A Hybrid Model for the Detection of Phishing URLs Based on Machine Learning

By

Christone Odhiambo Oduor

171837

Submitted in Partial Fulfilment of the Requirements for the Degree of Master of Science in Information Technology at Strathmore University

School of Computing & Engineering Sciences

Strathmore University

Nairobi, Kenya

June, 2025

Declaration and Approval

Declaration

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

Name: Christone Oduor Odhiambo

Signature ……

Date …20th April 2025……

Approval

The thesis of Christone Oduor Odhiambo was reviewed and approved for examination by the following:

Dr. Vitalis Ozianyi,

Lecturer, School of Computing & Engineering Sciences,

Strathmore University

Eng. Dr. Julius Butime,

Dean, School of Computing and Engineering Sciences,

Strathmore University

Prof. Bernard Shibwabo,

Director of Graduate Studies,

Strathmore University

Abstract

Phishing attacks have become increasingly prevalent and sophisticated, posing a significant threat to online security for both individuals and organizations. Traditional phishing detection methods, such as blacklists and heuristic-based systems, are often ineffective against modern phishing tactics due to their inability to adapt to the evolving nature of these attacks. Machine Learning (ML) can be applied to process signature databases of phishing URLs to enhance the effectiveness of blacklists. This research explores the potential of using ML to enhance phishing URL detection. It proposes a hybrid model that will utilise a combination of two or more of the following algorithms: Decision trees which work by creating a model that splits the dataset into branches based on feature values, such as URL length, presence of suspicious keywords, or domain age, each internal node represents a decision based on a feature, and each leaf node represents a classification as either Legitimate or phishing; XGBoost which implements an ensemble learning approach to classify URLs by combining multiple decision trees, it processes features like URL length and character composition and improves accuracy through gradient boosting; Convolutional Neural Networks which work by analysing visual representations of URLs and website content, by extracting features such as character composition from these visual inputs, CNNs can identify patterns indicative of phishing sites and Support Vector Machines which functions by finding the optimal hyperplane that separates phishing from legitimate sites based on extracted features. The developed solution will utilise the PhishTank database, which provides a community-driven repository of known phishing URLs, to analyse historical data and identify common patterns in previously flagged URLs. The expected outcome is a more precise and comprehensive blacklist of phishing URLs. This approach strengthens existing blacklists and facilitates proactive identification of emerging phishing threats based on evolving patterns. The solution also implements a feedback mechanism for model retraining, where misclassified URLs are identified and used to update the training dataset. The outcome will be improved detection accuracy and adaptability of the solution. The study will use a modified Agile development methodology with five stages: Planning, Design, Development, Testing, and Review/Feedback.

Table of Contents

List of Figures

List of Tables

List of Abbreviations/ Acronyms

AI	Artificial Intelligence.
APWG	Anti-Phishing Working Group.
AUC	Area Under the Curve.
CNN	Convocational Neural Networks.
DL	Deep Learning.
DNS	Domain Name System.
DoS	Denial of Service.
DT	Decision Tree.
DARTH	Detection of Attacks through Research and Technology for Humans
FBI	Federal Bureau of Investigation.
GPU	Graphics Processing Unit.
KNN	K-Nearest Neighbors.
RFECV	Recursive Feature Elimination with Cross-Validation.
LR	Linear regression.
LSD	logistic regression, support vector machine and decision tree
LSTM	Long Short-Serm Memory Network.
ML	Machine Learning.
MLP	Multilayer Perceptron.
NB	Naive Bayes.
NLP	Natural Language Processing.
OFVA	Optimal Feature Vectorization Algorithm.
RBF	Radial Basis Function.
RF	Random Forest.
RNN	Recurrent Neural Networks.
ROC	Receiver Operating Characteristics
SML	Supervised Machine Learning.
SMS	Short Message Service.
SVC	Support Vector Classifier.
SVM	Support Vector Machine.
URL	Uniform Resource Locator.

XGBoost Extreme Gradient Boosting.

Acknowledgement

I would like to acknowledge God almighty for providing me with life to enable me to complete my project, my parents Mr. Chrispine Oduor and Mrs. Grace Anyango Oduor for funding my education and providing me with the necessary moral and physical support throughout the duration of my project, my lecturers for providing me with the necessary guidance to undertake the project and Strathmore university for providing me with the necessary resources to develop and complete my project.

Dedication

I would like to dedicate this project to my family, friends, and to all those who aided me in one way or another to complete my project

Chapter 1: **Introduction**

1.1 Background of the Study

Phishing attacks have become an increasingly prevalent and sophisticated threat in the digital age, posing a significant challenge to individuals and organizations a like. There are various types of phishing attacks that cybercriminals employ to deceive victims. These include smishing, which involves phishing attempts through text messages, vishing, where attackers use phone calls to impersonate legitimate organizations, search engine phishing, where malicious sites are designed to rank highly in search results, and the more well-known email phishing. These different types of phishing attacks leverage a range of techniques to exploit victims, such as impersonating trusted entities, using social engineering tactics to create a false sense of urgency or authority, and exploiting vulnerabilities across various communication channels like email, SMS, and voice calls. The rapid growth of phishing activities and the continuously evolving tactics employed by cybercriminals underscore the urgent need for robust and reliable detection mechanisms to safeguard internet users and protect against these malicious, ever-changing practices. (Aljofey et al., 2020) (Rendall et al., 2020)

Phishing URL detection is a critical aspect of cybersecurity that involves analysing various features of URLs to determine their legitimacy and identify potential malicious intent. This process employs a combination of techniques focusing on Traditional methods i.e. Blacklisting, URL-based, content-based, and behavioural features, as well as machine learning models.

Traditional methods for detecting phishing and malicious activities encompass blacklisting, rule-based systems and URL filtering, each with unique mechanisms and inherent limitations. Blacklisting involves maintaining a list of known malicious URLs. When a user attempts to access a URL, the system checks this list to determine if the URL is harmful. Although effective for known threats, blacklisting struggles to detect new or evolving phishing tactics, as attackers often change their URLs to avoid detection, leading to potential vulnerabilities. Rule-based systems use explicit, predefined rules to classify URLs based on their characteristics. They typically rely on static decision trees, where each branch represents a rule derived from specific attributes, such as checking for keywords commonly associated with phishing. These systems require manual updates to adapt to emerging threats and can become cumbersome as the number of rules increases, hindering their effectiveness. URL filtering is another method that organizations utilize to block access to known malicious URLs by evaluating their content,

structure, and context. Filters may deny access to URLs containing certain keywords or patterns and often rely on external reputation services to assess URL safety based on historical data. However, URL filtering can result in false positives, where legitimate sites are blocked, and may fail to detect new threats that have not yet been catalogued.

Another method in phishing detection is the analysis of URL-based features. This approach examines the structural components of a URL, including its domain, path, and query parameters, to identify patterns or anomalies that may indicate phishing attempts. For instance, phishing URLs often use deceptive domain names that closely resemble legitimate ones. A common example is a phishing URL like https://www.paypal-secure.com, which may mislead users into thinking it is the official PayPal site, whereas a legitimate URL would be https://www.paypal.com. Additionally, analysing the path structure can reveal suspicious patterns; a legitimate banking site might use a simple path like /login, while a phishing site could employ a more complex path such as /secure/login.php?id=12345. Such deviations from expected patterns can serve as red flags for potential phishing attempts.

In addition to URL-based features, content-based feature analysis plays a crucial role in detecting phishing URLs. This method examines the webpage linked to the URL for signs of deception or impersonation. For example, by analysing the textual content on a webpage, algorithms can detect common phishing phrases or misleading information. A phishing page might contain urgent language such as "Your account will be suspended unless you verify your information now!" designed to create panic among users. Furthermore, visual elements such as layout and design can also indicate phishing attempts. A legitimate website typically maintains consistent branding and professional design elements, while a phishing site may feature poor-quality images or inconsistent styles. For instance, if a webpage mimics the layout of a bank's site but uses low-resolution images or different colour schemes, it may be flagged as suspicious.

Behavioural features are another important aspect of phishing detection. This technique involves monitoring user interactions with URLs to detect suspicious activities. Abnormal user behaviour can serve as an indicator of phishing attempts. For example, if a user clicks on a link in an email and immediately fills out sensitive information without hesitation, this behaviour could be flagged as potentially dangerous. Additionally, monitoring how long users spend on a page or their mouse movements can provide insights into their intentions; erratic mouse movements or very short dwell times on potentially malicious pages could suggest that the user is being misled.

Finally, machine learning models are employed to analyse large datasets of URLs and classify them as either legitimate or phishing. Techniques such as logistic regression can identify relationships between features extracted from URLs and their classification. For instance, logistic regression may learn that longer URLs containing certain keywords are more likely to be associated with phishing attempts. Random forests utilize multiple decision trees to improve classification accuracy; if several trees classify a URL as phishing based on its features—such as length and suspicious keywords, the final decision will likely reflect this consensus. Deep neural networks can also learn complex patterns from raw data; for example, Convolutional Neural Networks (CNNs) can analyse both textual content and visual elements of webpages for detecting phishing attempts (Wei et al., 2023).

Machine learning has emerged as a promising approach for addressing the phishing detection problem. Traditional methods, such as blacklisting, have proven to be limited in their effectiveness. Blacklisting involves maintaining a list of known phishing URLs, which can be quickly outdated as cybercriminals continuously create new phishing sites. Heuristic-based techniques rely on a set of predefined rules to identify suspicious URL patterns, but these rules can be easily bypassed by attackers through techniques like URL obfuscation and mimicking legitimate website structures. Machine learning, on the other hand, offers the potential to adaptively learn the patterns and characteristics of phishing URLs, enabling more accurate and real-time detection. (Odeh et al., 2021).

Recent research has explored various machine learning models for phishing detection, including traditional algorithms and deep learning techniques. These models have shown promising results in terms of accuracy, with some reaching over 95% in their detection capabilities. (Aljofey et al., 2020). However, there is still room for improvement, hybrid models, particularly in terms of balancing the trade-off between false-positive and false-negative rates and enhancing the ability to generalize to new and evolving phishing tactics. Hybrid models can address these challenges by leveraging the complementary strengths of different machine learning algorithms and feature engineering techniques.

This this paper proposes a hybrid model for the detection of phishing URLs based on machine learning. The model aims to leverage the strengths of multiple machine learning algorithms i.e Convolutional Neural Networks (CNNs) excel at learning complex patterns directly from raw data, achieving high accuracy rates, often exceeding 99%, which makes them particularly effective in identifying subtle characteristics of phishing URLs. Random Forests are robust

ensemble methods that handle imbalanced datasets well and provide valuable insights into feature importance, aiding interpretability. Gradient Boosting algorithms, such as XGBoost, improve predictive accuracy by sequentially correcting errors from previous models, making them adept at capturing intricate patterns in phishing data. Support Vector Machines (SVMs) are powerful classifiers that excel in high-dimensional spaces and can effectively separate classes using non-linear decision boundaries. Finally, K-Nearest Neighbors (KNN) offers simplicity and adaptability, allowing for quick classification based on proximity to known data points, which can serve as a useful complement to more complex algorithms in a hybrid approach.

A potential combination for the hybrid model could involve integrating CNNs with Random Forests and Gradient Boosting. This combination would allow the model to leverage the CNN's ability to extract complex features from URLs while utilizing Random Forests and Gradient Boosting to improve classification accuracy and manage imbalanced data effectively. Additionally, the proposed model would include a feedback loop where misclassified URLs (false positives and false negatives) are collected and used to update the training dataset to improve detection accuracy or Active Learning: In active learning, the model identifies uncertain predictions and requests additional labelling from human experts enabling continuous learning and adaptation to the evolving phishing tactics, ultimately providing a more robust and reliable solution for combating the ever-changing phishing threat.

Selecting algorithms for the hybrid model will be guided by the following metrics. First and foremost, performance metrics such as accuracy, precision, recall, and F1 score will play a crucial role in evaluating the effectiveness of the algorithms. Additionally, the ability of the algorithms to handle and select relevant features from the dataset is essential; this ensures that the model focuses on the most informative aspects of the data. Speed of detection is another critical factor, as the algorithms must be capable of processing and identifying phishing attempts quickly to mitigate threats in real time. Finally, adaptability to new phishing techniques is vital; the chosen algorithms should be able to evolve and respond to emerging tactics used by cybercriminals.

To demonstrate the potential of hybrid approaches in phishing detection, a recent study by Gurung et al. (2023) combined a CNN with a Random Forest classifier to develop a hybrid CNN-LSTM-Random Forest model, which achieved an impressive accuracy of 98.99% in detecting phishing URLs.

This hybrid model utilized character sequence features extracted by the CNN and combined them with statistical features processed by the Random Forest classifier. The integration of LSTM allowed the model to capture temporal patterns, further improving its detection capabilities. Specifically, the CNN component was responsible for automatically extracting features from the URL structure, while the Random Forest component provided a robust and interpretable classification mechanism.

1.2 Problem Statement

According to research by Das Guptta et al. (2022), phishing attacks remain a critical and growing threat to online security, as cybercriminals continue to exploit users through the use of deceptive URLs. Existing phishing detection methods often lack the necessary adaptability and accuracy required to effectively combat the evolving and increasingly sophisticated phishing techniques employed by attackers. While traditional rule-based approaches have had limited success, the need for more robust and adaptive solutions, such as those leveraging machine learning, is becoming increasingly pressing to address this persistent cybersecurity challenge.

1.3 Research Objectives

I. To develop and evaluate the performance of individual machine learning algorithms in detecting complex and evolving phishing attacks

II. To evaluate the effectiveness of combining various algorithms to achieve higher accuracy.

III. To develop a hybrid model capable of achieving higher accuracy than their individual counterparts.

IV. To validate the model.

1.4 Research Questions

I. What is the performance of current machine learning algorithms in detecting phishing URLs compared to traditional rule-based methods?

II. How effective are different combinations of machine learning algorithms in achieving higher accuracy for detecting phishing URLs?

III. What combination of machine learning techniques can be used to design a hybrid model that improves the accuracy of phishing URL detection?

IV. How do the comparative results of the hybrid model's accuracy and efficiency measure against different individual algorithms in detecting phishing URLs?

1.5 Justification

Phishing attacks remain a significant cybersecurity threat, exploiting users through deceptive URLs. As outlined by Das Guptta et al. (2022), these attacks are growing more frequent, increasing in complexity, and are becoming much harder to mitigate using traditional mechanisms. The necessity of an improved detection tools is justified by the fact that phishing remains one of the most effective and common cybercrime techniques, boasting of high effectiveness in user impersonation and information stealing. Phishing scams account for nearly 22% of all data breaches that occur thus securing it a position as one of the most prevalent cybercrimes in the FBI's 2021 IC3 Report (Palatty, 2024).

1.6 Scope, Limitations and Assumptions

1.6.1 Scope

The scope of this research is limited to the development and evaluation of a hybrid model for the detection of phishing URLs. The proposed model will focus on combining multiple machine learning algorithms and feature engineering techniques to achieve higher accuracy in detecting phishing URLs, which are one of the primary vectors for phishing attacks.

1.6.2 Limitation

While the proposed hybrid model will aim to be robust and adaptive in detecting phishing URLs, it is important to note that this research is limited to the detection of phishing URLs specifically. The model may not address other aspects of phishing, such as email-based phishing attacks or social engineering tactics, which can also pose significant security threats. The focus of this work will be on leveraging multiple machine learning algorithms and feature engineering techniques to enhance the accuracy and reliability of phishing URL detection, providing a more comprehensive solution for combating the phishing threat. However, the broader challenges posed by phishing, including the use of multiple attack channels and social engineering aspects, may require further research and the development of complementary detection and mitigation strategies.

The model's evaluation is based on a static dataset, which represents a snapshot in time. URL characteristics, attacker behavior, and benign site structures are dynamic and subject to change.

Consequently, a static dataset may not capture the temporal evolution of threats, which limits the longevity and adaptability of the model.

The study relies solely on lexical features, excluding potentially rich context from host-based or content-based attributes (e.g., IP addresses, domain reputation, website content). This narrow feature scope may result in lower resilience against adversarial URLs that appear lexically benign but are malicious based on external data.

1.6.3 Assumptions

A foundational assumption of the research is that malicious and benign URLs exhibit distinguishable patterns within their lexical structure. However, attackers continuously adapt by crafting URLs that closely mimic benign structures or using obfuscation techniques such as URL shortening, encoding, or misleading domain names. As a result, models trained on specific datasets may not generalize well to newer or more sophisticated phishing strategies. This raises concerns about the robustness of the model in real-world, evolving environments where the distribution and structure of malicious URLs differ significantly from the training set

Another assumption is related to the class distribution in the training dataset. In many practical scenarios, benign URLs far outnumber malicious ones, creating an imbalanced dataset. While this study used a moderately imbalanced dataset to ensure fair training and evaluation, this does not reflect real-world distributions. The consequence is that performance metrics such as accuracy, precision, and recall might be overly optimistic and not indicative of deployment performance. A classifier trained under balanced conditions might suffer from higher false positive rates when exposed to real-world data, potentially leading to user distrust or system inefficiency.

Chapter 2: **Literature Review**

2.1 Introduction

Phishing attacks have become one of the most prominent cybersecurity threats, where malicious actors use deceptive methods to obtain sensitive information from unsuspecting users. As phishing techniques evolve rapidly, traditional detection mechanisms, such as rule-based systems, have proven to be increasingly inadequate in mitigating these threats. Consequently, the application of machine learning (ML) in phishing detection has gained momentum due to its adaptive capabilities and potential for higher accuracy in identifying phishing attempts.

This chapter offers a comprehensive review of the literature on phishing detection methods, focusing on machine learning algorithms and their role in detecting malicious URLs. It examines both theoretical and empirical perspectives on existing methods and highlights the limitations that have led to the growing adoption of ML-based approaches. Several machine learning techniques are explored, including supervised learning algorithms like support vector machines (SVMs) and deep learning methods such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). Hybrid models, which integrate multiple algorithms to improve detection accuracy, are also discussed.

2.2 Theoretical Review

2.2.1 Routine Activity Theory (RAT)

According to the routine activity theory, a phishing attack is more likely to occur when three key elements converge: a motivated offender, a suitable target, and the absence of a capable guardian. In the context of phishing, the motivated offender is typically a cybercriminal who aims to deceive and exploit victims, often through social engineering tactics. The suitable target refers to the user who is susceptible to the phishing attempt, potentially due to a lack of awareness or security knowledge. The absence of a capable guardian represents the lack of effective security measures or user education to prevent such attacks (Rendall et al., 2020).

This theoretical framework suggests that a comprehensive phishing detection system should address all three of these elements in a multi-pronged approach. This includes identifying potential vulnerabilities in systems and user behaviour, educating users on phishing tactics and how to recognize them, and implementing robust technical safeguards to detect and mitigate phishing threats. By addressing the offender, the target, and the lack of guardianship,

organizations can develop a more holistic and effective strategy to combat the growing phishing threat.

2.2.2 Game Theory

Game theory can also provide valuable insights into the dynamics of phishing attacks and the development of countermeasures. Phishing can be viewed as a game between the attacker and the defender, where both parties make strategic decisions to maximize their respective outcomes. From the attacker's perspective, the goal is to craft phishing messages that are convincing enough to deceive the target and extract valuable information or gain unauthorized access. The attacker's strategy may involve techniques like social engineering, website spoofing, and leveraging psychological biases to increase the likelihood of success. On the other hand, the defender's objective is to protect users and systems from phishing attacks. The defender's strategy may involve implementing technical measures like URL filtering, email authentication, and machine learning-based detection, as well as educating users on phishing awareness and response. By analysing the game-theoretic interactions between the attacker and the defender, researchers can develop more effective countermeasures that anticipate and adapt to the attacker's evolving strategies.

2.2.3 Information Theory

Phishing attempts can be viewed as a communication channel where the phisher transmits a message with the intent of deceiving the victim and extracting sensitive information. The effectiveness of a phishing attack depends on the attacker's ability to minimize the noise and uncertainty in the communication channel, while the victim's ability to detect and resist the attack is influenced by the quality and reliability of the received information.

Information theory can be used to model the likelihood of a phishing attack based on the entropy, or uncertainty, of certain features within the communication channel. By analyzing the information content, patterns, and anomalies within phishing messages and URLs, information theory principles can be leveraged to develop more robust phishing detection models that can quantify the probability of a successful phishing attempt.

2.2.4 Optimization Theory

Optimization theory can provide a mathematical framework for designing effective phishing detection models. In the context of phishing, the goal of an optimization-based approach is to identify the optimal set of features, algorithms, and parameters that can accurately distinguish

between legitimate and phishing URLs or emails. Phishing detection can be formulated as a binary classification problem, where the objective is to minimize the misclassification rate (the sum of false positives and false negatives). This can be achieved through the use of various optimization techniques, such as linear programming, convex optimization, or metaheuristic algorithms.

By leveraging optimization theory, researchers can explore the trade-offs between detection accuracy, false-positive rates, and computational complexity. This can lead to the development of more efficient and effective phishing detection models that can be deployed in real-world scenarios. Moreover, optimization techniques can be used to continuously refine and update the phishing detection models as new threats and patterns emerge, ensuring the system remains robust and adaptable to the evolving phishing landscape.

2.2.5 Big O Notation

When designing a hybrid model for phishing URL detection, it is crucial to consider the computational complexity and scalability of the proposed solution. Big O notation can be used to analyse the time and space complexity of the various components of the model, ensuring that the overall system can handle large-scale phishing datasets and maintain efficient performance. For example, the use of machine learning algorithms, such as convolutional neural networks or multilayer perceptions, may have different time and space complexities depending on the model architecture, training process, and inference stage. Similarly, the complexity of feature extraction and preprocessing steps can also impact the overall performance of the hybrid model. By understanding the Big O complexity of each component, researchers can make informed decisions about the design and implementation of the hybrid model, optimizing for both accuracy and efficiency.

2.3 Empirical Review

The Anti-Phishing Working Group (APWG) identified 1.3 million unique phishing sites in the final quarter of 2022 alone, marking a record high. (Ozsahan & Worthington, 2024). This indicates the rapid evolving nature of phishing attacks It also indicates the need of a model that can keep up with rapid evolved nature of these phishing attacks.

A study titled "Towards a Multi-Layered Phishing Detection" (Rendall et al., 2020) presented a two-layered detection system that employs supervised machine learning techniques to identify phishing attacks. The authors evaluated their proposed system using a dataset of active phishing attacks and found that its performance, in terms of accuracy exceeding 95%, false-

positive rate below 5%, and false-negative rate below 5%, was comparable to the “state-of-the-art” in phishing detection. This study was conducted using a limited dataset, as it only used active phishing attacks, which may not capture the full spectrum of phishing tactics. However, the authors' use of supervised learning techniques and their focus on achieving high accuracy while minimizing false positives and negatives is commendable and aligns with the goals of effective phishing detection. This study can be compared to other machine learning-based approaches that have also shown promising results in phishing detection, and it can be connected to the broader trend of leveraging advanced techniques to combat the evolving phishing threat.

In another paper by Al mujahid et al, (2024). The authors compare various machine learning algorithms for detecting phishing websites, finding Convolutional Neural Networks to be the most effective. They emphasize the importance of using diverse datasets, employing two in their study: Mendeley and UCI. Their work highlights that CNNs outperform individual algorithms used like Random Forest and Support Vector Machines, achieving 99% accuracy after tunning as illustrated on Table 2.1. The authors connect this superior performance to CNNs' ability to capture complex patterns within data, crucial for identifying phishing websites mimicking legitimate ones. This study emphasizes the importance of feature engineering, using techniques like Term Frequency-Inverse Document Frequency, to improve accuracy, it also shows the algorithms that should be focused on by looking at their individual performance. While highlighting CNNs' effectiveness, the authors acknowledge limitations and suggest investigating the performance of ensemble methods, which combine multiple models, as a potential avenue for improving detection accuracy and addressing potential biases in datasets to develop even more robust phishing detection systems.

Table 2.1: Accuracy of models before and after Hyperparameter tuning Al mujahid et al, (2024)

Model	Accuracy (Before Tuning)	Accuracy (After Tuning)
CNN	97%	99%
XGBoost	93%	98%
DL	95%	98%
DT	97%	97%
RF	97%	97%
SVM	95%	96%
LR	94%	95%

KNN	95%	95%

Another study by Guptta et al., (2022), proposed a hybrid model for phishing URL detection, integrating multiple machine learning algorithms and feature extraction techniques. They developed a model that combines K-Nearest Neighbors, Naive Bayes, and Decision Trees, along with feature extraction methods like Term Frequency-Inverse Document Frequency and Latent Dirichlet Allocation. The hybrid nature of the model allows it to leverage the strengths of different algorithms and feature sets, leading to improved detection performance. Evaluation of the model on a diverse dataset of phishing and legitimate URLs shows that the hybrid approach outperforms individual machine learning algorithms, achieving an accuracy of over 97% and a low false-positive rate. This study demonstrates the potential benefits of combining multiple machine learning techniques and feature engineering methods for enhancing phishing detection capabilities.

2.3.1 Challenges of Current Phishing Detection Methods.

Existing phishing detection methods, whether based on machine learning or rule-based approaches, face several challenges that limit their effectiveness in real-world scenarios (Zuhair et al., 2016) (Das et al., 2020).

2.3.1.1 Ability to Handle the Evolving Nature of Phishing Attacks.

One key challenge is the ability to handle the evolving nature of phishing attacks. Attackers continuously develop new techniques and strategies to bypass existing detection mechanisms, requiring detection models to be continuously updated and refined (Vergelis, 2022; Wang et al., 2021). The dynamic and rapidly changing landscape of phishing threats poses a significant obstacle for phishing detection systems, as they must be able to adapt and evolve in response to the latest attack methods and tactics employed by cybercriminals. This requires ongoing research, development, and deployment of advanced detection algorithms and models that can effectively identify and mitigate novel phishing schemes as they emerge.

2.3.1.2 Volume of the Attacks.

The sheer volume of phishing attacks is another significant challenge that detection systems must address. According to the latest APWG Trends Report, a staggering 493.2 million phishing attacks were recorded in the previous quarter, representing a 173% increase from the previous quarter's 180.4 million attacks (The Anti-Phishing Working Group, 2024). Cybercriminals are constantly launching new phishing campaigns, resulting in a massive and

ever-increasing number of phishing URLs and emails that need to be analysed and classified. This high-volume threat landscape requires detection models that can scale efficiently, maintaining accurate and timely identification of phishing threats without compromising performance or resource utilization.

