**PhishSecure: A Phishing Website Detection**

**using Machine learning**

**Abstract**

Cybersecurity is about protecting of computer systems, networks and information from harm, theft and unauthorized use from cyber threats such as phishing, malware, and hacking. This involves tools like firewalls, antivirus software, and detection systems to prevent unauthorized access and protect sensitive information. Among these threats, phishing is a common internet scam where attackers send fake messages pretending to be from trusted sources. These messages frequently contain URLs that when clicked, can steal personal information or infect systems with malware. Existing systems primarily focus on features, such as HTTPS presence and URL preprocessing, to identify phishing attacks. However, the proposed system emphasizes the lexical features of URLs in conjunction with machine-learning algorithms. In this approach, URLs received by users are analyzed using machine learning models to classify them as phishing or legitimate. Various algorithms, such as Support Vector Machines (SVM), Neural Networks, Random Forest, Decision Tree, and XG Boost, can be applied for this purpose. Among these, the gradient boosting classifier was selected for its robustness and accuracy. By extracting and comparing the key characteristics of legitimate and phishing URLs, the proposed system effectively identifies phishing websites in real-time. The results show that this method is accurate and effectively distinguishes safe websites from phishing ones, providing a reliable solution against phishing attacks.

**1.Introduction**

In today’s world internet is essential for our daily life, helping in banking, entertainment, online payments and education. These technologies help the people do tasks easily and quickly. Mobile and wireless networks have made this even better by providing internet access anytime and anywhere. However, many people are using digital platforms they face various security threats such as spam, phishing and fraud. Among this phishing is a cybercrime and very popular these days. Phishing is an internet scam sending fraudulent links that appear to come from the reputable sources. According to [1] IBM's 2019 "Cost of a Data Breach" report, the average cost of a data breach has risen 12% over the past 5 years and now costs $3.92 million on average. Additionally, [2] Cybersecurity Ventures predicts that global cybercrime costs will grow by 15 percent per year over the next five years, reaching $10.5 trillion USD annually by 2025. Among these attacks, phishing is one of the most widespread and critical, causing both financial losses and intangible damages.

The infection starts when a user unknowingly clicks on a phishing link or downloads a harmful attachment. This can result in malware being installed, stealing of login details, or direct theft of money. The Domain Name System (DNS) is crucial in phishing attacks because attackers use fake domains that look like real ones to trick users. DNS spoofing and cache poisoning are often used to send users to fake phishing websites. Attackers often change the structure of URLs to make them look real. They might replace characters (like "g00gle.com" instead of "google.com"), use subdomains to mimic trusted sites, or use HTTPS to seem secure. They also use certain top-level domains (TLDs) like .xyz or .info because they are cheaper and have fewer rules.

Traditional methods for detecting phishing are not enough to handle the changing threats. The blacklist method, which updates lists of bad URLs and IP addresses, doesn't work well against new phishing attacks. Attackers can easily avoid these lists by hiding URLs, changing them quickly, or creating new ones automatically. Heuristic-based detection methods, which look for common signs of phishing attacks, also have problems. These signs are not always there and can lead to many false alarms. These limitations show the need for better and more flexible phishing detection methods. This has led to exploring new approaches like machine learning-based solution.

As of 2022, the average cost of a data breach is $4.35 million (IBM) [5]. Phishing attacks account for 90% of all data breaches [3] (Verizon). Business Email Compromise (BEC) [4] scams resulted in losses of over $43 billion between 2016 and 2022 (FBI).Machine learning is used to detect phishing attacks because it can find patterns and unusual behavior in data, making detection more accurate. Traditional methods with fixed rules don't work well against new phishing tricks, but machine learning can adapt to new threats. Its ability to learn over time makes it a great solution.

This research proposes a novel approach to phishing detection by leveraging the lexical features of URLs in conjunction with machine learning algorithms. By analyzing the URL's linguistic characteristics, this approach aims to improve the accuracy and effectiveness of phishing detection. Through a comprehensive evaluation of prominent machine learning algorithms, including SVM, Neural Networks, Random Forest, Decision Tree, and XG Boost, this study seeks to identify the optimal solution for detecting phishing attacks and enhancing online security.

