Phishing URL Detection Using Machine Learning Approach

Mukthar Babu Shaik
University of New Hampshire at Manchester
400 Commercial Street
Manchester, NH 03101
603-717-4998
Muktharbabu.Shaik@unh.edu

ABSTRACT
Phishing attacks pose a significant threat to internet users by deceiving them into divulging sensitive information. This paper presents a machine learning-based approach for detecting phishing URLs, leveraging a Gradient Boosting Classifier and a feature extraction module to evaluate 30 URL attributes. The system, implemented as a browser extension, provides instant feedback on website legitimacy while highlighting risky features. Testing on a dataset of 450,176 URLs with 30 features achieved high accuracy, demonstrating the system's effectiveness and scalability. This project marks a step forward in combating phishing attacks through a transparent, user-friendly solution.

Keywords
Phishing, Machine Learning, Malicious website, URL, Classification, legitimate

1. INTRODUCTION
In recent years, phishing attacks have become one of the most common and serious cyberthreats. These attacks deceive unsuspecting users into revealing sensitive information such as login credentials, financial data, or personal identification details by following legitimate websites. Phishing attacks can have severe consequences for both individuals and companies, including identity theft, financial losses, and data breaches.

Blocklists and heuristic-based systems, two conventional methods of preventing phishing attacks, are highly ineffective. Because blocklists are based on precompiled databases of known phishing URLs, they cannot quickly adjust to the constantly changing strategies used by attackers [1]. In the same way, heuristic-based techniques frequently fail to apply to various phishing strategies, leading to low accuracy and a high false positive rate.

This project suggests a machine learning-powered real-time phishing URL detection system to overcome the drawbacks of conventional techniques. The solution uses an efficient feature extraction module combined with a Gradient Boosting Classifier that was trained on a huge set of real and malicious URLs. The technology efficiently detects phishing URLs and gives users immediate feedback on their validity by examining 30 structural, behavioral, and security-related characteristics.

The integration of real-time detection, transparency, and a user-centric design makes this project a significant step forward in combating phishing attacks. It not only enhances the accuracy of detection but also provides an intuitive solution that can be seamlessly integrated into users' daily online activities.

1.1 What is phishing?
Phishing is one of the most common ways of obtaining personal data. To minimize the damage from a phishing attack, detecting it as early as possible is necessary. Almost every type of phishing attack uses phishing URLs. Phishing URLs are links to websites or web pages designed to look like legitimate websites. Still, they are malicious sites created by cyber attackers to steal personal information such as login credentials, credit card numbers, and other sensitive data [3].

1.2 Machine Learning
Machine learning is an artificial intelligence technique in which computer algorithms are trained based on large amounts of data. In the case of the phishing URL detection question, machine learning can be used to detect suspicious patterns in link addresses that may indicate phishing attacks.

Phishing URL detection using machine learning analyzes large amounts of data, including various features such as the URL, the web page's appearance, the context, and so on. Machine learning models that are used to detect phishing URLs can be trained on real examples of phishing sites and sites that are not phishing, allowing them to identify malicious links based on the trained model. Thus, using machine learning to detect phishing URLs can be an effective method to protect users from phishing-related cyberattacks.

1.3 Phishing Detection Methods
To combat phishing attacks, several methods have been developed to detect phishing URLs. These methods include:

1.3.1 Blocklists
Blocklists contain lists of known phishing URLs identified by security experts. These lists can be used by browsers, email providers, and security software to block users from accessing known phishing websites. Many popular brands, including Google, Microsoft, Apple, and many others, use blocklisting as a tool to protect against phishing URLs. These companies use various methods to maintain and update their blocklists, such as automated crawlers and user reports.

For example, Google's Safe Browsing service maintains a constantly updated list of unsafe websites, including those involved in phishing attacks, and warns users before visiting them. Microsoft's SmartScreen filter, built into the Edge and Internet Explorer web browsers, also uses blacklisting to protect users from potentially harmful websites.

1.3.2 DNS filters

Domain Name System filters can be used to block access to known phishing URLs. When a user attempts to access a known phishing website, the DNS filter redirects the user to a safe page or block access to the site altogether.

