Phishing Website Detection using ML [Report]

1. ABSTRACT

Phishing attacks are one of the most common cyber threats, where attackers create fake websites to deceive users and steal sensitive information. This project proposes a machine learning based detection system that classifies URLs as phishing or legitimate using URL-level, domain-level and content-based features. We extract lexical and structural features from URLs (e.g., URL length, presence of IP address, '@' symbol, redirects), domain and DNS features (domain age, DNS records) and web-traffic/hosting indicators. Multiple classifiers such as XGBoost, Random Forest and Logistic Regression are trained, tuned and compared. The final model demonstrates robust detection performance on real-world datasets and is packaged as a lightweight web API for real-time URL analysis.

2. INTRODUCTION

Phishing is one of the most common and dangerous cyber threats in the digital era. It involves creating fake websites that mimic legitimate ones to steal sensitive information such as usernames, passwords, and banking details. These attacks exploit user trust and often lead to identity theft and financial losses. This project aims to develop an automated Phishing Website Detection System using Machine Learning techniques to accurately detect malicious URLs. By analyzing structural, lexical, and domain-based features, and leveraging algorithms like XGBoost, Random Forest, and Logistic Regression, the system ensures high accuracy and real-time detection. The model is integrated into a web-based interface, offering instant classification and clear insights for users and analysts.

3. PURPOSE & SCOPE

PURPOSE :

The main purposes of the Phishing Website Detection system are:

1. **Real-Time Malicious URL Detection**
   System detects phishing URLs instantly using machine learning.
2. **Behavioural Pattern Recognition**
   Focuses on structural and lexical patterns which are difficult for attackers to hide.
3. **Automation & High Accuracy**
   Eliminates manual URL inspection.
4. **User Protection**
   Prevents credential theft by giving instant warnings.
5. **Lightweight & Fast Deployment**
   Can be used in browsers, security tools, and email systems.

SCOPE :

The system can be deployed in:

- Banks and financial institutions
- Email filtering systems
- SOC (Security Operations Center) tools
- Corporate cybersecurity solutions
- Antivirus and browser security extensions
- Mobile and desktop-based URL scanners

4. SYSTEM REQUIREMENTS

Hardware Requirements:

• Processor: Intel i5 or above

• RAM: 8 GB or higher

• Storage: Minimum 50 GB

Software Requirements:

• Operating System: Windows / Linux

• Programming Language: Python 3.8+

• Libraries: scikit-learn, xgboost, pandas, numpy, BeautifulSoup4, requests

• Tools: Jupyter Notebook, Flask, Git

5. DETAILED METHODOLOGY

The development of the *Phishing Website Detection System using Machine Learning* follows a structured sequence of steps. Each step ensures that the final model is reliable, accurate, and suitable for real-time detection. The complete methodology used in the project is described below.

1. Data Collection

The first step involved gathering phishing and legitimate URLs from trusted online sources. Phishing URLs were taken from **Phish Tank** and **Kaggle datasets**, which regularly publish verified phishing links.
Legitimate URLs were collected from **Alexa Top Websites** and other trusted domains. This created a balanced dataset containing both safe and malicious links, which is essential for accurate model training.

2. Data Preprocessing

Raw URLs are often noisy and unstructured, so preprocessing was done before training:

- Removing duplicate entries
- Eliminating empty or corrupt URL rows

- Converting all URLs to lowercase
- Standardizing labels:
    - **1 → phishing**
    - **0 → legitimate**

After preprocessing, the dataset became clean, uniform, and ready for feature extraction.

3. URL Parsing

Each URL was broken into structured components:

- Protocol (http/https)
- Domain name
- Subdomain
- Path
- Query parameters

This step is important because many phishing characteristics appear inside these URL segments.

6. Feature Engineering

Feature engineering is the most important part of phishing detection.
In this project, features were created from the URL's structure, length, characters, and domain details.