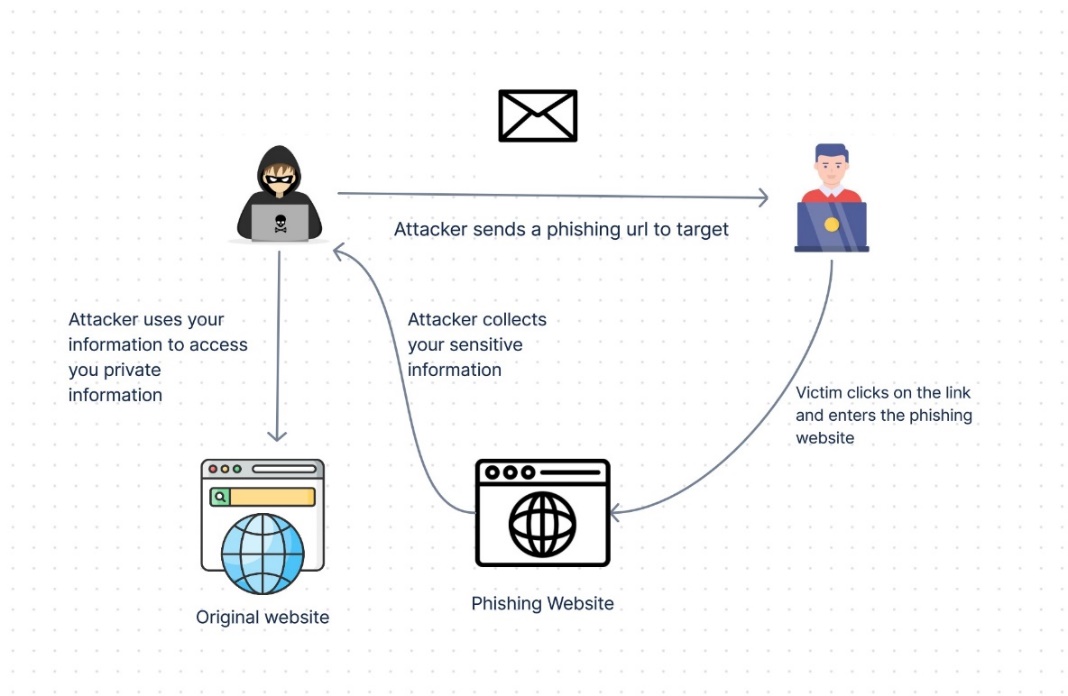


Fig 1 Life cycle of a phishing attack

**2.Literature Review**

In this section, existing system approach for phishing detection and their gaps and limitations are summarized.

1. *Traditional List-Based Phishing Detection*

List-based phishing detection uses blacklists of known phishing websites and whitelists of trusted websites. Whitelists have verified, legitimate websites and only allow access if the URL is on the list. Blacklists record known phishing websites and block access to those URLs. In [10] Researchers have found whitelists effective, as shown in studies where access is limited to websites on the whitelist. In [11] Blacklist-based systems are commonly used in tools like Google Safe Browsing API and PhishNet, as well as in research. However, they have major challenges. Small changes in a phishing URL can avoid detection, and new threats, called zero-day attacks, go unnoticed because the lists are updated reactively. These issues show the need for additional methods, like machine learning or heuristic analysis, to improve the effectiveness of list-based detection.

1. *Rule-Based detection*

Rules-based phishing detection systems identify harmful websites by using predefined rules to analyze features like URL structure, domain details, and webpage behavior [12] .They also flag behaviors such as hidden content in iframes, multiple redirects, or disabled right-clicks. While these systems are efficient and straightforward, they need constant updates to keep up with new threats. They can also produce false positives and struggle with advanced phishing tactics. Therefore, they work best when combined with dynamic methods like machine learning.

*c. Machine learning based detection*

Machine learning-based phishing detection systems use algorithms to analyze and classify websites based on patterns and features from data. These systems learn from past data, looking at key details like URL structure, domain age, HTTP usage, and webpage content. Algorithms such as Random Forest, Support Vector Machines (SVM), and Neural Networks are often used for these tasks. Machine learning is good at finding complex and subtle patterns that traditional methods might miss, making it effective at detecting new phishing attacks.

In [6] proposed a machine learning-based system for phishing detection using eight algorithms tested on three datasets, focusing on URL features. While the system achieved high accuracy, it had some gaps. It lacked diverse features, real-time performance testing, and a web application for users. It also didn't include adaptive learning or a detailed comparison of algorithm performance. These gaps show the need for systems that are accurate, user-friendly, adaptable, and capable of real-time detection.

The paper [7] proposes an efficient phishing detection method using machine learning, specifically with Support Vector Machine (SVM) classifiers. This method achieves 95.66% accuracy in identifying phishing websites using only 22.5% of new features. The authors compare their approach with standard phishing datasets and show promising results. However, the paper does not discuss optimizing feature selection or dealing with high computational costs and false positives in detail.

The research paper [8] suggests a Google Chrome extension to detect phishing websites using blacklisting and semantic analysis. It uses a database of known phishing sites and checks on-site data like text, links, and images to find patterns. The method was tested and compared with other methods, showing it works well. However, it depends on pre-collected databases, which might miss new phishing tricks. The approach combines blacklisting with semantic analysis and redirection detection to better prevent phishing.

In [9] The paper suggests a system to detect phishing URLs using machine learning, aiming to warn users about phishing attacks in real-time. It highlights the challenge of quickly changing phishing tactics. The main gap is the lack of an automated process for detecting phishing attacks. The method involves analyzing URLs for patterns and features that indicate phishing, allowing for the detection of multiple phishing attempts at once.

In the system outlined in [13], the researchers enhanced the number of NLP vectors and evaluated the performance of three machine learning algorithms to assess their accuracy. The algorithms tested were Random Forest, SMO (Sequential Minimal Optimization), and Naïve Bayes. Among these, the Random Forest algorithm outperformed the others, achieving the highest accuracy of 97.2% when used in a hybrid approach.