2.3.1.3 Evasion Techniques

Attackers often employ sophisticated evasion techniques, such as URL obfuscation, spoofing, and dynamic content generation, to bypass traditional detection methods. These evasion strategies exploit weaknesses in existing detection systems, requiring the development of more advanced, adaptive, and resilient detection models that can effectively identify and mitigate such sophisticated phishing attempts.

2.3.1.4 Multi-Channel Attacks

Phishing attacks have evolved beyond the traditional email vector, with attackers now leveraging a wide range of communication channels to distribute their malicious content. In addition to email, cybercriminals are increasingly targeting social media platforms, instant messaging applications, and SMS/text messaging to reach and manipulate their victims. This multi-channel approach allows them to exploit the various communication preferences and behaviours of potential targets, making it more challenging for users to identify and defend against these diversified phishing threats.

2.3.1.5 Personalized Phishing Techniques

Furthermore, the increasing prevalence of personalized phishing techniques, where attackers meticulously craft their messages and lures to target individual victims, adds a significant layer of complexity to the detection process. By tailoring their phishing attempts to specific individuals, based on personal information and behavioural patterns, attackers can bypass traditional detection mechanisms and exploit the human element more effectively. This personalized approach makes it increasingly challenging for detection systems to identify and mitigate such sophisticated phishing threats, underscoring the need for advanced techniques that can adapt to these evolving attack strategies.

2.3.1.6 False Positive and False Negatives

Another crucial aspect to consider is the balance between false-positive and false-negative rates in phishing detection systems. False-positive detections, where legitimate URLs are incorrectly identified as phishing, can result in user frustration and decreased trust in the detection system. Conversely, false negatives, where phishing URLs evade detection, can lead to successful

attacks and compromise sensitive user information. A study by Rao et al. (2022) emphasizes this balance by examining various performance metrics, including true positive rate (TPR), false positive rate (FPR), and false negative rate (FNR).

2.4 Framework Review

2.4.1 TensorFlow

The TensorFlow framework, developed by the Google Brain Team in 2015, has significantly influenced the field of machine learning (ML) and deep learning (DL), particularly in applications such as phishing detection.

TensorFlow is characterized by its flexibility and scalability, enabling developers to construct complex neural networks efficiently. Its architecture is based on data flow graphs where nodes represent mathematical operations and edges represent multidimensional data arrays known as tensors (Abadi et al., 2016). This modular design allows TensorFlow to be deployed across various platforms, including mobile devices and cloud environments, making it a versatile tool for diverse applications (Padilha & de Lucena, 2020). For instance, a systematic review highlighted TensorFlow's application in educational contexts for decision-making processes, showcasing its adaptability beyond traditional ML tasks (Padilha & Catrambone, 2020).

In the realm of phishing detection, hybrid models that leverage TensorFlow have shown promising results. One study proposed a hybrid approach that combines machine learning techniques with heuristic methods to enhance detection accuracy. This model utilized TensorFlow to implement a feature extraction process that analysed both URL characteristics and webpage content (Almutair & Alshoshan, 2023). The integration of these diverse data sources allowed for improved detection rates compared to models relying solely on URL features. This comparison underscores the effectiveness of hybrid models in addressing multifaceted threats like phishing. However, while TensorFlow provides robust tools for developing such models, it is not without limitations. The framework's complexity can pose challenges for new users. For example, while TensorFlow offers extensive APIs for building models, some researchers argue that frameworks like PyTorch provide a more user-friendly experience for rapid prototyping (Pang & Nijkamp, 2019). This highlights the importance of usability in selecting an appropriate framework for specific projects. Connecting these findings reveals a broader trend in cybersecurity where TensorFlow plays a crucial role in advancing phishing detection methodologies. A notable example is the use of deep learning techniques within TensorFlow to create sophisticated models capable of identifying phishing attempts with

high accuracy. Studies have demonstrated that models leveraging recurrent neural networks (RNNs) and long short-term memory (LSTM) architectures within TensorFlow can effectively analyse patterns in phishing URLs and emails (Zhang et al., 2021).

TensorFlow stands out as a powerful framework that has revolutionized machine learning applications across various sectors, including cybersecurity. Its ability to support hybrid models enhances phishing detection capabilities by integrating multiple data sources and methodologies.

2.4.2 PyTorch

The PyTorch framework, developed by Facebook's AI Research lab and officially launched in 2016, has rapidly gained prominence in the field of machine learning (ML) and deep learning (DL). PyTorch is distinguished by its imperative programming style, which allows for dynamic computation graphs that are constructed on-the-fly during execution (Chintala et al., 2019). This design choice contrasts sharply with TensorFlow's static graph approach, where the computational graph must be defined before execution (TechTarget, 2024). The dynamic nature of PyTorch facilitates easier debugging and experimentation, making it particularly appealing for researchers and developers who require flexibility in model design. For instance, a study highlighted that this feature enables real-time modifications to models during training, which is advantageous for rapid prototyping (Raschka, 2021).

In the context of phishing detection, PyTorch has been employed in various hybrid models that integrate multiple techniques to enhance detection accuracy. One notable study utilized PyTorch to implement a hybrid model combining machine learning algorithms with deep learning techniques. This model achieved significant improvements in detecting phishing websites by analysing URL features alongside content analysis using convolutional neural networks (CNNs) (Almutair & Alshoshan, 2023). The effectiveness of this approach underscores how PyTorch’s capabilities can be harnessed to address complex cybersecurity challenges.

However, despite its advantages, PyTorch is not without limitations. PyTorch offers ease of use and flexibility, it may not be as robust for deployment in production environments compared to TensorFlow. TensorFlow has a more extensive ecosystem for model deployment and scalability (InfoWorld, 2018). This highlights the importance of considering project requirements when choosing a framework; while PyTorch excels in research and development phases, TensorFlow may be more suitable for large-scale production applications.

Connecting these insights reveals a growing trend in the cybersecurity domain where frameworks like PyTorch are increasingly utilized to develop sophisticated phishing detection systems. A recent study demonstrated that integrating recurrent neural networks (RNNs) with traditional ML methods within a PyTorch-based framework led to improved detection rates for phishing attempts (Zhang et al., 2021). This connection illustrates how advancements in deep learning architectures are being leveraged to enhance cybersecurity measures against evolving threats.

PyTorch also stands out as a powerful framework that has transformed machine learning applications across various sectors, including cybersecurity. Its dynamic computation capabilities and user-friendly interface make it an excellent choice for developing hybrid models aimed at phishing detection.

2.4.3 Keras

Keras, a high-level neural network API developed by François Chollet and released in 2015, has become an essential tool in the field of deep learning, particularly for applications such as phishing detection.

Keras is designed to simplify the complexities associated with deep learning by providing a user-friendly interface that allows for rapid experimentation (Keras.org, 2023). Its modular nature enables users to build and train neural networks with minimal code while supporting various backends such as TensorFlow, Theano, and CNTK (Javatpoint, 2024). This flexibility allows Keras to be deployed across different platforms and devices, making it suitable for diverse applications. For example, Keras has been widely adopted in industries like healthcare and finance for tasks ranging from image classification to time series forecasting (GeeksforGeeks, 2024). In the context of phishing detection, Keras has been utilized in hybrid models that combine various techniques to enhance detection accuracy. One study implemented a hybrid deep learning technique for spoofing website URL detection using Keras. The researchers employed character-level tokenization to convert URLs into numerical forms before processing them through convolutional neural networks (CNNs) and long short-term memory (LSTM) layers (Alhassan et al., 2023). This approach demonstrated significant improvements in detecting phishing attempts compared to traditional methods that relied solely on URL features. The integration of Keras facilitated efficient model training and deployment, showcasing its effectiveness in addressing cybersecurity challenges. However, while Keras offers numerous advantages, it is not without limitations. Its high-level abstraction can lead

to slower performance compared to lower-level frameworks like TensorFlow when fine-tuning complex models (Simplilearn.com, 2024). Although Keras simplifies model development and experimentation, this trade-off may affect performance in production environments where speed is critical. This emphasizes the importance of evaluating project requirements when selecting a framework.

Keras stands out as a powerful framework that has transformed deep learning applications across various sectors. Its user-friendly interface and modular design facilitate rapid prototyping and experimentation in developing hybrid models aimed at phishing detection.

2.4.4 Amazon SageMaker

Amazon SageMaker is a fully managed machine learning (ML) service that provides a comprehensive suite of tools for building, training, and deploying machine learning models at scale.

SageMaker is designed to streamline the entire machine learning lifecycle, encompassing data preparation, model training, and deployment (XenonStack, 2024). Its architecture integrates several AWS services, allowing users to manage compute resources efficiently through a simple API. For instance, when a model is trained on SageMaker, the service automatically provisions the necessary Elastic Compute Cloud (EC2) instances and manages the underlying infrastructure (MLinProduction, 2024). This automation simplifies the process significantly compared to traditional methods where users would manually configure resources and manage dependencies. The platform also supports various built-in algorithms and frameworks, making it versatile for different ML tasks.

However, while SageMaker offers numerous advantages, it also has limitations. Its reliance on AWS infrastructure can lead to vendor lock-in, which may be a concern for organizations looking for flexibility in their ML deployments (XenonStack, 2024). Additionally, while SageMaker simplifies many aspects of model training and deployment, users may still need to possess a certain level of expertise in AWS services to navigate its features effectively.

Amazon SageMaker stands out as a powerful platform that has significantly impacted machine learning applications across various sectors, including cybersecurity. Its end-to-end capabilities streamline the development lifecycle of machine learning models while providing robust tools for experimentation and deployment.

2.5 URL Phishing Detection Algorithms

Among the common types of phishing attacks, malicious URLs have grown to be one of the most potential dangers to internet users and organizations. Of late, the emergence of machine learning has introduced new methods used for their detection. In this regard, the following literature review discusses the different machine learning-based algorithms used in URL phishing detection and provides a breakdown regarding their advantages, disadvantages and their performance.

2.5.1 Machine Learning Algorithms

2.5.1.1 Supervised Learning

Supervised learning is a type of machine learning whereby the algorithm learns from a labelled dataset. Think of it like this: a student learning with a teacher, where the teacher provides examples and the correct answers-or labels-and the student learns to find patterns and relationships between those examples and their corresponding answers. Common algorithms include:

2.5.1.1.1 support vector machine

A support vector machine (SVM) is a supervised machine learning algorithm that classifies data by finding an optimal line or hyperplane that maximizes the distance between each class in an N-dimensional space (IBM, 2023).

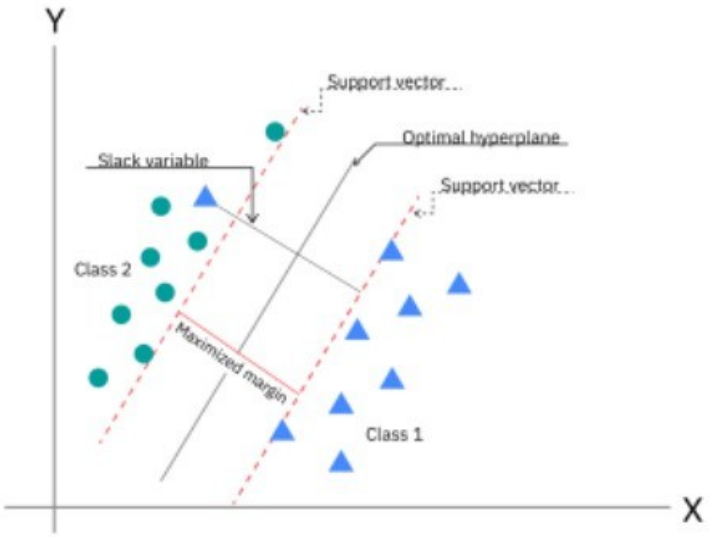


Figure 2.1: SVM (IBM, 2023).

Support Vector Machines (SVMs) in regards to phishing as illustrated in figure 2.1, works by finding the optimal hyperplane that divides phishing site from legitimate sites using data points in a multi-dimensional space. The hyperplane found in this case maximizes the margin between both classes; legitimates sites and phishing sites, and will lead to good generalization of unseen, new data. With different kernel functions, SVMs can deal with linear and non-linear data, and can be applied to different scenarios of URL phishing. SVM are particularly efficient in high dimensional spaces, where there are more features than samples. SVM is used in phishing detection, for example in task such as, URL classification, email classification and website content analysis. In this case, they classify the provided URLs, emails, and websites based on the features extracted from them. Although they are reliable, SVMs may be too expensive for use on large datasets, and their performance may be sensitive to the choice of kernel function.

In (Wahyudi et al. 2022) they implemented a system based on the Support Vector Machine (SVM) algorithm to detect phishing websites. The authors point out that the rate of phishing attacks has been rising. To counter this, they propose machine learning technique to identify and classify phishing website, they employ SVM. The system involves two main components: an implementation system and an evaluation system. For the evaluation system, they optimize the parameters of the SVM algorithm, while in the implementation system, they extract features from URLs to detect phishing websites. In this case, the page based, and network-based features are extracted to train the SVM model. The results of the study compare the performance of SVM algorithm with different kernels (linear, polynomial, RBF) with different machine learning algorithms i.e. Decision Tree and K Nearest Neighbours. Their results show that the SVM algorithm with polynomial kernel achieves the best phishing website detection accuracy (85.71%). Finally, they conclude using their proposed system, coupled with SVM and feature extraction, successfully detects phishing websites with high accuracy. This approach can help to reduce the risks of phishing attacks and improve online security.

2.5.1.1.1.1 Advantages of SVM

SVM have the following advantages:

i. Robust to Overfitting: By maximizing the margin between data points and the decision boundary, SVM can reduce overfitting, especially when used with an appropriate kernel and regularization.

ii. Versatile with Different Kernels: SVM supports different kernel functions (linear, polynomial, radial basis function (RBF), sigmoid) to model complex, non-linear relationships between features, making it highly adaptable to various problems.

iii. Clear Margin of Separation: SVM maximizes the margin between the classes, providing a strong geometric interpretation of the classification process. The larger margin often leads to better generalization.

2.5.1.1.1.2 Disadvantages of SVM

SVM have the following disadvantages:

i. Computationally Expensive: Training an SVM can be time-consuming and computationally expensive, especially on large datasets, since it involves solving a complex optimization problem.

ii. Sensitive to the Choice of Kernel and Parameters: SVM requires careful selection of the right kernel and tuning of parameters like the regularization parameter C and kernel parameters. Poor tuning can lead to suboptimal model performance.

iii. Not Suitable for Large Datasets: SVM struggles with very large datasets because the training complexity increases significantly with the size of the data. This makes SVM less suitable for real-time or large-scale applications.

iv. Poor Performance with Noisy Data: SVM is sensitive to noise and overlapping classes. In the presence of mislabelled data or noise, SVM can fail to find a good decision boundary.

2.5.1.1.2 Decision Trees

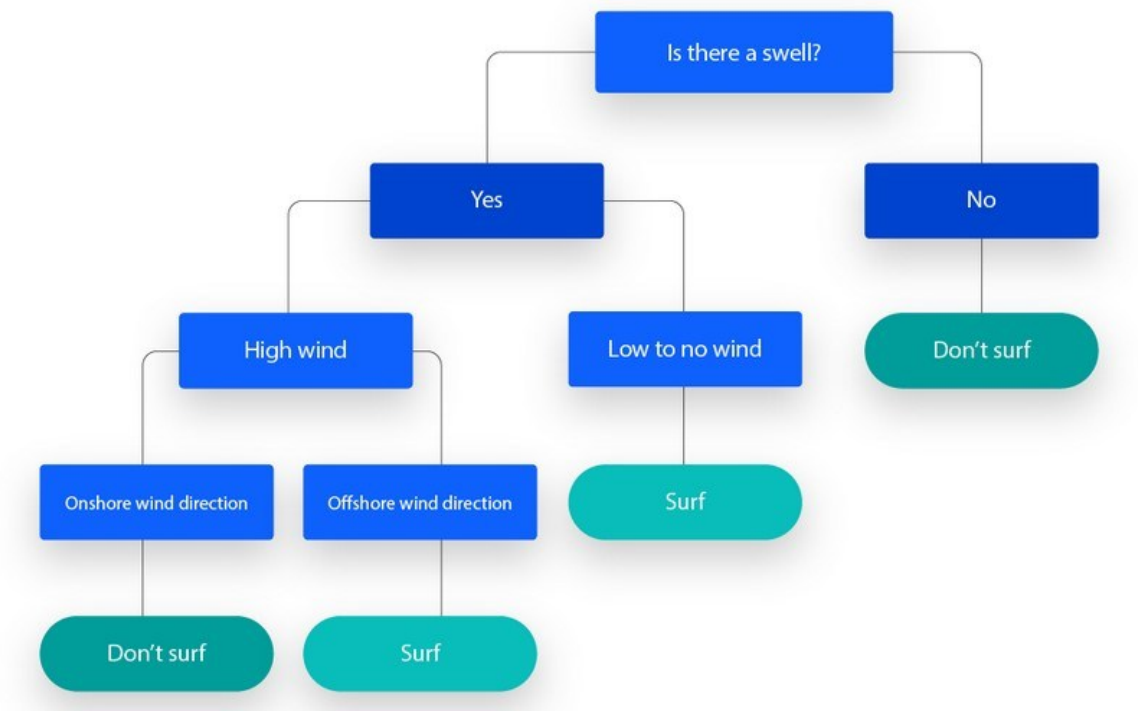


Figure 2.2: An illustration of a decision tree (IBM, n.d.)

The decision tree algorithm, as illustrated in the Figure 2.2, works by following a series of logical steps to reach a final decision – in this case, whether or not to surf. It uses a tree-like structure with branches representing different conditions and outcomes.

Here's how it works:

i. **Start at the Root:** The algorithm begins at the top of the tree with the root node, which asks the most important question: "Is there a swell?". This is the initial condition that determines the subsequent path.

ii. **Evaluate the Condition:** The algorithm checks if the condition at the current node is met. If there is a swell, it follows the "Yes" branch; otherwise, it follows the "No" branch, leading to the decision "Don't surf".

iii. **Branching:** Each branch leads to another node, which represents a further condition or a decision. In this case, the "Yes" branch leads to assessing wind conditions.

iv. **Recursive Evaluation:** The algorithm continues to evaluate the conditions at each subsequent node. If there's high wind, it further checks the wind direction (onshore or offshore). If there's low to no wind, it directly leads to the decision "Surf".

v. **Reaching the Leaf:** The algorithm keeps branching until it reaches a leaf node, which represents the final outcome or decision. In this diagram, the green ovals are the leaf nodes, indicating whether to "Surf" or "Don't surf".

Essentially, the decision tree algorithm systematically breaks down a complex decision into a series of simpler, conditional steps, ultimately leading to a clear and definitive outcome based on the input conditions.

2.5.1.1.2.1 Advantages of Decision Trees

Decision Trees have the following advantages.

i. Easy to Understand and Interpret: Decision trees are highly interpretable models. The tree-like structure makes it easy to visualize the decision-making process and understand how predictions are made
ii. Simple to Implement: Decision trees are straightforward to implement, requiring minimal data preparation like scaling or normalization.
iii. Requires Little Data Preprocessing: Unlike other models, decision trees don't require extensive data cleaning, like removing missing values or scaling variables.
iv. Can Handle Missing Values: Many implementations of decision trees can manage missing values by assigning probabilities to possible outcomes based on other branches of the tree

2.5.1.1.2.2 Disadvantages of Decision Trees

Decision Trees have the following disadvantages.

i. Prone to Overfitting: Decision trees tend to overfit, especially with small datasets or if the tree grows too deep. This can lead to poor generalization on unseen data.
ii. Unstable: Small changes in the data can lead to completely different tree structures. This lack of stability can be problematic for consistency in predictions.
iii. Bias Toward Dominant Classes: Decision trees are biased toward the majority class in imbalanced datasets, making it harder for the model to predict minority classes accurately

2.5.1.1.3 Random Forests

Random forest is a commonly used machine learning algorithm, trademarked by Leo Breiman and Adele Cutler, that combines the output of multiple decision trees to reach a single result.

Its ease of use and flexibility have fuelled its adoption, as it handles both classification and regression problems.

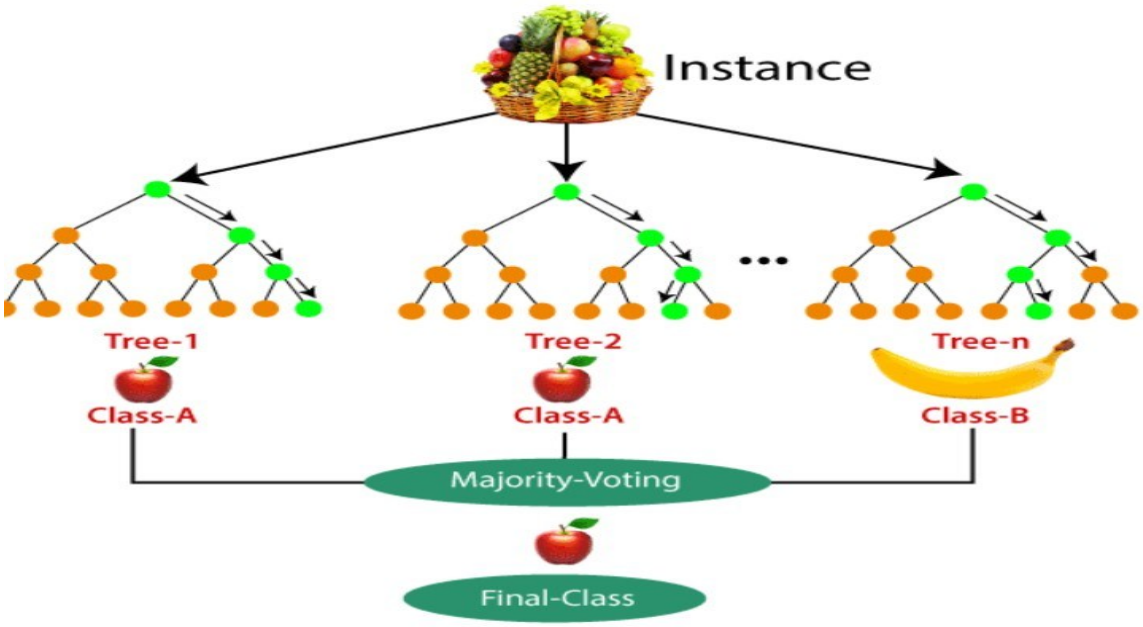


Figure 2.3: An illustration of a Random Forest Singh, Dwivedi, and Rastogi (2023).

Assume there is a dataset including several fruit photos. So, this dataset is sent into the Random Forest classifier. The dataset is separated into subsets and assigned to each decision tree. During the training phase, each decision tree generates a prediction result; when a new data point is encountered, the Random Forest classifier predicts the final decision based on the majority voting. Consider Figure 2.3.

In a recent work, Shah (2021) investigates the use of machine learning methods in particular the Random Forest algorithm to detect phishing emails. The study also calls for an increase in awareness about the increasingly prevalent phishing attacks. The author implements a machine learning based model to identify and classify phishing emails. The research employs the Random Forest algorithm to detect phishing emails with better accuracy and speed. He argues that Random Forest can handle large datasets, reduce classification time. They utilize a dataset of phishing and non-phishing emails, get the features, and train the Random Forest algorithm on the data to classify emails as either phishing or legitimate. The author also considers different machine learning algorithms: Naive Bayes, Support Vector Machines, Logistic Regression, and Decision Trees. Several metrics such as accuracy, precision, F1-score and recall were used in the evaluation of the models. Their results show that the Random Forest algorithm has the best accuracy in phishing email identification among the used algorithms.

The author concludes that the Random Forest algorithm can effectively detect phishing emails and thus making such an approach effective in the detection of phishing emails.

2.5.1.1.3.1 Advantages of Random Forest

Random Forests have the following advantages.

i. Robustness to Overfitting: Random Forest reduces the risk of overfitting by creating multiple decision trees on random subsets of the data and then averaging their predictions. This ensemble approach helps in creating a model that generalizes better than individual decision trees.
ii. High Accuracy: It often provides higher accuracy compared to simpler models, especially for classification tasks, due to its ability to capture complex data patterns.
iii. Works Well with Both Categorical and Numerical Data: Random Forest can handle a mix of categorical and numerical data without much preprocessing.
iv. Handles Missing Values: It can handle missing values by estimating missing data based on proximity to other observations within the forest.

2.5.1.1.3.2 Disadvantages of Random Forest

Random Forests have the following disadvantages.

i. Complexity and Interpretability: Random Forest is more complex and harder to interpret compared to simpler models like a single decision tree. Understanding the specific reasoning behind individual predictions can be challenging.
ii. Complexity and Interpretability: Random Forest is more complex and harder to interpret compared to simpler models like a single decision tree. Understanding the specific reasoning behind individual predictions can be challenging.
iii. Longer Training Time: Training a Random Forest model can be computationally expensive and time-consuming, especially with large datasets, since it involves training multiple decision trees.
iv. High Memory Usage: Because it requires the creation of numerous trees, it can consume a significant amount of memory and storage, especially for large datasets.
v. Slower Predictions: Due to the ensemble nature of the model, predictions can be slower compared to simpler models, especially if the forest has a large number of trees.

2.5.2 Detection using Deep Learning

Deep learning (DL) has emerged as a promising approach for identifying phishing URLs and websites due to its ability to automatically learn features from raw data. This literature review synthesizes recent studies that explore various deep learning techniques applied in phishing detection.

2.5.2.1 Convolutional Neural Networks (CNNs)

The architecture of Convolutional Neural Networks (CNNs) has become a prominent approach in the field of phishing detection, particularly due to their effectiveness in pattern recognition tasks. CNNs excel at analyzing raw data inputs, such as website screenshots and textual content, making them suitable for identifying phishing URLs. This literature review explores the application of CNNs in phishing detection, highlighting their architecture, performance metrics, and comparative advantages over traditional methods.

CNNs are designed to process grid-like topology data, such as images or sequences. The architecture typically consists of multiple layers, including convolutional layers, pooling layers, and fully connected layers. The convolutional layers apply filters to the input data to extract relevant features, while pooling layers reduce dimensionality and computational load by down-sampling the feature maps. This hierarchical feature extraction enables CNNs to learn complex patterns that are indicative of phishing attempts. Recent studies have shown that CNNs can be effectively tailored for phishing detection tasks. For instance, Wei et al. (2023) demonstrated a hybrid model combining CNN with Multi-Head Self-Attention (MHSA), achieving an impressive accuracy of **98.3%** in classifying phishing URLs. This model outperformed traditional approaches by leveraging attention mechanisms to focus on significant features within the URL data.

