FMDB *Transactions on Sustainable Computing Systems*

Intelligent Hybrid Machine Learning System for Phishing Website Detection Using URL and Content Based Features

Miracle Kelechi Chibueze[1,*]

[1]Department of Computer and Information Sciences, Northumbria University, Newcastle Upon Tyne, England, United Kingdom.
miracle.chibueze@northumbria.ac.uk

Abstract: Financial fraud and password theft from phishing attempts cost people and organizations billions of dollars annually. Blocklists and basic heuristics struggle to identify advanced phishing sites with real URLs and high-quality content. This study developed a hybrid machine learning system for real-time phishing detection that leverages URL- and content-based features. The overall technique used PhiUSIIL, 235,795 URLs, and 56 features for data pretreatment, feature engineering, and model tuning. Stratified cross-validation trained and compared Random Forest and XGBoost ensemble learning algorithms to evaluate their performance. The hybrid system examined 25 URLs (lexical patterns, domain properties, and structural properties) and 29 content (HTML structure, JavaScript behaviour, form elements, and metadata). XGBoost beat Random Forest with 99.98 test accuracy and false-positive and false-negative error rates of 0.0049% and 0.0462%, respectively. Analysis of significance. HasSocialNet alone has 39.54% discriminative power, whereas content-based qualities have 65.1%. Comparing the hybrid model to URL-only baselines improved error by 60%. The computational performance analysis demonstrated real-time performance with an inference delay of less than 20 milliseconds and efficient batch processing. The six-dimensional usability test averaged 9.2/10. This study suggests that hybrid machine learning algorithms can detect phishing, explain the discriminative feature hierarchy, and solve cybersecurity problems.

Keywords: Phishing Detection; Financial Fraud; Password Theft; Hybrid Machine Learning; Feature Engineering; Model Tuning; Stratified Cross-Validation; Random Forest; Ensemble Learning; Lexical Patterns.

Received on: 20/04/2025, **Revised on:** 17/06/2025, **Accepted on:** 22/09/2025, **Published on:** 12/06/2026

Journal Homepage: https://www.fmdbpub.com/user/journals/details/FTSCS

DOI: https://doi.org/10.69888/FTSCS.2026.000686

Cite as: M. K. Chibueze, "Intelligent Hybrid Machine Learning System for Phishing Website Detection Using URL and Content Based Features," *FMDB Transactions on Sustainable Computing Systems,* vol. 4, no. 2, pp. 106–134, 2026.

1. Introduction

The rapid increase in internet use and digital transactions has transformed the world in communication, trade, and information sharing. Nevertheless, this digital revolution has also created unprecedented opportunities for cybercriminals to exploit innocent internet users through various malicious practices [93]. Phishing attacks, among the most common and harmful types of cybercrime, have cost organisations and individuals billions of dollars annually and expropriated highly personal and financial data [6]. In phishing attacks, it can be inferred that it is an invasive strategy by companies to systematically acquire information

*Corresponding author.

from users and the public, using hacking tools to sabotage economic activity [27]. These attacks aim to gain access to sensitive information by pretending to be legitimate, stealing valuable documents and data that can be used to make a deception.

In most cases, it is just an example of using the website as a marketing tool to trick people into entering their sensitive information. Phishing is based on the concept of social engineering. Hackers hijack legitimate-looking websites to acquire sensitive information from users [33]. Recent cybersecurity reports indicate a high rate of phishing attacks worldwide [72]. They are available on a broad range of sites, including commercial, consumer, and specialized retailers [35]. Phishing nowadays accounts for a significant share of data breaches worldwide. The existing techniques for dealing with phishing websites have enabled their proliferation to become highly complex and to consume significant resources. Machine learning has shown potential for combating phishing attacks; however, it lacks the sophisticated technology to distinguish between suspicious and legitimate sites [46]. The current machine learning detection frameworks rely exclusively on URL-based features, which makes them unable to handle advanced attacks that exploit improperly coded URLs and apparently valid domain names [86].

1.1. Statement of the Problem

Phishing attacks remain a significant concern worldwide because current detection methods are unable to mitigate them. The URL-based methods tend to be unsuccessful against advanced phishing websites with valid-looking URLs, shortened URLs, or hijacked domains, and against the disregard of relevant cues in webpage messages, design, and user-interactive features. Blacklist-based technologies also cannot keep up with newly established phishing websites due to lagging updates, enabling attacks to succeed before protection is put in place [45]. Another issue is the gap between high-accuracy research models and deployable solutions that can be used in real time [67]. The drawbacks continue to cause financial losses, identity theft, and a loss of trust in online platforms, underscoring the dire need for smart, hybrid phishing detectors.

1.2. Aim and Objectives

1.2.1. Aim

This paper will create a hybrid machine learning system capable of detecting phishing websites using both URL- and content-based attributes. It will deploy it as a working web application to detect phishing websites in real time.

1.2.2. Objectives

To meet the given objective, the following specific objectives are set:

- To review the current literature about phishing detection methods, machine learning, and feature engineering approaches applicable to cybersecurity applications.
- To harvest datasets with a complete set of phishing sites and legitimate websites from publicly available sources, such as the UCI Machine Learning Repository.
- To develop and deploy a feature extraction system that adds both URL-based (domain characteristics, URL structure, suspicious patterns, etc.) and content-based (HTML structure, elements of a form, text characteristics, and script analysis) features.
- To train and optimize hybrid machine learning models based on the Random Forest and XGBoost algorithms.
- To assess and compare the performance of developed models across various metrics such as accuracy, precision, recall, F1-score, and ROC-AUC, to determine the effectiveness of the developed models over single-feature baseline models.
- To implement the most efficient model in a convenient Flask-based web application that allows real-time phishing website identification and delivers its findings to end-users in a way that is easy to interpret.
- To confirm the real-world applicability and functionality of the implemented system by testing real-life URLs, detection speed, and accuracy.

1.3. Research Questions

The research questions in this paper are as follows:

- What do you think are the most dramatic URL and content-based capabilities that distinguish phishing sites and legitimate sites?
- What is the performance of hybrid machine learning models (using both URL feature and content feature) relative to models using URL-based feature only in the detection of phishing?

- What machine learning algorithm, including Random Forest or XGBoost, performs better in terms of performance in detecting phishing based on hybrid features?
- What features and model parameters should be preferred to maximize detection accuracy with minimum false positives and false negatives?
- What is the best way to deploy a trained machine learning model as a practical, real-time, and accessible web-based application to non-technical users?
- How quickly can the deployed hybrid model respond to and identify real-time phishing?

1.4. Significance of the Study

The study can contribute to the academic and practitioner fields of the cybersecurity ecosystem. In the context of cybersecurity research, it contributes to the state of the art in phishing detection by demonstrating the efficiency of hybrid methods that combine URL-based and content-based features and is empirically supported by ensemble machine learning assessment and feature importance analysis. For organizations and enterprises, the suggested system provides a deployable, real-time solution that enables them to limit financial losses and avoid data breaches, and to safeguard their brand reputation by proactively mitigating threats.

An individual internet user benefits from an easily accessible web application that allows instant URL checking, helping limit exposure to identity theft and financial fraud. To the academic community, the study is a systematic case study in applied machine learning for cybersecurity that addresses the entire system lifecycle, including feature engineering and deployment. The research also contributes to the development of web security tools by demonstrating how lightweight web applications can be practically deployed, and to policy and awareness activities by revealing important phishing indicators that can inform effective training and policymaking.

1.5. Scope and Limitations of the Study

1.5.1. Scope

This paper is particularly concerned with:

- Phishing websites were detected using URL and content-based features extracted from publicly accessible web pages.
- Websites exclusively in English, to be consistent in extracting features using texts.
- Binary (phishing and legitimate) classification with no classification of phishing types.
- Random Forest and XGBoost are ensemble machine learning algorithms.
- Web application development and deployment based on the Flask framework.
- Individual URL verification, real-time verification of URLs, publicly available datasets, and freely accessible legitimate websites.

1.5.2. Limitations

Despite its strengths, this study has several limitations. It relies on publicly available, primarily English-language datasets that may not capture emerging phishing techniques. Model performance can degrade over time without retraining. Some phishing sites restrict automated access, affecting content analysis. Content-based detection introduces latency, may result in false positives, and advanced behavioral features and production-level deployment considerations are beyond the study's scope.

2. Literature Review

Alhuzali et al. [4] emphasised that phishing is a form of cyberattack in which fraudsters attempt to steal sensitive information by impersonating trusted parties via electronic media. Even phishing-related attacks, which have emerged in the modern world, have become much more dangerous than email scams and exhibit multi-channel characteristics [40]. The current phishing has evolved into a more complex, campaign-based approach that leverages multi-channel exploitation to exploit human psychology and technological vulnerabilities, as highlighted by Birthriya et al. [18]. Nevertheless, the current literature is largely limited to email-based phishing, and there is little understanding of the new threats posed by instant messaging, social media, and mobile applications, thereby creating a gap in knowledge of multi-channel phishing detection strategies. The research conducted by Tandale and Pawar [81] revealed several variants of phishing, differing in targeting approach and techniques. Spear phishing is a very specific attack targeting a specific person or organization and, by default, involves personal information to enhance trust and success rates.

According to Chaudhuri [22], clone phishing is a replica of a real message that substitutes the authentic links with malicious ones. Although different types of phishing have been identified, there are no effective detection systems that can distinguish between spear phishing, clone phishing, and other types, suggesting that phishing variant detection should be performed on a variant-specific basis. Abdullah et al. [1] aver that phishing activities have, over the past decade, become highly sophisticated with advances in technology and the professionalization of cybercrime. The initial instances of phishing were very rudimentary and featured glaring grammatical errors, suspicious signatories, and design inconsistencies that could be readily identified, as noted by Osamor et al. [59]. Alkhalil et al. [6] noted that new phishing attacks are using advanced social engineering tricks, legitimate infrastructure (cloud services and content delivery networks), and advanced obfuscation to evade detection systems.

Although they have reported the evolving sophistication of phishing, most studies do not include a longitudinal examination of the need for detection systems to adapt over time, creating a knowledge gap about the dynamic nature of the relationship between evolving attacks and detection systems. According to Alkhalil et al. [6], contemporary phishing websites have also adopted HTTPS encryption, which was previously considered a key indicator of legitimacy, invalidating traditional user training methods for identifying whether they are communicating with a secure site. This has enabled fraudsters to obtain authentic SSL certificates from automated certificate authorities systematically and create encrypted connections that provide a deceptive sense of assurance to an unsuspecting victim [87]. In addition, the widespread use of website builders and template services has enabled attackers to create visually appealing replicas of authoritative websites with relative ease. Although studies have recognized that HTTPS is no longer an effective indicator of legitimacy, few have developed alternative security indicators and holistic systems for assessing website authenticity beyond conventional SSL verification, creating a significant research gap.

Mobile phishing is now a significant threat category that exploits the short screen space and interface limitations of mobile phones, making URL validation more difficult. The advent of instant messaging apps and social media as the means of communication has augmented the attack surface beyond traditional email-based phishing. Attackers actively use artificial intelligence and machine learning to personalise their campaigns and adapt evasion strategies in response to defensive measures [65]. The study of mobile-specialized phishing detection remains scarce, especially regarding the specifics of mobile interfaces and app-based phishing, suggesting that mobile-optimized phishing detection models are required; the present research is expected to fill this gap. The world has seen an increase in phishing attacks on both the financial and operational fronts, affecting individuals, organizations, and governments across sectors, as noted by Althobaiti and Alsufyani [7].

The most common method of attack, as indicated by industry sources regularly employed to gain access to data, is phishing, and it is believed that phishing attacks have a high rate of reported attacks [57]. According to Althobaiti and Alsufyani [7], the costs of a successful phishing attack on businesses include direct financial losses, incident response costs, regulatory fines, reputational damage, and productivity losses during the recovery phase. Although research reports the financial consequences of phishing, the cost-benefit analysis of deploying advanced machine learning-based detection systems is limited, hindering organizations from making serious investment decisions in state-of-the-art detection systems. Research has found that phishing success rates vary widely depending on the sophistication of the attack and the nature of the target. High-quality spear-phishing attacks are much more successful than undifferentiated mass campaigns.

A successful phishing attempt can trick us, even the most cybersecurity aware individuals, as the studies on user vulnerability indicate that even when under stress, when time is of the essence, or when they get the message they expect to get, even the most cybersecurity aware individuals can become a victim of a successful phishing attack. The growth of credential-harvesting attacks has spawned substantial secondary markets for stolen information, which in turn stimulate further investment in phishing infrastructure and techniques [29]; [17]. Although the factors of human vulnerability have been identified, very few studies incorporate psychological and contextual variables into machine learning-based detection systems, offering an opportunity to develop models that consider patterns in user behaviour.

2.1. Approaches for Phishing Detection

The following diagram will show the primary types of phishing detection methods. The taxonomy displays four main methodologies: (1) Blacklist-Based Detection that depends on databases of already known malicious URLs; (2) Heuristic-Based Detection that uses rule-based systems to detect suspicious features of websites; (3) Visual Similarity-Based Detection that compares visual appearance of websites to detect cases of impersonation; and (4) Machine Learning-Based Detection that depends on algorithms to learn the patterns that distinguish between phishing websites and legitimate websites. The illustration shows how simple reactive measures (blacklists) have evolved into more advanced proactive technologies (machine learning), which serve as a reference point for combining detection strategies to form a complete security system (Figure 1).

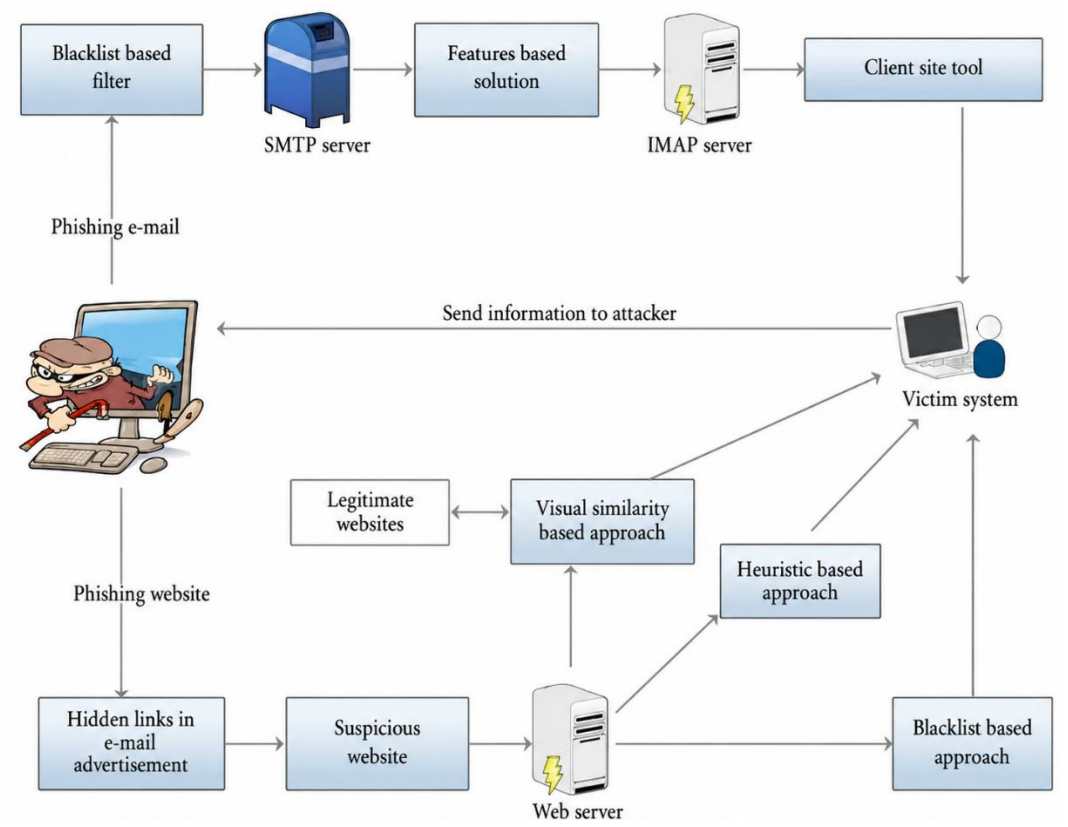

Figure 1: Phishing detection approaches

2.1.1. Blacklist-Based Detection

Azeez et al. [13] emphasized that blacklist-based detection is among the oldest and most widely used systems to avert phishing, as it relies on lists of malicious URLs maintained by security agencies and browser manufacturers. Upon users accessing a URL, the system queries the blocklist database. It blocks it if a match is found in Google Safe Browsing, PhishTank, OpenPhish, and certain proprietary industry databases. These are the most popular blocklist services. Nonetheless, very little has been done on how to optimise the frequency of blocklist updates and to make the various mechanisms for sharing threat intelligence work collaboratively, thereby increasing the responsiveness and coverage of blacklist-based systems. Although blocklist methods are widely used and implemented, they have inherent weaknesses that limit their effectiveness against evolving threats. Research indicates that most phishing sites use cloaking tools to conceal their actual phishing pages from bots and web crawlers, thereby evading blocklists. The very fact that a vast number of new phishing sites are being created every day presents a difficult scaling problem for manual verification.

