. All computations were performed using Python 3.10.9 in the Jupyter Notebook environment. Initially, we must explain the structure of the proposed system, which consists of three main stages, as illustrated in Figure 7. The first stage involves extracting 40 URL-based features (see Figure 2). In the second stage, the extracted features are

analyzed using the XGBoost model. The third stage presents the prediction result based on whether the provided URL is legitimate or a phishing site, as shown in Figure 8. The XGBoost model recorded an average response time of 0.026225 seconds per URL. Therefore, our model meets the low response time, providing protection for users before they load and interact with the page.

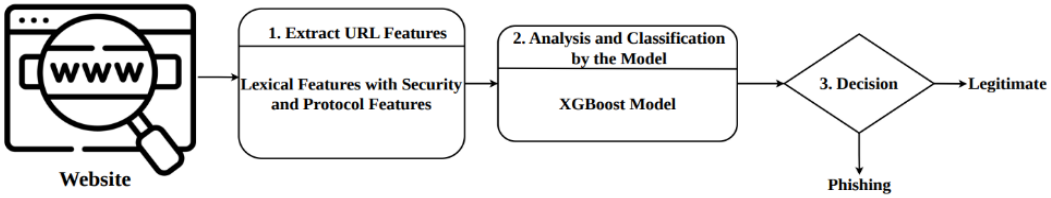


Figure 7. System architecture of the proposed model

```
URL: https://www.nike.com
Prediction: Legitimate
Response Time: 0.027925 seconds

URL: https://bafybeihdtrresaqam2v4pgxeligzs2jug3t2ykl5shyydcmg5vjydbqwpe.ipfs.cf-ipfs.com/main.html
Prediction: Phishing
Response Time: 0.026928 seconds

URL: https://www.office.com
Prediction: Legitimate
Response Time: 0.024933 seconds

URL: https://engines-komatsu.com
Prediction: Phishing
Response Time: 0.025931 seconds

URL: https://wallet-matamesk.gitbook.io/web
Prediction: Phishing
Response Time: 0.025930 seconds

URL: https://www.cnn.com
Prediction: Legitimate
Response Time: 0.025931 seconds

URL: https://kirakeenilogin.webflow.io/
Prediction: Phishing
Response Time: 0.024933 seconds

URL: https://www.apple.com
Prediction: Legitimate
Response Time: 0.041891 seconds

URL: https://www.google.com
Prediction: Legitimate
Response Time: 0.026924 seconds
```

Figure 8. Example of real-world URLs to test the efficiency of the XGBoost model

5.3. Comparison with existing approaches

This section compares the proposed methodology with relevant studies in phishing website detection using URL-based features. The comparison can be divided into two main types: a methodological comparison, which includes all reviewed studies, and a performance comparison limited to studies whose models were re-implemented and evaluated on the same dataset used in this work. This comparative analysis serves two main objectives: first, to evaluate the classification performance of the proposed URL-based feature sets against existing state-of-the-art approaches. Second, to demonstrate the advantage of the proposed methodology in terms of detection performance. The methodological comparison is based on several criteria, including the number of features, the model used, the dataset size, and reliance on third-party services. The comparison also includes the analysis of three advanced URL-based feature types: IPFS analysis, Punycode analysis, and URL component encoding analysis, which have received limited attention in previous studies. The results of the comparison are summarized in Table 6.

On the other hand, performance comparison criteria focus on classification performance metrics (accuracy, precision, recall, and F1-score). The study [12] relies on a limited dataset comprising 15 traditional URL-based features, some of which depend on third-party services. When the LightGBM model was applied to our dataset, which contains 95,736 URLs and uses 40 features, including traditional and advanced features independent of third-party services, the results showed an accuracy of 96.23%, exceeding the reported accuracy of 86%, as shown in Table 7. Similarly, studies [11] and [7] rely on a limited number of traditional features, 9 and 20 features, respectively. When their configurations of RF models were applied to our dataset, the model's accuracy decreased slightly from 99.57% to 97.28% and from 99% to 97.20%, respectively. This is likely due to the greater complexity and diversity of our dataset compared to those used in those studies.

