Fake It Till You Get Caught Phishing Website Detection Using Machine Learning Algorithms

Arjun A[1], Dr.S.Menaka[2]

[1] *MCA, Department of Computer Applications ,Nehru Institute of Information Technology and Management, Coimbatore, Tamilnadu, India*

[2] *Head Of Department (HOD), Department of Computer Applications, Nehru Institute of Information Technology and Management, Coimbatore, Tamilnadu, India*

---***---

Abstract -With the increasing reliance on digital platforms, phishing attacks have become a significant cybersecurity threat, exploiting user vulnerabilities through deceptive websites that mimic legitimate services. Traditional security measures like antivirus programs and firewalls often fail to prevent such attacks, especially zero-day exploits. This paper proposes a machine learning-based phishing detection system, employing four different algorithms to analyze URLs and leveraging distinct datasets for comprehensive evaluation. The study compares the performance of various machine learning techniques in classifying phishing websites, demonstrating high accuracy in detection. The results highlight the effectiveness of anomaly-based approaches in identifying phishing attempts, offering a dynamic solution to enhance cybersecurity. The findings contribute to the ongoing efforts to mitigate phishing risks, providing a robust framework for real-world applications.

***Key Words*:**phishing detection, machine learning, cybersecurity, URL analysis, anomaly detection, zero-day attacks.

1. INTRODUCTION

The exponential growth of internet usage over the past decade has brought unprecedented convenience and connectivity but has also given rise to an increasing number of cybersecurity threats. Among these, phishing has emerged as one of the most frequent and harmful forms of attack. Phishing typically involves creating deceptive websites that appear identical to legitimate ones with the intent to trick users into revealing sensitive information such as usernames, passwords, banking credentials, and personal data. These malicious sites often closely imitate trusted platforms, such as online banking portals, shopping websites, and social media pages, making them difficult to identify for the average user. The damage caused by phishing extends beyond financial loss, leading to identity theft, data breaches, and erosion of trust in online systems.

Traditionally, the defense against phishing has relied on rule-based systems and blacklists that contain known phishing URLs. While these approaches provide a first line of defense, they suffer from significant limitations. Blacklists are often incomplete, as they depend on prior reports and confirmations of malicious activity. As a result, they are ineffective against newly generated phishing websites—commonly referred to as zero-day attacks—which have not yet been documented or reported. This limitation has prompted researchers and cybersecurity professionals to explore intelligent and adaptive solutions that can identify malicious intent based on URL patterns and behaviors rather than explicit inclusion in a list.

To address these limitations, this project proposes the development of a phishing detection system powered by machine learning algorithms. Machine learning models can analyze the structural and behavioral features of URLs to distinguish phishing sites from legitimate ones, even if the phishing sites have never been encountered before. For this study, a dataset containing a combination of verified phishing and benign URLs was collected and used to train and evaluate multiple machine learning models. These models are trained on a range of features extracted from the URLs, including attributes such as URL length, use of special characters, presence of HTTPS, IP address usage, domain age, and more. These features are known indicators of phishing behavior, and machine learning models can uncover complex relationships among them that may be indicative of fraudulent intent.

Several supervised learning algorithms are utilized at different stages of the project to assess their effectiveness in phishing detection. The K-Nearest Neighbors (KNN) algorithm serves as a baseline classifier. It operates by comparing new input data with existing labeled data points and classifies the input based on the majority class of its nearest neighbors. Although KNN is simple and intuitive, its performance can degrade with high-dimensional data or large datasets due to its reliance on distance metrics.

Next, the Support Vector Machine (SVM) is applied to the problem. SVM is particularly effective in binary classification tasks like phishing detection, where it tries to find the optimal hyperplane that maximally separates the two classes. By using a linear kernel in this project, SVM is able to identify decision boundaries based on the extracted URL features, even when the separation is subtle and complex.

The Random Forest algorithm is also implemented as a robust ensemble learning method. It builds multiple decision trees during training and combines their outputs to make the final prediction. Random Forest is known for its ability to handle noisy data and avoid overfitting by averaging multiple weak learners. Its performance on phishing detection tasks is generally strong, as it captures feature interactions and nonlinear patterns effectively.