The main groups of features are:

a. Lexical Features (based on characters of URL)

These features study how the URL *looks*:

- URL length
- Number of dots (.)
- Number of digits
- Number of special characters (@, -, _, ?, etc.)
- Presence of prefix or suffix (-) in the domain
- Depth of the URL (number of path segments)

Phishing URLs often contain many unusual symbols and long confusing structures.

b. Structural Features

These features describe the internal structure:

- Number of redirections (//)
- Number of parameters in the URL
- Use of URL shorteners (bit.ly, tinyurl, etc.)

Phishing websites use redirections and shortened URLs to hide the actual destination.

c. Domain-Based Features

These features study domain identity:

- Whether URL uses an **IP address** instead of a domain
- Whether DNS records exist
- Age of the domain (phishing domains are usually very new)

d. Security/Protocol Features

- Whether HTTPS is used
- Whether SSL certificate is present

Although HTTPS alone does NOT guarantee safety, phishing links often avoid it.

7. Model Training

After feature extraction, three machine learning models were trained and compared:

- **Logistic Regression**
- **Random Forest**
- **XGBoost**

  XGBoost performed the best because it handles complex feature patterns and provides higher accuracy.

  Training involved:

- Splitting dataset into training and testing sets
- Running hyperparameter tuning
- Checking performance with accuracy, precision, recall, and F1-score

8. Design Analysis

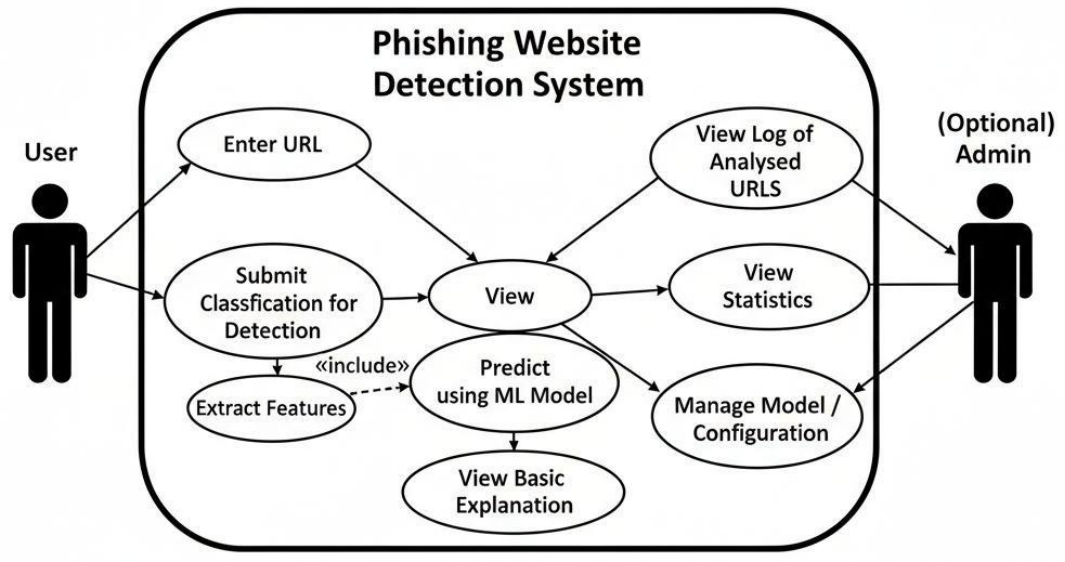
8.1 Use Case Diagram

The Use Case Diagram shows how different users interact with the Phishing Website Detection System. It highlights the main functionalities offered to the user and explains what actions take place inside the system.

This diagram helps in understanding what operations the system provides from a user's point of view.

Key Points:

- User enters a URL that needs to be analysed.
- System validates the URL format.
- System extracts features internally.
- Machine Learning model predicts whether the URL is *Phishing* or *Legitimate*.
- User receives classification result and a simple explanation.
- (Optional) Admin can view logs and system statistics.