Several popular brands, including Cisco, Barracuda Networks, Sophos, McAfee, and Symantec, use DNS filtering to protect against phishing URLs. These companies provide DNS filtering services that can help organizations block access to known phishing sites and malicious IP addresses and domains. Organizations can proactively protect their networks and users from phishing attacks by leveraging these services

1.3.3. Machine Learning Algorithms

Machine learning algorithms can detect phishing URLs by analyzing the characteristics of the URL, such as the domain name, the URL length, and the presence of specific keywords. Certain keywords. Such algorithms can also detect similarities between phishing URLs and known phishing websites. Machine learning is a more practical approach to detecting phishing URLs than blocklisting or DNS filtering because it can adapt to new and evolving threats. Blocklisting and DNS filtering rely on maintaining lists of known malicious URLs or domains, which can quickly become outdated as attackers create new URLs or domains.

Additionally, machine learning can analyze various features beyond just the URL or domain, such as the appearance of the web page and the context in which the link is presented. This makes it more difficult for attackers to circumvent detection using different URLs or domains. Overall, machine learning is a more proactive and adaptive approach to detecting phishing URLs, making it a better tool for protecting against evolving cyber threats.

To conclude, detecting phishing URLs is essential to preventing phishing attacks. Several methods have been developed to detect phishing URLs, including blocklists, DNS filters, machine learning algorithms, and user awareness training. These methods can help individuals and organizations stay safe from phishing attacks and protect their sensitive information.

2. METHODOLOGY

The methodology of this project revolves around the seamless integration of a machine learning model with a browser extension to provide real-time phishing URL detection. The system begins by accepting user-input URLs through a browser extension, which serves as the user interface for interaction. These URLs are then Sent to a Flask backend server, where a feature extraction module. Analyzes various structural, behavioral, and security-related attributes, such as the presence of HTTPS, URL length, domain age, and DNS records.

The extracted features are processed by a Gradient Boosting Classifier, which predicts whether the URL is phishing or legitimate. This prediction and the features contributing to the classification are sent back to the browser extension, providing users with immediate feedback and transparency. The process is designed for scalability and real-time performance, ensuring practical applicability in detecting phishing attempts during daily browsing activities.

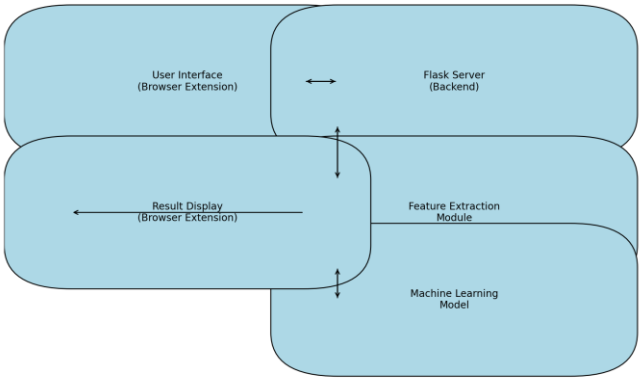


Fig 1: Architecture

2.1 Dataset

The dataset's source is Kaggle, where we can find multiple datasets related to phishing [11]. The service provides various data formats, such as CSV, JSON, and many more, and it is updated hourly. This dataset has 450,176 URLs, including 104,000 malicious URL data and 345,000 00 Non malicious URL data. From this, a collection of 11000+ website URLs with samples of 30 website parameters and a class label identifying it as a phishing website (1 or -1) are used for training.

The next step after collecting the dataset is:

2.1.1 Data preprocessing

Data preprocessing, including merging the data, is a major challenge when attempting to add a dataset to the machine learning model. Because of this, all null values are removed before adding the dataset to the machine learning model.

2.1.2 Extract the Features

Feature extraction, in this process, lexical domain-based features from the final dataset are extracted using Python modules such as urlparse and whois [6] [7].

2.1.3 Apply ML model

Finally, we apply the machine learning model to all the features generated by the feature extraction module, using machine learning algorithms such as the Random Forest Classifier, Decision Tree, and GBC algorithms.

2.2 Feature Extraction

Feature extraction [6][7][10] is the process of identifying and quantifying relevant characteristics or patterns in raw data to make it understandable for machine learning models. The main purpose of this feature extraction is to convert raw URL data into numerical features that the machine learning model can use for classification. These features ensure the model can accurately predict whether a URL is phishing or legitimate.

We have implemented a python program to extract features from URLs. Below are the features we have extracted to detect phishing URLs.