Additionally, the project incorporates Extreme Gradient Boosting (XGBoost), a highly optimized and scalable gradient boosting algorithm. XGBoost constructs a series of decision trees where each subsequent tree attempts to correct the errors of its predecessors. Its efficiency and accuracy make it well-suited for phishing detection, especially when dealing with imbalanced datasets or subtle feature variations. XGBoost's ability to fine-tune learning through parameters like learning rate and maximum tree depth offers a significant advantage in achieving high precision and recall.

Each of these algorithms brings unique strengths to the classification task, and their performances are evaluated and compared using accuracy, precision, recall, and F1-score metrics.

Through experimental analysis, the goal is to determine which model performs best under the specific constraints of phishing website detection. The combination of feature engineering and multiple learning algorithms provides a comprehensive framework to understand the effectiveness of machine learning in combating modern phishing threats.

This project not only demonstrates the applicability of machine learning models in real-world cybersecurity problems but also contributes toward developing proactive solutions capable of adapting to evolving attack strategies. By relying on intelligent models that learn from data, rather than static rule-based systems, the proposed approach enhances the ability to detect phishing websites dynamically and in real-time, providing a vital layer of defense for today's digital users.

2. RELATED TO WORK

2.1 PROBLEM STATEMENT

In today's digital era, the exponential growth of internet usage has made individuals increasingly vulnerable to various forms of cyberattacks, with phishing emerging as one of the most widespread and damaging threats. Phishing attacks primarily exploit human trust by masquerading as legitimate websites or services in order to trick users into divulging confidential information such as usernames, passwords, banking credentials, or other personal data. These fraudulent websites often employ URLs that are deceptively similar to those of reputable platforms, making it challenging for even experienced users to distinguish between authentic and malicious domains.

Traditional methods of phishing detection, such as blacklists or manual reporting, often fall short due to the dynamic and ever-evolving tactics used by attackers. New phishing URLs can emerge and disappear within hours, rendering static detection techniques largely ineffective. Hence, there is a critical need for a more adaptive and intelligent solution that can analyze the intrinsic characteristics of a URL and its associated metadata to predict its legitimacy in real time.

This project addresses the pressing issue of phishing detection by proposing the development of an intelligent, web-based detection system powered by machine learning. The system is designed to accept a user-submitted URL, perform comprehensive feature extraction encompassing syntactic, lexical, and host-based properties, and utilize pre-trained machine learning algorithms to classify the URL as either safe or malicious. Moreover, to foster transparency and build user confidence, the system provides an explanatory summary highlighting the key features that contributed to the final decision, especially in cases where a URL is deemed unsafe.

By automating the detection process through machine learning and enhancing user awareness through clear feedback, this project aims to significantly reduce the risk posed by phishing attacks and contribute toward a safer online experience.

2.2. METHODOLOGY

The proposed phishing detection system leverages **supervised machine learning techniques** to effectively classify URLs as either phishing or legitimate. The methodology is structured around a systematic pipeline that ensures both accuracy and efficiency in detecting malicious web addresses.

To begin with, datasets were obtained from two credible sources: **PhishTank** and the **UCI Machine Learning Repository**. These datasets collectively consist of approximately **10,000 URLs**, carefully balanced to include an equal distribution of phishing and legitimate entries, thereby supporting unbiased model training and evaluation.

The development workflow for the system includes the following key stages:

1. **Data Preprocessing and Cleaning:**
Raw URL data often contains noise, inconsistencies, and redundant information. A thorough preprocessing phase was implemented to remove duplicates, normalize URL formats, and handle missing values to ensure high data quality.

2. **Feature Engineering and Extraction:**
A diverse set of **heuristic, lexical, and behavioral features** was extracted from each URL. These features include, but are not limited to:

- Length of the URL
- Presence of special characters (such as @, //, or -)
- Use of HTTPS protocol
- Frequency of redirection
- Domain age and registration validity
- Use of IP addresses in place of domain names

3. **Dataset Construction and Serialization:**

he extracted features were compiled into a structured and labeled dataset suitable for machine learning algorithms. This dataset was then saved in a **serialized (.pkl) format** to facilitate quick loading and reusability during model training and testing phases.

