



PHISHING URL DETECTION USING MACHINE LEARNING



A MINI PROJECT II REPORT

Submitted by

ABISHEK PS	(1901002)
ADITYA KUSHWAHA	(1901004)
ARJUN RU	(1901016)
ASHWIN BALAJI PL	(1901023)

*In partial fulfilment for the award of the degree
of*

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

SRI RAMAKRISHNA ENGINEERING COLLEGE

[Educational Service: SNR Sons Charitable Trust]

[Autonomous Institution, Reaccredited by NAAC with 'A+' Grade]

[Approved by AICTE and Permanently Affiliated to Anna University, Chennai] [ISO

9001:2015 Certified and All Eligible Programmes Accredited by NBA]

Vattamalaipalayam, N.G.G.O. Colony Post,

COIMBATORE - 641 022

JUNE 2022

ANNA UNIVERSITY: CHENNAI - 600 025

BONAFIDE CERTIFICATE

16CS266 - MINI PROJECT – II

This is to certify that Mini Project - II Report, “**PHISHINNG URL DETECTION USING MACHINE LEARNING**” is the bonafide work of “**ABISHEK PS, ADITYA KUSHWAHA, ARJUN RU, ASHWIN BALAJI PL**” who carried out the project under my supervision.

SIGNATURE

Dr. Grace Selvarani A

HEAD OF THE DEPARTMENT

Professor,
Computer Science and Engineering,
Sri Ramakrishna Engineering
College, Coimbatore - 641022

SIGNATURE

Dr. Vijayakumar R

SUPERVISOR

Assistant Professor,
Computer Science and Engineering,
Sri Ramakrishna Engineering
College, Coimbatore - 641022

Submitted for the Mini Project Viva-Voice Presentation held on 09/06/2022

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our gratitude to **Sri. D LAKSHMINARAYANASWAMY**, Managing Trustee, **Sri. R SUNDAR**, Joint Managing Trustee, SNR Sons Charitable Trust, Coimbatore for providing excellent facilities to carry out our project.

We express our deepest gratitude to our Principal, **Dr. N R ALAMELU, Ph.D.**, for her valuable guidance and blessings.

We are indebted to our Head of the Department, **Dr. A GRACE SELVARANI, Ph.D.**, Department of Computer Science and Engineering who modeled us both technically and morally for achieving great success in life.

We express our thanks to our Project Coordinator, **Mrs. S PRINCE SAHAYA BRIGHTY**, Assistant Professor (Sr. Grade), Department of Computer Science and Engineering for her great inspiration.

Words are inadequate to offer thanks to our respected guide. We wish to express our sincere thanks to **Dr. R VIJAYAKUMAR**, Assistant Professor (Sl. Grade), Department of Computer Science and Engineering, who gives constant encouragement and support throughout this project work and who makes this project a successful one.

We also thank all the staff members and technicians of our department for their help in making this project a successful one.

ABSTRACT

Most of the financial frauds are caused by phishing attacks. It is one of the most dangerous threats to online accounts and data, because these kinds of exploits hide behind the guise of being from a reputable company or individual and use elements of social engineering to make victims far more likely to fall for the scam.

The important thing is to exercise common sense and a good deal of caution about any message that the user receives which looks faintly suspicious and has tell-tale signs like spelling mistakes or odd phrasing, errors that malware authors often make, urges you to do something ‘right now’, or has a link or attachment which seems even remotely dodgy. A message which comes from a trusted source such as higher officials in the workplace or from a reputed organization or even close friends, where their email address or details could easily have been spoofed.

The model which we have proposed is based on machine learning where the features are extracted from the input URL based on its characteristics and behaviour. The machine learning model used here is extreme gradient boosting or XGBoost, a supervised machine learning algorithm which uses more accurate approximations to find the best tree model using the training dataset to predict the URL is phishing or legitimate as target variable.

LIST OF FIGURES

Figure / Table	TITLE	Page no.
4.4.1	Dataset Generation	14
4.4.2	Phishing Detection	14
5.1	Test Accuracy	16
5.2	Train Accuracy	1
5.3	Accuracy Score	17
7.2.1	URL Entry	30
7.2.2	Submitting Legitimate URL	30
7.2.3	Redirection – Not Phishing	30
7.2.4	Submitting fake/phishing URL	30
7.2.5	Warning	31
7.2.6	Prediction – Phishing	31
7.2.7	Continue with RISK	31

LIST OF ABBREVIATIONS

ML	-	MACHINE LEARNING
SVM	-	SUPPORT VECTOR MACHINE
XG	-	EXTREME GRADIENT
KNN	-	K – NEAREST NEIGHBOUR
URL	-	UNIFORM RESOURCE LOCATOR
DNS	-	DOMAIN NAME SYSTEM