- **Presence of IP address in URL:** If the IP address is present in the header, it is set to 1se set-1. Most benign sites do not use an IP address as a URL to download a webpage. Using an IP address in the URL indicates that the attacker is trying to steal sensitive information.

- **Number of dots in Hostname:** Phishing URLs have many dots in the URL. For example, http://shop.fun.amazon.phishing.com, in this URL, phishing.com is an actual domain name, whereas the "amazon" word is used to trick users into clicking on it. The average number of dots in benign URLs is 3. If the number of dots in URLs exceeds 3, the feature is set to 1; otherwise, to -1.
- **Prefix or Suffix separated by (-) to the domain:** If the domain name is separated by a dash (-) symbol, the feature is set to 1. Else, to -1. The dash symbol is rarely used in legitimate URLs. Phishers add a dash symbol (-) to the domain name so that users feel they are dealing with a legitimate webpage. For example, the Actual site is http://www.onlineamazon.com, but phishers can create another fake website like http://www.online-amazon.com to confuse the users.
- **URL redirection:** If "//" is present in the URL path, the feature is set to 1 or -1. The existence of "//" within the URL path means that the user will be redirected to another website.
- **HTTPS token in URL:** If an HTTPS token is present in the URL, the feature is set to 1; otherwise, it is set to -1. Phishers may add the "HTTPS" token to the domain part of a URL to trick users.
- **Information submission to Email:** A phisher might use "mail()" or "mailto:" functions to redirect the user's information to his email. If such functions are present in the URL, the feature is set to 1; otherwise, it is set to -1.
- **URL Shortening Services "TinyURL":** TinyURL service allows phishers to hide long phishing URLs by making them short. The goal is to redirect users to phishing websites. If the URL is crafted using shortening services (like bit.ly), then the feature is set to 1 else –1
- **Length of Hostname:** The average length of the benign URLs is 25; if the URL's length is greater than 25, the feature is set to 1. Else, to –1
- **Presence of Unicode in URL:** Phishers can use Unicode characters to trick users into clicking on it. For example, the domain "xn--80mb6sk92e.com" is equivalent to "apple.com." The visible URL to the user is "apple.com," but after clicking on this URL, the user will visit "xn--80mb6sk92e.com," which is a phishing site.
- **Age of SSL Certificate:** The existence of HTTPS is significant in giving the impression of website legitimacy. However, the minimum age of the SSL certificate of a benign website is between 1 year and 2 years.
- **URL of Anchor:** We extracted this feature by crawling the URL's source code. A tag defines the URL of the anchor. If the tag has a maximum number of hyperlinks from the other domain, then the feature is set to 1; otherwise, it is set to -1.
- **Iframe:** We have extracted this feature by crawling the URL's source code. This tag is used to add another web page to the existing main webpage. Phishers can use the "iframe" tag and make it invisible, i.e., without frame borders. Since the border of the inserted webpage is invisible, the user thinks that the inserted webpage is also part of the leading web page and can enter sensitive information.
- **Website Rank:** We extracted the rank of websites and compared it with the first one hundred thousand websites in the Alexa database. If the rank of the website is greater than 10,0000, then the feature is set to 1; otherwise, it is set to -1.
- **Presence of @ symbol in URL:** If the @ symbol is present in the URL, the feature is set to 1. Else set to -1. Phishers add a special symbol @ in the URL, which leads the browser to ignore everything preceding the "@" symbol, and the actual address often follows the "@" symbol.
- **Number of slashes in URL:** The number of slashes in benign URLs is found to be 5; if the number of slashes in the URL is greater than 5, then the feature is set to 1; otherwise, it is set to -1.
- **Presence of sensitive words in URL:** Phishing sites use sensitive words in their URLs so that users feel that they are dealing with a legitimate webpage. Below are the words that are found in many phishing URLs: - 'confirm,' 'account,' 'banking,' 'secure,' 'ebyisapi,' 'webscr', 'signin', 'mail,' 'install,' 'toolbar,' 'backup,' 'PayPal,' 'password,' 'username,' etc.;
- **DNS Record:** WHOIS is a registry containing domain name information, such as registration and contact details. If there is no DNS record, then the value assigned to this feature is set to 1; otherwise, it is set to –1.
- **Sub-domain:** Whenever the "." count in the URL is greater than 3, that website is malicious, and the value assigned to it is 1 or -1.
- **Google Indexing:** Verifies if the site is indexed by Google, as phishing sites are often not indexed.