In evaluating the performance of CNN-based models in phishing detection, various metrics such as accuracy, precision, recall, and F1-score are commonly utilized. A recent study proposed a fine-tuned 1D CNN model that achieved an impressive accuracy of **99.85%**, demonstrating its robustness against existing benchmark models (Gupta, Gaurav, & Chui, 2023). This high level of accuracy indicates the model's effectiveness in accurately distinguishing between phishing and legitimate URLs. Similarly, another research effort utilizing a dataset from Kaggle reported a training accuracy of **97.2%**, further indicating the model's capability to generalize well across different data distributions (Gupta et al., 2023).

Additionally, ensemble methods that combine Convolutional Neural Networks (CNNs) with different architectures—such as Long Short-Term Memory (LSTM) networks—have shown promising results in phishing detection. For instance, integrating 1D CNN with LSTM layers has been found to improve accuracy in identifying malicious URLs compared to models that rely solely on LSTM architectures. A study demonstrated that this hybrid approach achieved an accuracy of **94.3%**, highlighting its effectiveness in capturing both spatial and temporal patterns within URL datasets (Opast Publishers, 2023).

Moreover, another study reported that the CNN-LSTM model provided an accuracy of **96.32%** for anomaly detection tasks, further reinforcing the benefits of combining these architectures (ACM, 2023). Such hybrid approaches leverage the strengths of both CNNs and LSTMs, enabling them to effectively learn from complex data structures and improve detection capabilities against phishing attacks.

2.5.2.2 Recurrent Neural Networks (RNNs)

The architecture of RNNs is characterized by their recurrent connections, which allow them to maintain a memory of previous inputs. This feature enables RNNs to capture temporal dependencies within sequential data. A common variant, Long Short-Term Memory (LSTM), addresses some limitations of standard RNNs by incorporating mechanisms to retain or forget information over longer sequences. This capability is particularly beneficial in phishing detection, where the context provided by preceding words or characters can be crucial for accurate classification.

Recent studies have demonstrated the effectiveness of RNNs in detecting phishing attempts. For instance, a study proposed a phishing content classifier based on RNNs that identifies features without human input. The results indicated that the RNN system outperformed state-of-the-art tools by effectively analyzing the textual structure of emails (Nurse et al., 2019). This approach highlights the potential of RNNs to enhance phishing detection systems by focusing on the nuances of language that may signal malicious intent.

The performance of RNN-based models in phishing detection is often assessed using various metrics such as accuracy, precision, recall, and F1-score. In one study, researchers reported that their RNN model achieved competitive classification accuracy in predicting phishing websites (Zhu et al., 2021). This demonstrates the model's capability to generalize well across different datasets and contexts. Additionally, the computational efficiency of RNNs is notable. The ability to process sequences in a single pass allows for faster inference times compared to

models requiring extensive feature extraction or preprocessing. This efficiency is crucial for real-time applications where timely detection can mitigate potential threats.

RNNs offer several advantages over traditional machine learning methods in phishing detection. One key benefit is their capacity for automatic feature extraction from raw input data, reducing the need for manual feature engineering. This adaptability allows RNNs to learn from new patterns and trends in phishing attacks more efficiently than conventional algorithms. Moreover, RNNs can be integrated with different architectures, such as Convolutional Neural Networks (CNNs), to create hybrid models that leverage the strengths of both approaches. For example, combining CNNs with LSTM layers has been shown to improve accuracy in malicious URL identification compared to models relying solely on LSTMs (Gurung et al., 2023). Such hybrid architectures enable more robust detection capabilities by capturing both spatial and temporal features within datasets.

Recurrent Neural Networks represent a powerful architecture for detecting phishing attacks through their ability to analyse sequential data effectively. Studies have consistently demonstrated high accuracy rates when employing RNNs for this purpose, often surpassing traditional methods (Nurse et al., 2019; Zhu et al., 2021). As phishing tactics continue to evolve, further research into hybrid models and enhancements in interpretability will bolster the effectiveness and reliability of RNN-based phishing detection systems.

2.5.3 Performance of DL techniques

The effectiveness of deep learning models in phishing detection is typically evaluated using metrics such as accuracy, precision, recall, and F1-score. Studies consistently show that deep learning methods outperform traditional machine learning techniques in these metrics due to their ability to learn complex feature representations from large datasets.

2.6 Phishig Detection Architectures

2.6.1 DARTH – Framework Architecture

In Mittal et al. (2022), a machine learning based architecture, DARTH was utilised to accurately identify phishing emails. Using both Natural Language Processing (NLP) and neural network-based techniques, the framework assesses each composite feature independently and combines its results for the classification of the emails as malicious or legitimate. They were able to utilize 150,000 emails and they acquired training data from across multiple sources including the authors own emails and phishtank.com, with precision score of 99.97% and an F

score of 99.98% and correctly classifying phishing emails with an accuracy of 99.98%. By combining multiple machine learning techniques in various ensembles across the landscape of composite features, they were able to achieve a high accurate in the identification of phishing emails.

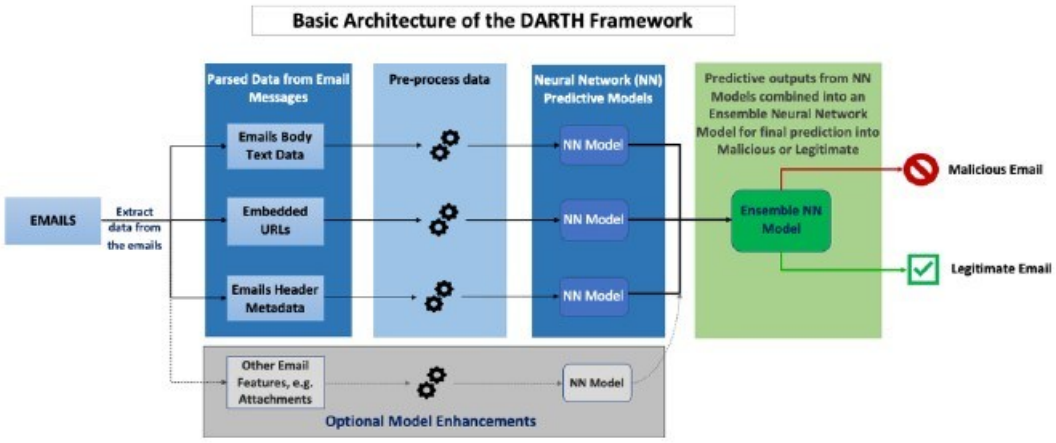


Figure 2.4: DARTH architecture Mittal et al. (2022).

With the DARTH framework as illustrated in Figure 2.4, an email is split into several elements i.e. the body text data, email header metadata, embedded URLs in the emails, and other feature i.e. attachment. First, the data extracted from the emails are vectorized and composite data is added to the data to provide additional features to the data. Various individual machine learning models are used to analyse the pre-processed data. The final step is that the output from each of the individual models are passed through an ensemble neural network. Their target of the ensemble model was to predict if an email is phishing or legitimate. One of the reasons that DARTH works is each of the individual models such as URLs and attachments are trained on a different dataset made public in the studies and research prior. The framework is flexible enough to add any new neural network model on a combination of individual composite features of the emails or over a more complicated model to improve its output.

2.6.2 LR+SVC+DT (LSD) Architecture.

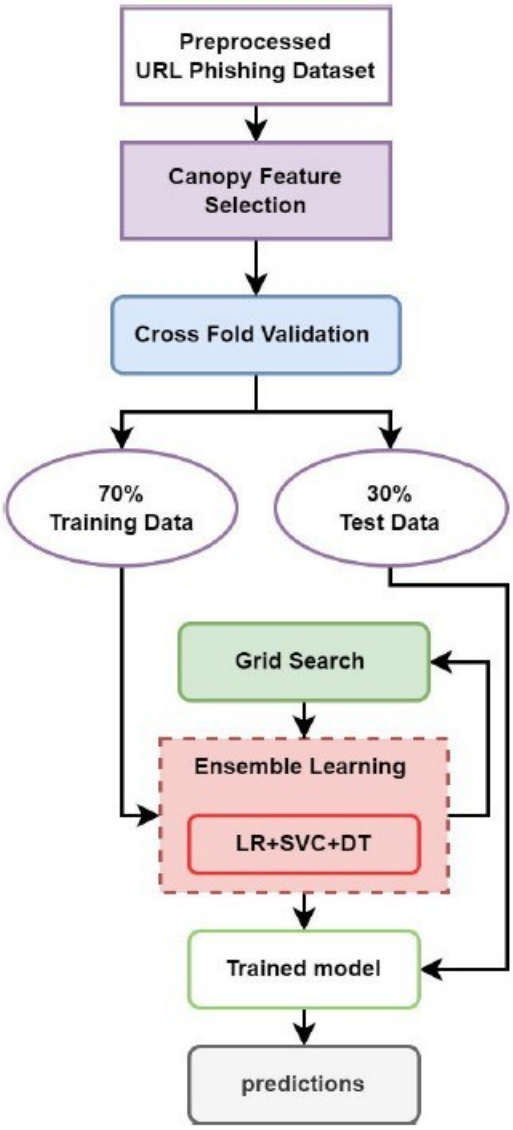


Figure 2.5:Canopy feature selection with LR+SVC+DT (LSD) ensemble learning model using grid search hyperparameter tuning and cross fold validation. Alhassan, Alhassan, and Abubakar (2023).

According to Alhassan, Alhassan, and Abubakar (2023) the architecture is models architecture as illustrated in Figure 2.5 is composed of several key components, designed to effectively identify phishing URLs: A dataset of URLs is collected from Kaggle, a renowned dataset repository following the Data Collection and Preprocessing step. Over 11,000 URLs make up this dataset, each tagged as "phishing" or "legitimate." The data set is pre-processed to remove null values and filling missing values to enhance data quality for training the model. Canopy Feature Selection: A feature selection is performed using a Canopy Clustering algorithm. This algorithm determine which features to use in the data set to improve model efficiency and

accuracy. Cross-Fold Validation and Grid Search: In order to ascertain how well the model generalizes across data subsets, the model is evaluated using cross fold validation. Hyperparameter tuning is taken care of via Grid search. Thus this technique systematically explores different combinations of hyperparameters to find the setting which results in best model performance. This architecture used 3 different machine learning algorithms, Linear Regression (LR): A statistical method for modelling the relationship between a dependent variable and one or more independent variables. Support Vector Machine (SVC): A supervised algorithm which tends to classify the data points in different classes by creating a hyperplane. Decision Tree (DT): A model based on an imaginary tree which makes its decisions based on a series of rules learned from the data The algorithms are combined together in to form a hybrid model which is an LR+SVC+DT model. The predictions of the three individual models are compared and the final result is picked using majority voting. Evaluation Metrics: They evaluated the system using various metrics such as accuracy, precision, recall and F1-score. This phishing detection system has a hybrid architecture which combines the strength of various machine learning algorithms. Voting, feature selection and hyper parameter tuning to improve its accuracy in URL phishing detection.

2.6.3 Stacking Ensemble Classifier Architecture

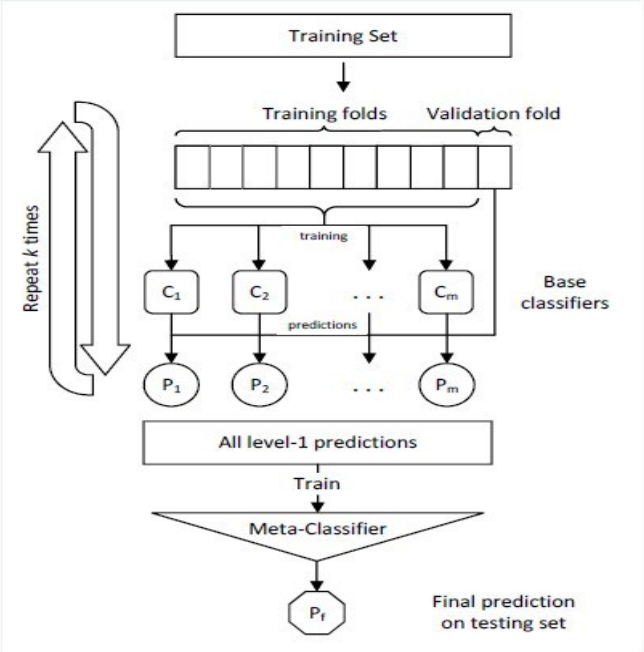


Figure 2.6: Stacking Ensemble Classifier Architecture by Newaz & Haq, (2023)

The architecture in Figure 2.6 is for a stacking ensemble classifier (Newaz & Haq, 2023). The training set is split into 2, a training fold and a validation fold to come up with a 10 fold

validation . These base classifiers are trained on 9 folds while the prediction is done on the remaining one-fold. The same process is repeated. 10 classification algorithms are considered as candidates for the base classifiers. In the feature selection process, Recursive Feature Elimination with Cross-Validation (RFECV) is used. Class probabilities from the base classifiers are utilized for training the meta-classifier. To choose the base classifiers for the stacking model, a greedy algorithm was utilised. As a meta-classifier, a Multilayer Perceptron (MLP) model was utilised.

This architecture is highly effective because of the following reasons

i. Improved Accuracy: This approach can determine correct results through integration of multiple base classifiers and a meta-classifier and has higher accuracy than any individual classifiers.
ii. Robustness: The ensemble approach is shown to be less prone to noise and overfitting than the base approach.
iii. Flexibility: It can combine predictions from a wide range of classifiers, allowing for greater flexibility in model design.

2.7 Gaps In Literature

One significant gap is the limited adaptability of existing models to the rapidly evolving nature of phishing attacks. Many traditional rule-based systems and single-algorithm machine learning models, such as those relying on blacklisting or heuristic methods, struggle to detect new or modified phishing URLs. These methods often fail to keep up with the sophisticated tactics employed by cybercriminals, who continuously alter their strategies to evade detection. The proposed hybrid model addresses this limitation by integrating multiple machines learning algorithms, including Convolutional Neural Networks (CNNs), Random Forests, and XGBoost, which collectively enhance the model's ability to adapt to emerging threats. Additionally, the inclusion of a feedback mechanism ensures continuous learning and updates, allowing the model to remain effective against evolving phishing techniques.

Another critical gap in the literature is the challenge of balancing false positive and false negative rates in phishing detection. Many existing models, such as those based on Support Vector Machines (SVMs) or Decision Trees, struggle to achieve an optimal balance, often resulting in either legitimate URLs being incorrectly flagged as phishing or phishing URLs going undetected. This trade-off undermines the reliability and usability of detection systems. The proposed hybrid model aims to mitigate this issue by leveraging the complementary

strengths of its constituent algorithms. For instance, CNNs excel at capturing complex patterns in URL structures, while Random Forests provide robust feature importance metrics. By combining these capabilities, the model is expected to achieve a better balance between precision and recall, thereby reducing both false positives and false negatives.

Another significant gap is the limited generalization of existing models across diverse datasets. Many models are trained and tested on limited datasets, which may not fully represent the diversity of phishing attacks in real-world scenarios. This can lead to poor generalization when the model is applied to new or unseen data. The proposed model addresses this issue by utilizing a large, diverse dataset from PhishTank, which includes a wide range of phishing and legitimate URLs. This approach is expected to improve the model's ability to generalize across different types of phishing attacks and reduce the risk of overfitting.

2.8 Conceptual Framework

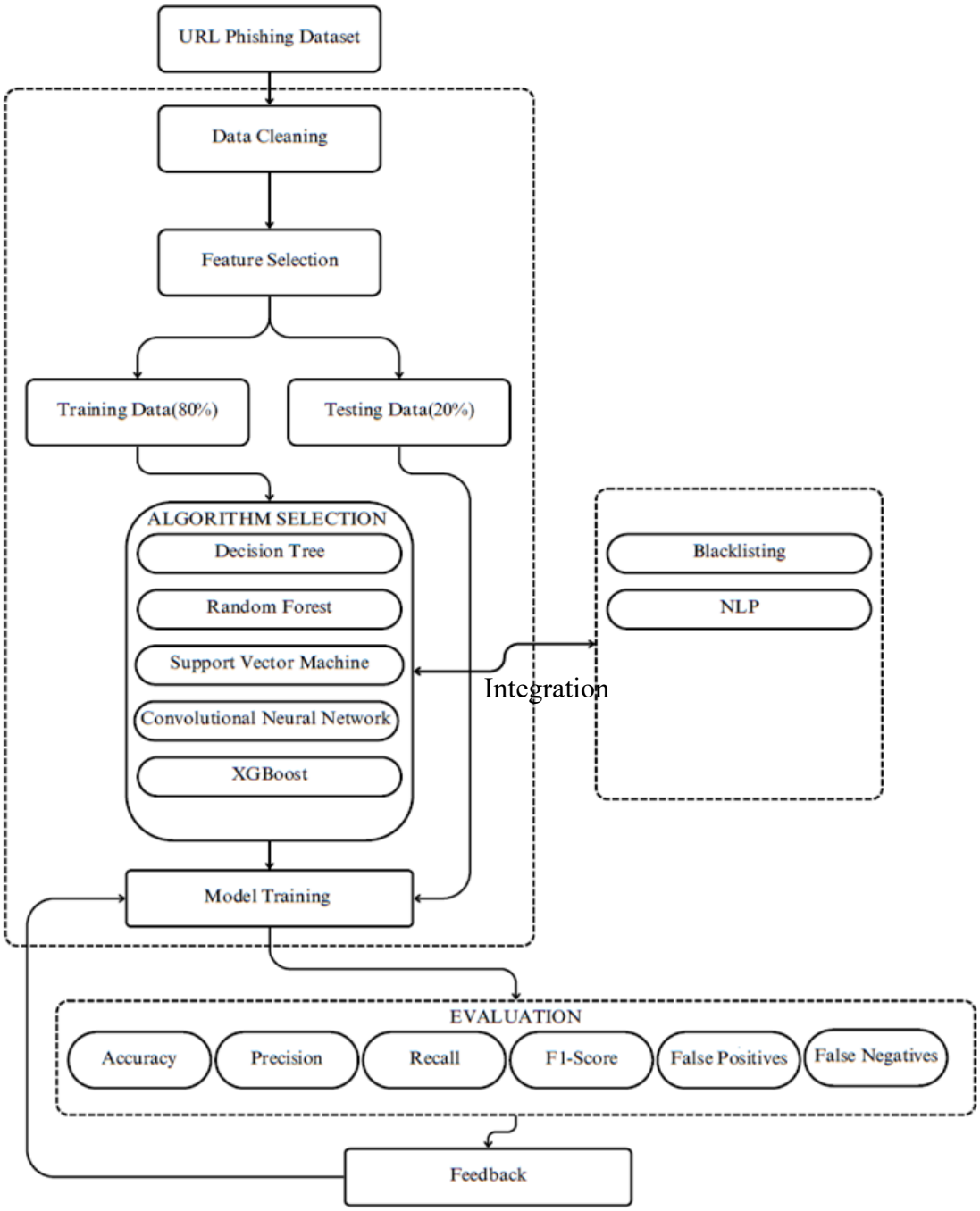


Figure 2.7: Conceptual model

The conceptual framework in figure 2.7 outlines the proposed hybrid model for detecting phishing websites. It involves the following components:

i. Phishing Dataset: A labelled data set of 134,850 legitimate and 100,945 phishing URLs will be utilized. This dataset will be used to train and test the detection model..

ii. Data Cleaning: This step involves cleaning the dataset by handling missing values, removing duplicates, and correcting errors to ensure the quality of the data used for training the models.

iii. Feature Selection: Phishing URLs have specific characteristics that distinguish them from legitimate ones. This step involves selecting relevant features (e.g., URL length, use of special characters, presence of login forms, etc.) that can be used to effectively identify phishing attempts

iv. Training Data (80%) / Testing Data (20%): The cleaned dataset is split into two parts:
    a. Training Data: Used to train the machine learning models to recognize patterns associated with phishing URLs.
    b. Testing Data: Used to evaluate the performance of the trained models on unseen data.

v. Algorithm Selection and Training: This stage involves identifying and selecting the most suitable machine learning algorithms for integration. Each algorithm is trained independently using the designated dataset, ensuring optimal learning of phishing patterns based on extracted features. The performance of each model is then evaluated against predefined metrics, such as accuracy, precision, recall, and F1-score, to determine their effectiveness in classifying phishing URLs. After evaluation, the best-performing algorithms are integrated into a hybrid model, leveraging their complementary strengths to enhance detection accuracy, adaptability and reduce false positives and negatives. This iterative process ensures that the final model is both robust and adaptable to evolving phishing techniques.

vi. Integration: The integration phase involves combining the identified machine learning algorithms to enhance phishing detection accuracy, adaptability and efficiency. Depending on the algorithm selected, one potential method is Ensemble Voting, where multiple classifiers independently analyse a URL before aggregating their predictions. In Hard Voting, each model such as CNN, XGBoost, Random Forest, and SVM classifies a URL, and the final decision is determined by the majority vote. Alternatively, Soft Voting assigns different weights to each model based on their performance, ensuring that higher-accuracy models, have a greater influence on the final prediction than lower-performing models.

    Another potential approach is Stacking, where multiple base models generate initial predictions, which are then passed to a secondary model for refined classification. For example, CNNs/XGBoost/ SVMs, or Random Forest serve as base classifiers, each

analysing the URL and producing an output. These outputs are then fed into a meta-model, such as logistic regression or a neural network, which learns from their combined predictions to make a more accurate final decision. This technique leverages the strengths of different algorithms while reducing individual weaknesses, resulting in a more robust phishing detection system.

Finally, Hierarchical Filtering is another potential method of integration, it introduces a multi-stage filtering mechanism. A fast and lightweight model, such as Random Forest, is used in the first stage to quickly classify URLs, filtering out obvious phishing and legitimate cases. URLs that remain uncertain or ambiguous are then passed to more complex and computationally intensive models, such as CNNs or XGBoost, for deeper analysis. This hierarchical approach improves efficiency by ensuring that only the most challenging cases require high-resource processing, reducing the overall computational cost while maintaining high detection accuracy.

vii. Model Training: Based on the identified algorithms, the hybrid model will be trained using the 80% subset of the PhishTank dataset, ensuring it learns meaningful patterns for phishing detection. The training process involves optimizing hyperparameters, applying feature selection techniques, and leveraging ensemble learning to improve classification accuracy. Cross-validation will be performed to prevent overfitting and ensure generalization, allowing the model to adapt effectively to unseen data. The iterative refinement process guarantees that the final model is robust, accurate, and capable of detecting evolving phishing threats with high reliability.

viii. Evaluation: Following training, the hybrid model is rigorously tested using the remaining 20% of the dataset to assess its performance. The evaluation process involves measuring key performance metrics, including accuracy, precision, recall, and F1-score, to determine the model's effectiveness in classifying phishing URLs. True positive and false positive rates are analysed to balance detection sensitivity and minimize incorrect classifications.

ix. Feedback: To maintain high detection accuracy and adaptability, a continuous feedback mechanism is implemented to refine the model over time. User feedback, along with misclassified URLs, are analysed to retrain and update the model periodically. Additionally, active learning techniques are employed, where the model identifies uncertain predictions and requests further validation to improve its learning process. This approach ensures that the hybrid model evolves alongside emerging phishing tactics, maintaining its effectiveness in real-time phishing detection.

Chapter 3: **Research Methodology**

3.1 Introduction

Research methodology describes how a researcher intends to carry out their study, detailing what data will be collected, from whom it will be collected, how it will be collected, and how it will be analyzed (Grad Coach, 2024). It can include various approaches such as qualitative, quantitative, or mixed methods, each suited for different types of research questions and objectives

3.2 Research design

The research design will take an applied approach, focusing on developing a practical solution to the problem of detecting phishing URLs. This involves collecting and analyzing real-world data, testing the proposed hybrid model on relevant datasets, and evaluating its performance and effectiveness in a practical setting. The goal is to create a model that can be deployed and used in real-world phishing detection systems, rather than a purely theoretical or academic exercise.

The research philosophy underlying this approach is pragmatism, which emphasizes the practical application and consequences of knowledge. This aligns well with the aim of improving the performance of phishing URL detection in a tangible way. The pragmatic approach also allows for the use of multiple research methods and data sources, as needed, to address the research objectives.

3.3 Population and Sampling

3.3.1 Population

According to Hu (2014), "a population is an entire group about which some information is required to be ascertained," and it can include various characteristics based on the research question or purpose of the study.

The population for this study is defined as phishing victims. However, we will be using secondary data rather than collecting primary data from the phishing victims directly. This approach is chosen due to the practical challenges and ethical considerations involved in directly engaging with phishing victims, who may have experienced psychological or financial harm. By utilizing a publicly available dataset, we can still investigate the problem of phishing URL detection without directly involving the victims, while ensuring the research remains sensitive to their well-being.

3.3.2 Sampling

Sampling refers to the process of selecting a subset of the population to study, rather than examining the entire population. This is often necessary due to practical or resource constraints, such as the large size of the population or the difficulties in accessing all members of the population.

For this research, the dataset obtained from the website https://archive.ics.uci.edu/dataset/967/phiusiil+phishing+url+dataset will be split into two parts: 80% for training the hybrid model and 20% for testing the model's performance. This common technique of splitting datasets will ensure that the model is trained on a representative sample of the population and its performance can be evaluated on a separate, unseen set of data.

3.4 Data Collection Methodology and Analysis

3.4.1 Data Collection

According to Creswell and Creswell (2018), "data collection is the process of gathering information from a variety of sources to answer research questions." The data for this research will be obtained from the publicly available dataset hosted on the website https://archive.ics.uci.edu/dataset/967/phiusiil+phishing+url+dataset. This dataset contains a total of 235,795 URLs, including 134,850 legitimate URLs and 100,945 phishing URLs. It was selected because it is recent (published on 3/3/2024), has no missing values, and is suitable for classification tasks. To ensure the integrity and relevance of the dataset, clear inclusion and exclusion criteria have been established:

3.4.1.1 Inclusion Criteria

i. URLs that are explicitly labeled as phishing or legitimate.

ii. URLs that contain complete and verifiable domain information.

iii. URLs that are unique and do not appear multiple times within the dataset.

iv. URLs that originate from verified sources such as cybersecurity repositories and open-access databases.