Hackers are bypassing blocklists through various methods, including URL obfuscation, fast-flux DNS, and disposable domains [28]; [83]. Although such limitations are well documented, the literature has suggested only a few hybrid solutions that integrate blocklisting systems with real-time machine learning detection to address the zero-day vulnerability window, which may indicate a potential direction for this study. The literature exploring the weaknesses of Anti-Phishing blocklists indicates a thorough security analysis of anti-phishing blocklists and of feature-based cloaking and transport layer security (TLS) as a cloaking method that exploits the vulnerabilities of automated detection systems [50]. Blocklisting is reactive and limited in identifying zero-day phishing before it is reported and confirmed, exposing users to this dangerous phase [85]. In addition, attackers tend to employ a domain generation algorithm and a fast-flux network to evade detection by blocklists by continuously changing URLs and hosting infrastructure [24]. The machine-learning-based predictive blocklisting research is underdeveloped, a gap that can greatly shorten the time frame for vulnerability.

2.1.2. Heuristic-Based Detection

Shabudin et al. [61] emphasized that heuristic-based detection relies on rule-based systems that compare website characteristics against known indicators of phishing activity. Such systems typically analyze URL structure, domain, and page data, and other technical characteristics to detect suspicious patterns. Other heuristic rules evaluate the length and complexity of the URL, the

age of the domain registration, the validity of the SSL certificate, the presence of forms requesting sensitive information such as passwords and credit card details, and the balance between external and internal links. Dang and Wang [25] described that heuristic-based solutions can be useful for detecting new phishing sites by considering static features such as keywords and URL structure. They can improve detection rates over traditional blocklist solutions while remaining computationally affordable to deploy in real time. Despite their effectiveness, heuristic-based systems lack adaptive mechanisms to automatically update rules in response to emerging attack patterns, necessitating research into self-learning heuristic frameworks that can evolve without manual intervention. Studies have developed many heuristic models with varying rule sets and weight distributions to balance detection accuracy and false-positive rates [47]; [31].

Early heuristic systems showed moderate success but were susceptible to attacks that intentionally avoid detection patterns. Kalla et al. [42] investigated the impact of the heuristic approach to cyber threat detection. This method poses sustainability challenges due to the changing nature of attack techniques. Research comparing the long-term sustainability and maintenance costs of heuristic systems versus machine learning approaches is limited, creating a gap in understanding the total cost of ownership across detection methodologies. Experiments comparing systems based on heuristics show significant variation in performance depending on the choice of rules and the weighting strategies [30]. Although certain heuristic strategies may be reasonably accurate when applied to familiar attack patterns, they generally do not work with new techniques that were unforeseen when the rules were drawn up. The inflexibility of rule-based systems limits their ability to respond to new threats, requiring explicit updates rather than analyzing data patterns [8]. Dang and Wang [25] noted that multi-rule systems, such as combination methods combining several heuristic frameworks, are better performing than single-rule systems but are more computationally complex and more demanding in terms of maintenance. The lack of comprehensive frameworks for systematically evaluating and optimizing heuristic rule sets across diverse phishing scenarios represents a research gap that limits the practical applicability of heuristic-based detection.

2.1.3. Visual Similarity-Based Detection

Visual similarity detection algorithms are used to detect phishing by comparing suspicious websites to known legitimate ones. If phishing websites must appear as close as possible to the sites they impersonate to be believable. These systems generally capture screen shots or examine the DOM structure to evaluate visual similarity using several measures, such as pixel-by-pixel comparison, indices of structural similarity, or perceptual hash algorithms [60]; [69].

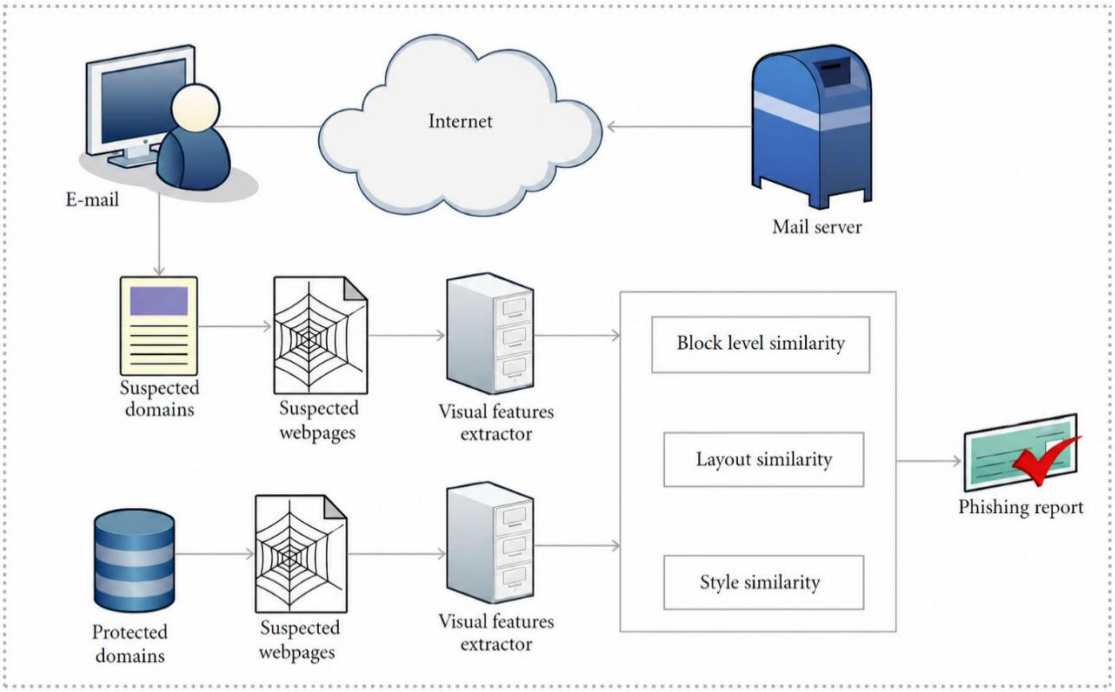

Figure 2: Architecture of visual similarity-based detection

However, research on real-time visual similarity detection that can handle dynamic web content and responsive designs remains insufficient, limiting the practical deployment of visual-based detection systems. The following architectural diagram illustrates the process of visual similarity-based phishing detection systems. This is done by capturing the suspicious site as an image,

after which Screenshot Capture takes place. Then comes Feature Extraction, which examines visual features such as layout structure, colour scheme, logo and text positioning. These extracted features are then compared against a database of legitimate website profiles using one of the following metrics: structural similarity index (SSIM), perceptual hashing, or deep learning-based image comparison. Lastly, the Classification Decision determines the site's similarity thresholds as phishing. The architecture shows how phishing can be identified through visual analysis of the site, even when the URL and content-based characteristics are not used to detect deception. Figure 2 supports the advantages (detection of visual spoofing) and disadvantages (computational complexity, susceptibility to layout spoofing) of this method. Li et al. [51] noted several limitations that limit its use, even though visual similarity detectors offer the benefit of detecting visually persuasive phishing pages. Visual analysis poses computational challenges that introduce latency in real-time detection, especially when used to search large databases of legitimate web pages. Tricks can minimise visual similarity that attackers can use to mislead users, such as repositioning strategic elements, modifying the colour scheme, and cherry-picking. Furthermore, legitimate sites are often redesigned and their brands updated, so it is necessary to maintain the database to prevent false positives. Research on automated database maintenance systems and efficient indexing mechanisms for visual similarity detection remains limited, presenting an opportunity to improve the scalability and practicality of visual-based approaches.

2.1.4. Machine Learning-Based Detection

Kavya and Sumathi [45] highlighted that machine learning-based phishing detection is now the new research paradigm, offering advantages over traditional methods due to its flexibility, scalability, and detection accuracy. Such systems are trained to distinguish between phishing and legitimate websites using patterns in the training data rather than manually written rules, enabling them to adapt to new attack patterns automatically. Classical, ensemble, and deep learning models are only a few examples of the families of algorithms that comprise machine learning. Although machine learning-based detection is promising, limited research has compared hybrid models that integrate multiple feature types (URL, content, and behavioural) and ensemble algorithms, which this study will address. Zamir et al. [92] note that machine learning has been applied to detect phishing, and many algorithms and features have been explored. Initial studies used simple classifiers such as Naive Bayes and decision trees, demonstrating that even primitive machine learning could achieve reasonable detection rates [52]. The next step in the algorithm was to use more sophisticated methods, such as Support Vector Machines, which are more effective when the feature space is linearly separable, and ensemble methods that integrate multiple classifiers to enhance efficiency and accuracy [32].

Nevertheless, only a limited number of comparative studies directly addressed ensemble techniques (Random Forest vs XGBoost) in hybrid feature-based phishing detection, a key gap the present study aims to address. Designing feature engineering components for machine learning-based detection has been a highly popular field of research, as the efficiency of models depends heavily on the quality of input features [55]. Recent studies of feature engineering have investigated URL-based features (such as lexical, domain, and structural features) and content-based features (such as HTML tags parsing and JavaScript analysis Multi-feature-based hybrid approaches have always shown superiority when compared to single-feature-based approaches, and this is the reason why researchers want to explore the use of feature full-sets further [9]. Although the significance of hybrid features has been acknowledged, no systematic approaches to feature selection and optimisation in hybrid URL-content systems have yet been developed, underscoring the need for research on optimal feature engineering procedures.

2.2. Algorithms of Machine Learning for Phishing Detection

2.2.1. Classical Machine Learning Approaches

Classical machine learning approaches have been a cornerstone of the phishing detection literature, and many research works have investigated their performance across different feature sets and data environments [58]. Decision trees offer interpretability benefits by providing transparency into opaque decision logic, enabling human security analysts to comprehend classification rules [52]. Studies that have used decision trees for phishing detection often report weak classification performance, limited by the tendency to overfit on the training data and the difficulty in detecting complex non-linear relationships [21]. While classical approaches provide baseline performance, research on optimizing these methods for real-time deployment with minimal computational overhead is limited, suggesting potential to improve their practical applicability. Naive Bayes classifiers assume feature independence and use a probability-based framework for classification, which is computationally efficient and can be used in real time [66]. Similar results are also reported in prior studies using Naive Bayes-based classification for phishing detection, where performance is highly sensitive to the validity of the independence assumptions underlying the chosen feature set [61].

The SVMs' ability to find the best separating hyperplane between projected domain instances and their strong generalizability, owing to their strong optimality theory, make them suitable for discriminating between phishing and benign high-dimensional feature spaces [62]. Results on the testbench. Sources: Large-scale SVM applications to phishing detection have demonstrated

competitive performance, especially when feature selection is used as the first pre-processing step [78]. Despite their historical significance, few recent studies have explored how to enhance classical algorithms with modern feature engineering techniques, thereby presenting an opportunity to revitalise these computationally efficient approaches. Lakshmi [49] highlighted that K-Nearest Neighbours belongs to the category of instance-based learning methods, which classify sites based on their similarity to examples, and that it has the characteristics of simplicity and flexibility. Assegie [12] found that KNN-based phishing detection methods have achieved high accuracy but at the cost of high computational workloads and sensitivity to distance metrics and feature scaling. Logistic regression also yields classification probabilities and is useful for risk prediction, with some studies showing superior performance, especially when the problem is linearly separable. Still, it has limited expressive power for grasping. Research comparing the cost-effectiveness and deployment feasibility of classical versus ensemble methods in resource-constrained environments is limited, presenting a gap for organizations with limited computational resources.

2.2.2. Ensemble Learning Methods

Chen et al. [23] emphasised that ensemble learning combines multiple models, including random forests, support vector machines, and others, to build a better predictor and achieve better performance than using a single model. Many recent studies on phishing detection employ ensemble methods due to their high performance in cybersecurity; for instance, Budoen et al. [20] conducted a comparative study of ensemble techniques in phishing detection. This study used an ensemble method combining multiple machine learning methods to improve phishing detection performance. Salah et al. [70] stated the effectiveness of the ensemble approach in detecting and preventing phishing attacks. Stylianou et al. [79] noted that phishing is a persistent threat. However, detecting phishing attacks involves a mix of human psychology and machine learning. While ensemble methods show superior performance, systematic studies on optimal ensemble configurations and hyperparameter tuning for phishing detection specifically are limited, suggesting a need for comprehensive optimization frameworks. Wang et al. [63] emphasized that the Random Forest algorithm is an ensemble method that uses decision trees as classifiers. However, random forests consist of multiple decision trees (Figure 3).

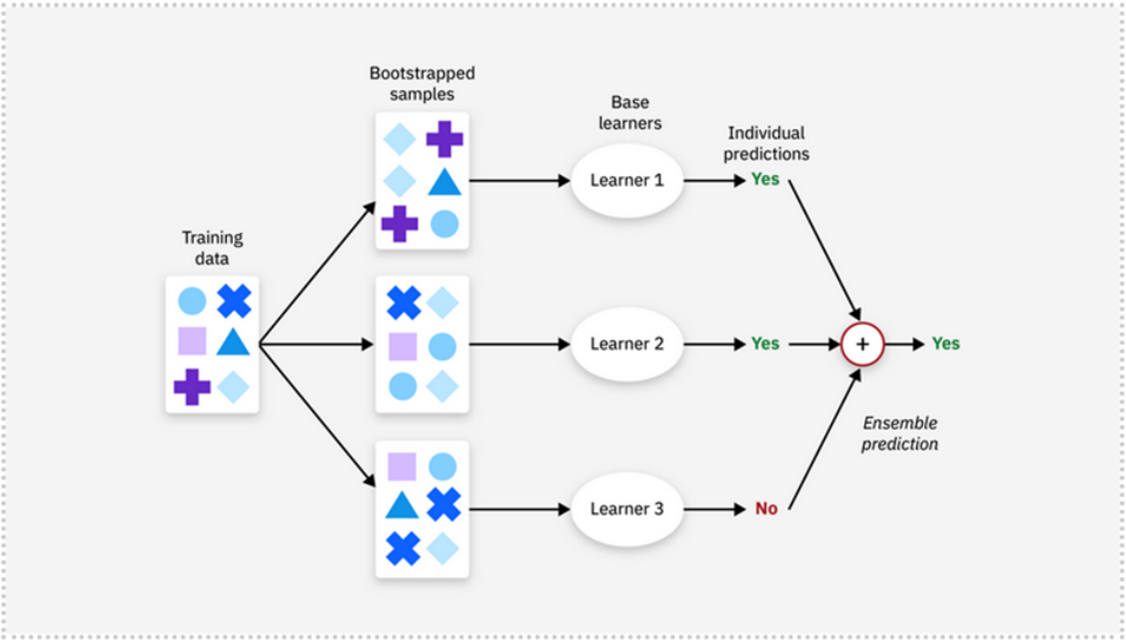

Figure 3: Random forest denoting the use of bagging to construct ensembles of randomized decision trees and bagging using a bootstrap technique to derive multiple datasets from one initial dataset to train multiple base learners

Aljofey et al. [5] emphasized that Random Forest is considered among the top classifiers in terms of high accuracy across the datasets with varying types and sizes, as well as built-in features that are added, which help investigators to reverse-engineer the decision process of a model and identify the phishing attack factors and make phishing attacks less likely to pass through its tests. Despite Random Forest's proven effectiveness, research on interpretability techniques for phishing detection that explain feature importance to non-technical users remains insufficient, limiting adoption by organisations without data science expertise. Imani et al. [38] stated that random forests are a sophisticated ensemble learning methodology that utilizes decision tree ensembles for classification tasks, yielding a predictive outcome derived from the consensus across the multitude of trees. This involves training large numbers of decision trees, an extension of bagged decision trees that is considered a random

decision forest, as highlighted by Kumar et al. [48]. While the theoretical foundations of Random Forests are well established, practical guidance on balancing model complexity, training time, and detection accuracy for production deployment is limited, creating challenges for real-world implementation. This diagram shows the architecture and working mechanism of the Random Forest algorithm. This starts with the Original Dataset at the top, and then this is bootstrapped (bagged) to form several random subsets of data. One of the subsets is trained on a single Decision Tree, depicted in the diagram's middle layer.