4. **Data Splitting:**

To evaluate the model's generalization capability, the dataset was split into **training (80%) and testing (20%) subsets**. This stratified split ensures that both subsets maintain the class balance and represent the overall data distribution.

5. **Model Training and Evaluation:**

Multiple classification algorithms were trained on the training set. These included ensemble-based methods, support vector machines, and decision tree classifiers. The models were assessed on various performance metrics such as accuracy, precision, recall, and F1-score using the testing set.

The selected features aim to capture the intrinsic characteristics that differentiate phishing URLs from legitimate ones. This includes both **structural patterns** (like URL format and content attributes) and **contextual behaviors** (like domain history and security indicators). Together, these steps form a robust pipeline for phishing URL detection using machine learning techniques.

2.3 ALGORITHMS

Multiple machine learning algorithms were employed and compared in this study to evaluate their effectiveness in phishing detection:

- **K-Nearest Neighbors (KNN):** A simple, instance-based learning algorithm that classifies a URL based on the majority class among its k-nearest neighbors in the feature space.

- **Support Vector Machine (SVM):** A robust classifier that seeks to find the optimal hyperplane that separates phishing and legitimate URLs with the widest possible margin.

- **Random Forest (RF):** An ensemble method that constructs multiple decision trees during training and outputs the mode of their predictions. It is particularly useful for handling high-dimensional data with nonlinear interactions.

- **Extreme Gradient Boosting (XGBoost):** An advanced boosting algorithm that builds models sequentially to correct the errors of previous models. It is highly efficient and often delivers superior accuracy compared to traditional classifiers.

2.4 IMPLEMENTATION

The system is built using Python, leveraging popular libraries such as:

- pandas and numpy for data handling,

- scikit-learn for preprocessing and classical machine learning models,

- xgboost for gradient boosting,

- tensorflow/keras for deep learning,

- matplotlib for visualization, and

- pickle for serializing trained models.

Development and experimentation were conducted in Google Colab due to its free access to GPU resources.

The web interface was developed using HTML and CSS, while Flask was used to serve the model as a lightweight web application. The user interacts with the system through a simple front-end where they can enter a URL and receive instant feedback on its safety status.

2.5 RESULTS AND DISCUSSION

Each model was evaluated on both training and testing datasets to assess their generalization capability:

Table -1 : Accuracy Comparison

Algorithm	**Training Accuracy**	**Testing Accuracy**
KNN	85.2%	85.5%
Random Forest	82.1%	82.3%
XGBoost	86.7%	86.4%
SVM	80.0%	80.7%

From the results, it is evident that XGBoost outperforms other models, offering the best balance of training and testing accuracy. KNN also performs well, but may suffer from inefficiencies at scale. Although SVM has the lowest accuracy among the models tested, it still shows consistent generalization. DNN was also explored for future enhancements, though its results are not tabulated here due to the scope of experimentation. The use of multiple algorithms provided valuable insights into which features and models are most effective in identifying phishing websites.

2.6 PERFORMANCE EVALUATION

The performance of each model was evaluated using standard classification metrics, including **accuracy, precision, recall, and F1-score**. These metrics help understand the trade-off between false positives (safe URLs classified as phishing) and false negatives (phishing URLs classified as safe).

- **Accuracy** measures the overall correctness of the model.
- **Precision** measures how many predicted phishing URLs were actually phishing.
- **Recall** (sensitivity) captures how many real phishing URLs were correctly detected.
- **F1-score** balances both precision and recall.

Additionally, **confusion matrices** were generated for each classifier to visualize their prediction distribution.

3. REQUIREMENT SPECIFICATION

The requirement specification outlines the essential hardware and software components needed to develop and execute the project efficiently. This section defines the baseline technical environment, ensuring the system operates smoothly and meets the intended performance benchmarks.