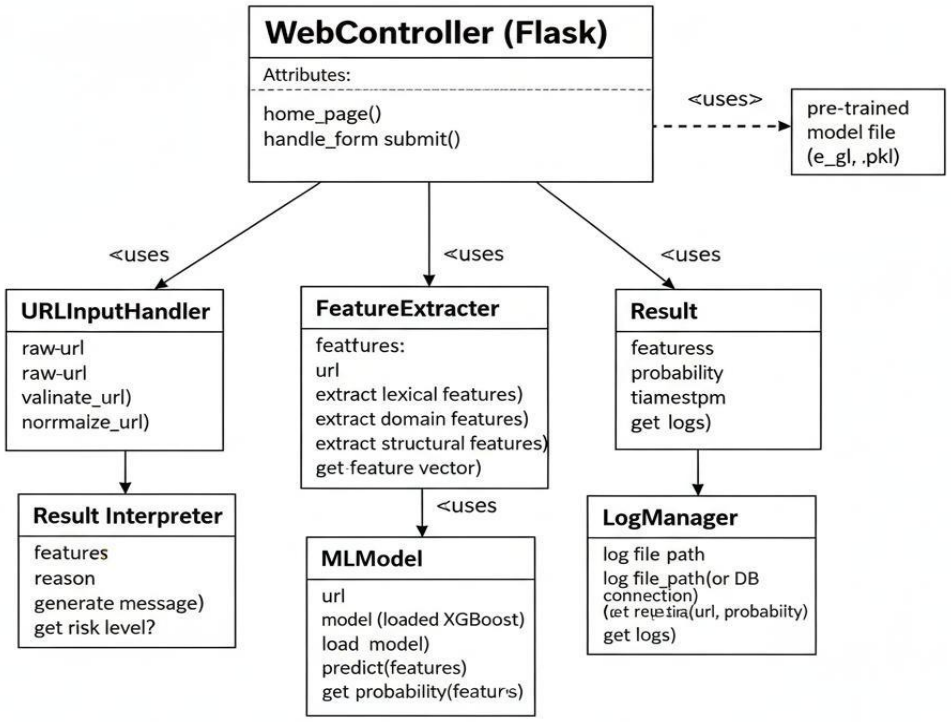


8.2 Class Diagram

The Class Diagram represents the internal structure of the system. It shows the major classes used in the backend, their attributes, methods, and how they interact with one another.

This design helps maintain modularity and clean separation between functionalities.

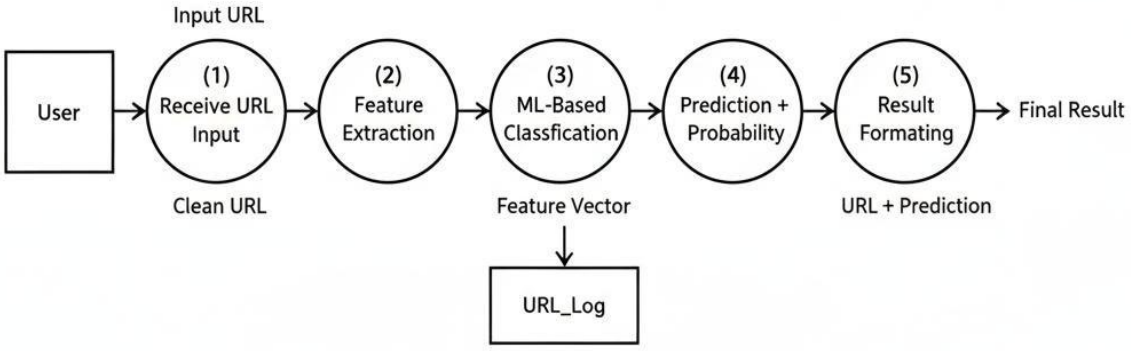
- **URLInputHandler** – Handles, validates, and cleans the input URL.
- **FeatureExtractor** – Generates all lexical, structural, and domain-based features.
- **MLModel** – Loads the trained XGBoost model and performs prediction.
- **ResultInterpreter** – Converts the model output into a user-friendly message.
- **LogManager** – Records URL, prediction result, and timestamp into logs.
- **WebController (Flask)** – Manages user requests &coordinates all backend component.



8.3 DFD Level 1

Level 1 DFD breaks the main process into smaller meaningful steps.
It shows how the URL flows through multiple internal components before producing the final output.