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
	ABSTRACT	i
	LIST OF FIGURES	ii
1	INTRODUCTION	
	1.1 Phishing	1
	1.2 Phishing Threats	2
	1.3 Phishing Attacks	3
2	LITERATURE REVIEW	
	2.1 An Enhanced Blacklist Method to Detect Phishing Websites	4
	2.2 Malicious URL Detection using Machine Learning: A Survey	4
	2.3 Detection of URL based Phishing Attacks using Machine Learning	5
	2.4 Limitations	5
3	SYSTEM DESCRIPTION	
	3.1 Hardware Requirements	7
	3.2 Software Requirements	7
	3.3 Description	8
	3.4 Existing Solutions	8
4	MODULE DESCRIPTION	
	4.1 Dataset	10
	4.2 Feature Extraction	10
	4.3 Machine Learning Models	12
	4.4 Flowcharts	13
	4.5 Merits and Demerits	15
5	RESULTS	16

6	CONCLUSIONS AND FUTURE SCOPE	
	6.1 Conclusion	18
	6.2 Future Scope	18
7	APPENDICES	
	7.1 Sample Code	19
	7.2 Snapshots	30
8	REFERENCES	32

CHAPTER 1

INTRODUCTION

1.1 PHISHING

The world is evolving with cutting edge technology, the people around the world are interconnected with each other through the internet with the help of electronic devices and smart gadgets. There are about 5 billion active internet users worldwide. Due to the pandemic that started at the end of 2019, many traditional industries have shifted from offline mode to online. With the increase of internet usage and online services, cyber-attacks escalated around the world.

One of the majors cyberattacks faced by the people now is phishing, where the attackers use illegitimate websites to obtain victims data and use it illegally. Phishing can easily be imposed using the email and by other modes of communication. The attackers grab data easily and get valuable data. The phishing attacks lead to malware infections, loss of data, identity theft etc. The data in which these cyber criminals are interested is the crucial information of a user such as passwords, OTP, credit/ debit card details, sensitive data related to business, medical data, confidential data. Even for cautious users it's sometimes difficult to detect phishing attacks. Properly designing and deploying a Phishing URL will help block the intruders.

The attackers attempt to gain the users trust by using the same user interface, almost same URL, and cloned websites. An efficient way to detect and prevent phishing is by using an automated approach. Machine learning helps to achieve this. It is the subset of artificial intelligence used by computer systems which uses algorithms, statistical models, patterns, and inferences to complete certain tasks without human intervention. Machine learning provides the ability to automatically learn and improve from its experience without being overtly programmed. In our proposed system the algorithm goes through the process of learning with the training datasets. The major advantage of the algorithm is to allow the system to learn and decide automatically whether the website is phishing or legitimate. In terms of machine learning, a larger number of data will increase the accuracy of the model significantly. So, the prediction of phishing websites will deal with larger dataset to train the model for better accuracy.

A phishing URL and the parallel page have many features which are different from the malignant URL. The domain name of the phishing URL can be a very long and confusing name of the domain. This is very easily visible. Sometimes they use the IP address instead of using the domain name. In some cases, they can also use a shorter domain name which will not be relevant to the

original legitimate website. Apart from the URL based feature of phishing detection there are many different features which can also be used for the detection of Phishing websites namely the Domain-Based Features, Page-Based Features and Content-Based Features.

1.2 PHISHING THREATS

Phishing is a term which refers to the practice of tricking Internet users to reveal personal or confidential information. Phishing was not reported until 1996 when it was first mentioned by a popular hacker newsletter after an attack. Since then, there has been an exponential increase in phishing attacks, with it becoming one of the most prevalent methods of cybercrime.

India among top 3 Asian nations affected by phishing cyber-attacks. Asia recorded an increase of 15 per cent in average cost of a phishing attack than the previous years. According to the Ministry of Home Affairs, 2,00,000+ cybercrime cases have been reported in one year with more than 90% of them being financial frauds.

According to Verizon's 2021 Data Breach Investigations Report (DBIR), phishing is the top "action variety" seen in breaches in the last year and 43% of breaches involved phishing and/or pretexting. Increase in the phishing attacks compared to previous year's reports. In 2019, phishing played a part in 78% of all Cyber-Espionage incidents and 87% of all installations of malware in the first quarter of 2019. In the earlier report by Widup (2018), it is reported that 78% of people didn't click a single phishing link all year, meaning that 22% of the people did click one and were victims of phishing attacks. Moreover, only 17% of these phishing campaigns were reported by users. It is also emphasised that even though training can reduce the number of incidents, phish happens.

Only a single message or email is needed to compromise an entire organisation, protection against it should be taken seriously. Cyber-criminals use phishing attacks to either harvest information or steal money from their victims through deceiving them with a reflection of what would seem like a regular email or website. By redirecting the victim to their disguised website, they can see everything the victim inserts in any forms, login pages or payment sites. Cyber-criminals either copy the techniques used by digital marketing experts or take advantage of the fuss created by viral events to guarantee a high click rate. Regular phishing attacks are usually deployed widely and are very generic, such that they can be deployed to target as many people as possible. A Spear Phishing attack, instead, targets a specific individual, but requires that information be gathered about the victim prior to crafting a successful spear-phishing email. A

more advanced version of this attack is a Whaling attack, which specifically targets a company's senior executives to obtain higher-level access to the organisation's system. Targeted phishing attacks are increasingly gaining popularity because of their high success rates.