**3.Methodology**

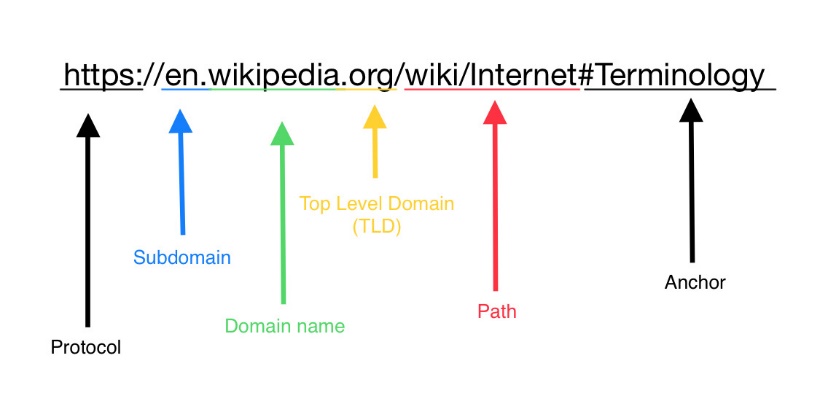
In this study, we concentrated on creating a phishing detection system by examining the URLs of web pages. A URL is a complex string that syntactically and semantically represents an online resource. A detailed analysis of the structural composition of a URL is depicted in Figure 2.

Fig 2: Structure of URL

Fields such as domain, subdomain, top-level domain (TLD), protocol, directory, file name, path, and query are used to form different URLs. In phishing URLs, these fields often vary from those in legitimate URLs. This makes URLs crucial for detecting phishing attacks, as they help in quickly classifying web pages.

*3.1 Dataset Overview:*

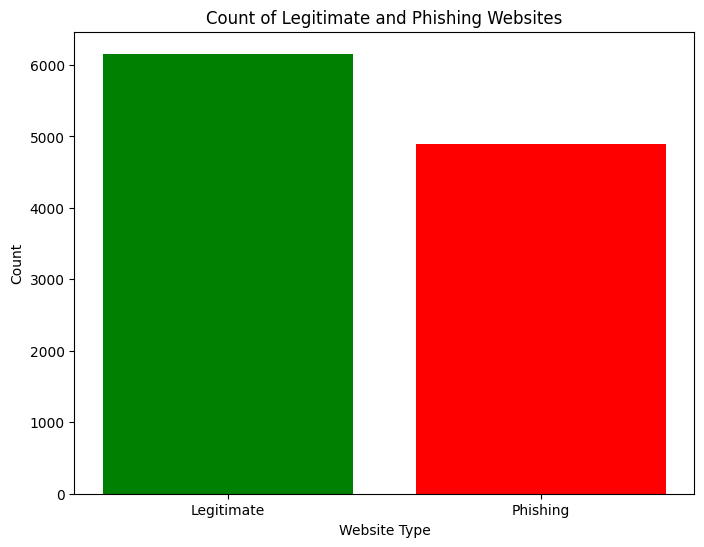
The dataset used in this study is taken from Kaggle and is available in two formats: a .txt file and a .csv file. The .txt file does not have headers, so users need to add them manually, while the .csv file includes headers for easy use. The dataset has 11,054 entries with 32 features. Each entry contains 30 parameters describing the characteristics of a website, such as its structure and behavior. It also has a class label that shows if the website is phishing (1) or legitimate (-1). The features provide details about different parts of the URL. For example, the UsingIP feature checks if an IP address is used in the URL, and LongURL identifies very long URLs. The ShortURL feature detects the use of shortened URLs. Other features, such as HTTPS and DomainRegLen, check if the URL uses HTTPS and the length of domain registration. Features like RequestURL and AnchorURL analyze objects and links in the URL.

Fig 3 Count of phishing and legitimate in the dataset

*3.2 Data preprocessing:*

In the data preprocessing stage, we first examined the dataset to understand its structure and content. We checked the number of rows and columns and reviewed the column names for clarity. Using data.info(), we retrieved basic information such as data types and non-null counts. We also assessed the uniqueness of values in each column with data.nunique(). An irrelevant column, "Index", was removed to streamline the dataset and focus on important features for phishing detection. These steps ensured the dataset was clean and ready for further analysis.

*3.3 Feature Extraction:*

The performance of our trained system relies on the features used and the machine learning algorithms applied. To identify the most important features, we reviewed existing research thoroughly. This included studies on URL analysis as well as features from other categories like content and website characteristics.

In our study, we analyzed different parts of the URLs, such as the hostname, domain, and path. We extracted 30 unique features from the URLs to capture their characteristics. These features were generated using Python scripts to ensure accurate and efficient extraction for phishing detection.

In this project, we identified 30 attributes from website URLs. These attributes help differentiate phishing websites from legitimate ones.

We analyzed URLs to detect IP addresses, URL length, shortened URLs, special characters like "@", and redirect behavior. For domain properties, we looked at hyphens, subdomains, HTTPS usage, and domain registration length. We also examined HTML and website content for anomalies in favicon links, request URLs, anchor tags, script tags, and form handlers. We assessed behavioral features like website forwarding, status bar customization, disabled right-click, pop-ups, and iframe redirection. We also measured domain and traffic metrics such as domain age, DNS records, website traffic, page rank, Google indexing, and links pointing to the page. Additionally, we identified suspicious URLs and IP addresses using predefined patterns.