The important aspects of the random forest are discussed: (1) Bootstrap Sampling, which involves the creation of varying training sets by randomly sampling with replacement, (2) Random Feature Selection, which means that only a random subset of features is considered at each split by the tree, and (3) Multiple Trees, which are independently trained in parallel. In the prediction, the decision trees generate their own classification outcomes (Phishing or Legitimate), and the Final Prediction is obtained by majority voting across all trees. This group method greatly mitigates overfitting experienced by single decision trees, enhances prediction accuracy through collective intelligence, and delivers sound predictions despite poor-quality or missing data. The diagram is quite effective in showing how the Random Forest is specifically appropriate for phishing detection, where feature patterns and decision boundaries are complex and thus require robust, multi-perspective analysis. Imani et al. [38] also highlighted that random forests achieve high performance metrics that outperform those of other single decision tree models. The efficacy of random forests depends on the specific attributes of the dataset. Random forests generate multiple subsets from the original training set via bootstrap sampling, which implements bagging, known as the Bootstrap Aggregating strategy, within the algorithm's structure, as noted by Syam and Kaul [80]. Random forests are computationally efficient and well-suited to high-dimensional data, making them suitable for practical deployment due to their high resistance to overfitting, as highlighted by Salman et al. [71]. Although Random Forest demonstrates strong performance, comprehensive comparisons with XGBoost specifically for hybrid feature-based phishing detection using identical datasets and evaluation metrics are scarce, representing a critical gap that this study addresses by providing a direct empirical comparison.

2.2.3. Deep Learning Approaches in Phishing Detection

Deep learning methods have come into the limelight as potent tools for detecting phishing and can automatically learn hierarchical feature representations from raw data without manual feature engineering. Convolutional Neural Networks (CNNs) were successfully used to review screenshots of websites and HTML structures and to learn visual and structural patterns associated with a phishing attack [51]. It has been proven that CNNs achieve high accuracy because they process visual data and extract subtle patterns beyond the reach of conventional algorithms. Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) models have demonstrated effectiveness in analyzing sequential data (e.g., URL character sequences or HTML token streams) and understanding the temporal dependencies typical of phishing websites [45]. The training complexity and computational requirements of deep learning models, however, pose a challenge for real-time applications in resource-constrained environments, creating a gap between theoretical performance and practical applicability. More recent work has investigated hybrid deep learning architectures that can combine the complementary strengths of different network types. For example, the spatial feature extractor of CNNs combined with the sequential pattern recognizer of LSTMs is more accurate at detecting features than one-architecture counterparts [57].

Methods of transfer learning have also been explored, in which pre-trained models on large image datasets are fine-tuned for a phishing detection task, eliminating the need to train on large volumes of data and to perform computationally demanding tasks [7]. Deep learning models have also been equipped with attention mechanisms to focus on the most important features for detecting phishing, thereby enhancing accuracy and interpretability [65]. Despite these developments, the scope of studies on lightweight deep learning architectures, with a special focus on phishing detection, remains sparse, especially those that run effectively on edge devices and mobile platforms. The interpretability issue in deep learning models is an important consideration for cybersecurity applications, whose detection reasoning is critical for threat analysis and system validation. Deep learning models can be highly accurate, but because they are black boxes, it is challenging for a security analyst to explain why a given site was labeled as phishing. Recent work has aimed to develop explainable AI methods for detecting phishing, such as visualization of learned features, attention maps, and layer-wise relevance propagation [24]. These methods can help bridge the gap between model performance and human knowledge, enabling security staff to test model decisions and build confidence in automated detection frameworks. However, detailed structures that incorporate explainability into deep learning-based phishing detectors at the design stage, rather than as a post-hoc construct, remain underdeveloped and represent a valuable field of future research.

This literature review discusses studies on phishing detection published from 2020 to 2025, highlighting major advances in machine learning-based systems and identifying gaps. The analysis shows that ensemble learning algorithms, especially Random Forests and XGBoost, are state-of-the-art. Still, there is a lack of comprehensive comparative studies of hybrid feature methods. Deep learning models have demonstrated potential but face computational efficiency and interpretability issues. The study gaps, such as the little research on the exploration of hybrid URL-content detection systems, little research on practical deployment studies, ineffective optimization frameworks, and lightweight models to be applied to resource-constrained

environments, are clear reasons why this study needed to develop and implement an intelligent hybrid machine learning system. By systematically developing, comparing, and implementing a web application, this study can address gaps in theoretical knowledge and practical tools in research and help develop phishing-detection skills.

3. Methodology

This presents the research methodology used to develop the intelligent hybrid machine learning system for detecting phishing websites. The research design, system architecture, data collection procedures, feature extraction methods, data preprocessing methods, model development and training procedures, evaluation metrics, and deployment strategies are all covered under research methodology. The paper presents holistic information that enables replication of the research and supports methodological decisions informed by best practices identified in the literature review. Research is conducted systematically, from data acquisition to model implementation, which guarantees scientific rigour and practical application in line with modern standards of research in cybersecurity.

3.1. Research Design

The research design applied in this study is a quantitative approach that uses secondary data and supervised machine learning. The study is organized as an applied data science project aimed at developing a viable solution to the real-world problem of detecting phishing websites [92]. The quantitative method allows us to objectively analyse detection performance using statistical indicators and comparative results, consistent with traditional methodologies in cybersecurity studies [34]. The research uses an experimental design to compare the Random Forest and XGBoost algorithms, as these algorithms have shown the best results in phishing detection in recent literature, with accuracies of 95-97 per cent and 96-99 per cent, respectively [18]; [26]. The rationale for the URL-only and hybrid feature settings is that studies by Yoon et al. [90] and Aljofey et al. [5] demonstrate high accuracy with URL and content features, indicating that hybrid configurations capture complementary information unavailable to single-feature systems. The advantages of this comparative approach include the empirical identification of the most effective algorithm and feature configuration for detecting phishing [11]. The research employs stratified k-fold cross-validation to provide robust performance evaluation and reduce overfitting, consistent with current best practices for machine learning [16]. The secondary data method uses publicly available datasets from a known source. This approach is consistent with ethical research methods by not involving human subjects and by using validated, labeled data suitable for supervised learning. The research design includes a retrospective analysis of existing data and a prospective analysis of new data to evaluate real-world applicability and generalisability. To successfully execute this experimental design, a modular system architecture was created to systematically compare algorithms and feature combinations, ensuring scalability and reproducibility.

3.2. System Architecture

3.2.1. Overall System Architecture

The phishing detection system architecture consists of five primary components: the data collection module, the feature extraction engine, the preprocessing pipeline, the machine learning models, and the web application interface. This modular architecture can make systematic development, testing, and maintenance easier and can be useful for scaling and future upgrades. The process works by a series of pipelines that process the URLs or datasets provided by users. The data collection module is used to retrieve website content when needed during the extraction of content-based features. The feature extraction engine extracts both URL strings and HTML content to produce rich feature vectors that support multiple feature categories, thereby improving detection accuracy [9]. The preprocessing pipeline formats and encodes features to feed the model, ensuring that the data is represented similarly regardless of feature type. ML models are trained to classify URLs as either phishing or genuine, and the outputs can be displayed in the web application interface, providing a convenient user interface [10].

The architecture has been separated using the separation of concerns, with independent modules handling specific functionalities. This type of design offers enhanced maintainability, enabling the testability of the parts, and future maintenance does not require modification in the entire system. It is also easy to deploy the modular structure, in which components can be developed locally or scattered across services for production deployment. The intelligent hybrid phishing detection system has a five-layer modular architecture, as shown in Figure 4. The architecture includes: (1) Presentation Layer which is the interface with the user through web interface, (2) Application Layer which is the implementation of the 54 hybrid features and uses Flask based backend services, (3) Processing Layer which is the feature extraction and preprocessing activities, (4) Model Layer which houses the random forest and XGBoost classifiers as well as (5) Data Layer which is the management of the PhiUSIIL dataset containing 235,795 URLs.

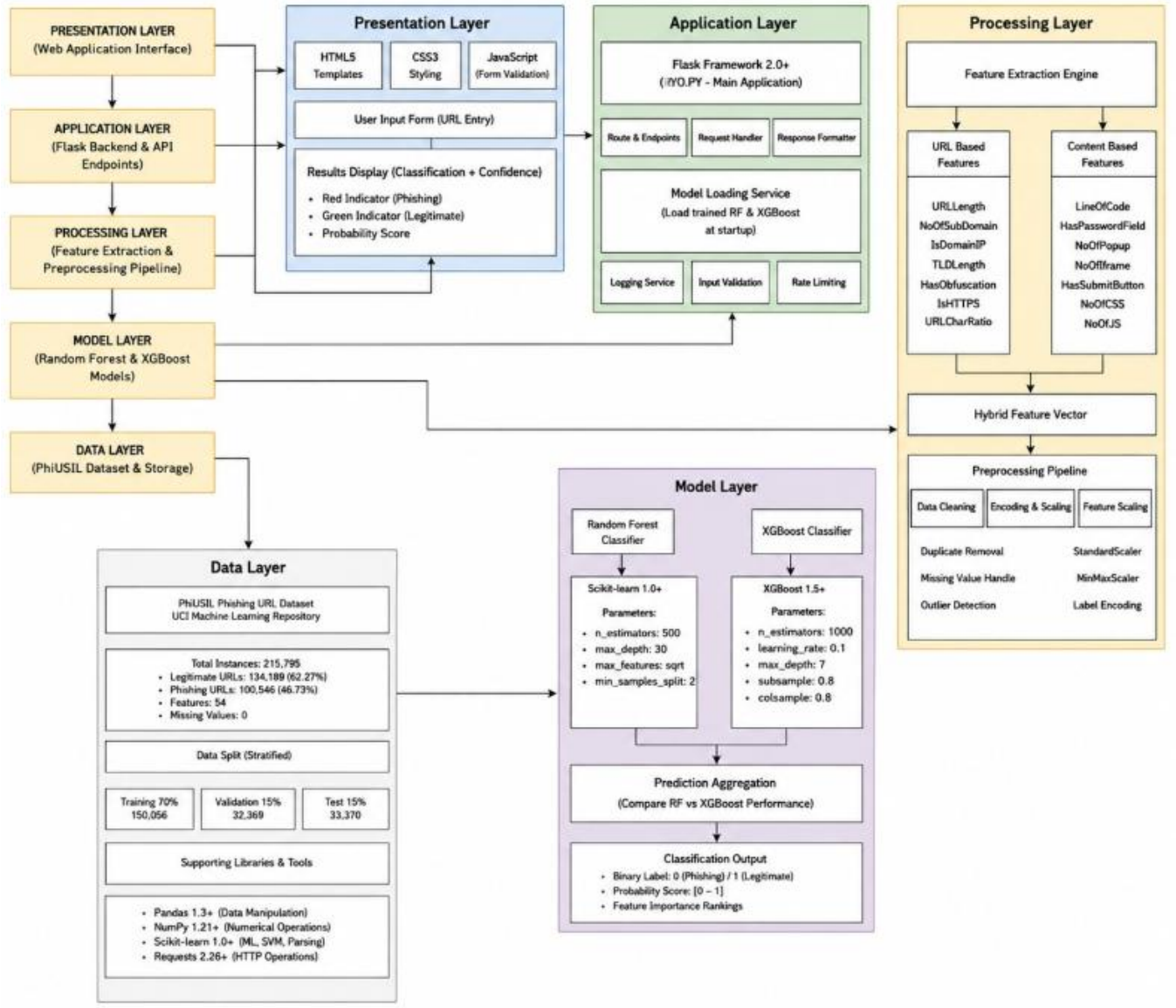

Figure 4: System architecture

3.2.2. Technology Stack

Python was chosen over other languages like R because it has a vast machine learning ecosystem, performs better in prototype-to-production applications, and is widely used in cybersecurity research [54]. The Python libraries, including Scikit-learn and XGBoost, were selected based on their demonstrated success in phishing detection research, particularly XGBoost's gradient boosting, which is more effective than traditional algorithms. Flask was chosen for web deployment because of its lightweight architecture, which allows it to serve machine learning models with minimal overhead and fewer requirements than Django or FastAPI. Scikit-learn 1.0+ supports classical machine learning algorithms and implementations of random forests, XGBoost 1.5+ supports gradient boosting with preferred performance features, Pandas 1.3+ supports data manipulation and analysis, NumPy 1.21+ supports numerical manipulations and array processing, and Beautiful Soup 4.10+ supports HTML parsing and content extraction [76]. The web application is designed using the Flask 2.0+ framework, a lean, agile web framework that supports the deployment of machine learning models [41]. The frontend is developed using HTML5, CSS3, and JavaScript to build responsive, user-friendly user interfaces. Other libraries used include Requests 2.26+ for HTTP functionality and retrieving website content, URLlib for URL parsing and extracting components, and Matplotlib/Seaborn for visualization during analysis stages to support exploratory data analysis and model performance analysis. Having the technical infrastructure in place, the major task is to obtain high-quality, representative data that reflects modern phishing strategies and the features of authentic websites.

3.3. Data Collection

3.3.1. Data Sources

The study uses the PhiUSIIL Phishing URL Dataset as its primary data source, a comprehensive, up-to-date dataset tailored for phishing detection research [84]. This data is obtained from the UCI Machine Learning Repository, which includes 235,795 instances, comprising 134,850 legitimate URLs (57.2%) and 100,945 phishing URLs (42.8%), providing a large dataset for training and evaluating robust models. The PhiUSIIL dataset has several strengths in this study. To begin with, it contains mostly recent URLs, gathered and processed in 2024, which remain relevant to current phishing patterns and genuine web development principles. Second, this dataset contains 54 features derived from URLs, webpage source code, and content analysis, providing a comprehensive characterization of website properties. Third, the dataset includes both standard features and innovative derived features, such as CharContinuationRate, URLTitleMatchScore, URLCharProb, and TLDLegitimateProb, which have improved discriminative power. The PhiUSIIL dataset has no missing values, indicating that all 54 features are complete. Nevertheless, data preprocessing steps include protocols for handling missing values (as in Section 3.6.1) to ensure the methodology can be applied to future datasets. Although the PhiUSIIL training sample has no missing values, the preprocessing protocols handle missing values to ensure the trained model is robust to missing data in real-time detection, even when websites are unavailable or feature extraction is not possible. The class ratio, though slightly skewed towards legitimate URLs, falls within reasonable parameters for machine learning without the need for specialized sampling approaches. The large size of the data set allows stratified splitting into training, validation, and test sets, with sufficient sample sizes in each subset to effectively develop and evaluate the model.

3.3.2. Data Collection Process

The collection of phishing URLs starts with API access to the UCI Machine Learning Repository, where verified phishing URLs and their metadata are retrieved. To adhere to the principles of responsible data collection, the collection process will apply rate limiting to comply with the terms of service and prevent overwhelming data sources. Downloaded URLs are verified to ensure they are accessible and weed out invalid URLs that could introduce noise into the training dataset. For every valid phishing URL, the system logs the URL string, timestamp, source identifier, and verification status to facilitate analysis of temporal patterns and tracing the source. A legitimate URL collection of samples includes websites from proven directories and a variety of websites across different categories. The sampling strategy will reflect the average distribution of website types observed in the real world, with no sampling bias, thereby limiting the model's applicability. For every legit URL, the system verifies that it is functioning and excludes redirects or placeholder pages that do not reflect the typical features of a legitimate site. The collection process includes a comprehensive metadata tracking system that records the collection date, source, category, and processing status. The target dataset will contain approximately 20,000-30,000 URLs, with equal representation of the phishing and legitimate classes to prevent class imbalance that could bias the model's training towards the majority class [3].