1.3 PHISHING ATTACKS

Homograph spoofing is an attack which depends on the replacement of characters in a domain name with other visually similar characters. Characters from other alphabets such as Greek have also been used in the past for such attacks. The Greek o character is visually indistinguishable from the English o even though their ASCII codes are different and would redirect to different websites. Content polymorphism is addressed, as well, using visual similarity analysis of the contents.

Typo squatting targets common typographic errors in domain names. For example, an attacker could use the domain amazoon.in to target users who incorrectly type amazon.in or to trick them into clicking on a regular link, combat this issue using the K-Means Clustering Algorithm to observe the lexical differences between benign and malicious domains to extract features, and propose a majority voting system that takes into consideration the outputs of five different classification algorithms.

Sound squatting leverages on the use of words that sound alike (homophones) show that for a domain www.amazon.in, an adversary may use dot-omission typing errors such as wwwamazon.in, missing-character errors like www.amzon.in, character permutation errors like www.amaozn.in, character replacement errors and character insertion errors. They illustrate how they used Alexa's top one million domain list to create and register their sound squatting domains, measuring the traffic from users accidentally visiting them. Through research have proven the significance of homophone confusion through abuse of text-to-speech software when tackling the issue of squatting.

Combo squatting is different from other approaches as it depends on altering the target domain by adding familiar terms inside the URLs. An example of this technique would be indianbank.com or amazon-customer-support.com. Research performed by Kintis, shows a steady increase in the use of combo squatting domains for phishing as well as other malicious activities over time. It is also reported that combo squatting domains are more resilient to detection than typing error and that most of the combo squatting domains they were monitoring remained active for extended periods, thus suggesting that the measures set in place to counter these are inadequate.

CHAPTER 2

LITERATURE REVIEW

2.1 AN ENHANCED BLACKLIST METHOD TO DETECT PHISHING WEBSITES (*Routhu Srinivasa Rao, Alwyn Roshan Pais*)

Existing anti-phishing techniques like whitelist or blacklist detect the phishing sites based on the database of approved and unapproved URLs. Most of the current phishing attacks are replicas or variations of other attacks in the database. In this paper, we propose an enhanced blacklist method which uses key discriminate features extracted from the source code of the website for the detection of phishing websites. The focus of our work is to detect the phishing sites which are replicas of existing websites with manipulated content. Each phishing website is identified with a unique fingerprint which is generated from the set of proposed features. We used Simhash algorithm to generate fingerprint for each website. The features used for calculating fingerprint are filenames of the request URLs (js, img, CSS, favicon), pathnames of request URLs (CSS, scripts, img, anchor links), and attribute values of tags (H1, H2, div, body, form). Our experimentation detected 84.36% of phishing sites as replicas of other phishing websites with manipulated content while maintaining zero false positive rate. The proposed method is like that of traditional blacklist with an advantage that it can detect replicated and manipulated phishing sites efficiently.

2.2 Malicious URL Detection using Machine Learning: A Survey (*Doyen Sahoo, Chenghao Liu, Steven C.H. Hoi*)

Malicious URL, a.k.a. malicious website, is a common and serious threat to cybersecurity. Malicious URLs host unsolicited content (spam, phishing, drive-by exploits, etc.) and lure unsuspecting users to become victims of scams (monetary loss, theft of private information, and malware installation), and cause losses of billions of dollars every year. It is imperative to detect and act on such threats in a timely manner. Traditionally, this detection is done mostly through the usage of blacklists. However, blacklists cannot be exhaustive, and lack the ability to detect newly generated malicious URLs. To improve the generality of malicious URL detectors, machine learning techniques have been explored with increasing attention in recent years. This article aims to provide a comprehensive survey and a structural understanding of Malicious URL Detection techniques using machine learning. We present the formal formulation of Malicious URL Detection as a machine learning task and categorize and review the contributions

of literature studies that addresses different dimensions of this problem (feature representation, algorithm design, etc.). Further, this article provides a timely and comprehensive survey for a range of different audiences, not only for machine learning researchers and engineers in academia, but also for professionals and practitioners in cybersecurity industry, to help them understand the state of the art and facilitate their own research and practical applications. We also discuss practical issues in system design, open research challenges, and point out some important directions for future research.

2.3 DETECTION OF URL BASED PHISHING ATTACKS USING MACHINE LEARNING (*Sophiya Shikalgar, Dr. Sawarkar*)

A grift attempt to get sensitive and personal information like password, username, and bank details like credit/debit card details by masking as a reliable organization in electronic communication. The phishing website will appear the same as the legitimate website and directs the user to a page to enter personal details of the user on the fake website. Through machine learning algorithms one can improve the accuracy of the prediction. The proposed method predicts the URL based phishing attacks based on features and gives maximum accuracy. This method uses uniform resource locator (URL) features. We identified features that phishing site URLs contain. The proposed method employs those features for phishing detection. The proposed systems predict the URL based phishing attacks with maximum accuracy. We shall talk about various machine learning, the algorithm which can help in decision making and prediction. We shall use more than one algorithm to get better accuracy of prediction. Different machine learning algorithms are used in the proposed system to detect URL based phishing attacks. The hybrid algorithm approach by combining the algorithms will increase accuracy.