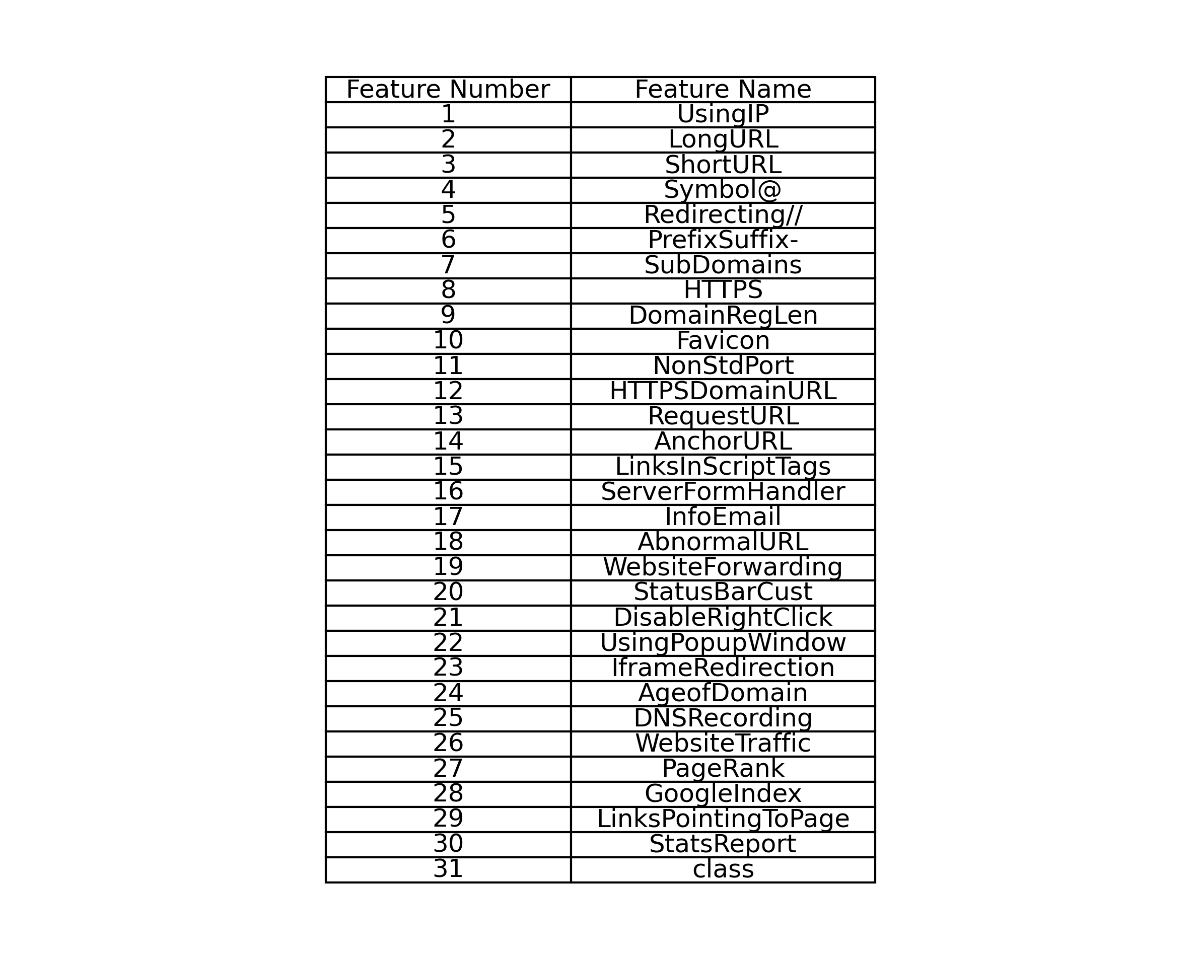


Table 1: Features in the dataset

*3.4 Model Building & Training:*

In this study, we used supervised machine learning techniques to detect phishing websites. We split the dataset into 80% for training and 20% for testing, using a random split (random\_state = 42) for reproducibility. We separated the target variable, 'Label', from the features, which were used as inputs for the models. We trained several classification models, including Logistic Regression, k-Nearest Neighbors, Support Vector Classifier, Naive Bayes, Decision Tree, Random Forest, Gradient Boosting, Xgboost, and Multilayer Perceptrons. We evaluated the models using Accuracy and F1 score to measure their performance. This helped us find the most effective model for detecting phishing websites based on the given features.

