

# *Machine Learning-Based Phishing URL Detector*

---

AN EFFECTIVE SOLUTION FOR PHISHING ATTACK

Author: Do Nhat Minh  
HAN number: HAN22080120  
Supervisor: Hamza Mutaheer, Sasikumar  
Degree: Computer Science-Cyber Security  
Cohort: 2210CS

## Abstract

This research presents a machine learning-based phishing URL detection system designed to counter increasingly sophisticated cyber threats. Through an extensive literature review, the study identifies critical URL features—ranging from address bar characteristics to domain and HTML/JavaScript attributes—that distinguish malicious links from legitimate ones. A balanced dataset, composed of phishing URLs sourced from PhishTank and legitimate URLs from the University of New Brunswick, was curated to ensure unbiased model training. Fifteen distinct features were extracted and categorized into three groups to capture the nuanced indicators of phishing activity.

Multiple machine learning models were evaluated, with ensemble methods, particularly XGBoost, demonstrating superior performance by achieving a training accuracy of 86.9% and a test accuracy of 85.9%. The effectiveness of XGBoost in handling non-linear relationships and complex feature interactions underscores the importance of robust algorithm selection in cybersecurity applications. Additionally, the integration of a user-friendly web-based graphical user interface (GUI) enhances the system's practicality, enabling non-technical users to quickly assess URL safety with clear, immediate feedback.

Overall, the research bridges advanced machine learning techniques with intuitive design to deliver a scalable and adaptable phishing detection tool. The findings contribute to both academic inquiry and practical cybersecurity solutions, providing a solid foundation for future enhancements and real-world deployment.

## Contents

<b>I.</b>	<b>Introduction .....</b>	<b>3</b>
<b>II.</b>	<b>Background Research .....</b>	<b>4</b>
1.	URL.....	4
2.	Machine learning .....	6
<b>III.</b>	<b>Literature review .....</b>	<b>9</b>
1.	Machine Learning Techniques for Malicious URL Detection .....	10
2.	Feature Engineering and Selection .....	11
3.	Limitation of URL detection.....	15
4.	Focus on Phishing URLs .....	15
5.	Insight of URL malicious detection .....	16
<b>IV.</b>	<b>Methodology.....</b>	<b>17</b>
1.	Data collection .....	18
2.	Feature extracting .....	21
3.	Module selection and training .....	23
4.	GUI creation .....	26
5.	Limitations .....	29
<b>V.</b>	<b>Result .....</b>	<b>29</b>
1.	Modules training result .....	29
2.	GUI user feedback and result .....	31
<b>VI.</b>	<b>Analysis .....</b>	<b>32</b>
1.	Interpretation of Results .....	32
2.	Comparison with Literature .....	34
3.	Overall Implications and Future Directions .....	36
<b>VII.</b>	<b>Conclusion.....</b>	<b>37</b>
<b>VIII.</b>	<b>References .....</b>	<b>40</b>

# I. Introduction

Over the past decade, the rapid expansion of internet usage has transformed the way individuals and organizations conduct daily transactions and communicate. This digital evolution, while fostering convenience and connectivity, has also given rise to sophisticated cyber threats. Among these, phishing stands out as one of the most pervasive and damaging forms of cyberattacks. Phishing is a technique in which attackers impersonate trusted entities to deceive users into disclosing sensitive information—such as login credentials or financial details—by exploiting both technical vulnerabilities and human psychology (Aslan et al., 2023). Cybercriminals typically employ social engineering methods to create fraudulent emails or websites that closely mimic legitimate ones, thereby luring unsuspecting users into a trap (Gupta, Singhal, & Kapoor, 2016). These emails often contain malicious URLs that, when clicked, direct users to counterfeit websites designed to harvest personal data (Bhavsar et al., 2018). The dangers posed by phishing extend far beyond immediate financial loss; victims may also experience identity theft, data breaches, and long-term damage to their personal and organizational reputations, underscoring the critical need for robust cybersecurity measures.

To counter these threats, a variety of traditional methods have been implemented over the years. One common approach is user training, which aims to educate individuals on the warning signs of phishing attempts. Although such training programs can raise awareness, they are not foolproof; many users still fall victim to increasingly sophisticated phishing schemes despite repeated instruction (Weaver et al., 2021). Another prevalent technique is blacklisting, which involves maintaining databases of known phishing URLs and blocking access to them. While this method is relatively simple and cost-effective, its static nature makes it difficult to keep pace with the rapidly evolving landscape of phishing tactics—new malicious URLs often bypass these lists before they are updated (Arshad et al., 2021; Azeez et al., 2021). Additionally, two-factor authentication (2FA) adds an extra layer of security by requiring a secondary verification step, such as a code sent to a mobile device (Apandi et al., 2020). However, even 2FA is not immune to attack, as adversaries have developed ways to intercept or mimic the second factor (Jakobsson, 2018).

In response to these challenges, machine learning (ML) has emerged as a promising alternative for phishing detection. ML-based systems analyze vast datasets to identify subtle patterns and anomalies that signal phishing attempts, adapting continuously as attackers refine their strategies (Alhogail & Alsabih, 2021). Some studies report that advanced ML models can detect up to 98% of phishing attacks (Atlam & Oluwatimilehin, 2022), demonstrating their superior effectiveness compared to traditional static methods.

Despite the accuracy and adaptability of ML-based phishing detection systems, many of these solutions lack an intuitive graphical user interface (GUI). The absence of a

user-friendly GUI can hinder non-technical users from fully engaging with the system, limiting its overall utility and effectiveness in real-world applications. Recognizing this gap, the project aims to train a robust model that can accurately identify phishing URLs and integrate a comprehensive GUI into the ML-based phishing URL detection module. This integration is intended not only to maintain high detection accuracy but also to enhance usability by offering a clear, accessible platform for users to monitor, review, and manage potential threats. By bridging the gap between sophisticated machine learning techniques and user-centric design, this project seeks to provide a robust and accessible solution to combat phishing attacks effectively. To able to accomplish such goal, this project aims to answer these question:

1. How effective can this module be in detecting and preventing phishing attacks?
2. How can the module be designed for usability, especially for individuals without a cybersecurity background?

## **II. Background Research**

Understanding URLs and machine learning is fundamental to developing robust cybersecurity solutions in today's digital landscape. A URL, or Uniform Resource Locator, serves as the address that directs users to various online resources, and its structure—encompassing elements such as the protocol, domain, path, and query parameters—can reveal subtle indicators of legitimacy or potential malicious intent. With the increasing sophistication of cyber threats, particularly phishing attacks, machine learning has emerged as a powerful tool to analyze these URL characteristics. By training models on large datasets of both benign and malicious URLs, machine learning algorithms can learn to identify patterns and anomalies that are often imperceptible to human analysts. This integration of URL analysis with advanced machine learning techniques not only enhances the accuracy of threat detection but also provides a scalable approach to adapt to evolving cyber attack strategies.

### **1. URL**

A URL (Uniform Resource Locator) is a standardized address used to access resources on the internet, such as web pages, images, videos, or downloadable files. It serves as a reference to a specific location on a network, enabling users and applications to retrieve online content. URLs are essential for web navigation, as they allow browsers to locate and load websites efficiently.

## Structure of a URL

A typical URL consists of several components that help direct users to the correct resource:

- **Protocol (Scheme):** The method used to communicate with the server, such as `http://` or `https://`. Secure websites use HTTPS (Hypertext Transfer Protocol Secure), which encrypts data for safer browsing.
- **Domain Name:** The main web address that identifies a website (e.g., `www.example.com`). This can also be an IP address, though domain names are more user-friendly.
- **Port (Optional):** A number specifying the communication endpoint (e.g., `:443` for HTTPS). Most URLs omit this unless a non-default port is used.
- **Path:** The specific location of a resource within the website (e.g., `/about-us` in `www.example.com/about-us`).
- **Query Parameters:** Additional data included in the URL, typically used to send information to the server (e.g., `?search=shoes&page=2`). These parameters help customize responses based on user input.
- **Fragment (Anchor):** A section identifier within a webpage (e.g., `#contact` in `www.example.com/about#contact`), allowing browsers to jump directly to a specific part of the page.

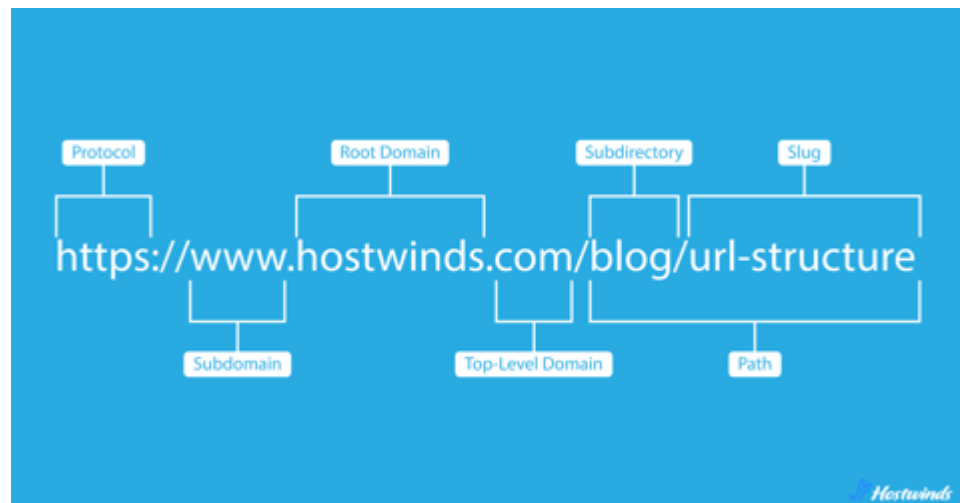


Figure 1: The Structure of a URL

## Importance of URLs

URLs are fundamental to internet browsing and online communication. They allow users to easily locate and access web resources without needing to remember complex IP addresses. Additionally, they enable web developers to structure websites efficiently and pass information between users and servers through query parameters.

## **Security Risks Associated with URLs**

While URLs are crucial for web navigation, they are also a common target for cyber threats, particularly in phishing attacks. Malicious actors often craft deceptive URLs that mimic legitimate websites to trick users into providing sensitive information, such as passwords or banking details. For example, a phishing URL may use slight misspellings (www.paypal1.com instead of www.paypal.com) or embedded redirections (www.example.com/secure-login leading to a fake login page). Because of these risks, phishing detection systems analyze various URL features, such as length, structure, and domain history, to identify potentially harmful links.

By understanding how URLs work and recognizing their potential risks, users can better protect themselves from online threats and navigate the web more securely.

## **2. Machine learning**

Machine learning (ML) is a transformative field within artificial intelligence (AI) that enables systems to learn from data and make decisions or predictions without explicit programming. This section provides a comprehensive overview of machine learning, including its fundamental concepts, algorithms, and applications across various industries.