3.1 HARDWARE REQUIREMENTS

Hardware refers to the physical components of a computer system that are necessary to support the software and operating system. The following are the minimum hardware specifications required for the successful deployment and functioning of the project:

Table -2 : Hardware Requirements

Processor	Intel Dual-Core or higher
Clock Speed	Minimum 2.4 GHz
Primary Memory (RAM)	Minimum 4 GB (Recommended: 8 GB)
Hard Disk	At least 1 TB of storage
Processing Speed	600 MHz or above
Keyboard	Standard 104-key layout
Display	Minimum 720p resolution display

3.2 SOFTWARE REQUIREMENTS

Software requirements detail the programs and development tools needed to build and run the application. These tools must be properly installed and configured before the system can be executed. The following are the minimum software requirements for this project:

Table -3 : Software Requirements

Programming Language (Frontend)	Python
Development Environment (IDE)	Anaconda Navigator or Jupyter Notebook
Operating System	Microsoft Windows 10 or higher (64-bit recommended)
Additional Libraries	Required Python libraries (e.g., pandas, scikit-learn, Flask) should be installed based on the project modules.

4. SYSTEM IMPLEMENTATION

The system implementation phase focuses on transforming the design and methodology into a functional, executable application. In the case of this phishing URL detection system, the implementation is divided into several modular components, each playing a crucial role in ensuring accurate and efficient detection of malicious URLs.

4.1 DATA HANDLING AND PREPROCESSING

The initial step involves acquiring and cleaning datasets obtained from trusted sources such as PhishTank and the UCI Machine Learning Repository. The combined dataset includes a balanced mix of phishing and legitimate URLs. Data preprocessing techniques such as removal of duplicates, normalization of URL strings, and handling of missing values were applied to prepare the data for feature extraction.

4.2 FEATURE EXTRACTION MODULE

A dedicated Python module was implemented to extract a comprehensive set of lexical, structural, and behavioral features from each URL. This includes characteristics such as:

- URL length and token count
- Presence of suspicious symbols (e.g., @, //, -)
- Use of IP addresses instead of domain names
- Age and expiration details of the domain
- Presence or absence of HTTPS protocol
- Redirection count and anchor tag analysis

The extracted features were stored in a structured format using Pandas DataFrames, enabling easy manipulation and integration with machine learning pipelines.

4.3 MODEL TRAINING AND EVALUATION

Multiple machine learning algorithms—including Random Forest, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and XGBoost—were implemented using the Scikit-learn and XGBoost libraries. The dataset was split into training and testing subsets in an 80:20 ratio. Each model was trained using the extracted features, and their performance was evaluated based on key metrics such as accuracy, precision, recall, and F1-score.

4.4 WEB APPLICATION INTEGRATION

To enhance usability, a lightweight web interface was developed using the Flask web framework. The interface allows users to input a URL and receive a prediction indicating whether it is safe or potentially phishing. The backend integrates the trained machine learning model and feature extraction logic to deliver real-time responses. The application is hosted locally for testing and can be deployed on a cloud platform if needed.

4.5 SYSTEM WORKFLOW

The overall system operates through the following sequence:

1. User inputs a URL through the web interface.
2. The backend extracts features from the URL.
3. These features are passed to the trained classifier.
4. The system returns a classification result: *Phishing* or *Legitimate*, along with reasoning if applicable.

5. SAMPLE OUTPUT SCREEN’S

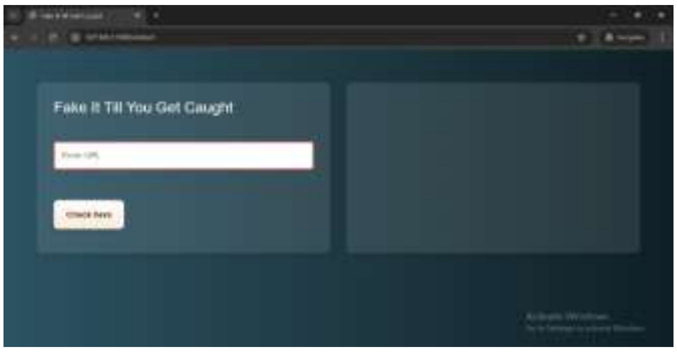
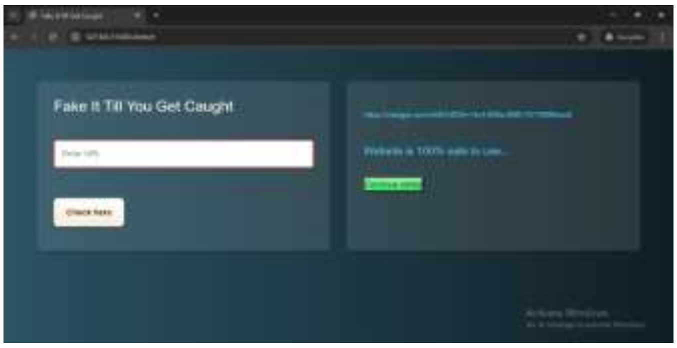


Figure 2.1 :URL Phishing Detection Page



Website is 100% safe to use...