These are all the dataset features that were extracted using Python by using the language. These features are related to Domain-based, HTML and JavaScript-based, and Address-based features.

In Fig.2 we can see the histogram Visualization of the data set and also how the data is distributed of all the features.

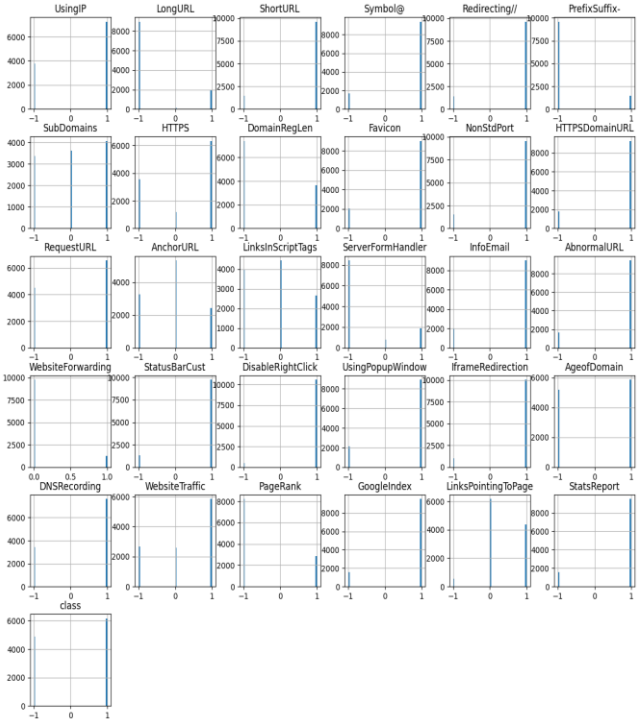


Fig 2: Data Distribution

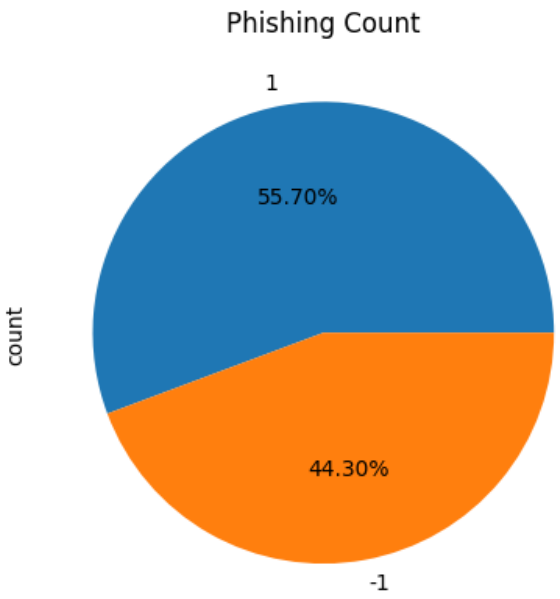


Fig 3: Count Distribution

Fig. 4 represents a correlation heat map that shows how the data set's features are related to each other.

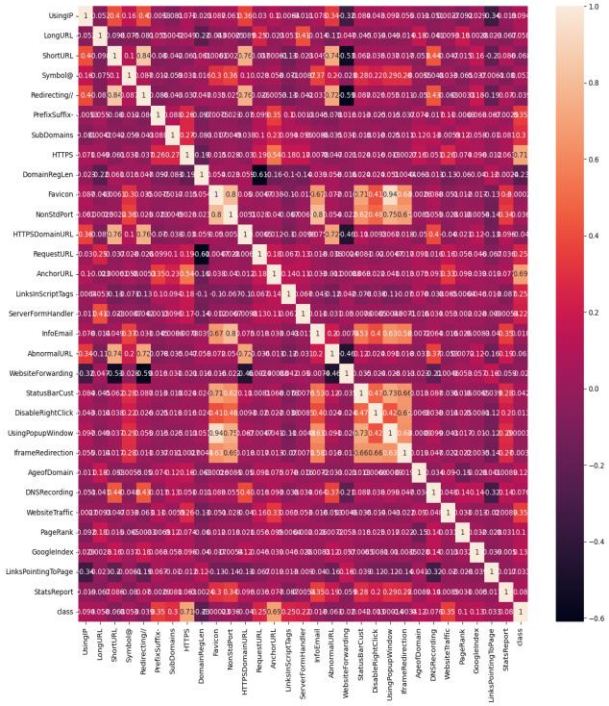


Fig 4: Correlation Heat Map

2.3 Machine Learning Algorithm

The proposed approach deals with supervised machine learning algorithms. Since it is a classification problem, we can use various classifiers and decide which one is best based on the accuracy of a model.

