UMYU Scientifica, Vol. 5 NO. 1, March 2026, Pp 179 – 199
https://doi.org/10.56919/usci.2651.016
ISSN: 2955 – 1145 (print); 2955 – 1153 (online)
A periodical of the Faculty of Natural and Applied Sciences, UMYU, Katsina

ORIGINAL RESEARCH ARTICLE

An Ensemble Machine Learning Scheme for Real-time Phishing URL Detection and Browser-Level Deployment

Daniel Dauda Wisdom[1], Alabi Orobosade Adewunmi[2], Joshua Bature Hassan[3], Esther Odunayo Oduntan[4], Taiwo David Ajayi[1] and Olusegun Adeosun[5]

1Department of Cybersecurity, Data Science, College of Computing Sciences, Federal University of Agriculture, Abeokuta, Ogun State, Nigeria
2Department of Computer Science, Federal University of Agriculture, Abeokuta, Ogun State, Nigeria
3Department of Computer Sciences, Federal University, Oye Ekiti, Ekiti State, Nigeria
4Department of Computer Science, Federal University of Technology Ilaro, Ogun State, Nigeria
5Department of Computer Science, Federal University of Agriculture Abeokuta (FUNAAB), Abeokuta, Ogun State, Nigeria

ABSTRACT

Cybercriminals continue exploiting means to perpetrate crimes, using fake links and malicious websites to send invites, display compelling bids, and attempt to gain access to organizations' transaction data to steal or disrupt organizations' operations, resulting in heavy losses. As the digital age advances, cyber threats also advance, with illegitimate web links sent to millions of people, reaching a level of sophistication that results in numerous victims. Thus, a phishing detection scheme was developed; it evaluates Universal Resource Locators (URLs) and classifies them as legitimate (Good) or malicious (Bad) sites. The scheme used an algorithmic process state powered by machine learning (ML) models, demonstrating its effectiveness in accurately recognizing and categorizing URLs. The developed model incorporates essential features to manage vital datasets, used to identify patterns, and provide accurate predictions. Stress testing, load testing, and reaction time analysis were conducted to assess the scheme's scalability and reliability. These tests were essential for determining how well the system performs under various demand scenarios, ensuring that the developed scheme remains responsive and reliable even during peak demand. The scheme handled increased traffic without a decrease in performance. The performance verified that it could manage a range of user input levels while maintaining efficiency. Kaggle datasets were used and implemented in Python for training (30%) and validation (70%). In addition, a password generator and checker were developed to educate organizations about the importance of password combinations and how they are critical to enhancing security. The model informed individuals/organizations on how to efficiently secure data with a superior level of security to protect organizations' business transactions.

ARTICLE HISTORY

Received December 03, 2025
Accepted March 07, 2026
Published March 25, 2026

KEYWORDS

Phishing-Detection, Password-Generator, Password-Checker, Machine-Learning, URLs-Analysis, Software Mechanism Development and Testing

INTRODUCTION

Phishing attacks are among the most common cyber-threats on the internet today (Abdillah, 2022), due to the nature of the sensitive data involved, such as organizations transaction data, bank account numbers with a potential access to larger computerized systems, using fraudulent email or website requests (Das *et al.,* 2020), which indicates that the attackers often perform actions similar to an entity, to steal information from members or users. Phishing, according to Alhassan et al. (2020), is a cybercrime tactic in which fraudulent communications, typically disguised as legitimate messages, are used to deceive individuals into disclosing sensitive information such as passwords, credit card numbers, or personal data. These attacks often focus on requests to change identity, passwords, and other important information, using email, social media, and other tactics (Ahmad *et al.,* 2020; Wisdom *et al.,* 2024). In the industrial sector, the Anti-Phishing Working Group stated that the main targets of these attacks were presently webmail, financial institutions, payments, social media, and e-commerce (Ahmad *et al.,* 2021; Hassan *et al.,* 2025).

Phishing attacks also involve the utilization of the world's top internet services, such as Namecheap (24%), Google (16%), Public Domain Registry (PDR) (19%), NameSilo (6%), Tucows (7%), and other channels (28%) (AlEroud & Karabatis 2020; Nitesh, *et al.,* 2022). Machine learning (ML) plays a crucial role in the ongoing battle against phishing attacks. Its ability to analyze vast amounts of data efficiently allows it to identify subtle patterns and anomalies indicative of phishing attempts that might evade traditional rule-based methods (Gupta & Islam,

Correspondence: Daniel Dauda Wisdom. Department of Cybersecurity, Data Science, College of Computing Sciences (COLCOMPS), Federal University of Agriculture, Abeokuta, Ogun State, Nigeria. ✉ danieldw@funaab.edu.ng.

How to cite: Wisdom, D. D., Adewunmi, A. O., Hassan, J. B., Oduntan, E. O., Ajayi, T. D., & Adeosun, O. (2026). An Ensemble Machine Learning Scheme for Real-time Phishing URL Detection and Browser-Level Deployment. *UMYU Scientifica*, 5(1), 179 – 199. https://doi.org/10.56919/usci.2651.016

2020; Alhassan *et al.,* 2020). Using algorithms to examine email content, sender information, and URLs for patterns suggestive of phishing attacks, ML plays a critical role in phishing detection (Aljofey et al., 2020; Wisdom & Vincent, 2024b). Natural language processing (NLP) recognizes linguistic indicators in email text, identifying warning signs such as dubious URLs, luring strategies, and attempts at impersonation, while supervised learning distinguishes between authentic and phishing cases (Canfield *et al.,* 2021).

Malicious activities carried out through human interactions can psychologically influence a person to divulge confidential information or to break security procedures (Wisdom *et al.,* 2024b). Due to these human interactions, social engineering attacks are the most powerful, as they threaten all systems and networks. Phishers use trusted sources, such as bank help desks, to deceive victims into disclosing their sensitive information (Hassan & Abdelfettah, 2017; Farooq et al., 2024; Kumar, 2020).

Phishing attacks have reached unprecedented levels, especially as emerging technologies such as mobile and social media proliferate. For instance, from 2017 to 2020, phishing attacks increased by 14% among businesses in the United Kingdom, with a large proportion originating from social media (Zainab *et al.,* 2021).

Phishers usually hunt their potential victims by using text and information related to the COVID-19 pandemic. The data revealed a significant rise in phishing attacks and their consequent impact, specifically during the COVID-19 pandemic. In recent years, phishers have focused on Software as a Service (SaaS) and webmail, which accounted for 33% of attacks across a range of industry sectors (Wisdom *et al.,* 2025). IBM found that 27% of phishing attacks in 2018 targeted webmail services. It was also noted that 29 percent of the attacks against businesses analyzed by X-Force identified the breach source as a phishing email (Kumar et al., 2020; Wisdom *et al.,* 2024b).

Symantec found that in the underground economy "custom phishing page services" are being sold for between 3 US Dollars and 12, indicating that the overhead for setting up a custom phishing attack is minimal, and with the advent of generative AI capable of generating code, the ability to clone legitimate websites have become much easier to achieve by anybody with access to the internet. More so, gift cards are now among the most common ways for a scammer to cash out their earnings. The FBI estimated the 2018 victim losses from phishing at USD 48,241,748, with 26,379 people affected by this type of scam (Kumar 2020; Abdillah 2022; Zainab *et al.,* 2021; Wisdom *et al.,* 2025).

Despite significant progress in machine learning (ML) based phishing detection, existing URL-based detection systems still exhibit several real-world and procedural limitations. First, many studies rely on outdated or static datasets, which limit the ability of trained models to detect emerging phishing patterns and zero-day attacks (Hong *et al.,* 2022; Makkar *et al.,* 2021; Wisdom *et al.,* 2025c). The availability of large, up-to-date, labelled datasets remains a major challenge, hindering the robustness and generalization of phishing detection models (Gupta & Islam, 2020). Second, most prior work primarily evaluates models using offline accuracy metrics on limited benchmark datasets, often without testing scalability or robustness under real traffic conditions. As a result, systems that report very high accuracy (often 96-99%) still struggle when deployed in real-world environments where latency, high request volume, and evolving attack patterns affect performance (Hung et al., 2017; Hassan *et al.,* 2025). Third, many studies provide little evidence of real-world deployment, focusing mainly on algorithmic performance rather than practical implementation, system integration, or operational load testing. For instance, several recent works developed deep learning models such as convolutional neural networks (CNNs), long short-term memory (LSTMs), or ensemble approaches that achieve high classification accuracy, yet they are typically validated only through experimental datasets rather than production-level systems (Kumar *et al.,* 2020; Farooq & Jabbar, 2024).

To address these gaps, this study presents an Ensemble ML Scheme for Real-time Phishing URL Detection and Browser-Level Deployment, designed with a unique ML prediction pipeline. The scheme first introduced a unique end-to-end pipeline that included dataset preprocessing, feature extraction, model training, and evaluation using modern ML libraries. Second, the study evaluates the system not only in terms of traditional classification metrics but also through performance testing under simulated load, examining response time, throughput, and stability under high request volumes. Third, the research demonstrates a practical deployment scenario, showing how the detection model can be integrated into a real-time application environment, such as an Application Programming Interface (API-based service or web interface), thereby bridging the gap between experimental research and operational cybersecurity tools. Lastly, the study developed a password generator and checker, educating organizations and individuals on the vital role of password combinations in improving security while mitigating cyber threats.

The key objectives of the study were to investigate: How effectively can an ML-based Python system detect phishing URLs using lexical and structural URL features? How does the developed software detection scheme perform under high request loads or stress conditions, in terms of response time, accuracy, and system stability? Will the developed pipeline be reproducibly deployed in a real-world environment, providing reliable phishing detection in practical terms? And the study proved that ML Scheme training on phishing URL datasets can achieve high detection accuracy with low false-positive rates. The scheme maintains stable performance and acceptable latency under increased request loads. A reproducible pipeline with a deployment-ready architecture for real-time phishing detection was developed beyond purely experimental evaluation. Addressing dataset quality, scalability testing, and

deployment considerations simultaneously, this paper aims to improve on existing phishing detection studies from normal algorithmic evaluations toward an operationally deployable cybersecurity scheme.

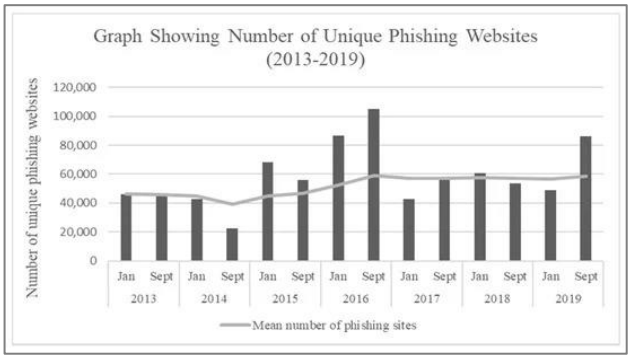

Figure 1: Number of unique phishing websites between 2013 and 2019.

REVIEW OF RELATED LITERATURE

Various related studies also described the techniques used to detect phishing attacks, although had several undisclosed issues, such as, the methods by which the dataset was distributed against the phishing attacks, the use of popular techniques for phishing types, the use of phishing type evaluations, and the use of parameters for phishing classification techniques (Rao & Pais 2019; Nitesh *et al.,* 2022). These studies contribute valuable insights into phishing detection and prevention techniques, but there is a need for more empirical evidence and clearer methodologies to validate and improve the effectiveness of these approaches.

A crucial component of cybersecurity is phishing detection, which aims to spot and stop fraudulent attempts to trick users into divulging valuable information. To tackle this difficulty, a plethora of systems and methodologies have been developed, using diverse techniques such as behavior analysis, heuristics, and ML (Shirazi 2018; Siti et al. 2020).

2.1 Phishing

Phishing has evolved significantly since its inception in the late 1990s, adapting to technological advancements and exploiting vulnerabilities in communication channels Initially characterized by rudimentary email scams, phishing has diversified into sophisticated schemes, including spear phishing, whaling, and pharming, targeting specific individuals or organizations (Yanz 2019 and Wang, *et al.,* 2020). Phishing perpetrators employ various techniques to deceive their targets, often utilizing social engineering tactics to exploit human psychology and manipulate emotions. Common tactics include masquerading as reputable entities, creating urgency or fear to prompt immediate action, and using persuasive language to elicit trust and compliance (Hong & Lee, 2020).

The impacts of phishing extend beyond financial losses to include reputational damage, identity theft, and compromised cybersecurity infrastructure. Individuals and organizations alike suffer the consequences of phishing attacks, experiencing diminished trust in online communications and increased susceptibility to future cyber threats (Yu et al., 2020; Wisdom *et al.,* 2025b). Effective countermeasures against phishing involve a combination of technical solutions, user education, and organizational policies (Ahmad et al. 2021). Technologies, such as email filtering, website authentication, and two-factor authentication, mitigate the risk of phishing attacks. Additionally, raising awareness among users through cybersecurity training programs and implementing stringent verification procedures can enhance resilience against phishing attempts (Hong & Lee, 2020; Wisdom *et al.,* 2023)

2.2 Machine Learning

Machine learning (ML) has emerged as a transformative technology across various domains, revolutionizing how we analyze data, make predictions, and automate tasks (Wisdom *et al.,* 2025). Supervised learning involves learning a mapping between input data and desired outputs using labelled examples. Popular algorithms include linear regression, decision trees, and support vector machines (SVM). Unsupervised learning identifies patterns and structures within unlabeled data. Clustering, dimensionality reduction, and anomaly detection are common tasks. Reinforcement learning, this method learns through trial and error, interacting with an environment and receiving rewards for desired actions, has made significant breakthroughs in game playing and robotics (Makkar *et al.,* 2021). ML algorithms could classify and analyze images with human-level accuracy, impacting areas like medical diagnosis, autonomous vehicles, and content moderation 2. Natural language processing (NLP): ML plays a crucial role in tasks such as machine translation, sentiment analysis, and chatbots 3. Recommender systems, ML algorithms personalize user experiences by suggesting relevant products, content, or services (Nitesh *et al.,* 2022).