3.4.1.2 Exclusion Criteria

i. URLs that have missing or incomplete labels (i.e., unclear classification between phishing and legitimate).

ii. URLs that are duplicated within the dataset.
iii. URLs that redirect to inaccessible or non-functional web pages.
iv. URLs containing corrupted or inconsistent data that could impact analysis accuracy.

3.4.2 Data Analysis

According to Britannica (n.d.), "data analysis is the process of systematically collecting, cleaning, transforming, describing, modelling, and interpreting data," which is essential for deriving insights that can guide future actions and decisions.

The data analysis for this research will involve several key steps. First, the collected dataset of URLs will be cleaned and pre-processed to ensure consistency and remove any irrelevant or redundant information. This process may include handling missing values, removing duplicates, and standardizing URL formats.

Next, the pre-processed dataset will be split into training and testing sets using a cross-validation approach. This will involve dividing the dataset into multiple folds, where the model will be trained on a subset of the data (the training set) and then evaluated on the remaining data (the testing set).

The hybrid model, for example which combines a Convolutional Neural Network and a Random Forest classifier, will then be trained on the training set. The CNN component will be responsible for automatically extracting features from the URL structure, while the Random Forest component will provide a robust and interpretable classification mechanism. The feature importance metrics from the Random Forest algorithm will also be analysed to gain insights into the key characteristics of phishing URLs.

3.5 Research Quality and Reliability

Research Quality refers to the overall rigor and credibility of a study, encompassing various attributes that determine the trustworthiness and validity of the research findings. According to Boaz and Ashby (2003), quality research is characterized by a well-defined research topic, clear hypothesis, appropriate methodological approaches, and the ability to produce reliable results that can be replicated. Quality is not merely about obtaining positive results; it also involves adhering to ethical standards and ensuring that the research design is appropriate for the questions being investigated (Prolific, 2022). Reliability, on the other hand, specifically pertains to the consistency of a measure or method over time. It indicates how stable and

dependable the results are when the same study or measurement is repeated under similar conditions. As noted by Creswell and Creswell (2018), reliability can be assessed through various methods, such as test-retest reliability, inter-rater reliability, and internal consistency. High reliability in research means that the findings are likely to be replicable and that they accurately reflect the phenomenon being studied. To ensure the quality of the quality and reliability of the research, the following measures will be implemented:

i. Data Quality: The dataset will be sourced from a reliable source with a history of providing quality data sets
ii. Model Validation: The hybrid model will undergo Cross validation to asses its ability to generalize new data
iii. Test-retest reliability: The model will be tested on different subsets of data to ensure that is results are consistent

3.6 Development Methodology

The development methodology for this model outlines a structured approach to planning, executing, and managing the development process. This section will detail the chosen methodology that will guide the development of the hybrid phishing detection model.

The development methodology for this model will follow a slightly modified agile approach as it will not include the deployment and Launch stage as illustrated in Figure 3.1. Agile methodology emphasizes iterative and incremental development which will be useful in selecting the best algorithms to integrate, with a focus on adaptability, collaboration, and continuous improvement. This approach will allow for the continuous refining of the hybrid phishing detection model based on feedback and evolving needs.

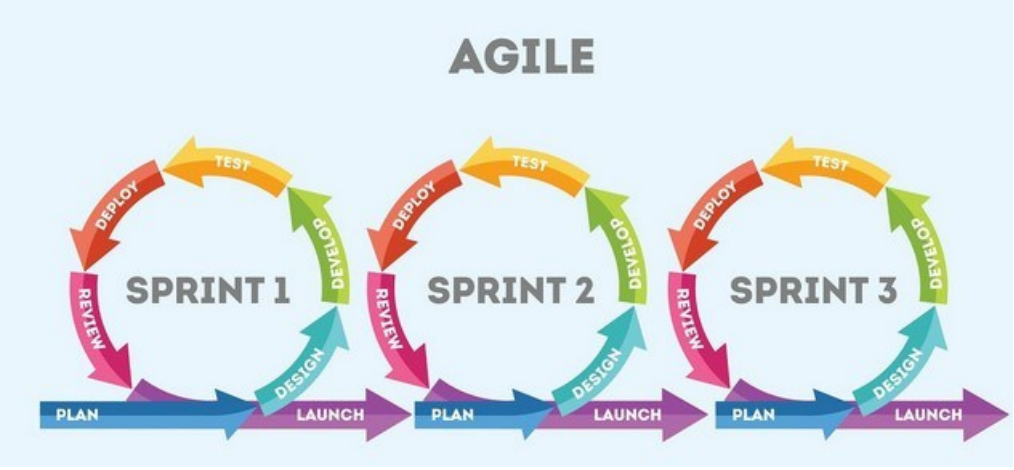


Figure 3.1: Agile methodology (Slawek-Polczynska, 2020)

3.6.1 Planning

The planning stage is a crucial component of the agile methodology for the development of the hybrid phishing detection model. During this stage, I together with my supervisor will define the project scope and objectives, identifying the key requirements for the hybrid model. we will develop a detailed plan, including timelines, milestones, and resource allocation, as well as establishing clear communication protocols and collaboration mechanisms within the team. Additionally, the planning stage will involve outlining the data sources and preprocessing steps, as well as selecting the initial set of machine learning models and techniques to be evaluated. This comprehensive planning will lay the foundation for the successful execution of the project using the agile approach.

3.6.2 Design

The design stage involves the detailed specification of the hybrid phishing detection model, including the architecture, algorithms, and data flow. During this stage, I will explore different machine learning techniques, such as ensemble methods, deep learning, and natural language processing, to determine the optimal combination for the hybrid model. I will also define the input and output data formats, as well as the preprocessing steps required to prepare the data for the model. Prototypes and wireframes may be developed to visualize the model's structure and functionality, enabling the team to refine the design based on feedback.

3.6.3 Development

The development stage involves the actual implementation of the hybrid phishing detection model based on the design specifications. During this stage, I will write the code, integrate the selected machine learning techniques, and build the necessary components of the model. I will also implement data preprocessing pipelines, experiment with different model configurations, and fine-tune the hyperparameters to optimize the model's performance. Throughout the development stage, I will engage in continuous integration and testing to ensure the model's functionality and reliability. The development stage is a crucial step in bringing the hybrid phishing detection model to life, and I will work collaboratively with my supervisor to ensure success.

3.6.4 Testing

The testing stage involves rigorously evaluating the hybrid phishing detection model to ensure its accuracy, reliability, and performance. In this phase I will conduct a series of tests, including:

i. Unit testing: Verifying the functionality of individual components and modules within the model.
ii. Integration testing: Ensuring the seamless integration and communication between the various components of the hybrid model.
iii. Performance testing: Evaluating the model's speed, scalability, and ability to handle large volumes of data.
iv. Accuracy testing: Assessing the model's ability to correctly identify phishing URLs across a diverse dataset.

The models performance will be evaluated according to the following parameters:

i. Accuracy: To measure the overall correctness of the model's predictions.
ii. Precision: To assess the model's ability to correctly identify positive cases (phishing URLs).
iii. Recall: To measure the model's ability to identify all actual positive cases.
iv. F1-score: To provide a balanced measure of precision and recall.
v. False Positives (FP): The number of legitimate URLs incorrectly identified as phishing.
vi. False Negatives (FN): The number of phishing URLs that were incorrectly identified as legitimate.

3.6.5 Review and Feedback

The review and feedback phase involves collaboration between me and the supervisor to evaluate the performance and effectiveness of the hybrid phishing detection model. During this phase, the team I will present the results of the testing stage, including the model's accuracy, reliability, and overall performance. The supervisor will provide feedback and guidance, helping the me to identify areas for improvement or additional testing. This collaborative process will enable me to refine the model, address any issues or concerns, and ensure the final solution meets the project's objectives. The review and feedback phase is a crucial step in the agile methodology, as it allows for continuous improvement and ensures the hybrid phishing detection model delivers the expected results.

3.7 Utilization and Dissemination of Search Results

3.7.1 Utilization of Search Results

The findings from this study will be utilized to enhance phishing detection systems by integrating the hybrid model into existing cybersecurity frameworks. Organizations, ie financial institutions, e-commerce platforms, and government agencies, can leverage the model to strengthen their security protocols. The results will also contribute to academia, providing a reference for future research on machine learning applications in cybersecurity. Additionally, the hybrid model can be adapted for real-time implementation in browser extensions as browser add on.

3.7.2 Dissemination of Search Results

The dissemination of research findings will occur through multiple channels to ensure fair distribution and accessibility of benefits derived from the research. The primary platform for publication will be Strathmore University's academic repository, ensuring accessibility to the academic community. Additionally, the developed model and relevant dataset will be made available on the open-source platform GitHub to encourage collaboration and continuous improvement. This approach fosters transparency and innovation in cybersecurity by allowing researchers and practitioners to replicate, refine, and enhance the model. Efforts will be made to ensure equitable access to these research benefits by sharing findings through conferences, workshops, and publicly accessible reports. By leveraging these dissemination strategies, the study aims to contribute meaningfully to both academic research and practical advancements in phishing detection and cybersecurity strategies.

3.8 Ethical Considerations

One of the key ethical considerations in this research is data privacy. This study employs the PhishTank database, which contains a comprehensive collection of known phishing URLs. Ensuring data privacy is crucial because it protects individuals from potential harm, such as identity theft and financial fraud, that can arise from unauthorized access to personal information. The PhishTank database does not contain any personal identifiable information (PII), meaning it does not disclose any personal data, thereby safeguarding users' identities.

Another consideration is the potential for algorithmic bias. Machine learning models can inadvertently reflect biases present in training datasets, leading to unfair outcomes against

certain groups. It is crucial to implement strategies for identifying and mitigating bias throughout the model development process (Barocas et al., 2019).

Furthermore, the risk of misuse of technology developed for phishing detection is a critical ethical consideration. While the primary goal of such technologies is to enhance cybersecurity and protect users from malicious attacks, there exists a significant potential for these tools to be exploited for nefarious purposes if they fall into the wrong hands. One major concern is that advanced phishing detection systems, particularly those utilizing machine learning algorithms, could be repurposed by cybercriminals to develop more sophisticated phishing attacks. For instance, the same techniques used to identify phishing URLs could be employed to create more convincing and harder-to-detect phishing schemes. This phenomenon is often referred to as "adversarial machine learning," where attackers manipulate input data to deceive algorithms (Carlini & Wagner, 2017).

Chapter 4: **System Analysis, Design and Architecture**

4.1 Introduction

This chapter details the development and architecture of the hybrid phishing URL detection model. It covers the development process, highlighting the framework selection based on machine learning technique evaluation. The chapter also presents the models design, including functional and non-functional requirements, and analyses component interactions using diagrams.

4.2 System Analysis

System analysis involves determining the needs and specifications of a system, including its intended functionality. In the context of this study, the goal was to create an algorithm capable of identifying phishing URLs and distinguishing them from legitimate ones using a hybrid model. This section provides details on both the functional and non-functional requirements considered during the development of the implemented solution.

4.2.1 Requirement Gathering

The requirement gathering phase is crucial in the development of the hybrid phishing URL detection model, as it ensures that the final system meets the necessary functional and non-functional expectations. This phase involves identifying user needs, system constraints, dataset requirements, and key functionalities that must be incorporated for effective phishing detection.

4.2.1.1 Functional Requirements

Functional requirements define the core capabilities of the phishing URL detection system. These requirements ensure that the system can efficiently analyze URLs, classify them accurately, and provide actionable insights to users. Below are the key functional requirements:

i. The system must allow users to input URLs for phishing detection.
ii. Users must be able to receive an immediate classification result indicating whether a URL is phishing or legitimate.
iii. The system must extract only URL-based features for classification.
iv. The system must be able integrate multiple machine learning models, including CNNs, XGBoost, Random Forest, and SVM, to classify URLs accurately
v. The model must support continuous learning, where misclassified URLs are flagged and used for model updates.

4.2.1.2 Non Functional Requirements

Non-functional requirements refer to the aspects of a system that relate to its design, quality, constraints, and external factors that may influence its performance. In this study, the following non-functional requirements were identified:

Table 4.1: Nonfunctional requirements

Non-Functional Requirement	Description
Availability	The model should always be accessible for use by users.
Reliability	The model should consistently produce accurate and reliable results when analysing URLs.
Accuracy	The model should have a minimal false positive and false negative rate to ensure phishing detection is highly precise.
Usability	The model should be user-friendly, with a graphical user interface (GUI) that allows users to easily input URLs and view results.
Maintainability	The model should be easy to maintain, update, and debug in case of any errors or performance issues.
Performance	The model should provide real-time URL classification results within a few milliseconds to support immediate decision-making.

4.3 System Design and Architecture

System architecture refers to abstract, conceptualization-oriented, global, and focused to achieve the mission and life cycle concepts of the system. Its purpose is to define a comprehensive solution based on principles, concepts, and properties logically related to and consistent with each other (Faisandier, Garry & Rick, 2022).

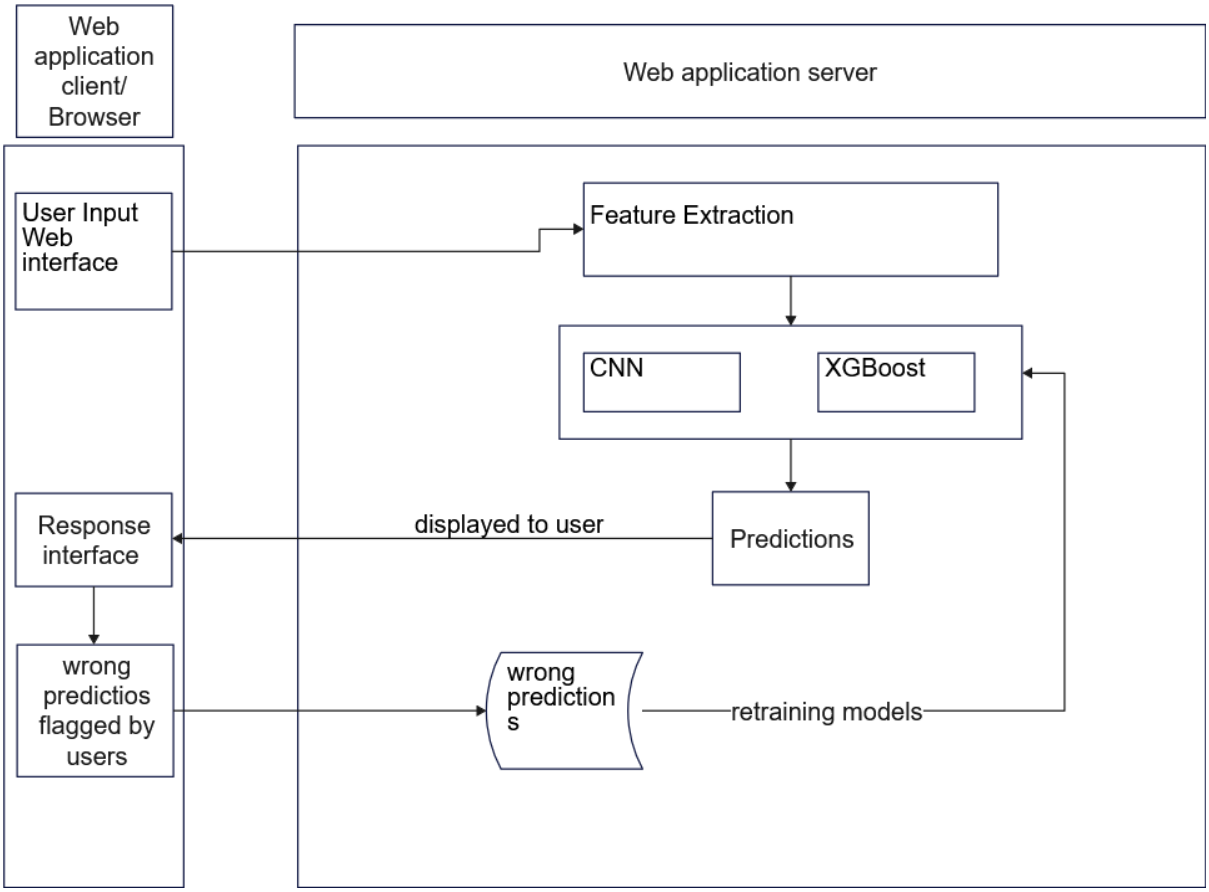


Figure 4.1: System Architecture

The phishing detection system illustrated in Figure 4.1 is designed as a web-based application that utilizes both a Convolutional Neural Network (CNN) and an XGBoost model to classify URLs as either legitimate or phishing. Users interact with the system through a web interface, where they can submit URLs for evaluation. Once a URL is entered, the web application server processes it by first extracting relevant features, such as lexical and structural attributes. These extracted features are then fed into the hybrid model, which analyse the input and generate a prediction.

The predicted classification is displayed back to the user through the response interface. If users identify any incorrect predictions, they have the option to flag them as wrong. These flagged instances are collected and fed back into the system to retrain the models, ensuring continuous improvement. The feedback loop allows the system to learn from its mistakes and enhance its accuracy over time by incorporating real-world misclassifications into future training cycles.

4.4 Use Case Diagram

The UML use case diagram represents possible interactions between an actor and or actors and the system. The phishing link detection application will include 2 main actors as show on Figure 4.2:

I. The User who will be submitting links and flagging incorrect predictions.
II. The retraining Function to be triggered when at least 10 new URLS have been added.

Table 4.2: Classification of Link Use Case

Use Case	Classification of Link
Primary Actor	User
Preconditions	User has submitted a link
Post Conditions	Response is sent back to the user and flagged links are captured.

Table 4.3: Classification of Link

Main Success Scenario	
Actors Responsibility	**System Responsibility**
1. User accesses the application via a web browser	
2. User submits a link	
	3. System Receives the link
	4. Hybrid Model classifies the link
	5. Prediction is displayed to the user.
6. User receives the classification response.	
7. If response is correct user exits	
8. If classification is wrong User flags it and exits	
	9. Link flagged by user is stored in csv file

	10. If more than 10 new links are added, the models are retrained incrementally on the new data

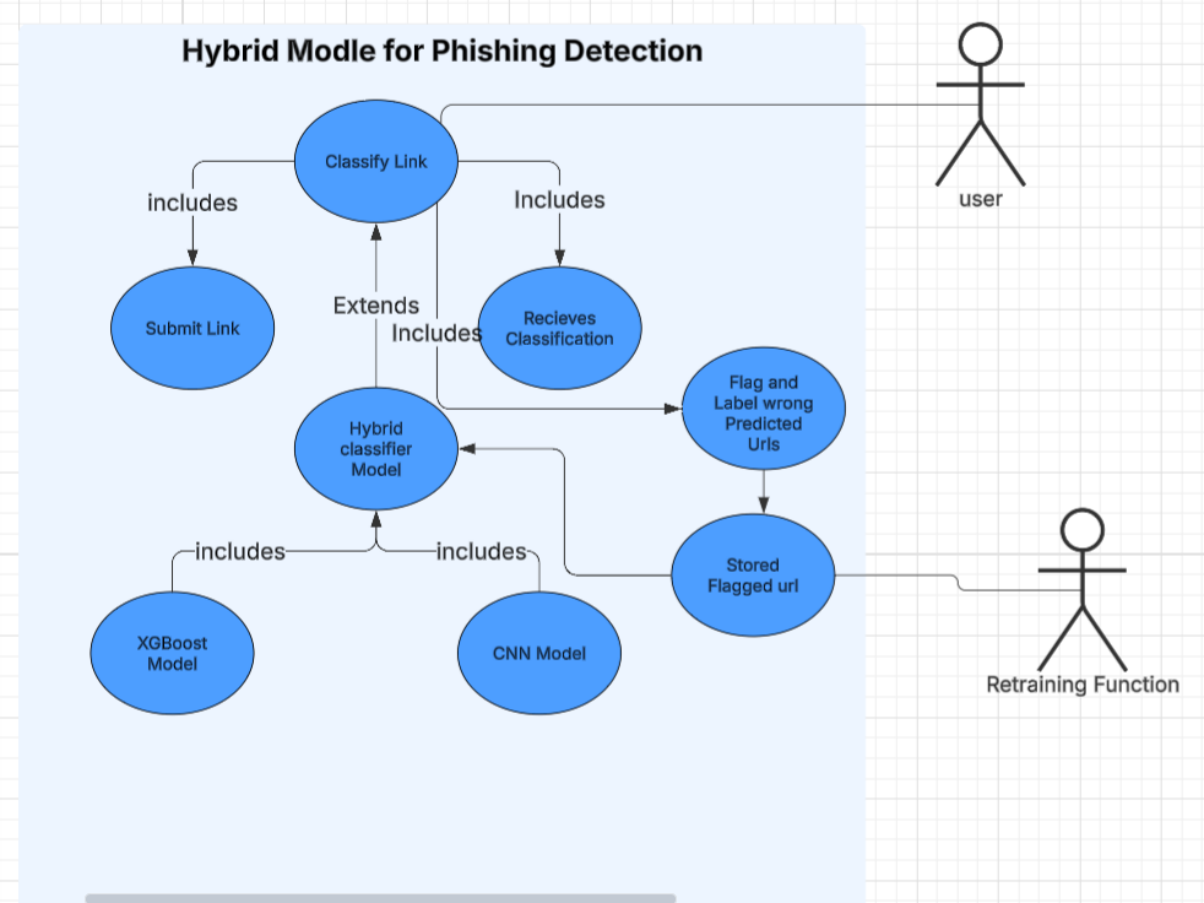


Figure 4.2: Use Case Diagram

The Use case diagram in Figure 4.2 describes a case where user accesses the web application through a browser and submits a URL for classification. The system receives the link and processes it using a hybrid model that combines CNN and XGBoost to determine whether the URL is legitimate or phishing. The classification result is then displayed to the user. If the user finds the prediction accurate, they simply exit the application. However, if the classification is incorrect, the user flags the misclassified URL before exiting. The system then stores flagged URLs in a CSV file. Once the number of flagged URLs surpasses ten, the model undergoes incremental retraining using the newly added data to improve its accuracy over time.

4.5 Data Flow Diagram

A data-flow diagram (DFD) is a visual representation of how data moves through a system or a process. The DFD additionally gives details about each entity's inputs and outputs as well as

the process itself. In this study, the information is submitted by the user, processed by the system, and classification status relayed back to the client.

4.5.1 Sequence Diagram

A sequence diagram is a diagram created using the Unified Modelling Language (UML) that shows the flow of messages during an interaction between objects. The illustration on Figure 4.3 breaks down the sequence of actions to demonstrate how the hybrid model used in phishing detection accomplishes its goal.

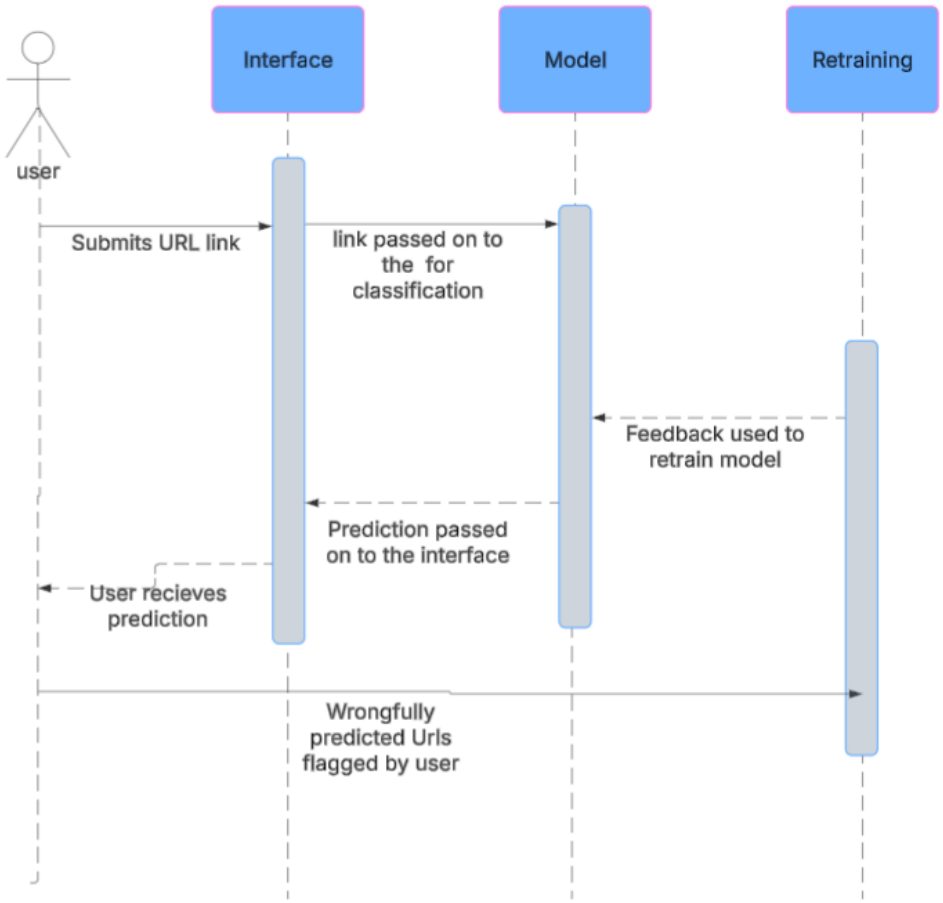


Figure 4.3: Sequence diagram

4.6 Wireframes

A wireframe is a schematic or blueprint used to facilitate communication between software and or website development stakeholders regarding the organizational layout of the same. Figure 4.4 shows the user interface where users will have access to in order to submit and receive the

predictions, it also contains a feedback section where users can flag and label the URLs with the correct label (phishing or legitimate) to be used for retraining the model.

