

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“Jnana Sangama”, Belagavi-590014, Karnataka



2024-25

A Main Project Report On

“PHISHING WEBSITE DETECTION USING ML”

Submitted in partial fulfillment for the academic year 2024-2025

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by,

SYED OWAIZ UZ ZAMAN 1GC21CS098

Under the Guidance of
Mr. Irfan Khan B.M

Asst. Professor CSE

Department of CSE



Department of Computer Science and Engineering
GHOUSIA COLLEGE OF ENGINEERING

2024-25

GHOUSIA COLLEGE OF ENGINEERING

RAMANAGARA-562159

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certificate that the Main Project report on topic "**PHISHING WEBSITE DETECTION USING ML**" has been successfully submitted by **SYED OWAIZ UZ ZAMAN(1GC21CS098)** a Bonafide students at **Ghousia College of Engineering** affiliated to **Visvesvaraya Technological University, Belagavi** during the year 2024- 2025. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report submitted to the department. The Main Project report has been approved as it satisfies the academic requirement in respect of the main project work as prescribed in 7th semester.

.....
(Signature of the Guide)

Mr. Irfan Khan B.M
Asst. Professor CSE, GCE,
Ramanagaram

.....
(Signature of HOD)

Dr. Dilshad Begum
Professor & HOD, CSE, GCE,
Ramanagaram

.....
(Signature of Principal)

Dr. N S Kumar
Principal, GCE,
Ramanagaram

Name of the Examiner

1. _____

2. _____

Signature with Date

ACKNOWLEDGEMENT

The sense of contentment and elation that accomplishes the successful of completion of our task would be incomplete without mentioning the names of the people who helped in accomplishment of this Main Project, whose constant guidance, support and encouragement resulted in its realization

I take this opportunity to thank our principal, **Dr. N S Kumar** for providing us with serene and healthy environment within the college, which helped us in concentrating on our task.

I express our deep sense of gratitude to our professor **Dr. Dilshad Begum** Head of Department of CSE for her constant encouragement and useful suggestions in carrying out this project successful

I thank my guide professor, **Mr. Irfan Khan B.M** Department of CSE for having provided the necessary guidance and facilities to carry out the project.

I thank all teaching and non -teaching staff of the department of CSE, who has helped me in completing the project.

I wish to thank my friends for their useful guidance on various topics. Last, but not least, I would like to thank my parents for the support.

.....
Syed Owaiz Uz Zaman
(1GC21CS098)

ABSTRACT

The Phishing is one of the most widespread and dangerous cyber threats, aiming to steal sensitive user information by impersonating legitimate websites. The increasing sophistication of phishing techniques makes it challenging for traditional detection methods, such as blacklists and rule-based systems, to remain effective. In this project, we present a machine learning-based solution for phishing website detection using a Gradient Boosting Classifier. The model is trained on a datasets containing both legitimate and phishing URLs, with features extracted from the URL structure, domain information, and web-page content. Gradient Boosting is chosen for its high accuracy, ability to handle complex data patterns, and resilience to over fitting. Our trained model demonstrates excellent performance in identifying phishing websites with high precision. To make the solution user-friendly, we deploy it through a Flask-based web application, allowing users to enter a URL and receive instant feedback on its legitimacy. This project showcases the practical application of machine learning in cyber security and provides an effective tool to combat phishing attacks in real-time.

TABLE OF CONTENTS

Chapter No.	Title	Page No.
	ACKNOWLEDGMENT	i
	ABSTRACT	ii
	TABLE OF CONTENTS	iii
	LIST OF FIGURES	iv
1	Introduction	1
	1.1 Existing system and Its Limitations	
2	LITERATURE SURVEY	4
	2.1 Relevant recent paper summary	
3	PROBLEM STATEMENT AND SCOPE	7
	3.1 Problem Statement	
	3.2 Existing System	
	3.3 Proposed System	
4	SYSTEM ARCHITECHTURE	8
	4.1 System architecture	
5	OBJECTIVES	10
6	IMPLEMENTATION	11
	6.1 Data Collection	
	6.2 Dataset	
	6.3 Data Preparation	
	6.4 Model Selection	
	6.4.1 What is boosting	
	6.4.2 What is a Gradient boosting Algorithm	
7	RESULT AND SNAPSHOTS	22
8	CONCLUSION	29
	8.1 Conclusion	
	8.2 Future Enhancements	
9	REFERENCES	30

LIST OF FIGURES

Figure No.	Name	Page No.
4.1	System architecture	8
4.2	Data Flow Diagram	9
6.4.2.1	Regression Tree	16
6.4.2.2	Regression Tree After Calculation	17
7.1	Home Interface for Phishing Website Detection	22
7.2	Prediction Result	23
7.3	Performance Analysis	24
7.4	Chart Analysis	24
7.5	Login page	25
7.6	Upload page	25
7.7	Preview of Dataset (i)	26
7.8	Preview of Dataset (ii)	26
7.9	Preview of Dataset after Training	27
7.10	Phishing URL Detection	27
7.11	Prediction Result	28

CHAPTER 1

INTRODUCTION

Phishing is a malicious cyber activity that has emerged as one of the most significant threats to internet users and organizations worldwide. It involves the creation of fraudulent websites or emails that appear to originate from trusted sources, such as banks, e-commerce platforms, or government agencies. The main objective is to manipulate users into providing sensitive information, including usernames, passwords, credit card details, and banking credentials.

The sophistication of phishing attacks has increased dramatically in recent years. Unlike early phishing attempts, which were relatively easy to identify due to obvious grammatical mistakes and poorly designed interfaces, modern phishing websites are nearly indistinguishable from legitimate ones. They often use HTTPS certificates, clone the layout and branding of real websites, and utilize advanced evasion techniques to bypass detection systems.

According to recent cybersecurity reports, phishing is responsible for more than 90% of all social engineering attacks and contributes significantly to data breaches, identity theft, and financial fraud. The damage caused by these attacks is not limited to individuals but also affects businesses and governments, resulting in loss of revenue, legal penalties, and reputational damage.

Traditional phishing prevention methods such as browser-based warnings, community-driven blacklists, and basic URL filters are no longer sufficient. These methods often fail to detect newly generated phishing pages (zero-hour attacks) and can be easily evaded by attackers through minor URL alterations or changes in web content.

In response to these challenges, the use of artificial intelligence (AI), particularly machine learning (ML) and deep learning (DL), has gained traction. By analyzing multiple layers of information—including lexical features of URLs, web content, HTML structure, and domain metadata—AI-based systems can significantly improve the accuracy and speed of phishing detection.

1.1 Existing System and Its Limitations

1. Blacklist-Based Detection

One of the most commonly used phishing detection techniques is based on URL blacklists. These blacklists store a database of known phishing URLs and prevent users from accessing them by issuing warnings or blocking connections. Although simple and widely supported by modern browsers and antivirus software, blacklist-based methods are reactive rather than proactive. They depend heavily on the manual reporting of phishing URLs or automated crawlers that may not catch newly launched phishing sites in time.

Limitations:

- Cannot detect new or unknown phishing sites.
- Delay in updating blacklists reduces effectiveness.
- Vulnerable to dynamic and short-lived phishing attacks.

2. Heuristic-Based Detection

Heuristic-based systems rely on predefined rules to detect suspicious URLs or website content. These rules may include checking for unusual domain patterns (e.g., use of IP addresses instead of domain names), presence of suspicious JavaScript functions, or excessive use of redirects and iframes. While more flexible than blacklists, heuristic systems are prone to misclassification, especially when attackers deliberately design their pages to avoid triggering the rules.

Limitations:

- Rule sets need constant updates to match evolving threats.
- High false positives (flagging legitimate websites).
- May not generalize well across diverse website structures.
- Malware that completely rewrites its code can still evade heuristic detection.
- Dynamic analysis techniques like sandboxing can introduce delays in threat detection.

3. Machine Learning-Based Detection

Machine learning (ML) methods analyze patterns in URLs and webpage characteristics using supervised learning algorithms. These models are trained on labeled datasets of phishing and legitimate websites. Common features used include URL length, number of special characters, presence of SSL certificates, and domain registration information. Once trained, the model can predict the likelihood of a new site being phishing.

Limitations:

- Heavy reliance on manual feature extraction and domain expertise.
- Performance depends on the quality and diversity of the training dataset.
- May not perform well on websites with unseen structures or languages.

4. Content-Based and Visual Similarity Systems

Some modern detection systems use visual similarity metrics and OCR (Optical Character Recognition) to compare suspect websites with known legitimate templates. These systems aim to detect phishing sites that visually mimic target brands. While promising, these systems are computationally expensive and require large storage for image templates.

Limitations:

- High resource usage and latency in detection.
- Limited to popular websites with known visual signatures.
- Can be bypassed using dynamic content loading or image distortions.

Common Challenges Across All Systems

- **Zero-Day Detection:** New phishing sites are often active for only a few hours, making it difficult for traditional methods to catch them in time.
- **Evasion Tactics:** Attackers use encryption (HTTPS), short URLs, and redirect chains to confuse detection mechanisms.
- **Balance Between Precision and Recall:** A good system must minimize both false positives (to avoid user frustration) and false negatives (to avoid security breaches).