2.4 LIMITATIONS

There exist many techniques which rely on either whitelist or blacklist information. These techniques use whitelist or blacklist information of websites for the calculation of similarity score between suspicious and target website. The whitelist information is extracted from URL, images, layout, styles, digital certificates.

The detection using machine learning uses about nine features for the process and different machine learning algorithms are used in the proposed

system to detect URL based phishing attacks. The hybrid algorithm approach by combining the algorithms will increase accuracy.

Blacklisting information is extracted from URL, text, and DOM tags. These do check the outline characteristics of a URL. Existing techniques either consider HTML plaintext, tags or layouts, images for the extraction of characteristics of website. These features are used for generating blacklist of fingerprints using Simhash algorithm. Due to the property of Simhash algorithm, the fingerprints of near duplicate documents would differ with a small number of bits. Hence, it is fast and robust to manipulation.

The machine learning model uses about 9 features which are a basic set of features which are easily identifiable by anyone with a minimal knowledge about the phishing attacks.

CHAPTER 3

SYSTEM DESCRIPTION

3.1 HARDWARE REQUIREMENTS

- **COMPUTER**

A computer with at least of intel Celeron processor or AMD Ryzen processor and minimum of 4 gigabits of RAM

- **NETWORK INTERFACE CARD**

A network interface card is required for the internet connection. Ethernet or Wi-Fi or any tethered wired medium.

3.2 SOFTWARE REQUIREMENTS

- **VISUAL STUDIO CODE**

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

- **PYTHON**

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically typed, and garbage collected. It supports multiple programming paradigms, including structured, object-oriented, and functional programming.

Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data. The implementation of machine learning algorithm is common when using python.

Python community has developed many modules to help programmers implement machine learning. Modules like numpy, scipy and scikit-learn modules are most common in machine learning. It can be installed using simple commands.

- **WEB BROWSER**

A web browser is an application software for accessing the World Wide Web. When a user requests a web page from a particular website, the web browser retrieves the necessary content from a web server and then displays the page on the user's device. A web browser connects to a website's server and display its web pages, a user must have a web browser installed. Web browsers are used on a range of devices, including desktops, laptops, tablets, and smartphones. The most used browsers are Google Chrome, Safari, Firefox, Opera.

3.3 DESCRIPTION

The process of identifying phishing URLs by automated methods is expected to reduce phishing attacks. Machine Learning helps in automating the prediction. The model should be trained in such a way, so that, it can predict with high accuracy. A machine learning approach predicts the URL whether it is a phishing or not. It is achieved by the extracted features from the input URL. A list of features in to be extracted from the URL from passive methods. Few passive methods include third party involvement such as whois, Alexa, PageRank, Google Index, etc. There are about 30 different features extracted from the given URL. The machine learning model used is supervised learning algorithm which requires a label that weather the referred segment of values are of what kind. From the repositories such as phishtank, openphish existing phishing and legitimate websites were extracted and formed a dataset with different features. The purpose of the proposed methodology is to obtain a high accuracy in detecting phishing sites over the Internet.

3.4 EXISTING SOLUTIONS

There are a few approaches for the issue. Many techniques rely on either whitelist or blacklist information. These techniques use whitelist or blacklist information of websites for the calculation of similarity score between suspicious and target website. The whitelist information is extracted from URL, images, layout, styles, digital certificates etc.

The proposed layout similarity approach where layout of suspicious site is compared with whitelist of all trusted layouts for the calculation of similarity score. If the score is greater than a threshold, then it is classified as phishing else legitimate. It compares the styles of suspicious and whitelisted styles for

identifying the status of website. If the similarity score is above the threshold, then the site is classified as phishing. Visual similarity-based techniques maintain list of whitelisted images and the suspicious website image is compared with the database. If the similarity is above the threshold, then it is classified as phishing.

The Blacklist information is extracted from URL, text, and DOM tags. Some of the techniques which use blacklist information are listed below. Existing techniques either consider HTML plaintext, tags or layouts, images for the extraction of characteristics of website. Our work differs from the above as we rely on the noisy part of HTML rather than content of the website. We consider attribute values of HTML elements and filenames of request URLs. These features are used for generating blacklist of fingerprints using Simhash algorithm. Due to the property of Simhash algorithm, the fingerprints of near duplicate documents would differ with a small number of bits. Hence, it is fast and robust to manipulation. Our technique can also be employed in whitelist-based methods.

The detection using machine learning uses about nine features for the process and different machine learning algorithms are used in the proposed system to detect URL based phishing attacks. The hybrid algorithm approach by combining the algorithms will increase accuracy.