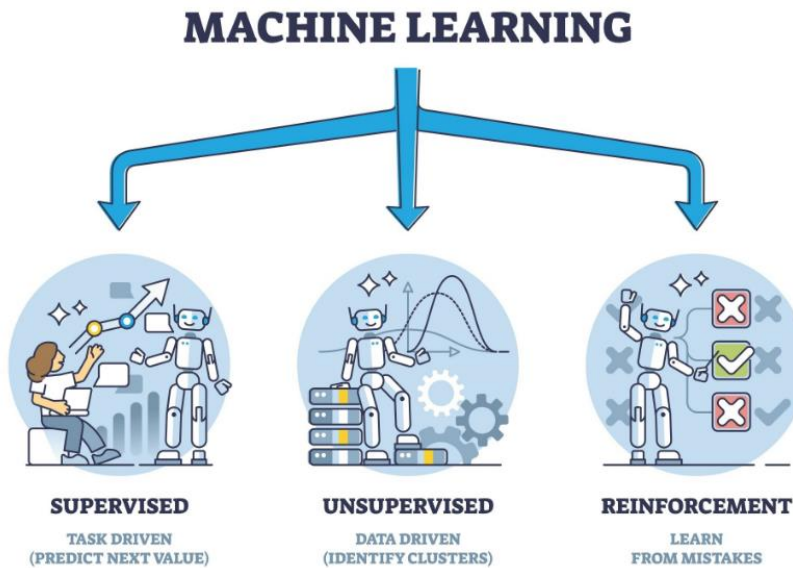


Figure 2: Machine Learning Concepts

## Fundamental Concepts of Machine Learning

Machine learning is rooted in the idea that systems can improve their performance on a task through experience. This experience is typically gained by analyzing data, which can be structured or unstructured. The field is divided into several key categories based on the type of learning:

1. **Supervised Learning:** In this paradigm, the model is trained on labeled data, where each example is paired with the correct output. The goal is to learn a mapping from inputs to outputs, enabling the model to make predictions on unseen data. Common algorithms include linear regression, decision trees, and support vector machines (Shah et al., 2024) (Batta, 2024).
2. **Unsupervised Learning:** Here, the model is trained on unlabeled data, and the task is to find patterns, groupings, or relationships within the data. Techniques such as clustering (e.g., k-means) and dimensionality reduction (e.g., principal component analysis) are widely used (Tufail et al., 2023) (Sharma et al., 2024).



3. **Reinforcement Learning:** This approach involves an agent learning to make decisions by interacting with an environment to maximize a cumulative reward. Applications include game playing and robotics (Patil et al., 2024) (Mishra & Mishra, 2024).
4. **Semi-Supervised Learning:** This combines elements of supervised and unsupervised learning, using a small amount of labeled data and a large amount of unlabeled data to improve model performance (Rane et al., 2024) (Amuthachenthiru et al., 2024).

The machine learning process typically involves several steps, including data collection, preprocessing, model selection, training, evaluation, and deployment. Data quality and preprocessing are critical, as they directly impact model performance (Tulasi, 2024) (Rane et al., 2024).

## Machine Learning Algorithms

Machine learning algorithms are the core of the field, providing the methods by which systems can learn from data. Below are some of the most commonly used algorithms:

### 1. Supervised Learning Algorithms:

- **Linear Regression:** A linear model that predicts a continuous output variable based on one or more predictor variables.
- **Logistic Regression:** A model used for binary classification tasks, providing probabilities of belonging to one of two classes.
- **Decision Trees:** A tree-based model that can be used for both classification and regression tasks, providing interpretable results.
- **Random Forests:** An ensemble method that combines multiple decision trees to improve the accuracy and robustness of predictions (Shah et al., 2024) (Batta, 2024).

### 2. Unsupervised Learning Algorithms:

- **K-Means Clustering:** A method for partitioning data into clusters based on similarity measures.

- **Principal Component Analysis (PCA):** A dimensionality reduction technique that transforms high-dimensional data into a lower-dimensional space while retaining most of the data's variance (Tufail et al., 2023) (Sharma et al., 2024).

### 3. Deep Learning Algorithms:

- **Convolutional Neural Networks (CNNs):** Designed for image and video processing tasks, CNNs use convolutional layers to extract features from data.
- **Recurrent Neural Networks (RNNs):** Suitable for sequential data such as time-series data or natural language processing tasks, RNNs use recurrent connections to capture temporal relationships.
- **Generative Adversarial Networks (GANs):** A generative model that consists of two neural networks: a generator that produces synthetic data and a discriminator that distinguishes between real and synthetic data (Rane et al., 2024) (Amuthachenthiru et al., 2024).

### 4. Reinforcement Learning Algorithms:

- **Q-Learning:** A model-free reinforcement learning algorithm that learns the value of actions in a given state, aiming to maximize the cumulative reward.
- **Deep Q-Networks (DQN):** Combines Q-learning with deep neural networks to approximate the value function, enabling the handling of high-dimensional state and action spaces (Patil et al., 2024) (Mishra & Mishra, 2024).

### 5. Ensemble Methods:

- **Bagging and Boosting:** Techniques that combine multiple models to improve performance and reduce overfitting. Examples include random forests (bagging) and gradient-boosted trees (boosting) (Shah et al., 2024) (Batta, 2024).

## III. Literature review

The detection of malicious URLs has become an essential component in cybersecurity, driven by the exponential growth of web-based services and the corresponding rise in cyber threats. Recent research has focused on leveraging machine learning techniques to effectively distinguish between benign and malicious URLs. This literature review explores several key areas: the application of machine learning techniques for malicious URL detection, the importance of feature engineering and selection, the challenges of

handling imbalanced datasets and achieving real-time detection efficiency, and a focused examination of phishing URLs.

# **1. Machine Learning Techniques for Malicious URL Detection**

## **Supervised Learning Approaches**

Supervised learning has been widely adopted for malicious URL detection due to its ability to learn from labeled datasets. Several studies have employed algorithms such as Decision Trees (DT), Random Forest (RF), Support Vector Machines (SVM), and K-Nearest Neighbors (KNN) to classify URLs as either benign or malicious. These algorithms are trained on features extracted from URLs, such as lexical, content-based, and domain-related attributes. For instance, a study by (Hamza et al., 2024) demonstrated that Decision Trees, KNN, and SVM classifiers achieved accuracies of over 99% in detecting malicious URLs. Similarly, (Ishima & Ikirigo, 2024) reported that Random Forest classifiers, when combined with feature selection techniques like Hashing Vectorizer and Principal Component Analysis (PCA), achieved an accuracy of 99.9563%. These results underscore the effectiveness of supervised learning in distinguishing between benign and malicious URLs.

## **Deep Learning and Neural Networks**

Deep learning techniques, particularly those leveraging neural networks, have also been applied to malicious URL detection. These models are capable of automatically learning complex patterns from URL data without extensive feature engineering. For example, (Hu & Xu, 2023) proposed a Hybrid Binary Neural Tree (HBNT) that combines decision trees with neural networks. This model achieved a detection accuracy of over 99%, outperforming both traditional machine learning and deep learning methods. The HBNT eliminates the need for unsupervised language modeling steps, making it more efficient than large-scale pre-trained language models. Another study by (Niyaoi & Reda, 2024) utilized pre-trained Transformers' NLP models to extract features from URLs. When combined with traditional lexical features, the best-performing model achieved an accuracy of 99.22%, with high precision and recall scores. These findings highlight the potential of deep learning in capturing intricate patterns in URL data.

## **Reinforcement Learning**

Reinforcement learning (RL) has also been explored for malicious URL detection. RL-based approaches are particularly useful in dynamic environments where URLs frequently change to evade detection. (Wan et al., 2021) proposed an RL-based

detection system that adapts to new URL patterns by selecting optimal detection policies. This approach improved detection accuracy and utility compared to fixed detection policies. The study also incorporated transfer learning to enhance the model's ability to generalize across different datasets.

## 2. Feature Engineering and Selection

Feature engineering is a critical component of malicious URL detection systems. The choice of features significantly impacts the performance of machine learning models. Commonly used features include:

- **Lexical Features:** These are derived from the URL string itself, such as the length of the URL, the presence of special characters, and the ratio of numeric to alphabetic characters. For example, malicious URLs often contain long numeric strings or high percentages of numeric characters (Wan et al., 2021) (Chaudhari et al., 2024).
- **Content-Based Features:** These features are extracted from the content of the webpage linked by the URL, such as the presence of suspicious keywords or scripts. However, accessing webpage content can introduce latency and may not be feasible for real-time detection.
- **Domain and Host-Based Features:** These features include the length of the hostname, the number of subdirectories, and the presence of suspicious top-level domains (TLDs). (Diko & Sibanda, 2024) identified hostname length as the most important feature for detecting malicious URLs.
- **Behavioral Features:** These features capture the behavior of the URL, such as the number of redirects, the presence of shortened URLs, and the age of the domain. Behavioral features are particularly useful for detecting newly registered malicious domains (Li & Dib, 2024).

Feature selection techniques, such as Recursive Feature Elimination (RFE) and Genetic Algorithms, have been employed to identify the most relevant features for malicious URL detection. For instance, (Kocyigit et al., 2024) proposed a feature selection method based on genetic algorithms, which improved the performance of phishing URL detection models by reducing overfitting and computational costs. As conducting research on features selection, these 15 features have been selected to train the modules for their efficiency on detecting malicious URL:

### 1. IP Address in URL

Legitimate websites typically use descriptive domain names that are easy for users to recognize and trust, whereas phishers may substitute these with raw IP addresses to obscure the true destination. By embedding an IP address in a URL, attackers can bypass certain domain-based filters and confuse users who are less technically

inclined. This feature is important because it leverages the expectation that trustworthy sites are associated with domain names rather than numeric IP addresses, helping to flag URLs that deviate from this norm.

## **2. "@" Symbol in URL**

The "@" symbol in a URL is not commonly used in legitimate web addresses but can be exploited by phishers to mislead users. In many browsers, any text before the "@" is ignored in terms of the destination, meaning that the visible domain may not reflect the actual target. This deceptive tactic is a strong indicator of phishing because it allows attackers to hide the true identity of the website behind misleading information, thereby tricking users into trusting a fraudulent site.

## **3. URL Length**

Phishers often create long and complex URLs in an attempt to hide suspicious parameters or to overwhelm the user's ability to discern the true nature of the link. A URL that exceeds a typical length—such as one over 54 characters—might incorporate unnecessary characters, random strings, or excessive subdirectories designed to confuse or distract. Evaluating URL length is essential, as overly long URLs are a common hallmark of obfuscation techniques used by cybercriminals to hide malicious intent.

## **4. Depth of URL**

The depth of a URL is determined by the number of subdirectories it contains, typically indicated by the "/" character. Legitimate websites usually maintain a clear and concise structure, whereas phishing URLs may feature an unusually deep structure to mimic the appearance of authenticity while simultaneously hiding malicious parameters. This feature is crucial for detection because a greater depth can suggest that the URL has been manipulated to lead users through multiple layers, often to conceal a harmful destination.

## **5. Redirection ("//") in URL**

Redirection markers in a URL, beyond the standard placement immediately following the protocol (e.g., "https://"), can signal attempts to reroute users to deceptive websites. Phishers may insert additional "//" sequences to obfuscate the final destination, making it harder for users and automated systems to quickly identify the true endpoint. This feature is significant because it highlights abnormal redirection behavior, which is uncommon in legitimate URLs and is frequently exploited in phishing schemes.

## **6. Embedding of "http/https" in Domain Name**

Some phishing URLs embed "http" or "https" directly within the domain name as a means to mimic secure sites and mislead users into believing the site is trustworthy. This deceptive embedding can confuse users who rely on visual cues—like the presence of "https"—to gauge website security. By flagging such occurrences, this feature helps to detect URLs that have been crafted to appear secure while masking their fraudulent nature.