CHAPTER 2

LITERATURE SURVEY

The growing prevalence of phishing attacks has driven extensive research into effective detection methods. Early approaches, such as rule-based systems and URL blacklisting, proved inadequate against evolving threats, particularly zero-day attacks that bypass predefined filters. As a result, recent studies have focused on machine learning techniques to enhance phishing detection. For instance, some researchers have implemented probabilistic neural networks combined with clustering algorithms to reduce detection time and improve classification accuracy.

Others have examined domain name analysis using statistical measures to identify algorithmically generated domains often used by phishing websites. Beyond technical methods, certain works have explored human-centric interventions like behavioral nudges and warning systems, though these often fall short due to user inattention or poor usability. Overall, the literature highlights the importance of hybrid approaches that integrate machine learning, domain analysis, and user behavior modeling to form robust and adaptive phishing detection systems.

Despite significant advancements, challenges remain in terms of scalability, generalization to new attack patterns, real-time performance, and the interpretability of deep learning models. These gaps underline the need for continued innovation to develop effective, explainable, and user-friendly phishing detection solutions.

2.1 Relevant recent paper's summary

Research in phishing website detection has expanded considerably due to the increasing scale and sophistication of phishing attacks. Several recent studies offer innovative detection approaches and valuable insights into user behaviour and system vulnerabilities.

One key study, "*A Literature Review on Phishing Detection*" by Andrew Jones et al. [1], presents a broad classification of phishing countermeasures into four main categories:

detection, offensive defines, correction, and prevention. The review highlights that no single countermeasure is sufficient due to the diverse vectors used in phishing attacks.

It emphasizes a layered defines approach where multiple solutions, each targeting specific vulnerabilities, are implemented together. However, the study notes a major limitation—variability in user behaviour and the evolving nature of attacks make it difficult to design a universally effective system.

Another influential work, *“Nudges for Privacy and Security: Understanding and Assisting Users’ Choices Online”* by Acquits et al. [2], explores behavioural interventions such as nudges to enhance user security decisions. The authors argue that well-designed nudges can effectively improve user choices without restricting freedom. Nevertheless, the study also warns of ethical challenges and contextual dependencies. Nudges may fail or even backfire if not properly aligned with user expectations and cognitive load.

In contrast, *“Priming and Warnings Are Not Effective to Prevent Social Engineering Attacks”* by Overink et al. [3] offers a critical perspective on awareness mechanisms. Through controlled experiments, the authors found that participants continued to disclose sensitive data even after receiving priming cues and warnings. In some cases, warnings even increased the likelihood of disclosure. This finding suggests that traditional education and awareness campaigns may not suffice and must be supplemented with stronger technological safeguards. From a technical standpoint, *“Detection of Phishing Websites Based on Probabilistic Neural Networks and K-Medoids Clustering”* by El-Alfy and El-Sayed M. [4] proposes a hybrid approach combining probabilistic neural networks (PNNs) with K-medoids clustering. The model showed a detection accuracy of 97.4%, with a 40.26% reduction in processing time compared to traditional classifiers. This hybrid method improves classification speed and scalability. However, generalizability to unknown attack patterns and adaptability to evolving phishing strategies remain challenges.

Finally, *“Detecting Algorithmically Generated Malicious Domain Names”* by Hao et al. [5] focuses on domain-based phishing detection. The authors use statistical metrics such as Kullback-Leibler divergence and Edit distance to identify algorithmically generated domains (AGDs), often used in botnet operations. Their method achieves high accuracy in detecting botnets like Conficker and Torpig. Still, the approach depends heavily on access to clean and timely DNS data and may not perform well in encrypted or obfuscated communication channels.

These studies collectively underscore the importance of both technical robustness and human-centric considerations in developing next-generation phishing detection systems.

2.2 Conclusion about literature survey

The literature indicates a significant evolution in phishing detection methods. While machine learning and hybrid models have dramatically improved detection accuracy, they often face limitations in terms of adaptability, interpretability, and real-time deployment. Simultaneously, behavioral studies reveal that user-focused defenses such as nudges and warnings are not always reliable, emphasizing the necessity for a multi-pronged defense combining technical, behavioral, and policy-level interventions. Our project builds upon these insights to develop a phishing website detection system using intelligent learning-based classifiers and human-factor-aware interfaces.

CHAPTER 3

PROBLEM STATEMENT AND SCOPE

3.1 PROBLEM STATEMENT

Phishing attacks continue to escalate in frequency and sophistication, posing a significant risk to online users and digital platforms. Existing detection mechanisms are often unable to cope with newly generated, previously unseen phishing websites. These systems are either too reliant on static blacklists or require extensive manual feature engineering, making them inefficient and outdated in the face of modern cyber threats. Furthermore, traditional detection systems struggle with scalability, real-time performance, and often produce inaccurate results due to high false-positive and false-negative rates. There is an urgent need for an intelligent, automated, and adaptive phishing detection solution capable of identifying both known and unknown threats with high precision.

3.2 EXISTING SYSTEM

The current landscape of phishing detection relies heavily on traditional techniques such as blacklist-based filtering, rule-based systems, and machine learning algorithms trained on handcrafted features. These systems often identify phishing websites by analysing static characteristics such as the presence of IP addresses in URLs, abnormal domain lengths, usage of HTTP instead of HTTPS, or the presence of special characters in the link. While such methods offer some level of protection, they are largely reactive, relying on known phishing patterns or URLs. As a result, they fail to detect zero-day phishing attacks and new types of threats that do not follow the pre-established rules or signatures.

3.3 PROPOSED SYSTEM

The proposed system aims to effectively detect phishing websites by implementing a **Gradient Boosting Classifier (GBC)**, a powerful ensemble machine learning technique known for its high prediction accuracy and robustness. This approach is particularly well-suited for phishing detection tasks, where subtle variations and feature interactions must be captured with high precision.

CHAPTER 4

SYSTEM ARCHITECTURE

4.1 SYSTEM ARCHITECTURE

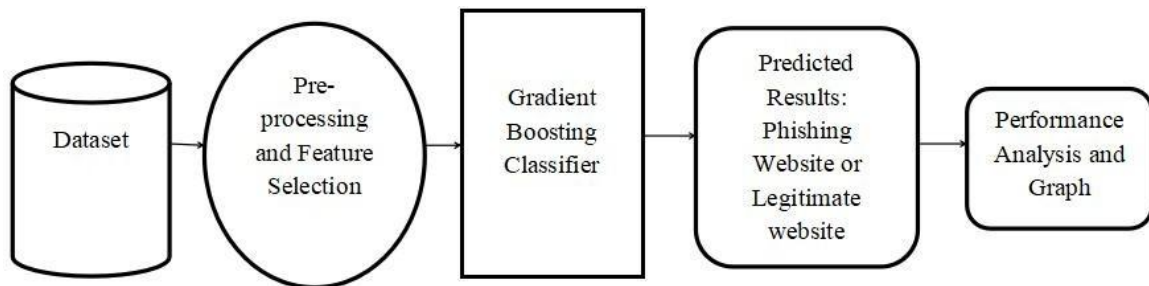


Figure 4.1: System Architecture

1. **Dataset:** The process begins with a dataset, which presumably contains information about various websites, labeled as either phishing or legitimate.
2. **Preprocessing and Feature Selection:** The dataset undergoes preprocessing, where the data is cleaned and transformed. Feature selection involves identifying the most relevant characteristics of a website (e.g., URL structure, domain information) that help in distinguishing between phishing and legitimate sites.
3. **Gradient Boosting Classifier:** A gradient boosting classifier, a type of machine learning algorithm, is used. This classifier is trained on preprocessed data with selected features to learn patterns that indicate whether a website is phishing or not.
4. **Predicted Results:** The trained classifier is then used to predict whether a given website is a phishing website or a legitimate one.
5. **Performance Analysis and Graph:** Finally, the system's performance is evaluated. This is likely involves calculating metrics like accuracy, precision, and recall assessing how well the system can detect phishing websites. The results of the analysis may be presented in a graph for visualization.

DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

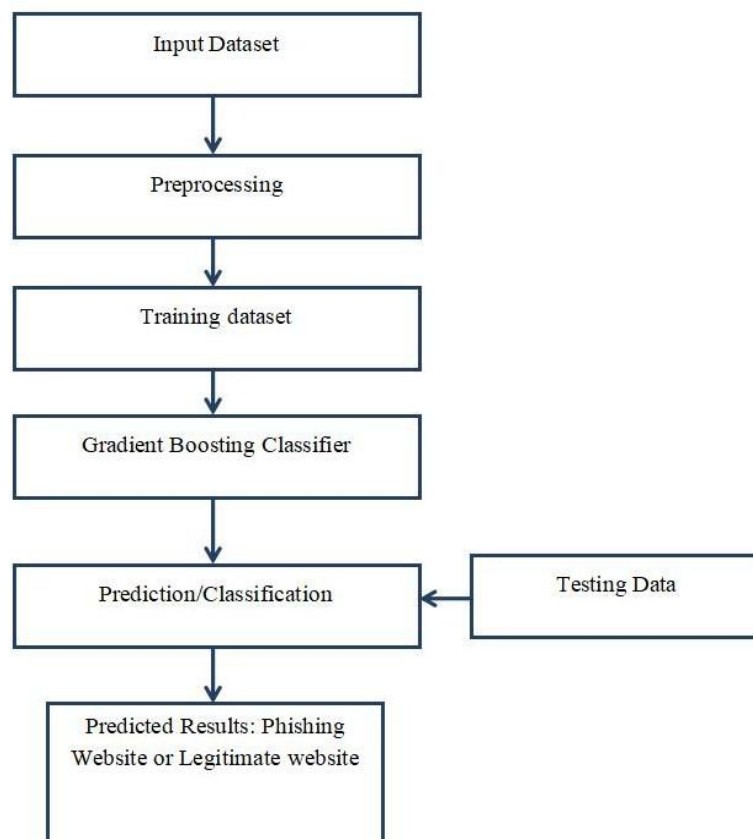


Figure 4.2: Data Flow Diagram

CHAPTER 5

OBJECTIVES

The central objective of employing machine learning for phishing website detection lies in the creation of intelligent, automated systems capable of accurately discerning malicious websites designed for illicit data harvesting from legitimate online resources. These systems strive for enhanced accuracy compared to traditional methods by learning intricate patterns and adapting to the ever-changing landscape of phishing tactics, enabling proactive identification of novel attacks not yet catalogued in conventional security measures. By automating the detection process, machine learning reduces the need for manual analysis, offering a scalable solution capable of efficiently processing vast amounts of web data. Furthermore, the adaptability of these models through continuous retraining ensures sustained effectiveness against evolving threats, ultimately contributing to a significantly safer digital environment by preventing financial fraud, identity theft, and reputational harm. Beyond mere detection, these systems can also highlight crucial characteristics indicative of phishing attempts, offering valuable insights for bolstering overall security strategies and enhancing user awareness.

- 1 **Automate and Improve Detection:** Use machine learning algorithms (like the Gradient Boosting Classifier shown) to automate the process of identifying phishing websites, improving detection speed and accuracy compared to manual methods.
- 2 **Learn from Data:** Train the machine learning model on a dataset of website features to enable it to recognize patterns and characteristics that distinguish phishing sites from legitimate ones.
- 3 **Generalize to New Threats:** Develop a system that can generalize and accurately detect new, previously unseen phishing websites by learning the underlying patterns of phishing attacks.
- 4 **Enhance Predictive Capability:** Go beyond simply listing known phishing sites (blacklists) by predicting which sites are likely to be phishing sites before they cause harm.
- 5 **Optimize Performance:** The "Performance Analysis and Graph" stage in the image suggests an objective to evaluate and optimize the machine learning model's performance, focusing on metrics like accuracy, precision, and recall.

CHAPTER 6

IMPLEMENTATION

6.1 Data Collection:

In the first module we develop the data collection process. This is the first real step towards the real development of a machine learning model, collecting data. This is a critical step that will cascade in how good the model will be, the more and better data that we get; the better our model will perform.

There are several techniques to collect the data, like web scraping, manual interventions. The dataset is referred from the popular dataset repository called kaggle. The following is the dataset link for the Detection of Phishing Websites Using Machine Learning.

Kaggle Dataset Link:

<https://www.kaggle.com/datasets/jayaprakashpondy/phishing-websites-feature-dataset>

6.2 Dataset:

The dataset consists of 11054 individual data. There are 32 columns in the dataset, which are described below.

Feature	Type	Values
Index	Index ID	-
UsingIP	Categorical (signed numeric)	{ -1, 1 }
LongURL	Categorical (signed numeric)	{ 1, 0, -1 }
ShortURL	Categorical (signed numeric)	{ 1, -1 }
Symbol@	Categorical (signed numeric)	{ 1, -1 }
Redirecting://	Categorical (signed numeric)	{ -1, 1 }
PrefixSuffix-	Categorical (signed numeric)	{ -1, 1 }
SubDomains	Categorical (signed numeric)	{ -1, 0, 1 }
HTTPS	Categorical (signed numeric)	{ -1, 0, 1 }
DomainRegLen	Categorical (signed numeric)	{ -1, 1 }
Favicon	Categorical (signed numeric)	{ 1, -1 }
NonStdPort	Categorical (signed numeric)	{ 1, -1 }
HTTPSDomainURL	Categorical (signed numeric)	{ -1, 1 }
RequestURL	Categorical (signed numeric)	{ 1, -1 }
AnchorURL	Categorical (signed numeric)	{ -1, 0, 1 }

Feature	Type	Values
LinksInScriptTags	Categorical (signed numeric)	{ -1, 0, 1 }
ServerFormHandler	Categorical (signed numeric)	{ -1, 0, 1 }
InfoEmail	Categorical (signed numeric)	{ -1, 1 }
AbnormalURL	Categorical (signed numeric)	{ -1, 1 }
WebsiteForwarding	Categorical (signed numeric)	{ 0, 1 }
StatusBarCust	Categorical (signed numeric)	{ -1, 1 }
DisableRightClick	Categorical (signed numeric)	{ -1, 1 }
UsingPopupWindow	Categorical (signed numeric)	{ -1, 1 }
IframeRedirection	Categorical (signed numeric)	{ -1, 1 }
AgeofDomain	Categorical (signed numeric)	{ -1, 1 }
DNSRecording	Categorical (signed numeric)	{ -1, 1 }
WebsiteTraffic	Categorical (signed numeric)	{ -1, 0, 1 }
PageRank	Categorical (signed numeric)	{ -1, 1 }
GoogleIndex	Categorical (signed numeric)	{ -1, 1 }
LinksPointingToPage	Categorical (signed numeric)	{ -1, 0, 1 }
StatsReport	Categorical (signed numeric)	{ -1, 1 }
Class	Categorical (signed numeric)	{ -1, 1 }

6.3 Data Preparation:

Wrangle data and prepare it for training. Clean that which may require it (remove duplicates, correct errors, deal with missing values, normalization, data type conversions, etc.).

Randomize data, which erases the effects of the particular order in which we collected and/or otherwise prepared our data. Visualize data to help detect relevant relationships between variables or class imbalances (bias alert!), or perform other exploratory analysis.

Split into training and evaluation sets.

6.4 Model Selection:

We used Gradient Boosting Classifier machine learning algorithm. We got an accuracy of training Accuracy 98.9% so we implemented this "***Gradient Boosting Classifier Algorithm***" algorithm.

6.4.1 What is boosting?

While studying machine learning you must have come across this term called Boosting. It is the most misinterpreted term in the field of Data Science. The principle behind boosting algorithms is first we built a model on the training dataset, then a second model is built to rectify the errors present in the first model. Let me try to explain to you what exactly does this means and how does this works.

Suppose you have n data points and 2 output classes (0 and 1). You want to create a model to detect the class of the test data. Now what we do is randomly select observations from the training dataset and feed them to model 1 (M_1), we also assume that initially, all the observations have an equal weight that means an equal probability of getting selected.

Remember in assembling techniques the weak learners combine to make a strong model so here $M_1, M_2, M_3, \dots, M_n$ all are weak learners.

Since M_1 is a weak learner, it will surely misclassify some of the observations. Now before feeding the observations to M_2 what we do is update the weights of the observations which are wrongly classified. You can think of it as a bag that initially contains 10 different colour balls but after some time some kid takes out his favourite colour ball and put 4 red colour balls instead inside the bag. Now off-course the probability of selecting a red ball is higher. This same phenomenon happens in Boosting techniques, when an observation is wrongly classified, its weight get's updated and for those which are correctly classified, their weights get decreased. The probability of selecting a wrongly classified observation gets increased hence in the next model only those observations get selected which were misclassified in model.

Similarly, it happens with M_2 , the wrongly classified weights are again updated and then fed to M_3 . This procedure is continued until and unless the errors are minimized, and the dataset is predicted correctly. Now when the new datapoint comes in (Test data) it passes through all the models (weak learners) and the class which gets the highest vote is the output for our test data.

6.4.2 What is a Gradient boosting Algorithm?

The main idea behind this algorithm is to build models sequentially and these subsequent models try to reduce the errors of the previous model. But how do we do that? How do we reduce the error? This is done by building a new model on the errors or residuals of the previous model.

When the target column is continuous, we use **Gradient Boosting Regressor** whereas when it is a classification problem, we use **Gradient Boosting Classifier**. The only difference between the two is the “*Loss function*”. The objective here is to minimize this loss function by adding weak learners using gradient descent. Since it is based on loss function hence for regression problems, we’ll have different loss functions like Mean squared error (**MSE**) and for classification, we will have different for e.g **log-likelihood**.

Understand Gradient Boosting Algorithm with example

Let’s understand the intuition behind Gradient boosting with the help of an example. Here our target column is continuous hence we will use Gradient Boosting Regressor.

Step -1 The first step in gradient boosting is to build a base model to predict the observations in the training dataset. For simplicity we take an average of the target column and assume that to be the predicted value as shown below:

Why did I say we take the average of the target column? Well, there is math involved behind this. Mathematically the first step can be written as:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^n L(y_i, \gamma)$$

Looking at this may give you a headache, but don’t worry we will try to understand what is written here.

- Here L is our loss function
- γ is our predicted value
- $\arg\min$ means we have to find a predicted value/ γ for which the loss function is minimum.