2.3 ML Strategy on Phishing Attacks

i. Improved Detection Accuracy: Well-trained ML models can achieve high accuracy in detecting phishing attempts, reducing the number of legitimate emails flagged incorrectly. ML models can generalize their learningto identify phishing attempts with different characteristics, tthereby hereby enhancing their effectiveness diversediverse contexts (Nitesh *et al.,* 2022).

ii. Real-Time Response: ML-based systems can automate the process of analyzing emails for phishing indicators, enabling real-time detection and filtering of malicious attempts. This reduces the burden on human analysts and facilitates a proactive approach to combating phishing threats (Nitesh et al., 2022). iii. Adaptability: ML models adapt to evolving phishing tactics. By continuously learning from new data, including successful and unsuccessful phishing attempts, ML models can improve their accuracy in detecting novel and previously unseen phishing attacks (Yu *et al.,* 2020). iv. Reduced False Positives: ML algorithms excel at identifying subtle patterns and relationships within data that might be

difficult for humans to detect. This allows them to extract valuable features from emails, such as language patterns, sender information, and URL characteristics, which can be used to effectively distinguish phishing attempts from legitimate emails (Ahmad *et al.,* 2021). v. Scalability and Efficiency: ML algorithms can analyze vast amounts of data efficiently, identifying patterns and anomalies indicative of phishing attempts that traditional rule-based methods might miss. This scalability is crucial as phishing attacks become increasingly common and sophisticated. The advantages of ML include increased detection accuracy, real-time response, adaptability to changing threats, reduced false positives, scalability, efficiency, and an improved user experience. Organizations may strengthen their cybersecurity defenses and guard against the growing threat of phishing attacks by leveraging ML (Nitesh et al., 2022)

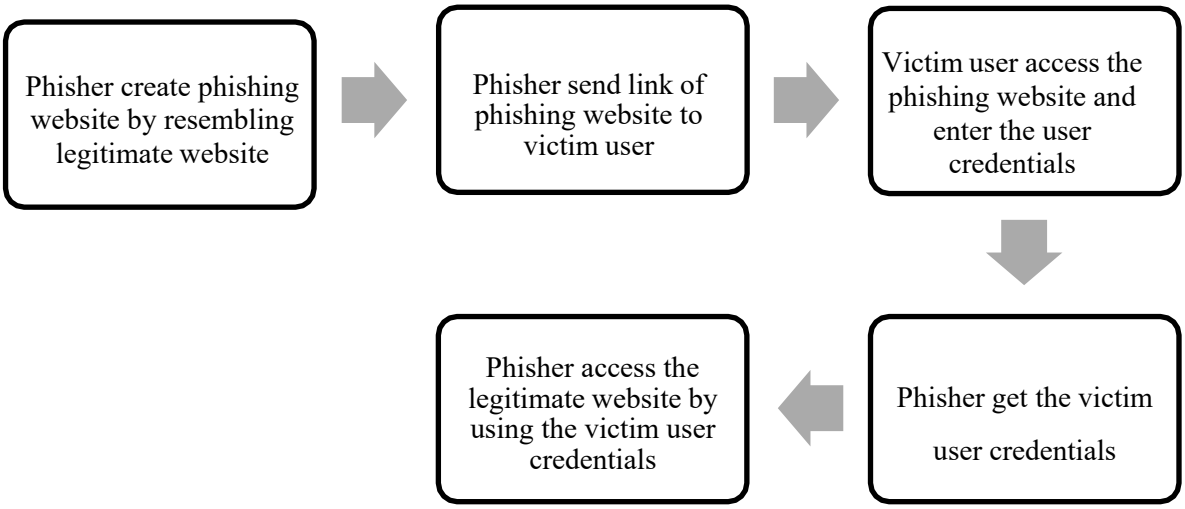

Figure 2: Life cycle of phishing attacks

2.4 The anatomy of phishing attacks

Most phishing attacks start with a malicious email message and fake links. These emails may be directed toward a particular organization, group, or individual and may be distributed at random to prospective users (Raj *et al.,* 2022). The attack could also be launched via a variety of other vectors,, namel, physical letters, phone conversations, and instant messaging. Nonethelesss, it's critical to understand these phases to develop an anti-phishing strategy. The phishing attack process is divided into five phases: planning, setup, attack, collection, and cash. These phases include preparing for the attack, sending a malicious program using the selected vector, observing the user's reaction to the attack, tricking the user into disclosing their confidential information, which will be transmitted to the phisher, and obtaining the targeted funds. The early phase includes initializing the attack, creating the phishing email, and sending it to the victim. The Second phase includes receiving an email from the victim disclosing their information; this is the case for the respondent, and it is the final phase in which the defrauding is successful. All phishing schemes, however, consist of three main stages: first, the target is asked for sensitive information; second, the target provides the information to the phisher; and lastly, the phisher uses the information for malevolent objectives. These phases can be classified into sub-processes based on phishing trends. Thus, a new anatomy for phishing attacks has been proposed in this paper, which expands and integrates previous strategies to strategically cover the full life cycle of a phishing attack. The proposed developed strategy consists of 4 phases as depicted in Figure 12.

A phishing attack constitutes a mix of technical and social engineering tactics (Siti et al., 2020). The three components in the phishing attack include 1. Medium, 2. Vector and 3. Technical approaches. For the medium, there are three: internet, voice, and short messaging service (SMS). The internet is a commonly used medium for phishing because it offers a huge opportunity for phishers to deploy phishing attacks. The vector is a vital position to launch the phishing attack. The vectors for the internet are email, websites, and social networks. The technical approaches to phishing attacks can be divided into two categories: social engineering and malware-based phishing. Social engineering exploits the user's fear of losing something valuable, leading the user to reveal personal information to the phisher. In a malware-based phishing attack, malicious programs are secretly installed to give the phisher access to the user's computer (Nitesh et al., 2022; Wisdom *et al.,* 2024c).

2.5 Strategies in Mitigating Phishing Attacks

Safeguarding against phishing calls requires a multifaceted approach. Organizations should train staff to recognize and report suspicious messages, in addition to using technologies such as multi-factor authentication and email filters. It's also essential to use web filtering tools, regularly update software, and establish specific security standards. Defenses are further strengthened by cooperation and keeping up with current phishing techniques, which eventually mitigate the danger and magnitude of these attacks (Hung *et al.,* 2017). Figure 4 depicts Anti-Phishing Solutions

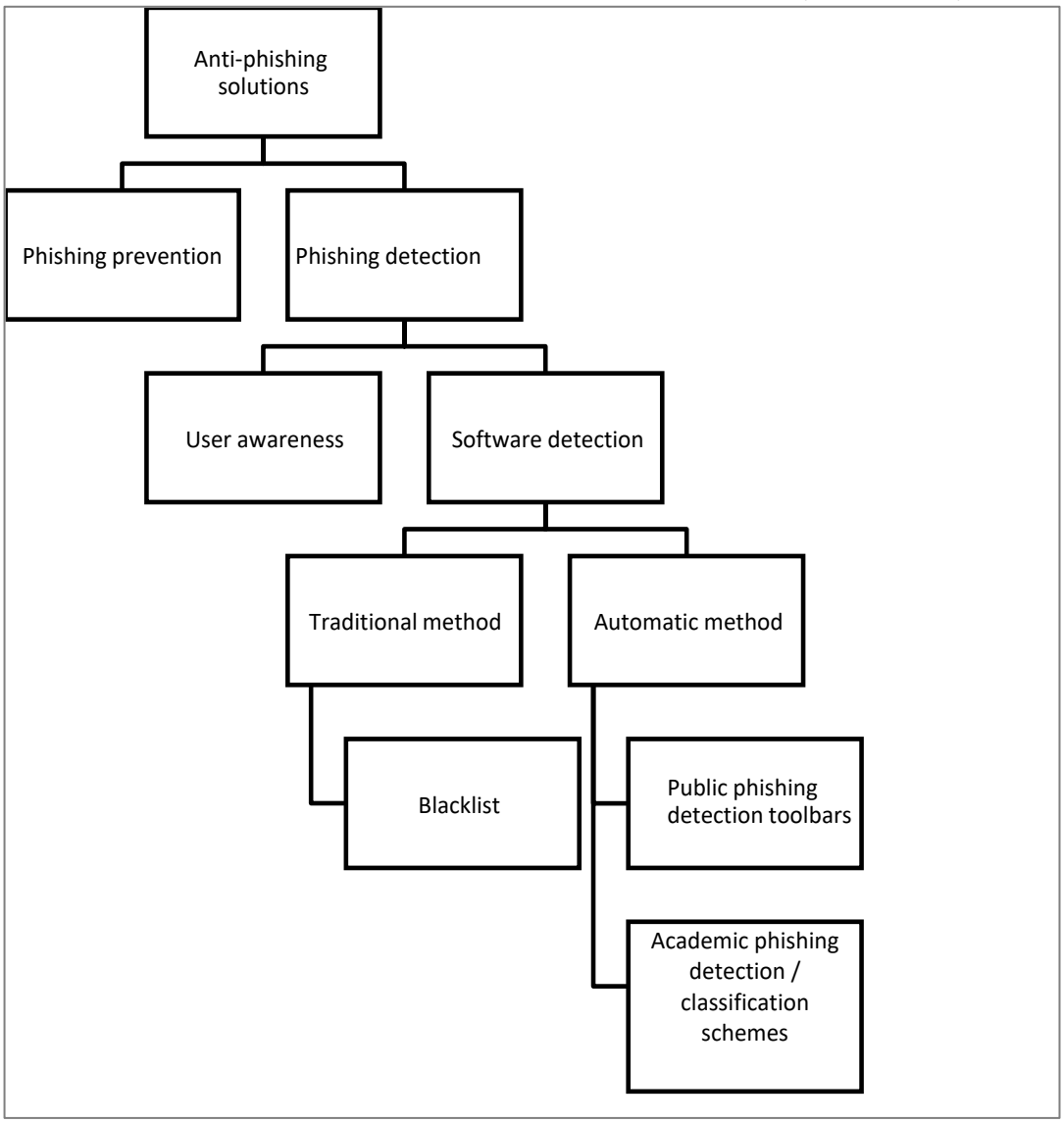

Figure 3: Proposed approach for mitigating attacks.

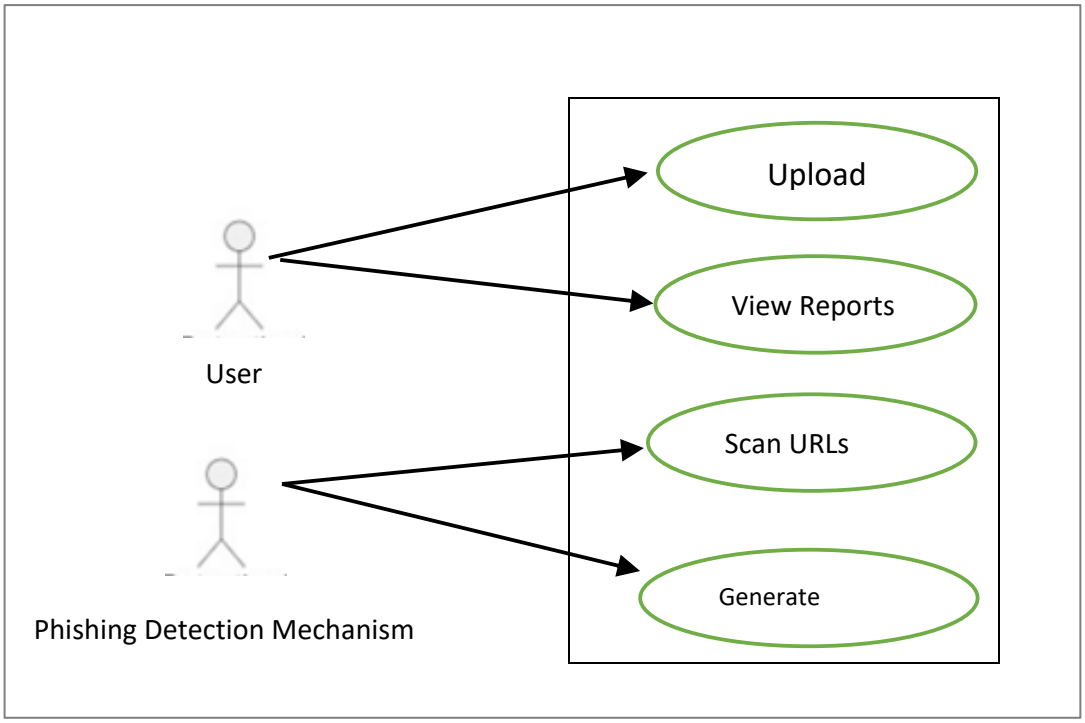

Figure 4: Developed Use-Case Diagram

Table 1. Summary of Related Literature (Contributions and Limitations) on Phishing Detection