2.3.1 Decision Tree Algorithm

A decision tree [8] is a supervised machine-learning technique that can solve regression and classification problems. A decision tree is a flow chart-like tree structure; sorting them based on the attribute value classifies instances. At the ending nodes of a decision tree, a decision is made. They are the leaves of the tree. Each node in the tree represents a feature in a classification instance. The decisions or tests are made based on the characteristics of the given data collection. Every branch denotes an output of the test; each leaf node holds the class label. Instances are classified from starting based on the value of a feature. For the classification of the data set, it generates the rule.

2.3.2 Random Forest Classifier

Random Forest [2] is based on ensemble learning, a method for solving a complex problem and improving the model's performance by mixing multiple classifiers. It's a classifier that uses numerous decision trees on different subsets of a dataset and averages the results to increase the dataset's predicted accuracy. Instead of relying on a single decision tree, the random forest collects the forecasts from every tree and predicts the output data based on most predictions' votes. The more trees in a random forest, the greater the accuracy and the less chance of overfitting.

2.3.3 Support Vector Machine Algorithm

Support vector machine [8] is another robust algorithm in machine learning technology. In this algorithm, each data item is plotted as a point in n-dimensional space, and the algorithm constructs a separating line for classifying two classes; this separating line is well known as a hyperplane.

The support vector machine seeks the closest points, which are called support vectors, and once it finds the nearest point, it draws a line connecting to them. The support vector machine then constructs a separating line that bisects and is perpendicular to the connecting line. To classify data perfectly, the margin should be maximum. Here, the margin is the distance between the hyperplane and support vectors. In real scenarios, separating complex and nonlinear data is impossible. To solve this problem, a support vector machine uses a kernel trick, transforming lower dimensional space to higher dimensional space.

2.3.4 CatBoost Classifier

CatBoost (Categorical Boosting) is a high-performance, open-source gradient boosting algorithm that handles categorical features and minimizes overfitting. It builds decision trees sequentially, with each new tree correcting the errors of the previous ones. CatBoost uses efficient techniques like ordered boosting to prevent overfitting and target leakage. Unlike traditional gradient boosting, CatBoost integrates categorical features encoding directly into the algorithm, making it particularly effective for datasets with categorical variables. It is designed to provide superior accuracy and speed, often outperforming other gradient-boosting algorithms such as XGBoost and LightGBM in scenarios with mixed data types or high-cardinality categorical features. The ability to generalize across various datasets makes CatBoost a versatile choice for classification tasks.

2.3.5 Gradient Boosting Classifier

Gradient Boosting Classifier is a robust ensemble learning algorithm that builds a strong predictive model by combining the

outputs of multiple weak learners, typically decision trees. It works by iteratively training new models to correct the errors of the existing ones, minimizing the loss function using gradient descent. Each tree in the ensemble focuses on reducing the residual errors of its predecessors, leading to improved accuracy. Gradient Boosting is known for its flexibility, as it allows customization of loss functions and supports regression and classification tasks. While it can achieve high accuracy, careful tuning of hyperparameters is required to avoid overfitting and ensure optimal performance. Its ability to capture complex relationships in data makes it a popular choice for machine learning tasks.

3. Implementation and Results

The feature-extracted data set is divided into training and testing data in a ratio of 80-20. This training data is used to train the algorithms, and the testing data is used to test and determine the algorithm's accuracy.

In this study, we evaluated the performance of ten machine learning classifiers to identify the most effective model for phishing URL detection. The classifiers tested include Logistic Regression, k-nearest Neighbors (k-NN), Support Vector Classifier (SVC), Naive Bayes, Decision Tree, Random Forest, Gradient Boosting, CatBoost, XGBoost, and Multilayer Perceptrons (MLP).