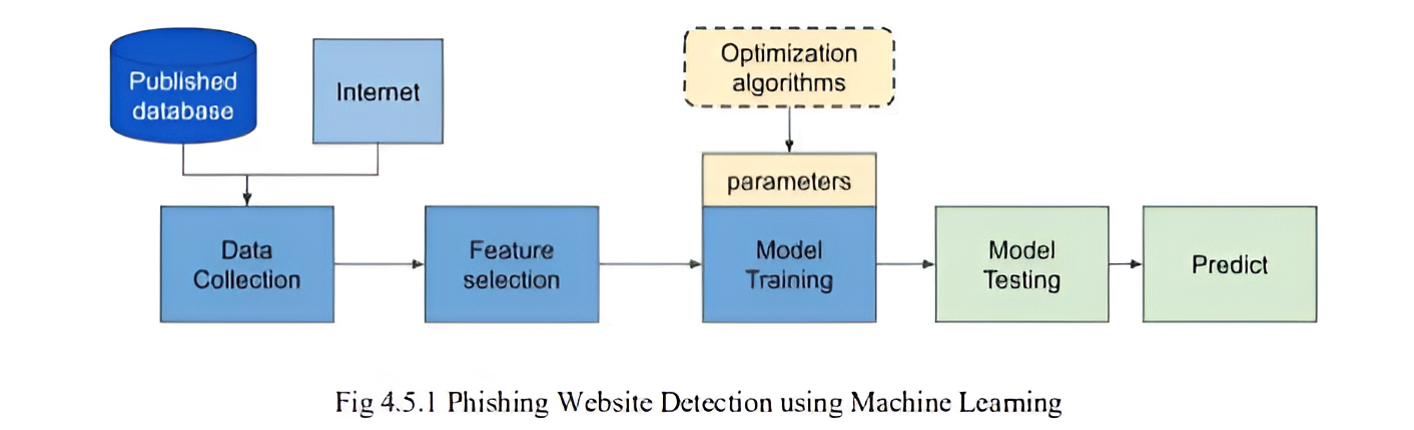
*Logistic Regression* was used to predict whether a website is phishing or legitimate, making it ideal for binary classification. This method calculated the probability of a website belonging to one of the two categories, providing a straightforward and reliable approach for the task*. K-Nearest Neighbors (K-NN)* is a method to determine if a website is safe or a scam. It compares a new website to known safe and scam websites by looking at various details like the URL and content. K-NN finds the most similar websites and decides if the new one is safe or a scam based on whether the similar websites are mostly safe or scams. This method works well because similar websites tend to have similar risks. *Support Vector Machines (SVM)* were used to find the best line hyperplane that separates phishing websites from legitimate ones. By identifying this boundary, SVM accurately classified websites, even when the data was hard to separate. This ensured the algorithm effectively distinguished between phishing and legitimate websites.

*Naive Bayes*, based on Bayes' theorem, was chosen for its simplicity and speed, especially with data having many features. By assuming that features are independent, it provides quick and reliable predictions. This makes it ideal for analyzing text, which is common in detecting phishing attacks. *Decision Trees* were used to classify websites with a tree-like model. In this model, internal nodes represent website features, branches represent decision rules, and leaves indicate the final classification. Visualizing the decision tree helped identify the most important features for distinguishing phishing websites. *Random Forest*, which combines the results of many decision trees, was used to improve classification accuracy. By averaging the predictions from multiple trees, this method reduced the risk of overfitting and increased the model's reliability. This created a strong and effective model for detecting phishing websites.

*Gradient Boosting* combines many simple models to create a powerful one. It uses decision trees as the base models and fine-tunes them to balance accuracy and avoid overfitting. This approach greatly improved performance, making it very effective for complex phishing detection. *CatBoost* is an advanced machine learning algorithm that handles different types of features well, especially categorical ones. Its compatibility with deep learning tools and ability to manage various data types make it versatile for phishing classification, offers fast and accurate predictions**.** *Multi-layer Perceptron classifier*, is a type of classification algorithm that uses a neural network. This neural network uses multiple interconnected layers to learn complex relationships between website features and whether it's a phishing site. By capturing detailed patterns in the data, it greatly improves the accuracy of identifying phishing websites.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ML Model** | **Accuracy** | **f1\_score** | **Recall** | **Precision** |
| **Gradient Boosting Classifier** | **0.974** | **0.974** | **0.988** | **0.989** |
| **CatBoost Classifier** | **0.972** | **0.972** | **0.990** | **0.991** |
| **Random Forest** | **0.966** | **0.970** | **0.992** | **0.990** |
| **Multi-layer Perceptron** | **0.966** | **0.966** | **0.986** | **0.986** |
| **Support Vector Machine** | **0.964** | **0.968** | **0.980** | **0.965** |
| **Decision Tree** | **0.960** | **0.964** | **0.991** | **0.993** |
| **K-Nearest Neighbors** | **0.956** | **0.961** | **0.991** | **0.989** |
| **Logistic Regression** | **0.934** | **0.941** | **0.943** | **0.927** |
| **Naive Bayes Classifier** | **0.605** | **0.454** | **0.292** | **0.997** |