Since the target column is continuous our loss function will be:

$$L = \frac{1}{2} \sum_{i=1}^n (y_i - \gamma_i)^2$$

Here y_i is the observed value

And γ_i is the predicted value

Now we need to find a minimum value of γ such that this loss function is minimum. We all have studied how to find minima and maxima in our 12th grade. Did we use to differentiate this loss function and then put it equal to 0 right? Yes, we will do the same here.

$$dL/d\gamma = (2/2) \sum_{i=1}^n (y_i - \gamma_i) = - \sum_{i=1}^n (y_i - \gamma_i)$$

Let's see how to do this with the help of our example. Remember that y_i is our observed value and γ_i is our predicted value, by plugging the values in the above formula we get:

Hence for $\gamma=14500$, the loss function will be minimum so this value will become our prediction for the *base model*.

Step-2 The next step is to calculate the pseudo residuals which are (observed value – predicted value)

Again the question comes why only observed – predicted? Everything is mathematically proved, let's find where did this formula come from. This step can be written as:

$$r_{im} = - [\partial L(y_i, F(x_i)) / \partial F(x_i)] \text{ evaluated at } F(x) = F_{m-1}(x), \text{ for } i = 1, \dots, n.$$

Here $F(x_i)$ is the previous model and m is the number of DT made.

We are just taking the derivative of loss function w.r.t the predicted value and we have already calculated this derivative:

$$dL/d\gamma = - (y_i - \gamma_i) = - (\text{Observed} - \text{Predicted})$$

If you see the formula of residuals above, we see that the derivative of the loss function is multiplied by a negative sign, so now we get:

$$(\text{Observed} - \text{Predicted})$$

The predicted value here is the prediction made by the previous model. In our example the prediction made by the previous model (initial base model prediction) is 14500, to calculate the residuals our formula becomes:

$$(\text{Observed} - 14500)$$

Step- 3 We will build a model on these pseudo residuals and make predictions. Why do we do this? Because we want to minimize these residuals and minimizing the residuals will eventually improve our model accuracy and prediction power. So, using the Residual as target and the original feature Cylinder number, cylinder height, and Engine location we will generate new predictions.

Note that the predictions, in this case, will be the error values, not the predicted car price values since our target column is an error now. let's say $hm(x)$ is our DT made on these residuals.

Step- 4 In this step we find the output values for each leaf of our decision tree. That means there might be a case where 1 leaf gets more than 1 residual, hence we need to find the final

output of all the leaves. TO find the output we can simply take the average of all the numbers in a leaf, doesn't matter if there is only 1 number or more than 1.

Let's see why do we take the average of all the numbers. Mathematically this step can be represented as:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^n L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$

Here $h_m(x_i)$ is the DT made on residuals and m is the number of DT. When $m=1$ we are talking about the 1st DT and when it is " M " we are talking about the last DT.

The output value for the leaf is the value of gamma that minimizes the Loss function. The left-hand side "*Gamma*" is the output value of a particular leaf. On the right-hand side $[F_{m-1}(x_i) + \gamma h_m(x_i)]$ is similar to step 1 but here the difference is that we are taking previous predictions whereas earlier there was no previous prediction.

Let's understand this even better with the help of an example. Suppose this is our regressor tree:

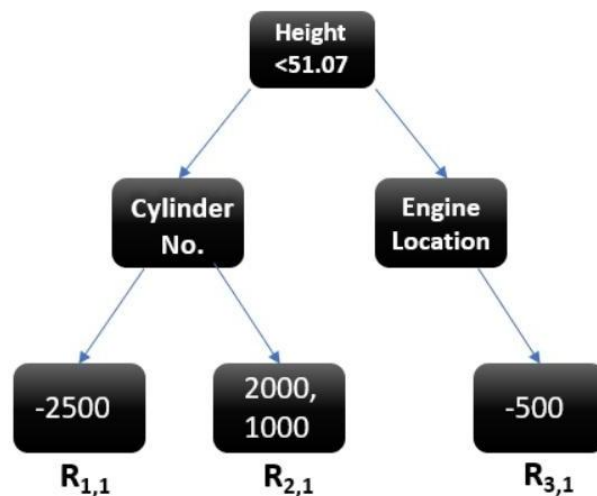


fig 6.4.2.1:regression tree

We see 1st residual goes in R1,1 ,2nd and 3rd residuals go in R2,1 and 4th residual goes in R3,1 .

Let's calculate the output for the first leaf that is R1,1

$$\gamma_{1,1} = \arg\min_{\gamma} \frac{1}{2} (12000 - (14500 + \gamma))^2$$

$$\gamma_{1,1} = \arg\min_{\gamma} \frac{1}{2} (-2500 - \gamma)^2$$

Now we need to find the value for gamma for which this function is minimum. So we find the derivative of this equation w.r.t gamma and put it equal to 0.

$$d/d\gamma [(1/2)(2000 - \gamma)^2 + (1/2)(1000 - \gamma)^2] = 0$$

$$2000 - \gamma + 1000 - \gamma = 0$$

$$3000 - 2\gamma = 0$$

$$\gamma = 3000 / 2$$

$$\gamma = 1500$$

$$d/d\gamma [(1/2)(-2500 - \gamma)^2] = 0$$

$$-2500 - \gamma = 0$$

$$\gamma = -2500$$

Let's take the derivative to get the minimum value of gamma for which this function is we end up with the **average** of the residuals in the leaf R_{2,1}. Hence if we get any leaf with more than 1 residual, we can simply find the average of that leaf and that will be our final output.

Now after calculating the output of all the leaves, we get:

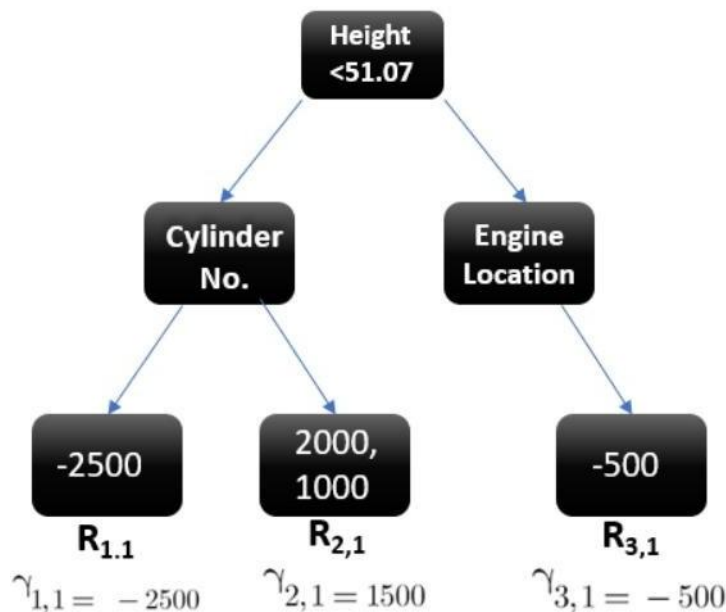


Fig 6.4.2.2: regression tree after calculation

Step-5 This is finally the last step where we have to update the predictions of the previous model. It can be updated as:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + v_m h_m(\mathbf{x})$$

where m is the number of decision trees made.

Since we have just started building our model so our $m=1$. Now to make a new DT our new predictions will be $F_{m-1}(\mathbf{x})$ is the prediction of the base model (previous prediction) since $F_{1-1}=0$, F_0 is our base model hence the previous prediction is 14500.

$H_m(\mathbf{x})$ is the recent DT made on the residuals.

Let's calculate the new prediction now:

Suppose we want to find a prediction of our first data point which has a car height of 48.8. This data point will go through this decision tree and the output it gets will be multiplied with the learning rate and then added to the previous prediction.

Now let's say $m=2$ which means we have built 2 decision trees and now we want to have new predictions.

This time we will add the previous prediction that is $F_1(\mathbf{x})$ to the new DT made on residuals. We will iterate through these steps again and again till the *loss is negligible*.

What is Gradient Boosting Classifier?

A gradient boosting classifier is used when the target column is binary. All the steps explained in the Gradient boosting regressor are used here, the only difference is we change the loss function. Earlier we used Mean squared error when the target column was continuous but this time, we will use log-likelihood as our loss function.

Let's see how this loss function works, to read more about log-likelihood I recommend you to go through where I have given each detail you need to understand this. The loss function for the classification problem is given below:

$$L = -\sum_{i=1}^n [y_i \log(p) + (1 - p) \log(1 - p)]$$

Our first step in the gradient boosting algorithm was to initialize the model with some constant value, there we used the average of the target column but here we'll use log(odds) to get that constant value. The question comes why log(odds)?

When we differentiate this loss function, we will get a function of log(odds) and then we need to find a value of log(odds) for which the loss function is minimum.

Confused right? Okay let's see how it works:

Let's first transform this loss function so that it is a function of $\log(\text{odds})$, I'll tell you later why we did this transformation.

$$L = - [\sum (\text{from } i=1 \text{ to } n) (y_i \log(p) + (1 - y_i) \log(1 - p))]$$

$$= - y * \log(p) - (1 - y) * \log(1 - p)$$

$$= - y * \log(p) - \log(1 - p) + y * \log(1 - p)$$

$$= y * [\log(p) - \log(1 - p)] - \log(1 - p)$$

$$= - y * [\log(p) / \log(1 - p)] - \log(1 - p)$$

$$= - y * \log(p / (1 - p)) - \log(1 - p)$$

$$\text{Hence, } -y * \log(\text{odds}) - (-\log(1 + e^{\log(\text{odds})}))$$

$$L = -y * \log(\text{odds}) + \log(1 + e^{\log(\text{odds})})$$

Now this is our loss function, and we need to minimize it, for this, we take the derivative of this w.r.t to $\log(\text{odds})$ and then put it equal to 0, Here y are the observed values you must be wondering that why did we transform the loss function into the function of $\log(\text{odds})$. Actually, sometimes it is easy to use the function of $\log(\text{odds})$, and sometimes it's easy to use the function of predicted probability " p ".