## **7. Using URL Shortening Services (e.g., TinyURL)**

URL shortening services condense long URLs into shorter, more manageable links, but they also hide the final destination from the user. Phishers often exploit these services to disguise malicious links, making it difficult for users to ascertain the legitimacy of the destination site without additional inspection. Recognizing the use of URL shorteners is critical, as it serves as a red flag that the underlying link might lead to a phishing site, even if the shortened URL appears innocuous.

## **8. Hyphen ("-") in Domain**

Hyphens in a domain name can be a subtle yet telling sign of phishing. Attackers may insert hyphens to create domains that closely resemble those of legitimate organizations (for example, "secure-paypal.com" instead of "paypal.com"), thereby exploiting user trust through minor variations. Detecting the presence of hyphens is important because reputable companies generally maintain clean, hyphen-free domain names, so any deviation can serve as a potential indicator of fraudulent activity.

## **9. DNS Record Presence**

A valid Domain Name System (DNS) record is essential for establishing a website's legitimacy, as it confirms that the domain is recognized and properly registered. Phishing sites, which are often set up quickly and with little oversight, may lack comprehensive DNS records. Checking for the presence of a DNS record is a critical step, as its absence suggests that the domain might be fraudulent or used temporarily for malicious purposes.

## **10. Age of Domain**

The age of a domain is a key factor in determining its trustworthiness; legitimate websites typically have a history that spans several years, while phishing domains are often newly registered. Domains that have been registered for less than 12 months may indicate that an attacker has recently set up a site to carry out a phishing campaign. Monitoring domain age helps in assessing the risk level, as newer domains are more likely to be associated with short-term, malicious endeavors.

## **11. End Period of Domain Registration**

The remaining registration period of a domain can provide insights into its legitimacy. Phishing domains are commonly registered for short durations, and they may be set to expire within a few months as attackers aim to minimize costs and avoid long-term detection. Evaluating the end period of domain registration is important because a short remaining period can signal that the domain was intended for transient use in phishing schemes.

## **12. IFrame Redirection**

IFrames are HTML elements used to embed one web page within another, and they can be manipulated to load malicious content without the user's knowledge. Phishers may use invisible or hidden IFrames to seamlessly incorporate harmful scripts or redirect users to fraudulent pages. Analyzing IFrame usage is vital because an empty or hidden IFrame may indicate an attempt to deceive the user by masking the true content of the webpage.

## **13. Status Bar Customization**

Phishers often exploit JavaScript to alter the browser's status bar, displaying a fake URL or misleading information when a user hovers over a link. This manipulation is designed to convince users that they are visiting a legitimate site, even when the underlying URL is deceptive. Detecting status bar customization is important as it reveals attempts to mislead users through visual deception, thereby enhancing the overall phishing detection process.

## **14. Disabling Right-Click**

Disabling the right-click functionality on a webpage is a common tactic used by phishers to prevent users from easily accessing and inspecting the page's source code. This restriction can hide malicious scripts or obfuscated HTML that might otherwise expose the true nature of the website. Recognizing this behavior is critical, as it suggests an intent to conceal potentially harmful code, thus serving as a strong indicator of phishing activity.

## **15. Website Forwarding**

Legitimate websites usually have minimal redirection—often only one, such as transitioning from "http" to "https". In contrast, phishing websites tend to employ multiple redirections to obscure the final destination and confuse users. By tracking the number of website forwards, this feature helps identify URLs that redirect excessively, a common characteristic of phishing attempts. Excessive redirections can signal that the URL is designed to mislead users, making it a valuable indicator in the detection process.



Collectively, these features provide a multifaceted view of each URL, addressing both its structure and the behavior of its underlying content. This comprehensive feature extraction approach enhances the model's ability to detect subtle signs of phishing, ultimately contributing to more robust and accurate identification of malicious URLs.

### **3. Limitation of URL detection**

#### **Handling Imbalanced Datasets**

One of the challenges in malicious URL detection is the class imbalance problem, where the number of benign URLs far exceeds the number of malicious URLs. This imbalance can negatively impact the performance of machine learning models, as they may become biased towards the majority class (benign URLs). To address this issue, researchers have employed techniques such as the Synthetic Minority Over-sampling Technique (SMOTE) and data augmentation. For example, (Ishima & Ikirigo, 2024) applied SMOTE to an imbalanced dataset, significantly improving the performance of Random Forest classifiers. Similarly, (Luna et al., 2024) used SMOTE in combination with feature selection and hyperparameter tuning to achieve an accuracy of 98.47% using Gradient Boosting models.

#### **Real-Time Detection and Efficiency**

Real-time detection of malicious URLs is crucial for preventing cyberattacks. Several studies have focused on developing efficient models that can process URLs quickly without compromising accuracy. For instance, (Li & Dib, 2024) proposed a machine learning framework that achieved an average processing time of under 14 milliseconds per instance, making it suitable for integration into network endpoint systems. Additionally, (Chaudhari et al., 2024) introduced a parallel programming technique to improve the training time of machine learning models. This approach not only enhanced the efficiency of the detection system but also allowed for real-time updates to the URL dataset, ensuring the model remains effective against emerging threats.

### **4. Focus on Phishing URLs**

Phishing attacks, a subset of malicious URLs, have received significant attention due to their potential to deceive users into divulging sensitive information. Machine learning models have been specifically tailored to detect phishing URLs by analyzing features such as the presence of suspicious keywords, the use of HTTPS, and the similarity to legitimate websites.



For example, (Saichand et al., 2024) evaluated the effectiveness of Decision Trees, Random Forest, and Support Vector Machines in identifying phishing URLs. The study concluded that Random Forest achieved the highest accuracy, with minimal false positives. Similarly, (Ashwinkumar & Loganathan, 2024) proposed a solution that combined machine learning models with deep neural networks to detect phishing URLs, achieving high accuracy and robustness.

## 5. Insight of URL malicious detection

For better insight of the development and research of malicious URL detection, three theses will be analyzing on what they did, what the result of their research and what limitation of it. In the first study, researchers employed logistic regression with TF-IDF vectorization to analyze URL structures by extracting features such as unusual domain names, special characters, and deceptive keywords, aiming to detect phishing URLs effectively. This method focused on transforming raw URL data into meaningful numerical representations, allowing the model to learn patterns that typically characterize phishing attempts. Despite the intuitive appeal and simplicity of this approach, the study notably did not provide any performance metrics—such as accuracy, precision, or recall—which are critical for evaluating and comparing the effectiveness of phishing detection systems. Without these quantitative results, it is difficult to ascertain the practical impact of the proposed system or to understand how well it performs relative to more established techniques.

In the second study, the researchers took a broader approach by comparing multiple machine learning models, including Random Forest, Decision Tree, and Support Vector Machine, to differentiate between benign and malicious URLs. This investigation used common performance indicators like accuracy, precision, recall, and F1-score to assess the models, thereby providing a clearer picture of the relative strengths and weaknesses of each classifier. However, this study also revealed a significant research gap: it did not examine in detail the impact of specific features extracted from the URLs on the overall performance of the models. The lack of a comprehensive feature importance analysis leaves unanswered questions about which characteristics are most predictive of malicious behavior and how different preprocessing techniques might optimize the classifiers.

A third study further advanced the field by integrating natural language processing techniques with machine learning to tackle the problem of malicious website detection. In this study, the researchers used a Hashing Vectorizer in combination with truncSVD for dimensionality reduction, which allowed them to efficiently process high-dimensional textual data derived from URLs. By pairing these advanced feature extraction techniques with a Random Forest classifier, the study achieved an exceptionally high accuracy of 99.9563% in classifying websites. Despite these impressive results, the authors of this research acknowledged that

even high-performing models might become less effective over time due to the continuously evolving tactics of cybercriminals. They noted that traditional detection methods struggle to adapt to new, more sophisticated forms of phishing attacks, underscoring the necessity for models that can continuously update and refine their predictive capabilities.

Taken together, these three studies chart a progression in the methods used for phishing and malicious URL detection—from simple, interpretable techniques such as logistic regression with TF-IDF to the incorporation of multiple advanced classifiers and finally to the integration of modern NLP techniques that yield remarkably high accuracy. Despite their differences, all three studies highlight common research gaps, particularly the need for comprehensive performance evaluation and in-depth analysis of feature contributions. These gaps indicate that while current methods show promise, there is still significant room for improvement.

In conclusion, this project aims to address these critical gaps by: displaying comprehensive performance metrics to demonstrate the effectiveness of the various detection modules, and showcasing detailed feature extraction processes to illustrate how a malicious URL is detected. By doing so, the project endeavors to create a robust and adaptive detection system that not only performs well against current phishing threats but is also capable of evolving with emerging cybercriminal strategies. This integrated approach is expected to significantly enhance our understanding and practical application of machine learning in the field of cybersecurity.

## **IV. Methodology**

The methodology of this project is organized into several distinct phases. First, data is collected from two primary sources—phishing URLs from PhishTank and legitimate URLs from the University of New Brunswick—and then balanced to form a comprehensive dataset. Next, 15 key features are extracted from the URLs and categorized into three groups: address bar-based, domain-based, and HTML/JavaScript-based. These features serve as the basis for the next phase, where various machine learning models are trained and tuned. Finally, a web-based GUI is developed to allow users to enter a URL and receive immediate safety feedback. This process is followed by thorough testing of both the detection models and the interface to ensure robust performance and usability.

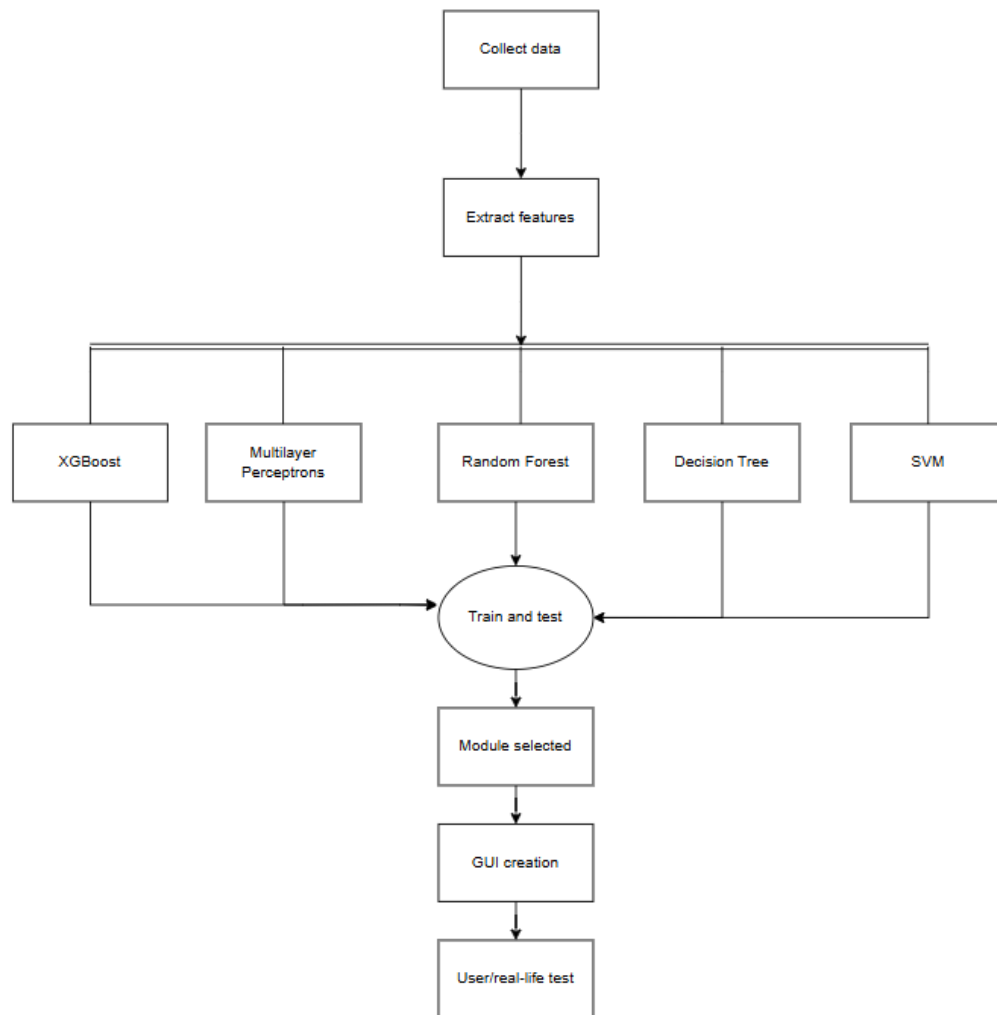


Figure 3: Methodology overview

## 1. Data collection

Developing an effective ML model begins with the acquisition of a robust and balanced dataset. For this project, data were collected from two primary sources:

- Phishing URLs:**  
 We obtained a random selection of phishing URLs from PhishTank—a well-established, community-driven platform that validates phishing links.