- Receive URL Input – Takes the raw URL entered by the user.
- Feature Extraction – Converts the URL into a numerical feature vector.
- ML-Based Classification – Applies the XGBoost model to predict phishing or legitimate.
- Result Formatting – Generates the final user-friendly output message.
- Logging – Saves the URL, prediction result, and timestamp for records.



9. System Implementation and Output

This section shows how the Phishing Website Detection System works in real life.
The following screenshots explain each stage—from running the backend server to checking URLs and getting the final prediction.
These outputs help to understand how the project behaves practically, beyond theory.

9.1 Flask Server Running (Backend Started)

This screenshot shows the project backend running in the Parrot OS terminal.

Here the virtual environment is activated, and the Flask app is started using

`python3 app.py`.

Once it runs, the application becomes available at:

`http://127.0.0.1:5000/`

This confirms that the system is ready to accept user requests.

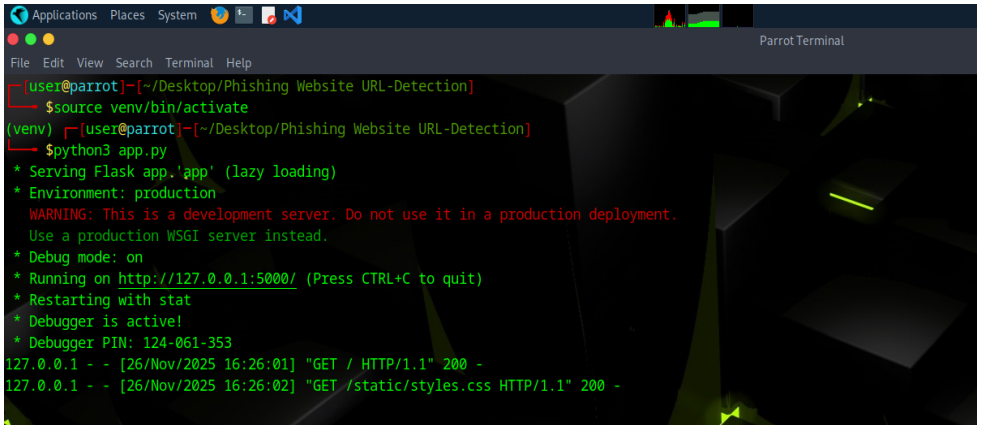


Figure 9.1: Flask Server Running

9.2 Homepage – URL Input Page

This is the main interface of the system where any URL can be pasted for checking.
The page gives a small description about phishing websites and guides the user on how to check if a website is safe or not.
This is the first thing users see when they open the tool in their browser.

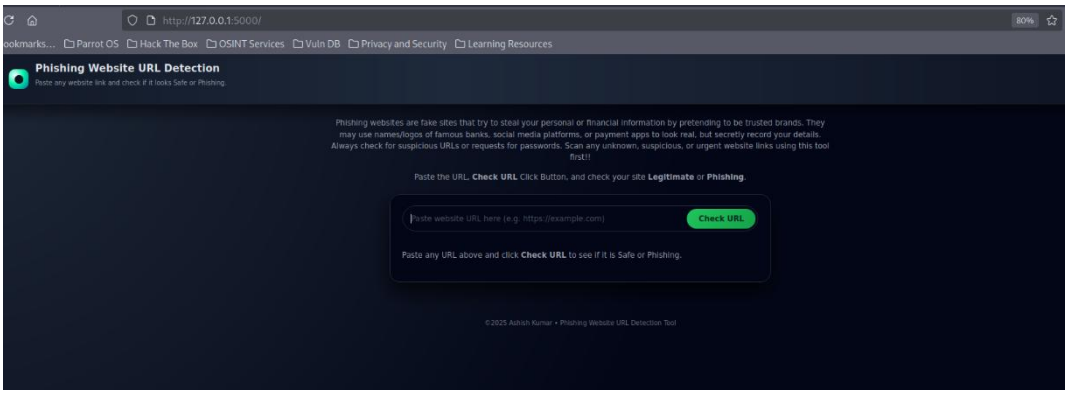


Figure 9.2: Homepage of the Application

9.3 User Enters a URL for Checking

In this step, the user enters a website link (e.g., `https://google.com`) into the input box. When the user clicks on the **Check URL** button, the backend processes the URL, extracts features, and sends it to the ML model.