After thorough experimentation and comparison, we selected five classifiers for detailed analysis: Decision Tree, Random Forest, Gradient Boosting Classifier (CGB), CatBoost, and Support Vector Classifier (SVC). These models demonstrated the highest accuracy among all the classifiers, with the Gradient Boosting Classifier achieving the best accuracy of **0.974**, surpassing all others. This result highlights the capability of ensemble methods like Gradient Boosting and CatBoost to effectively capture complex patterns in phishing and legitimate URLs, making them ideal choices for this problem.

Gradient Boosting Classifier Classification Report

	Precision	Recall	F1-Score	Support
Class -1 (Phishing)	0.99	0.96	0.97	976.0
Class 1 (Legitimate)	0.97	0.99	0.98	1235.0
Overall	0.97	0.97	0.97	2211.0

Fig 5: Gradient Boosting Classifier metrics

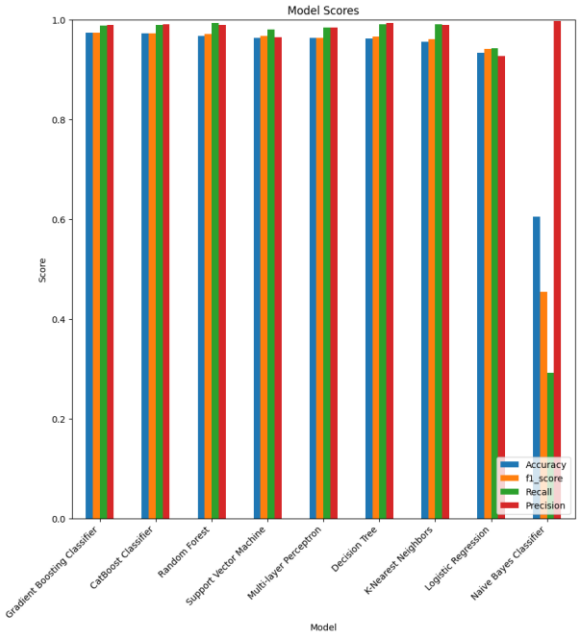


Fig 6: Comparison of Models

The above figure represents the Model Comparision of all algorithms

Accuracy of various model used for URL detection

	ML Model	Accuracy	f1_score	Recall	Precision
0	Gradient Boosting Classifier	0.974	0.977	0.994	0.986
1	CatBoost Classifier	0.972	0.975	0.994	0.989
2	Multi-layer Perceptron	0.969	0.973	0.995	0.981
3	Random Forest	0.967	0.971	0.993	0.990
4	Support Vector Machine	0.964	0.968	0.980	0.965
5	Decision Tree	0.960	0.964	0.991	0.993
6	K-Nearest Neighbors	0.956	0.961	0.991	0.989
7	Logistic Regression	0.934	0.941	0.943	0.927
8	Naive Bayes Classifier	0.605	0.454	0.292	0.997

Fig 7: Accuracy score of all algorithms

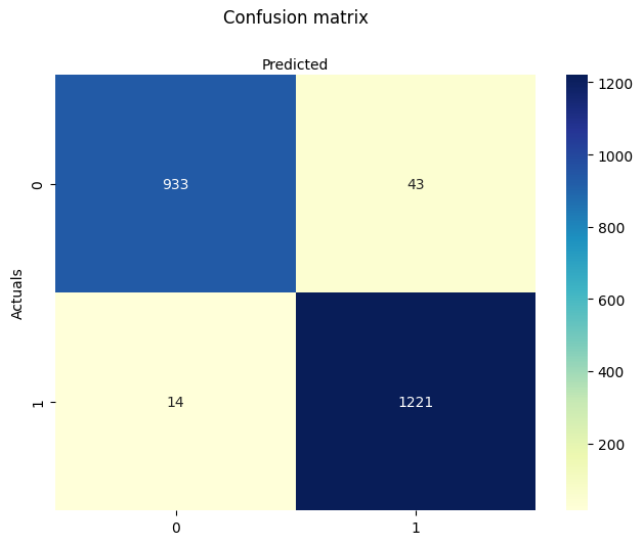


Fig 8: Confusion Matrix

Figure 8 represents the confusion matrix for the Gradient Boosting Classifier

4. MODEL DEPLOYMENT

The above classification report and the confusion matrix clearly show that the Gradient Boosting Classifier (GBC) demonstrates higher accuracy compared to all other algorithms. Consequently, this GBC model is selected for deployment and stored using the Python pickle library for future use. The deployment is carried out through two main components: a web-based interface and a browser extension, both powered by a Flask API.