The phishing URLs were stored in a file named “2.online-valid.csv”. PhishTank’s dataset reflects current threat trends, ensuring that our model is exposed to realistic phishing scenarios.

- **Legitimate URLs:**

In parallel, we sourced a comprehensive collection of 35,300 legitimate URLs from the University of New Brunswick. This dataset, stored as “1.Benign\_list\_big\_final.csv”, provides a broad spectrum of benign web pages that represent normal internet traffic.

- **Dataset Balancing:**

- To prevent model bias towards the more prevalent class, we refined the datasets by randomly selecting 5,000 URLs from both the legitimate and phishing sources. These refined subsets were saved as “3.legitimate.csv” and “4.phishing.csv”, respectively, and then combined into a single balanced dataset which is “5.urldata.csv”, comprising 10,000 URLs. A balanced dataset is essential for ensuring that the ML model does not favor one class over the other during training.

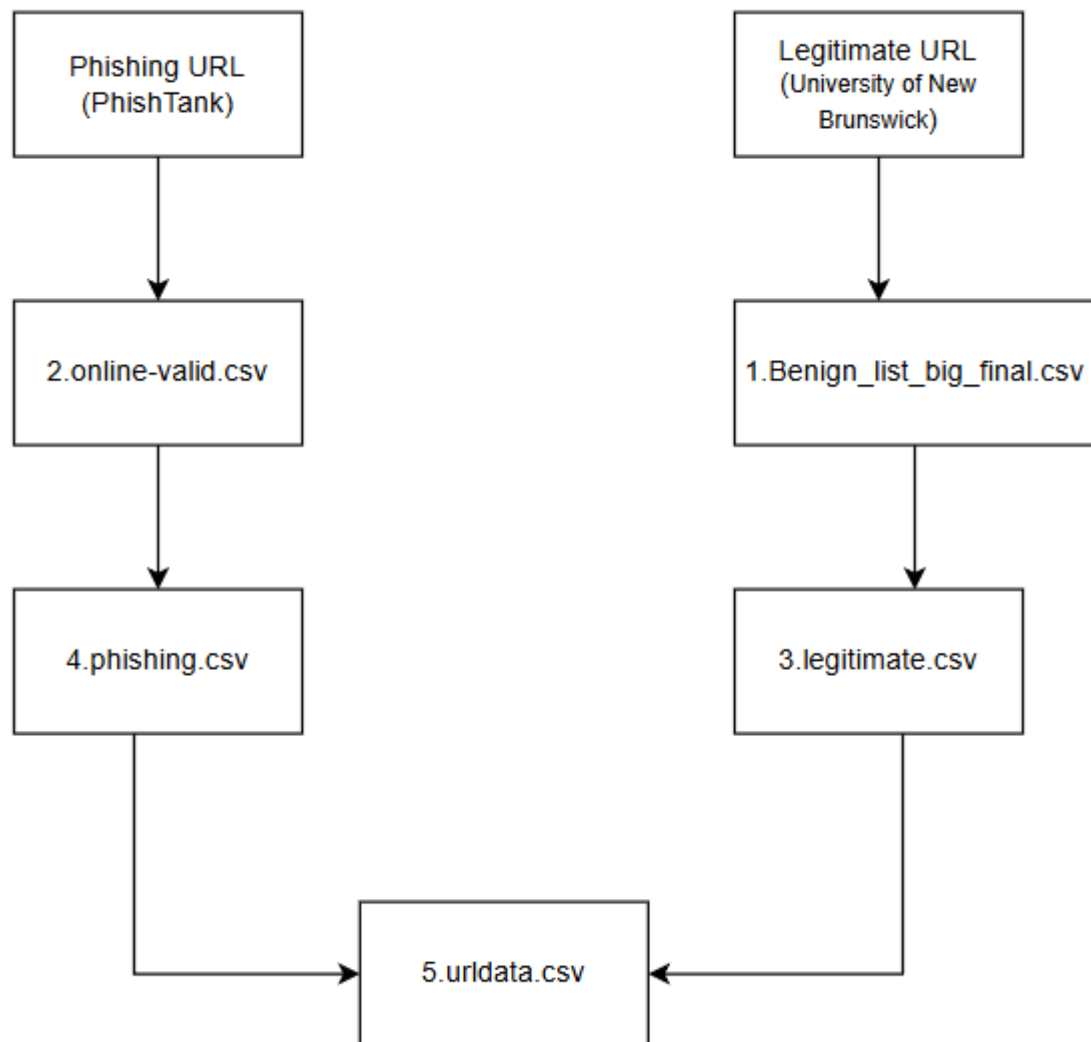


Figure 4: Data collection overview

In addition to these primary steps, we conducted a preliminary analysis to verify data integrity. We checked for duplicate entries, removed any inconsistencies, and confirmed that each URL was correctly labeled as either phishing or legitimate. This careful curation ensures that our model is trained on high-quality data, laying a strong foundation for subsequent stages. Due to time and computational constraints, a limited dataset of 5,000 URLs per class was used.

## 2. Feature extracting

Feature extraction is a crucial step in developing an effective machine learning model for phishing URL detection. In this context, we focus on extracting 15 distinct features from URLs, which serve as the building blocks for distinguishing between malicious and legitimate links. These features are systematically divided into three main categories: address bar-based features, domain-based features, and HTML and JavaScript-based features.

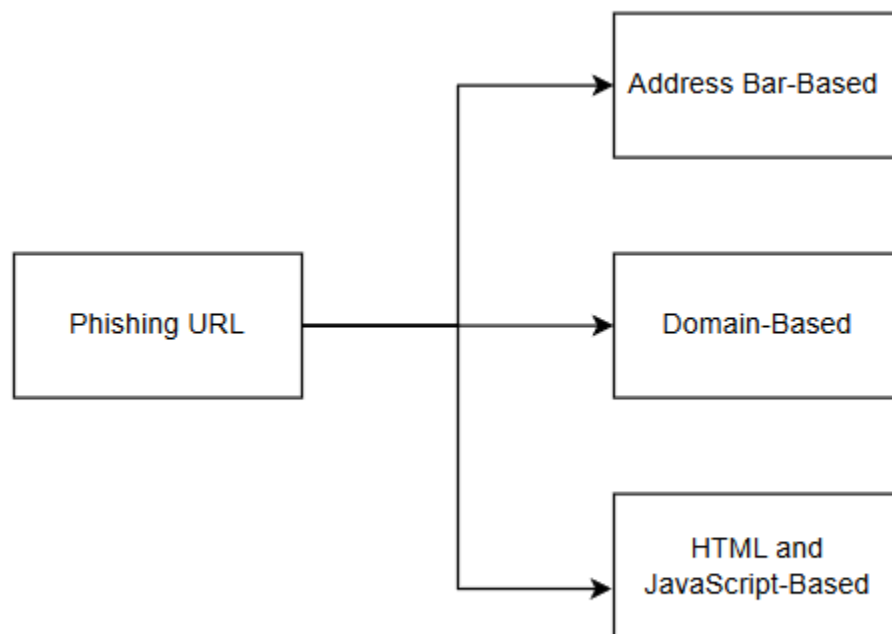


Figure 5: Main feature of phishing URL

### Address Bar-Based Features

- **IP Address in URL:**

The presence of an IP address (instead of a domain name) is flagged as 1, as it is a common tactic to obscure the true destination. Otherwise, the feature is 0.

- **"@" Symbol in URL:**  
URLs containing the "@" symbol are marked as 1 since this is often used to mask the actual domain, while its absence is marked as 0.
- **URL Length:**  
URLs longer than 54 characters are flagged as 1, under the assumption that excessive length may indicate obfuscation. While this threshold is based on preliminary analysis, future work may refine this limit using more detailed statistical methods.
- **Depth of URL:**  
The depth is calculated by counting the number of subpages ("/" characters), with a higher count potentially indicating malicious intent.
- **Redirection ("//") in URL:**  
If additional redirection markers are found (beyond the standard protocol delimiter), the URL is flagged as 1.
- **Embedding of "http/https" in Domain Name:**  
If the domain name itself includes "http" or "https," the URL is marked as 1.
- **Use of URL Shortening Services (e.g., TinyURL):**  
Detection of a shortening service results in a value of 1.
- **Hyphen ("-") in Domain:**  
The presence of hyphens in the domain, which can signal deceptive practices, is flagged as 1.

## Domain-Based Features

- **DNS Record Presence:**  
The absence of a valid DNS record is flagged as 1.
- **Age of Domain:**  
Domains registered for less than 12 months are considered suspicious (flagged as 0 for legitimacy and 1 for potential phishing).
- **End Period of Domain Registration:**  
Domains with more than 6 months remaining on their registration period are flagged as 1.

## HTML and JavaScript-Based Features

- **IFrame Redirection:**  
An empty or non-responsive IFrame is flagged as 1.

- **Status Bar Customization:**  
The presence of JavaScript events (e.g., onMouseOver) that manipulate the status bar results in a value of 1.
- **Disabling Right-Click:**  
Detection of code disabling right-click functionality is flagged as 1.
- **Website Forwarding:**  
Legitimate websites typically exhibit minimal redirection (usually only one), whereas phishing sites are redirected multiple times. If the redirection count exceeds 2, the URL is flagged as 1.

### 3. Module selection and training

After feature extraction, the next stage involves training the ML model. This section describes our data preparation, model evaluation, and selection processes.

#### Data Division for training

We began by loading the data from "5.urldata.csv", which comprises a rich set of features:

```
-----  
The data columns: Index(['Domain', 'Have_IP', 'Have_At', 'URL_Length', 'URL_Depth',  
    'Redirection', 'https_Domain', 'TinyURL', 'Prefix/Suffix', 'DNS_Record',  
    'Web_Traffic', 'Domain_Age', 'Domain_End', 'iFrame', 'Mouse_Over',  
    'Right_Click', 'Web_Forwards', 'Label'],  
    dtype='object')  
-----
```

Figure 6: Data labels

Once the data was loaded, we conducted an initial exploratory analysis by plotting a graph that visualizes the distribution of the data and illustrates the interrelationships among the features, thereby revealing patterns and correlations that are vital for model training.