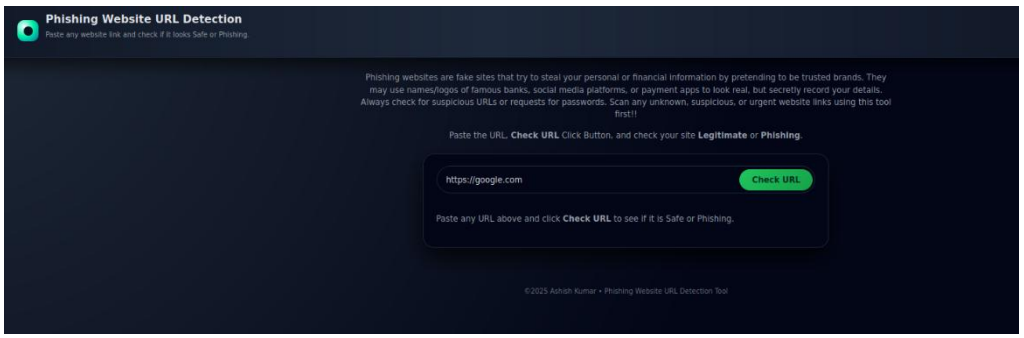


Figure 9.3: URL Entered Before Checking

9.4 Output – Legitimate Website Detected

After checking the URL, the model returns the result.
In this case, the system correctly identifies **google.com** as a safe and legitimate website.
The output clearly displays the scanned URL and the prediction result.

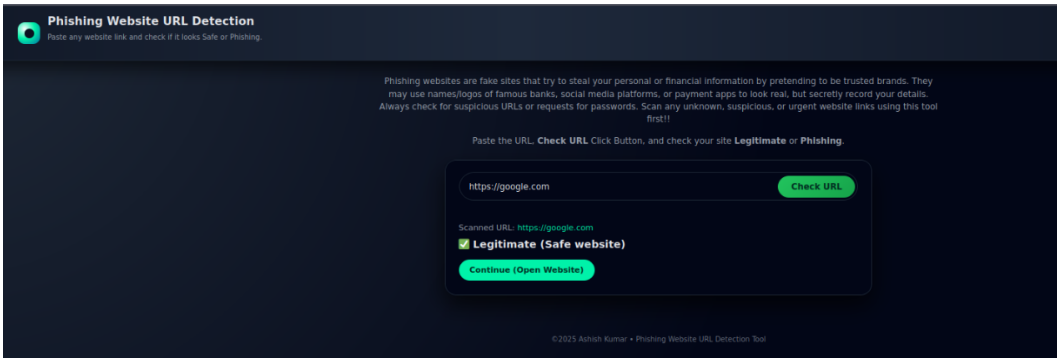


Figure 9.4: Legitimate (Safe) Website Result

9.5 Output – Phishing Website Detected

This screenshot shows the system detecting a suspicious URL like `https://paypa1.com`, which is a fake version of PayPal where "L" is replaced with "1".
The system correctly marks it as a phishing website and warns the user.