Author/ year	Contributions to knowledge	Limitations / Weaknesses
Abdillah *et al.*, 2022	The study serves as a valuable resource for understanding the current state of phishing classification techniques and provides valuable insights for future research directions	Its impact on improving the field's methodology and practical applications may be limited by its focus on evaluating existing research and absence of new algorithm suggestions and quantitative results.
Siti *et al.*, 2020	The study provides a descriptive overview and classification of phishing attacks and anti-phishing solutions without specifying a particular research methodology	It would benefit from a deeper analysis of the effectiveness of various anti-phishing techniques
Zainab *et al.*, 2021	Makes valuable contributions to understanding phishing attacks and proposed preventative measures	The study would benefit from a clearer methodology and more empirical evidence to support its claims.
Hung *et al.*, 2017	Proposed a deep learning-based URL detector. Authors argued that the method can produce insights from URL.	Deep learning methods demand more time to produce an output. In addition, it processes the URL and matches with the library to generate an output.
Hong *et al.*, 2022	Developed a crawler to extract URLs from data repositories. Applied lexical features approach to identifying the phishing websites.	The performance evaluation was based on crawler-based dataset. Thus, there is no assurance for the effectiveness of the URL detector with real time URLs.
Kumar *et al.*, 2020	Proposed a URL detector based on blacklisted dataset. A lexical feature approach was employed to classify malicious and legitimate websites.	Authors employed an older dataset which can reduce the performance of the detector with real—time URLs.
Hassan & Abdelfettah 2017	Suggested a URL detector for classifying websites and predicting the phishing websites. They used Generative adversarial technique to improve the performance.	The performance of GA based URL detector was better; nonetheless, the predicting time was huge with complex set of URLs.
Rao & Pais 2019	Authors employed page attributes include logo, favicon, scripts and styles.	The method employed a server for updating the page attributes that reduces performance of the detecting system.
Aljofey *et al.*, 2020	A CNN based detecting system for identifying the phishing page. A sequential pattern was used to find URLs.	The existing research shows that the performance of CNN is better for retrieving images rather than text.
AlEroud & Karabatis 2020	Generative adversarial (GA) network is used in research to bypass a detection system.	Neural Network based detection systems can identify the impression of an adverse network by learning about the environment.

Table 2. Showing Hardware Requirements of the Proposed Phishing Detection Mechanism

Network Connectivity	Wi-Fi and mobile data	Wi-Fi and 4G LTE
	Minimum Requirements	Recommended Requirements
Processor Speed	at least 2.0 GHz	2.5 GHz or higher
Memory Capacity	a minimum of 8 GB RAM	16 GB RAM or higher
Storage Space	at least 256 GB of storage	512 GB of storage or higher

Table 3. Class, Precision, Recall, F1 Score and Support

Class	Precision	Recall	F1 Score	Support
Normal	0.8235	0.8936	0.8571	141
Attack	0.8000	0.6897	0.7407	87
Accuracy	0.8158	-	-	228
Macro Avg	0.8118	0.7916	0.7989	228
Weighted Avg	0.8146	0.8158	0.8127	228