Figure 2.2 : URL Phishing Detection - output safe

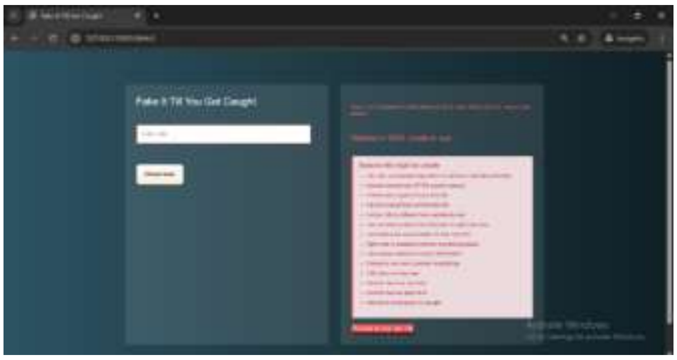


Figure 2.3 :URL Phishing Detection - output not safe

3. CONCLUSIONS

The increasing sophistication and frequency of phishing attacks have made traditional blacklist-based detection techniques insufficient for protecting users from malicious websites. In this project, we addressed the pressing need for a more intelligent and proactive approach to identifying phishing URLs by developing a machine learning-based detection system. By analyzing patterns in URL structures and extracting meaningful features, we demonstrated that ML algorithms can provide accurate, scalable, and automated solutions for detecting phishing threats in real time.

We used two reliable datasets—PhishTank and UCI—containing a balanced set of 5000 phishing and 5000 legitimate URLs. From these, we extracted a set of heuristic-based and lexical features such as the presence of special characters, domain length, use of HTTPS, redirection patterns, and many more. These features were fed into various machine learning models including K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Random Forest, and XGBoost. The models were trained using an 80-20 split between training and testing sets, ensuring both sufficient learning and fair evaluation. Among the models, XGBoost achieved the highest accuracy on both training and testing data, demonstrating its effectiveness for this task.

The entire workflow—from data preprocessing, feature engineering, and model training to final integration into a web application using Flask—was designed to be lightweight and efficient. Users can submit a URL through the interface and receive an immediate response indicating whether the link is safe or suspicious, along with potential reasons for why a URL is flagged as unsafe. This makes the system not only functional but also user-friendly and informative.

A significant outcome of this project is the demonstration that machine learning models can generalize well to new, unseen URLs when trained with carefully selected features. Furthermore, the models avoid dependence on content-based features, which may not be available for short-lived phishing pages, thereby ensuring broader applicability.

Despite the strong results, we also recognize certain limitations. For instance, the performance of the model can be affected by rapidly changing phishing techniques that disguise malicious intent through obfuscation or mimicry. Additionally, the static nature of the training data could limit the long-term adaptability of the model if it is not regularly updated.

In conclusion, this project provides a strong foundation for automated phishing detection using machine learning. It proves that even without relying on page content or third-party reputation services, intelligently extracted URL features can serve as powerful indicators of phishing activity. With further enhancements, such systems can significantly reduce user exposure to online fraud, and serve as reliable tools in broader cybersecurity frameworks

4. FUTURE SCOPE

Looking ahead, several enhancements can further improve the reliability and robustness of phishing detection systems:

- **Hybrid Feature Integration:** Combining lexical, content-based, and host-based features—such as HTML structure, JavaScript behavior, and server metadata—can enhance detection precision.
- **Real-Time Detection Systems:** Implementing this solution in browser extensions or network-level tools can enable instant identification of suspicious URLs, reducing the window of exposure for users.
- **Deep Learning Techniques:** Employing deep neural networks and hybrid models such as CNN-DNN or RNN-based classifiers may boost adaptability and improve detection of visually deceptive websites.
- **Adversarial Robustness:** As attackers increasingly use obfuscation and evasion techniques, developing models resistant to adversarial manipulation is crucial. Research into adversarial machine learning could strengthen defenses.
- **Explainable AI (XAI):** Integrating explainability can help users and security teams understand the rationale behind a model's predictions, increasing transparency and trust in automated systems.
- **Mobile and Lightweight Deployment:** With phishing attacks becoming common on mobile platforms, there's a need to develop optimized models suitable for smartphones and resource-constrained environments.
- **Crowdsourced Blacklist Integration:** Collaborating with community-driven reporting platforms can help detect zero-day phishing URLs that bypass traditional blacklists.
- **Automated Feature Engineering:** Leveraging deep learning for automatic feature extraction, along with synthetic data generation techniques, can expand the training dataset and improve model generalization.
- **Enterprise-Scale Integration:** Embedding phishing detection into broader security infrastructures such as SIEM systems and secure email gateways can offer more comprehensive protection for organizations.

By pursuing these directions, future systems can become more dynamic, accurate, and scalable, ensuring stronger defenses against evolving phishing threats.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who supported and guided me throughout the course of this project.

First and foremost, I extend my heartfelt thanks to my project guide, Dr.S.Menaka, for their constant encouragement, valuable insights, and constructive feedback, which helped shape this project from the ground up. Their expertise and support were instrumental in every phase of the work.

I am also thankful to the faculty and staff of the Department of Computer Applications[Nehru Institute of Information Technology and Management], for providing the necessary resources and a positive environment conducive to research and development.

I would like to acknowledge the creators and maintainers of the datasets and tools used in this project, such as PhishTank and the

UCI Machine Learning Repository, which were essential to building and validating the phishing detection model.

A special thanks to my peers and friends for their continuous motivation, support, and helpful discussions throughout the project duration.

Lastly, I am deeply grateful to my family for their unwavering support, patience, and encouragement during this academic journey.

REFERENCES

[1] T. Peng, I. Harris, and Y. Sawa, “Detecting Phishing Attacks Using Natural Language Processing and Machine Learning,” 2018 IEEE 12th International Conference on Semantic Computing (ICSC), pp. 300-301, 2018.

[2] Republic of Turkey, “National Cyber Security Strategy, 2016,” Ministry of Transport Martime Affairs and Communications.

[3] R. Loftus, “What cybersecurity trends should you look out for in 2020?,” Daily English Global blogkasperskycom. [Online]. Available: https://www.kaspersky.com/blog/secure futures-magazine/2020- cybersecurity-predictions/32068/. [Accessed: 09-Mar-2020].

[4] E. Buber, Ö. Demir and O. K. Sahingoz, "Feature selections for the machine learning based detection of phishing websites," 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, 2017, pp. 1-5.

[5] “Retruster,” Retruster. [Online]. Available: https://retruster.com/blog/2019-phishing-and email-fraud-statistics.html. [Accessed: 09-Mar-2020].

[6] “Phishing Activity Trends Reports, 1st-2nd-3rd Half” APWG. [Online]. Available: https://apwg.org/trendsreports/. [Accessed: 09-Mar-2020].

[7] Y. Cao, W. Han, and Y. Le, “Anti-phishing based on automated individual white-list,” Proceedings of the 4th ACM workshop on Digital identity management - DIM 08, pp. 51–60, 2008.

[8] M. Sharifi and S. H. Siadati, “A phishing sites blacklist generator,” 2008 IEEE/ACS International Conference on Computer Systems and Applications, pp. 840–843, 2008.

[9] M. Khonji, Y. Iraqi, and A. Jones, “Phishing Detection: A Literature Survey,” IEEE Communications Surveys & Tutorials, vol. 15, no. 4, pp. 2091–2121, 2013.

[10] Y. Zhang, J. I. Hong, and L. F. Cranor, “Cantina,a content based approach to detecting phishing web sites” Proceedings of the 16th international conference on World Wide Web - WWW 07, pp. 639-648, 2007.