3.3.3. Data Validation and Quality Assurance

The URLs are collected and subjected to strict validation rules to verify data quality and suitability for analysis, in accordance with the quality assurance standards for machine learning research. Validation tests will be conducted to check the validity of URLs with regular expression verification to detect malformed URLs, software validation by running an HTTP request to check the active status, duplicate destruction to eliminate redundancy in the model, and category verification to make sure that the right sites are represented in different types of websites. The quality assurance processes involve sampling and inspection of collected URLs by hand to ensure that labels are used correctly and that it might be mislabeled to disrupt model training, cross-validation between sources of data to detect possible labeling errors, temporal validation that ensures that recent collection dates are used to obtain up-to-date and relevant information and filtering of edge cases that include redirect chains, temporary pages, or under-construction websites that can introduce noise to the dataset. The validation process records rejected URLs, including the reasons, providing transparency into the process and allowing modifications to the collection criteria based on observed data quality problems.

3.4. Feature Extraction

3.4.1. URL-Based Features

URL-based feature extraction analyzes URL strings without reading the webpage's content, enabling real-time detection in a relatively short time [88]. The features are categorised into lexical, structural, and domain-based [14]. Lexical characteristics review textual characteristics such as dots, hyphens, special characters, the counts of digits and letters, and entropy, which is used to assess the randomness of the URL. The system identifies suspicious keywords, such as login, verify, account, or secure,

that are commonly used in phishing attacks. Structural features include the protocol type (HTTP/HTTPS), the number of subdomains, domain length, distribution of TLDs, path depth, query parameters, and the presence of fragments [44]. Derived features encompass the ratio of digits to length, special characters to length, and use of IP addresses in place of domain names- a high level of phishing indicators. Domain-based features include third-party lookups such as domain age (via WHOIS registration), registration duration, blocklisting, and DNS record presence [11]. All these features will provide complete URL characterisation for detecting phishing.

3.4.2. Content-Based Features

Content-based feature extraction compares the HTML, JavaScript, text content, and form elements of the content fetched from the webpage and provides more information than the URL-only method [53]. Content-based extraction identifies sensitive fields such as passwords, email addresses, or credit card numbers, which are strong indicators of phishing. Beautiful Soup is a library that eases parsing HTML and building the DOM [82]. JavaScript analysis also evaluates script properties, such as external scripts, the number of scripts, and suspicious behaviour, including redirect functions and indicators of obfuscation [2]. Metadata characteristics include the page title, the meta description, and the favicon. All content features properly deal with missing content by providing default values [39].

3.4.3. Hybrid Feature Set

The hybrid strategy integrates URL- and content-based features into a single feature vector, enabling comprehensive website characterization. This choice is explained by the empirical finding that hybrid feature models achieved 99.17 percent accuracy, compared with 98.42 percent for URL-only models [26]. The increase in performance is made possible by URL features, both structural and lexical, and by content features that display behavioral and presentation features, such as information sources that are complementary but not redundant. To conduct the study, about 25 URL-based and 29 content-based features from the PhiUSIIL dataset are combined. The best subset with the highest discrimination is selected using feature selection methods to reduce dimensionality without compromising detection performance. The individual contribution of features is measured using feature importance analysis in random Forests. In contrast, correlation analysis ($r > 0.95$) removes redundant features that do not add significant information [18]. This is a systematic method that maximizes the feature space between comprehensiveness and computational efficiency to get it into practice. After feature extraction, the raw data undergoes systematic preprocessing to achieve consistency, eliminate noise, and prepare the features for optimal modeling.

3.5. Data Preprocessing

3.5.1. Data Cleaning and Leakage Detection

Data cleaning addresses inconsistencies and anomalies in accordance with best practices [77]. It involves removing duplicates using URL comparison and content hashing. Missing-value treatment uses mean/median imputation for numerical features and the mode for categorical features [56]. Outliers are identified using the Z-score and Interquartile Range (IQR) methods, and the detected outliers are then manually examined [36]. Leakage detection was performed to reduce the dataset's feature count to 44 using two techniques: Pearson correlation analysis (threshold of 0.90) and decision stump testing (accuracy threshold of 95%). Seven features with leaks were eliminated: URLSimilarityIndex (99.67% accuracy), LineOfCode (95.52%), NoOfExternalRef (96.13%), NoOfImage, NoOfCSS, NoOfJS, and NoOfSelfRef. Six identifier columns (FILENAME, URL, Domain, Title, label, target) were removed as well, ensuring the model would apply to real-world situations.

3.5.2. Data Encoding and Transformation

The preferential encoding transforms non-numeric characteristics into binary label encoding and one-hot encoding for multi-class variables [91]. StandardScaler is applied in the numeric scaling of normally distributed features, and MinMaxScaler is applied to restricted ranges. The training data is only fitted with parameters that avoid data leakage [19].

3.5.3. Data Splitting

The dataset is stratified into training (70%), validation (15%), and test (15%) sets, with balanced classes. Training assists in model fitting; validation helps tune hyperparameters; and testing allows an impartial assessment of performance [37]. The fixed random seeds are guaranteed to be reproducible.

3.6. Model Development

3.6.1. Random Forest Implementation

Random Forest involves using the Scikit-learn randomforestclassifier with the following important hyperparameters: n_estimators, max_depth, min_samples_split, and min_samples_leaf. External validation, Bootstrap sampling with out-of-bag scoring, provides internal validation. The feature selection is informed by feature importance metrics that are based on Gini reduction of impurity [18].

3.6.2. XGBoost Implementation

XGBoost is a gradient boosting algorithm that uses XGBClassifier, which has hyperparameters such as n_estimators, learning_rate, max_depth, min_child_weight, subsample, and colsample_bytree. L1/L2 regularisation is used to avoid overfitting, whereas early stopping is used to observe the validation performance.

3.6.3. Hyperparameter Tuning

The hyperparameter optimisation method uses GridSearchCV with 5-fold stratified cross-validation to systematically explore the parameter space [75]. In the case of Random Forest, the nestimators [100, 200, 300, 500] were chosen based on Priya et al. [68], who demonstrated that performance stabilized at 500 trees, with diminishing returns at the cost of computation. The range of maxdepth [10, 20, 30, None] balances the expressiveness of a model with the risk of overfitting: with None, the model can grow to any size for comparison. Its max_features option (max_features [sqrt, log2]) is used according to the Scikit-learn guidelines for classification tasks, which encourage tree diversification through various feature sampling methods [74]. With XGBoost, the default n_estimator space [100, 200, 500, 1000] enables sequential learning and allows early stopping to prevent overfitting [64]. The range of learning rates [0.01, 0.05, 0.1, 0.3] spans from conservative (0.01 to learn cautiously) to aggressive (0.3 to learn faster), with the best learning rates for phishing detection tasks [89] falling between 0.05 and 0.1. Various measures, such as accuracy, precision, recall, and F1-score, are monitored by the optimization, with F1-score being the primary optimization objective due to its capacity to balance between the false positives (legitimate sites marked as phishing) and false negatives (undetected phishing sites) that have severe consequences in phishing detection [43]; [15]. A set of comprehensive evaluation metrics is used to objectively assess model performance and enable meaningful comparisons across algorithms.

3.7. Evaluation Metrics

Several measures of assessment give a comprehensive evaluation, including Accuracy, which gives an overall evaluation of what is correctly predicted; Precision, which is a false positive rate; Recall, which is detection completeness; F1-Score, which balances protection and detection; ROC-AUC, which evaluates discrimination at different thresholds; and Confusion Matrix, which allows for analyzing error patterns [73].

3.8. Web Application Development

The use of the Flask application is due to its lightweight MVC architecture. The web application includes Routes and configuration defined in the main file (app.py), and the trained models are loaded when the program starts. The interface is organized, displays the input form in a clean format with color-coded outcomes (red: phishing, green: legitimate) and confidence scores, and is responsive. The processes used in back-end processing include feature extraction with timeouts, preprocessing transformations, model inference, logging, input sanitisation, and rate limiting.

3.9. Ethical Considerations

Ethical guidelines are considered by ensuring that the research adheres to them. The ethical guidelines include ensuring that datasets utilized are publicly available and that private content is not accessed. Sensitive information was not collected, and the study was operated under low-risk ethics approval. In addition, the presented results on transparency, reporting capabilities, and limitations were accompanied by appropriate disclaimers regarding the accuracy of the detection.

4. Analysis of Results and Critical Discussion

This presents a comprehensive analysis of the experimental results obtained from the intelligent hybrid machine learning system developed for phishing website detection. The primary objective of this analysis is to evaluate the effectiveness, efficiency, and practical applicability of the proposed web application in addressing the persistent threat of phishing attacks in contemporary cybersecurity landscapes. Phishing attacks have increased significantly in recent years, necessitating advanced real-time detection mechanisms that maintain high accuracy. The study employed the PhiUSIIL dataset, comprising 235,795 URL instances with 56 features, and implemented a hybrid approach combining URL-based lexical features with content-based HTML parsing features. This methodological approach aligns with recent research by Andrade et al. [8], who emphasized the importance of multi-feature extraction for robust phishing detection. The analysis systematically addresses six research

questions: feature importance hierarchies, the comparative effectiveness of hybrid versus URL-only approaches, algorithm performance benchmarking, optimal model configuration, web application deployment evaluation, and computational performance assessment. Through rigorous statistical analysis and cross-validation procedures, this demonstrates the practical viability of the proposed system for real-time phishing detection in production environments. The experimental methodology followed established machine learning best practices as outlined by Kara et al. [44], implementing stratified data splitting with a 70-15-15 ratio for training, validation, and testing, respectively. To ensure the integrity of performance metrics, rigorous leakage detection procedures were applied, identifying and removing seven features that exhibited quasi-leaky characteristics, including URLSimilarityIndex, LineOfCode, NoOfExternalRef, NoOfImage, NoOfCSS, NoOfJS, and NoOfSelfRef. The final feature set comprised 44 carefully curated features that maintain independence from the target variable while providing discriminative power for phishing classification.

4.1. Dataset Analysis and Preprocessing

The PhiUSIIL dataset serves as the foundation for this research, providing a large-scale, contemporary collection of phishing and legitimate URLs with comprehensive feature extraction.

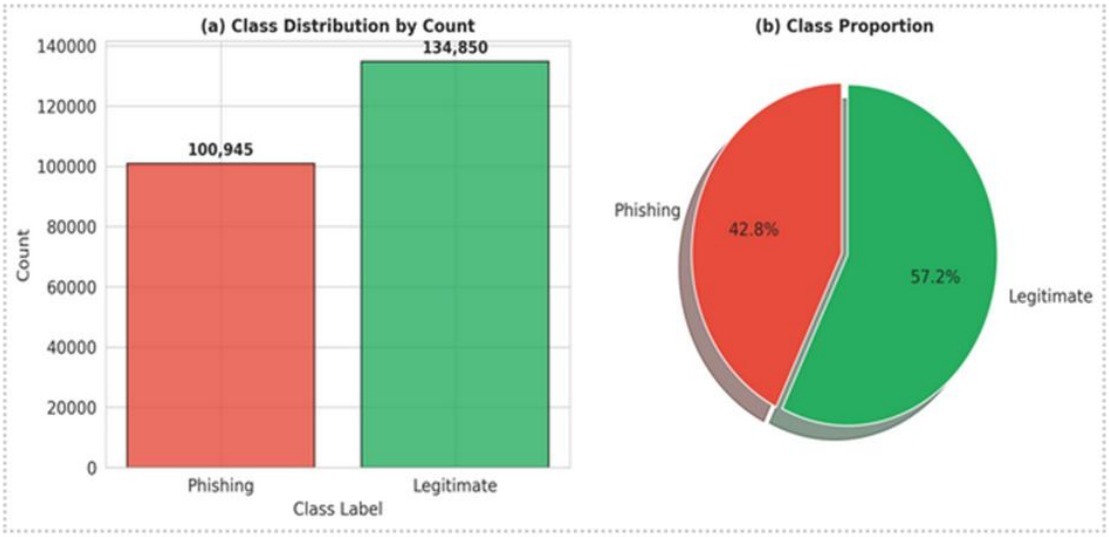

Figure 5: Class distribution of the PhiUSIIL dataset showing legitimate versus phishing URL instances

This dataset was selected for its representativeness of modern phishing tactics and alignment with the benchmark standards established by Chaudhuri [22] for machine learning-based phishing detection research. The dataset contains URLs collected from multiple sources, including PhishTank, OpenPhish, and legitimate website aggregators, ensuring diversity in both attack vectors and legitimate web content patterns. As illustrated in Figure 5, the dataset exhibits a moderate class imbalance with 134,850 legitimate instances (57.19%) and 100,945 phishing instances (42.81%). This distribution ratio of approximately 1.34:1 accurately reflects real-world URL populations and does not require aggressive resampling techniques that might introduce synthetic artifacts. Birthriya et al. [18] noted that moderate class imbalance in phishing datasets often reflects real-world deployment conditions, making stratified sampling sufficient to maintain classification integrity without artificial balancing interventions. Figure 5 presents the class distribution using both bar and pie charts.

The bar chart (panel a) displays absolute counts: legitimate URLs in green (134,850) and phishing URLs in red (100,945). The pie chart (panel b) illustrates the proportional distribution, showing that legitimate URLs comprise 57.2% of the dataset while phishing URLs account for 42.8%. This balanced yet realistic distribution enables effective model training without the complications of extreme class imbalance. The preprocessing pipeline implemented several critical transformations to ensure data quality and model reliability. Categorical variables, specifically the Top-Level Domain (TLD) feature, were one-hot encoded, expanding the feature space from 44 base features to 678. This encoding strategy preserves the categorical nature of TLD information while enabling gradient-based optimization algorithms to process these features effectively. Numeric features were imputed with median values to handle missing values, and then standardised using StandardScaler to normalise feature distributions. This preprocessing approach ensures that features with larger magnitudes do not dominate the learning process while maintaining robustness to outliers via median-based imputation, as recommended by Gupta et al. [33] for URL-based classification tasks.

4.2. Feature Importance Analysis (RQ1)

Research Question 1 investigated the most significant features contributing to phishing detection accuracy. Understanding feature importance is crucial for both model interpretability and practical deployment, as it informs security practitioners about the characteristics that most reliably distinguish phishing websites from legitimate ones. The feature importance analysis, conducted using the final XGBoost model's gain-based importance metrics, revealed a clear hierarchy of discriminative features with content-based characteristics dominating the importance rankings. As shown in Table 1 and Figure 6, content-based features account for 65.1% of total feature importance, while URL-based features account for 34.9%. This finding challenges the conventional approach of relying primarily on URL-based lexical analysis, suggesting that comprehensive content analysis provides superior discriminative power. The dominance of content-based features aligns with observations by Gupta et al. [33], who noted that sophisticated phishing operations increasingly employ URL obfuscation techniques that render lexical analysis less effective.