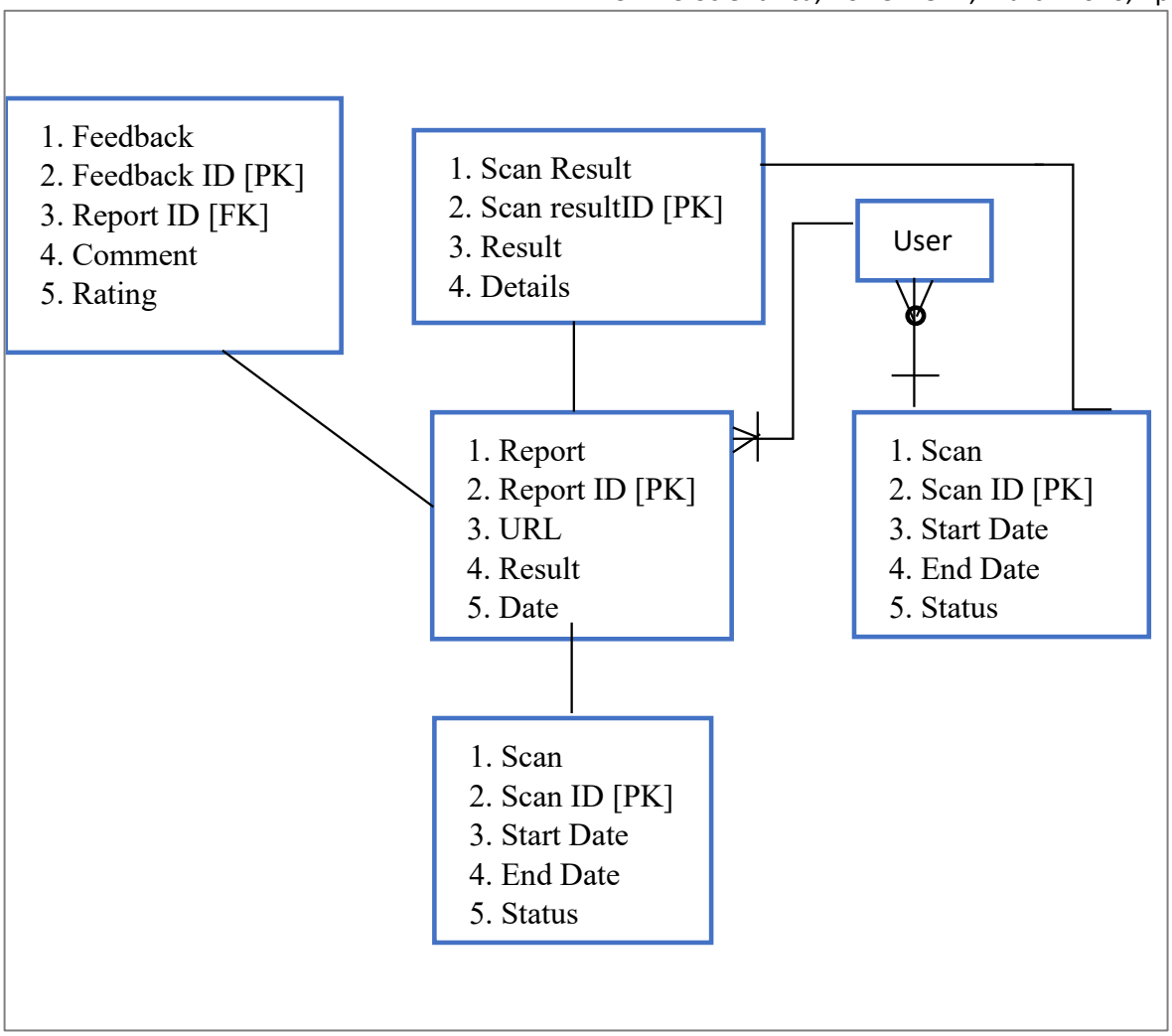

Figure 5: Entity Relationship Diagram

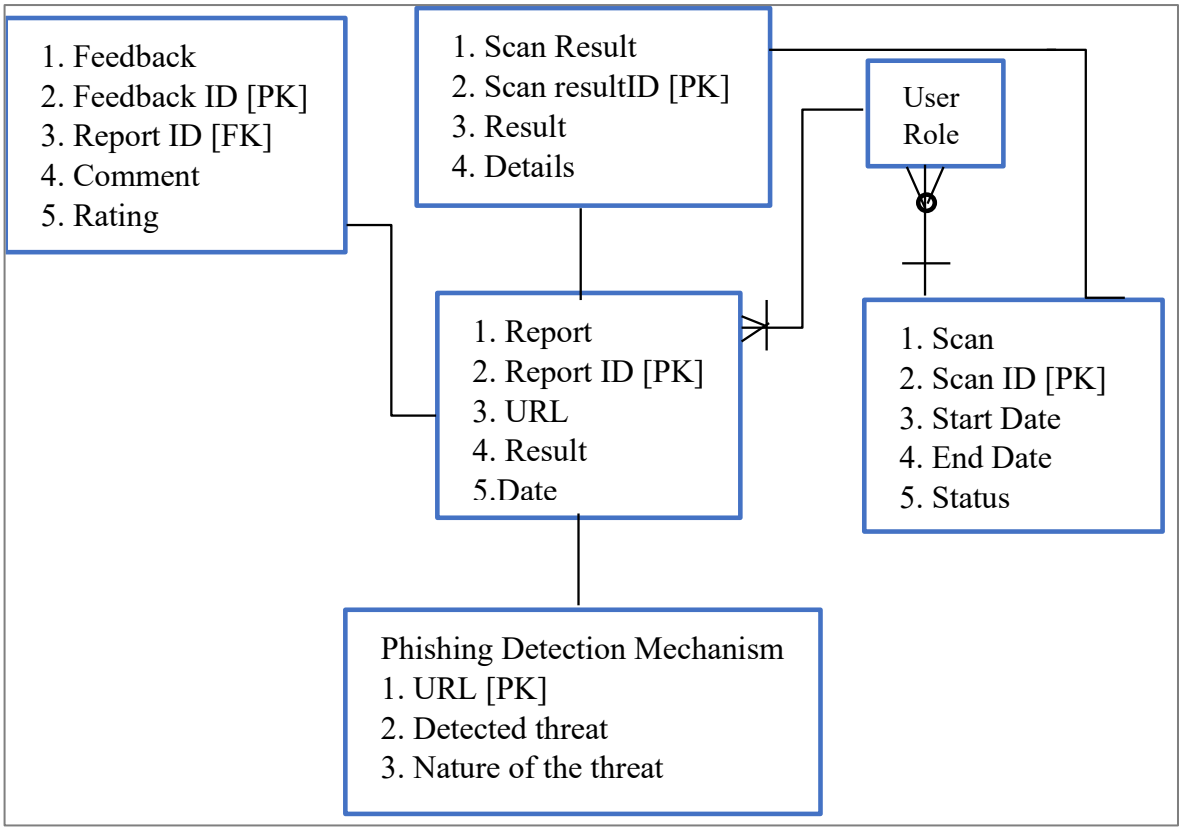

Figure 6: Class Diagram of the Proposed Mechanism

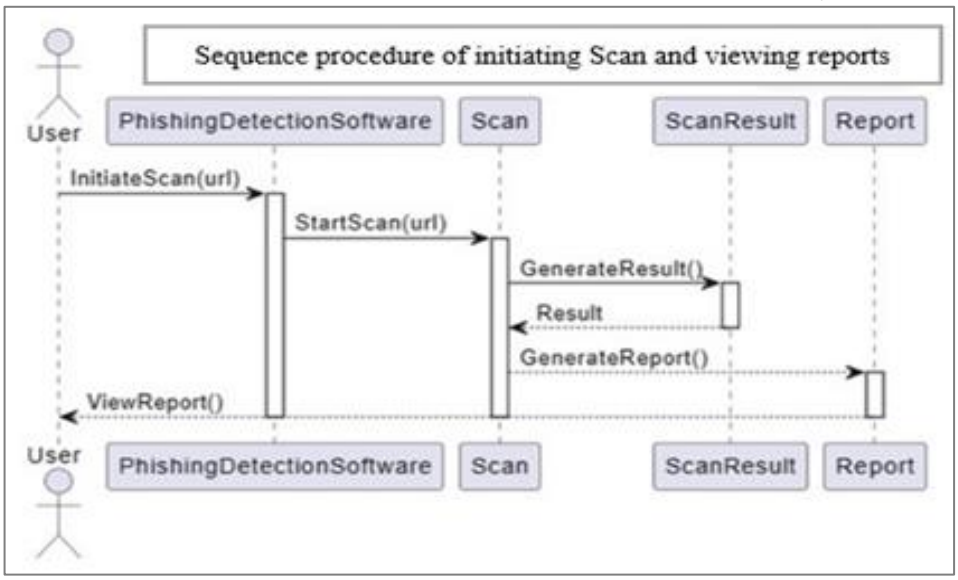

Figure 7: Developed Sequence Diagram

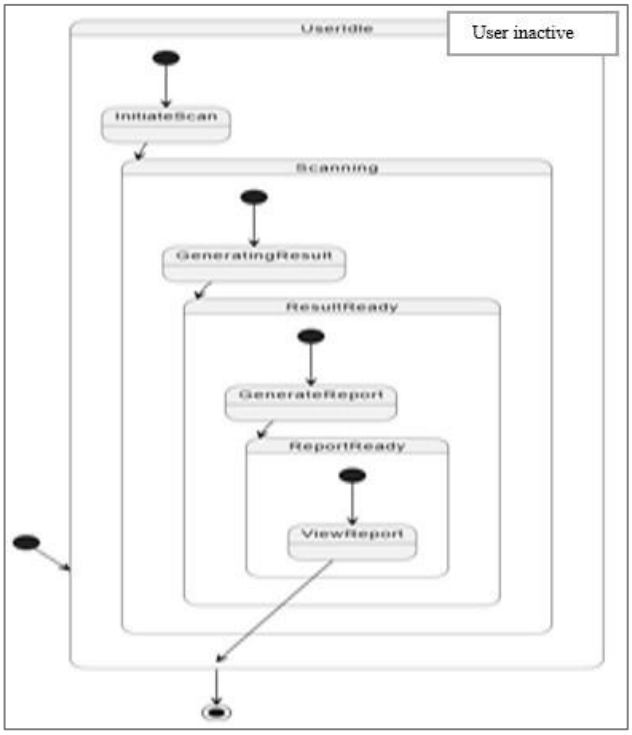

Figure 8: State Diagram of the Proposed Phishing Detection Scheme

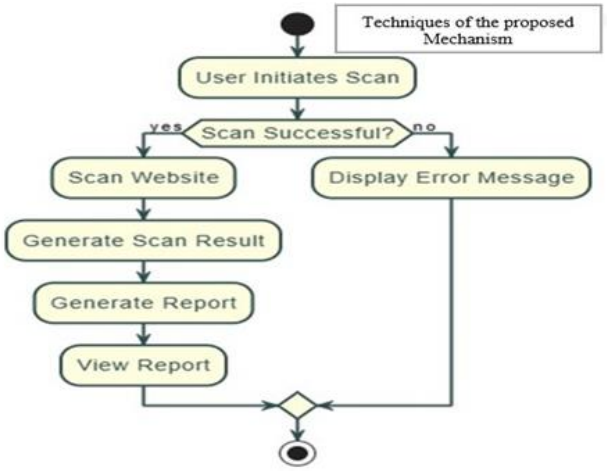

Figure 9: Proposed Flowchart Diagram

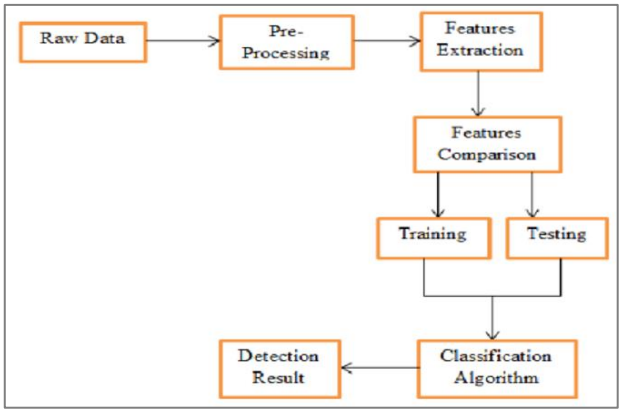

Figure 10: Data Pre-processing stages

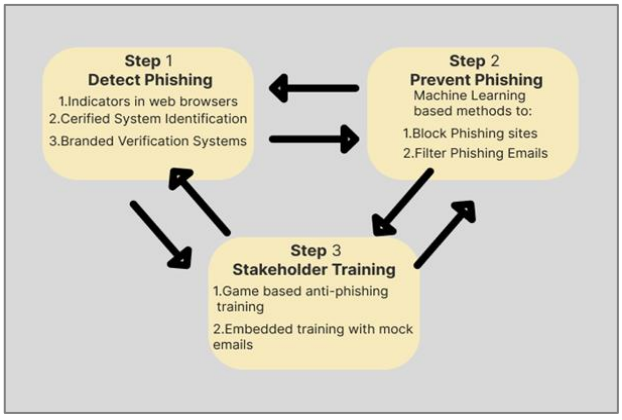

Figure 11: Proposed Phishing detection strategy

Algorithm 1: The three lifecycles in Phishing

1. Medium
   (i) The internet
   (ii) The persuading audio voice messages
   (iii) Short enticing text messages (SMS)
2. Vector
   (i) Excellent positions to lunch the attack
   (ii) Links
   (iii) Emails
   (iv) A malicious clone of a well-known website

(v) Social networks (Facebook, Twitter, Instagram, LinkedIn etc.)

3. Technical Approach

(i) social engineering

a. Research on Target individual or organizations
b. Crafting a Message for the organizations
c. Setting a Trap such as fake login page links, Malware attachments to infect systems ones opened, Requests for sensitive information such as bank details, credentials.
d. Distribution of the Attacks, by sending the phishing content through Emails, SMS known as smishing, Voice calls as vishing and social media platforms.
e. Victims Interaction, Victims are tricked into: Clicking a link, downloading a file and providing personal information
f. Data Harvesting or Infection, if successful: Credentials are collected, Malware installed to spy, steal, or control systems.
g. Exploitation, stolen data is used to: Access sensitive systems, Steal money. Launch other attacks as Business Email Compromise.
h. Covering Tracks, Attackers may delete traces, Use proxies, VPNs, or botnets to mask their activities etc.

(ii) malware-based attacks

a. Ransomware Attacks encrypting files and demand payment to unlock them. For example, the WannaCry ransomware.
b. Spyware Attacks: secret, remote monitoring of user activity on the infected system to steal information such as account login details.
c. Trojan Horse Attacks, disguising malicious software's that opens a back door for attackers as legitimate software
d. Worm Attacks, Self-replicating programs that spread across networks without needing to attach to other files.
e. Rootkit Attacks hides malware presence and give attackers administrator-level control.

Summarized malware attacks strategies

- Phishing emails with malicious attachments or links.
- Drive-by downloads when visiting compromised websites.
- Malicious apps from unofficial app stores.
- Infected USB drives or software downloads.

METHODOLOGY

The methodology includes Python 3.12, essential libraries such as NumPy and Pandas, and Jupyter notebooks. The planning phase focuses on defining the research purpose, objectives, and scope. It involves understanding the research requirements and conducting a feasibility study to determine if the study is viable. The initiation phase was an important step in the study as it ensured that the research was well-defined and that there is a clear understanding of the goals, objectives, and scope. The initiation phase also helps identify potential risks and challenges, enabling the development of mitigation plans. A detailed analysis was conducted, as the phishing detection software mechanism will need to have access to the CPU to provide sufficient processing power for training ML models and performing real-time phishing detection. Also, the amount of CPU resources required was determined by the dataset size and the complexity of the developed scheme. A machine with a multi-core processor and 8GB of RAM was used, with memory access to store datasets, ML models, and other relevant data. The amount of memory used was one (1) terabytes of storage. With access to a stable internet connection in order to download datasets, update the system, monitor and track the progress, as well as assessing performance, managing Changes, and ensuring that the research remains on track to achieve its objectives. Measured performance using predetermined metrics and Key Performance Indicators (KPIs). Conducting periodic performance analysis to evaluate the inclusive performance of the mechanism. Assess variances, analyze trends, and identify areas for improvement. Used the analysis to make informed decisions, revise the research plans where needed, and implement corrective actions. Monitoring and control throughout the study lifecycle.

The algorithms used include decision trees, random forests, support vector machines, and neural networks. The developed software scheme included a user interface that allows users to interact with the system, view phishing-detection results, and manage settings and configurations.

The scheme was developed to detect phishing websites quickly and accurately, with minimal latency trade-off and high throughput to handle a large number of requests. The model was developed to allow adding more resources or nodes to the system without affecting performance. The mechanism was developed with reliability in mind, ensuring it is available at all times and can recover from failures and challenges without losing data or affecting the user experience. Easy to maintain and update, well-documented, allowing developers to make changes without affecting the rest of the system.

VS Code was used as the primary code editor. Jupyter Notebook, Used for exploratory data analysis (EDA) and model development. The Python Core language was used to implement the phishing detection logic, data processing, and the ML model. Python Libraries and models, such as Pandas and NumPy, were used for Data manipulation and analysis. Matplotlib and Seaborn for data visualization. The developed software mechanism was easy to use and understand, with user-friendly interface and provide clear feedback to users about detected phishing attacks.

The software was developed to be able to analyze URLs and determine if they are potentially phishing websites. The scheme used ML models to improve phishing

detection accuracy over time. Providing a user-friendly interface for users to interact with and receive notifications about potential phishing threats.

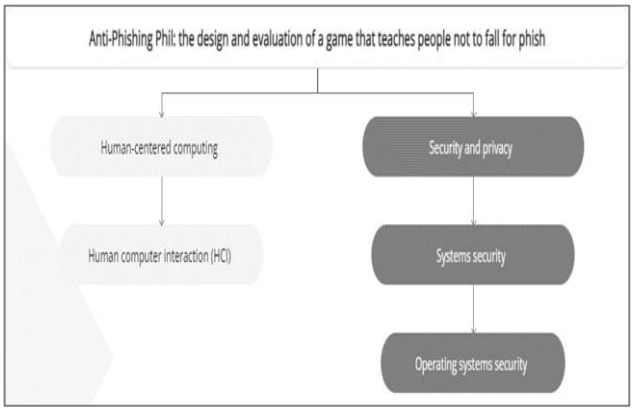

Figure 12: Algorithm for Anti-Phishing Phil

This study used NumPy, Scikit-learn, Matplotlib, and Seaborn, and model serialization was performed using Pickle, while the user interface was implemented with HTML and CSS during development. The dataset consisted of phishing URLs collected from Kaggle and PhishTank, and legitimate URLs obtained from Alexa Internet top sites, which were cleaned through deduplication and validation to create a balanced dataset. URL-based lexical, host, and security features (e.g., URL length, number of dots and hyphens, presence of IP address, HTTPS usage, domain age, and SSL indicators) were extracted using Python scripts. Data pre-processing involved tokenization of URL strings, numerical encoding of categorical features, handling missing values, and feature scaling prior to model training. The dataset was split into training (30%), validation (70%), and testing sets with a fixed random seed to ensure reproducibility, and multiple ML algorithms including logistic regression, decision tree, random forest, and support vector machine were trained using hyper parameter tuning via grid search. Model performance was evaluated using accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrix visualization, while inference latency was also measured.

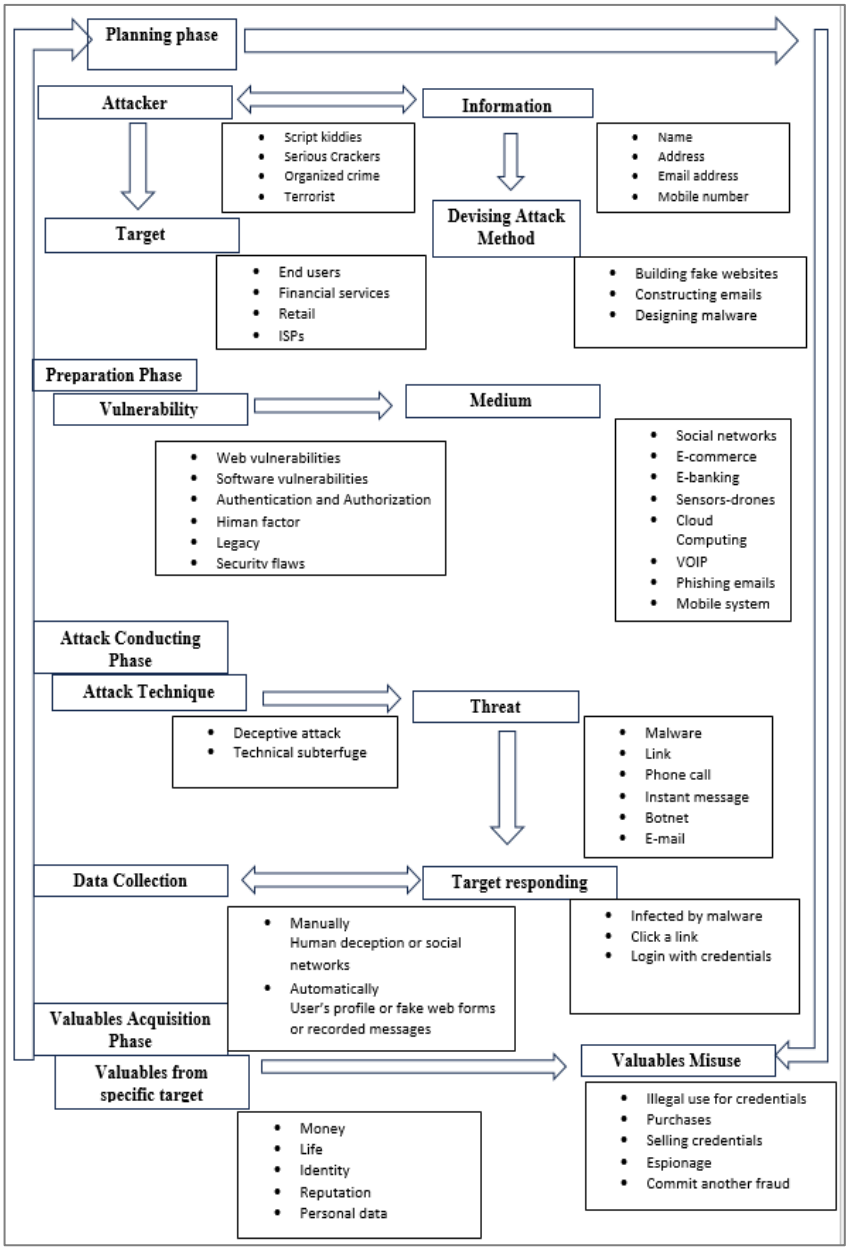

Figure 13: Phishing attack types and techniques

```
<body>
    <main class="container" role="main" aria-labelledby="title">
        <h1 id="title">Phishing Detection Software</h1>
        <p>This is a Phishing Website Prediction Model</p>
        <p>Enter a URL to check if it's a potential phishing link. This tool analyzes the link and provides a prediction.</p>
        <form action="{{ url_for('predict') }}" method="POST" aria-describedby="form-instructions">
            <label for="url">Enter URL:</label>
            <input type="text" id="url" name="url" required aria-required="true">
            <button type="submit">Check</button>
        </form>
        {% if prediction %}
```