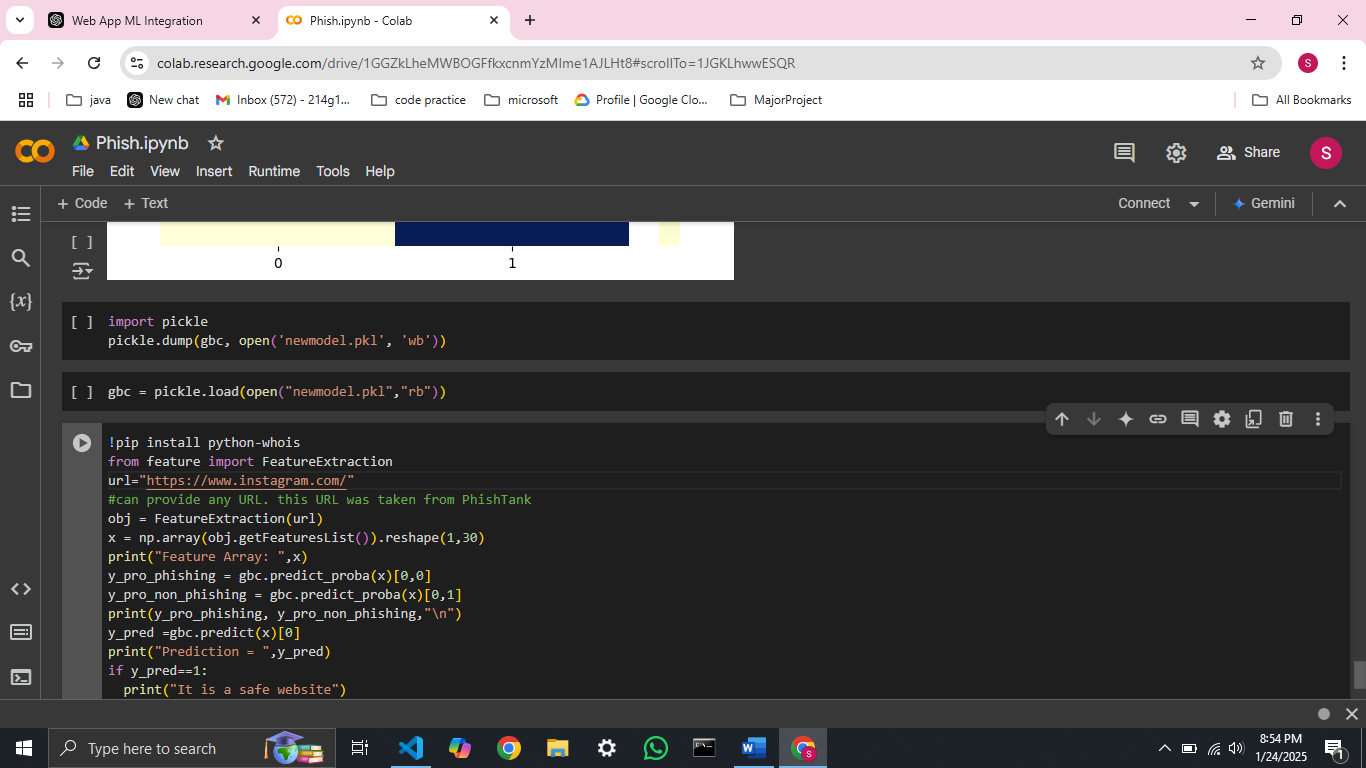
Table 2: Test results on the dataset

The table2 shows the performance of machine learning models for detecting phishing websites. Gradient Boosting had the highest accuracy (97.4%) and F1 score (0.974), followed by CatBoost (97.2%). Random Forest and Multi-layer Perceptron also performed well. Naive Bayes had the lowest accuracy, making it less effective **for this task.

*3.5 Web Application:*

*3.5.1 Machine Learning Integration:*

Integrating the machine learning model into the web application is crucial for real-time phishing website detection. This section explains how the trained Gradient Boosting Classifier (GBC) is incorporated into the backend system to make predictions based on user inputs. To deploy the machine learning model, we used the Python pickle library to serialize it. Serialization saves the trained model as a file, which can then be loaded into the web application to make predictions. The following code snippet shows how this is done

**

*3.5.2 Backend Implementation:*

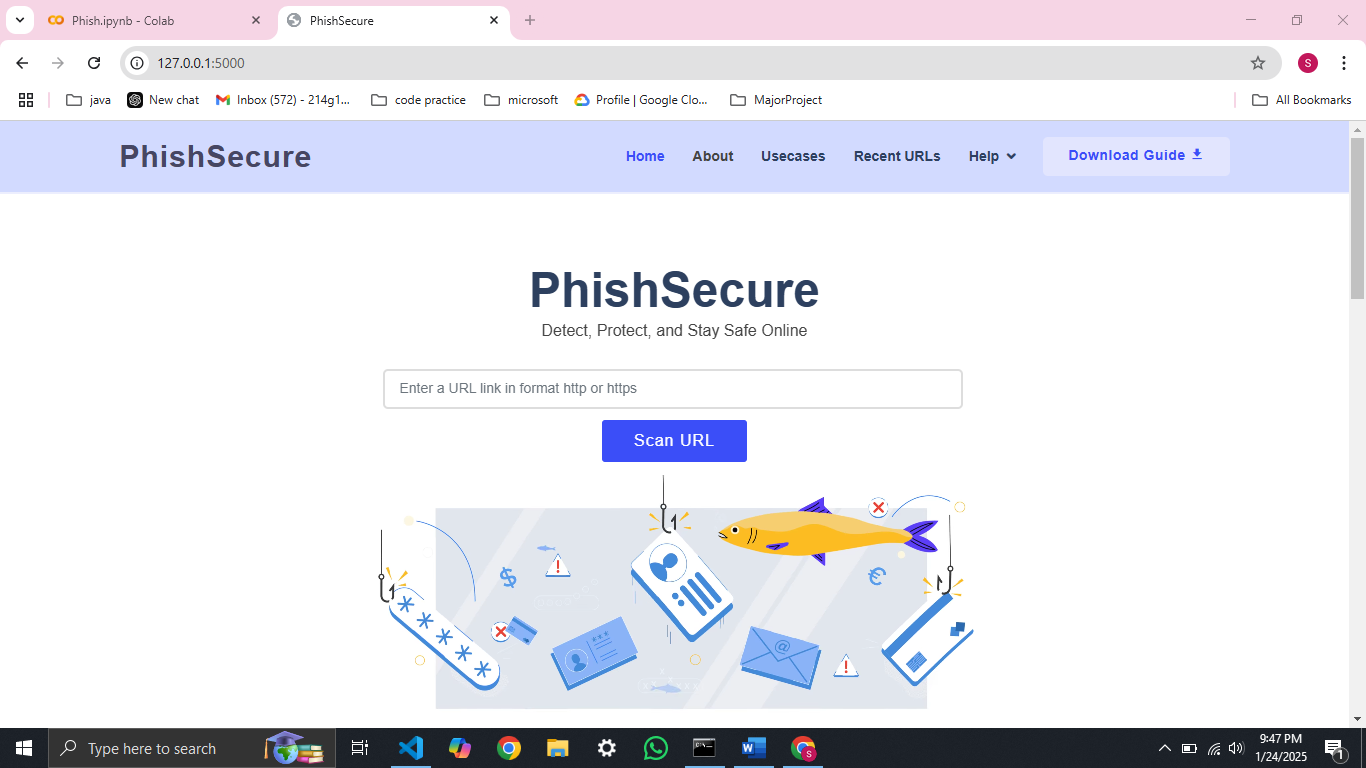
The backend is the core of the phishing detection web application, allowing smooth communication between the user interface and the machine learning model. We used Flask, a lightweight Python web framework, for the backend because it is simple, flexible, and supports quick development and integration with machine learning models.