It is not compulsory to transform the loss function, we did this just to have easy calculations. Hence the minimum value of this loss function will be our first prediction (base model prediction)

Now in the Gradient boosting regressor our next step was to calculate the pseudo residuals where we multiplied the derivative of the loss function with -1 . We will do the same but now the loss function is different, and we are dealing with the probability of an outcome now.

After finding the residuals we can build a decision tree with all independent variables and target variables as "Residuals".

Now when we have our first decision tree, we find the final output of the leaves because there might be a case where a leaf gets more than 1 residuals, so we need to calculate

the final output value. The math behind this step is out of the scope of this article so I will mention the direct formula to calculate the output of a leaf.

Finally, we are ready to get new predictions by adding our base model with the new tree we made on residuals. We will do the same but now the loss function is different, and we are dealing with the probability of an outcome now.

There are a few variations of gradient boosting and a couple of them are momentarily clarified in the coming article.

Analyze and Prediction:

In the actual dataset, we chose only 30 features:

Feature	Type	Values
UsingIP	Categorical (signed numeric)	{ -1, 1 }
LongURL	Categorical (signed numeric)	{ 1, 0, -1 }
ShortURL	Categorical (signed numeric)	{ 1, -1 }
Symbol@	Categorical (signed numeric)	{ 1, -1 }
Redirecting://	Categorical (signed numeric)	{ -1, 1 }
PrefixSuffix-	Categorical (signed numeric)	{ -1, 1 }
SubDomains	Categorical (signed numeric)	{ -1, 0, 1 }
HTTPS	Categorical (signed numeric)	{ -1, 0, 1 }
DomainRegLen	Categorical (signed numeric)	{ -1, 1 }
Favicon	Categorical (signed numeric)	{ 1, -1 }
NonStdPort	Categorical (signed numeric)	{ 1, -1 }
HTTPSDomainURL	Categorical (signed numeric)	{ -1, 1 }
RequestURL	Categorical (signed numeric)	{ 1, -1 }
AnchorURL	Categorical (signed numeric)	{ -1, 0, 1 }
LinksInScriptTags	Categorical (signed numeric)	{ -1, 0, 1 }
ServerFormHandler	Categorical (signed numeric)	{ -1, 0, 1 }
InfoEmail	Categorical (signed numeric)	{ -1, 1 }
AbnormalURL	Categorical (signed numeric)	{ -1, 1 }
WebsiteForwarding	Categorical (signed numeric)	{ 0, 1 }
StatusBarCust	Categorical (signed numeric)	{ -1, 1 }
DisableRightClick	Categorical (signed numeric)	{ -1, 1 }
UsingPopupWindow	Categorical (signed numeric)	{ -1, 1 }
IframeRedirection	Categorical (signed numeric)	{ -1, 1 }
AgeofDomain	Categorical (signed numeric)	{ -1, 1 }

Feature	Type	Values
DNSRecording	Categorical (signed numeric)	{ -1, 1 }
WebsiteTraffic	Categorical (signed numeric)	{ -1, 0, 1 }
PageRank	Categorical (signed numeric)	{ -1, 1 }
GoogleIndex	Categorical (signed numeric)	{ -1, 1 }
LinksPointingToPage	Categorical (signed numeric)	{ -1, 0, 1 }
StatsReport	Categorical (signed numeric)	{ -1, 1 }
Class	Categorical (signed numeric)	{ -1, 1 }

Accuracy on test set:

We got an accuracy of 97.6% on test set.

Saving the Trained Model:

Once you're confident enough to take your trained and tested model into the production-ready environment, the first step is to save it into a .h5 or .pkl file using a library like pickle.

CHAPTER 7

RESULT AND SNAPSHOTS

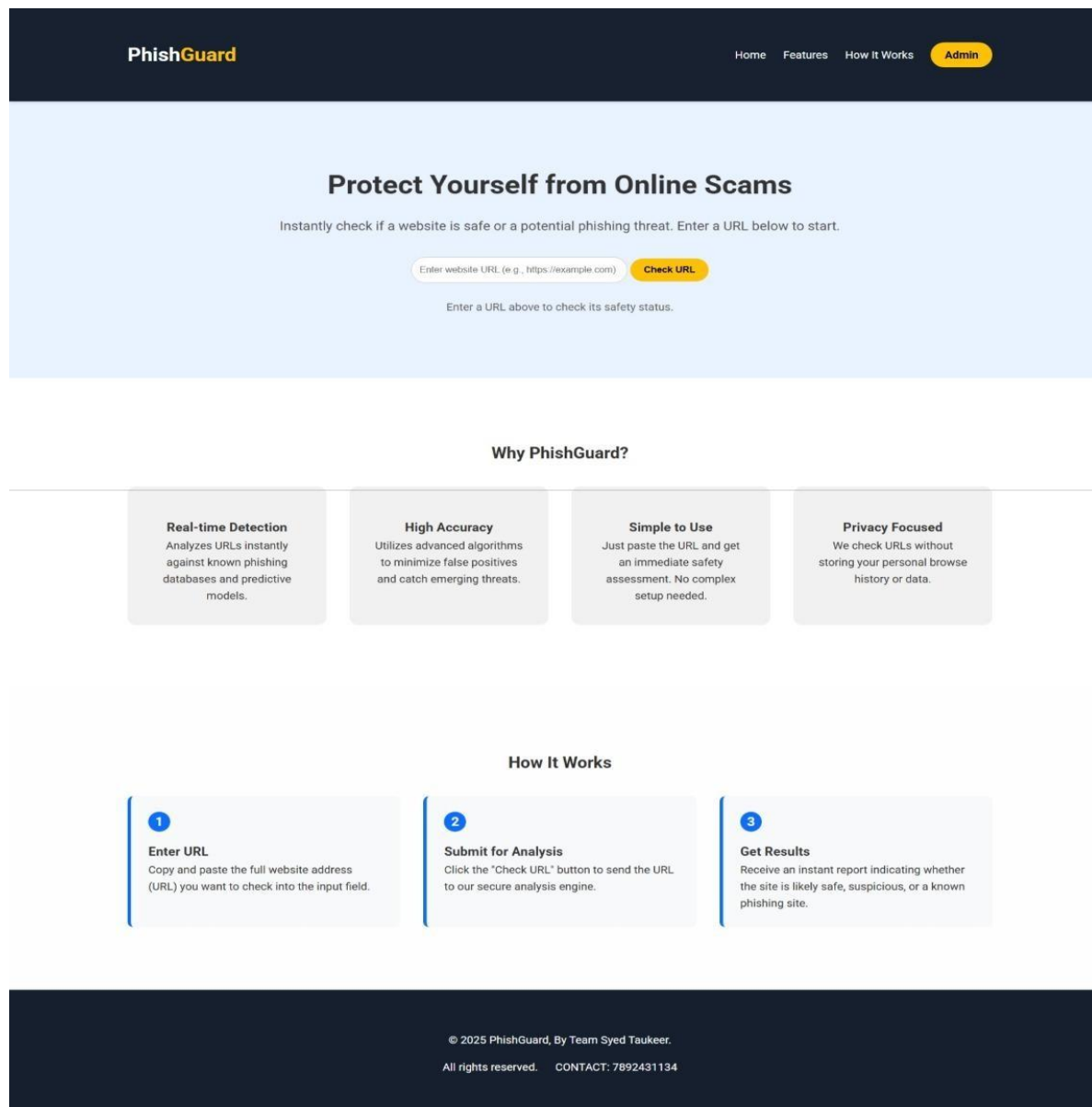


Figure 7.1: Home Interface for phishing website detection

PhishGuard is a web-based platform designed to protect users from online scams and phishing threats. By allowing users to enter website URLs, the system provides real-time detection and analysis to determine if a website is safe, suspicious, or malicious. The tool leverages advanced algorithms for high accuracy, minimizing false positives and effectively identifying emerging threats. PhishGuard emphasizes ease of use requiring no setup and prioritizes user privacy by ensuring that no personal browsing data is stored. The process is

straightforward users input a URL, submit it for analysis, and receive immediate safety results.