Figure 14: Screenshot of the User Input module

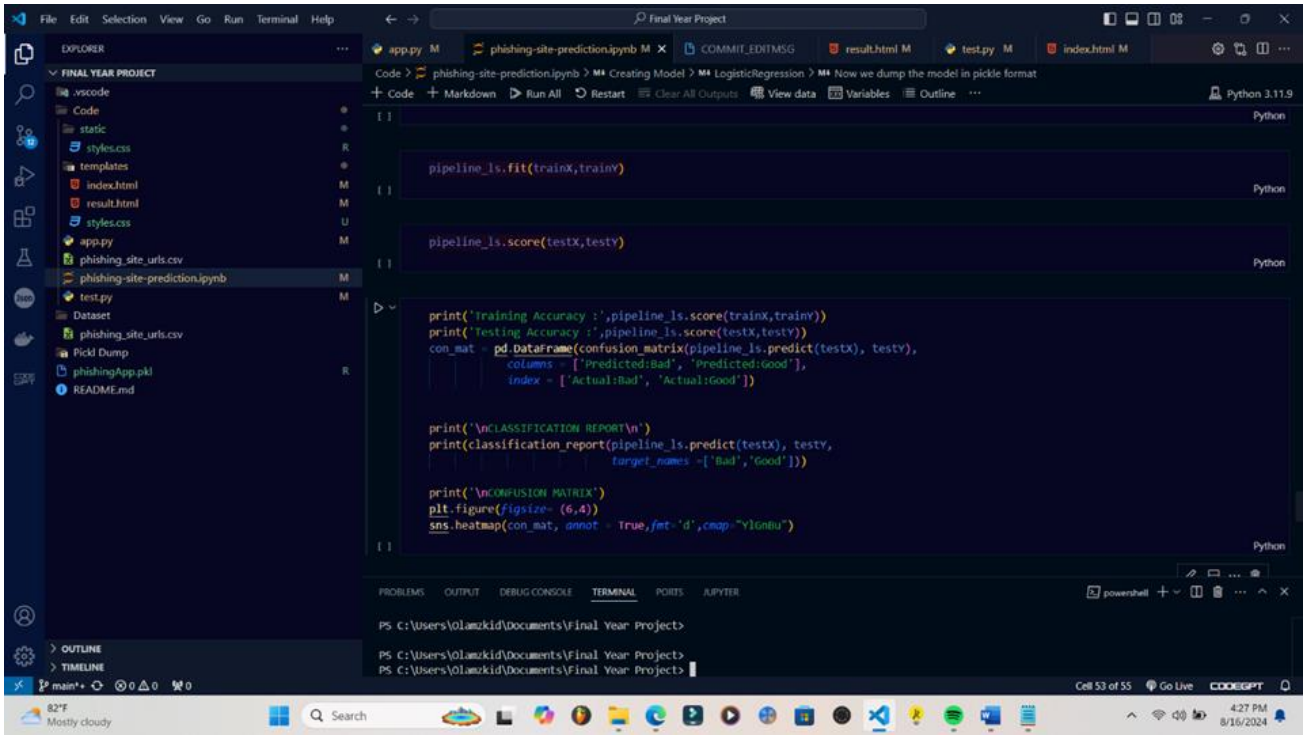

Figure 15: Screenshot of the prediction module (Logistic Regression)

```
if request.method == 'POST':
    url = request.form['url']

    # Predict using the loaded model pipeline
    prediction = model.predict([url])

    return render_template('result.html', prediction=prediction[0])

if __name__ == '__main__':
    app.run(debug=True)
```

Figure 16: Screenshot of the response module (A separate HTML file deployed with Flask)

To assess real-world performance, stress testing was conducted using Apache JMeter with simulated concurrent users, measuring mean latency, 95th and 99th percentile response times, and throughput on a standard workstation environment. Ethical considerations were addressed by avoiding the collection of personal user data, anonymizing system logs, and restricting data retention. In addition, the developed password generator and checker similarly used python enabling organizations to be able to verify their Password combinations, password strength while educating individuals on the vital roles of passwords in safeguarding organizations sensitive transactions data. Table 1 summarized related literature (contributions and limitations) on phishing detection.

3.1 Research Design

This section presents the developed mechanism design, as depicted in Figures 4, 5, 6, 7, and 9, respectively.

Figure 4 depicts the Proposed Phishing Detection Mechanism. In the diagram a user is an actor who

interacts with the software mechanism. The Phishing Detection mechanism to another actor representing the developed system itself. The mechanism features include Uploading data, so that users can upload suspicious URLs or files for analysis. View Reports, which allows users to view the reports generated by the system. The developed system scans websites for potential phishing content and generates reports based on its analysis and findings.

Figure 17: Screenshot of the model management module (pickle)

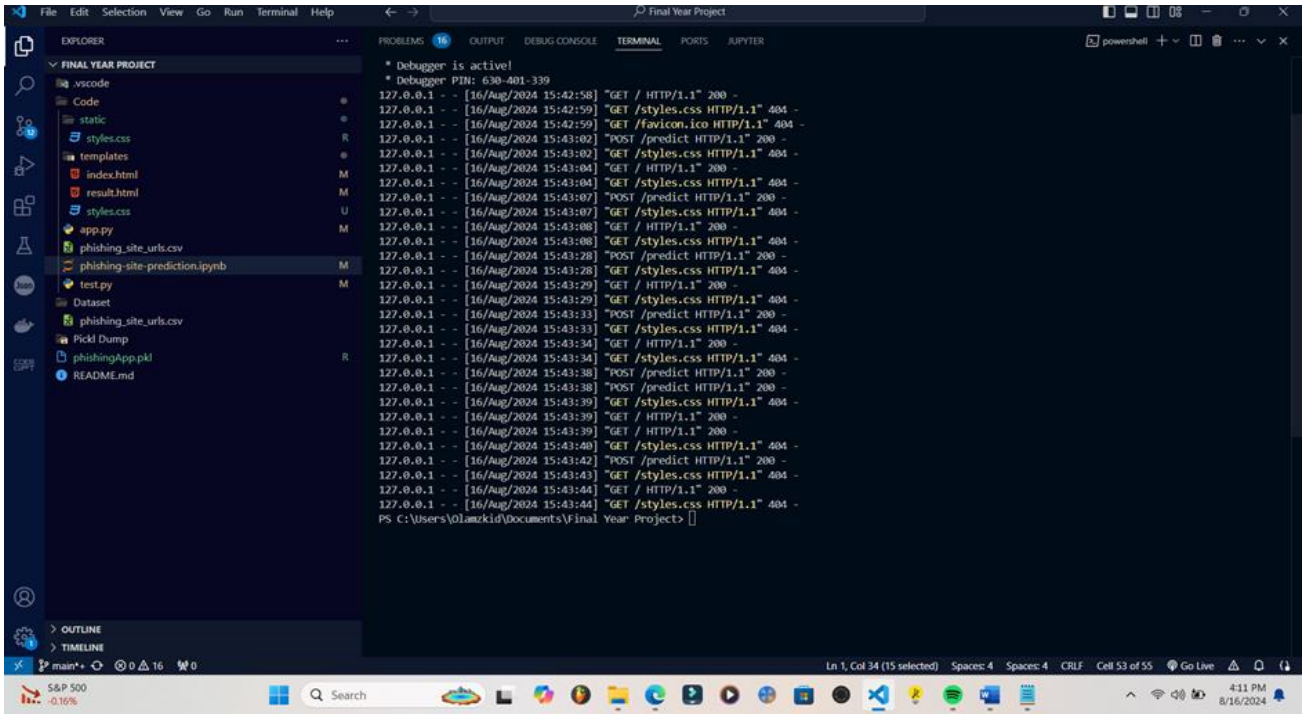

Figure 18: Screenshot of the logging and analytics module (vs code Terminal)

The Entity relationships diagram depicted in Figure 5 shows how information flows between them in the context of a Phishing Detection Mechanism, indicating how one user can be associated with multiple reports, such as one-to-many relationships. The relationship indicates that a report is associated with a phishing website. This implies that the report contains information about the detection of phishing content on a specific website. Also indicating that one user can initiate multiple scan operations with a one-to-many relationship. These relationships indicate that a scan result is associated with a scan operation and a report. This implies that the scan result is generated as an output of a specific scan operation and is related to the corresponding report. The relationship also indicated that a feedback entry is associated with a report. Implying that the feedback is related to a specific report, allowing users to provide input on the detection results.

In Figure 6, the User interacts with the Phishing Detection system by initiating a website scan and viewing the report. The system analyzes the website content, checks for phishing indicators, determines the likelihood of phishing, and generates a report. Once the report is ready, the mechanism notifies the user, who then views the report to see the details of the phishing detection analysis.