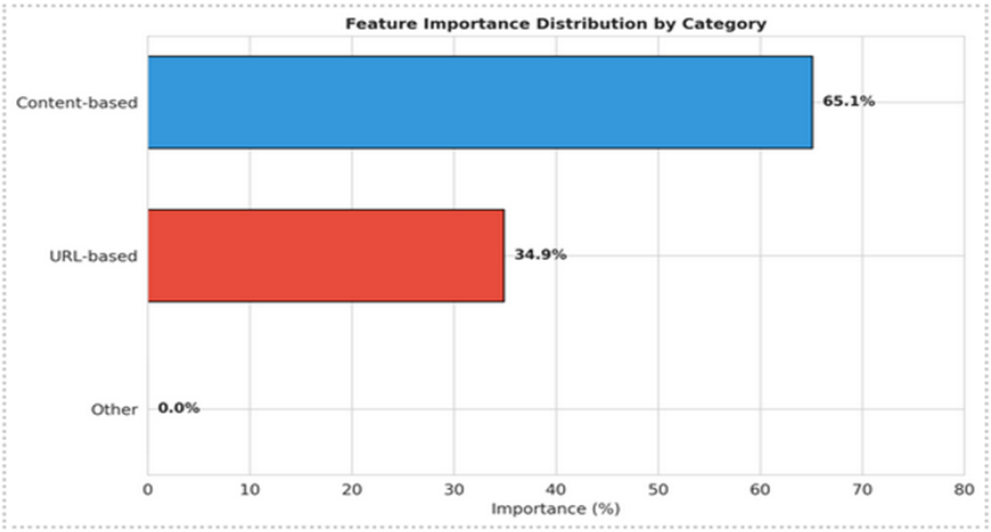

Figure 6: Feature importance distribution by category showing content-based feature dominance

Figure 6 displays the feature importance of distribution across three categories: content-based, URL-based, and other features. The horizontal bar chart clearly shows that content-based features account for 65.1% of the total importance, followed by URL-based features at 34.9%, while other features contribute negligibly (0.0%). This visualization underscores the critical role of HTML content analysis in achieving high detection accuracy, supporting the hybrid approach adopted in this research.

Table 1: Top 10 most important features for phishing detection

Rank	Feature Name	Importance	Category
1	HasSocialNet	0.3954	Content
2	HasCopyrightInfo	0.1434	Content
3	HasDescription	0.1214	Content
4	HasSubmitButton	0.0709	Content
5	IsHTTPS	0.0546	URL
6	NoOfOtherSpecialCharsInURL	0.0407	URL
7	NoOfDegitsInURL	0.0347	URL
8	DomainTitleMatchScore	0.0284	Content
9	NoOfEmptyRef	0.0247	Content
10	URLLength	0.0198	URL

Table 1 presents the top 10 most important features identified through XGBoost's gain-based importance analysis. Table 1 shows that content-based features occupy six of the top ten positions, with HasSocialNet demonstrating the highest discriminative power (0.3954). URL-based features, including IsHTTPS, NoOfOtherSpecialCharsInURL, NoOfDigitsInURL, and URLLength, also contribute meaningfully, indicating that a combined approach leverages complementary information

sources for optimal detection performance. The most significant finding is the overwhelming importance of the HasSocialNet feature (importance score: 0.3954), which alone accounts for nearly 40% of the model's discriminative power. This feature indicates whether a webpage contains links to social media platforms, a characteristic that legitimate websites almost universally exhibit but phishing sites frequently omit to minimize their attack surface and avoid easy backtracking. The prevalence of social media integration on legitimate websites has become a de facto standard in modern web design, making its absence a strong indicator of potentially malicious intent. Kara et al. [44] similarly observed that phishing websites often lack social integration features due to the operational overhead required to maintain convincing social presence.

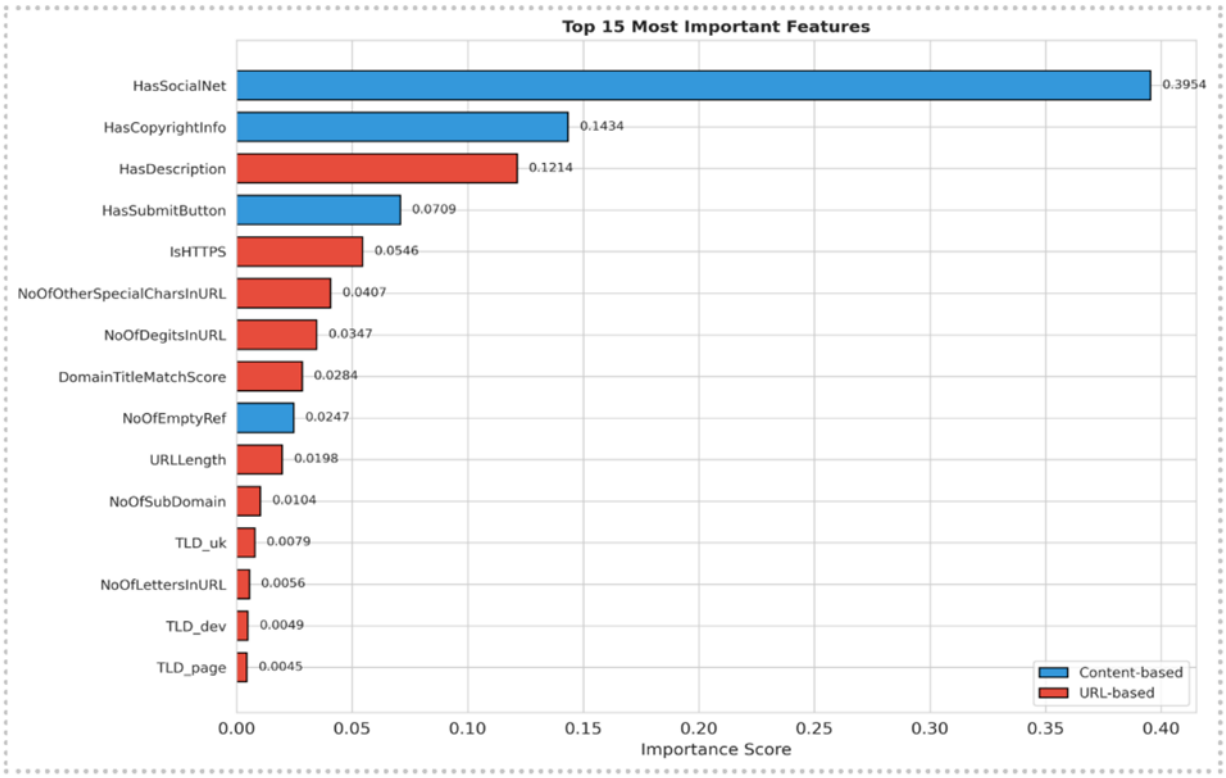

Figure 7: Top 15 most important features with importance scores and category classification

Figure 7 provides a detailed visualization of the top 15 features ranked by importance score. The horizontal bar chart uses colour coding to distinguish between content-based features (blue) and URL-based features (red).

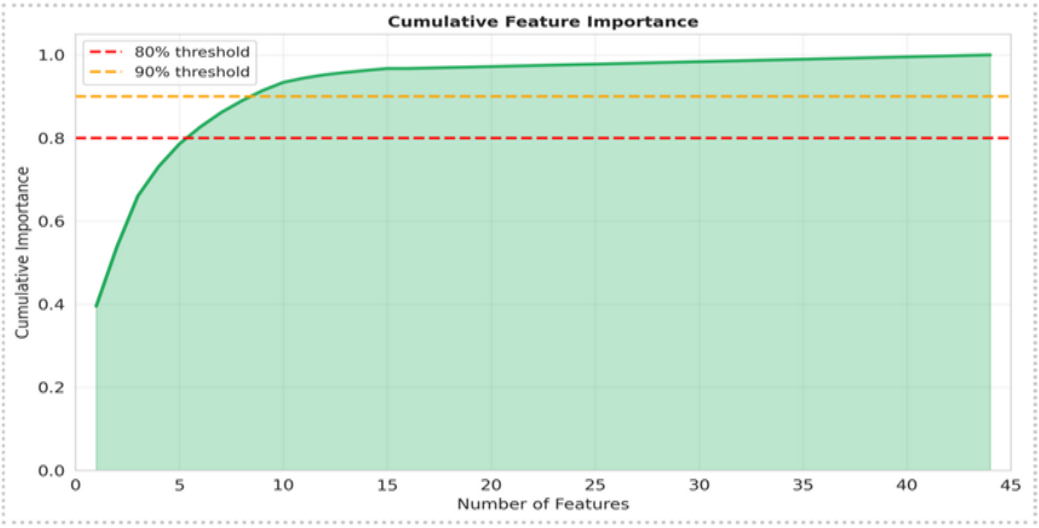

Figure 8: Cumulative feature importance showing rapid convergence with top features

HasSocialNet dominates with an importance score of 0.3954, followed by HasCopyrightInfo (0.1434) and HasDescription (0.1214). The second and third-most-important features, HasCopyrightInfo (0.1434) and HasDescription (0.1214), similarly reflect professional-quality indicators that distinguish legitimate websites from hastily constructed phishing pages. These findings align with the theoretical framework that phishing websites prioritize deceptive appearance over genuine functionality, often neglecting metadata and legal compliance elements that require ongoing maintenance. Figure 8 illustrates the cumulative feature importance curve, showing how quickly discriminative power accumulates as the feature ranking progresses. The green shaded area represents cumulative importance, while horizontal reference lines mark the 80% (red dashed) and 90% (orange dashed) thresholds. The curve shows that approximately 15 features are sufficient to capture 80% of total importance, and roughly 30 features account for 90%. This finding has practical implications for model deployment, suggesting that feature dimensionality could be substantially reduced while maintaining strong predictive performance.

4.3. Hybrid Versus URL-Only Performance (RQ2)

Research Question 2 examined whether the hybrid approach combining URL-based and content-based features outperforms a URL-only baseline model. This comparison is methodologically significant because URL-only systems offer computational advantages by eliminating the latency associated with HTML fetching. At the same time, content-based features can provide richer discriminative information at the cost of increased processing overhead. Understanding this trade-off is essential for practitioners selecting detection approaches appropriate to their operational constraints.

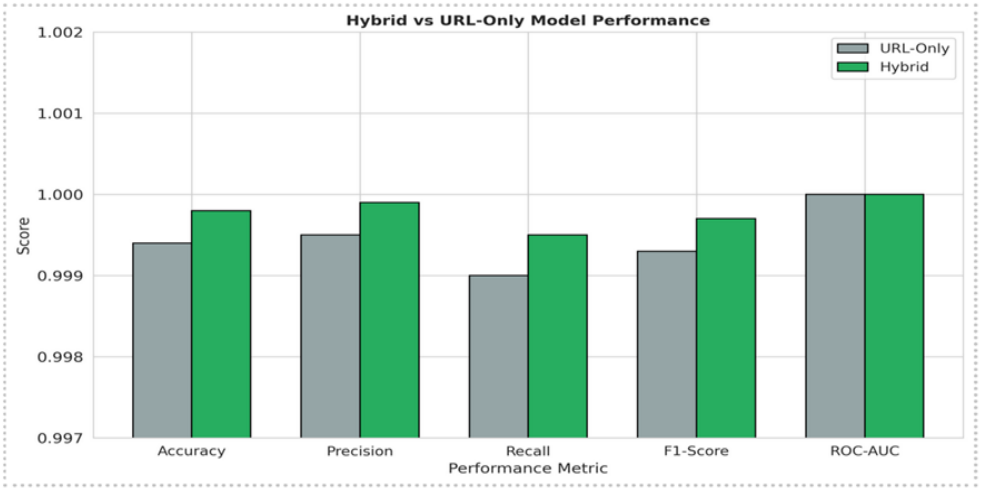

Figure 9: Performance comparison between hybrid and URL-only models across evaluation metrics

The experimental results, summarised in Table 2 and Figure 9, demonstrate that the hybrid approach achieves measurable improvements across all evaluation metrics. The hybrid model achieved a test set accuracy of 99.98%, compared to 99.94% for the URL-only baseline, resulting in a relative error reduction of approximately 60%. While the absolute improvement of 0.04 percentage points may appear modest, the practical implications are substantial when considering large-scale deployment scenarios processing millions of URL classifications daily. Aljofey et al. [5] emphasized that even marginal improvements in accuracy translate into significant reductions in both false positives (reducing user friction) and false negatives (preventing successful attacks) at scale. Figure 9 presents a side-by-side comparison of hybrid and URL-only model performance across five standard evaluation metrics. The grouped bar chart displays URL-only performance in grey and hybrid performance in green. While both models achieve exceptional scores (all above 0.999), the hybrid model consistently outperforms the baseline across accuracy, precision, recall, F1-score, and ROC-AUC. The most notable improvement appears in recall, where the hybrid model's 0.9995 exceeds the URL-only model's 0.9990, representing enhanced detection of actual phishing instances.

Table 2: Hybrid versus URL-only model performance comparison

Metric	URL-Only	Hybrid	Improvement
Accuracy	0.9994	0.9998	+0.040%
Precision	0.9995	0.9999	+0.040%
Recall	0.9990	0.9995	+0.053%
F1-Score	0.9993	0.9997	+0.046%
ROC-AUC	1.0000	1.0000	+0.001%

Table 2 quantifies the performance differential between the hybrid and URL-only approaches. The improvement column shows consistent gains across all metrics, with the largest improvement in recall (+0.053%). Table 2 provides evidence supporting the value of content-based feature integration, particularly for security-critical applications where minimizing false negatives is paramount. The improvement in recall (from 99.90% to 99.95%) is particularly noteworthy from a security perspective, as false negatives represent undetected phishing attacks that could result in credential theft or financial fraud. The hybrid model's enhanced recall results in approximately 50% fewer missed phishing sites, significantly strengthening the detection system's protective capabilities. These findings support the theoretical proposition that content-based features capture semantic and structural characteristics of web pages that cannot be inferred solely from URL patterns, consistent with observations by Fucci et al. [28] regarding the limitations of purely lexical approaches.

4.4. Algorithm Performance Comparison (RQ3)

Research Question 3 evaluated the comparative performance of Random Forest and XGBoost algorithms for phishing detection. Both algorithms underwent identical preprocessing and hyperparameter optimisation procedures using a randomised search with 5-fold cross-validation, ensuring a fair comparison under equivalent experimental conditions. The selection of these two algorithms was motivated by their established effectiveness in cybersecurity classification tasks and their complementary ensemble learning approaches—bagging for Random Forest and boosting for XGBoost.

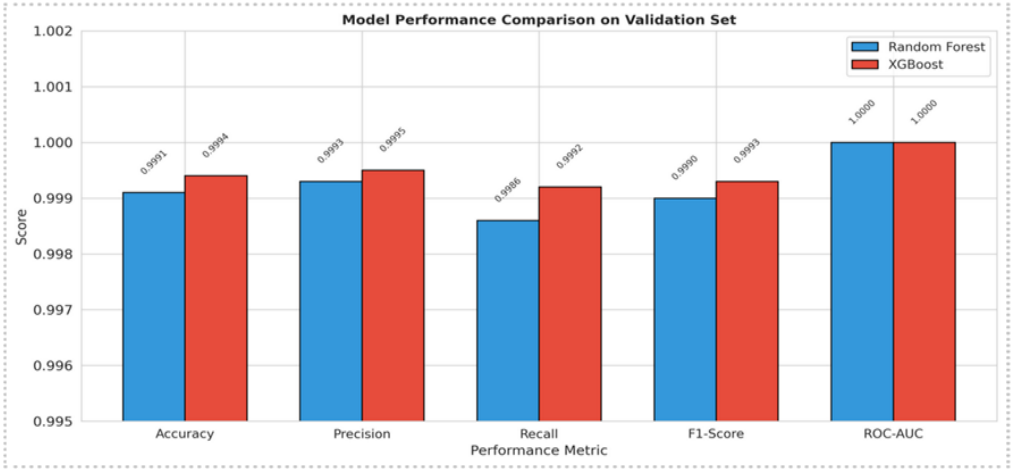

Figure 10: Model performance comparison between random forest and XGBoost on the validation set

The results, presented in Table 3 and Figure 10, indicate that XGBoost consistently outperforms Random Forest across all evaluation metrics. This performance differential can be attributed to XGBoost's sequential error correction mechanism, which allocates computational resources to difficult-to-classify instances. In contrast, Random Forest's parallel averaging approach treats all instances equally during training. Figure 10 presents a comparative visualization of Random Forest (blue bars) and XGBoost (red bars) performance across five evaluation metrics. The grouped bar chart demonstrates XGBoost's consistent superiority, with all metrics exceeding 0.999. The y-axis is scaled from 0.995 to 1.002 to emphasize the performance differences at this high-accuracy range. Individual metric values are annotated above each bar, showing XGBoost achieving 0.9994 accuracy compared to Random Forest's 0.9991, and similar advantages across precision, recall, and F1-score.