As an additional experiment, the response time of the online evaluation was measured across different feature set sizes using XGBoost. Using the 9 features proposed in [11], our hyperparameter configuration of XGBoost achieved an average response time of 0.009807 seconds per URL and 96% on the test set. Expanding to the selected 40 features increased the response time to 0.026225 seconds per URL. The proposed model achieved an accuracy of 98% on the test set. This accuracy, along with the lower online response time (0.026225 seconds per URL), indicates that the proposed feature set provides a better balance between accuracy and efficiency.

Table 6. Comparison of the proposed approach with previous studies

Ref.	Model	Dataset size	Number of features	Advanced URL-based features			Third-party dependence
				IPFS analysis	Punycode analysis	URL component encoding analysis	
[11], 2021	RF	9,964 phishing URLs and 10,000 legitimate URLs.	9	×	×	×	×
[7], 2021	RF	Legitimate: 35,378, Phishing: 9,965, Malware: 11,566, and Spam: 11,942.	20	×	×	×	×
[16], 2021	RNN-GRU	60,000 legitimate URLs and 60,000 phishing URLs.	100	×	×	×	×
[12], 2022	LightGBM	1,500 phishing URLs and 1,500 legitimate URLs.	15	×	×	×	✓
[17], 2022	Logistic Regression	30,000 phishing URLs and 30,000 legitimate URLs.	38	×	✓	×	×
[9], 2024	RF	36,173 phishing URLs and 51,316 legitimate URLs.	44	Not mentioned	Not mentioned	Not mentioned	×
[14], 2023	LSD model with Canopy feature selection	5,000 phishing URLs and 6,000 legitimate URLs.	33	Not mentioned	Not mentioned	Not mentioned	Not mentioned
Our study	XGBoost	56,369 phishing URLs and 39,367 legitimate URLs.	40	✓	✓	✓	×

Table 7. Performance comparison of previous studies and the proposed system, including both reported results and results obtained by applying baseline models to our dataset

Ref.	Model	Method	Results reported in the original paper (test set)				Results on our dataset (test set)			
			Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
[11], 2021	RF	URL-based	99.57%	99.70%	99.46%	99.58%	97.28%	98%	98%	98%
[7], 2021	RF	URL-based	99%	99%	99%	100%	97.20%	97.48%	98%	98%
[12], 2022	LightGBM	URL-based	86%	Not mentioned	Not mentioned	Not mentioned	96.23%	97.16%	96.42%	97%
Our study	XGBoost	URL-based	-	-	-	-	98%	98.3%	98%	98%

6. DISCUSSION WITH RECOMMENDATIONS

Phishing attacks are a constantly evolving field. Therefore, this study aimed to identify a wide range of effective features for detecting phishing attacks. As a result, the study identified 73 URL-based features independent of third-party services. The findings indicate that effective phishing URL detection depends

more on feature quality than feature quantity. Thus, selecting the most informative features (40 URL-based features) enabled the model to maintain predictive performance while improving computational efficiency. The superior performance of the proposed XGBoost model can be attributed to its gradient boosting framework, which effectively captures complex non-linear relationships among URL features, while iteratively correcting the errors of previous trees. This also enables the proposed model to achieve higher classification performance than the other evaluated ML models. Although the decision tree achieved the lowest processing time, its classification performance was inferior to that of the proposed model. In contrast, the XGBoost model offered a strong balance between detection accuracy and processing speed. Accordingly, the study achieved its main objective of maximizing the use of powerful features while maintaining fast detection time, which can support a browser-based detection system with acceptable accuracy.

Furthermore, this research is consistent with the recommendations of the study [48] in constructing a lightweight detection model that balances accuracy and speed to meet the requirements of effective detection. Likewise, the generated dataset aligns with the requirements in [8], which emphasized the need for models trained on large datasets and equipped with advanced features for detecting complex phishing URLs. Accordingly, our dataset included advanced features that have received limited attention in prior studies. Advanced features include identifying Punycode in the host (5F), detecting URL-encoded symbols, such as question marks (?) encoded as '%3F' (9F), capturing the presence of backslash encoding (43F), and double encoding techniques in the URL (46F). In addition to analyzing IPFS (47F), where "#F" refers to the property number in Table 1 (Supplementary 1). As shown in Table 8, the majority of these advanced URL-based features were identified by the XGBoost model as among the most significant contributors to detection, with the exception of features 43F and 46F, which were ranked among the least important. It should be noted that we have improved the extraction method of the FHD feature with the aim of reducing potential false positives, as described in Section 3.2.3, Feature Selection. This is to ensure a more accurate distinction between phishing URLs and legitimate URLs.