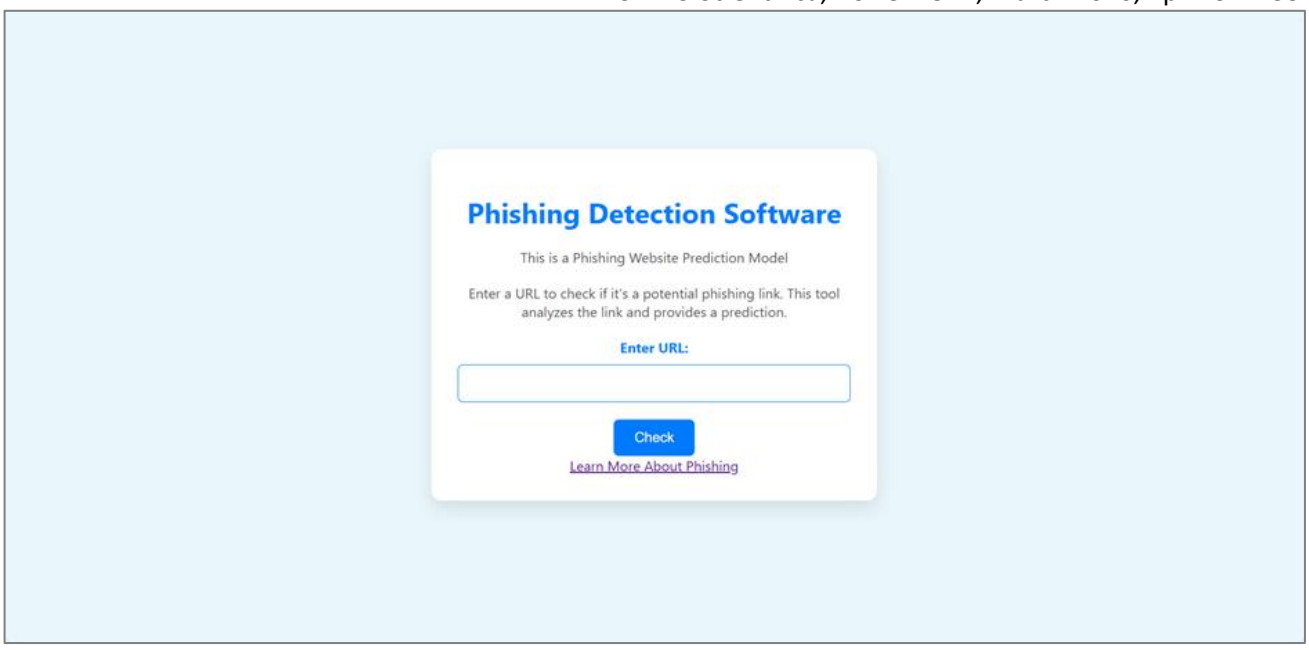

Figure 19: Screenshot of the User Interface module

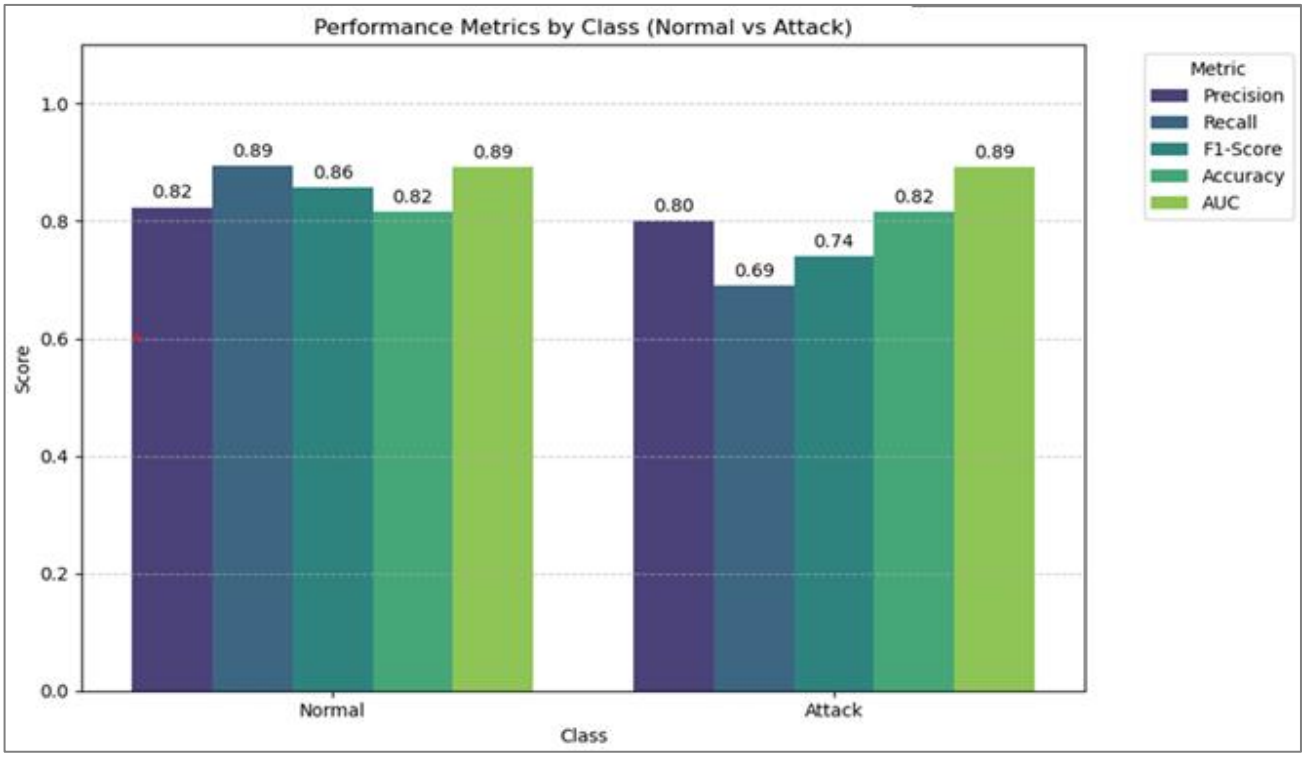

Figure 20: **Barchart for model performance**

From the Proposed Phishing Detection mechanism sequence diagram depicted in Figure 7. The User actor initiates a scan by calling the Initiate Scan (URL) method on the Phishing Detection Strategy. The Phishing Detection mechanism activates and calls the Start Scan (URL) method on the Scan object. The Scan object activates and calls the Generate Result method on the Scan Result object to generate the scan result. The Scan Result object returns the result to the Scan, which then generates a report using the Generate Report method on the Report object. Finally, the Report object sends the report to the User for viewing.

As depicted in Figure 8 the initial state is user Idle, where the user is inactive. The user can then transit either from User Idle to initiate Scan when they initiate a scan. They initiate Scan state transitions to Scanning, representing the scanning process. After scanning, the system transitions to Generating Result and then to Result Ready when the scan result is ready. From Result, the system then generates a report and transition to Report Ready. Finally, the user then views the report and then transition back to User Idle to state.

Figure 9 depicts the Proposed Phishing Detection mechanism procedure; the process starts when the user initiates a scan. The system then scans the website for phishing content. After scanning, the system generates a Scan result. Based on the scan result, the system generates a report. Where a user can both view the report, and complete the process.

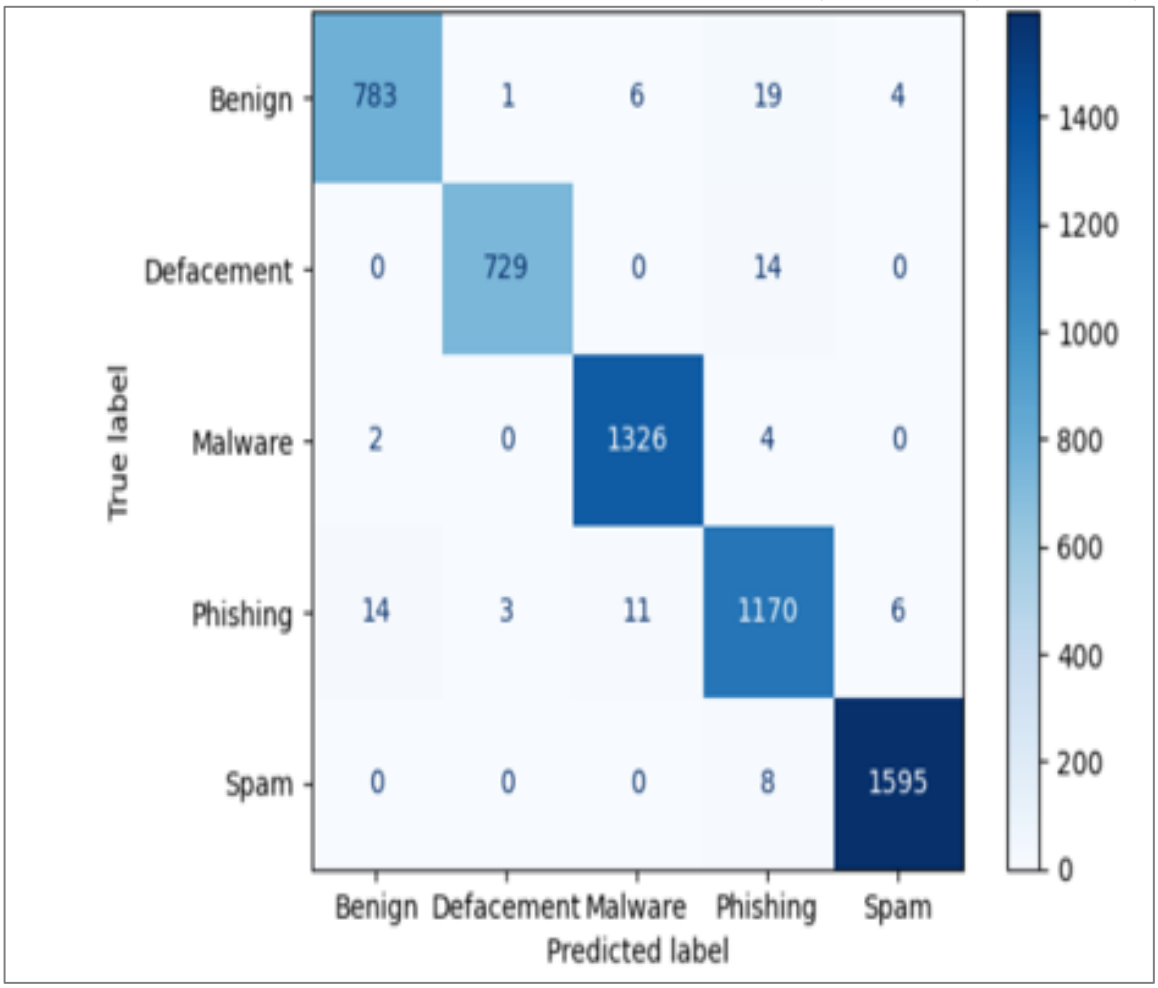

Figure 21: **Confusion Matrix**

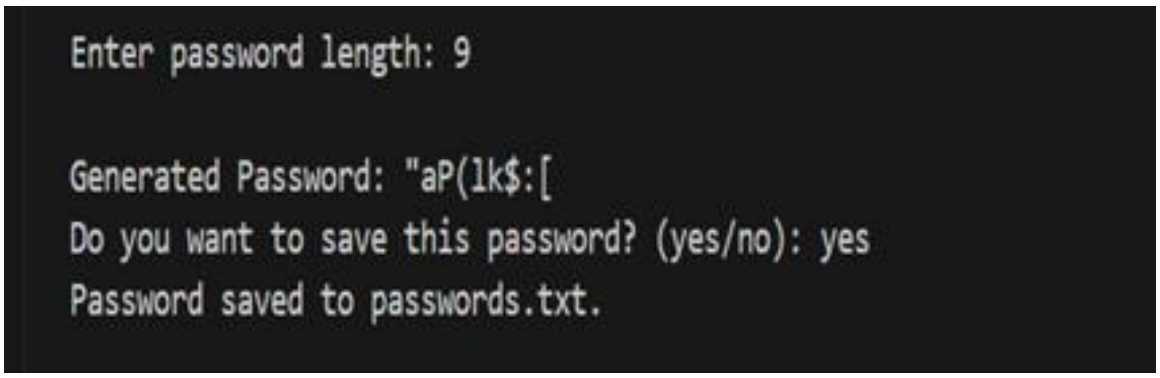

Figure 24: Depicts saving the password to a text file. If the user selects the option to not save the password, the program gracefully returns to the main menu.

3.2. System analysis

System Components of a Phishing Detection Mechanism includes Data Collection Module, this module gathers data from various sources such as emails, websites, and network traffic. As well as information gathered from users as flagged phishing emails. ML Model, which was trained on the selected features to classify incoming data as either phishing or legitimate, was then employed. The algorithms used include decision trees, random forests, support vector machines, and neural networks. The developed software mechanism included a user interface that allows users to interact with the system, view the results of the phishing detection, and manage settings and configurations.

Algorithm 2: Phishing Attack Strategies

1. Initialize an attack
2. Creating phishing emails, web links
3. Send a phishing email to the victim (s)
4. Receive email from the victim
5. The target is asked for sensitive information
6. The victims disclose their sensitive information
7. Defrauding the victim (s) successfully
8. The phisher uses the data for malicious purposes

9. Serious Financial losses, organizations reputation Damage etc.

3.3 Implementation Phase

The implementation phase of the developed Phishing Detection mechanism involved the process of creating and deploying the software. This phase includes the following steps, namely, Data Collection and Preprocessing, data were Gathered and preprocessed from various sources, such as phishing emails, websites, and network traffic, to prepare it for analysis. The developed mechanism chooses an appropriate ML model, namely, decision trees, random forests, and neural networks and train the ML model on the preprocessed data to detect phishing attacks. Figure 10 depicts Data Pre-processing stages.

Deployment and Integration: Deployed the trained model into a production environment and integrate it with existing systems for real-time phishing detection and response.

This phase includes namely, Unit testing, which is the process of testing individual units of code to ensure that they are working properly. This includes functions, classes, and modules.

Integration testing is the process of testing how different units of code interact with each other. This includes testing how different functions, classes, and modules work together.

System testing is the process of testing the entire software system to ensure it functions properly. This includes testing the user interface, the functionality of the scheme, and the performance of the developed mechanism.

Acceptance testing is the process of testing a mechanism to ensure it meets the user's requirements. This included testing the software against the requirements defined in the software requirements specification (SRS) following the steps outlined. We ensured that the mechanism was tested properly and was working perfectly.

4 Proposed Mechanism

This research suggests three strategies to address phishing attacks: identify them before users encounter them, identify them once users arrive at the phishing website, or teach users how to identify or stop phishing attacks on their own. While each alternative has its own positive and negative aspects, a combination of all three is the most effective strategy. By tackling all three, we increase the chances that phishing attempts will be detected and stopped, as they continue to evolve to evade detection and get past these defenses. The developed strategy via the proposed anti-phishing detection strategy is depicted in Figures 4 and 11.

Algorithm 3: Phishing Detection Mechanism

1. Analyze URL links
2. Classify URL Links as malicious or good
3. Employ ML Mechanism to filter malicious links
4. Secure your organization or Personal device using ML Mechanism
5. Engage in stakeholders Training to stay informed on Emerging phishing variant strategies

Step 1 – DETECT PHISHING

In the first step of the developed strategy to counter phishing, the focus is on detecting phishing attempts by identifying potential phishing sites or by providing users with warnings about malicious sites, even after they have opened a potentially harmful email. Many web browsers incorporate defenses against phishing, employing either passive indicators or active indicators. Passive indicators do not interrupt the user's task and may go unnoticed, whereas active indicators, such as pop-up windows that warn of suspected forgery or unsafe sites, are more effective. However, users may still trust initially trusted sites, making a verification system for secure sites essential. Implementing a certification and branding system that users regularly see on legitimate sites can help them notice the absence of such verification on fake websites, enhancing user awareness and confidence in website authenticity.

Step 2 – PREVENT PHISHING

The Secondary step in preventing phishing involves proactive measures such as blacklisting or blocking phishing sites and filtering out phishing emails through manual or automated methods, including ML. While blacklisting sites may not catch all phishing attempts, the more effective approach is to block phishing emails to prevent users from being exposed to malicious links. Successful spam filters are widely used by email servers, but developing effective phishing filters is challenging due to the complexity of phishing. ML techniques, as demonstrated in the study 'Classification of Phishing Email Using Random Forest ML Technique,' analyze various characteristics to identify phishing emails. The experiment reported an accuracy of 99.7% and a false-positive rate of approximately 0.06%, indicating the effectiveness of ML in combating phishing, especially in adapting to evolving attack strategies.

Step 3 – STAKEHOLDER TRAINING

The third approach is precisely the developed techniques that involve organizations and stakeholder training to counter phishing. Traditional training methods are often generic and ineffective against advanced phishing attacks, as users tend to disregard emailed warnings. Thus, to address this important problem, this research proposed innovative anti-phishing training through engaging games or embedding training systems into email servers. Games such as 'Anti-Phishing Phil', have proved successful in teaching users to identify suspicious elements in phishing scams, thereby mitigating persistent attacks. More so, embedded training involves sending mock phishing emails, allowing organizations or individual users to learn by encountering potential threats. This research aims to educate organizations and individual users about the consequences of phishing attacks and to develop

measures to analyze, identify, and detect fake links to mitigate the potential threats posed by cybercriminals. The developed model was designed and trained to evaluate links and classify them as malicious or benign. The developed mechanism evaluates each URL, identifying various links with flaws and providing tips to avoid future phishing attempts. The intelligent mechanism is able to detect and reveal malicious fake sites or links to organizations or users, thereby transforming potential phishing victims into well-informed individuals and organizations, while enhancing awareness and minimizing potential vulnerable organizations and individuals to phishing threats.