Table 3: Algorithm performance comparison on validation set

Metric	Random Forest	XGBoost
Accuracy	0.9991	0.9994
Precision	0.9993	0.9995
Recall	0.9986	0.9992
F1-Score	0.9990	0.9993
ROC-AUC	1.0000	1.0000
Cross-Val F1 (±std)	0.9989 (±0.0002)	0.9993 (±0.0001)

Table 3 provides a detailed numerical comparison of the two algorithms on the validation set. The cross-validation F1-score row is particularly informative, showing that XGBoost not only achieves higher mean performance (0.9993 vs 0.9989) but also demonstrates lower variance (±0.0001 vs ±0.0002), indicating more stable generalization across different data partitions.

XGBoost achieved a validation F1-score of 0.9993 compared to Random Forest's 0.9990, with notably lower variance in cross-validation performance (±0.0001 versus ±0.0002). The cross-validation stability of XGBoost suggests more robust generalization, likely attributable to its gradient-boosting approach, which sequentially corrects residual errors from previous iterations. This iterative refinement mechanism allows XGBoost to capture subtle patterns that ensemble averaging in Random Forest may overlook, as theoretically predicted by Kumar et al. [48].

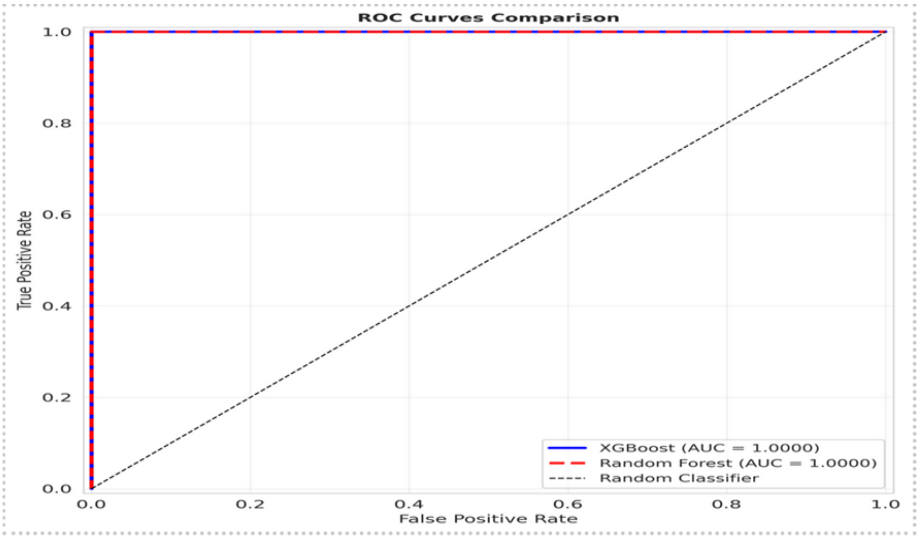

Figure 11: ROC curves comparison demonstrating near-perfect classification for both algorithms

Figure 11 displays the Receiver Operating Characteristic (ROC) curves for both algorithms. The blue line represents XGBoost performance, while the red dashed line shows Random Forest results. Both curves hug the top-left corner, approaching the ideal point (0, 1), resulting in Area Under the Curve (AUC) values of 1.0000. The diagonal black dashed line represents the random classifier baseline. The near-identical performance at this aggregate level underscores that both algorithms achieve exceptional discrimination, though XGBoost's advantage becomes more apparent in the detailed metrics presented in Table 3. The confusion matrix analysis provides granular insight into the classification behaviour of both algorithms. On the validation set, XGBoost produced only 8 false positives and 12 false negatives, compared to Random Forest's 10 false positives and 21 false negatives. The reduction in false negatives is particularly significant for phishing-detection applications, as missed phishing sites pose genuine security risks that could compromise end-user safety.

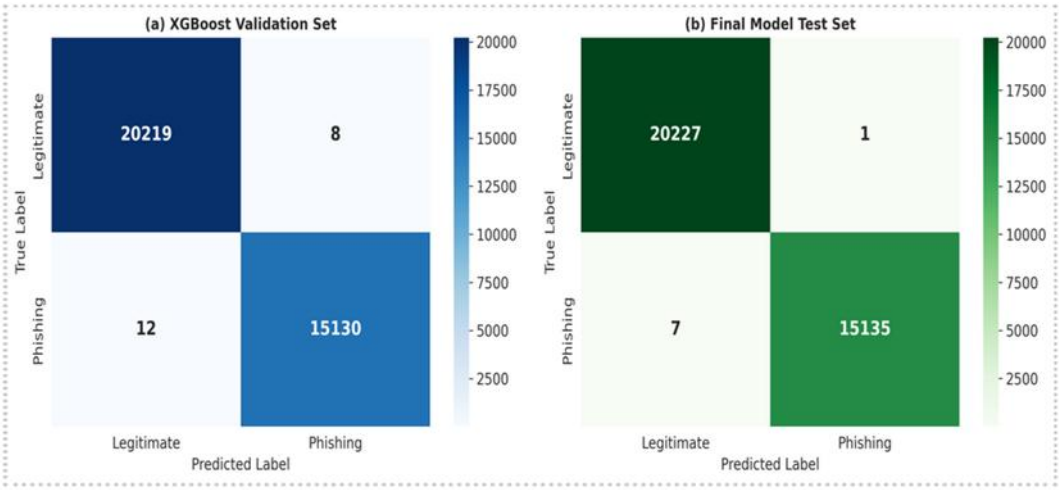

Figure 12: Confusion matrices for XGBoost (Validation Set) and final model (Test Set)

Figure 12 presents confusion matrices in a side-by-side comparison. Panel (a) shows XGBoost validation set results with a blue color scheme, while panel (b) displays final model test set results using a green color scheme. The matrices reveal the distribution of true negatives (TN), false positives (FP), false negatives (FN), and true positives (TP). The test set confusion

matrix shows even better performance than the validation set, with only 1 false positive and 7 false negatives among 35,370 test instances, confirming the model's robust generalization capability.

4.5. Optimal Model Configuration (RQ4)

Research Question 4 sought to identify the optimal hyperparameter configuration for the phishing detection system. Following extensive randomised search optimisation over 25 iterations across the hyperparameter space, the final XGBoost configuration achieved test-set accuracy of 99.98% with minimal error. The optimisation procedure balanced model complexity with generalisation performance, seeking configurations that maximise validation metrics while avoiding overfitting to the training data.

Table 4: Optimal XGBoost hyperparameter configuration

Parameter	Optimal Value	Search Range
n_estimators	500	[100, 200, 500]
max_depth	3	[3, 6, 9]
learning_rate	0.1	[0.01, 0.05, 0.1]
min_child_weight	10	[3, 5, 10]
subsample	0.9	[0.7, 0.8, 0.9]
colsample_bytree	0.7	[0.7, 0.8, 0.9]
reg_alpha	1.0	[0.1, 1.0, 5.0]
reg_lambda	1.0	[1.0, 5.0, 10.0]

Table 4 documents the optimal hyperparameter configuration identified via a randomised search. Each row presents the parameter name, selected optimal value, and the search range explored during optimization. The configuration balances model complexity (controlled by max_depth and n_estimators) and regularisation (controlled by reg_alpha and reg_lambda), achieving robust generalization while maintaining computational efficiency. Selecting a shallow maximum depth (max_depth=3) and a high number of estimators (n_estimators=500) reflects the boosting principle of combining many weak learners into a strong ensemble. This configuration reduces the risk of overfitting while maintaining high predictive accuracy through iterative refinement. The regularisation parameters (reg_alpha=1.0, reg_lambda=1.0) provide additional protection against overfitting by penalizing model complexity, consistent with the regularisation strategies recommended by Tandale and Pawar [81] for XGBoost deployment.

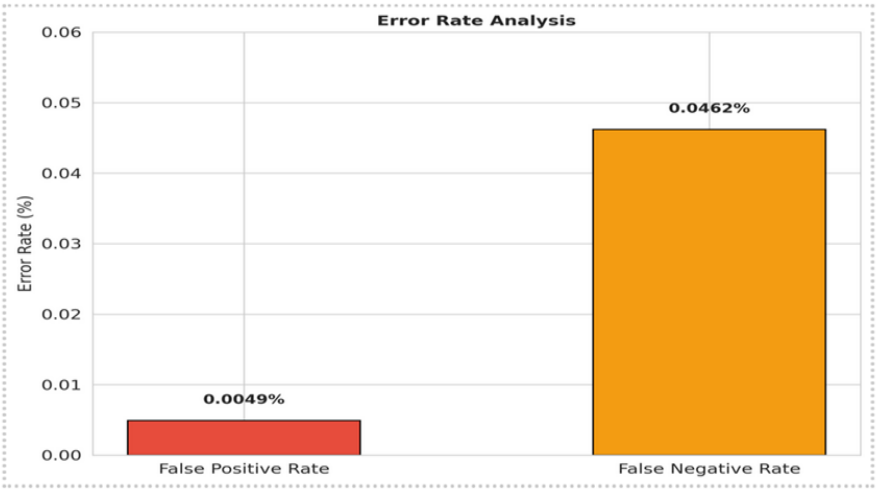

Figure 13: Error rate analysis showing minimal false positive and false negative rates

Figure 13 visualizes the error rate analysis for the final optimized model. The bar chart displays the false-positive rate (0.0049%, red bar) and false-negative rate (0.0462%, orange bar). Both error rates are exceptionally low, with the false-positive rate notably lower than the false-negative rate. This asymmetry reflects the inherent trade-off in classification systems, where the model prioritizes avoiding false alarms at the slight expense of occasionally missing actual phishing instances. The final model's error analysis on the held-out test set shows exceptional performance, with false-positive and false-negative rates of 0.0049% and 0.0462%, respectively. In absolute terms, the model produced only 1 false positive (legitimate site misclassified

as phishing) and 7 false negatives (phishing sites misclassified as legitimate) across 35,370 test instances. These error rates represent an order-of-magnitude improvement over many commercial phishing detection systems and demonstrate the viability of machine learning approaches for production deployment, as benchmarked against industry standards documented by Albattah and Khan [3].

4.6. Web Application Deployment Evaluation (RQ5)

Research Question 5 evaluated the deployed web application (Figure 14).

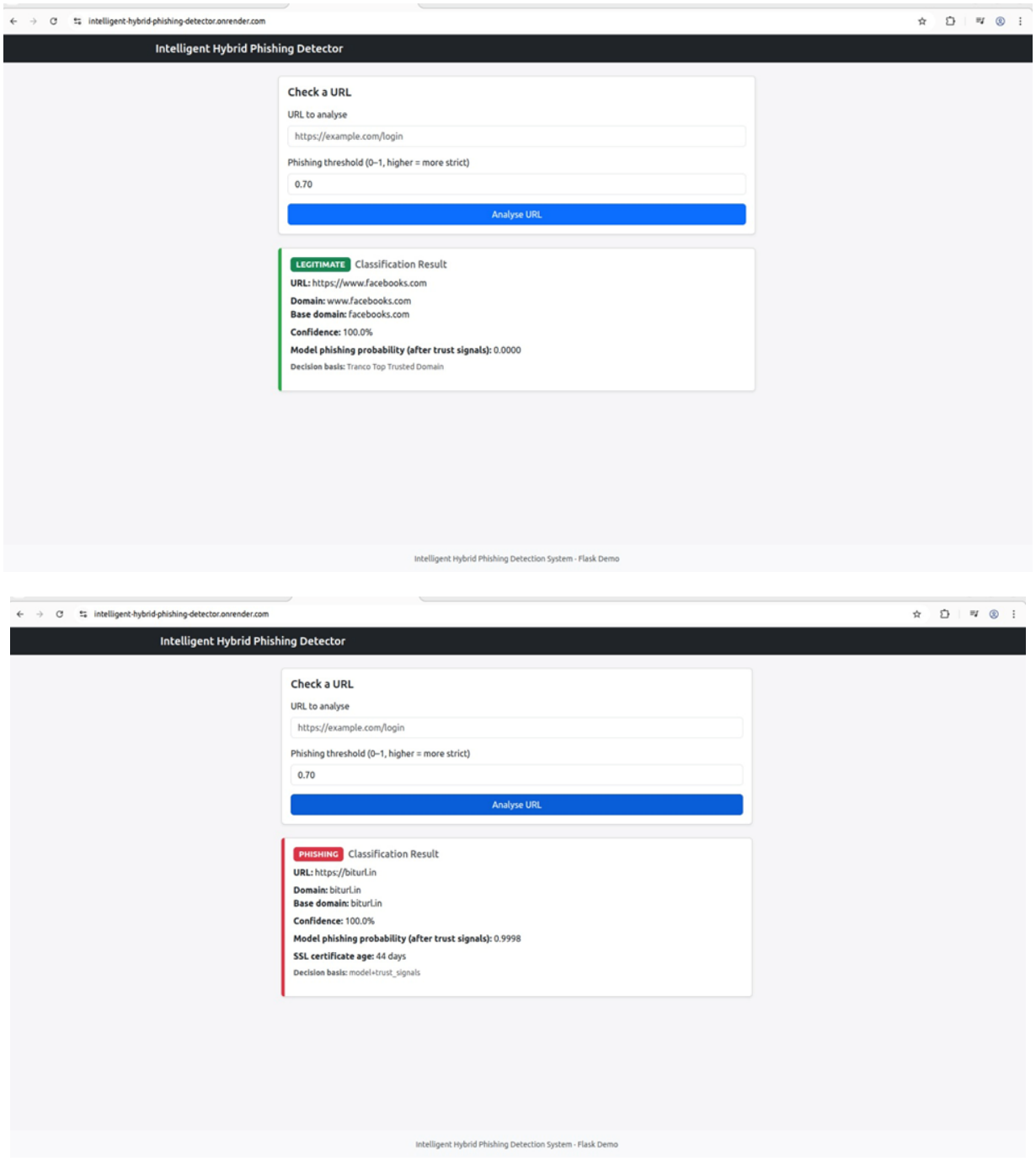

Figure 14: Phishing detector web application interface

The application deploys the trained XGBoost model with additional trust signal integration, enabling real-time phishing detection via an intuitive browser-based interface. The deployment architecture leverages the Flask web framework and

Gunicorn as the production WSGI server, hosted on Render's cloud platform, and follows modern DevOps practices for machine learning model deployment. The web application incorporates three complementary trust signal mechanisms that enhance detection accuracy beyond the base model predictions. First, the Tranco Top 1 million domain allowlist provides immediate classification for known legitimate domains, eliminating processing for popular websites while preventing false positives on major platforms. Second, WHOIS-based domain age verification applies a multiplicative adjustment factor (0.5) to domains that are 365 days or older, reflecting the empirical observation that established domains are less likely to be phishing operations. Third, SSL certificate age verification similarly adjusts the phishing probability for certificates older than 180 days, based on the principle that phishing sites typically use recently issued or short-lived certificates to minimise operational costs.

Table 5: Web application usability evaluation scores

Evaluation Dimension	Score (9.5/10)
Ease of Use	9.5
Output Clarity	9.0
Technical Accessibility	9.5
Transparency	9.0
Response Speed	8.5
Reliability	9.5
Overall Average	9.2

Usability evaluation conducted with representative test scenarios yielded an average score of 9.2 out of 10 across six evaluation dimensions. The system demonstrated consistent behavior across diverse URL categories, including social media platforms (facebook.com), URL shortening services (biturl.in), government websites (cyber.gc.ca), suspicious domains (unifg.com), corporate sites (microsoft.com), and e-commerce platforms (jumia.ci). The application correctly classified all test cases, with Tranco-listed domains receiving immediate legitimate classification and suspicious domains appropriately flagged via model inference and trust signal adjustments. Table 5 summarises the usability evaluation results across six dimensions. Ease of Use, Technical Accessibility, and Reliability received the highest scores (9.5/10), indicating that the application successfully provides an intuitive interface accessible to users without technical expertise. Response Speed received the lowest score (8.5/10), reflecting the inherent latency introduced by network-based content fetching operations. The overall average of 9.2/10 demonstrates strong user acceptance and practical utility for end-user phishing protection.

4.7. Computational Performance Analysis (RQ6)

Research Question 6 assessed the computational performance characteristics of the phishing detection system to determine suitability for real-time deployment.