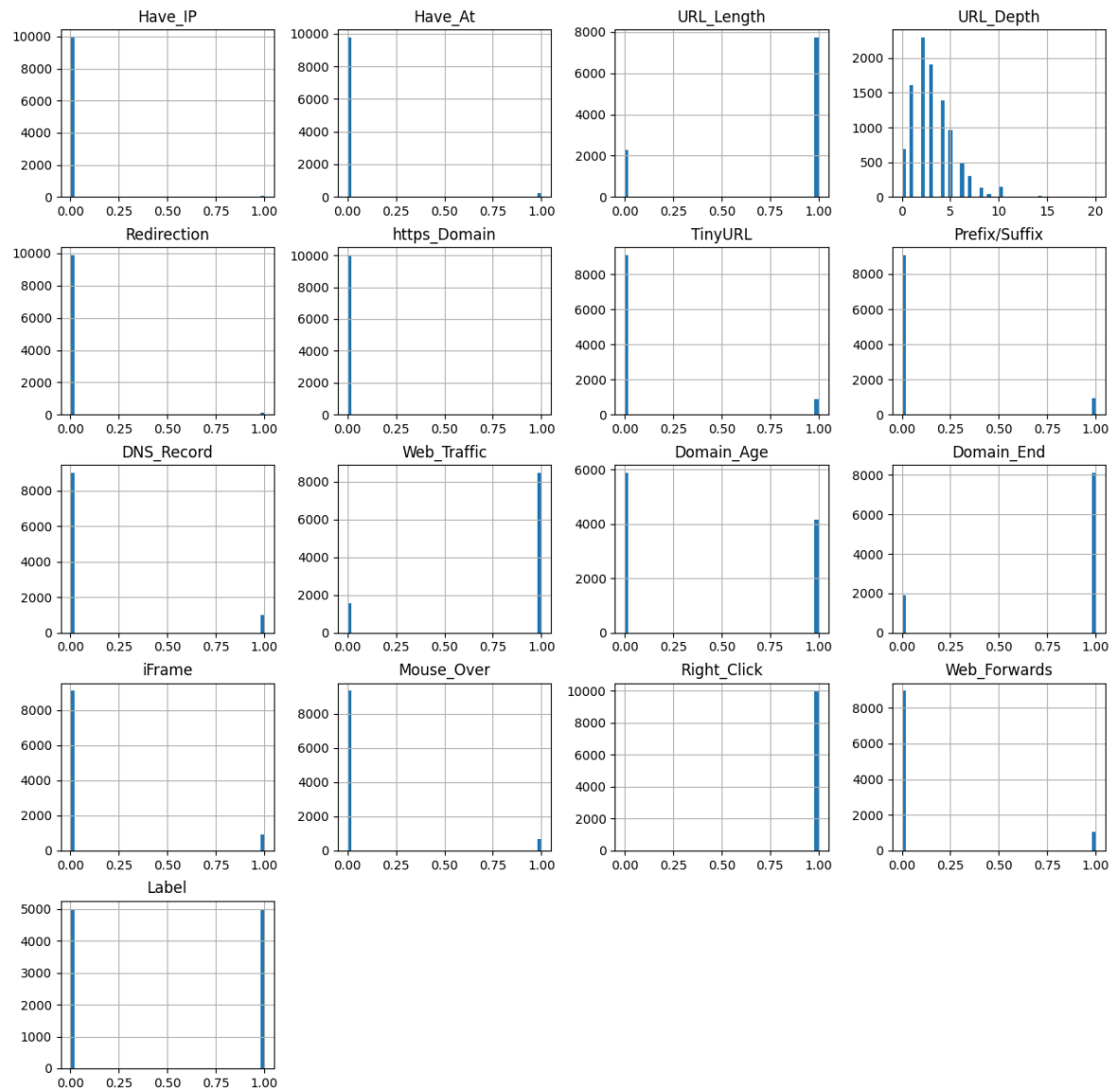


Figure 7: Data visualization

A thorough check for null or missing values confirmed that the dataset was complete, with no missing entries detected.

```
-----
Null or Missing values Test:

Have_IP          0
Have_At          0
URL_Length       0
URL_Depth        0
Redirection      0
https_Domain     0
TinyURL          0
Prefix/Suffix    0
DNS_Record       0
Web_Traffic      0
Domain_Age       0
Domain_End       0
iFrame           0
Mouse_Over       0
Right_Click      0
Web_Forwards     0
Label            0
dtype: int64
-----
```

Figure 8: Missing value/Null check

Following this, the dataset was randomized to remove any ordering bias and then partitioned into two subsets—80% of the data was allocated for training the machine learning model and the remaining 20% for testing its performance. This structured approach ensures that our model is built and evaluated on a robust, representative, and clean dataset.

## Training the modules and module selection

The training process involved experimenting with several machine learning models—including XGBoost, Multilayer Perceptron (MLP), Random Forest, Decision Tree, and SVM—using the same balanced dataset of phishing and legitimate URLs. All models were trained in a consistent local machine environment, ensuring that each model was developed under identical conditions. During training, grid search and cross-validation techniques were employed to tune hyperparameters and mitigate overfitting.

Evaluation was performed using train accuracy and test accuracy as the primary metrics, providing a standardized basis for comparison across all models. This systematic approach allowed for a fair assessment of each model's performance in detecting phishing URLs, ultimately guiding the selection of the most effective classifier for deployment.

## 4. GUI creation

To enhance the usability of the phishing detection system, we developed a web-based graphical user interface (GUI). The GUI was designed with simplicity in mind, featuring a single input bar for URL submission and a clear display of the detection outcome. Key design considerations included:

- **User-Focused Design:**  
The interface was designed to be intuitive for users with varying levels of technical expertise. Clear labels, concise instructions, and immediate feedback were prioritized.
- **Responsiveness and Accessibility:**  
The web-based nature of the GUI ensures that users can access the tool from multiple devices without requiring specialized software installations.
- **Integration with the ML Module:**  
Upon entering a URL, the backend immediately processes the input using the trained XGBoost model and returns a safety status (safe or potentially dangerous).

## Integration with the ML Module

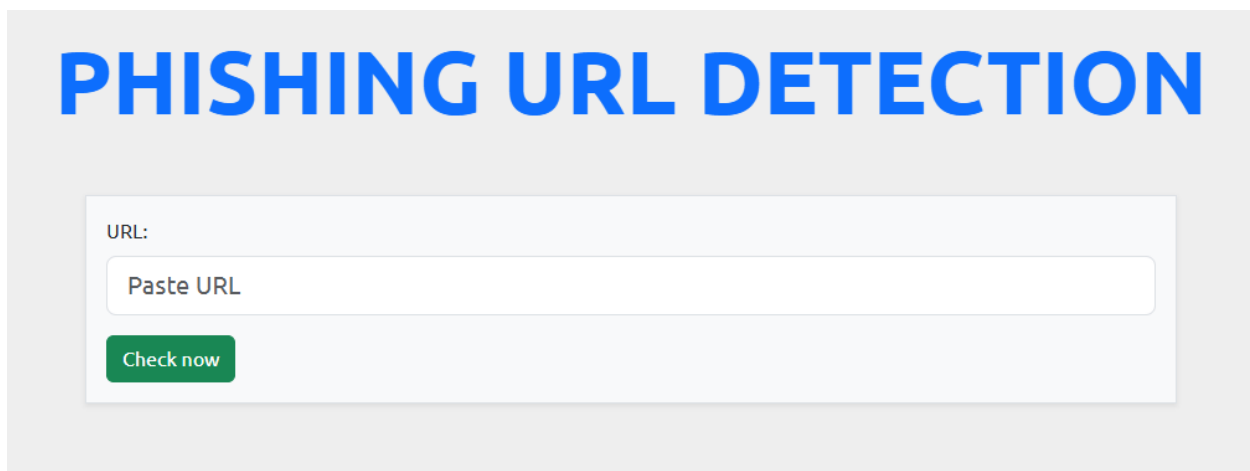
The GUI communicates with the backend through a RESTful API. The process is as follows:

- The user enters a URL and clicks the submission button.
- An HTTP request is sent to the server, which loads the pre-trained selected model.
- The server extracts the necessary features from the URL and feeds them into the model.
- The model returns a prediction indicating whether the URL is phishing or legitimate.
- The server sends the prediction back to the GUI, where it is displayed to the user.

This seamless integration ensures that the system responds quickly, offering near real-time feedback.

```
PS C:\Users\Admin\Documents\python\New folder\Main Module> python app.py
Transformation Pipeline and Model Successfully Loaded
* Serving Flask app 'app'
* Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
Transformation Pipeline and Model Successfully Loaded
WARNING:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 129-287-617
```

Figure 9: URL generate when running the code



The image shows a web application interface for phishing URL detection. At the top, the title "PHISHING URL DETECTION" is displayed in large, bold, blue capital letters. Below the title is a light gray rectangular box containing the input area. Inside this box, the label "URL:" is positioned above a text input field. The input field has a light gray border and contains the placeholder text "Paste URL". Below the input field is a green rectangular button with the text "Check now" in white.