5. Implementation

The detailed development of the Phishing Detection mechanism was presented in this section. The theoretical ideas presented in the preceding sections were put into practice during the implementation phase. The system requirements, implementation tools, development processes, program modules, and user interface used in the mechanism development is presented as follows. Establishing the necessary system requirements to ensure the software functions optimally encompasses both hardware and software aspects.

Minimum Hardware Requirements, Processor Speed: A multicore processor with a clock speed of at least 2.0 GHz was used. The processing power was crucial for efficiently analyzing incoming data, handling multiple detection algorithms simultaneously, and processing real-time traffic data. The system had a minimum of 8 GB RAM for Adequate memory, which was vital for storing and processing large datasets, running ML models, and managing high-volume traffic data without latency. A solid-state drive (SSD) with at least 256 GB of storage was used. The SSD provides faster read/write speeds, which were crucial for swiftly accessing and updating databases, log files, and threat intelligence feeds. The Processor Speed of 2.0 GHz, 2.5 GHz, or higher was used. A Memory Capacity of 8 GB RAM or higher and a Storage Space of at least 256 GB or higher were also used. Reliable, high-speed internet connectivity was used for real-time monitoring and data exchange with threat intelligence sources, enabling timely detection and response to phishing attempts.

Table 2 shows the software requirements used for the developed scheme successful implementation. Operating System, the software was compatible with various operating systems, including Linux, Windows, and macOS. However, Linux was preferred for its security, stability, and wide support for networking and monitoring tools.

5.1 The Development Environment

VS Code was used as the primary code editor whereas Jupyter Notebook was used for exploratory data analysis and model development. Python was used as Core language for implementing the phishing detection logic, data processing, and the ML model. Python Libraries and models, such as Pandas and NumPy, were used for Data manipulation and analysis. Matplotlib, Seaborn for Data visualization. Scikit-learn was also used for the Implementation of Logistic Regression, Multinomial Naive Bayes, and other models. NLTK for Text preprocessing, including tokenization and stemming. Pickle for Model serialization and later use. Flask as the backend framework, serving the web interface of the phishing-detection software mechanism. The Frontend of this web application was developed using JavaScript, HTML, CSS. While Heroku was used for deploying the phishing detection mechanism, making it accessible and visible online.

Agile Methodology was used which is an iterative and incremental approach focused on flexibility, collaboration, and delivering value quickly. It involves breaking the project into smaller, manageable tasks called user stories, which were prioritized and worked on in short development cycles called sprints. Agile methodologies, such as Scrum and Kanban, were ideal for studies where requirements may evolve over time or where regular stakeholder feedback was essential.

5.2 Program Modules and Interface

The phishing detection software's architecture comprised distinct modules that collectively contributed to its functionality. The User Input Module, Captured and validates user input, specifically URLs. While the Functionality, Accepts URLs entered by the user through a text box. Ensure that the input has a valid URL format, then preprocess it for analysis. The validated and preprocessed URL is then passed to the Feature Extraction Module for further analysis.

Feature Extraction Module: Purpose: Extracts relevant features from the input URL for model analysis.

- Functionality: Analyzes the URL to extract features such as domain length, presence of special characters, use of IP addresses, and suspicious keywords. Convert these features into a numerical format suitable for the machine learning model.
- Interaction: Sends the extracted features to the Prediction Module for classification.

The Prediction Module predicted whether the input URL was a phishing or legitimate URL. This uses a pre-trained ML model, namely Logistic Regression or Multinomial Naive Bayes, to classify a URL as "Good" or "Bad" based on extracted features. And received the features from the Feature Extraction module, as depicted in Figure 15, and returned the prediction result to the Response Module.

The Response Module, Displays the result of the URL analysis to the user, presenting the prediction outcome as ("Good" or "Bad") along with a brief explanation or suggestion to the user. If the URL is classified as phishing, the scheme provides safety tips or warnings while Receiving the prediction result from the Prediction Module and delivering it to the user interface.

The Management Module manages the training, updating, and deployment of the ML model, handling model

updates, retraining with new data, and optimization to improve prediction accuracy. Also manages version control of the deployed model. I interacted with the Prediction Module to ensure the latest model was used for URL classification.

Logging and Analytics Module: Logs user inputs to predict outcomes for future analysis and improvement. Records the URLs analyzed, their predictions, and timestamps, which data are used for model performance tracking, user behavior analysis, and identifying potential improvements. Operates independently to collect and analyze data across the system, providing feedback for future model updates.

The User Interface Module provides a user-friendly interface for inputting Universal Resource Locators (URLs) and viewing results. Offering a text box for URL input, a button to trigger analysis, and displaying the prediction result. Ensures smooth user experience with clear instructions and responsive design. Interfaces with the User Input Module to capture user input and the Response Module to display the prediction result, respectively.

The bar chart in Figure 20 is a visual representation of the performance of the ML model trained with ML classifier algorithm. The model was trained with Python with Jupyter Notebook from Anaconda Navigator, using (30%) for training the dataset and (70%) for validation. It includes the score or performance metric value on the Y-axis and the two Different classes it predicts on the X-axis. The bar chart helps understand how well the scheme can properly predict the classes used in training.

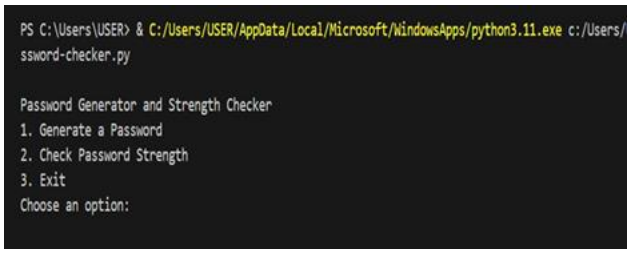

Figure 22: The list of operations to be carried out. The user decided to generate a password or check its strength.

The confusion matrix in Figure 21 shows how well the model classifies all the Classes (benign, Defacement, Malware, Phishing, and Spam). The diagonal line running from left to right shows the correct predictions, while the rest show the errors. The X-axis shows the True label, while the Y-axis shows the predicted label or what the true label was predicted to be by the model. For example, the first row shows that 783 actual benign samples were correctly classified as benign, 1 actual benign sample was misclassified as Defacement, 6 Benign samples were misclassified as Malware, 19 actual benign samples were misclassified as Phishing, and 4 actual benign samples were misclassified as Spam. The bar on the right shows the range of values in the matrix (0-1400).

Table 3 Depicts the actual variant classes (normal, attack, accuracy, macro aug, and weighted aug), as well as the precise output results of the analysis, namely Precision, Recall, F1 Score, and Support.

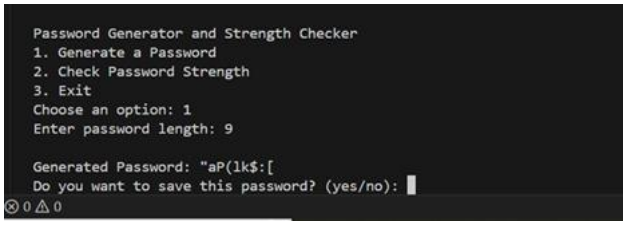

Figure 23: The first operation (password generation) is being carried out. The user enters the desired password length, and the system generates a random number of characters from the pool, matching the specified length. Then the user has to decide whether to save the password.

Figure 25: Password strength and entropy check-being carried out. The entropy of a password is a measure of how hard it is to guess. The higher the entropy, the stronger the password and the harder it is to guess, i.e., the password is less vulnerable.

RESULTS AND DISCUSSION

This paper represents a significant advancement in cybersecurity, combining Python with machine learning (ML) to enhance the security of business transactions in the digital age. This study aims to improve awareness, safeguard users, and enable secure transactions for organizations, businesses, financial institutions, and private individuals, mitigating potential financial losses from cybercriminals while protecting online trust and optimizing cybersecurity resilience using ML techniques. The study tries to educate and inform organizations on the important role of passwords, and the potential consequence of phishing attacks, safeguarding individual organizations, on the use of URL links, websites, as well as the importance of correct password combinations, password length, and why it all matters in the security of business transactions. The emerging risks associated with the use of the internet, websites, and links are mitigated by introducing an intelligent phishing-detection scheme to preserve privacy, enhance financial confidentiality, and increase trust, while encouraging improved security mechanisms and driving security innovations. Thus, an intelligent phishing detection scheme was developed. The mechanism evaluates Universal Resource Locators (URLs) entered by users and carefully classifies them as legitimate (Good) or malicious (Bad) sites. The study employed an algorithmic process state using ML models, offering a clear framework that illustrated the system's

effectiveness in accurately recognizing and categorizing URLs. The developed scheme incorporated essential features to manage vital datasets used to identify patterns and provide accurate predictions. Stress testing, load testing, and reaction time analysis were among the rigorous performance testing scenarios used to assess the scheme scalability and dependability. These tests were essential for determining how well the system would perform under various conditions and demands, ensuring the developed scheme would remain responsive and reliable even during peak usage. The developed scheme employed simulation to test how multiple users interact with the system simultaneously through load testing. Kaggle datasets were used and split into training (30%) and validation (70%) sets in Python. The study assessed how effectively the system handled increased traffic without a decrease in performance.

The goal of reaction time analysis was to determine how quickly the program interpreted user input and produced predictions, providing prompt, effective answers that enhance the user experience. Exploring the developed scheme beyond its typical operational limits, stress testing assessed the scheme's resilience in harsh environments and helped identify potential breaking points. Identifying potential weak points before unanticipated surges in usage, this procedure made it possible to make proactive modifications and optimizations. The scheme's readiness for real-world deployment was demonstrated by the successful completion of performance tests, which verified that it could handle a range of user input levels while maintaining high performance standards. This extensive testing increased the system's overall credibility and efficiency, endorsing the developed scheme's scalability and durability with assured performance even in emergency situations. The scheme organized and processed data, and developed an intelligent system that aided in phishing detection, informing organizations while protecting users from phishing threats.

The developed phishing detection scheme demonstrates improved performance compared to existing approaches reported in (Siti, *et al.*, 2020; Raj, & Jothi 2022; Nanda, *et al.,* 2024) Many earlier phishing detection systems relied on single classifiers such as Support Vector Machines (SVM) or Decision Trees and typically achieved accuracy levels between 94% and 97% when trained on phishing datasets. In contrast, the developed model, trained on Kaggle and PhishTank datasets, achieved higher classification performance due to improved feature extraction and model learning. The integration of diverse datasets enabled the system to learn more general phishing patterns, leading to better generalization in distinguishing malicious URLs from legitimate ones.

Another key improvement is the developed scheme's ability to capture complex relationships among URL-based features. Existing systems often focus on a limited set of structural features, thereby reducing their effectiveness against sophisticated phishing URLs that employ obfuscation or domain-manipulation techniques. The developed scheme utilized a richer feature set and an optimized training process, enabling it to identify subtle patterns such as abnormal domain structures, excessive subdomains, and suspicious character usage. As a result, the developed scheme demonstrated higher detection accuracy and improved robustness compared to earlier ML-based phishing detection Schemes.

Algorithm 4: Password Combinations

1. Password Length: Use at least 12–16 characters.
2. Uppercase Letters, Includes A–Z.
3. Lowercase Letters, Includes a–z.
4. Password Numbers, Add 0–9.
5. Special Characters, use symbols like @, #, !, %.
6. Avoid using names or dictionary words in password generation
7. Mix character types of randomly stronger security

When generating passwords, always try to make them longer and use correct password combinations. With at least twelve (12) to sixteen (16) characters, including both uppercase and lowercase letters, numbers, and special symbols, is essential; more special symbols are difficult to predict. Avoid using the same words repeatedly to generate a password, no matter how long it appears, such as names or other easy-to-guess information. When creating a password, your goal should always be to create a secret code that only you can remember. The more unpredictable and difficult it is, the better it is for any attacker to crack. And from time to time, change your passwords for safety reasons.

KEY FINDINGS

This research developed an intelligent phishing-detection scheme that assesses the Universal Resource Locators (URLs) entered by users and correctly classifies them as legitimate (Good) or malicious (Bad). The research methodology used machine learning (ML) models, providing a clear framework that demonstrated the system's effectiveness in accurately recognizing and categorizing URLs. (30%) of the dataset was used for training, while (70%) used for validation. These models were essential to the program because they manage enormous datasets, identify patterns, and provide accurate predictions. Stress testing, load testing, and reaction time analysis were among the rigorous performance testing scenarios used to assess the mechanism's scalability and dependability. These tests were essential for determining how well the program would perform under various stressors and demands, ensuring it would remain responsive and reliable even during periods of high usage. By simulating numerous users interacting with the software at once through load testing, assessing how effectively the system handles an increase in traffic without experiencing a decrease in performance. The reaction time analysis aimed to determine how quickly the program interpreted user input and produced predictions

to provide prompt, effective answers that enhance the user experience. By engaging the developed mechanism beyond its typical operating limits, stress testing assessed the scheme's resilience in harsh environments and helped identify potential breaking points before unanticipated surges in usage. This procedure enabled proactive modifications and optimizations. The scheme's readiness for real-world deployment was demonstrated by the successful completion of these performance tests, which verified that it could handle a range of user input levels while maintaining high performance standards. This extensive testing increased the system's overall credibility and efficacy by confirming the software's scalability and durability, giving assurance that it could be counted on in emergency situations.