We recommend that researchers focus on modern phishing techniques, such as attackers' use of double encoding and the sophisticated use of IPFS. They also use directory traversal techniques, part of cross-site scripting (XSS) attacks, for phishing purposes, exploiting the vulnerabilities of anti-phishing mechanisms. We also encourage researchers to refine the decision tree model by tuning its hyperparameters to enhance its performance and capitalize on its rapid detection capabilities. Despite the effectiveness of the proposed model, the study is not without limitations. One of the main limitations is that relying on URL features alone limits the detection of content-based phishing attacks.

Table 8. New features explored in our research

Importance in detecting phishing (XGBoost)	Feature (s)
Yes	5F, 9F, 47F
No	43F, 46F

7. CONCLUSION

This study evaluated the effectiveness of the MLP and traditional ML models in detecting URL-based phishing attacks using 40 key features. The XGBoost model outperformed all tested models, achieving a balanced performance between accuracy and speed. This work reinforces ongoing efforts in phishing detection research by providing a recent and unbiased dataset that includes effective features that are independent of third-party services. In addition, the study presents a novel table summarizing features identified from recent studies, carefully extracted based on the analysis of our dataset, which will be useful to researchers conducting studies in this field. We believe that this proposed model could be integrated into the field of cybersecurity to protect against phishing attacks. While the proposed approach is effective, it relies solely on URL features. Consequently, it may lead to miss sophisticated phishing attacks that require content analysis, such as JavaScript and HTML. In future work, we plan to enhance our system by analyzing content-based features to create a more robust and comprehensive detection system that minimizes false negative predictions while conserving resources by reducing computational demands. We will also explore adaptive DL algorithms to ensure effectiveness against emerging threats and zero-day attacks. Additionally, we plan to implement the proposed system as a browser plugin that provides real-time phishing alerts.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Jawaher Alharbi	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	
Manal Bayousef	✓	✓		✓	✓	✓				✓	✓	✓	✓	
Hind Almisbahi	✓		✓	✓	✓	✓				✓	✓	✓	✓	

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

The authors state no conflict of interest.

INFORMED CONSENT

The authors received no financial support for the research, authorship, and/or publication of this article.

DATA AVAILABILITY

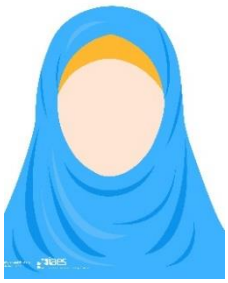
The data that support the findings of this study are available at: https://www.kaggle.com/datasets/jawaheraalharbi/url-based-phishing-detection-without-third-party