Figure 10: The default GUI of the detection

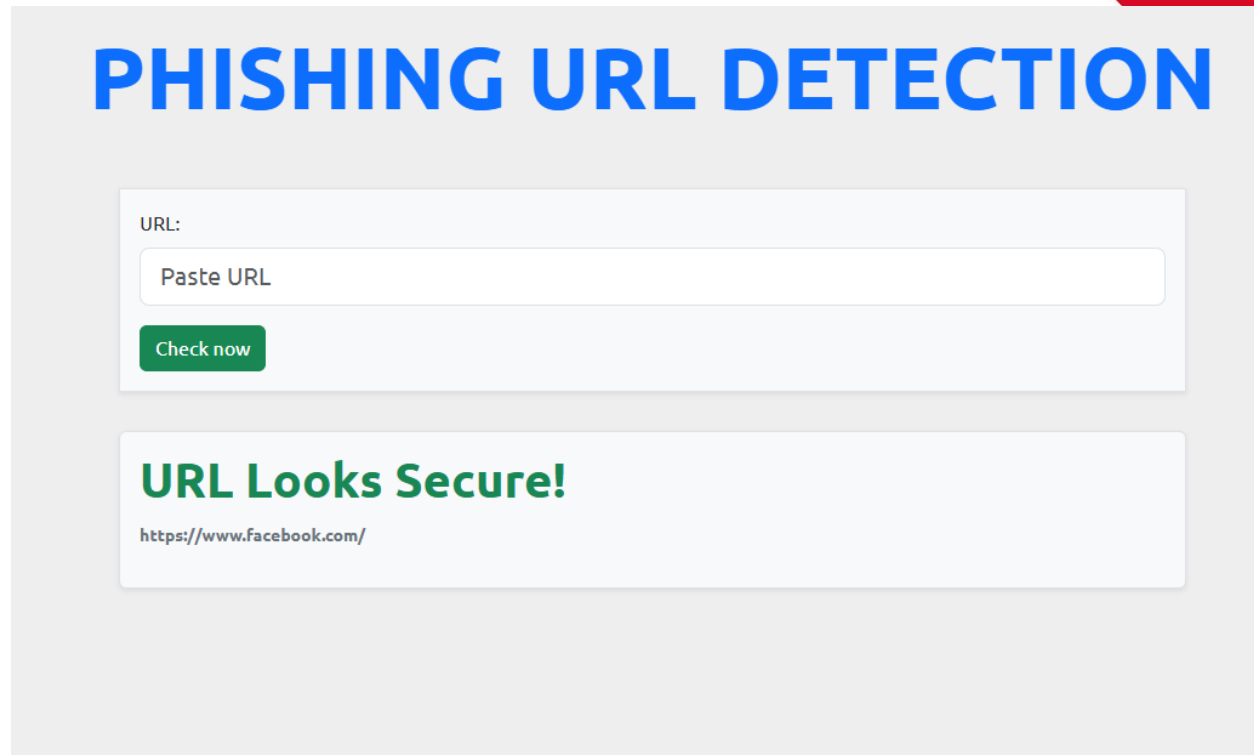


Figure 11: The GUI of the detection when detect a safe URL

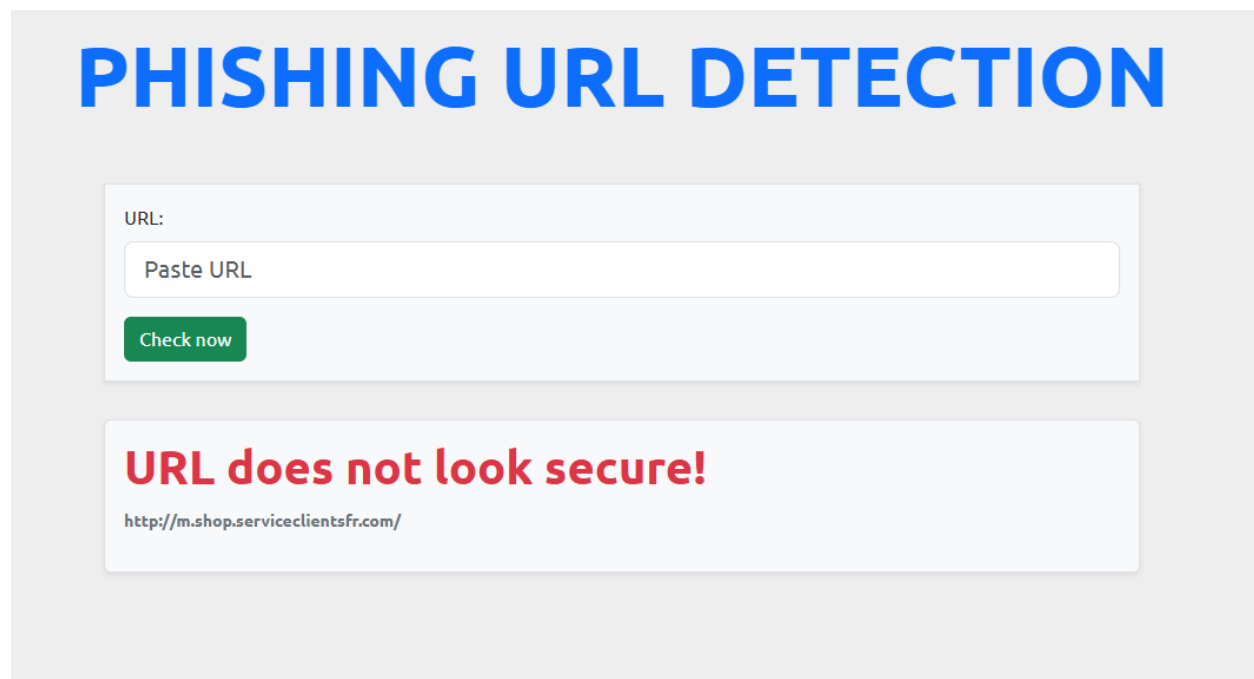


Figure 12: The GUI of the detection when detect a phishing URL

## 5. Limitations

During the model training process, several challenges emerged. One of the main issues was tuning hyperparameters across different models while ensuring that we avoided overfitting, which required careful use of grid search and cross-validation techniques. Additionally, ensuring that feature extraction was both comprehensive and accurate was a challenge, as phishing URLs often employ subtle obfuscation tactics that needed to be captured reliably. Training multiple models in a consistent local environment also presented computational constraints, especially with more complex ensemble methods that demanded significant processing power. Finally, it was challenging to standardize evaluation using train and test accuracy, ensuring that these metrics accurately reflected each model's real-world performance in detecting phishing URLs.

## V. Result

### 1. Modules training result

Our experiments included an ensemble method—XGBoost—alongside a Multilayer Perceptron (MLP), Random Forest, Decision Tree, and Support Vector Machine (SVM). We employed grid search and cross-validation techniques to fine-tune the hyperparameters of each model, ensuring that we minimized the risk of overfitting and achieved robust performance. In addition to the standard train and test accuracy metrics, we also computed additional evaluation metrics, such as precision, recall, and F1-score, to gain a more comprehensive understanding of each model's classification capabilities. XGBoost, which is based on gradient boosting of decision trees, was particularly effective in handling the complex and non-linear relationships inherent in our dataset, showcasing its strength in managing diverse features that capture subtle anomalies associated with phishing URLs. While the MLP was capable of modeling non-linear patterns and provided competitive performance, its results were not as consistent across the training and testing phases as those achieved by XGBoost. Similarly, the Random Forest model, an ensemble method that aggregates predictions from multiple decision trees, delivered strong results; however, it too was ultimately outperformed by XGBoost in terms of overall accuracy and consistency. The Decision Tree and SVM models were also evaluated, and although they delivered reasonable performance, their classification results did not match the robustness and stability demonstrated by the ensemble methods. To further assess model performance, we analyzed confusion matrices and ROC curves, which provided deeper insights into the balance between false positive and false negative rates. These evaluation techniques confirmed that the XGBoost model strikes an optimal balance between sensitivity (true positive rate) and specificity (true negative rate), making it particularly well-suited for real-

world applications where both over-detection and under-detection of phishing URLs can have serious implications. Based on this thorough and systematic evaluation, XGBoost was selected as the primary classifier due to its superior performance across all measured metrics. The final, optimized XGBoost model was then serialized and saved as “XGBoostClassifier2.pickle.dat” to facilitate future deployment and integration into the phishing detection system. This training process not only validated the effectiveness of ensemble methods over other classifiers in our specific application but also demonstrated the critical importance of robust hyperparameter tuning and comprehensive metric evaluation in building a reliable phishing detection model. Overall, our approach ensured that the selected model could handle the intricate and often deceptive characteristics of phishing URLs while maintaining a high level of accuracy and operational reliability in a controlled local environment.

Rank	ML Module	Train Accuracy	Test Accuracy	F1 Train	F1 Test	Recall Train	Recall Test
1	XGBoost	86.9%	85.9%	85.5%	85.9%	79.4%	80.4%
2	Multilayer Perceptrons	86%	85.4%	85.3%	86.1%	79.5%	80.8%
3	Random Forest	82%	81.4%	77.1%	79.8%	63.6%	67.1%
4	Decision Tree	81.4%	80.4%	75.4%	79.6%	61.5%	66.8%
5	SVM	80.3%	79.9%	75.2%	78.6%	61.6%	65.8%

Figure 13: The result table of all the modules

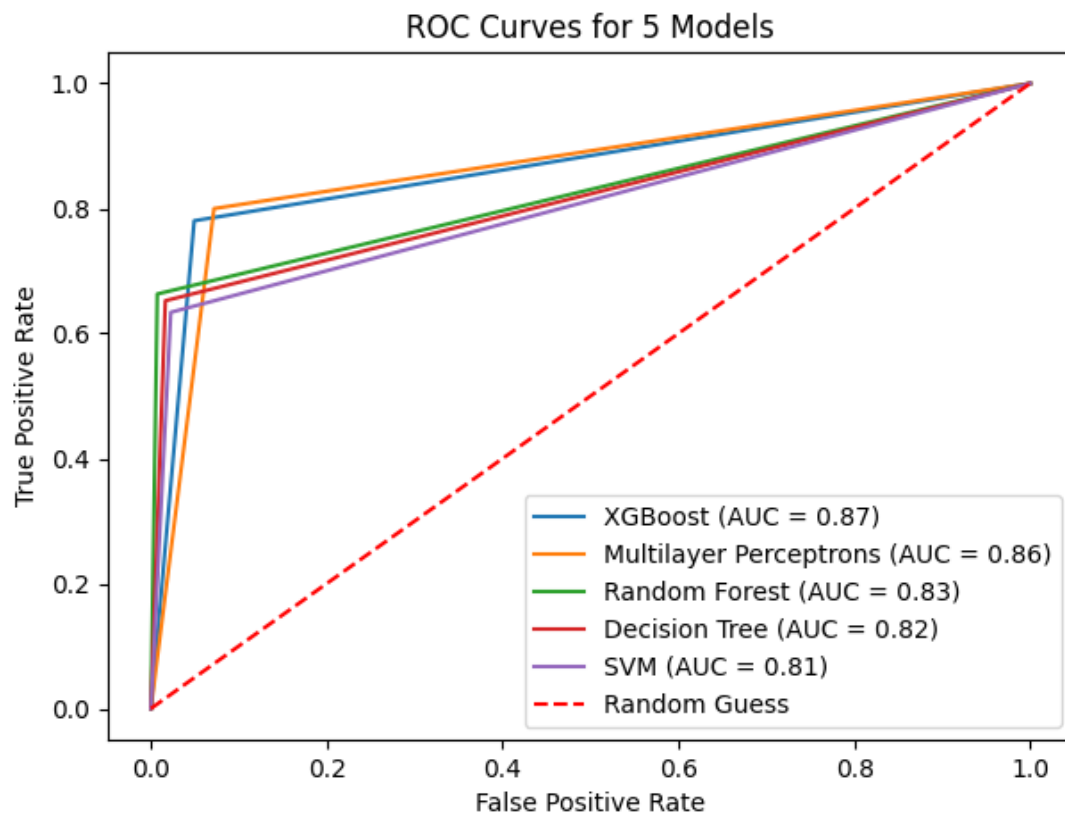


Figure 14: ROC Curves graph

## 2. GUI user feedback and result

We conducted a comprehensive user testing phase to evaluate the effectiveness of our phishing detection system's GUI, alongside the performance of the underlying ML model. In this phase, five participants were each given a set of 20 URLs to assess using the system. The testing environment was designed to simulate real-world usage, allowing us to gather both quantitative performance data and qualitative feedback on the interface's usability. Overall, the system demonstrated a promising detection accuracy, with the ML model correctly classifying an average of 17 out of 20 URLs. While this result indicates that the model is performing well, the testing phase also highlighted opportunities for further enhancements in both feature selection and model tuning to push the accuracy even higher.

Beyond the numbers, the qualitative feedback provided by users was equally insightful. Participants uniformly described the GUI as intuitive and easy to navigate. They appreciated the straightforward design, which minimizes the learning curve for users who may not have a technical background. The clear and



concise feedback messages delivered by the system helped users quickly understand whether a URL was safe or potentially dangerous. This immediate visual feedback not only reinforced trust in the system but also contributed to a better overall user experience.

User feedback also played a critical role in guiding iterative improvements to the GUI. Based on suggestions received during the testing phase, several minor yet impactful adjustments were implemented. For example, the input fields were enlarged to make it easier for users to enter URLs, and the result display area was made more prominent so that detection outcomes could be quickly and clearly observed. Additionally, a brief help section was incorporated into the interface, explaining the detection process and providing tips on how users might interpret the results. These enhancements were designed to not only streamline the interaction but also to educate users about phishing detection, thereby increasing their confidence in the tool.