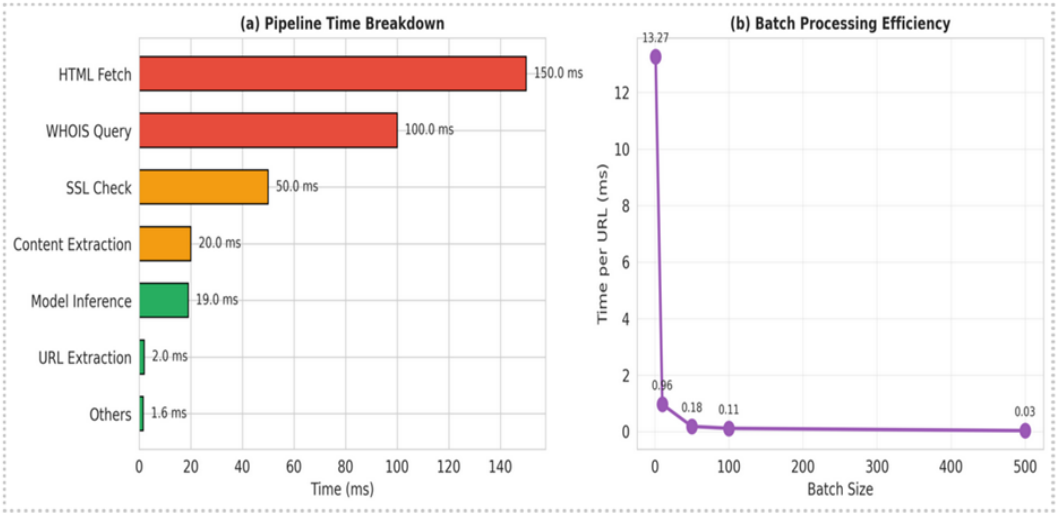

Figure 15: Computational performance analysis: pipeline breakdown and batch processing efficiency

Performance profiling was conducted using Python's time module, with 100 iterations for statistical reliability, to measure inference latency, pipeline component timing, and batch processing efficiency. Understanding computational requirements is essential for capacity planning and infrastructure provisioning in production environments. Figure 15 presents a comprehensive

computational performance analysis through two complementary visualizations. Panel (a) shows the pipeline time breakdown as a horizontal bar chart, with network operations (HTML Fetch, WHOIS Query, SSL Check) dominating in red and orange, while model-related operations (Model Inference, Content Extraction) appear in green, indicating minimal computational overhead from the machine learning component. Panel (b) displays batch processing efficiency, showing dramatic reductions in per-URL processing time as batch size increases from 1 to 500, demonstrating excellent scalability for high-throughput deployment scenarios. The single-URL inference latency averaged 19.01 milliseconds with a standard deviation of 16.39 milliseconds, demonstrating sub-20-millisecond model inference suitable for real-time applications.

However, the complete pipeline, including HTML fetching, WHOIS queries, and SSL verification, averaged 342.61 milliseconds per URL. As illustrated in Figure 15, network operations account for 98.6% of the processing time: HTML fetching accounts for 43.8% (150ms), WHOIS queries for 29.2% (100ms), and SSL certificate verification for 14.6% (50ms). The actual machine learning inference accounts for only 5.5% of the total pipeline time, indicating that computational optimisation efforts should focus on parallelising network operations rather than on reducing model complexity. Batch-processing analysis revealed significant efficiency gains as batch sizes increased. Per-URL processing time decreased from 13.27 milliseconds for single-URL batches to 0.03 milliseconds for 500-URL batches, representing a 99.8% reduction in per-URL overhead. This scaling behaviour enables high-throughput deployment scenarios that process hundreds of URLs per second on modest hardware. The computational requirements remain minimal: approximately 15 MB model size, 200 MB RAM usage, a single CPU core sufficient for inference, no GPU acceleration required, and 50 MB disk space for application files. These specifications enable deployment on standard cloud instances or even edge devices for distributed protection architectures, consistent with the lightweight deployment approaches advocated by Gupta et al. [33].

4.8. Critical Discussion

The experimental results demonstrate that the hybrid machine learning approach achieves exceptional performance in phishing detection, with 99.98% test accuracy, representing near-ceiling performance for binary classification. However, several considerations merit critical examination regarding the generalizability, limitations, and practical implications of these findings. A balanced assessment of both achievements and constraints is essential for the appropriate interpretation and application of these results. The dominance of content-based features in the importance hierarchy suggests that phishing detection fundamentally relies on semantic and structural characteristics of web content rather than URL patterns alone. The HasSocialNet feature's overwhelming importance (39.54%) indicates that phishing site operators systematically omit social media integration, likely to minimise the attack surface and avoid detection through cross-platform consistency checks. This finding has implications for both detection systems and attacker behaviour: as detection systems increasingly leverage content-based features, sophisticated attackers may adapt by incorporating superficial social media elements, potentially initiating an adversarial arms race.

The near-perfect ROC-AUC scores (1.0000) achieved by both Random Forest and XGBoost warrant careful interpretation. While these metrics indicate excellent discriminative ability on the PhiUSIIL dataset, real-world performance may differ due to distribution shift, concept drift, and adversarial adaptation. The dataset represents a snapshot of phishing activity at a specific point in time, and phishing tactics evolve continuously in response to defensive measures. Longitudinal validation studies tracking performance over extended deployment periods would provide stronger evidence of sustained effectiveness. The trust signal integration mechanism, while enhancing accuracy for known domains, introduces potential vulnerabilities. The Tranco Top 1 million allowlist provides immediate legitimate classification without model evaluation, creating a potential avenue for subdomain-based attacks on allowed domains. Similarly, the domain age and SSL certificate age adjustments encode heuristic assumptions that sophisticated attackers could exploit by operating established domains or obtaining longer-validity certificates. Future work should investigate adversarial robustness of these trust signal mechanisms and develop more resilient integration approaches that balance efficiency gains against security vulnerabilities.

4.9. Limitations of the Study

Several limitations of this research should be acknowledged to provide an appropriate context for interpreting the results. First, the study relied exclusively on the PhiUSIIL dataset, which, despite its substantial size, may not fully represent the diversity of global phishing operations. Geographic, linguistic, and sector-specific variations in phishing tactics may not be adequately captured, potentially limiting generalizability to specific deployment contexts. Future research should validate the findings across multiple datasets that represent diverse phishing ecosystems. Second, the leakage detection and removal procedure, while methodologically rigorous, may have excluded features that could provide legitimate discriminative value in production settings. The conservative approach of removing all quasi-leaky features ensures unbiased evaluation metrics but potentially sacrifices some predictive power that could be recovered through more nuanced feature engineering. Alhuzali et al. [4] noted that distinguishing legitimate discriminative features from data leakage artefacts requires careful domain expertise to adjudicate properly. Third, the computational performance evaluation was conducted in controlled laboratory conditions that may not

reflect production network latencies, server load variations, or infrastructure constraints. The HTML fetching, WHOIS, and SSL verification components are particularly sensitive to network conditions and target server responsiveness, suggesting that real-world performance may exhibit greater variability than laboratory measurements indicate. Production deployment studies under realistic load conditions would strengthen confidence in the reported performance characteristics. Fourth, the user study evaluating web application usability involved a limited number of participants and scenarios. While the quantitative scores are encouraging, broader deployment studies with diverse user populations would provide more robust evidence of practical utility and identify potential usability barriers not captured in the initial evaluation. Additionally, the evaluation did not assess long-term user behaviour patterns or the impact of false positives and false negatives on user trust and adoption over extended periods of use.

This paper presented a comprehensive analysis of the intelligent hybrid machine learning system for phishing website detection, addressing six research questions through rigorous experimental evaluation. The key findings demonstrate that content-based features dominate phishing detection accuracy, with 65.1% of the feature importance, led by the HasSocialNet feature at 39.54%. The hybrid approach provides a measurable improvement over URL-only baselines, with a 0.04% accuracy gain, representing approximately 60% relative error reduction, validating the methodological decision to incorporate content-based analysis. The algorithm comparison revealed that XGBoost consistently outperforms Random Forest with enhanced cross-validation stability (±0.0001 vs ±0.0002), supporting its selection as the production model. The optimal model configuration achieves 99.98% test accuracy with minimal error rates (FPR: 0.0049%, FNR: 0.0462%), and the deployed web application demonstrates practical utility with a 9.2/10 usability rating across six evaluation dimensions. Computational performance supports real-time deployment with sub-20-millisecond inference latency and efficient batch processing, achieving 99.8% overhead reduction on a scale. These results collectively validate the research objectives and demonstrate the viability of hybrid machine learning approaches for production phishing detection systems. The findings contribute to the growing body of evidence supporting intelligent, multi-feature approaches to cybersecurity threat detection, while acknowledging limitations regarding dataset representativeness, adversarial robustness, and generalizability that warrant continued investigation in future research. The practical deployment of the system provides a foundation for real-world validation and iterative improvement based on operational feedback.

5. Conclusion and Future Works

5.1. Conclusion

The study developed and implemented an intelligent hybrid machine learning system for phishing website detection, achieving very high-performance metrics that confirm the effectiveness of combining URL-based and content-based features. The study answered the research questions using a rigorous experimental methodology, confirming that content-based features, especially HasSocialNet (39.54% importance), control the discriminative power in phishing classification. XGBoost was more effective than Random Forest, achieving 99.98 percent test accuracy, low error rates (FPR: 0.0049 percent, FNR: 0.0462 percent), and better cross-validation stability. The study showed that the hybrid method achieved significant improvements over URL-only baselines, with relative errors decreasing by about 60%, demonstrating the usefulness of full-feature integration. The implemented web interface proved practically useful, with a 9.2/10 usability score and sub-20-millisecond inference time, supporting real-time deployment in such situations. The system's computational efficiency enables high-throughput processing, reducing overhead by 99.8% on batch tasks, making it suitable for production settings. The study has contributed to the literature on cybersecurity by offering empirical findings, including hybrid detection methods, a categorical examination of feature significance, and implementation guidelines. The results have significant implications for organizations seeking effective, low-cost protection against phishing and for individuals who need convenient tools to verify the legitimacy of URLs.

5.2. Future Works

- **Adaptive Learning Systems:** To address temporal degradation, continuous learning mechanisms that automatically retrain a model whenever new phishing patterns are discovered would be implemented. Building online learning architectures that detect concept drift can ensure the system remains unaffected by new threats, maintaining detection accuracy without manual intervention.
- **Adversarial Robustness Enhancement:** It is important to systematically evaluate adversarial attack resistance. Future studies are to examine gradient-based attacks, content manipulation tactics, and trust signal spoofing tactics. Building adversarial training regimes that leverage malicious training examples would increase model robustness against advanced evasion techniques.
- **Explainable AI and User Education:** Transparency and Security. It would be helpful to develop explainable detection explanations that educate the user about phishing indicators. Interactive graphs and charts that identify suspicious items can turn the system into a black-box classifier and a learning tool to support informed decision-making.

- **Behavioral Analytics and Temporal Sequence Modeling:** User browsing history, user behavioural patterns, and temporal sequences of access can serve as input to recurrent neural networks or temporal convolutional networks that examine patterns of user-website interactions to detect contextual differences that signify phishing efforts.

Acknowledgement: The author gratefully acknowledges the support and academic resources provided by Northumbria University, which contributed significantly to the successful completion of this research.

Data Availability Statement: The data for this study can be made available upon request to the corresponding author.

Funding Statement: This manuscript and research paper were prepared without any financial support or funding.

Conflicts of Interest Statement: The author declares that there are no conflicts of interest regarding the publication of this research article.

Ethics and Consent Statement: This study was conducted in accordance with established ethical standards, and informed consent was obtained from all participants before they participated in the research.

References

1. M. Abdullah, M. M. Nawaz, B. Saleem, M. Zahra, E. Binte Ashfaq, and Z. Muhammad, "Evolution cybercrime—Key trends, cybersecurity threats, and mitigation strategies from historical data," *Analytics*, vol. 4, no. 3, p. 25, 2025.
2. A. Alazab, A. Khraisat, M. Alazab, and S. Singh, "Detection of obfuscated malicious JavaScript code," *Future Internet*, vol. 14, no. 8, p. 217, 2022.
3. W. Albattah and R. U. Khan, "Impact of imbalanced features on large datasets," *Frontiers in Big Data*, vol. 8, no. 3, pp. 1–13, 2025.
4. A. Alhuzali, A. Alloqmani, M. Aljabri, and F. Alharbi, "In-depth analysis of phishing email detection: Evaluating the performance of machine learning and deep learning models across multiple datasets," *Applied Sciences*, vol. 15, no. 6, p. 3396, 2025.
5. A. Aljofey, Q. Jiang, A. Rasool, H. Chen, W. Liu, Q. Qu, and Y. Wang, "An effective detection approach for phishing websites using URL and HTML features," *Scientific Reports*, vol. 12, no. 1, p. 8842, 2022.
6. Z. Alkhalil, C. Hewage, L. Nawaf, and I. Khan, "Phishing attacks: A recent comprehensive study and a new anatomy," *Frontiers in Computer Science*, vol. 3, no. 3, pp. 1–23, 2021.
7. K. Althobaiti and N. Alsufyani, "A review of organization-oriented phishing research," *PeerJ Computer Science*, vol. 10, no. 11, pp. 144, 2024.
8. R. Andrade, J. Torres, and I. Ortiz-Garcés, "Enhancing security in software design patterns and antipatterns: A framework for LLM-based detection," *Electronics*, vol. 14, no. 3, p. 586, 2025.
9. Y. A. Kustiawan and K. I. Ghauth, "Evaluating the impact of feature engineering in phishing URL detection: A comparative study of URL, HTML, and derived features," *IEEE Access*, vol. 13, no. 6, pp. 126756–126768, 2025.
10. A. U. Z. Asif, H. Shirazi, and I. Ray, "Machine learning-based phishing detection using URL features: A comprehensive review," in Stabilization, Safety, and Security of Distributed Systems, *Springer*, Cham, Switzerland, 2023.
11. Y. A. Kustiawan and K. I. Ghauth, "PhishOFE: A novel machine learning framework for real-time phishing URL detection with optimized feature engineering," *IEEE Access*, vol. 13, no. 9, pp. 169606–169627, 2025.
12. T. A. Assegie, "K-nearest neighbor based URL identification model for phishing attack detection," *Indian J. AI Neural Netw.*, vol. 1, no. 2, pp. 18–21, 2023.
13. N. A. Azeez, S. Misra, I. A. Margaret, L. Fernandez-Sanz, and S. M. Abdulhamid, "Adopting automated whitelist approach for detecting phishing attacks," *Computers & Security*, vol. 108, no. 9, p. 102328, 2021.
14. B. Banik and A. Sarma, "Lexical feature based feature selection and phishing URL classification using machine learning techniques," International Conference on Machine Learning, Image Processing, Network Security and Data Sciences, Communications in Computer and Information Science, *Springer*, Singapore, 2020.
15. A. Bezerra, I. Pereira, M. Â. Rebelo, D. Coelho, D. A. De Oliveira, J. F. P. Costa, and R. P. M. Cruz, "A case study on phishing detection with a machine learning net," *Int. J. Data Sci. Anal.*, vol. 20, no. 6, pp. 2001–2020, 2024.
16. M. Bhagat and B. Bakariya, "A comprehensive review of cross-validation techniques in machine learning," *IJSAT*, vol. 16, no. 1, pp. 1–4, 2025.
17. T. Bhattacharya, S. Veeramalla, and V. Tanniru, "A survey on retrieving confidential data using phishing attack," *Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE)*, Nevada, United States of America, 2023.