CHAPTER 4

MODULE DESCRIPTION

4.1 FLOW CHARTS

4.1.1 DATASET

This data is collected from various sources from the internet such as phishtank and openphish. Since the proposed model is supervised machine learning algorithm, we need to know the results or labels earlier. The URL data undergoes feature extraction, and the extracted features forms the dataset.

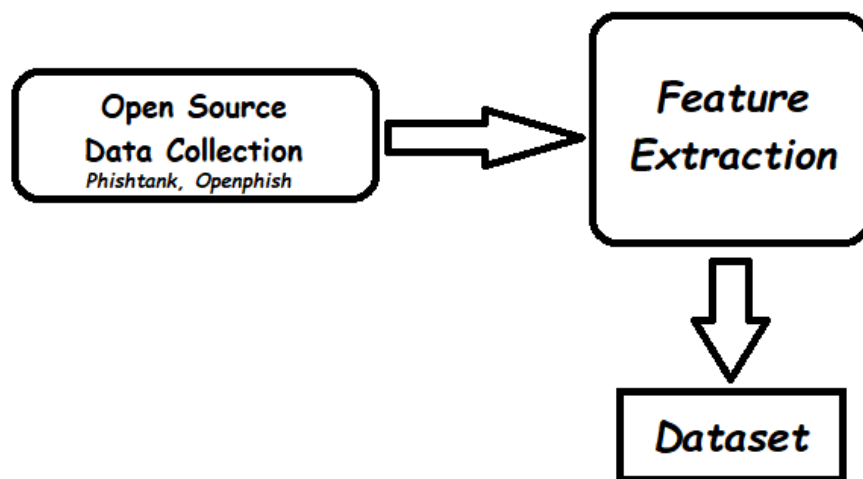


Fig 4.1.1 Dataset generation

4.1.2 DATA FLOW DIAGRAM

The data flow diagram is a graphical representation of the flow of the data. In this, the diagram says the data flow after the complaint registration. The user input gets the data and send to the admin's view so that, improper complaints and highly confidential complaints will not be shown to the users. Then the admin table displays all the data. From there the votes are monitored. Finally, the admin verifies the complaint and neither forward nor delete it.

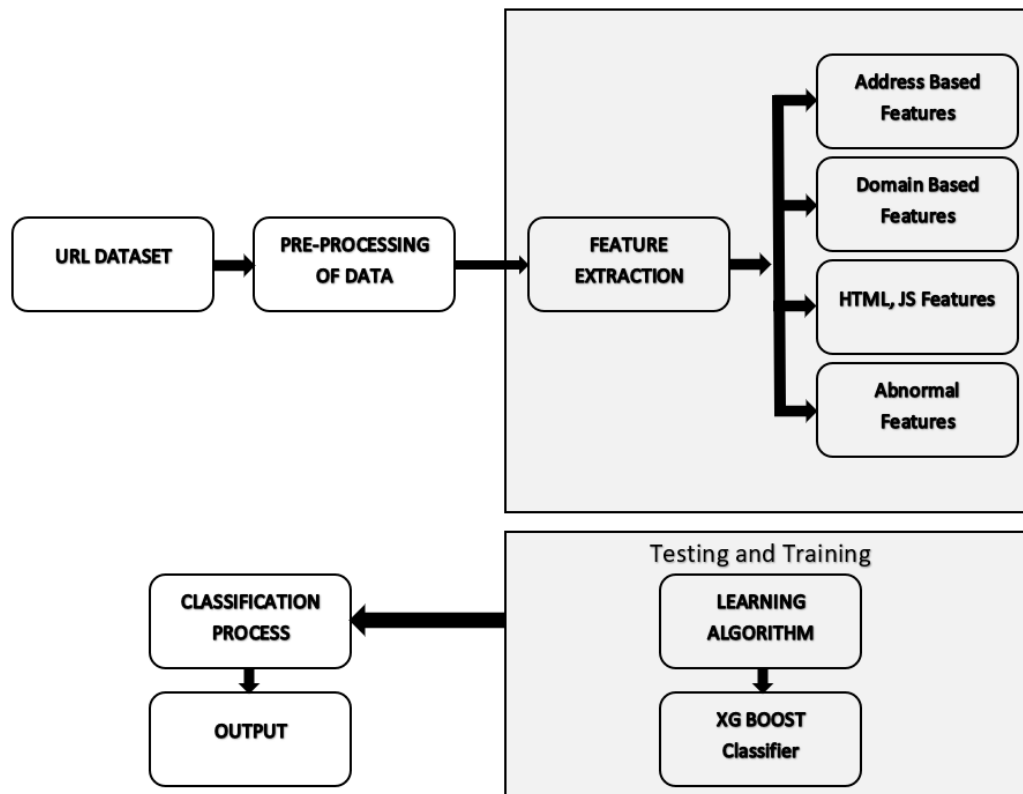


Fig 4.1.2 Phishing Detection

4.2 DATASET

The quality of the prediction of a ML algorithm is strongly related to the quality of its training dataset. The Machine Learning approach requires a supervised learning algorithm, and therefore the samples need to be labelled as either benign or malicious. Firstly, the raw data of phishing and legitimate URLs are extracted from the official websites of phish tank and open phish, PageRank and Alexa.