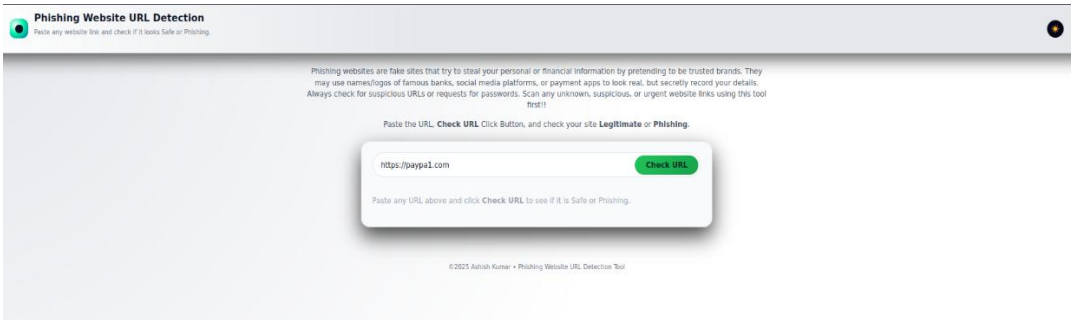
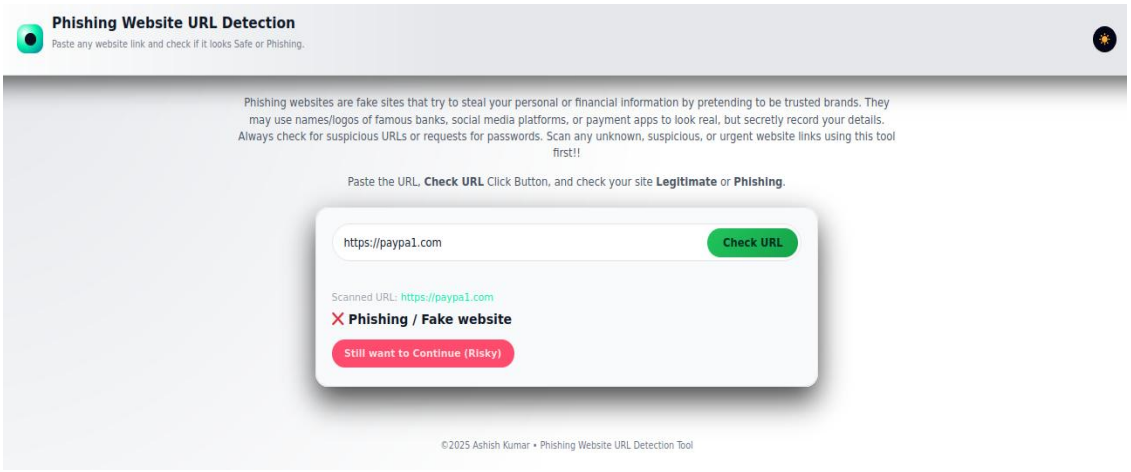


Figure 7.5: Phishing Website Result

After detecting the phishing URL, the system clearly displays a red warning message indicating that the website is unsafe. This final output confirms that the machine learning model is able to identify suspicious or look-alike domains that are commonly used in phishing attacks.



The interface also shows an optional “Continue (Risky)” button, which informs the user that proceeding to such websites may expose them to security threats.

10. RESULTS

The machine learning models were tested on the prepared dataset, and among all, the **XGBoost classifier delivered the best overall performance**.
It achieved high accuracy, precision, and recall, showing that the feature engineering approach used in the project was effective.
The system was also tested with real URLs, and it correctly identified most phishing and legitimate sites in real time.
The Flask-based interface responded quickly and provided clear output, proving that the

model is suitable for practical use.
Overall, the results demonstrate that the proposed ML-based system is both reliable and efficient for phishing website detection.

11. CONCLUSION

The project successfully implemented a machine-learning-driven phishing detection system based on URL features.
By extracting lexical, structural, and domain-level features and applying the XGBoost model, the system achieved strong predictive performance.
The solution is lightweight, fast, and capable of classifying URLs without requiring full website content, making it highly suitable for real-time security applications.
This project confirms that machine learning is an effective approach for reducing phishing risks, and the system can be further enhanced with browser integration, API deployment, or advanced content analysis in future word.

12. REFERENCES

1. PhishTank – Public Phishing URL Database
2. Kaggle – Phishing Website Datasets
3. Alexa Top Sites – Trusted Legitimate Domain Source
4. Scikit-learn Documentation – Machine Learning Tools
5. XGBoost Documentation – Gradient Boosting Framework
6. Flask Documentation – Web Application Framework
7. Fette, N., Sadeh, N., Tomasic, A. “Learning to Detect Phishing Websites,” WWW Conference
8. WHOIS Database – Domain Information Source