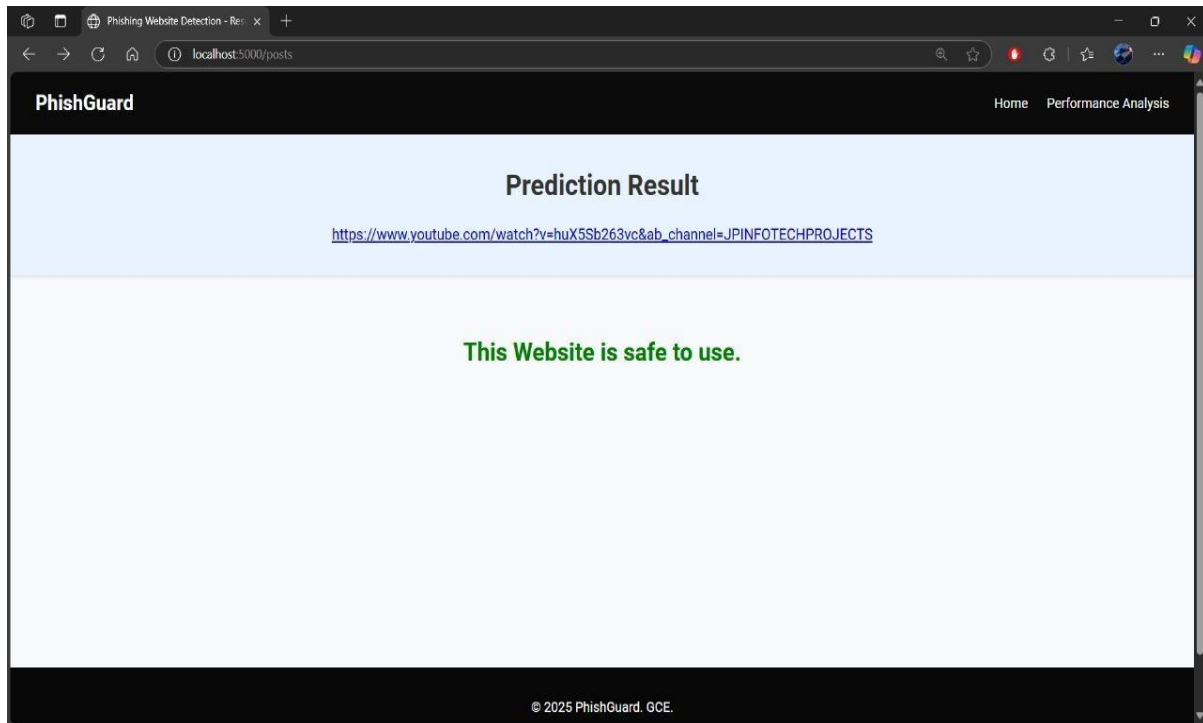


Figure 7.2: Prediction Result

The prediction result classifies a website as either "phishing" or "legitimate," often with a probability indicating the model's confidence in this assessment. This output is generated by analyzing the website's features using the trained machine learning model. These prediction results are generated by feeding the extracted features of the new website into the trained model, which then processes these features based on its learned parameters to produce the classification and the corresponding probability. The accuracy and reliability of these predictions are crucial and are rigorously evaluated during the model development process using appropriate metrics to ensure the system effectively distinguishes between genuine and deceptive websites.

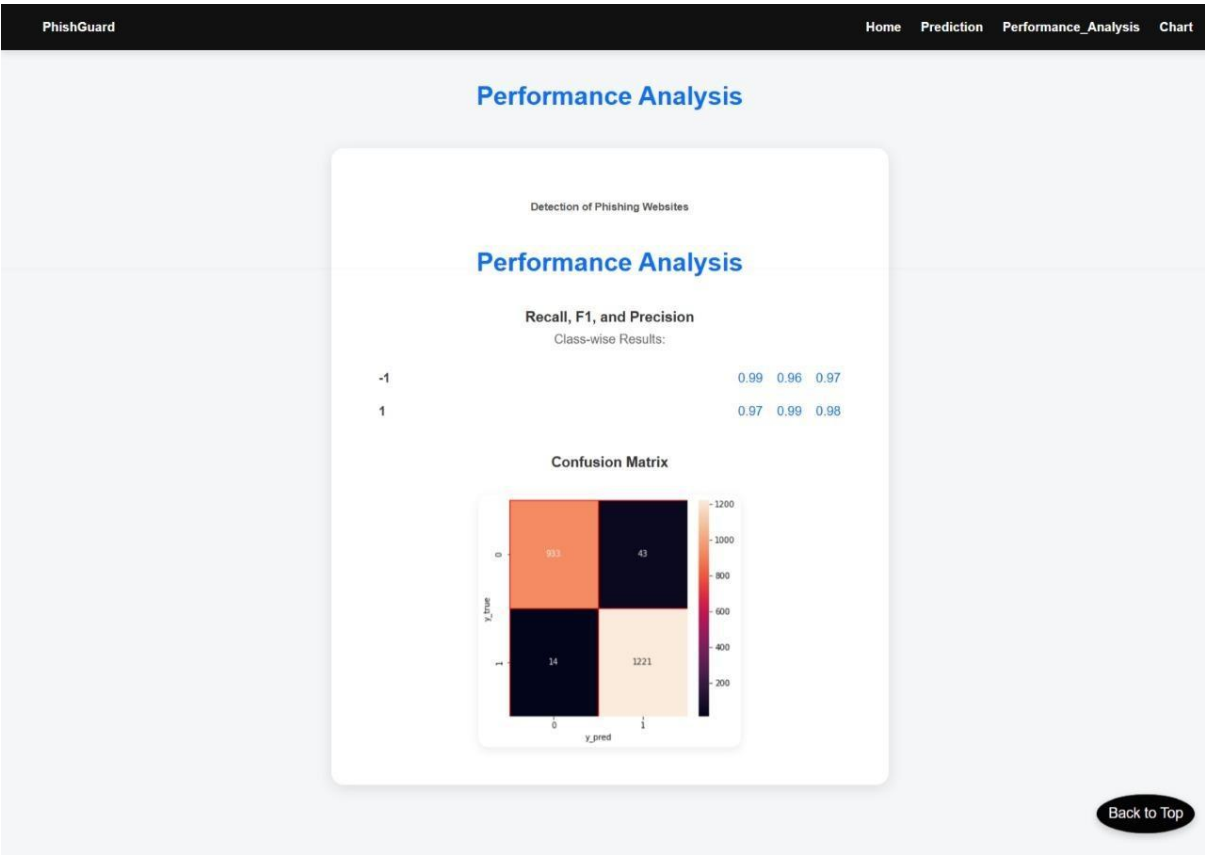


Figure 7.3: Performance Analysis

Shows the Confusion Matrix and Accuracy of the model.

Chart Analysis

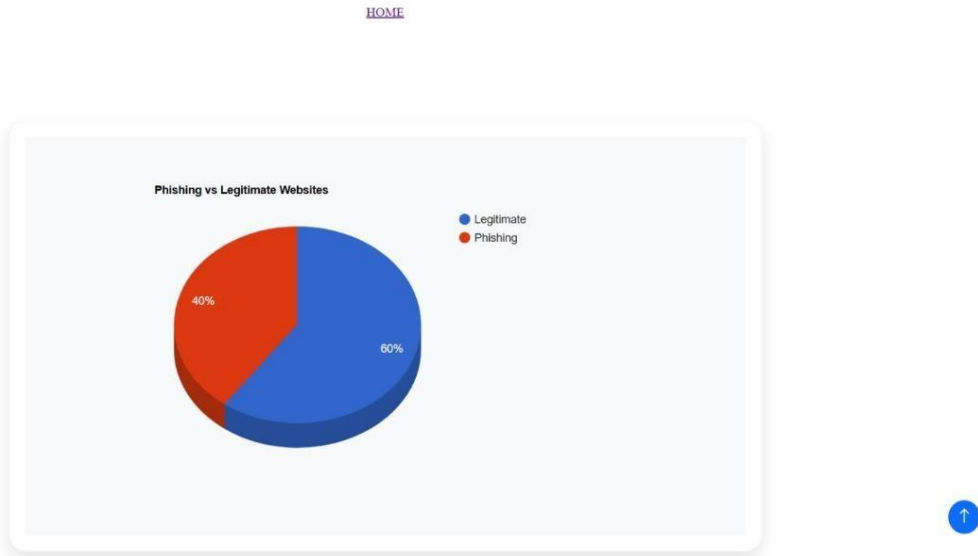
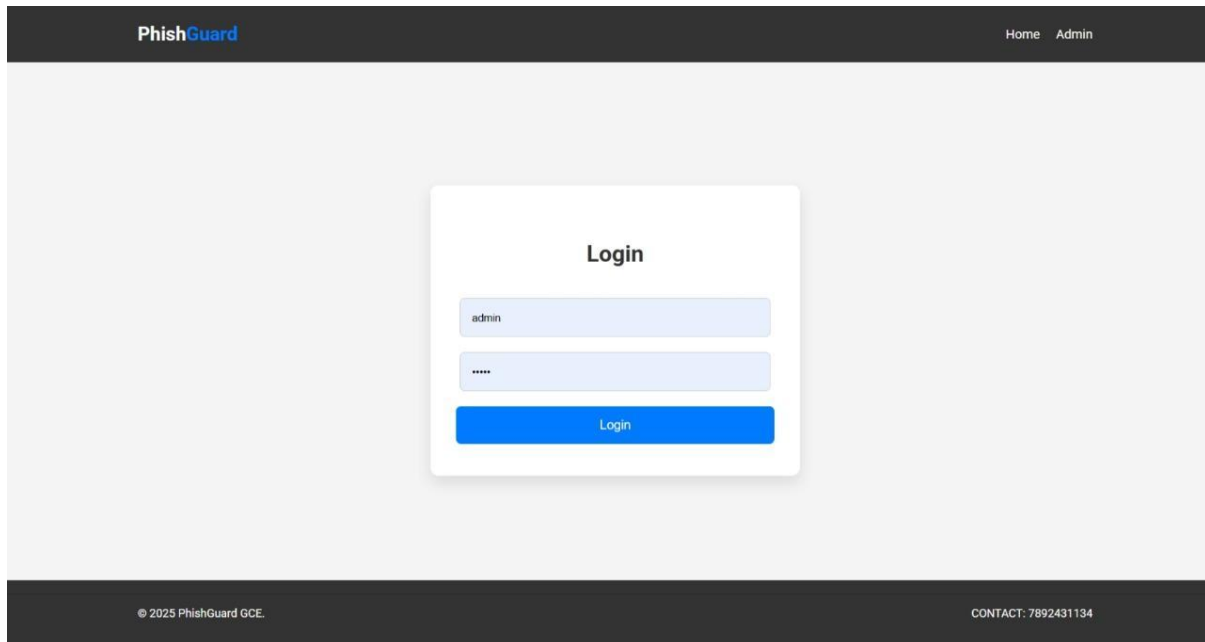