REFERENCES

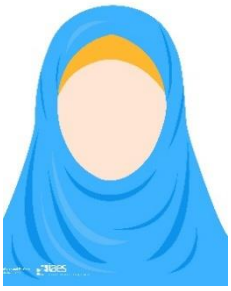
[1] N. Q. Do, A. Selamat, O. Krejcar, E. Herrera-Viedma, and H. Fujita, "Deep learning for phishing detection: Taxonomy, current challenges and future directions," *IEEE Access*, vol. 10, pp. 36429–36463, 2022, doi: 10.1109/ACCESS.2022.3151903.
[2] CloudSEK, "Phishing campaign targeting the saudi government service portal, absher." Accessed: Nov. 26, 2022. [Online]. Available: https://www.cloudsek.com/threatintelligence/phishing-campaign-targeting-the-saudi-government-service-portal-absher
[3] Anti-Phishing Working Group (APWG), "Phishing activity trends report: 1st quarter 2023." Accessed: Dec. 05, 2023. [Online]. Available: https://docs.apwg.org/reports/apwg_trends_report_q1_2023.pdf
[4] SlashNext, "Prepare for 2025: 2024 Phishing intelligence report." Accessed: Dec. 25, 2024. [Online]. Available: https://slashnext.com/press-release/2024-eoy-phishing-intelligence-report
[5] Esentire, "2023 official cybercrime report," 2023. Accessed: Mar. 20, 2023. [Online]. Available: https://www.esentire.com/resources/library/2023-official-cybercrime-report
[6] P. Yang, G. Zhao, and P. Zeng, "Phishing website detection based on multidimensional features driven by deep learning," *IEEE Access*, vol. 7, pp. 15196–15209, 2019, doi: 10.1109/ACCESS.2019.2892066.
[7] A. S. Raja, R. Vinodini, and A. Kavitha, "Lexical features based malicious URL detection using machine learning techniques," *Materials Today: Proceedings*, vol. 47, pp. 163–166, 2021, doi: 10.1016/j.matpr.2021.04.041.
[8] S. Asiri, Y. Xiao, S. Alzahrani, S. Li, and T. Li, "A survey of intelligent detection designs of HTML URL phishing attacks," *IEEE Access*, vol. 11, pp. 6421–6443, 2023, doi: 10.1109/ACCESS.2023.3237798.
[9] E. Kocyigit, M. Korkmaz, O. K. Sahingoz, and B. Diri, "Enhanced feature selection using genetic algorithm for machine-learning-based phishing URL detection," *Applied Sciences (Switzerland)*, vol. 14, no. 14, p. 6081, Jul. 2024, doi: 10.3390/app14146081.
[10] M. Korkmaz, E. Kocyigit, O. K. Sahingoz, and B. Diri, "A hybrid phishing detection system using deep learning-based URL and content analysis," *Elektronika ir Elektrotechnika*, vol. 28, no. 5, pp. 80–89, Oct. 2022, doi: 10.5755/j02.eie.31197.
[11] B. B. Gupta, K. Yadav, I. Razzak, K. Psannis, A. Castiglione, and X. Chang, "A novel approach for phishing URLs detection using lexical-based machine learning in a real-time environment," *Computer Communications*, vol. 175, pp. 47–57, Jul. 2021, doi: 10.1016/j.comcom.2021.04.023.
[12] S. H. Ahammad *et al.*, "Phishing URL detection using machine learning methods," *Advances in Engineering Software*, vol. 173, p. 103288, Nov. 2022, doi: 10.1016/j.advengsoft.2022.103288.
[13] M. G. Hr, A. Mv, S. G. Prasad, and S. Vinay, "Development of anti-phishing browser based on random forest and rule of extraction framework," *Cybersecurity*, vol. 3, no. 1, p. 20, Dec. 2020, doi: 10.1186/s42400-020-00059-1.
[14] A. Karim, M. Shahroz, K. Mustofa, S. B. Belhaouari, and S. R. K. Joga, "Phishing detection system through hybrid machine learning based on URL," *IEEE Access*, vol. 11, pp. 36805–36822, 2023, doi: 10.1109/ACCESS.2023.3252366.
[15] M. Al-Sarem *et al.*, "An optimized stacking ensemble model for phishing websites detection," *Electronics (Switzerland)*, vol. 10, no. 11, p. 1285, May 2021, doi: 10.3390/electronics10111285.
[16] L. Tang and Q. H. Mahmoud, "A deep learning-based framework for phishing website detection," *IEEE Access*, vol. 10, pp. 1509–1521, 2022, doi: 10.1109/ACCESS.2021.3137636.
[17] M. Sánchez-Paniagua, E. Fidalgo, E. Alegre, and R. Alaiz-Rodríguez, "Phishing websites detection using a novel multipurpose dataset and web technologies features," *Expert Systems with Applications*, vol. 207, p. 118010, Nov. 2022, doi: 10.1016/j.eswa.2022.118010.
[18] PhishTank, "Join the fight against phishing." Accessed: May 08, 2023. [Online]. Available: https://phishtank.org
[19] S. Ariyadasa, S. Fernando, and S. Fernando, "Phishing websites dataset," 2021. doi: 10.17632/n96ncsr5g4.1.
[20] Aks, "45+ M Domains with size & Language - Common Crawl," Kaggle. Accessed: Jul. 01, 2025. [Online]. Available: https://www.kaggle.com/datasets/bpmtips/46-million-domain-names-with-size-common-crawl