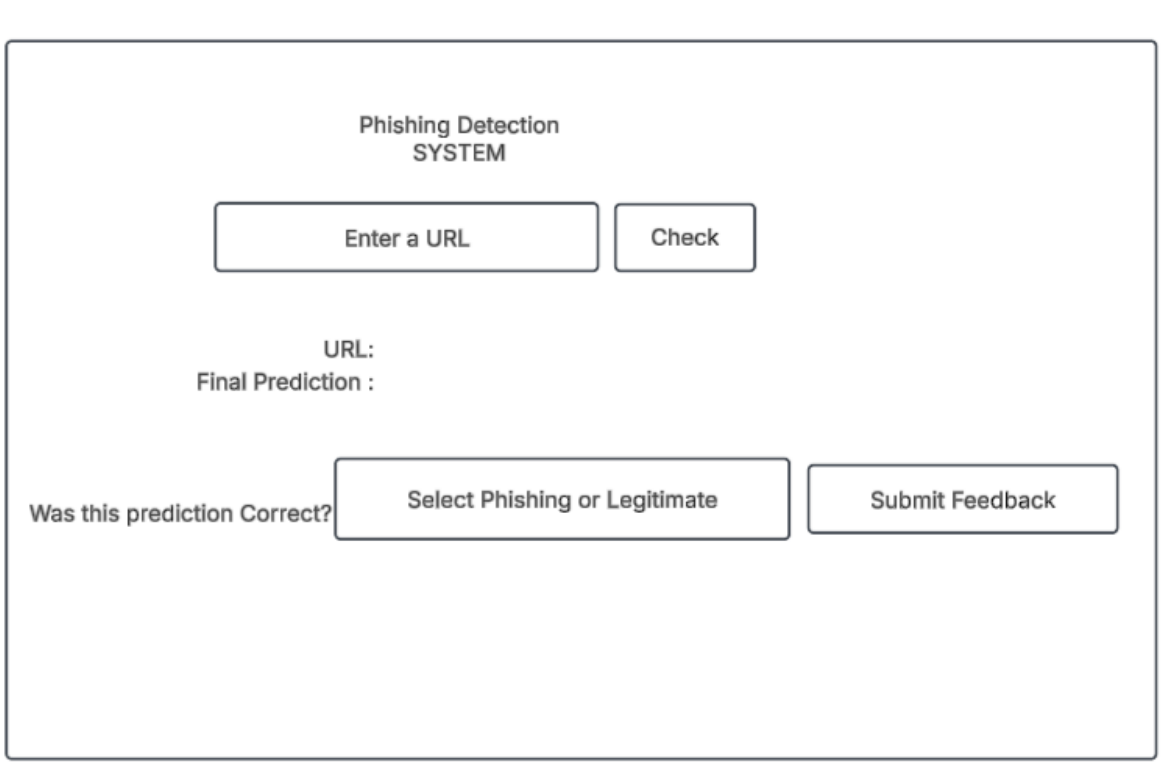


Figure 4.4: Submission and Feedback wireframe

Chapter 5: **Model Training Testing and Implementation**

5.1 Introduction

This chapter details the model training process, including dataset selection, feature extraction, training procedures, and evaluation metrics. The primary objective is to build a hybrid model capable of achieving higher accuracy than their individual models.

5.2 Model Training

5.2.1 Dataset and Feature Extraction

5.2.1.1 Dataset

The dataset used in this study is the PhiUSIIL Phishing URL Dataset, which consists of 134,850 legitimate URLs and 100,945 phishing URLs. For this research, only the URL and label columns were extracted from the dataset. The label column indicates whether a URL is legitimate (1) or phishing (0), ensuring a binary classification approach.

5.2.1.2 Feature extraction

Since only the URL column was extracted, the feature extraction process focused on deriving meaningful characteristics directly from the URLs. The following features were extracted:

i. URL Length: Measures the total number of characters in the URL.
ii. Special Characters Count: Counts occurrences of special characters (e.g., '@', '#', '&', '%', '$', '!').
iii. Subdomain Count: Identifies the number of subdomains in the URL structure.
iv. Suspicious Keywords: Scans the URL for keywords such as 'login', 'secure', 'verify', 'update', and 'bank'.
v. TLD Analysis: Determines whether the top-level domain (TLD) belongs to a set of common TLDs (e.g., .com, .org, .net).
vi. URL Structure Inconsistency: Checks if 'https' is missing from the URL.
vii. Lexical Features: Identifies patterns such as numeric sequences or mixed alphanumeric characters.

5.2.2 Individual Model Training

5.2.2.1 Random Forest Training.

A Random Forest classifier was initialized with 100 estimators, a maximum depth of 10, and a minimum sample split of 5. The model was trained using cross-validation with five folds to

evaluate its performance across different data subsets. The classifier was then fitted to the training data, and predictions were made on the test set. As shown in figure

```
# Train Random Forest
rf_model = RandomForestClassifier(n_estimators=100, max_depth=10, min_samples_split=5, random_state=42)
rf_scores = cross_val_score(rf_model, X_train, y_train, cv=5)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
y_pred_prob_rf = rf_model.predict_proba(X_test)[:, 1]
```

Figure 5.1: Random Forest Training

The model's efficiency was measured by calculating the average prediction time per URL as shown in figure 5.2.

```
# Measure Prediction Time per URL
start_time = time.time()
y_pred_rf = rf_model.predict(X_test)
prediction_time = (time.time() - start_time) / len(X_test)
print(f"Average Prediction Time per URL: {prediction_time:.6f} seconds")
```

Figure 5.2: Random Forest time function

A classification report was generated to evaluate performance metrics. Additionally, an ROC curve was plotted as shown in figure 5.3 to illustrate the trade-off between true positive and false positive rates, with the area under the curve serving as a performance indicator.

```
# ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred_prob_rf)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='grey', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred_rf)

plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Legitimate', 'Phishing'], yticklabels=['Legitimate', 'Phishing'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```

Figure 5.3: Random Forest ROC and Confusion Matrix

5.2.2.2 Decision Tree Model Training.

A Decision Tree classifier was trained with a maximum depth of 10 and a minimum split size of 5 as shown in figure 5.4. Its performance was evaluated using 5-fold cross-validation, followed by testing on unseen data.

```
# Train and evaluate Decision Tree
dt_model = DecisionTreeClassifier(max_depth=10, min_samples_split=5, random_state=42)
dt_scores = cross_val_score(dt_model, X_train, y_train, cv=5)
avg_cv_accuracy = np.mean(dt_scores)
print("Decision Tree Cross-Validation Accuracy:", avg_cv_accuracy)

dt_model.fit(X_train, y_train)
y_pred_dt = dt_model.predict(X_test)
test_accuracy = accuracy_score(y_test, y_pred_dt)
print("Decision Tree Accuracy:", test_accuracy)
```

Figure 5.4: Training and Evaluating a Decision Tree Model

The model's accuracy, precision, recall, and F1-score were assessed, The ROC curve and AUC score together with a confusion matrix were plotted as shown in Figure 5.5 to be used to measure the model's ability to distinguish between phishing and legitimate URLs.

```
# ROC Curve
y_prob_dt = dt_model.predict_proba(X_test)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_prob_dt)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred_dt)

plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Legitimate', 'Phishing'], yticklabels=['Legitimate', 'Phishing'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix - Decision Tree')
plt.show()
```

Figure 5.5: Decision Tree ROC and Confusion Matrix Plotting

Additionally, the model's efficiency was evaluated by calculating the average prediction time per URL. A final summary of results was presented, including cross-validation accuracy, test accuracy, and computational performance as shown in figure 5.6.

```
# Measure Prediction Time per URL
start_time = time.time()
y_pred_dt = dt_model.predict(X_test)
prediction_time = (time.time() - start_time) / len(X_test)
print(f"Average Prediction Time per URL: {prediction_time:.6f} seconds")
```

Figure 5.6: Decision Tree Time Prediction Function

5.2.2.3 XGBoost Model Training

An XGBoost classifier was defined with specific parameters, including a learning rate of 0.2, a maximum tree depth of 3, 300 boosting rounds, and subsampling at 80%. The model was then trained on the training set, and the time taken for this process was recorded as shown in Figure 5.7.

```
# Define the XGBoost model
xgb_model = xgb.XGBClassifier(
    objective='binary:logistic',
    random_state=42,
    eval_metric='logloss',
    colsample_bytree=1.0,
    learning_rate=0.2,
    max_depth=3,
    n_estimators=300,
    subsample=0.8
)

print("Training XGBoost Model with Fixed Parameters...")
start_time_train = time.time()

# Train the model
xgb_model.fit(X_train, y_train)

end_time_train = time.time()
train_duration = end_time_train - start_time_train
print(f"XGBoost Training Completed in {train_duration:.2f} seconds")
```

Figure 5.7: XGBoost Model Training

After training, the model was evaluated on the test set. Predictions were made using a 0.5 probability threshold to classify URLs as phishing or legitimate. Several performance metrics were calculated: accuracy, precision, recall, F1-score, and ROC AUC score as shown in Figure 5.8. The confusion matrix and a detailed classification report were also generated to provide insights into the model's performance across different classes. Additionally, the time taken to predict a single URL was measured to evaluate inference speed.

```
# Evaluate the model
y_pred_prob = xgb_model.predict_proba(X_test)[:, 1]
y_pred = (y_pred_prob > 0.5).astype(int)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
report = classification_report(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred_prob)

print("\nEvaluation of XGBoost Model:")
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-score: {f1:.4f}")
print(f"ROC AUC: {roc_auc:.4f}")
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", report)

# Time test
trial_url = "https://kucoinloyuugene.webflow.io/"
start_time_pred = time.time()
features_trial = extract_features([trial_url])
xgb_model.predict(features_trial)
end_time_pred = time.time()
time_per_url = end_time_pred - start_time_pred
print(f"\nTime taken to predict one URL: {time_per_url:.6f} seconds")
```

Figure 5.8: XGBoost Evaluation Code.

Finally, an ROC curve was plotted to visualize the model's ability to distinguish between phishing and legitimate URLs, with the AUC score included to assess its discriminative power.

5.2.2.4 Convocational Neural Network Model Training

The CNN model was designed using Keras with a 1D convolutional layer (128 filters, kernel size of 3) followed by ReLU activation. To prevent overfitting, a 30% dropout rate was applied after the convolutional layer. The output was flattened, then passed through a fully connected dense layer with 64 neurons and ReLU activation, followed by another dropout layer. The final output layer had one neuron with a sigmoid activation, predicting phishing (1) or legitimate (0).

The model was compiled with the Adam optimizer (learning rate of 0.001) and binary cross-entropy loss for binary classification. Accuracy was used as the metric, and the model was trained for 20 epochs with a batch size of 32 as shown in figure 5.9.

```
# Define CNN Model
model = keras.Sequential([
    layers.Conv1D(filters=128, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1)),
    layers.Dropout(0.3),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(1, activation='sigmoid')
])

# Compile model
model.compile(optimizer=keras.optimizers.Adam(learning_rate=0.001),
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Train model
model.fit(X_train, y_train, epochs=20, batch_size=32, validation_data=(X_test, y_test))
```

Figure 5.9: CNN Training

After training, the model was saved and evaluated using the test set. Predictions were made based on probabilities, which were converted into binary labels using a threshold of 0.5 as shown in figure 5.10.

```
# Predict probabilities for the test set
y_pred_prob = model.predict(X_test)

# Convert probabilities to binary class labels
y_pred = (y_pred_prob > 0.5).astype(int)
```

Figure 5.10 CNN Probability to Binary

To assess the model's performance, a confusion matrix and ROC curve were generated. The confusion matrix displayed classification results, and the ROC curve showed the trade-off between true positive and false positive rates, with the area under the curve (AUC) providing a measure of the model's discriminative power. The combination of these evaluations provided insights into the model's ability to distinguish between phishing and legitimate URLs.

5.2.2.5 SVM Model Training

To train the SVM model, Stratified K-Fold Cross-Validation was used to ensure that the class distribution was preserved across training and validation sets. In each fold, the SVM model, which used the RBF kernel, was trained on the training subset, and predictions were made on

the validation subset. The model was trained using the svm_model.fit() method, which learns from the features and labels in the training fold. After training, the model made predictions on the validation set using predict_proba() to generate probabilities, which were converted into binary class predictions by applying a 0.5 threshold as shown in Figure 5.11.

```
# Implement Stratified K-Fold Cross Validation
kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
fold_num = 1
svm_model = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42, probability=True)

accuracies = []
conf_matrices = []
roc_aucs = []
fprs = []
tprs = []
all_y_test = []
all_y_pred_probs = []

total_folds = kf.get_n_splits()
print("Starting Cross-Validation Training...")
for train_index, val_index in kf.split(X_train, y_train):
    print(f"\nTraining Fold {fold_num} ({(fold_num/total_folds)*100:.2f}% done)")
    X_train_fold, X_val_fold = X_train[train_index], X_train[val_index]
    y_train_fold, y_val_fold = y_train.to_numpy()[train_index], y_train.to_numpy()[val_index]

    start_train_time = time.time()
    svm_model.fit(X_train_fold, y_train_fold)
    end_train_time = time.time()
    train_duration = end_train_time - start_train_time
    print(f"Fold {fold_num} Training Completed in {train_duration:.2f} seconds")

    y_pred_prob = svm_model.predict_proba(X_val_fold)[:, 1]
    y_pred = (y_pred_prob > 0.5).astype(int)

    accuracies.append(accuracy_score(y_val_fold, y_pred))
    conf_matrices.append(confusion_matrix(y_val_fold, y_pred))

    fpr, tpr, _ = roc_curve(y_val_fold, y_pred_prob)
    roc_auc = auc(fpr, tpr)
    fprs.append(fpr)
    tprs.append(tpr)
    roc_aucs.append(roc_auc)

    all_y_test.extend(y_val_fold)
    all_y_pred_probs.extend(y_pred_prob)

    fold_num += 1
```

Figure 5.11: SVM K-fold validation

Performance metrics such as accuracy, confusion matrix, ROC curve, and AUC (Area Under the Curve) were then calculated to assess how well the model was performing for each fold. These metrics helped to evaluate the model's ability to correctly classify legitimate and phishing URLs.

Once the cross-validation process was complete, the model was retrained on the entire training dataset, and final performance was evaluated on the test set as shown in Figure 5.12.

```
# Final Training and Evaluation
start_time = time.time()
svm_model.fit(X_train, y_train)
y_pred_svm_prob = svm_model.predict_proba(X_test)[:, 1]
y_pred_svm = (y_pred_svm_prob > 0.5).astype(int)
end_time = time.time()

print("\nFinal Model Evaluation")
print("SVM Accuracy:", accuracy_score(y_test, y_pred_svm))
print(classification_report(y_test, y_pred_svm))
print(f"Total Training and Evaluation Time: {end_time - start_time:.2f} seconds")
```

Figure 5.12: SVM Final Training

This included generating the classification report, plotting the confusion matrix, and assessing the ROC curve and AUC as shown in Figure 5.13. Additionally, the time taken for training and evaluation is recorded to measure computational efficiency.

```
# Plot confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred_svm)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Legit', 'Phish'], yticklabels=['Legit', 'Phish'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Plot ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred_svm_prob)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```

Figure 5.13: SVM ROC and Confusion Matrix

5.2.3 Hybrid Model

The Hybrid model combines the XGBoost and a CNN model. The model is deployed as a web application using Flask, and it is exposed to the internet via Ngrok. First, the pre-trained XGBoost and CNN models, as well as a list of feature names are loaded. Several functions extract lexical and structural features from URLs, such as length, special characters, subdomains, suspicious keywords, TLD analysis, and structural inconsistencies as shown in

Figure 5.14

```
# Load models
cnn_model = tf.keras.models.load_model('cnn_model.h5')
xgb_model = xgb.Booster()
xgb_model.load_model('xgb_model.json')

# Load feature names
with open('feature_names.json', 'r') as f:
    feature_names = json.load(f)

# Feature extraction functions
def url_length(url):
    return len(url)

def special_characters(url):
    special_chars = ['@', '#', '&', '%', '$', '!', '?', '=', ';']
    return sum([1 for char in special_chars if char in url])

def subdomain_count(url):
    domain = urlparse(url).netloc
    subdomains = domain.split('.')
    return len(subdomains) - 2 if len(subdomains) > 2 else 0

def suspicious_keywords(url):
    suspicious_words = ['login', 'secure', 'verify', 'account', 'update', 'paypal', 'bank']
    return sum([1 for word in suspicious_words if word in url])

def tld_analysis(url):
    try:
        tld = urlparse(url).hostname.split('.')[-1]
        common_tlds = ['com', 'org', 'net', 'co', 'edu', 'gov']
        return 1 if tld in common_tlds else 0
    except:
        return 0

def url_structure_inconsistency(url):
    return 1 if 'https' not in url else 0

def lexical_features(url):
    return 1 if re.search(r'\d+[a-zA-Z]+|\w+\d+', url) else 0

def extract_features(url):
    return np.array([
        url_length(url),
        special_characters(url),
        subdomain_count(url),
        suspicious_keywords(url),
        tld_analysis(url),
        url_structure_inconsistency(url),
        lexical_features(url)
    ]).reshape(1, -1)
```

Figure 5.14: Feature extraction and loading of individual models

The Flask app provides a simple web interface where users can input a URL for phishing detection as shown in Figure 5.15

```
    <!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Phishing Detection</title>
        <style>
            body {{ font-family: Arial, sans-serif; text-align: center; margin: 50px; }}
            form {{ margin-top: 20px; }}
            input, select {{ padding: 10px; width: 300px; margin: 5px; }}
            button {{ padding: 10px; background-color: blue; color: white; border: none; cursor: p
            .result {{ margin-top: 20px; font-size: 20px; }}
        </style>
    </head>
    <body>
        <h2>Phishing Detection System</h2>
        <form method="POST">
            <input type="text" name="url" placeholder="Enter a URL" required>
            <button type="submit">Check</button>
        </form>
        <div class="result">
            <p><strong>URL:</strong> {url}</p>
            <p><strong>CNN Prediction:</strong> {cnn_pred:.4f}</p>
            <p><strong>XGBoost Prediction:</strong> {xgb_pred:.4f}</p>
            <p><strong>Final Verdict:</strong> <span style="color: {'green' if result == 'Legitima
            <form method="POST">
                <input type="hidden" name="url" value="{url}">
                <input type="hidden" name="model_prediction" value="{result}">
                <label>Was this prediction correct?</label>
                <select name="correct_label" required>
                    <option value="">Select</option>
                    <option value="Legitimate">Legitimate</option>
                    <option value="Phishing">Phishing</option>
                </select>
                <button type="submit">Submit Feedback</button>
            </form>
        </div>
    </body>
    </html>
    """
return """<html><body><h2>Phishing Detection System</h2><form method='POST'><input type='text' nam
```

Figure 5.15: Hybrid Webpage code

When a URL is submitted, the system extracts its features and generates predictions using both XGBoost and CNN. If the CNN's confidence level is above a predefined threshold (0.8), its prediction is used; otherwise, the XGBoost prediction is chosen, as shown in Figure 5.16. This approach follows a confidence-based conditional ensemble strategy, where model selection is dynamically guided by the confidence level of the CNN, enabling adaptive and context-aware integration of multiple classifiers.

```python
# Extract features
features = extract_features(url)
features_cnn = features.reshape(1, features.shape[1], 1)

# Get predictions
cnn_pred = cnn_model.predict(features_cnn).flatten()[0]
xgb_pred = xgb_model.predict(xgb.DMatrix(features))[0]

# Hybrid model prediction
threshold = 0.8
final_pred = cnn_pred if cnn_pred >= threshold else xgb_pred
prediction = int(final_pred > 0.5)
result = "Legitimate" if prediction == 1 else "Phishing"
```

Figure 5.16: Hybrid Predict Function

The final result is displayed as either "Legitimate" or "Phishing." If a user disagrees with the model's prediction, they can provide feedback by selecting the correct label. This feedback is stored in a CSV file. When at least 10 feedback entries accumulate, the system automatically retrains both the XGBoost and CNN models using the new data. The XGBoost model is updated with additional boosting rounds, while the CNN undergoes a short training session with the new samples as shown in Figure 5.17.

```python
# Retraining Function
def retrain_models():
    try:
        df = pd.read_csv("feedback_data.csv")

        # Ensure correct column names
        df.columns = ["url"] + feature_names + ["label"]

        if len(df) >= 10:
            X = df[feature_names].values
            y = df["label"].map({"Legitimate": 1, "Phishing": 0}).values  # Match dataset format

            # Retrain XGBoost
            dtrain = xgb.DMatrix(X, label=y)
            params = {'objective': 'binary:logistic', 'eval_metric': 'logloss'}
            new_xgb_model = xgb.train(params, dtrain, num_boost_round=10, xgb_model=xgb_model)
            new_xgb_model.save_model("xgb_model.json")

            # Retrain CNN
            cnn_model.fit(X.reshape(-1, len(feature_names), 1), y, epochs=3, verbose=1)
            cnn_model.save("cnn_model.h5")

            print("Models retrained successfully!")
            df.iloc[10:].to_csv("feedback_data.csv", index=False)  # Remove used data
    except Exception as e:
        print(f"Error in retraining: {e}")
```

Figure 5.17: Hybrid Retraining function

The app runs on port 5000, with Ngrok generating a public URL for external access indicated in Figure 5.18. The implementation integrates real-time feedback collection and model retraining, ensuring continuous improvement of the phishing detection system.

```
if __name__ == '__main__':
    public_url = ngrok.connect(5000)
    print(f"Public URL: {public_url}")
    app.run(port=5000, threaded=True)
```

Figure 5.18: Public Link Function

5.3 Testing

The main Objective of the study was to develop a hybrid model capable of achieving a higher accuracy than their individual counterparts. This was accomplished through integrating the CNN and Xgboost models. The table below summarizes the tests done.

Table 5.1: Hybrid Model Test Cases and Results

Test Case	Priority	Results
Does the application capture the link as submitted?	High	Pass
Dose the application classify links provided correctly?	High	Pass (The models accuracy is 96.63%)
Does the application record the provided feedback?	High	Pass
Does the application Use the provided feedback to train the hybrid model?	High	Pass

Chapter 6: **Discussion**

6.1 Random Forest Training Results

The Random Forest model was trained to classify phishing and legitimate URLs using a dataset containing phishing and legitimate URLs. Its performance was evaluated using cross-validation, test accuracy, and various classification metrics.

The cross-validation accuracy was 86.54%, indicating the model's consistency across different training and validation splits. When tested on unseen data, the model achieved an 86.42% accuracy, meaning it correctly classified URLs as phishing or legitimate in about 86% of cases.

Precision and recall metrics were calculated for both phishing and legitimate URLs. For phishing URLs (Class 0), the model had a precision of 96%, meaning that when it predicted a URL as phishing, it was correct 96% of the time. However, its recall for phishing URLs was 76%, indicating that it correctly identified 76% of all actual phishing URLs. For legitimate URLs (Class 1), the model had a precision of 80%, meaning that when it predicted a URL as legitimate, it was correct 80% of the time. The recall for legitimate URLs was 97%, showing that the model successfully detected 97% of all actual legitimate URLs.

The F1-scores, which balance precision and recall, were 85% for phishing URLs and 88% for legitimate URLs as indicated in figure 6.1. This suggests that while the model is highly effective at detecting legitimate URLs, it is more likely to misclassify phishing URLs as legitimate than the other way around which is not ideal for phishing detection.

```
Average Prediction Time per URL: 0.000007 seconds
Performance Metrics:
              precision    recall  f1-score   support

           0       0.96      0.76      0.85     20068
           1       0.80      0.97      0.88     20067

    accuracy                           0.86     40135
   macro avg       0.88      0.86      0.86     40135
weighted avg       0.88      0.86      0.86     40135


Summary of Results:
Random Forest Cross-Validation Accuracy: 0.865422
Random Forest Accuracy: 0.864233
Average Prediction Time per URL: 0.000007 seconds
```

Figure 6.1: Random Forest Performance Metrics

The Receiver Operating Characteristic (ROC) curve was plotted to analyse the trade-off between true positive and false positive rates, and the Area Under the Curve (AUC) score was 0.96, indicating strong discrimination between the two classes as indicated in Figure 6.2.

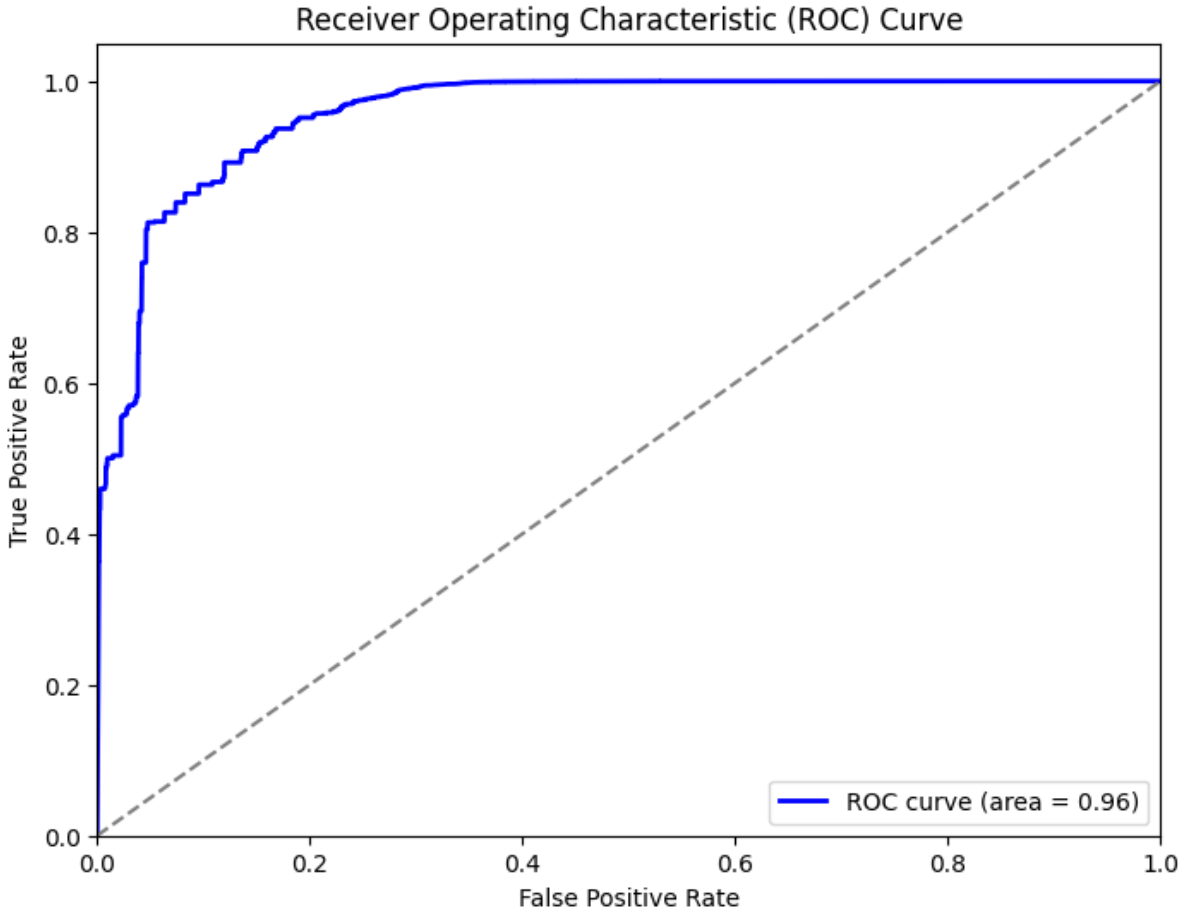


Figure 6.2: Random Forest ROC

The confusion matrix in Figure 6.3 shows that the model performs well in detecting phishing URLs, correctly classifying 15,402 phishing and 11,986 legitimate URLs. However, it also misclassifies 3,923 legitimate URLs as phishing (false positives). Meanwhile, false negatives are relatively low at 507, meaning the model rarely misses actual threats.