4.3 FEATURE EXTRACTION

The URLs undergo several processes and there are around 30 characteristics of phishing websites which are used to differentiate it from legitimate ones. Each category has its own characteristics of phishing attributes and values are defined.

- a) Address based features
- b) Abnormal features
- c) HTML and JS based features
- d) Domain based features

a) Address bar-based features:

i. Using IP address: If the domain of the URL of the suspected web page contains IP address, then we take it as a phishing page.

ii. Long URL to hide suspicious part: It has been a common observance that phishing web pages usually have long URLs that attempt to hide malicious URL fragments from the user. We take the assumption that a web page with a long URL is necessarily a phishing or suspicious site. In the event the assertion fails, i.e., for a legitimate web page with valid long URLs, the absence of other phishing attributes on the web page will balance the wrong assumption and correctly classify a legitimate web page as non-phishing.

iii. Use of URL shortening services: A shortened URL hides the real URL behind a redirection hop. A web page that uses a URL shortening service such as Tiny URL is highly suspicious and is likely to be a phishing attempt. Therefore, we set the rule that if the URL has been shortened using a URL shortening service, then it is a phishing page and legitimate otherwise.

iv. Use of "@" symbol: Needs verification The "@" symbol is a reserved keyword according to Web standards. So, the presence of "@" in a URL is suspicious and the web page is taken as phishing and legitimate otherwise.

v. Redirection with "///": The presence of "///" in the URL path indicates the page will be redirected to another page. If the position of "///" in the URL is greater than seven, then it is a phishing site and legitimate otherwise.

vi. Adding prefix or suffix separated by "-" to the domain: Phishers tend to add a prefix or suffix to the domain with "-" to give the resemblance of a genuine site.

vii. Sub domains and multi sub domains: If a URL has more than three dots in the domain part then it is considered as a phishing site and legitimate otherwise.

b) Abnormal based features:

viii. Request URL: A legitimate site usually has external page objects such as images, animations, files, etc. be accessed by a request URL which shares the same domain as the web page URL. We classify sites which fail this rule as phishing.

ix. URL portion of anchor tag: We check if the domain in the URL portion of all anchor tags match the main URL of the page and if the anchor tag has only URL fragments or JavaScript functions.

x. Links in <meta>, <script> and <link> tags: We check if the domain of the links in the <meta>, <script> and <link> tags match the domain in the mail URL.

xi. Server Form Handler (SFH): When a form is submitted, some valid action must be taken. So, if the action handler of a form is empty or "about: blank" or if the domain of the action URL is different from the domain of the main URL, then it is taken as a phishing site.

xii. Submitting Information to Email: If the webpage contains a "mailto:" function then it is taken as a phishing site and legitimate otherwise.

c) HTML and JavaScript based features:

xiii. Status bar customization: Phishers can modify the status bar using JavaScript to show a legitimate URL.

By analysing the "on Mouseover" events in the web page we can determine if such a modification has occurred.

xiv. Disabling right click option: Phishers can disable the right click option to prevent the user from checking the source code of the page. This is verified by analysing the source code.

xv. Using pop-up window: Legitimate sites rarely ask for user info on a pop-up window, whereas phishing sites generally use pop-up windows to get user info.

xvi. Iframe redirection: Phishers also use Iframe tags with invisible borders to get user info and redirect to the original site. We analyse the source code to check if Iframe tags are used.

d) Domain based features:

There are about 14 domain-based features such as Alexa PageRank, google page index, different who-is results, etc.

The specified characteristics are extracted for each URL and valid ranges of inputs are identified. These values are then assigned to each phishing website risk. For each input the values range from 0, 1 or -1 based on the unique features. The phishing attributes values are represented with number -1 and 1 which indicates the attribute is present or not.

4.4 MACHINE LEARNING MODELS

Depending on the application and nature of the dataset used, we can use any classification algorithms. As there are different applications, we cannot

differentiate which of the algorithms are superior or not. Each classifier has its own way of working and classification. After this the data is trained, we shall apply a relevant machine learning algorithm to the dataset. The machine learning algorithms used are Decision Tree, Random Forest, K Nearest Neighbours Classifier, Extreme Gradient Boost, and Support Vector Machine.

Decision Tree: This is the most powerful and popular tool for classification and prediction. This is a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

Random Forest: This classification algorithm is similar to ensemble learning method of classification. The regression and other tasks work by building a group of decision trees at training data level and during the output of the class, which could be the mode of classification or prediction regression for individual trees. This classifier accuracy for decision trees practice of overfitting the training data set.

Support vector machine (SVM): This is also one of the classification algorithms which is supervised and is easy to use. It can be used for both classification and regression applications, but it is more famous to be used in classification applications. In this algorithm each point which is a data item is plotted in a dimensional space, this space is also known as n dimensional plane, where the 'n' represents the number of features of the data. The classification is done based on the differentiation in the classes, these classes are data set points present in different planes.