[21] K. Adane and B. Beyene, "Machine learning and deep learning based phishing websites detection: The current gaps and next directions," *Review of Computer Engineering Research*, vol. 9, no. 1, pp. 13–29, May 2022, doi: 10.18488/76.v9i1.2983.
[22] G. M. Weiss, "Foundations of imbalanced learning," in *Imbalanced Learning: Foundations, Algorithms, and Applications*, Wiley, 2013, pp. 13–41. doi: 10.1002/9781118646106.ch2.
[23] R. Zieni, L. Massari, and M. C. Calzarossa, "Phishing or not phishing? a survey on the detection of phishing websites," *IEEE Access*, vol. 11, pp. 18499–18519, 2023, doi: 10.1109/ACCESS.2023.3247135.
[24] M. Sameen, K. Han, and S. O. Hwang, "PhishHaven - An efficient real-time AI phishing URLs detection system," *IEEE Access*, vol. 8, pp. 83425–83443, 2020, doi: 10.1109/ACCESS.2020.2991403.
[25] K. Pearson, "VII. Note on regression and inheritance in the case of two parents," *Proceedings of the Royal Society of London*, vol. 58, no. 347–352, pp. 240–242, Dec. 1895, doi: 10.1098/rspl.1895.0041.
[26] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, no. 1–3, pp. 389–422, 2002, doi: 10.1023/A:1012487302797.
[27] Netcraft, "Cybercrime on top level domains." Accessed: Jul. 05, 2023. [Online]. Available: https://trends.netcraft.com/cybercrime/tlds
[28] Google Registry, "Launch details for eight new TLDs." Accessed: Jul. 01, 2023. [Online]. Available: https://www.registry.google/announcements/launch-details-for-eight-new-tlds
[29] Netcraft, "Phishing attacks already using the .zip TLD." Accessed: Aug. 23, 2023. [Online]. Available: https://www.netcraft.com/blog/phishing-attacks-already-using-the-zip-tld
[30] P. Faltstrom, P. Hoffman, and A. Costello, "Internationalizing domain names in applications (IDNA)," Mar. 2003. doi: 10.17487/rfc3490.
[31] M. Sanchez-Paniagua, E. F. Fernandez, E. Alegre, W. Al-Nabki, and V. Gonzalez-Castro, "Phishing URL detection: A real-case scenario through login URLs," *IEEE Access*, vol. 10, pp. 42949–42960, 2022, doi: 10.1109/ACCESS.2022.3168681.
[32] M. Korkmaz, O. K. Sahingoz, and B. Diri, "Feature selections for the classification of webpages to detect phishing attacks: A survey," in *HORA 2020 - 2nd International Congress on Human-Computer Interaction, Optimization and Robotic Applications, Proceedings*, IEEE, Jun. 2020, pp. 1–9. doi: 10.1109/HORA49412.2020.9152934.
[33] A. Aljofey *et al.*, "An effective detection approach for phishing websites using URL and HTML features," *Scientific Reports*, vol. 12, no. 1, p. 8842, May 2022, doi: 10.1038/s41598-022-10841-5.
[34] T. Kim, N. Park, J. Hong, and S. W. Kim, "Phishing URL detection: A network-based approach robust to evasion," in *Proceedings of the ACM Conference on Computer and Communications Security*, New York, NY, USA: ACM, Nov. 2022, pp. 1769–1782. doi: 10.1145/3548606.3560615.
[35] A. K. Jain and B. B. Gupta, "Towards detection of phishing websites on client-side using machine learning based approach," *Telecommunication Systems*, vol. 68, no. 4, pp. 687–700, Aug. 2018, doi: 10.1007/s11235-017-0414-0.
[36] C. Patsakis and F. Casino, "Hydras and IPFS: a decentralised playground for malware," *International Journal of Information Security*, vol. 18, no. 6, pp. 787–799, Dec. 2019, doi: 10.1007/s10207-019-00443-0.
[37] S. S. Roy, U. Karanjit, and S. Nilizadeh, "A large-scale analysis of phishing websites hosted on free web hosting domains," Jan. 2024, [Online]. Available: http://arxiv.org/abs/2212.02563
[38] B. de Ville, "Decision trees," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 5, no. 6, pp. 448–455, Nov. 2013, doi: 10.1002/wics.1278.
[39] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967, doi: 10.1109/TIT.1967.1053964.
[40] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324.
[41] E. Bauer and R. Kohavi, "Empirical comparison of voting classification algorithms: bagging, boosting, and variants," *Machine Learning*, vol. 36, no. 1, pp. 105–139, Jul. 1999, doi: 10.1023/a:1007515423169.
[42] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA: ACM, Aug. 2016, pp. 785–794. doi: 10.1145/2939672.2939785.
[43] G. Ke *et al.*, "LightGBM: A highly efficient gradient boosting decision tree," *Advances in Neural Information Processing Systems*, vol. 30, pp. 3147–3155, 2017.
[44] I. D. Mienye and Y. Sun, "A survey of ensemble learning: concepts, algorithms, applications, and prospects," *IEEE Access*, vol. 10, pp. 99129–99149, 2022, doi: 10.1109/ACCESS.2022.3207287.
[45] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986, doi: 10.1038/323533a0.
[46] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python. The Journal of Machine Learning Research," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011, [Online]. Available: http://scikit-learn.sourceforge.net.
[47] T. N. Phyu, "Survey of classification techniques in data mining," in *Proceedings of the International Multiconference of Engineers and Computer Scientists*, 2009, pp. 727–731.
[48] H. Taherdoost, "Insights into cybercrime detection and response: A review of time factor," *Information (Switzerland)*, vol. 15, no. 5, p. 273, May 2024, doi: 10.3390/info15050273.