Figure 7.4: Chart Analysis

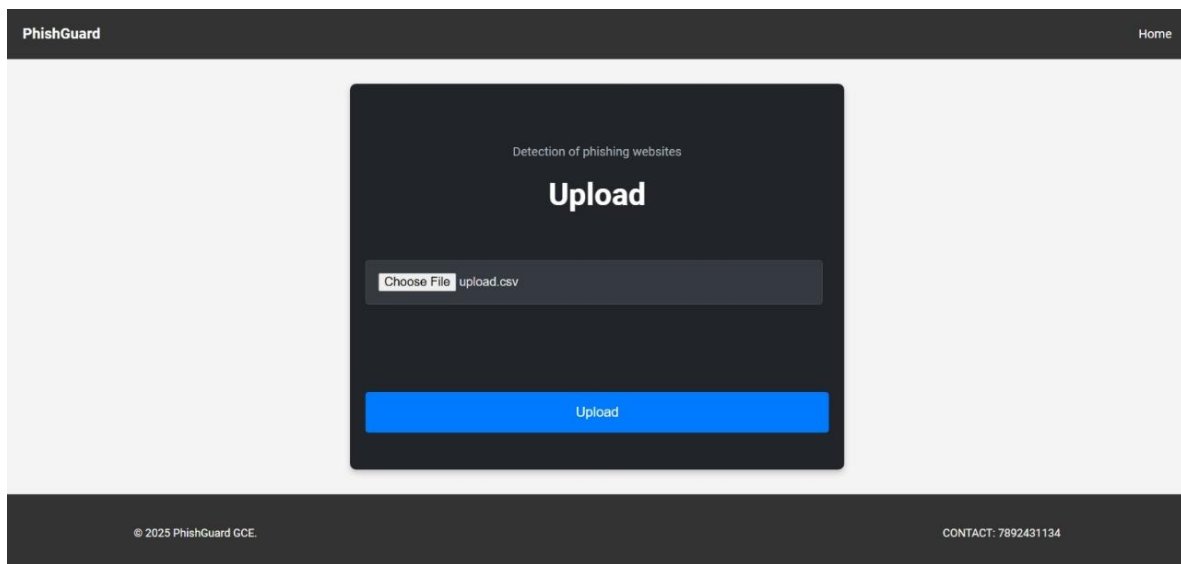
Shows how the model is trained meaning while training the model we used 60% of legit website and 40% of phishing website data.



The screenshot shows the PhishGuard login interface. At the top, a dark header contains the 'PhishGuard' logo on the left and 'Home' and 'Admin' links on the right. The main content area is light gray and features a white login card in the center. The card has the title 'Login' at the top, followed by two input fields: the first contains the text 'admin' and the second contains five asterisks. Below these fields is a blue 'Login' button. At the bottom of the page, a dark footer contains the copyright notice '© 2025 PhishGuard GCE.' on the left and 'CONTACT: 7892431134' on the right.

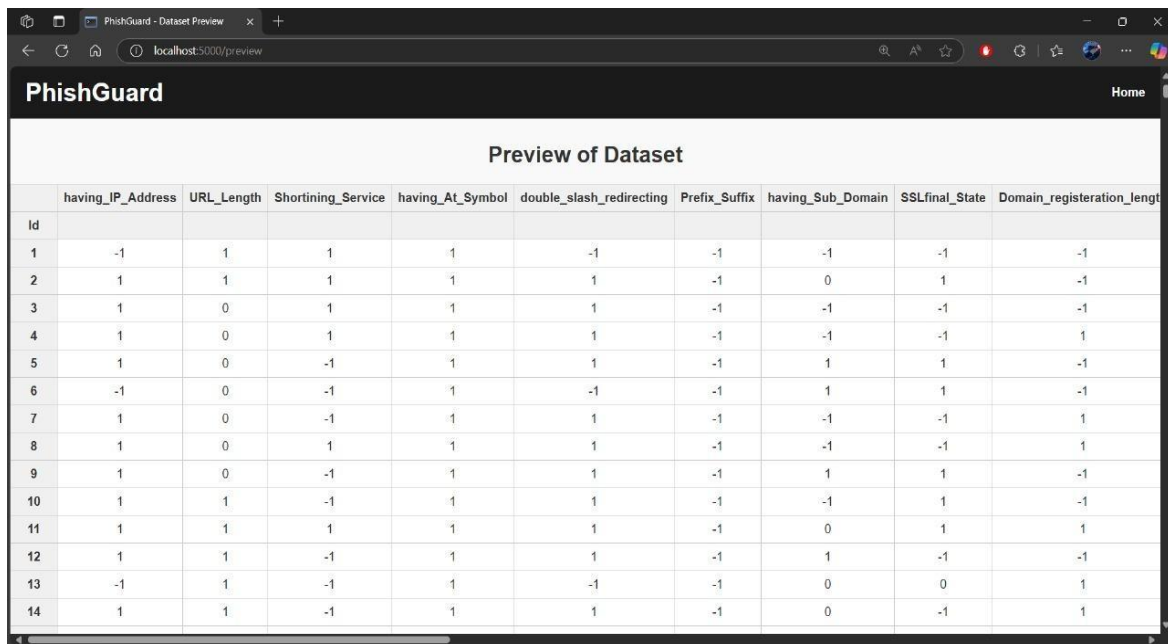
Figure 7.5: Login page.

In PhishGuard only admin has to login to update the Machine Learning Model and can upload new dataset as shown in below fig for better training to get more accurate results .



The screenshot shows the PhishGuard upload interface. At the top, a dark header contains the 'PhishGuard' logo on the left and a 'Home' link on the right. The main content area is light gray and features a dark gray upload card in the center. The card has the title 'Upload' at the top, preceded by the subtitle 'Detection of phishing websites'. Below the title is a file selection area showing 'Choose File' and 'upload.csv'. At the bottom of the card is a blue 'Upload' button. At the bottom of the page, a dark footer contains the copyright notice '© 2025 PhishGuard GCE.' on the left and 'CONTACT: 7892431134' on the right.

Figure 7.6: Upload page.

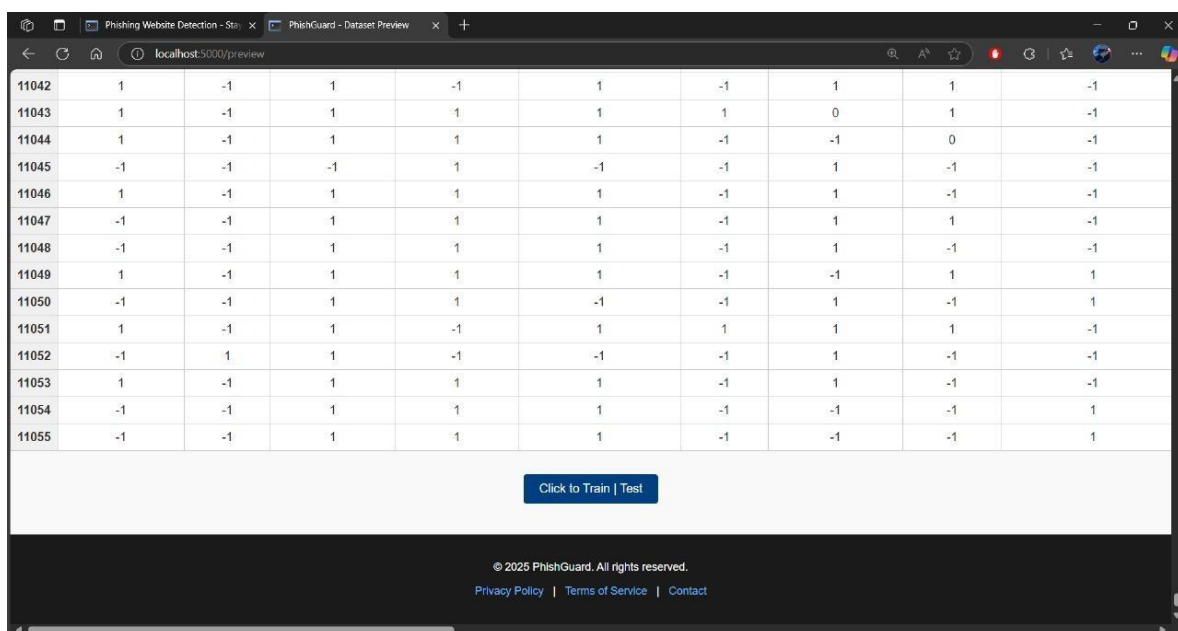


PhishGuard Home

Preview of Dataset

	having_IP_Address	URL_Length	Shortning_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSLfinal_State	Domain_registration_length
Id									
1	-1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	-1	0	1	-1
3	1	0	-1	-1	1	-1	-1	-1	-1
4	1	0	-1	-1	1	-1	-1	-1	1
5	1	0	-1	-1	1	-1	1	1	-1
6	-1	0	-1	-1	-1	-1	1	1	-1
7	1	0	-1	-1	1	-1	-1	-1	1
8	1	0	1	-1	1	-1	-1	-1	1
9	1	0	-1	-1	1	-1	1	1	-1
10	1	1	-1	-1	1	-1	-1	1	-1
11	1	1	1	-1	1	-1	0	1	1
12	1	1	-1	-1	1	-1	1	-1	-1
13	-1	1	-1	-1	-1	-1	0	0	1
14	1	1	-1	-1	1	-1	0	-1	1

Figure 7.7: Preview of Dataset.