K-Nearest Neighbor (KNN): K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. This algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified. It can be used for Regression as well as for Classification.

XGBoost: Recently, the researchers have come across an algorithm "XGBoost", and its usage is very useful for machine learning classification. It is very much fast, and its performance is better as it is an execution of a boosted decision tree. This classification model is used to improve the performance of the model and to improve the speed.

These models predicted the accuracy of the detection of the phishing URL and got desired results. The testing data and evaluating the prediction with different machine learning algorithms and with the extended number of features gives more accuracy which is a bit more than other existing systems.

4.5 MERITS AND DEMERITS

MERITS

The use of machine learning methods reduces dependency, cost, and license on third-party external software. Better insights can be provided into online behaviour of inputs. Real-time protection for the users who access malicious websites or click on phishing links and detect and prevent against unknown phishing attacks, as new patterns are created by attackers in the next level of intelligence on top of signature-based prevention techniques and blacklists. Centralized solution implemented org-wide and no dependency on client-side software move from real-time to proactive. The Feature selection highly improves the accuracy score after implementation. Use of feature selection reduces computational time; this can be used to build a rule-based system with associative rules to classify URLs. As it automatically trains classifiers to determine web page similarity from CSS layout features, which does not require human expertise. Machine Learning models gives high accuracy score, and highlights features that are necessary to extract.

DEMERITS

The machine learning approach may be unable to detect the attachment of DNS spoofs to legitimate web pages. High False positive rate may occur and it does not do well with a random dataset without applying a supervised resample filter which contains the label column which is already known. It uses 30 features for detection and limited dataset of 11055 instances. Method is lightweight as it only takes a few classes of features, CSS structure and JS structures. Limited by the size of the dataset and distribution of samples.

CHAPTER 5

RESULTS

We have got the desired results of testing whether the site is phishing or not by using five different classifiers. The partial implementation and the graphs of the results are given below. The project will work in such a way that when a URL is given as input to the system, it undergoes the feature extraction process specified in the Feature Extraction module. The extracted features will then go into the machine learning algorithm and predict whether the input URL is a phishing URL or a legitimate one.

The algorithm which predicts with the most accuracy is the one which we need. To find the best performing algorithm, we need the results of different algorithms. Let's see the performance of each algorithm used in the project. As this is a supervised machine learning model, the predicted values are already known so that we can find the exact accuracy of the models.

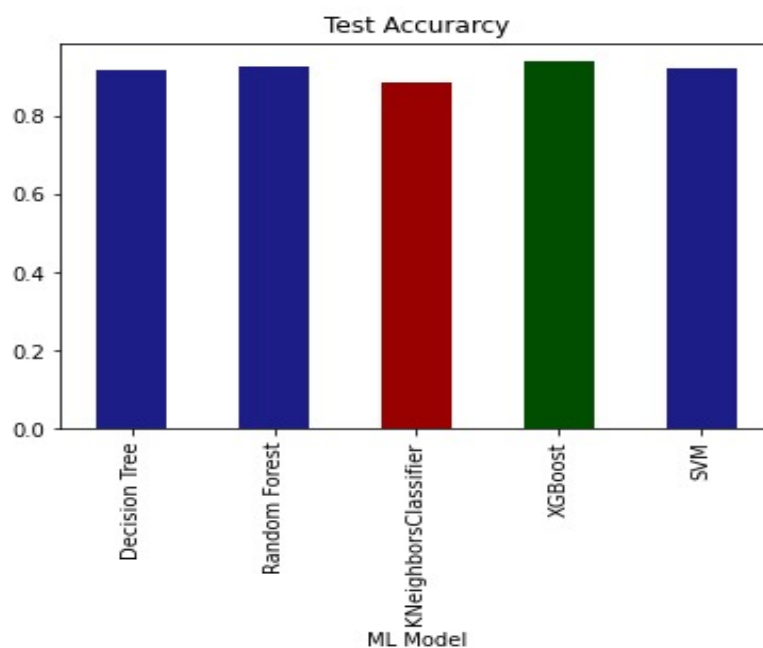


Fig 5.1 Test Accuracy

The Decision Tree algorithm shows train accuracy of 99.3% and test accuracy of 91.9%. The Random Forest algorithm which has the train accuracy of 93.5% and test accuracy of 91.9%. The K-Nearest Neighbor algorithm has the train accuracy of 99.9% and test accuracy of 88.2% while the Extreme Gradient Boosting algorithm has the train accuracy of 99.7% and test accuracy of 94.0% and The Support Vector Machine algorithm which has the train accuracy of 92.7% and

test accuracy of 92.2%. The following graph 1.1 and 1.2 specifies the performance of the different algorithms.