The web-based interface is designed as a user-friendly webpage that allows users to input URLs for phishing detection. It contains a textbox and a submit button, developed using Hyper Text Markup Language (HTML). Upon submission, the entered URL is sent to the backend for processing, where the GBC model predicts whether the URL is legitimate or phishing. The prediction results and contributing features are then displayed on the webpage.

In addition to the webpage, a browser extension is developed to enhance real-time phishing detection capabilities. The browser extension is implemented using HTML, CSS, and JavaScript and integrates seamlessly with the Flask API. The extension automatically captures the active tab's URL or allows the user to input a URL manually. It then sends the URL to the backend for analysis and displays the prediction results directly within the browser popup. The browser extension improves usability by providing instant feedback without requiring users to navigate to a separate webpage. It also enhances transparency by showing risky features that influence the decision, promoting trust in the system. Together, these deployment components ensure accessibility and practicality for end-users, making the solution effective for real-world phishing detection.

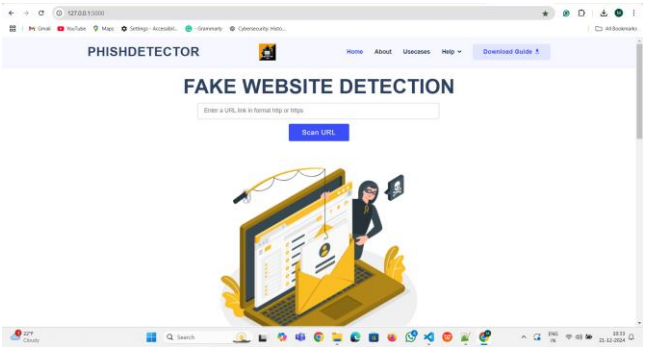


Fig 9: Webpage

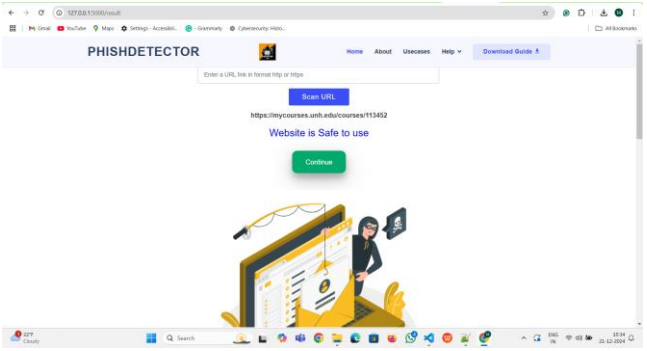
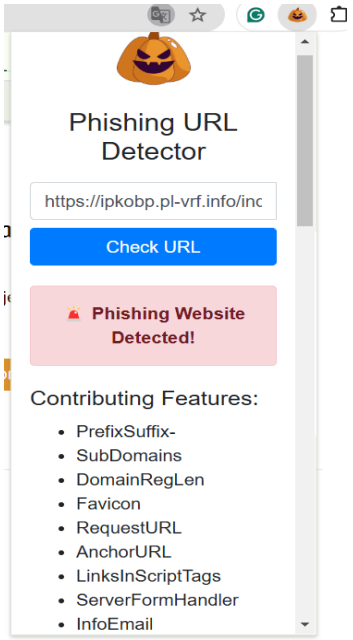
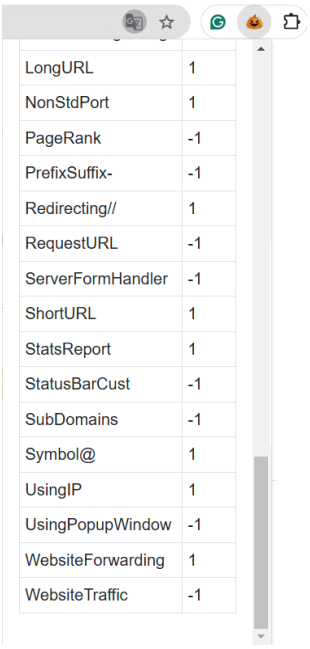