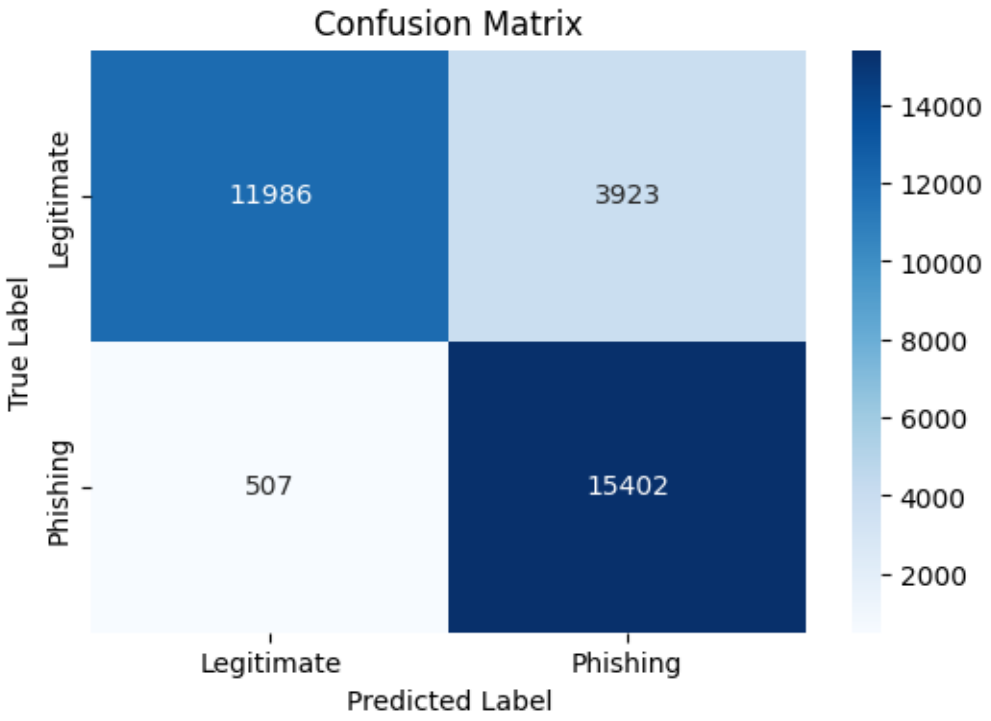


Figure 6.3: Random Forest Confusion Matrix

Finally, the average prediction time per URL was 0.000007 seconds, highlighting the model's efficiency in processing large volumes of URLs quickly and real time detection.

6.2 Decision Tree Model Training Results

The Decision Tree model achieved a cross-validation accuracy of 87.22% and a test accuracy of 87.69% as indicated in Figure 6.4, indicating a strong predictive capability.

```
Average Prediction Time per URL: 0.000000 seconds
Performance Metrics:
              precision    recall  f1-score   support

           0       0.94      0.81      0.87     20068
           1       0.83      0.95      0.89     20067

    accuracy                           0.88     40135
   macro avg       0.89      0.88      0.88     40135
weighted avg       0.89      0.88      0.88     40135


Summary of Results:
Decision Tree Cross-Validation Accuracy: 0.879244
Decision Tree Accuracy: 0.877189
Average Prediction Time per URL: 0.000000 seconds
```

Figure 6.4: Decision Tree Performance Metrics

From the confusion matrix, the model correctly classifies 6886 legitimate URLs and 11028 phishing URLs. However, it misclassifies 1699 legitimate URLs as phishing (false negatives), which may cause inconvenience to legitimate users. Additionally, it incorrectly classifies 815 phishing URLs as legitimate (false positives), which poses a security risk by allowing malicious URLs to go undetected.

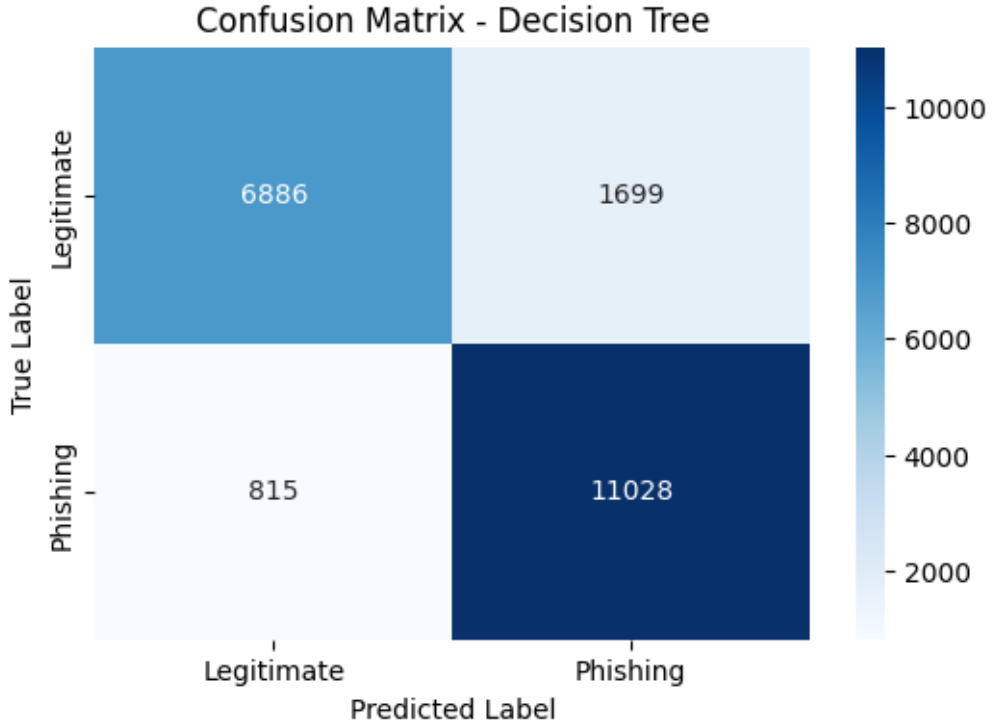


Figure 6.5: Decision Tree Confusion Matrix

The classification report further supports this, showing a precision of 89% for legitimate URLs and 87% for phishing URLs, meaning the model is slightly better at correctly identifying legitimate URLs. The recall for legitimate URLs is 80%, meaning 20% of legitimate URLs are mistakenly flagged as phishing, while phishing URLs have a recall of 93%, showing that most phishing sites are successfully detected.

The ROC curve, with an AUC of 0.92, which also indicates a strong discrimination between legitimate and phishing URLs as shown in Figure 6.6.

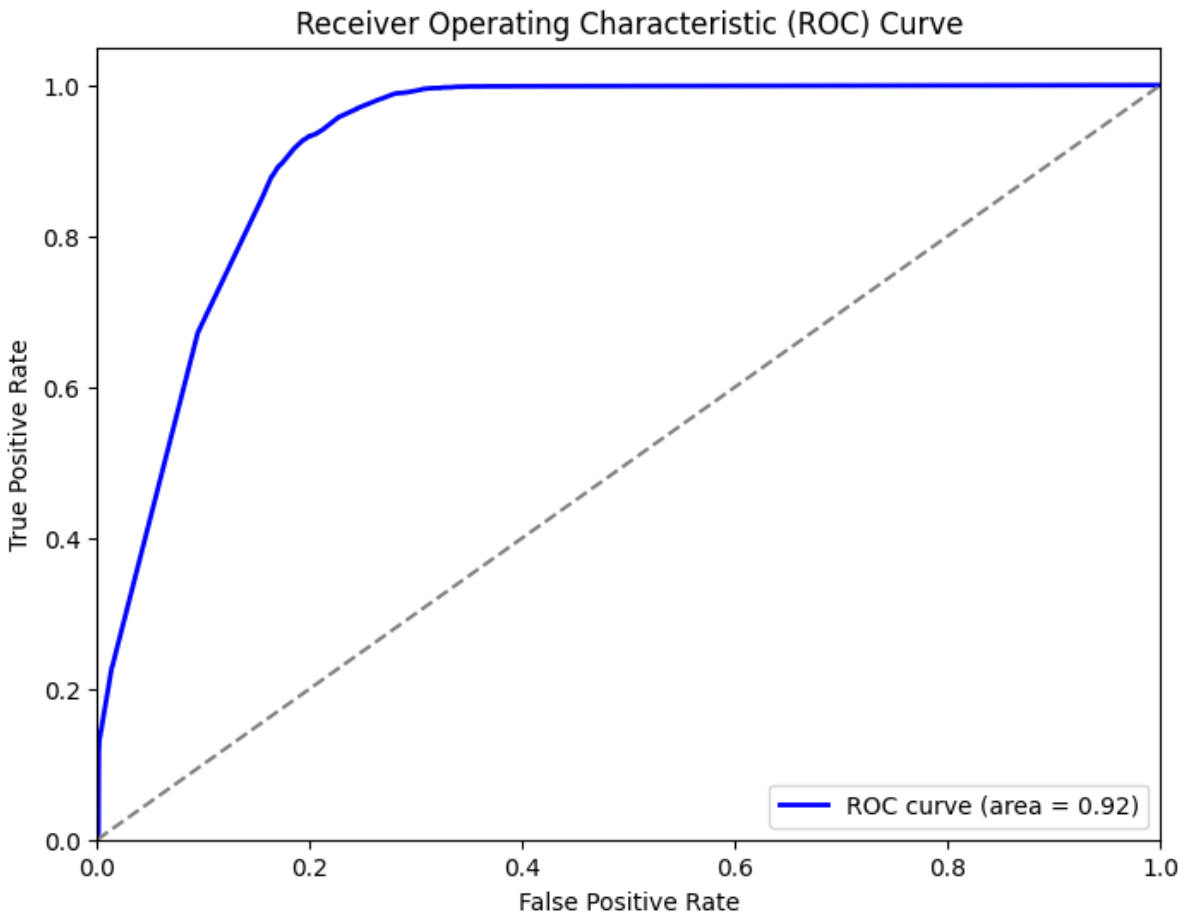


Figure 6.6: Decision Tree ROC

Additionally, the model demonstrated impressive computational efficiency, with an average prediction time per URL of 0.000000 seconds, making it highly suitable for real-time phishing detection. However, the false negative rate (legitimate sites classified as phishing) suggests a potential need for fine-tuning, such as adjusting class weights, pruning the tree, or exploring more advanced models.

6.3 XGBoost Model Training Results

The results from the XGBoost model demonstrate exceptional performance in phishing detection, with high accuracy, precision, and recall. The confusion matrix in Figure 6.7 reveals that the model correctly identified 18,479 phishing URLs and 26,592 legitimate URLs, while misclassifying only 1,710 phishing URLs as legitimate and 378 legitimate URLs as phishing. These figures indicate a low false positive and false negative rate, highlighting the model's strong reliability in distinguishing between phishing and legitimate websites.

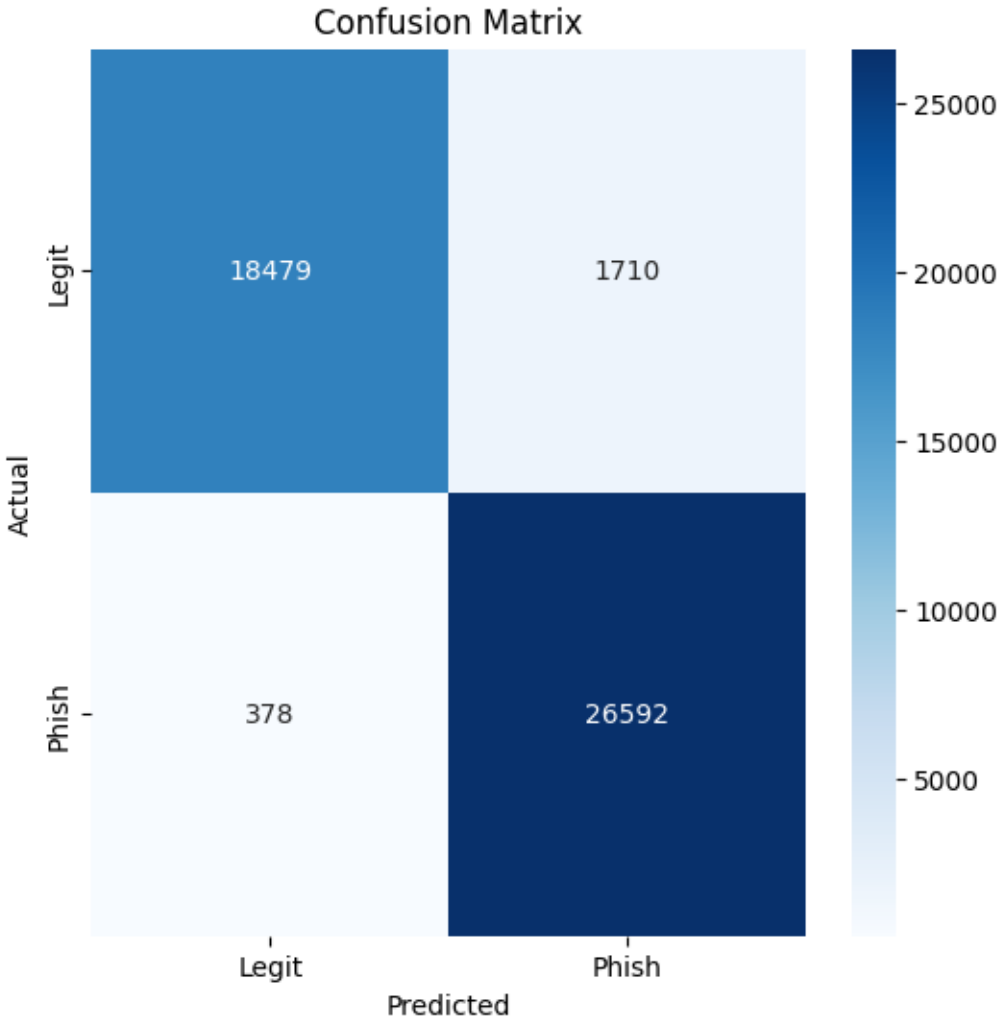


Figure 6.7: XGBoost Confusion Matrix

Further reinforcing its effectiveness, the Receiver Operating Characteristic (ROC) curve in Figure 6.8 exhibits an impressive Area Under the Curve (AUC) of 0.98, suggesting near-perfect discriminative power. This means the model is highly capable of accurately differentiating between the two classes.

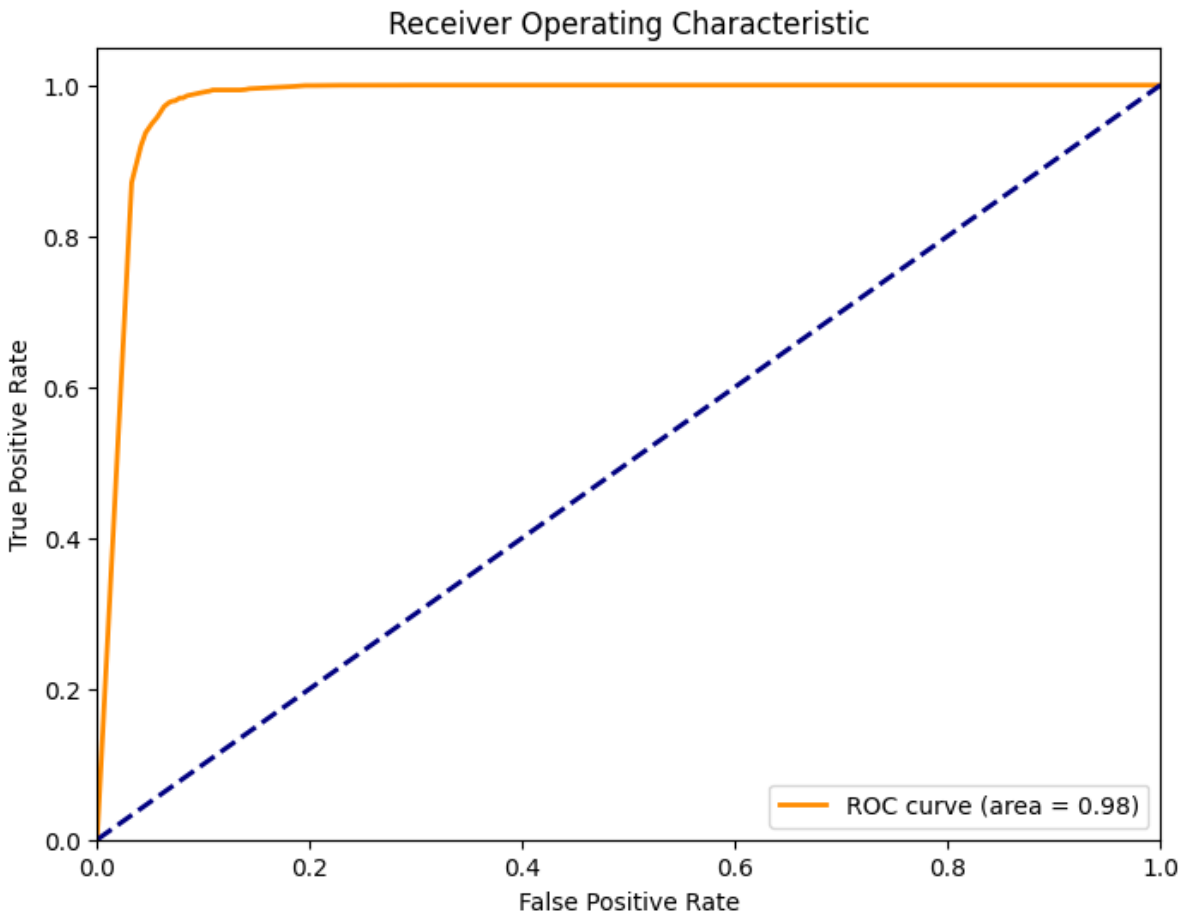


Figure 6.8: XGBoost ROC

The classification report provides deeper insights into the model's predictive capabilities. The precision for phishing detection stands at 98%, meaning the model rarely classifies legitimate websites as phishing. Conversely, the precision for legitimate sites is 95%, ensuring that most predictions for non-malicious sites are correct. The recall values further support this robustness, with 94% for phishing and 99% for legitimate websites, demonstrating the model's ability to correctly detect almost all instances of both classes. The F1-scores, which balance precision and recall, are also outstanding, scoring 96% for phishing and 97% for legitimate sites.

Overall, the model achieves an accuracy of 95.57%, confirming its reliability in real-world applications. The macro average metrics, with 96% precision, 95% recall, and 95% F1-score, indicate that the model performs consistently across both classes, while the weighted averages, also at 96% as shown in Figure 6.9, account for any class imbalance and ensure stability in predictions.

```
Evaluation of the Final XGBoost Model on the Test Set:
Accuracy: 0.9557
ROC AUC: 0.9785
Confusion Matrix:
 [[18479  1710]
 [  378 26592]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.92      0.95     20189
           1       0.94      0.99      0.96     26970

    accuracy                           0.96     47159
   macro avg       0.96      0.95      0.95     47159
weighted avg       0.96      0.96      0.96     47159
```

Figure 6.9: XGBoost Performance Metrics

Beyond accuracy, the model also excels in efficiency, with an average prediction time of just 0.001451 seconds per URL. This rapid prediction speed makes it highly suitable for real-time deployment.

6.4 Convocational Neural Network Model Training Results

The evaluation of the CNN model for phishing detection demonstrates a strong classification performance, though it lags behind XGBoost in some areas. The model achieved an accuracy of 91%, indicating its capability to differentiate phishing from legitimate URLs effectively.

From the confusion matrix in Figure 6.10, the CNN model correctly classified 18,667 phishing URLs but misclassified 1,521 phishing URLs as legitimate, leading to a false negative rate. This misclassification is crucial since incorrectly labelling phishing websites as legitimate poses a security risk. On the other hand, the model accurately classified 24,454 legitimate URLs, but 2,515 legitimate URLs were misclassified as phishing, resulting in a false positive rate.

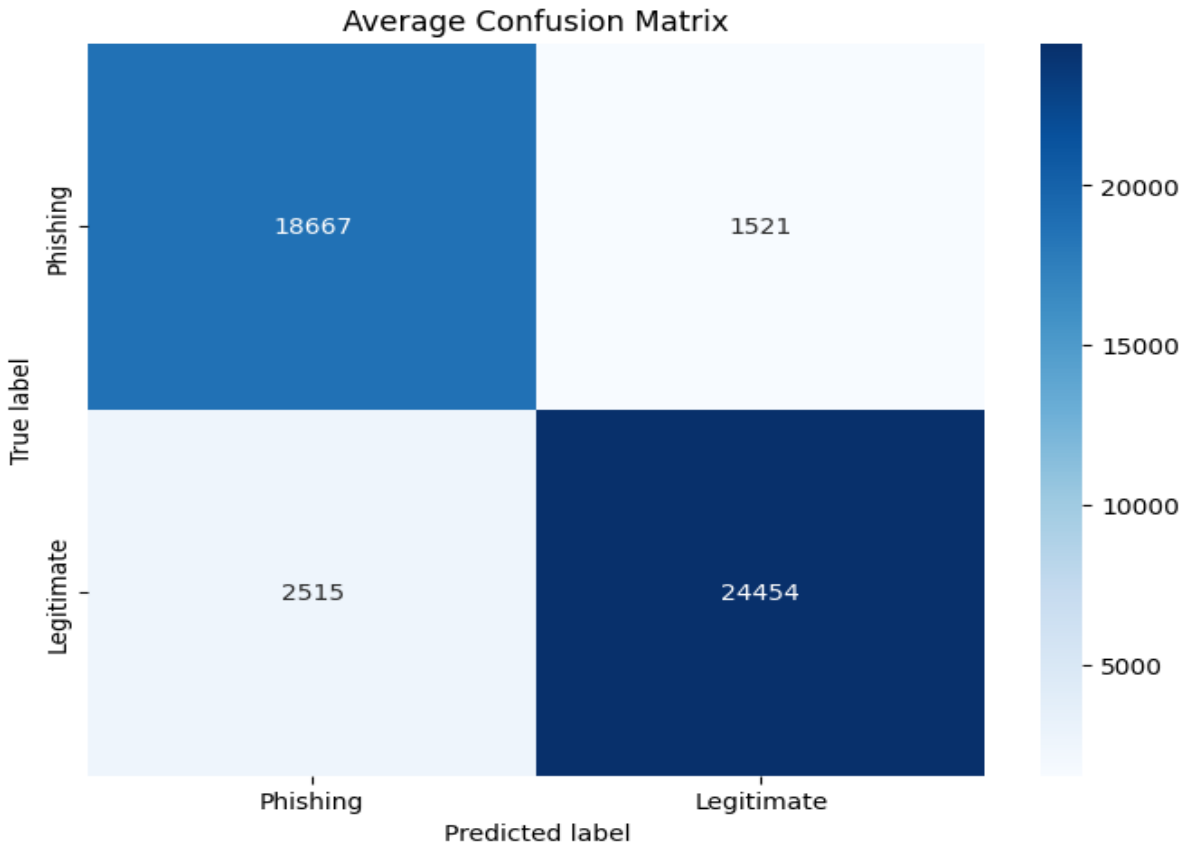


Figure 6.10: CNN Confusion Matrix

The ROC curve in Figure 6.11 shows an AUC of approximately 0.91, indicating good overall classification ability but slightly lower than XGBoost, which had an AUC of 0.98.

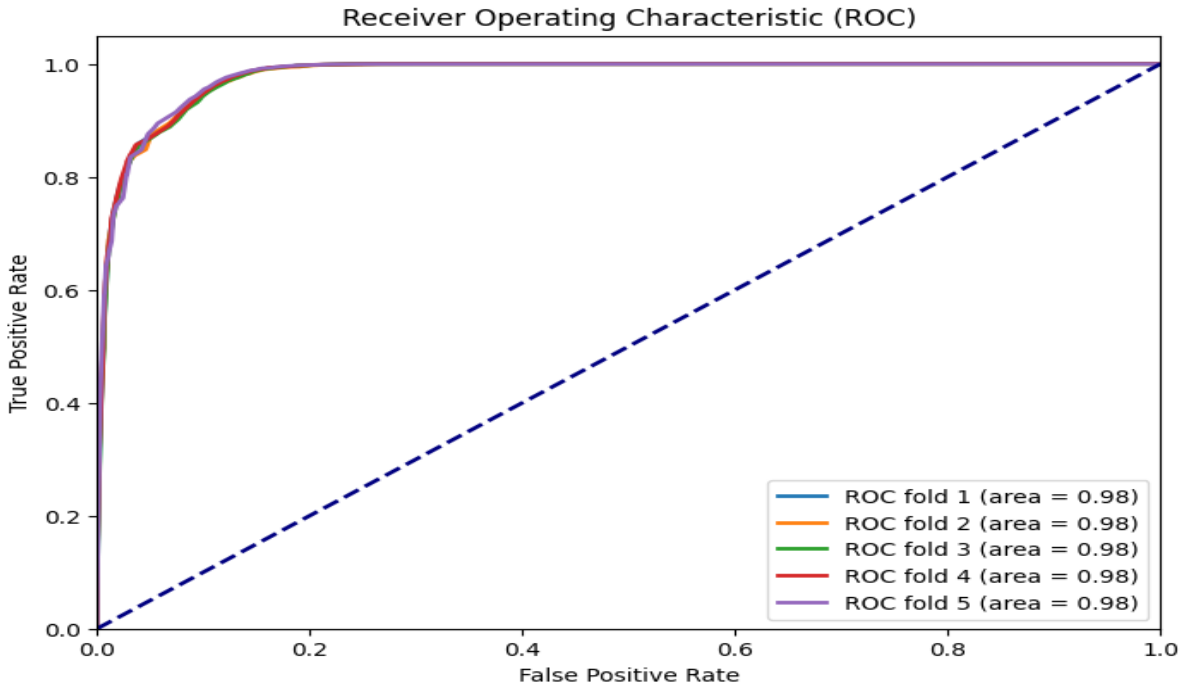


Figure 6.11: CNN ROC

The performance metrics further confirm this, with a precision of 0.88 for phishing URLs and 0.94 for legitimate URLs. The recall values of 0.92 for phishing and 0.91 for legitimate URLs indicate the model's ability to correctly identify true cases, while the F1-scores of 0.90 and 0.92 as shown in figure 6.12 reflect a well-balanced performance.