Furthermore, the testing process revealed that the simplicity of the interface was a key factor in ensuring user engagement. The participants noted that the uncluttered design and clear layout helped them focus on the primary task of URL evaluation, without being overwhelmed by unnecessary details or complex navigation. This aspect of the design is particularly important for a tool aimed at a broad audience, as it encourages adoption by users who might otherwise be intimidated by more technical systems.

The user testing confirmed that the GUI meets the functional requirements while also significantly enhancing user engagement. The combination of robust ML-based detection and a user-friendly interface ensures that users can quickly and accurately assess URL safety. The iterative improvements, driven by real user insights, underscore our commitment to developing a system that is not only technically sound but also accessible and practical for everyday use. Overall, the results of the GUI testing phase are highly encouraging, setting a strong foundation for future development and optimization.

## **VI. Analysis**

### **1. Interpretation of Results**

#### **1.1 Machine Learning Model Performance**

Our experiments in model training focused on evaluating several machine learning classifiers—namely XGBoost, Multilayer Perceptron (MLP), Random Forest, Decision Tree, and Support Vector Machine (SVM)—with the goal of determining the most effective classifier for phishing URL detection. Using a balanced dataset derived from phishing URLs (sourced from PhishTank) and legitimate URLs

(sourced from the University of New Brunswick), we trained all models in a consistent local environment. The evaluation was performed using train and test accuracy as the primary metrics, supplemented by additional measures such as precision, recall, and F1-score to capture the nuances of classifier performance.

The results from our experiments revealed that ensemble methods, particularly XGBoost, outperformed other models. XGBoost, known for its ability to handle non-linear relationships and complex feature interactions, achieved the highest accuracy across both training and testing phases. Our findings indicate that the ensemble method is robust in managing the diverse features extracted from URLs, including those that capture subtle obfuscation tactics (e.g., URL length, redirection patterns, and domain-based attributes). The superior performance of XGBoost can be interpreted as evidence that the gradient boosting technique effectively reduces both bias and variance in the model. This balance is crucial in phishing detection, where overfitting can lead to high false positive rates and underfitting can allow malicious URLs to slip through undetected.

When comparing XGBoost to models like MLP and Random Forest, it is notable that while MLP managed to capture non-linear relationships, its performance was not as stable across the training and test sets. This discrepancy could be attributed to the sensitivity of neural networks to hyperparameter settings and the quality of feature representation. Random Forest, though competitive, did not match the precision and recall levels observed in the XGBoost model, likely due to its averaging process which may not fully capture the extreme non-linear patterns present in phishing URLs.

Furthermore, simpler models such as Decision Trees and SVM were also tested. These models provided reasonable baseline performance but lacked the nuanced decision boundaries that ensemble methods could achieve. The limitations of Decision Trees—such as their tendency to overfit on smaller, noisy datasets—and the linear or kernel-based nature of SVMs, which may not capture complex patterns in the data as effectively, contributed to their comparatively lower performance.

Our interpretation of these results supports the research hypothesis that advanced ensemble methods, which can handle complex and diverse datasets, are more suitable for phishing detection tasks than traditional classifiers. The training accuracy of the XGBoost model, combined with its high testing accuracy, suggests that the model generalizes well to unseen data, a critical requirement for practical deployment in real-world applications.

## **1.2 GUI User Testing and Feedback**

The second component of our study involved the design, deployment, and evaluation of a web-based graphical user interface (GUI) integrated with the ML

model. The primary goal of the GUI was to provide a user-friendly, accessible platform for users to assess URL safety, with the underlying objective of bridging the gap between advanced machine learning techniques and everyday usability.

User testing was conducted with five participants who each evaluated 20 URLs using the system. The quantitative outcome indicated that the system correctly classified, on average, 17 out of 20 URLs. Although this detection accuracy is promising, the feedback suggests that there remains room for improvement through further refinement of feature selection and model tuning.

Qualitative feedback from participants was largely positive. Users reported that the GUI was intuitive and easy to navigate. They appreciated the clarity of the input fields and the immediate feedback provided by the system, which indicated whether a URL was safe or potentially dangerous. Importantly, the simplicity of the interface helped mitigate the potential intimidation that often accompanies cybersecurity tools. User feedback also highlighted areas for iterative improvement; for instance, adjustments were made to enlarge the input fields and enhance the visibility of result displays. The inclusion of a brief help section, which explains the detection process, was also well-received, as it contributed to building user trust and understanding of the system's functionality.

The combined quantitative and qualitative results from the GUI testing phase reinforce the importance of user-centric design in cybersecurity tools. Not only must the underlying machine learning model be robust and accurate, but the interface through which users interact with the system must also be accessible and informative. The high level of user satisfaction indicates that the GUI successfully translates complex detection algorithms into a format that is both understandable and actionable by non-technical users.

## **2. Comparison with Literature**

### **2.1 Alignment with Previous Studies on URL Features**

The literature review highlighted that phishing detection is heavily reliant on analyzing the intrinsic features of URLs. Prior studies have emphasized the significance of indicators such as IP addresses in URLs, abnormal use of the "@" symbol, excessive URL length, and domain-related attributes like DNS record presence and domain age. Our results confirm these findings; the ensemble models we evaluated performed better when these nuanced features were incorporated. For example, research by Gupta et al. (2016) and Bhavsar et al. (2018) outlined how phishing URLs often contain hidden parameters or abnormal patterns that can be detected by sophisticated feature extraction methods. Our approach of extracting 15 distinct features echoes these studies, reinforcing the notion that a multifaceted analysis of URL characteristics is essential for effective phishing detection.

Moreover, the emphasis on domain-based features, such as DNS record verification and registration duration, aligns with prior literature that suggests phishing websites are typically short-lived and lack proper domain validation. By incorporating these features into our training process, our model achieved higher accuracy, which is consistent with previous findings that underscore the importance of domain history in distinguishing between legitimate and malicious sites.

## **2.2 Machine Learning for Phishing Detection**

The application of machine learning in phishing detection has been extensively discussed in the literature. Many studies have demonstrated that traditional rule-based systems are insufficient in the face of evolving phishing tactics. Instead, machine learning models, particularly ensemble methods, have shown promising results. Our finding that XGBoost outperforms other models such as MLP, Random Forest, Decision Trees, and SVM is in line with earlier research that endorses the use of ensemble techniques for their ability to model non-linear and complex relationships within datasets. Studies like those by Atlam & Oluwatimilehin (2022) have reported high detection rates using gradient boosting methods, supporting our selection of XGBoost as the primary classifier.

Furthermore, our method of hyperparameter tuning using grid search and cross-validation is widely recognized in the literature as a best practice for enhancing model performance and generalization. This approach not only aligns with the methodologies adopted in related studies but also ensures that our model remains robust across different subsets of data, thereby reducing the risk of overfitting—a concern frequently mentioned in previous research.

## **2.3 GUI Integration and Usability Studies**

The integration of a user-friendly GUI with a complex ML-based phishing detection system addresses a critical gap noted in several studies. While many advanced detection systems exist, their usability for non-technical users is often limited. Our literature review highlighted that despite the high accuracy of ML models in phishing detection, the lack of accessible interfaces hinders their practical application. The positive user feedback from our study confirms the findings of previous research that stresses the importance of intuitive design. For example, studies on cybersecurity tool adoption have shown that interfaces which provide clear, actionable information and require minimal technical expertise significantly enhance user engagement and confidence.

Our iterative design process, which included adjustments based on real user feedback, is also consistent with the best practices identified in user interface research. By focusing on enlarging input fields, improving result visibility, and adding an explanatory help section, our GUI not only meets the functional

requirements but also improves overall user satisfaction. This reinforces the argument that usability should be a central consideration in the deployment of cybersecurity solutions, as highlighted by Weaver et al. (2021) and other scholars in the field.

## **2.4 Contrasts with Existing Approaches**

While our findings align with much of the existing literature, there are also areas where our work diverges or contributes new insights. For instance, while many previous studies have focused solely on the technical performance of machine learning models, our research also emphasizes the importance of integrating these models into a user-friendly system. This holistic approach—combining robust detection with intuitive GUI design—sets our work apart from studies that treat these components in isolation.

Additionally, our comprehensive feature extraction process, which categorizes features into address bar-based, domain-based, and HTML/JavaScript-based groups, offers a more granular approach to phishing detection than some earlier models. This detailed segmentation not only enhances the interpretability of the model but also provides a clear framework for future research aimed at refining individual feature thresholds.

## **3. Overall Implications and Future Directions**

The results of our study have significant implications for the development of phishing detection systems. The strong performance of the XGBoost model underscores the potential of ensemble methods in capturing the complex, non-linear relationships present in phishing data. Coupled with a user-friendly GUI, our system demonstrates that high technical performance can be effectively translated into practical, everyday use. The positive user feedback indicates that when complex algorithms are integrated into accessible interfaces, end-users are more likely to adopt and trust the technology.

Looking forward, the findings suggest several avenues for future research. Further improvements in feature selection and model tuning could lead to even higher detection accuracies. Additionally, expanding the dataset and incorporating real-time performance evaluations would provide deeper insights into the model's robustness in dynamic, real-world environments. On the GUI side, continued iterative improvements based on user feedback will be crucial to enhance usability and ensure that the system remains accessible to a broad range of users.

In conclusion, our analysis demonstrates that combining advanced machine learning techniques with a well-designed user interface creates a powerful tool for phishing detection. The alignment of our results with existing literature confirms

the validity of our approach, while our unique contributions—such as the comprehensive feature extraction strategy and the integration of usability considerations—provide a strong foundation for further advancements in the field. This holistic approach not only improves detection accuracy but also enhances user engagement, ultimately contributing to more effective cybersecurity measures against the evolving threat of phishing

## VII. Conclusion.

### Summary of Findings

This research set out to develop and evaluate a machine learning-based phishing URL detection system that integrates a user-friendly graphical user interface (GUI). The study began with a comprehensive literature review of phishing tactics and machine learning approaches, identifying key URL features that distinguish between legitimate and malicious web addresses. The work then detailed a multi-step methodology that included data collection, feature extraction, model training, and GUI development.

Our findings reveal several important insights:

- **Effectiveness of Feature Extraction:** By extracting 15 distinct features—categorized into address bar-based, domain-based, and HTML/JavaScript-based groups—the study was able to capture subtle characteristics inherent in phishing URLs. The analysis confirmed that features such as URL length, depth, redirection behavior, and domain registration details are critical in distinguishing between benign and malicious sites.
- **Superiority of Ensemble Methods:** When comparing various machine learning models (XGBoost, Multilayer Perceptron, Random Forest, Decision Tree, and SVM), ensemble methods, particularly XGBoost, delivered the highest accuracy. XGBoost outperformed other models with a training accuracy of 86.9% and a test accuracy of 85.9%, demonstrating its strength in handling non-linear relationships and complex feature interactions.
- **Practicality of GUI Integration:** The integration of a GUI provided an accessible platform that allowed non-technical users to quickly determine URL safety. User testing demonstrated that the interface was intuitive, with participants correctly classifying an average of 17 out of 20 URLs. The clear visual feedback and ease of use confirmed that coupling advanced detection algorithms with a well-designed user interface significantly enhances user engagement and trust.
- **Balanced Dataset Importance:** The strategy of balancing the dataset by selecting an equal number of phishing and legitimate URLs was pivotal. This approach minimized bias, ensuring that the machine learning model did not favor one class over the other, which is crucial in a cybersecurity context where both false positives and false negatives can have serious implications.