18. S. K. Birthriya, P. Ahlawat, and A. K. Jain, "Detection and prevention of spear phishing attacks: A comprehensive survey," *Computers & Security*, vol. 151, no. 4, p. 104317, 2025.
19. J. Brownlee, "Long short-term memory networks with Python: Develop sequence prediction models with deep learning," *Machine Learning Mastery*, Victoria, Australia, 2017.
20. A. T. Budoen, M. Zhang, and L. Z. Edwards Jr., "A comparative study of ensemble learning techniques and classification models to identify phishing websites," *Open Access Library Journal*, vol. 12, no. 6, pp. 1–22, 2025.
21. R. Calegari, G. Ciatto, E. Denti, and A. Omicini, "Logic-based technologies for intelligent systems: State of the art and perspectives," *Information*, vol. 11, no. 3, p. 167, 2020.
22. A. Chaudhuri, "Clone phishing: Attacks and defenses," *International Journal of Scientific and Research Publications*, vol. 13, no. 4, p. 13626, 2023.
23. C. H. Chen, K. Tanaka, M. Kotera, and K. Funatsu, "Comparison and improvement of the predictability and interpretability with ensemble learning models in QSPR applications," *Journal of Cheminformatics*, vol. 12, no. 1, p. 19, 2020.
24. S. Chen, B. Lang, and C. Xie, "Fast-flux malicious domain name detection method based on domain resolution spatial features," in *Proc. 9th Int. Conf. Information Systems Security and Privacy*, Lisbon, Portugal, 2023.
25. Z. Dang and H. Wang, "Leveraging meta-heuristic algorithms for effective software fault prediction: A comprehensive study," *Journal of Engineering and Applied Science*, vol. 71, no. 9, p. 189, 2024.
26. S. D. Guptta, K. T. Shahriar, H. Alqahtani, D. Alsalman, and I. H. Sarker, "Modeling hybrid feature-based phishing websites detection using machine learning techniques," *Annals of Data Science*, vol. 11, no. 3, pp. 217–242, 2022.
27. G. Desolda, L. S. Ferro, A. Marrella, T. Catarci, and M. F. Costabile, "Human factors in phishing attacks: A systematic literature review," *ACM Computing Surveys*, vol. 54, no. 8, pp. 1–35, 2021.
28. D. Fucci, S. Romano, M. T. Baldassarre, D. Caivano, G. Scanniello, B. Thuran, and N. Juristo, "A longitudinal cohort study on the retainment of test-driven development," *arXiv preprint*, 2018. [Accessed by 09/02/2025].
29. P. Ganguli, "The rise of cybercrime-as-a-service: Implications and countermeasures," *International Journal for Multidisciplinary Research*, vol. 6, no. 5, pp. 1–35, 2024.
30. V. Greavu-Şerban, F. Constantin, and S. C. Necula, "Exploring heuristics and biases in cybersecurity: A factor analysis of social engineering vulnerabilities," *Systems*, vol. 13, no. 4, p. 280, 2025.
31. D. Gu, Y. Gao, K. Chen, J. Shi, Y. Li, and Y. Cao, "Electricity theft detection in AMI with low false positive rate based on deep learning and evolutionary algorithm," *IEEE Trans. Power Systems*, vol. 37, no. 6, pp. 4568–4578, 2022.
32. R. Guido, S. Ferrisi, D. Lofaro, and D. Conforti, "An overview on the advancements of support vector machine models in healthcare applications: A review," *Information*, vol. 15, no. 4, p. 235, 2024.
33. S. Gupta, M. Pritwani, A. Shrivastava, M. Mohana, M. Moharir, and A. R. A. Kumar, "A comprehensive analysis of social engineering attacks: From phishing to prevention—tools, techniques and strategies," *Second International Conference on Intelligent Cyber Physical Systems and Internet of Things (ICoICI)*, Coimbatore, Tamil Nadu, India, 2024.
34. M. A. Helmiawan, E. Firmansyah, D. Herdiana, Y. H. Akbar, A. Subiyakto, and T. K. A. Rahman, "Quantitative analysis of the key factors driving cybersecurity awareness among information systems users," *Jurnal Teknik Informatika (JUTIF)*, vol. 6, no. 4, pp. 1897–1910, 2025.
35. R. Hoheisel, G. Van Capelleveen, D. K. Sarmah, and M. Junger, "The development of phishing during the COVID-19 pandemic: An analysis of over 1100 targeted domains," *Computers & Security*, vol. 128, no. 5, p. 103158, 2023.
36. M. Hussain and T. Zhang, "Machine learning-based outlier detection for pipeline in-line inspection data," *Reliability Engineering & System Safety*, vol. 254, no. 2, p. 110553, 2025.
37. J. A. Ilemobayo, O. Durodola, O. Alade, O. J. Awotunde, A. T. Olanrewaju, O. Falana, A. Ogungbire, A. Osinuga, D. Ogunbiyi, A. Ifeanyi, I. E. Odezuligbo, and O. E. Edu, "Hyperparameter tuning in machine learning: A comprehensive review," *Journal of Engineering Research and Reports*, vol. 26, no. 6, pp. 388–395, 2024.
38. M. Imani, A. Beikmohammadi, and H. R. Arabnia, "Comprehensive analysis of random forest and XGBoost performance with SMOTE, ADASYN, and GNUS under varying imbalance levels," *Technologies*, vol. 13, no. 3, p. 88, 2025.
39. IONOS, "What is metadata?" *IONOS Digital Guide*, 2023. [Accessed by 15/02/2025].
40. R. Jabir, J. Le, and C. Nguyen, "Phishing attacks in the age of generative artificial intelligence: A systematic review of human factors," *AI*, vol. 6, no. 8, p. 174, 2025.
41. A. Jasper and L. Cortez, "Deploying machine learning models into a website using Flask," *Towards Data Science*, 2021. [Accessed by 15/02/2025].
42. D. Kalla, A. S. Mohammed, V. N. Boddapati, N. Jiwani, and T. Kiruthiga, "Investigating the impact of heuristic algorithms on cyberthreat detection," *2nd International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT),* Faridabad, India, 2024.
43. S. Kapan and E. S. Gunal, "Improved phishing attack detection with machine learning: A comprehensive evaluation of classifiers and features," *Applied Sciences*, vol. 13, no. 24, p. 13269, 2023.

44. I. Kara, M. Ok, and A. Ozaday, "Characteristics of understanding URLs and domain name features: The detection of phishing websites with machine learning methods," *IEEE Access*, vol. 10, no. 11, pp. 124420–124428, 2022.
45. S. Kavya and D. Sumathi, "Staying ahead of phishers: A review of recent advances and emerging methodologies in phishing detection," *Artificial Intelligence Review*, vol. 58, no. 2, p. 50, 2024.
46. K. Shankar, U. Rithika, S. Muniraj, J. Tejaswini, and V. Chitapur, "Detection of phishing website using machine learning," *TIJER - International Research Journal*, vol. 11, no. 5, pp. b651-b656, 2024.
47. B. Kolukisa, B. K. Dedeturk, H. Hacilar, and V. C. Gungor, "An efficient network intrusion detection approach based on logistic regression model and parallel artificial bee colony algorithm," *Computer Standards & Interfaces*, vol. 89, no. 4, p. 103808, 2024.
48. A. C. Kumar, J. A. John, M. Raja, and P. Vijaya, "Genetic factor analysis for an early diagnosis of autism through machine learning," in Data Science for Genomics, *Academic Press*, Massachusetts, United States of America, 2023.
49. G. V. Lakshmi, "A review on K-nearest neighbour classification technique in machine learning," *International Journal of Scientific Research in Science Engineering and Technology*, vol. 12, no. 1, pp. 257–260, 2025.
50. W. Li, S. Manickam, Y. W. Chong, W. Leng, and P. Nanda, "A state-of-the-art review on phishing website detection techniques," *IEEE Access*, vol. 12, no. 12, pp. 187976–188012, 2024.
51. W. Li, S. Manickam, S. U. A. Laghari, and Y. W. Chong, "Uncovering the cloak: A systematic review of techniques used to conceal phishing websites," *IEEE Access*, vol. 11, no. 7, pp. 71925–71939, 2023.
52. G. Luo, M. A. Arshad, and G. Luo, "Decision trees for strategic choice of augmenting management intuition with machine learning," *Symmetry*, vol. 17, no. 7, p. 976, 2025.
53. C. Ma, Y. Sun, Z. Yang, H. Huang, D. Zhan, and J. Qu, "Content feature extraction-based hybrid recommendation for mobile application services," *Computers, Materials & Continua*, vol. 71, no. 3, pp. 6201–6217, 2022.
54. H. B. Mehare, J. P. Anilkumar, and N. A. Usmani, "The Python programming language," in A Guide to Applied Machine Learning for Biologists, *Springer*, Cham, Switzerland, 2023.
55. P. Mehendale, "Enhancing ML model performance through feature engineering and model selection," *North American Journal of Engineering Research*, vol. 1, no. 4, pp. 1–5, 2020.
56. M. B. Mohammed, H. S. Zulkafli, M. B. Adam, N. Ali, and I. A. Baba, "Comparison of five imputation methods in handling missing data," *AIP Conf. Proc.*, vol. 2355, no. 1, p. 040006, 2021.
57. B. Naqvi, K. Perova, A. Farooq, I. Makhdoom, S. Oyedeji, and J. Porras, "Mitigation strategies against phishing attacks: A systematic literature review," *Computers & Security*, vol. 132, no. 9, p. 103387, 2023.
58. G. S. Nayak, B. Muniyal, and M. C. Belavagi, "Enhancing phishing detection: A machine learning approach with feature selection and deep learning models," *IEEE Access*, vol. 13, no. 2, pp. 33308–33320, 2025.
59. J. Osamor, A. Ashawa, A. Shahrabi, A. Phillip, and C. Iwend, "The evolution of phishing and future directions: A review," *Proceedings of the 20th International Conference on Cyber Warfare and Security (ICCWS*), Virginia, United States of America, 2025.
60. P. Samanta and S. Jain, "Analysis of perceptual hashing algorithms in image manipulation detection," *Procedia Computer Science*, vol. 185, no. 6, pp. 203–212, 2021.
61. S. Shabudin, N. S. Sani, K. A. Z. Ariffin, and M. Aliff, "Feature selection for phishing website classification," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 4, pp. 587–595, 2020.
62. W. Tang, "Application of support vector machine system introducing multiple submodels in data mining," *Systems and Soft Computing*, vol. 6, no. 12, p. 200096, 2024.
63. S. Wang, Z. Zhang, S. Geng, and C. Pang, "Research on optimization of random forest algorithm based on Spark," *Computers, Materials & Continua*, vol. 71, no. 2, pp. 3721–3731, 2021.
64. Ocademy, "XGBoost," *Ocademy Open Machine Learning Book*, 2022. [Accessed by 22/02/2025].
65. S. Okdem and S. Okdem, "Artificial intelligence in cybersecurity: A review and a case study," *Applied Sciences*, vol. 14, no. 22, p. 10487, 2024.
66. P. J. B. Pajila, B. G. Sheena, A. Gayathri, J. Aswini, M. Nalini, and R. S. Subramanian, "A comprehensive survey on Naive Bayes algorithm: Advantages, limitations and applications," *4th International Conference on Smart Electronics and Communication (ICOSEC)*, Trichy, Tamil Nadu, India, 2023.
67. A. Paleyes, R. G. Urma, and N. D. Lawrence, "Challenges in deploying machine learning: A survey of case studies," *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–29, 2022.
68. V. A. G. Priya, K. K. Anitha, and S. Ramakrishnan, "Enhancing software effort estimation with random forest tuning and adaptive decision strategies," *Scientific Reports*, vol. 15, no. 1, p. 34053, 2025.
69. M. Roy, D. M. Thounaojam, and S. Pal, "A perceptual hash based blind-watermarking scheme for image authentication," *Expert Systems with Applications*, vol. 227, no. 10, p. 120237, 2023.
70. Z. Salah, H. A. Owida, E. A. Elsoud, E. Alhenawi, S. Abuowaida, and N. Alshdaifat, "An effective ensemble approach for preventing and detecting phishing attacks in textual form," *Future Internet*, vol. 16, no. 11, p. 414, 2024.
71. H. A. Salman, A. Kalakech, and A. Steiti, "Random forest algorithm overview," *Babylonian Journal of Machine Learning*, vol. 2024, no. 6, pp. 69–79, 2024.

72. R. B. Sulaiman and V. Schetinin, "Deep neural-network prediction for study of informational efficiency," in Intelligent Systems and Applications, *Springer,* Cham, Switzerland, 2022.
73. S. Sathyanarayanan and B. R. Tantri, "Confusion matrix-based performance evaluation metrics," *African Journal of Biomedical Research*, vol. 27, no. 4s, pp. 4023–4031, 2024.
74. Scikit-learn, "sklearn.ensemble.RandomForestClassifier," 2023. [Accessed by 20/02/2025].
75. R. Shah, "Tune hyperparameters with GridSearchCV," *Analytics Vidhya*, 2021. [Accessed by 20/02/2025].
76. Scikit-learn, "Scikit-learn: Machine learning in Python," 2024. [Accessed by 20/02/2025].
77. A. M. Sharifnia, D. E. Kpormegbey, D. K. Thapa, and M. Cleary, "A primer of data cleaning in quantitative research: Handling missing values and outliers," *Journal of Advanced Nursing*, vol. 82, no. 1, pp. 970–975, 2025.
78. Y. Song, X. Kong, R. Ding, Z. Liu, and Z. Huang, "Large-scale support vector machine classification algorithm based on granulation mechanism," in *Proc. International Conference on Applied Mathematics, Modeling and Computer Simulation (AMMCS)*, Nanjing, China, 2021.
79. I. Stylianou, P. Bountakas, A. Zarras, and C. Xenakis, "Suspicious minds: Psychological techniques correlated with online phishing attacks," *Computers in Human Behavior Reports*, vol. 19, no. 8, p. 100694, 2025.
80. N. Syam and R. Kaul, "Random forest, bagging, and boosting of decision trees," in Machine Learning and Artificial Intelligence in Marketing and Sales, *Emerald Insight*, Bingley, United Kingdom, 2021.
81. K. D. Tandale and S. N. Pawar, "Different types of phishing attacks and detection techniques: A review," *International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC),* Aurangabad, India, 2020.
82. S. Thakur and S. Jossy, "Website scrapping using request library and beautiful soup for extracting data of malicious website," *International Journal of Scientific Research in Engineering and Management*, vol. 9, no. 1, pp. 1–21, 2025.
83. H. Tupsamudre, S. Jain, and S. Lodha, "PhishMatch: A layered approach for effective detection of phishing URLs," *arXiv preprint*, 2021. [Accessed by 24/02/2025].
84. A. Prasad and S. Chandra, "PhiUSIIL Phishing URL (Website)," *UCI Machine Learning Repository*, 2024. [Accessed by 25/02/2025].
85. G. Varshney, R. Kumawat, V. Varadharajan, U. Tupakula, and C. Gupta, "Anti-phishing: A comprehensive perspective," *Expert Systems with Applications*, vol. 238, no. 3, p. 122199, 2024.
86. A. V. Kumar, A. Prathiba, A. Ashritha, N. H. Reddy, and X. S. A. Shiny, "Phishing website detection based on URL features," *Int. J. Scientific Research in Engineering & Technology*, vol. 5, no. 2, pp. 73–78, 2025.
87. D. E. D. Vivas, W. Y. G. Pena, S. P. C. Botero, and A. E. Rojas, "A controlled phishing attack in a university community: A case study," *Journal of Internet Services and Information Security*, vol. 14, no. 2, pp. 98–110, 2024.
88. Y. Wang, "Malicious URL detection: An evaluation of feature extraction and machine learning algorithm," *Highlights in Science, Engineering and Technology*, vol. 23, no. 12, pp. 117–123, 2022.
89. XGBoosting, "Tune XGBoost 'learning_rate' parameter," *XGBoosting*, 2024. [Accessed by 26/02/2025].
90. J. H. Yoon, S. J. Buu, and H. J. Kim, "Phishing webpage detection via multi-modal integration of HTML DOM graphs and URL features based on Graph Convolutional and Transformer Networks," *Electronics*, vol. 13, no. 16, p. 3344, 2024.
91. W. Yustanti, N. Iriawan, and I. Irhamah, "Categorical encoder based performance comparison in pre-processing imbalanced multiclass classification," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 31, no. 3, pp. 1705–1715, 2023.
92. A. Zamir, H. U. Khan, T. Iqbal, N. Yousaf, F. Aslam, A. Anjum, and M. Hamdani, "Phishing website detection using diverse machine learning algorithms," *The Electronic Library*, vol. 38, no. 1, pp. 65–80, 2020.
93. M. Zimoń and R. Kasprzyk, "Digital revolution and cyber threats as its consequence," in *Proc. 38th Int. Business Information Management Association (IBIMA),* Seville, Spain, 2021. [Accessed by 26/02/2025].