Performance Metrics:

	precision	recall	f1-score	support
0	0.88	0.92	0.90	100945
1	0.94	0.91	0.92	134850
accuracy			0.91	235795
macro avg	0.91	0.92	0.91	235795
weighted avg	0.92	0.91	0.91	235795

Figure 6.12: CNN Performance Metrics

6.5 SVM Model Training Results

The SVM model for phishing detection achieved an accuracy of 84.53%, demonstrating a strong ability to classify phishing and legitimate websites. The confusion matrix in Figure 6.13 shows that the model correctly identified 15,485 phishing samples and 12,752 legitimate samples, but misclassified 3,950 legitimate sites as phishing and 1,216 phishing sites as

legitimate. This indicates that while the model is highly effective at detecting phishing sites, it struggles somewhat with false positives, mistakenly flagging legitimate sites as malicious.

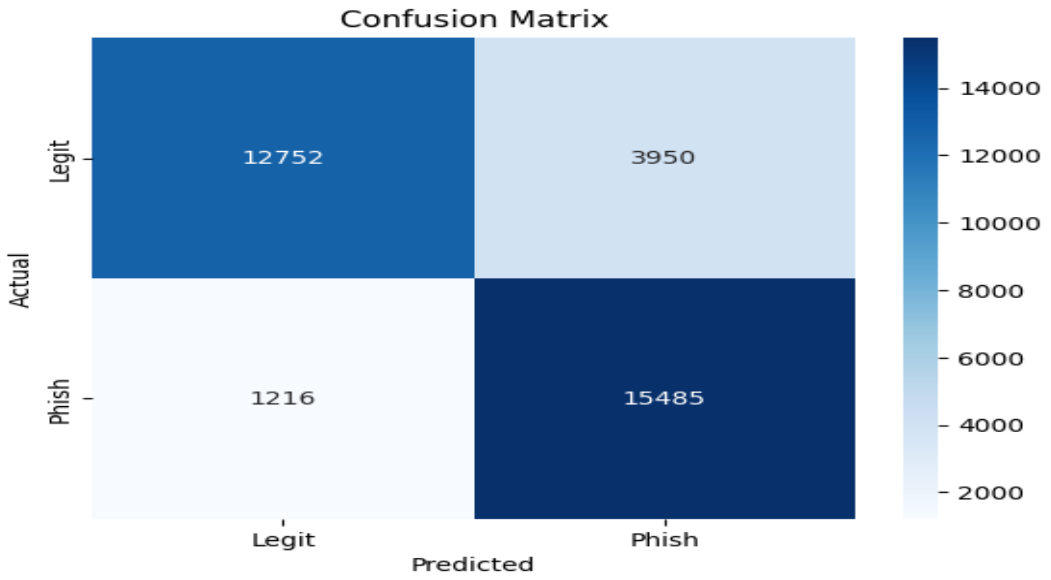


Figure 6.13: SVM Confusion Matrix

Looking at precision and recall, the model exhibits 91% precision for legitimate sites and 80% precision for phishing sites, meaning that when it predicts a website as legitimate, it is correct 91% of the time, and for phishing, it is correct 80% of the time. However, recall values tell a different story 76% for legitimate sites and 93% for phishing as indicated in Figure 6.14, showing that the model is more confident in identifying phishing sites but occasionally misses legitimate ones.

```
SVM Accuracy: 0.8453432326437745
              precision    recall  f1-score   support

           0       0.91      0.76      0.83     16702
           1       0.80      0.93      0.86     16701

    accuracy                           0.85     33403
   macro avg       0.85      0.85      0.84     33403
weighted avg       0.85      0.85      0.84     33403
```

Figure 6.14: SVM Performance Metrics

The ROC curve as shown in Figure 6.15 further supports the model's effectiveness, with an AUC of 0.92, indicating a strong ability to distinguish between phishing and legitimate websites. However, the trade-off between false positives and false negatives suggests room for improvement.

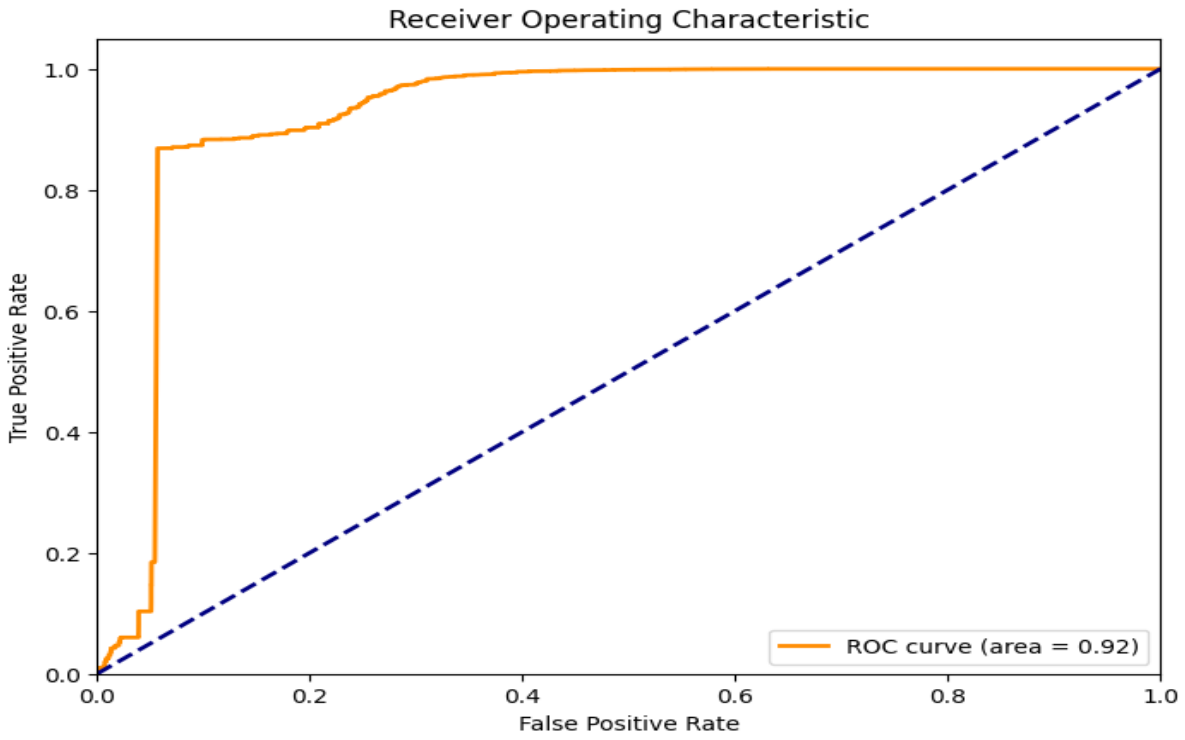


Figure 6.15: SVM ROC

6.6 Algo Results Summary

Table 6.1: Individual Algorithm Results Summary

Model	**Overall Accuracy (%)**	**Precision (class 1) (%)**	**Precision (class O) (%)**	**Recall (Class 1) (%)**	**Recall (Class 0) (%)**	**F1-Score (Class 1) (%)**	**F1-Score (Class 0) (%)**	**Prediction Time per URL (ms)**
CNN	91	94	88	91	92	92	90	0.079803
XGBoost	95.57	94	98	99	92	96	95	0.00145
Decision Tree	88	87	89	93	90	90	85	0.000000
Random Forest	86	88	88	86	86	86	86	0.000007
SVM	85	80	91	93	76	86	83	

Base on the results, a hybrid model with CNN as the base and XGBoost as the secondary predictor was chosen, driven by the strengths and trade-offs observed in their individual performances. CNN, while slightly less accurate overall (91%), demonstrated strong recall for both legitimate (91%) and phishing (92%) URLs, making it a reliable primary model for classification. However, its prediction time (0.0798 milliseconds per URL) is relatively higher than XGBoost, which is significantly faster at 0.00145 milliseconds per URL.

XGBoost, despite its higher overall accuracy (95.57%), performed exceptionally well in precision for phishing detection (98%) but showed a slight imbalance in recall, particularly for legitimate URLs (92%). This means it might occasionally misclassify legitimate websites as phishing. By using CNN as the primary model, we ensure a strong and balanced recall across both classes, reducing the risk of biased classifications. XGBoost serves as a secondary validator, stepping in when CNN's confidence in its prediction falls below a predefined threshold (0.80). This setup allows us to harness the strengths of both models CNN's feature extraction capabilities and XGBoost's speed and precision resulting in a more robust phishing detection system.

6.7 Hybrid Model Results

The evaluation of the hybrid phishing detection model, which combines a tuned XGBoost and CNN, demonstrates strong performance across multiple metrics. The model achieved an accuracy of 96.63%, with a precision of 95.50%, recall of 98.77%, and an F1-score of 97.11% as shown in Figure 6.16.

```
Evaluation of the Hybrid Model (with Tuned XGBoost and CNN):
Accuracy: 0.9663
Precision: 0.9550
Recall: 0.9877
F1-score: 0.9711
ROC AUC (approx.): 0.9628
Confusion Matrix:
 [[18934  1255]
 [  332 26638]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.94      0.96     20189
           1       0.96      0.99      0.97     26970

    accuracy                           0.97     47159
   macro avg       0.97      0.96      0.97     47159
weighted avg       0.97      0.97      0.97     47159


Time taken to predict one URL: 0.111962 seconds
1474/1474 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step
```

Figure 6.16: Hybrid Model Performance Metrics

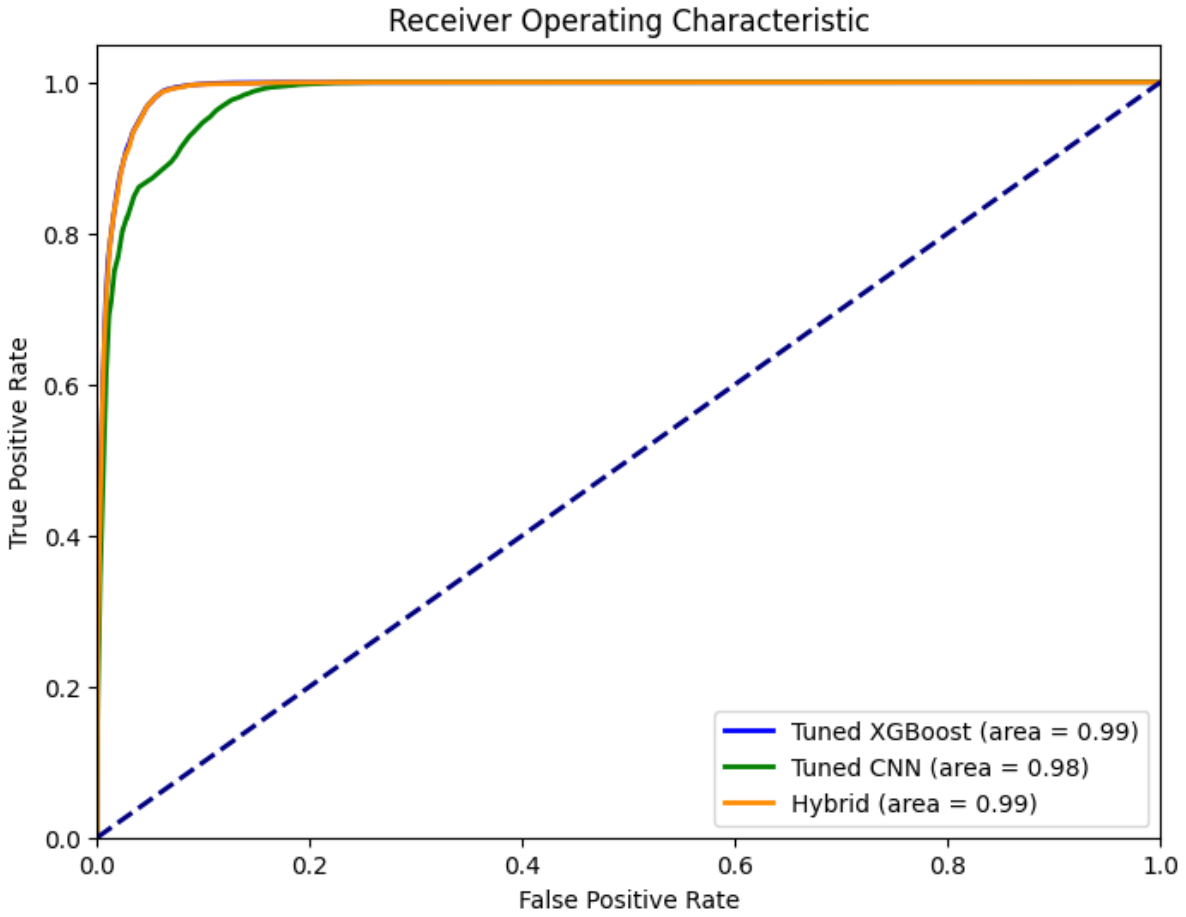


Figure 6.17: Hybrid Model ROC.

A comparison of ROC curves in Figure 6.17 highlights the superiority of the hybrid model over the individual tuned XGBoost and CNN classifiers, as it maintains a high AUC of 0.99, closely matching XGBoost while outperforming the CNN. The threshold hybrid approach effectively balances precision and recall, leveraging the strengths of both models.

The confusion matrix in Figure 6.18 reveals that the model correctly identified 18,934 legitimate URLs while misclassifying 1,255 as phishing. Similarly, it accurately detected 26,638 phishing URLs, with only 332 false negatives. This suggests that the model is particularly effective in identifying phishing attacks, as indicated by its high recall value.

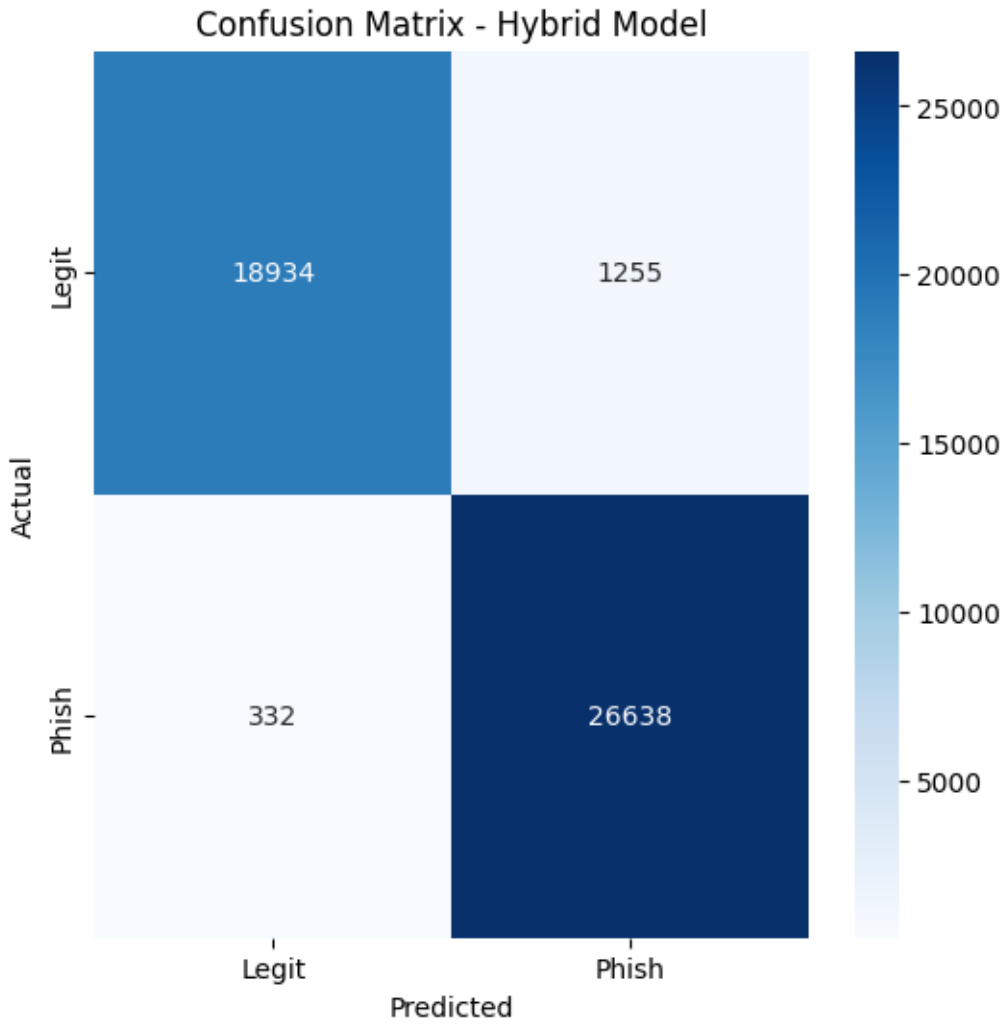


Figure 6.18: Hybrid Model Confusion Matrix.

Additionally, the model exhibits efficient performance, taking approximately 0.112 seconds to predict a single URL as shown in Figure 6.16. This suggests that it is well-suited for real-time phishing detection applications, providing both accuracy and speed in identifying threats.

6.7.1 Real time classification

The system is deployed as a web application using Flask, and it is exposed to the internet via Ngrok. Once the link to the app is opened it directs the user to a webpage where they can input a URL and receive classification as either legitimate or phishing as shown in Figure 6.19 and 6.20 respectively and if the prediction is wrong a user can flag wrong prediction and label it with the correct one ie either “Legitimate” or “Phishing” as show in 6.21

Figure 6.19: Legitimate URL prediction

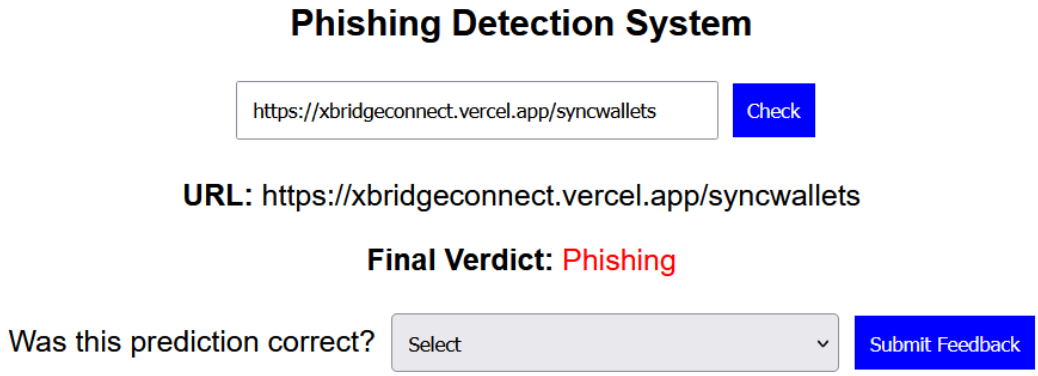


Figure 6.20: Phishing URL Prediction

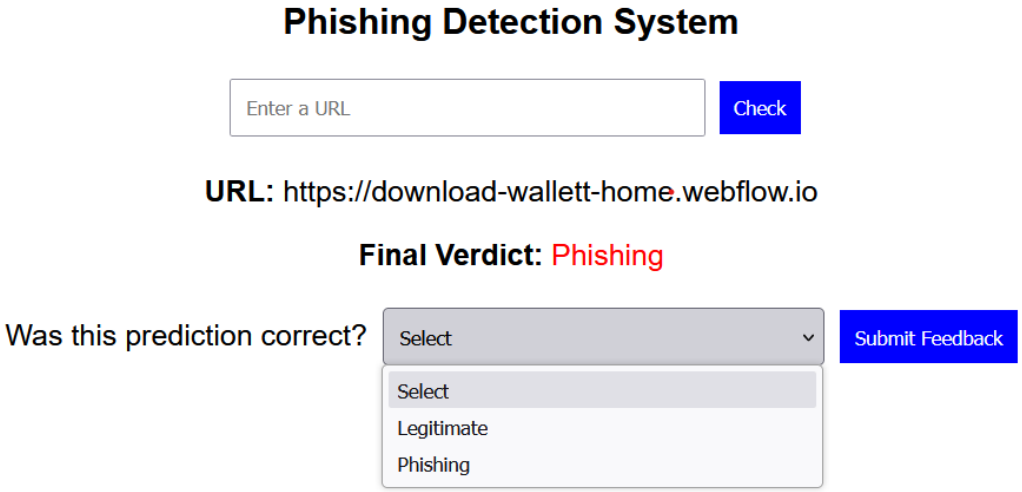


Figure 6.21: Wrong prediction feedback selection

6.8 Comparison with individual models

The performance of the proposed hybrid model was benchmarked against several individual machine learning algorithms, including Convolutional Neural Networks (CNN), XGBoost, Decision Trees, Random Forests, and Support Vector Machines (SVM). The comparison was conducted using standardized evaluation metrics such as Accuracy, Precision, Recall, F1-Score, and Prediction Time per URL. These metrics were chosen due to their effectiveness in measuring the classification performance and computational efficiency of machine learning models in phishing URL detection.

Table 6.2 All model results

Model	**Overall Accuracy (%)**	**Precision (class 1) (%)**	**Precision (class O) (%)**	**Recall (Class 1) (%)**	**Recall (Class 0) (%)**	**F1-Score (Class 1) (%)**	**F1-Score (Class 0) (%)**	**Prediction Time per URL**
Hybrid model	96.63	96	98	99	94	97	96	0.1119
CNN	91	94	88	91	92	92	90	0.079803
XGBoost	95.57	94	98	99	92	96	95	0.00145
Decision Tree	88	87	89	93	90	90	85	0.000000
Random Forest	86	88	88	86	86	86	86	0.000007
SVM	85	80	91	93	76	86	83	

As illustrated in Table 6.2, among all the models tested, the hybrid model achieved the highest overall accuracy at 96.63%, outperforming both XGBoost (95.57%) and CNN (91%). The hybrid approach also maintained a strong balance between precision and recall across both classes, notably achieving a phishing class (Class 1) recall of 99% and a legitimate class (Class 0) precision of 98%. These metrics are critical in phishing detection systems where misclassifying a phishing URL as legitimate can have severe consequences. Despite its slightly

higher computational complexity, the hybrid model's average prediction time of 0.1119 milliseconds per URL remained within acceptable limits for real-time detection tasks.

The CNN model demonstrated strong phishing detection capabilities, achieving a precision of 94% and a recall of 91% for Class 1. However, its performance on legitimate URLs was less robust, with a lower precision of 88%, which may lead to an increased number of false positives. This reflects a common trade-off in deep learning models trained on imbalanced or complex feature sets, where the emphasis on identifying malicious content may compromise performance on benign data.

XGBoost emerged as the best-performing individual model, closely trailing the hybrid system. It matched the hybrid model in recall for phishing URLs at 99% and surpassed others in precision for legitimate URLs at 98%. Additionally, it achieved a well-balanced F1-score and a remarkably low prediction time of just 0.00145 milliseconds per URL, making it especially suitable for high-speed environments where rapid classification is essential.

The Decision Tree and Random Forest models exhibited moderate classification capabilities. While Decision Tree reached a recall of 93% for phishing URLs and a reasonable F1-score, its overall accuracy remained comparatively low at 88%. Similarly, Random Forest posted balanced but unremarkable scores across most metrics, achieving 86% in both precision and recall categories. These models, however, had a significant advantage in terms of computational efficiency, with Decision Tree achieving an effectively instantaneous prediction time of 0.000 milliseconds and Random Forest registering 0.000007 milliseconds. Such efficiency may be beneficial in scenarios requiring rapid throughput over large URL volumes, though it comes at the cost of predictive accuracy.

The SVM model showed promising recall for phishing URLs at 93% but fell short in other areas, particularly in terms of Class 1 precision (80%) and Class 0 recall (76%). This imbalance suggests a tendency to misclassify legitimate URLs as phishing, resulting in a higher rate of false positives. While SVM remains a powerful classifier in high-dimensional spaces, its sensitivity to class imbalance limits its effectiveness in phishing URL detection when used in isolation.

In summary, while individual models like XGBoost and CNN performed well in specific areas, the hybrid model demonstrated superior overall performance by effectively combining the strengths of its constituent algorithms. It achieved the best balance between detection accuracy,

precision, recall, and real-time prediction latency, confirming its suitability for deployment in practical phishing detection systems where reliability and responsiveness are equally critical.

Chapter 7: **Conclusion Challenges and Future work**

7.1 Conclusion

This study was successful in developing and evaluating a hybrid phishing detection model that achieved a higher accuracy than the individual models. It combines Convolutional Neural Networks (CNN) and XGBoost. The integration of these models was aimed at leveraging their individual strengths to improve phishing detection accuracy. CNN served as the primary classifier, with XGBoost acting as a secondary validator for uncertain predictions. The hybrid approach successfully enhanced classification performance by maintaining high accuracy, precision, and recall.

The performance metrics demonstrated the hybrid model's superiority over individual classifiers. The hybrid model achieved an accuracy of 96.63%, precision of 95.50%, recall of 98.77%, and an F1-score of 97.11%. Comparing the hybrid model's accuracy with other models, CNN achieved an accuracy of 91%, XGBoost 95.57%, Decision Tree 88%, Random Forest 86%, and SVM 85%. The hybrid model surpassed these, achieving 96.63% accuracy, demonstrating its improved detection capability. In terms of precision, XGBoost had the highest at 98% for class 0, while CNN achieved 94% for class 1. The hybrid model balanced both classes effectively with 95.50% precision overall. Additionally, the hybrid model maintained high recall at 98.77%, outperforming Decision Trees (93%) and Random Forest (86%). Furthermore, the model exhibited a competitive prediction time per URL, ensuring real-time detection feasibility.