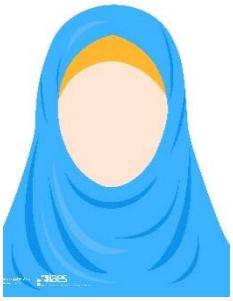
BIOGRAPHIES OF AUTHORS



Jawaher Alharbi is a Master's student in Information Technology at King Abdulaziz University, Jeddah, Saudi Arabia. She received her Bachelor's degree with First Class Honors in Computer Science from Umm Al-Qura University, Makkah, Saudi Arabia. Her research interests include cybersecurity with a special focus on phishing, cyber threat analysis, network security, and machine learning-based security systems. She can be contacted at email: jalharbi0086@stu.kau.edu.sa.



Dr. Manal Bayousef received her B.S. degree in Computer Science from King Abdulaziz University, Jeddah. She obtained her M.S. and Ph.D. in Information Security and Computer Science from Florida State University, USA. She is currently an Assistant Professor at the Faculty of Computing and Information Technology, King Abdulaziz University. Her research focuses on cybersecurity, with particular emphasis on cyber threat detection using machine learning techniques, as well as privacy-preserving systems and blockchain-based security frameworks. Her work also includes blockchain-based access control for secure and scalable distributed environments. In addition to her research, she teaches postgraduate-level courses in information security and supervises postgraduate students in cybersecurity research projects. She can be contacted at email: mbayousif@kau.edu.sa.



Hind Almisbahi is an Assistant Professor in the Information Technology Department at King Abdulaziz University (KAU), Jeddah, Saudi Arabia. She holds a Ph.D. in Machine Learning and Data Mining from the University of Surrey, where her research focused on analyzing graphene vibrations using molecular dynamics simulations and dimension reduction techniques. She also earned both her MSc and B.Sc in Computer Science from KAU. Dr. Hind's research interests include machine learning, deep learning, molecular dynamics simulation, and computational physics. Her work reflects an interdisciplinary approach, applying data-driven methods to scientific, environmental, and healthcare domains. She has authored several publications covering heat transfer performance, water recovery systems, and pollution prediction models. Her research also includes improving forecasting accuracy through CNN-based models and CWT scalograms. More recently, her research has expanded to include interpretable AI for healthcare applications, particularly in forecasting for Type 2 Diabetes, as well as real-time anomaly detection in multivariate time-series data using advanced deep learning techniques. She has also contributed to satellite-based environmental monitoring and predictive modeling of air pollution across cities with varying characteristics.
In addition to her research, Dr. Hind has teaching experience at KAU, supervising senior projects and teaching courses in data mining and programming. She is also actively involved in academic service, including curriculum development and accreditation efforts. She can be contacted by email at halmisbahi@kau.edu.sa.