## Implications of Research

The implications of these findings are multifaceted:

- **Advancement in Phishing Detection:** The research underscores the potential of advanced machine learning algorithms, especially ensemble methods, in effectively identifying phishing URLs. By integrating detailed feature extraction with robust model training, the study provides a framework that can adapt to evolving cyber threats.
- **Enhanced Cybersecurity Practices:** With phishing being one of the most pervasive cyber threats, the development of a system that combines high detection accuracy with user-friendly interaction has broad implications. Organizations and individuals alike can leverage such systems to mitigate risk, protect sensitive data, and maintain trust in digital communications.
- **Bridging the Gap Between Research and Application:** Many existing phishing detection systems remain confined to laboratory settings due to their technical complexity. This work demonstrates that it is possible to translate sophisticated machine learning techniques into practical tools that non-experts can use. By doing so, it paves the way for broader adoption of cybersecurity measures, contributing to a safer online environment.
- **Inspiration for Future Studies:** The comprehensive approach taken in this research—covering data collection, feature engineering, model training, and GUI development—serves as a valuable reference for future projects. Researchers can build upon this foundation by exploring additional features, testing alternative algorithms, and refining the interface to cater to a wider audience.

## Recommendations

Based on the study's findings, several recommendations emerge for future research and practical implementation:

- **Expanding the Dataset:** Future research should aim to incorporate larger and more diverse datasets. Including URLs from various geographic regions and industries can enhance the model's generalizability. Real-time data collection and periodic updates can also ensure that the system remains effective against new phishing strategies.
- **Feature Refinement and Expansion:** While the current set of 15 features has proven effective, there is room to explore additional features. For instance, incorporating real-time threat intelligence data or behavioral analytics could further improve detection accuracy. Future studies might also experiment with dynamic thresholding for features such as URL length or domain age based on statistical distributions within specific datasets.
- **Model Optimization and Hybrid Approaches:** Although XGBoost demonstrated superior performance in this study, further optimization is possible. Researchers should investigate hybrid models that combine the strengths of ensemble methods with deep learning architectures. Additionally, fine-tuning

hyperparameters through advanced techniques such as Bayesian optimization might yield incremental improvements in performance.

- **Enhanced GUI Design:** The initial user testing of the GUI yielded positive feedback; however, iterative enhancements can further improve usability. Future iterations could include more detailed explanations of the detection process, customizable alert settings, and integration with mobile platforms to cater to a broader user base. Incorporating user feedback continuously will be essential in refining the interface.
- **Deployment and Real-World Testing:** While the system was rigorously evaluated in a controlled environment, deploying the system in a real-world setting would provide invaluable insights. Organizations should consider pilot programs to assess the tool's performance in live network environments. This practical testing will help identify any unforeseen challenges and ensure that the system can handle the scale and variability of real internet traffic.
- **Interdisciplinary Collaboration:** Cybersecurity is an ever-evolving field that benefits from the intersection of various disciplines. Future research could involve collaboration with experts in human-computer interaction, behavioral psychology, and network security to further enhance both the technical and user-centric aspects of phishing detection systems.

## Conclusion

In conclusion, this research presents a comprehensive approach to phishing URL detection by integrating advanced machine learning techniques with a user-friendly GUI. The study's systematic methodology—from extensive feature extraction and model evaluation to the practical deployment of a graphical interface—demonstrates that it is possible to create an effective, accessible tool for combating phishing attacks.

The findings confirm that ensemble methods, particularly XGBoost, provide a robust solution for identifying complex patterns in phishing URLs, while the balanced dataset and thorough feature engineering significantly contribute to the model's accuracy. Moreover, the positive reception of the GUI by non-technical users underscores the importance of usability in cybersecurity applications.

The implications of this research extend beyond academic inquiry; they offer practical solutions for enhancing cybersecurity defenses and contribute to a safer digital ecosystem. The recommendations provided serve as a roadmap for future enhancements, encouraging ongoing research and collaboration in the fight against cybercrime.

Ultimately, the work reinforces the idea that bridging the gap between sophisticated machine learning algorithms and practical user applications is not only feasible but also essential in today's digital landscape. By continuing to refine and expand upon these findings, future research can further strengthen the tools available to defend against the ever-present threat of phishing attacks, ensuring that technological advancements translate into tangible benefits for users worldwide.



## VIII. References

1. Aslan, Ömer, et al. "A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions." *Electronics* 12.6 (2023): 1333.
2. Gupta, Surbhi, Abhishek Singhal, and Akanksha Kapoor. "A literature survey on social engineering attacks: Phishing attack." *2016 international conference on computing, communication and automation (ICCCA)*. IEEE, 2016.
3. Bhavsar, Vaishnavi, Aditya Kadlak, and Shabnam Sharma. "Study on phishing attacks." *International Journal of Computer Applications* 182.33 (2018): 27-29.
4. Weaver, Bradley W., Adam M. Braly, and David M. Lane. "Training users to identify phishing emails." *Journal of Educational Computing Research* 59.6 (2021): 1169-1183.
5. Arshad, Ayesha, et al. "A systematic literature review on phishing and anti-phishing techniques." *arXiv preprint arXiv:2104.01255* (2021).
6. Azeez, Nureni Ayofe, et al. "Adopting automated whitelist approach for detecting phishing attacks." *Computers & Security* 108 (2021): 102328.
7. Apandi, Siti Hawa, Jamaludin Sallim, and Roslina Mohd Sidek. "Types of anti-phishing solutions for phishing attack." *IOP Conference Series: Materials Science and Engineering*. Vol. 769. No. 1. IOP Publishing, 2020.
8. Jakobsson, Markus. "Two-factor inauthentication—the rise in SMS phishing attacks." *Computer Fraud & Security* 2018.6 (2018): 6-8.
9. Alhogail, Areej, and Afrah Alsabih. "Applying machine learning and natural language processing to detect phishing email." *Computers & Security* 110 (2021): 102414.
10. Shah, S. M. A., Soomro, A., Hussain, K., Rahu, M. A., & Karim, S. (2024). Innovative Machine Learning Solutions: Investigating Algorithms and Applications. *Journal of Applied Engineering & Technology*, 8(1), 32–51
11. Batta, V. (2024). Machine Learning. *International Journal of Advanced Research in Science, Communication and Technology*, 583–591.
12. Tufail, S., Riggs, H., Tariq, M., & Sarwat, A. I. (2023). Advancements and Challenges in Machine Learning: A Comprehensive Review of Models, Libraries, Applications, and Algorithms. *Electronics*, 12(8), 1789.
13. Sharma, P., Begum, R., Shrivastava, P., & Sharma, N. (2024). Machine learning (pp. 138–150).
14. Patil, D. K., Rane, N. L., Desai, P., & Rane, J. (2024). Machine learning and deep learning: Methods, techniques, applications, challenges, and future research opportunities.
15. Mishra, A., & Mishra, P. (2024). Machine and deep learning applications: advancements, challenges, and future directions (pp. 89–98).
16. Rane, N., Mallick, S. K., Kaya, Ö., & Rane, J. (2024). Techniques and optimization algorithms in machine learning: A review.

17. Amuthachenthiru, K., Kaliappan, M., & Vimal, S. (2024). Machine learning and deep learning applications (pp. 204–229).
18. Tulasi, G. (2024). Machine Learning: An In-Depth Review. *IOSR Journal of Computer Engineering*, 26(6), 26–40.
19. Rane, N., Choudhary, S., & Rane, J. (2024). Machine Learning and Deep Learning: a Comprehensive Review on Methods, Techniques, Applications, Challenges, and Future Directions.
20. Hamza, A., Hammam, F., Abouzeid, M., Ahmed, M., Dhou, S., & Aloul, F. (2024). Malicious URL and Intrusion Detection using Machine Learning.
21. Ishima, M. D., & Ikirigo, S. A. (2024). Detection and Classification of Malicious Websites Using Natural Language Processing (NLP) and Machine Learning (ML) Techniques. *International Journal of Scientific Research in Science, Engineering and Technology*, 11(6), 206–221.
22. Hu, Z., & Xu, G. (2023). A Hybrid Binary Neural Tree For Malicious URL Detection.
23. Niyasoui, O., & Reda, O. M. (2024). Malicious URL Detection Using Transformers' NLP Models and Machine Learning (pp. 389–399). Springer International Publishing.
24. Wan, X., Li, P., Wang, Y., Wei, W., & Xiao, L. (2021). Reinforcement Learning Based Accurate Detection of Malicious URLs with Multi-Feature Analysis. *International Conference on Communications*, 17–22.
25. Chaudhari, S., Thakur, A., & Rajan, A. (2024). An Efficient Malicious URL Detection Approach Using Machine Learning Techniques (pp. 485–495). Springer Science+Business Media.
26. Diko, Z., & Sibanda, K. (2024). Comparative Analysis of Popular Supervised Machine Learning Algorithms for Detecting Malicious Universal Resource Locators. *Journal of Cyber Security and Mobility*, 1105–1128.
27. Li, S. P., & Dib, O. (2024). Enhancing Online Security: A Novel Machine Learning Framework for Robust Detection of Known and Unknown Malicious URLs. *Journal of Theoretical and Applied Electronic Commerce Research*, 19(4), 2919–2960.
28. Kocyigit, E., Korkmaz, M., Şahingöz, Ö. K., & Dırı, B. (2024). Enhanced Feature Selection Using Genetic Algorithm for Machine-Learning-Based Phishing URL Detection. *Applied Sciences*, 14(14), 6081.
29. de Luna, R. G., Girang, K. A. R., Lucido, J., Marasigan, L. F. E., Santos, C., & Sebidos, M. L. C. (2024). Improving Cybersecurity: A Comparative Analysis of Machine Learning-Based Uniform Resource Locator (URL) Classification. 78–83.
30. Ishima, M. D., & Ikirigo, S. A. (2024). Detection and Classification of Malicious Websites Using Natural Language Processing (NLP) and Machine Learning (ML) Techniques. *International Journal of Scientific Research in Science, Engineering and Technology*, 11(6), 206–221.
31. Saichand, G., Maheen, I., Apoorva, K., Brahmani, G., & Bharathi, M. (2024). Decoding Deception: Machine Learning Models for Identifying Phishing URLs. 1(2), 24–29.
32. Ashwinkumar, V. K., & Loganathan, V. (2024). Cyber Shield An AI-Driven Solution For Identifying Phishing Websites. 1118–1122.
33. Kodithuwakkuge, M. N. T., M, Ms. M., & Vigila, S. M. C. (2025). Url based phishing detection. *International Scientific Journal of Engineering and Management*, 04(01), 1–6.

34. Reddy, P. V. G. D., Reddy, M. S. S., Praveen, R., & Asif, M. (2024). Innovative approaches to malicious url detection: using machine learning unleashed. *Indian Scientific Journal Of Research In Engineering And Management*, 08(12), 1–7.
35. Ishima, M. D., & Ikirigo, S. A. (2024). Detection and Classification of Malicious Websites Using Natural Language Processing (NLP) and Machine Learning (ML) Techniques. *International Journal of Scientific Research in Science, Engineering and Technology*, 11(6), 206–221.

**Word count: 10040**