7.2 Challenges

While the hybrid model demonstrated strong performance, some challenges were encountered during the study:

i. Computational Complexity – The CNN component required significant computational power, leading to longer training times compared to traditional machine learning models.
ii. False Positives – Despite high precision, the model occasionally misclassified legitimate URLs as phishing, which could result in unnecessary website restrictions.
iii. Data Imbalance – The dataset contained a higher number of phishing URLs than legitimate ones, which may have influenced classification bias despite efforts to balance training samples.

7.3 Recommendations

To further improve the effectiveness and efficiency of the phishing detection system, the following recommendations are proposed:

i. Optimize Model Efficiency – Employ techniques such as model pruning and quantization to reduce computational costs and improve prediction speed.
ii. Enhance Feature Engineering – Incorporate additional URL-based and content-based features to improve classification performance.
iii. Deploy in a Scalable Environment – Implement the system on cloud platforms such as AWS or Google Cloud to ensure scalability and continuous availability.

7.4 Suggested Feature Work

Future research should explore additional enhancements to the phishing detection model, including:

i. Integration of NLP Techniques – Applying Natural Language Processing (NLP) to analyse webpage content and metadata for better phishing detection.
ii. Incorporation of Graph-Based Analysis – Using graph-based models to examine relationships between domains and detect phishing networks.
iii. Hybridization with Reinforcement Learning – Exploring reinforcement learning for adaptive decision-making in phishing classification.
iv. Real-Time Browser Extension Development – Implementing the model as a browser extension for on-the-fly phishing detection while users browse the web.
v. Collaboration with Cybersecurity Experts – Partnering with cybersecurity professionals to continuously refine the detection model based on real-world attack trends.

By addressing these areas, phishing detection models can continue evolving to combat increasingly sophisticated cyber threats. The proposed hybrid model serves as a significant step towards achieving a more robust and adaptive phishing detection framework.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16) (pp. 265-283).

ACM. (2023). A Hybrid CNN-LSTM Based Approach for Anomaly Detection Systems in Software Defined Networks. *ACM Digital Library*. https://dl.acm.org/doi/fullHtml/10.1145/3465481.3469190 Opast Publishers. (2023). Phishing URL Detection Using CNN-LSTM and Random Forest Classifier. *International Journal of Medical Networks*, 2(5), 1-6. Retrieved from https://www.opastpublishers.com/open-access-articles/phishing-url-detection-using-cnnlstm-and-random-forest-classifier.pdf

Al mujahid, N F., Haq, M A., & Alshehri, M S. (2024, June 24). Comparative evaluation of machine learning algorithms for phishing site detection. PeerJ, Inc., 10, e2131-e2131. https://doi.org/10.7717/peerj-cs.2131

Alanezi, M. (2021). Phishing Detection Methods: A Review. Technium, 3(9), 19-35.

Aldakheel, E A., Zakariah, M., Gashgari, G A., Almarshad, F A., & Alzahrani, A I A. (2023, April 30). A Deep Learning-Based Innovative Technique for Phishing Detection in Modern Security with Uniform Resource Locators. Multidisciplinary Digital Publishing Institute, 23(9), 4403-4403. https://doi.org/10.3390/s23094403

Al hassan, A., Alhassan, M., & Abubakar, M. (2023). A hybrid deep learning technique for spoofing website URL detection in real-time applications. *Journal of Electrical Systems and Information Technology*, 10(1), 1-15.

Aljofey, A., Jiang, Q., Qu, Q., Huang, M., & Niyigena, J. (2020, September 15). An Effective Phishing Detection Model Based on Character Level Convolutional Neural Network from URL. Multidisciplinary Digital Publishing Institute, 9(9), 1514-1514. https://doi.org/10.3390/electronics9091514

A mutair, M., & Alshoshan, A. (2023). Modeling Hybrid Feature-Based Phishing Websites Detection Using Machine Learning Techniques. *Journal of Cybersecurity*, 12(13), 2823.

American Psychological Association. (2017). *APA Policy: Applied Behavior Analysis*. Retrieved from https://www.apa.org/about/policy/applied-behavior-analysis

Andriekutė, A. (2023b, December 19). What is URL phishing? Everything you need to know. NordVPN. https://nordvpn.com/blog/what-is-url-phishing/

Avetisyan, A. (2023, September 20). *Phishing Website Detection – An Ongoing Battle*. EasyDMARC. https://easydmarc.com/blog/phishing-website-detection-an-ongoing-battle/

Baker, K. (2024, October 2). *12 most common types of cyberattacks today - CrowdStrike*. crowdstrike.com. https://www.crowdstrike.com/cybersecurity-101/cyberattacks/most-common-types-of-cyberattacks/

Boaz, A., & Ashby, D. (2003). Standards of a quality research. *Journal of Research Evaluation*, 8(1), 1-9.

Britannica. (n.d.). Data analysis. In *Encyclopedia Britannica*. Retrieved from https://www.britannica.com/science/data-analysis

Chintala, S., et al. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. In *Proceedings of NeurIPS*.

Cobb, M., & Brush, K. (n.d.). *What is cybercrime and how can you prevent it?* Security. https://www.techtarget.com/searchsecurity/definition/cybercrime.

Computational Efficiency: Methods & Impact. (n.d.). StudySmarter UK. https://www.studysmarter.co.uk/explanations/engineering/artificial-intelligence-engineering/computational-efficiency/

Coursera.org. (2024). What Is Keras? Your 2024 Guide. Retrieved from https://www.coursera.org/articles/what-is-keras

Craig, L. (n.d.). *convolutional neural network (CNN)* (R. Awati, Ed.). TechTarget. https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network

Creswell, J. W., & Creswell, J. D. (2018). *Research design: Qualitative, quantitative, and mixed methods approaches* (5th ed.). Sage Publications.

Dalvi, S., Gressel, G., & Achuthan, K. (2019, December 30). Tuning the False Positive Rate / False Negative Rate with Phishing Detection Models. , 9(1S5), 7-13. https://doi.org/10.35940/ijeat.a1002.1291s52019

Das Guptta, S., Shahriar, K. T., Alqahtani, H., Alsalman, D., & Sarker, I. H. (2022). Modeling hybrid feature-based phishing websites detection using machine learning techniques. *Springer*.

Das, M., Saraswathi, S., Panda, R., Mishra, A K., & Tripathy, A K. (2020, September 11). Exquisite Analysis of Popular Machine Learning–Based Phishing Detection Techniques for Cyber Systems. Taylor & Francis, 16(4), 538-562. https://doi.org/10.1080/19361610.2020.1816440

Feng, J., Zou, L., Ye, O., & Han, J. (2020, January 1). Web2Vec: Phishing Webpage Detection Method Based on Multidimensional Features Driven by Deep Learning. Institute of Electrical and Electronics Engineers, 8, 221214-221224. https://doi.org/10.1109/access.2020.3043188

Gaurav, A., & Chui, J.K.T. (2023). Convolution Neural Network (CNN) Based Phishing Attack Detection. Retrieved from https://iceb.johogo.com/proceedings/2023/ICEB2023_paper_26.pdf

GeeksforGeeks. (2024). What is Keras? Retrieved from https://www.geeksforgeeks.org/what-is-keras/

Grad Coach. (2024). What is research methodology? Definition + examples. Retrieved from https://gradcoach.com/what-is-research-methodology/

Gupta, B. B., Gaurav, A., & Chui, K. T. (2023). Convolutional Neural Network (CNN) based phishing attack detection. In *The 23rd International Conference on Electronic Business* (pp. 725-730). Montclair, NJ, USA. Retrieved from https://iceb.johogo.com/proceedings/2023/ICEB2023_paper_26.pdf

Carlini, N., & Wagner, D. (2017). Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 1-12.

Guptta, S D., Shahriar, K T., Alqahtani, H., Alsalman, D., & Sarker, I H. (2022, March 21). Modeling Hybrid Feature-Based Phishing Websites Detection Using Machine Learning Techniques. Springer Science+Business Media, 11(1), 217-242. https://doi.org/10.1007/s40745-022-00379-8

Gurung, H., Nepal, R., & Nepal, S. (2023). Phishing URL Detection Using CNN-LSTM and Random Forest Classifier. *International Journal of Media and Networks*, 2(5), 1-6. Retrieved from https://www.opastpublishers.com/open-access-articles/phishing-url-detection-using-cnnlstm-and-random-forest-classifier.pdf

Hu, S. (2014). Study population. In A.C. Michalos (Ed.), *Encyclopedia of Quality of Life and Well-Being Research*. Springer, Dordrecht. https://doi.org/10.1007/978-94-007-0753-5_2893

IBM. (2023, December 27). What are support vector machines (SVMs)? *IBM*. https://www.ibm.com/topics/support-vector-machine#:~:text=A%20support%20vector%20machine%20(SVM,or%20hyperplane%20that%20maximizes%20the&text=SVMs%20typically%20perform%20better%20with,data%2C%20compared%20to%20logistic%20regression.

IBM. (n.d.). What is a decision tree? *IBM*. https://www.ibm.com/topics/decision-trees

InfoWorld. (2018). *PyTorch review: A deep learning framework built for speed*. Retrieved from InfoWorld.

Javatpoint. (2024). Keras Tutorial | Deep Learning with Python. Retrieved from https://www.javatpoint.com/keras

Karim, A., Shahroz, M., Mustofa, K., Belhaouari, S. B., & Joga, S. R. K. (2023). Phishing detection system through hybrid machine learning based on URL. *IEEE Access*, *11*, 36805–36822.

Kaspersky. (2021, December 9). *Powerful but short-lived: one third of phishing pages cease to be active after a day*. https://www.kaspersky.com/about/press-releases/powerful-but-short-lived-one-third-of-phishing-pages-cease-to-be-active-after-a-day.

Keras.org. (2023). Keras: The high-level API for TensorFlow. Retrieved from https://www.tensorflow.org/guide/keras

Kulkarni, A. D. (2023). Convolution Neural Networks for Phishing Detection. *International Journal of Advanced Computer Science and Applications*, *14*(4), 15–19.

MLinProduction. (2024). Deploying Models on AWS SageMaker - Part 1 Architecture. Retrieved from https://mlinproduction.com/sagemaker-architecture/

Mourtaji, Y., Bouhorma, M., Alghazzawi, D., Aldabbagh, G., & Alghamdi, A. (2021, January 1). Hybrid Rule-Based Solution for Phishing URL Detection Using Convolutional Neural Network. Wiley, 2021, 1-24. https://doi.org/10.1155/2021/8241104

Nagy, N., Aljabri, M., Shaahid, A., Ahmed, A. A., Alnasser, F., Almakramy, L., Alhadab, M., & Alfaddagh, S. (2023). Phishing URLs detection using sequential and parallel ML techniques: Comparative analysis. *Sensors, 23*(7), 3467. https://doi.org/10.3390/s23073467

Newaz, A., & Haq, F. S. (2023). A Sophisticated Framework for the Accurate Detection of Phishing Websites. arXiv preprint arXiv:2303.09735.

Nurse, J.R.C., et al. (2019). Catching the Phish: Detecting Phishing Attacks using Recurrent Neural Networks (RNNs). *arXiv*. https://arxiv.org/abs/1908.03640v1 Zhu, E., et al. (2021). Phishing Detection Using Deep Recurrent Neural Networks. *arXiv*. https://arxiv.org/pdf/2110.13424.pdf

Odeh, A., Keshta, I., & Abdelfattah, E. (2021, January 1). PhiBoost- A novel phishing detection model Using Adaptive Boosting approach. , 7(1), 64-64. https://doi.org/10.5455/jjcit.71-1600061738

Ozsahan, H., & Worthington, D. (2024b, March 12). *50+ phishing attack statistics for 2024*. JumpCloud. https://jumpcloud.com/blog/phishing-attack-statistics

Padilha, T. P. P., & Catrambone, R. (2020). *Use of the Tensorflow Framework to Support Educational Decision Making*. Proceedings of the International Conference on Education.

Padilha, T. P. P., & de Lucena, L. E. A. (2020). *A Systematic Review About Use of TensorFlow for Image Classification and Word Embedding*. Federal University of Paraiba - UFPB.

Palatty, N. P. (2024, June 5). 81 Phishing Attack Statistics 2024: The Ultimate Insight. Astra Security Blog. https://www.getastra.com/blog/security-audit/phishing-attack-statistics/

Pang, B., & Nijkamp, E. (2019). *Deep Learning With TensorFlow: A Review*. Journal of Educational and Behavioral Statistics.

Baracas, S., Hardt, M., & Narayanan, A. (2019). Fairness and Machine Learning. *Fairness, Accountability, and Transparency in Machine Learning*, 1-23.

Patidar, A. (2023, August 7). *Enhancing security: Leveraging 5 Real-Time techniques to detect phishing attacks*. https://www.loginradius.com/blog/identity/real-time-techniques-detect-phishing-attacks/

Petrosyan, A. (2024, August 19). *Worldwide digital population 2024*. Statista. Retrieved July 12, 2024, from https://www.statista.com/statistics/617136/digital-population-worldwide/

PhishGuard: A Convolutional Neural Network-Based Model for Detecting Phishing URLs. Retrieved from https://arxiv.org/html/2404.17960v1

Prolific. (2022). What is quality research? A guide to identifying the key features and achieving success. Retrieved from https://www.prolific.com/resources/what-is-quality-research-a-guide-to-identifying-the-key-features-and-achieving-success

Quang, N., Selamat, A., Krejcar, O., Herrera-Viedma, E., & Fujita, H. (2022, January 1). Deep Learning for Phishing Detection: Taxonomy, Current Challenges and Future Directions.

Institute of Electrical and Electronics Engineers, 10, 36429-36463. https://doi.org/10.1109/access.2022.3151903

Raschka, S. (2021). Book Review: Deep Learning With PyTorch - Sebastian Raschka. Retrieved from Sebastian Raschka's Blog.

Rendall, K., Nisioti, A., & Mylonas, A. (2020, August 13). Towards a Multi-Layered Phishing Detection. Multidisciplinary Digital Publishing Institute, 20(16), 4540-4540. https://doi.org/10.3390/s20164540

Shah, A. (2021). Classification and detection of email phishing using random forest supervised-unsupervised machine learning algorithms. MSc Research Project, Cybersecurity, National College of Ireland.

Simplilearn.com. (2024). What is Keras and Why is it so Popular in 2024? Retrieved from https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-keras

Slawek-Polczynska, A. (2020, November 26). *Is Agile always the best solution for software development projects?* SolDevelo. https://soldevelo.com/blog/is-agile-always-the-best-solution-for-software-development-projects/

Stryker, C. (2024, October 4). Recurrent Neural Network (RNN). *IBM*. https://www.ibm.com/topics/recurrent-neural-networks

Tamal, M. A., Islam, M. K., Bhuiyan, T., Sattar, A., & Prince, N. U. (2024). Unveiling suspicious phishing attacks: enhancing detection with an optimal feature vectorization algorithm and supervised machine learning. *Frontiers in Computer Science*, *6*, 1428013.

TechTarget. (2024). *What is PyTorch?* Retrieved from TechTarget. Zhang, Y., Wang, H., & Liu, Z. (2021). A hybrid DNN–LSTM model for detecting phishing URLs. *Artificial Intelligence Review*, 54(4), 1-20.

The Anti-Phishing Working Group. (2024). *APWG Q4 Report Finds 2023 Was Record Year for Phishing*. APWG. http://docs.apwg.org/reports/apwg_trends_report_q4_2023.pdf

The future of ransomware: Inside Cisco Talos threat hunters. (n.d.). Cisco. https://www.cisco.com/site/us/en/learn/topics/security/what-is-malware.htm

Wahyudi, D., Niswar, M., & Alimuddin, A. A. P. (2022). Website Phising Detection Application Using Support Vector Machine (SVM). *Journal of Information Technology and Its Utilization*, *5*(2), 18–24.

Wei, K., & Novak. (2023). Accurate and fast URL phishing detector: A convolutional neural network approach. A Deep Learning-Based Phishing Detection System Using CNN, LSTM, and LSTM-CNN. Electronics, 12(1), 232; https://doi.org/10.3390/electronics12010232

What is Spoofing? (n.d.). Forcepoint. https://www.forcepoint.com/cyber-edu/spoofing

Wikipedia. (2024). Amazon SageMaker. Retrieved from https://en.wikipedia.org/wiki/Amazon_SageMaker

Wong, D. (2023, October 12). The evolution of phishing attacks. *LevelBlue*. Retrieved September 10, 2024, from https://cybersecurity.att.com/blogs/security-essentials/the-evolution-of-phishing-attacks.

XenonStack. (2024). Amazon SageMaker: End-to-End Managed Machine Learning Platform. Retrieved from https://www.xenonstack.com/blog/amazon-sagemaker-machine-learning-platform

Yasar, K., Gillis, A. S., & Pratt, M. K. (n.d.). *What is unsupervised learning?* TechTarget|Enterprise AI. https://www.techtarget.com/searchenterpriseai/definition/unsupervised-learning

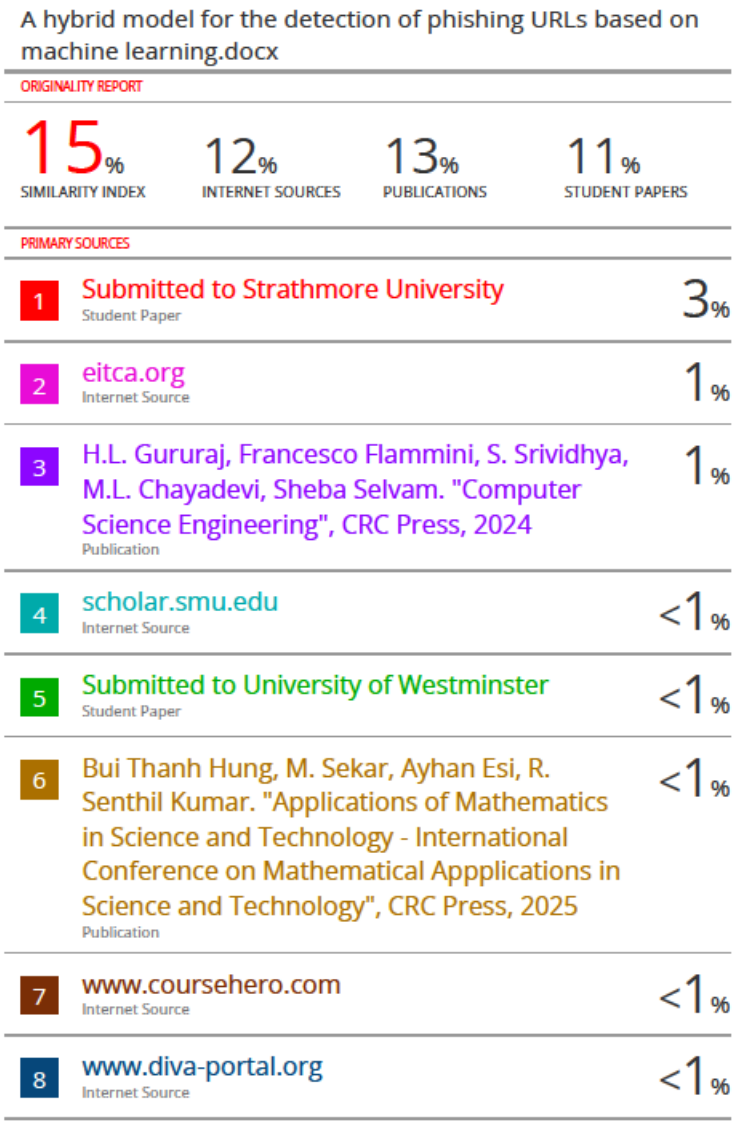
Zhang, Y., Wang, H., & Liu, Z. (2021). A hybrid DNN–LSTM model for detecting phishing URLs. *Artificial Intelligence Review*, 54(4), 1-20.

Zhang, Z., Liu, Q., & Hutchinson, S. (2020). Development of anti-phishing browser based on random forest and rule extraction framework. *Cybersecurity*, 3(1), 1-12; https://doi.org/10.1186/s42400-020-00059-1

Zuhair, H., Selamat, A., & Salleh, M. (2016, January 1). Feature selection for phishing detection: a review of research. Inderscience Publishers, 15(2), 147-147. https://doi.org/10.1504/ijista.2016.076495

Appendices

Appendix A: Similarity Report



A hybrid model for the detection of phishing URLs based on machine learning.docx

ORIGINALITY REPORT

15%	12%	13%	11%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Strathmore University Student Paper	3%
2	eitca.org Internet Source	1%
3	H.L. Gururaj, Francesco Flammini, S. Srividhya, M.L. Chayadevi, Sheba Selvam. "Computer Science Engineering", CRC Press, 2024 Publication	1%
4	scholar.smu.edu Internet Source	<1%
5	Submitted to University of Westminster Student Paper	<1%
6	Bui Thanh Hung, M. Sekar, Ayhan Esi, R. Senthil Kumar. "Applications of Mathematics in Science and Technology - International Conference on Mathematical Appplications in Science and Technology", CRC Press, 2025 Publication	<1%
7	www.coursehero.com Internet Source	<1%
8	www.diva-portal.org Internet Source	<1%

Figure 8.1 Turnitin Similarity Report First Page

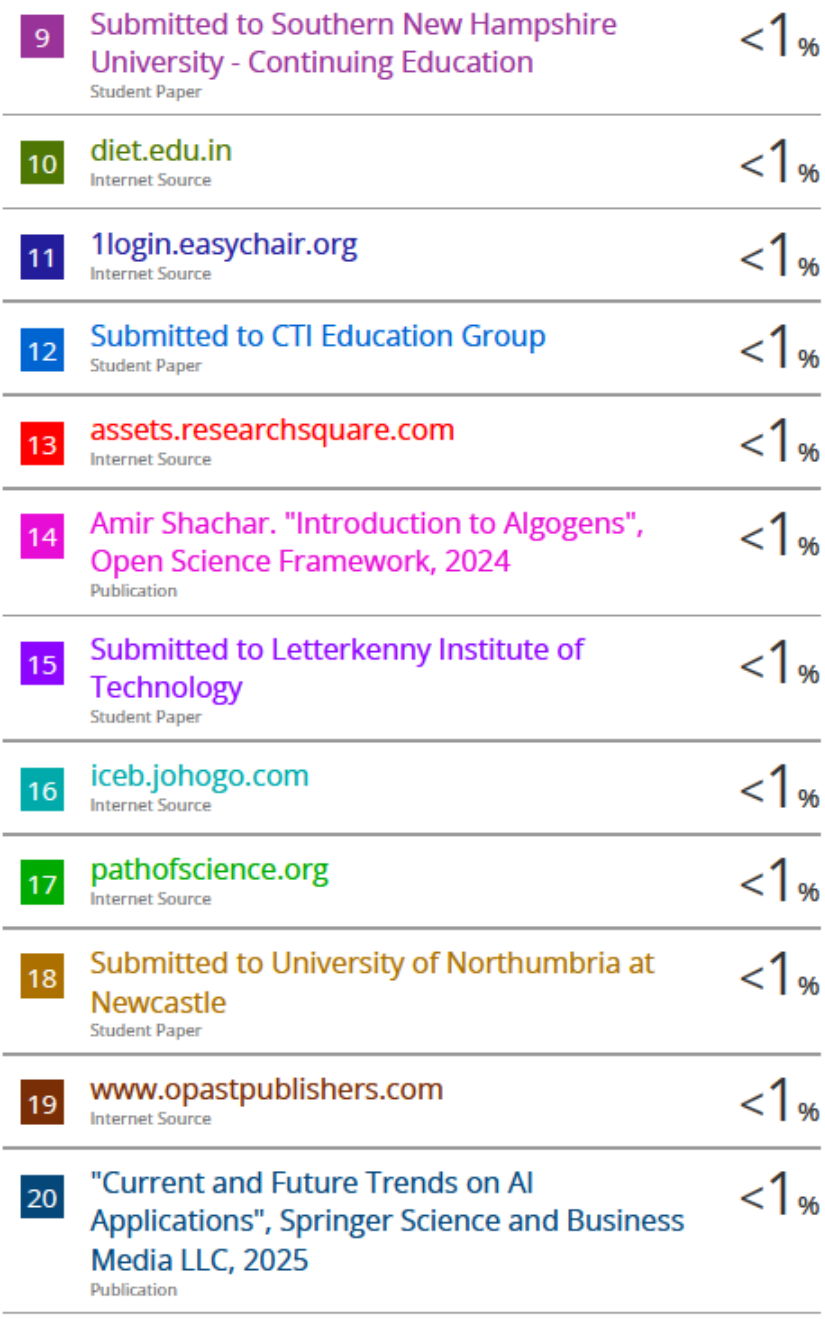


Figure 8.2: Turnitin Similarity Report Second Page

Appendix B: Ethical Clearance Confirmation

Strathmore UNIVERSITY

18th February 2025

Mr Oduor Christone,
odhiambo.oduor@strathmore.edu

Dear Mr Oduor,

RE: A Hybrid Model for the Detection of Phishing URLs based on Machine Learning

This is to inform you that SU-ISERC has reviewed and **approved** your above **SU-masters** proposal. Your application reference number is **SU-ISERC2653/25.** The approval period is from **18th February 2025 to 17th February 2026.**

This approval is subject to compliance with the following requirements:

i. Only approved documents including (informed consents, study instruments, MTA) will be used.
ii. All changes including (amendments, deviations, and violations) are submitted for review and approval by SU-ISERC.
iii. Death and life-threatening problems and serious adverse events or unexpected adverse events whether related or unrelated to the study must be reported to SU-ISERC within 72 hours of notification.
iv. Any changes anticipated or otherwise that may increase the risks or affected safety or welfare of study participants and others or affect the integrity of the research must be reported to SU-ISERC within 72 hours.
v. Clearance for the export of biological specimens must be obtained from relevant institutions.
vi. Submission of a request for renewal of approval at least 60 days prior to the expiry of the approval period. Attach a comprehensive progress report to support the renewal.
vii. Submission of an executive summary report within 90 days of completion of the study to SU-ISERC.

Before commencing your study, you will be expected to obtain a research license from National Commission for Science, Technology, and Innovation (NACOSTI) https://research-portal.nacosti.go.ke/ and obtain other clearances needed.

Yours sincerely,

Mr Ambrose Rachier,
Chairperson; SU-ISERC

Ole Sangale Rd, Madaraka Estate. PO Box 59857-00200, Nairobi, Kenya. Tel +254 (0)703 034000
Email admissions@strathmore.edu www.strathmore.edu

Figure 8.0.3: Letter of Ethical Approval