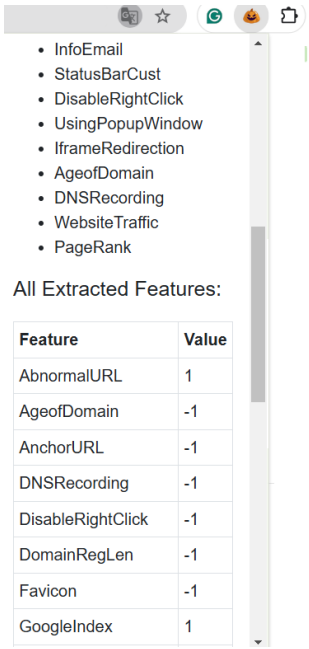
Fig 10: Legitimate

Fig 11: Phishing





LongURL	1
NonStdPort	1
PageRank	-1
PrefixSuffix-	-1
Redirecting//	1
RequestURL	-1
ServerFormHandler	-1
ShortURL	1
StatsReport	1
StatusBarCust	-1
SubDomains	-1
Symbol@	1
UsingIP	1
UsingPopupWindow	-1
WebsiteForwarding	1
WebsiteTraffic	-1



Feature	Value
AbnormalURL	1
AgeofDomain	-1
AnchorURL	-1
DNSRecording	-1
DisableRightClick	-1
DomainRegLen	-1
Favicon	-1
GoogleIndex	1

Fig 12: Results of a Browser Extension

Figure 12 represents the result of a phishing URL with the help of extracted features.

5. CONCLUSION

Internet users are at serious risk from phishing. Web security faces a significant issue due to the rapid development and spread of phishing methods. It is difficult to identify a fraudulent URL, but machine learning methods can help. In this study, we used various algorithms to investigate the URL's linguistic and domain-based properties and created a machine-learning model. The GBC algorithm produced the best outcomes out of the bunch. However, some URLs in the dataset are not in the Whois database. Thus, we cannot collect all of the features; as a result, we need to add more features and fresh URL data to enhance accuracy. We may use our machine learning model to build a search engine in the future, which will enable us to identify any fraudulent URLs and ban them, eliminating phishing and developing a framework that can discover new phishing attack types on its own by giving the surveillance process a more advanced feature.

6. REFERENCES

[1] D. Sahoo, “Malicious URL detection using machine learning: a survey”, 2022

[2] Shraddha Parekh, Dhwanil Parikh, Srushti Kotak, Smita Sankhe, A new method for detection of phishing websites: URL detection, IEEE, 2018, pp. 949–952.

[3] Gunter Ollmann, “The Phishing Guide Understanding & Preventing Phishing Attacks”, IBM Internet Security Systems, 2007.

[4] Blum A, Wardman B, Solorio T, Warner G. Lexical feature based phishing URL detection using online learning. In: Proceedings of the 3rd ACM workshop security and artificial intelligence, AISEC; 2010. https://doi.org/10.1145/1866423.1866434. 2010 October 8

[5] Mohammad R., Thabtah F. McCluskey L., (2015) Phishing websites dataset. Available:

https://archive.ics.uci.edu/ml/datasets/Phishing+Websites Accessed January 2016

[6] Sandeep Kumar Satapathy, Shruti Mishra, Pradeep Kumar Mallick, Lavanya Badiginchala, Ravali Reddy Gudur, Siri Chandana Guttha," Classification of Features for detecting Phishing Web Sites based on Machine Learning Techniques",International Journal of Innoative Technology and Exploring Engineering (IJITEE) ISSN:2278-3075, Volume-8 Issue-8S2, June 2019.

[7] Mahajan Mayuri Vilas, Kakadee Prachi Ghansham, Sawant Purva Jaypralash, Pawar Shila," Detection of Phishing Website Using Machine Learning Approach"," International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT)" 2019.

[8] Ilker Kara, Murathan Ok, Ahmet Ozaday, "Characteristics of Understanding URLs and Domain Names Features: The Detection of Phishing Websites with Machine Learning Methods", IEEE Access, vol.10, pp.124420-124428, 2022.

[9] Dong-Jie Liu, Jong-Hyouk Lee, "A CNN-Based SIA Screenshot Method to Visually Identify Phishing Websites", Journal of Network and Systems Management, vol.32, no.1, 2024.

[10] Anirudha Joshi, Prof. Tanuja R Pattanshetti, "Phishing Attack Detection using Feature Selection Techniques", SSRN Electronic Journal, 2019.

[11] www.phishtank.com