The trained Gradient Boosting Classifier (GBC) model, saved using the pickle library, is loaded into the Flask application when it starts. This ensures the model is ready to process inputs and make real-time predictions.

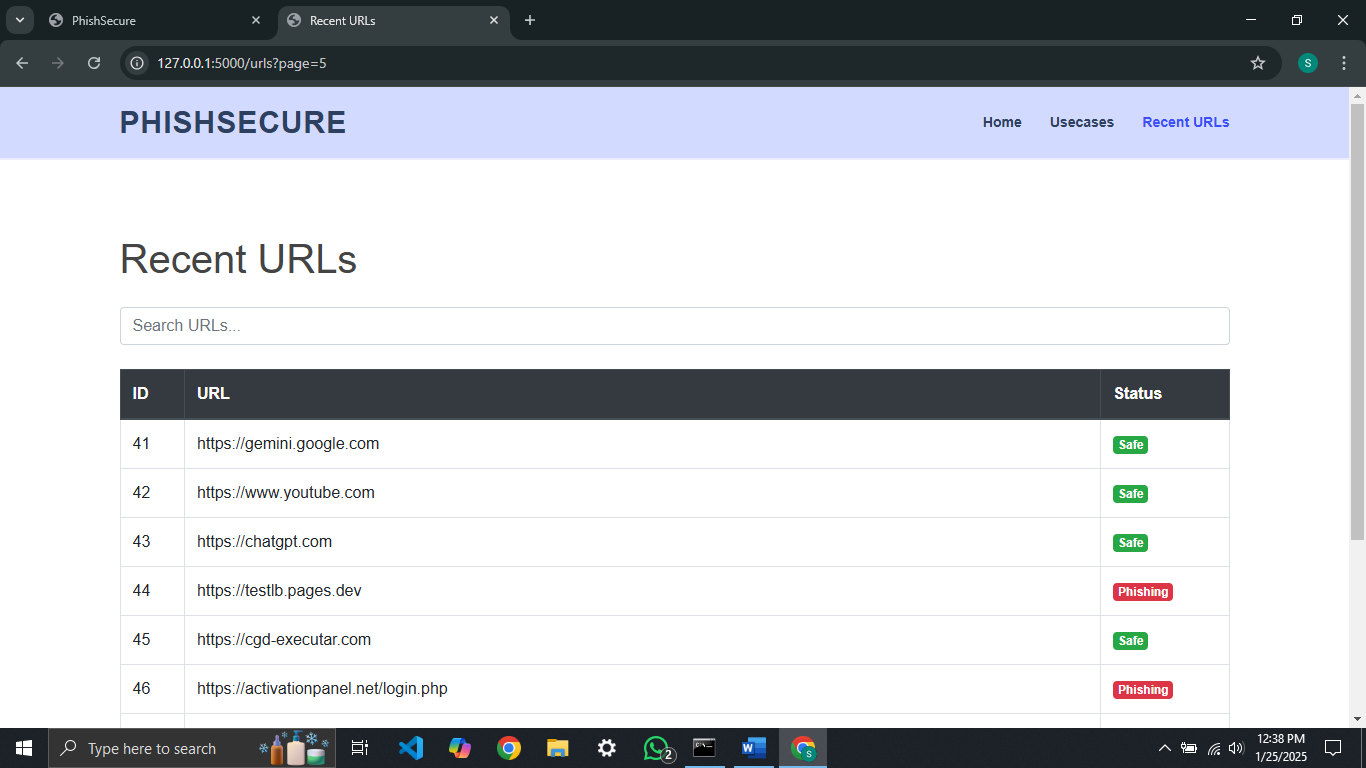
The backend of the phishing detection system has several routes to manage user interactions and predictions. The Home Route (/) shows the home page where users can submit URLs. The Prediction Route (/result) processes the submitted URL by extracting 30 features using the feature.py class. These features are then passed to the Gradient Boosting Classifier (GBC) model to predict if the URL is "Phishing" or "Legitimate." The results are displayed to the user and saved in the database. The Use Cases Route (/usecases) explains the application's uses, and the URLs Route (/urls) lists all analyzed URLs and their predictions. This setup ensures smooth user interaction and reliable record-keeping. The system starts by collecting URLs to check for phishing, which users can enter on a webpage. After gathering the URLs, the system extracts key information from the web pages. This information, known as features, is essential for training and testing the machine learning models. The features are derived from the domain, HTML content, and address bar of the URLs.