The developed scheme is vital for raising awareness and training organizations' staff, as well as educating employees. Providing personalized feedback to users, helping organizations identify potential vulnerabilities and strengthen their security posture while promoting a culture of security awareness within organizations and individuals and proposed a Password Generator and checker using python.

The Develop Password Generator and Checker imports all necessary libraries and initializes the function to generate passwords using the length parameter. If the user enters a length shorter than the required password length, the system issues a warning and stops. It creates a pool of characters gotten from the string library and stores it in a variable called 'characters'. It selects random sets of characters of the user-specified length and joins them. It then stores the selected characters in a variable called 'password' and returns the 'password'. The system defines a function to check password strength using the earlier-generated password, or, as seen later in the system, the password the user enters, and checks whether it contains special characters, digits, lowercase letters, and uppercase letters. These are the vital combinations the system was designed to check to determine whether a password combination is strong or weak.

The system was designed to add points to the password's score for every requirement met. If it contains digits, it gets assigned 1 point; if it contains special characters, it gets another point. It calculates the password's entropy by multiplying its length by the base-2 logarithm of the number of characters in the earlier generated pool. The system prints a detailed analysis of password education for organizations' staff, emphasizing the importance of strong passwords and the requirements. The developed system informs the organization whether the password contains letters, digits, symbols, etc., and displays the password length and its entropy. Returns whether a password is strong, weak, or moderate based on the earlier-calculated score. It defines the function to save passwords to the file 'passwords. txt'and uses a try-exception block to ensure the function runs properly. It opens the file in append mode and saves newly generated passwords on a new line. Saving and closing the file properly displays a message when it is done. If there is any error preventing this from happening i.e., an exception, it displays an error message and continues to try until it is successful. The system displays the main menu options, initializes the main function, and prompts the user to choose an option. It runs if the user selects the first option- to generate a password and prompts the user to enter the password length, printing the generated password and asking the user if they want to save the password. If they decide to save the password, it calls the function to save the password to the file. Otherwise displays an error message if the user enters a letter instead of a number when making a choice. The system displays options. If the user selects appropriate options, it prompts them to enter the password they want to check and then calls the function to check the password strength. It then prints the password's strength. If the user selects done, a goodbye message is displayed, and the program dismisses gracefully, printing an error message if the user enters an invalid option. It comprehensively educates organizations and individuals about emerging cyber threats and the important role of proper password combinations in potentially mitigating them.

CONCLUSION

An intelligent phishing-detection scheme was developed; its successful implementation marks a significant improvement in cybersecurity practices. The paper utilized a range of tools and technologies, namely a development environment, VS Code, and a Primary code editor. Jupyter Notebook: Used for exploration, data analysis, and model development. Python Programming Language. Core language for implementing phishing detection logic, data processing, and machine learning (ML) models. ML Libraries and Models, Pandas, NumPy For data manipulation and analysis. Matplotlib, Seaborn, for data visualization. Scikit-learn, Implementation of Logistic Regression, Multinomial Naive Bayes, NLTK, for text preprocessing, including tokenization and stemming. Pickle, for model serialization for later use. Flask, which was the Framework for building the web interface of the phishing detection Mechanism. For the front-end, Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS) were used to design the user interface. Heroku was the Platform used to deploy the phishing detection software, making it temporarily accessible online on the free plan. The integration of these tools resulted in a robust and efficient system capable of detecting phishing URLs with high accuracy. The mechanism successfully achieved its objectives of providing users with a way to identify malicious URLs and enhance their online security. The developed mechanism, including a password generator and checker, was designed to educate employees within organizations about the risks and tactics of phishing attacks, thereby enhancing the security of their financial data. While mitigating persistent phishing attacks with the proper use of well-informed password combinations. More so;

- The developed scheme achieved its best results using the top-performing model, which recorded approximately 98% accuracy, with strong precision and recall in distinguishing phishing

URLs from legitimate ones. This performance indicates improved detection capability compared to many traditional machine-learning approaches reported in related phishing detection studies.

- The system was trained and evaluated using publicly available phishing datasets from Kaggle and PhishTank. Using these widely recognized datasets improves reproducibility and allows other researchers to validate or extend the model using the same benchmark data sources.

- Future work will focus on adversarial robustness testing, integrating real-time phishing feeds, and deployment in practical environments, such as browser extensions or email filtering systems.

LIMITATIONS

The mechanism potential challenges in handling highly obfuscated URLs and the need for continuous updates to keep pace with evolving phishing techniques are limitations in this study.

RECOMMENDATIONS

Continuous Model Training, such as regular updates, updating the ML models with new phishing and legitimate URL data to maintain high prediction accuracy. Implement strong data security measures to protect user input data and ensure the privacy of users interacting with the Mechanism. Continue refining the user interface to ensure ease of use, with clear instructions for entering URLs and interpreting the findings. Expand the software's mechanisms to handle a higher volume of requests, ensuring consistent performance under heavy load. Consider developing browser extensions or plugins that integrate the phishing detection mechanism directly into web browsers for real-time URL analysis. Explore partnerships with existing cybersecurity platforms to enhance detection capabilities and broaden their scope.

ACKNOWLEDGEMENT

The authors wish to appreciate the chief editor and the reviewers who made comments that has improved the research.

ETHICAL STATEMENT

This study does not include any studies with human or animal subjects conducted by any of the authors.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest in this work.

REFERENCES

Abdillah, R. (2022). Phishing classification techniques: A systematic literature review. *Center for Cyber Security (CYBER), Faculty of Information Science and Technology*, 5–8. [Crossref]

Ahmad, F., Yousuf, A., & Baig, S. (2021). Machine learning-based phishing website detection and filtering: A critical review and future directions. *Journal of Network and Computer Applications, 193*, Article 103092.

Ahmad, M. A., Wisdom, D. D., & Isaac, S. (2020). An empirical analysis of cybercrime trends and its impact on moral decadence among secondary school level students in Nigeria. In *The 26th iSTEAMS Bespoke Multidisciplinary Conference, Accra, Ghana*. [Crossref]

AlEroud, A., & Karabatis, G. (2020). Bypassing detection of URL-based phishing attacks using generative adversarial deep neural networks. In *Proceedings of the Sixth International Workshop on Security and Privacy Analytics* (pp. 53–60). [Crossref]

Alhassan, S., Akinyemi, A. E., & Wisdom, D. D. (2020). A comparative performance study of machine learning algorithms for efficient data mining management of intrusion detection systems. *International Journal of Engineering Applied Sciences and Technology, 5*(6), 85–110. [Crossref]

Aljofey, A., Jiang, Q., Qu, Q., Huang, M., & Niyigena, J. (2020). An effective phishing detection model based on character-level convolutional neural network from URL. *Electronics, 9*(9), Article 1514. [Crossref]

Canfield, Haddai, & Treinen. (2021). A review of adversarial attacks and defenses for machine learning in cybersecurity. *The International Journal of Information and Computer Security*, 18–46.

Das, A., Baki, S., El Aassal, A., Verma, R., & Dunbar, A. (2020). A comprehensive reexamination of phishing research from the security perspective. *IEEE Communications Surveys & Tutorials*, (1), 671–708. [Crossref]

Farooq, M. S., & Jabbar, H. (2024). Phishing website detection using a combined model of ANN and LSTM (No. arXiv:2404.10780). *arXiv*. [Crossref]

Gupta, B., & Islam, M. (2020). A survey of machine learning approaches for phishing detection. *Artificial Intelligence Review*, 3433–3474.

Hassan, Y., & Abdelfettah, B. (2017). Using case-based reasoning for phishing detection. *Procedia Computer Science, 109*, 281–288. [Crossref]

Hassan J. B., Ayetuoma, I. O., Wisdom, D. D., Ugwunna, C. O., Falayi, C. F., & Salaudeen, L. G. (2025). Securing critical infrastructure: The impacts of 5G networks on internet security in the oil and gas industry: A survey. *KASU Journal of Computer Science (KJCS), 3*(2).

Hong, J., Kim, T., Liu, J., Park, N., & Kim, S. (2022). Phishing URL detection with lexical features and blacklisted domains. In *Autonomous Secure Cyber Systems*. Springer. [Crossref]

Hung, L., Quang, P., Doyen, S., & Steven, C. (2017). URLNet: Learning a URL representation with deep learning for malicious URL detection. *Conference'17, Washington, DC, USA*. arXiv:1802.03162.

Kumar. (2020b). A survey on web phishing detection methods. *ACM Computing Survey*, 1–37.

Kumar, J., Santhanavijayan, A., Janet, B., & Bindhumadhava, B. (2020). Phishing website classification and detection using machine learning. In *International Conference on Computer Communication and Informatics (ICCCI)*. IEEE. [Crossref]

Makkar, A., Kumar, N., Sama, L., Mishra, S., & Samdani, Y. (2021). An intelligent phishing detection scheme using machine learning. In D. Giri, R. Buyya, S. Ponnusamy, D. De, A. Adamatzky, & J. H. Abawajy (Eds.), *Proceedings of the Sixth International Conference on Mathematics and Computing*. Advances in Intelligent Systems and Computing (Vol. 1262). Springer, Singapore. [Crossref]

Nanda, M., Saraswat, M., & Sharma, P. (2024). Techniques for detecting intelligent phishing attempts in an insecure socially engineered network environment. *Recent Patents on Engineering*. [Crossref]

Nitesh, K., Ranjan, R., & Chatterjee, M. (2022). Phishing detection using machine learning: A survey of techniques, challenges, and future aspects. *Networks and Security*, 142–157.

Raj, M. M., & Jothi, J. A. A. (2022). Hybrid approach for phishing website detection using classification algorithms. *Paradigm Plus, 3*(3), Article 2. [Crossref]

Rao, R., & Pais, A. (2019). Jail-Phish: An improved search engine-based phishing detection system. *Computers & Security*, 67–83. [Crossref]

Shirazi, S. (2018). A review on phishing detection using machine learning. *International Journal of Machine Learning and Cybernetics*, 1591–1605.

Siti, H. A., Jamaludin, S., & Roslina, M. S. (2020). Types of anti-phishing solutions for phishing attack. *Faculty of Computing, College of Computing and Applied Sciences*, 1–15.

Wang, W., Zhang, Y., & Li, Z. (2020). Phishing detection based on machine learning: A review and comparison. *IEEE Access, 8*, 149273–149282.

Wisdom, D. D., & Vincent, O. R. (2024b). Cybersecurity concerns and risks in emerging healthcare systems. In *Cybersecurity in Emerging Healthcare Systems*. Institute of Engineering and Technology (IET), UK. [Crossref]

Wisdom, D. D., Vincent, O. R., Aborisade, D. O., & Omeike, M. O. (2025). Defensive walls against sophisticated ML-orchestrated attacks in edge computing. *Elsevier Book Chapter 11*. [Crossref]

Wisdom, D. D., Vincent, O. R., Christian, A. U., Igulu, K., Hyacinth, E. A., Odunayo, O. E., & Umar, B. (2024). Industrial IoT security infrastructure and threats. In A. Prasad, T. P. Singh, & S. D. Sharma (Eds.), *Communication technologies in IoT and their security challenges: Present and future*. Springer Nature. [Crossref]

Wisdom, D. D., Vincent, O. R., Igulu, K., Christian, A. U., Hyacinth, E. A., Baba, G. A., & Esther, O. O. (2025b). The protection of Industry 4.0 and 5.0: Cybersecurity strategies and innovations. In *CI-Industry-4.0: Computational Intelligence in Industry 4.0 and 5.0 Applications*. Taylor and Francis Group. [Crossref]

Wisdom, D. D., Vincent, O. R., Igulu, K. T., Arowolo, M. O., Hyacinth, E. A., Christian, A. U., Oduntan, O. E., & Baba, G. A. (2023). Mitigating cyber threats in healthcare systems: The role of artificial intelligence and machine learning. In *Artificial Intelligence and Blockchain Technology in Modern Telehealth Systems*. Institute of Engineering and Technology (IET), UK. [Crossref]

Wisdom, D. D., Vincent, O. R., Igulu, K. T., Baba, G. A., Oduntan, O. E., Akinyemi, A. E., & Umar, B. (2025c). Cyber-threats in healthcare systems: The role and application of artificial intelligence and machine learning. *Nova Science Publishers*.

Wisdom, D. D., Vincent, O. R., Oduntan, O. E., Hassan, J. B., Falayi, C. F., & Ajayi, T. D. (2024b). Improving security of business intelligent systems with AI and machine learning. In *SMARTBLOCK4AFRICA 2024 International Conference: Emerging and Resilient Technologies in Building and Securing African Nations*. Babcock University, Nigeria & Valley View University, Ghana, in collaboration with IEEE. [Crossref]

Yanz, Y., Zhao, Y., & Zeng, D. (2019). A comprehensive survey on phishing website detection. *ACM Computing Surveys (CSUR)*, 1–40.

Yu, Z., Liu, H., & Zhao, F. (2020). A deep learning approach for phishing detection based on ensemble learning. *International Journal of Machine Learning and Cybernetics*, 3919–3930.

Zainab, A., Chaminda, H., Liqaa, N., & Imtiaz, K. (2021). Phishing attacks: Recent comprehensive study. *Cardiff School of Technologies, Cardiff Metropolitan University, Cardiff, United Kingdom*, 12–23.