Phishing Website Detection - Sta **PhishGuard - Dataset Preview**

localhost:5000/preview

11042	1	-1	1	-1	1	-1	1	1	-1
11043	1	-1	1	-1	1	1	0	1	-1
11044	1	-1	1	-1	1	-1	-1	0	-1
11045	-1	-1	-1	-1	-1	-1	1	-1	-1
11046	1	-1	1	-1	1	-1	1	-1	-1
11047	-1	-1	1	-1	1	-1	1	1	-1
11048	-1	-1	1	-1	1	-1	1	-1	-1
11049	1	-1	1	-1	1	-1	-1	1	1
11050	-1	-1	1	-1	-1	-1	1	-1	1
11051	1	-1	1	-1	1	1	1	1	-1
11052	-1	1	1	-1	-1	-1	1	-1	-1
11053	1	-1	1	-1	1	-1	1	-1	-1
11054	-1	-1	1	-1	1	-1	-1	-1	1
11055	-1	-1	1	-1	1	-1	-1	-1	1

[Click to Train | Test](#)

© 2025 PhishGuard. All rights reserved.
[Privacy Policy](#) | [Terms of Service](#) | [Contact](#)

Figure 7.8: Preview of Dataset.

After uploading the dataset the admin can preview the csv file and see how the data is divided on it.

localhost:5000 says
Training finished!

OK

11042	1	-1	1	1	1	-1	1	1	-1
11043	1	-1	1	1	0	1	-1	-1	-1
11044	1	-1	1	1	-1	0	-1	-1	-1
11045	-1	-1	-1	1	-1	-1	1	-1	-1
11046	1	-1	1	1	1	-1	1	-1	-1
11047	-1	-1	1	1	1	-1	1	1	-1
11048	-1	-1	1	1	1	-1	1	-1	-1
11049	1	-1	1	1	1	-1	-1	1	1
11050	-1	-1	1	1	-1	-1	1	-1	1
11051	1	-1	1	-1	1	1	1	1	-1
11052	-1	1	1	-1	-1	-1	1	-1	-1
11053	1	-1	1	1	1	-1	1	-1	-1
11054	-1	-1	1	1	1	-1	-1	-1	1
11055	-1	-1	1	1	1	-1	-1	-1	1

Click to Train | Test

Training model... please wait.

© 2025 PhishGuard. All rights reserved.
[Privacy Policy](#) | [Terms of Service](#) | [Contact](#)

Figure 7.9: Preview of Dataset

After previewing the dataset admin can train the model and it is ready for testing.

Home Upload

PHISHING URL DETECTION
URL Prediction

URL:

Enter URL

Check here

© 2025 All rights reserved.

Figure 7.10: Phishing URL Detection

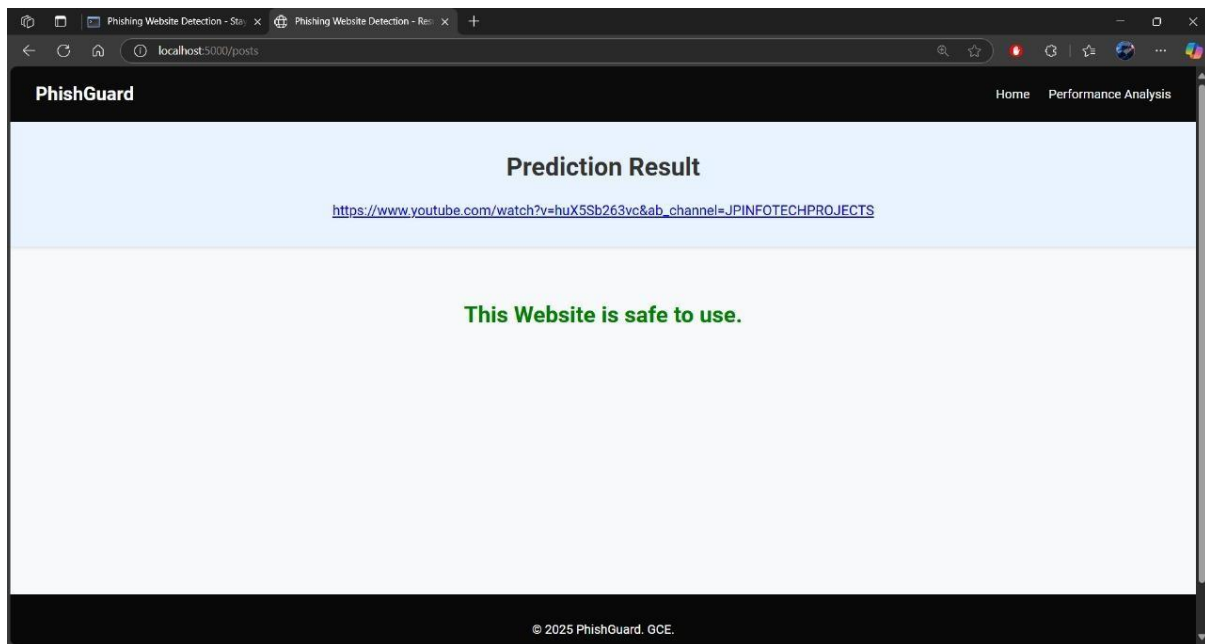


Figure 7.11: Prediction Result

After the testing is finished it will show the result as safe if the website is legitimate, If the website is fake it will show as “This Website is unsafe to use”.

CHAPTER 8

CONCLUSION

8.1 CONCLUSION

It is remarkable that a good anti-phishing system should be able to predict phishing attacks in a reasonable amount of time. Accepting that having a good anti-phishing gadget available at a reasonable time is also necessary for expanding the scope of phishing site detection. The current system merely detects phishing websites using Gradient Boosting Classifier. We achieved 97% detection accuracy using Gradient Boosting Classifier with lowest false positive rate.

8.2 Future Enhancements

Although the use of URL lexical features alone has been shown to result in high accuracy, phishers have learned how to make predicting a URL destination difficult by carefully manipulating the URL to evade detection. Therefore, combining these features with others, such as host, is the most effective approach .

For future enhancements, we intend to build the phishing detection system as a scalable web service which will incorporate online learning so that new phishing attack patterns can easily be learned and improve the accuracy of our models with better feature extraction.

CHAPTER 9

REFERENCES

1. Friedman, J. H. (2001). *Greedy Function Approximation: A Gradient Boosting Machine*. *Annals of Statistics*, 29(5), 1189–1232.
2. Natekin, A., & Knoll, A. (2013). *Gradient Boosting Machines, A Tutorial*. *Frontiers in Neurorobotics*, 7, 21.
3. Zhang, C., Sheng, S., Wardman, B., Warner, G., Cranor, L. F., & Hong, J. (2009). *Phishing Blacklists: An Empirical Study*. In CEAS 2009.
4. Khonji, M., Jones, A., & Iraqi, Y. (2013). *A Literature Review on Phishing Detection*. *IEEE Communications Surveys & Tutorials*, 15(4), 2091–2121.
5. Cranor, L., Xiang, G., Hong, J. I., & Rosé, C. P. (2011). *CANTINA+: A Feature-Rich Machine Learning Framework for Detecting Phishing Websites*. *ACM TISSEC*, 14(2).
6. Ye, C., Zhu, E., Liu, D., Liu, F., Wang, F., & Li, X. (2018). *An Effective Phishing Detection Model Using Neural Networks and Optimal Feature Selection*. In ISPA 2018.
7. Napoles, G., Falcon, R., Vanhoof, K., Köppen, M. (2016). *Detection of Dangerous URLs Using Machine Learning Algorithms*. In IEEE SSCI.
8. Hao, S., Invernizzi, L., Fang, Y., Kruegel, C., & Vigna, G. (2017). *Gossip: Detecting Malicious Domains from Mailing List Discussions*. In ASIA CCS.
9. Dou, Z., Khalil, I., Khreishah, A., Al-Fuqaha, A., & Guizani, M. (2017). *SoK: A Systematic Review of Software-Based Web Phishing Detection*. *IEEE Communications Surveys & Tutorials*.
10. Chiew, K. L., Tan, C. L., & Sze, S. N. (2014). *Phishing Website Detection Using URL-Assisted Brand Name Weighting System*. *IEEE ISPACS*.