*3.5.3 Frontend Implementation:*

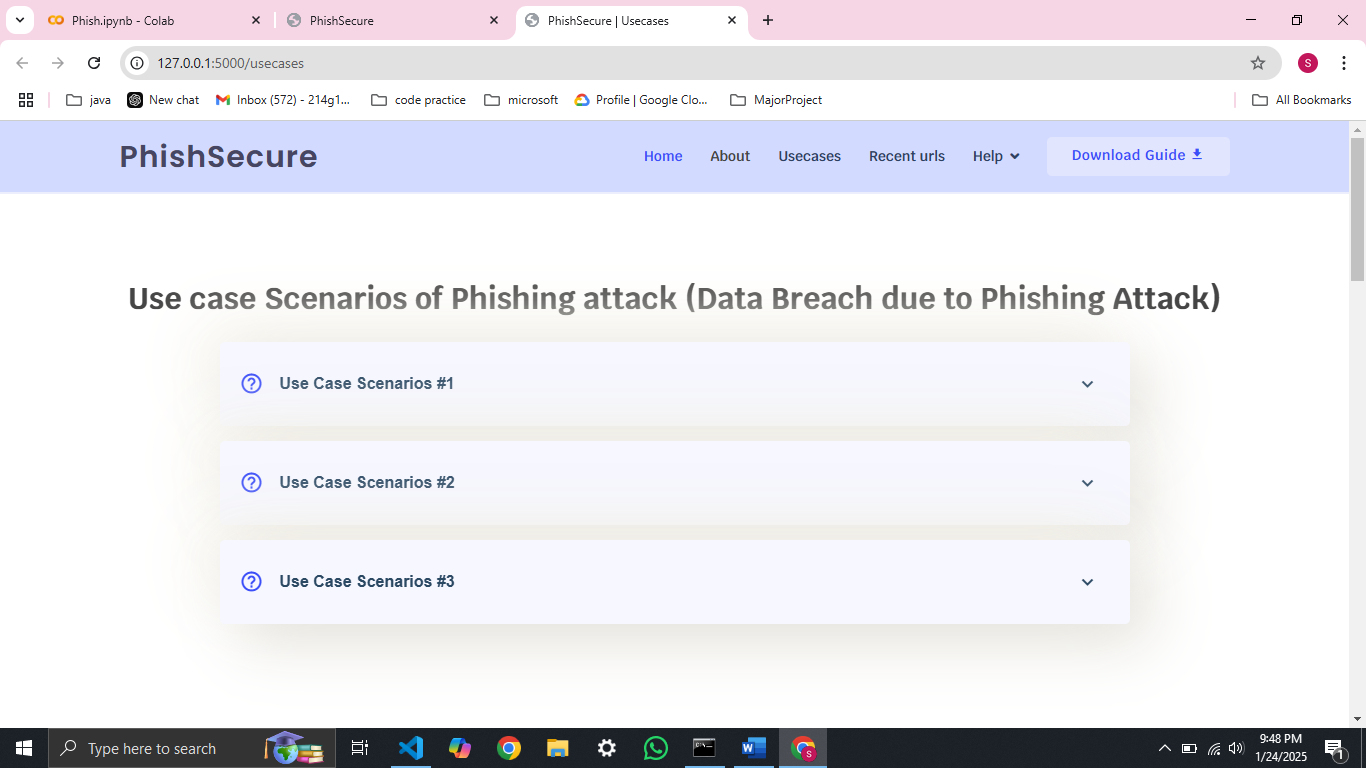
The frontend of the phishing detection system is designed to be user-friendly and interactive. Users can input URLs through a web page, which is built using HTML and rendered by Flask.



A simple form with a URL input field and a submit button for phishing analysis on the home page.



A list displaying all analyzed URLs along with their predictions, providing a record of past analyses*.*

**

A page describing various use cases of the phishing detection system to help users understand its practical applications*.*

*3.5.4 Database connectivity:*

Database connectivity is crucial for a phishing detection system as it ensures efficient storage, retrieval, and management of data. It allows storing analyzed URLs and their predictions, providing a persistent record for future reference. This setup supports user interaction by storing their submissions and predictions, enabling them to review past results. With a scalable database, the system can handle growing data volumes efficiently, ensuring quick data retrieval for both user-facing displays and backend processes.