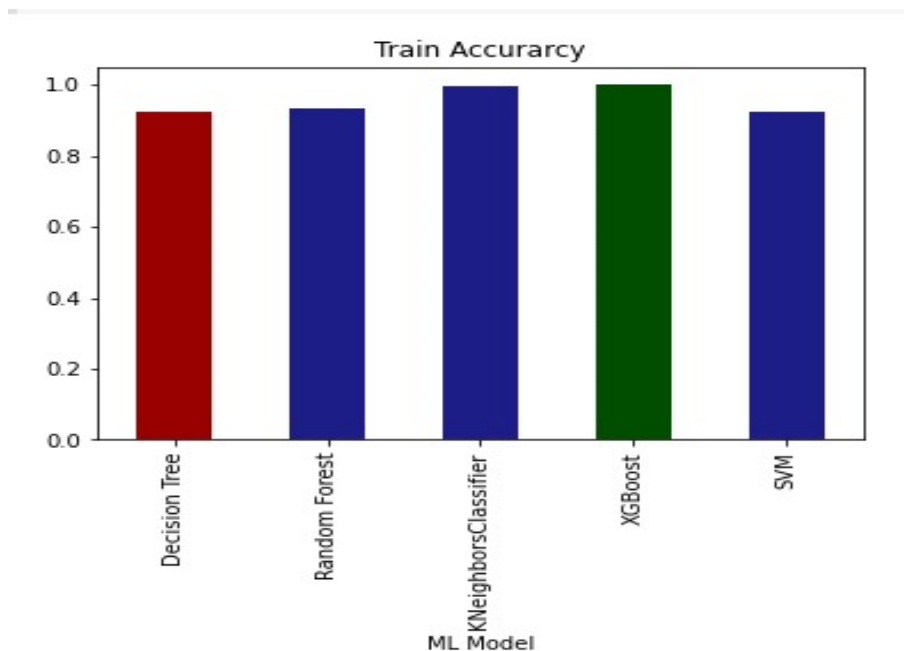


Fig 5.2 Train Accuracy

ML Model	Train Accuracy	Test Accuracy
Decision Tree	0.993	0.919
Random Forest	0.935	0.919
KNN	0.999	0.882
XG Boost	0.997	0.940
SVM	0.927	0.922

Table 5.3 Accuracy Score

CHAPTER 6

CONCLUSIONS AND FUTURE SCOPES

CONCLUSION

Phishing is a huge threat to the security and safety of the web and phishing detection is an important problem domain. Phishing attacks are very crucial, and it is important to have an automated mechanism to understand and avoid it. As very important and personal information of the user can be leaked through phishing websites, it becomes more critical to take care of this issue. This problem can be easily solved by using the machine learning algorithms with the classifier. We already have classifiers which give a good prediction rate of the phishing, but the most important thing is, the number of features. We have increased the number of features in the feature extraction part. When a right algorithm matches with the perfect model, the prediction rate will be good. On reviewing some of the traditional approaches to phishing detection, such as blacklist and heuristic evaluation methods and their drawbacks, we have tested five machine learning algorithms on the dataset collected from various sources such as phishtank, openphish, etc, and reviewed their results. Based on the best algorithm and its performance a Chrome extension has been built for detecting phishing web pages directly while trying to access it. The extension allows easy deployment of our phishing detection model to end users. 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. It can be concluded that the proposed model has proved it has high accuracy and enough features to take into action. Hopefully, phishing activities decrease and wipe out the term phishing attack from the Internet era.

FUTURE SCOPES

Although the use of URL lexical features alone has been shown to result in high accuracy (93.8%), 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 real-time live detection, so that no need of entering the webpages, the detector will warn before opening the links. New phishing attack patterns can easily be learned and improve the accuracy of our models with better feature extraction.

CHAPTER 7

APPENDICES

7.1 SAMPLE CODE

```
import re
import time
import whois
import socket
import pickle
import requests
import ipaddress
import urllib.request
from bs4 import BeautifulSoup
from googlesearch import search
from datetime import date, datetime
from dateutil.parser import parse as date_parse

## Feature Extraction

# Calculates number of months
def diff_month(d1, d2):
    return (d1.year - d2.year) * 12 + d1.month - d2.month

# Generate data set by extracting the features from the URL
def generate_data_set(url):

    data_set = []

    # Converts the given URL into standard format
    if not re.match(r"^https?", url):
        url = "http://" + url

    # Stores the response of the given URL
    try:
        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'html.parser')
    except:
        response = ""
        soup = -999
```

```

# Extracts domain from the given URL
domain = re.findall(r"://([^\s]+)/?", url)[0]
if re.match(r"^www.", domain):
    domain = domain.replace("www.", "")

# Requests all the information about the domain
whois_response = whois.whois(domain)

rank_checker_response =
requests.post("https://www.checkpagerank.net/index.php",
{
    "name": domain
})

# Extracts global rank of the website
try:
    global_rank = int(re.findall(r"Global Rank: ([0-9]+)",
rank_checker_response.text)[0])
except:
    global_rank = -1

# 1.having_IP_Address
try:
    ipaddress.ip_address(url)
    data_set.append(-1)
except:
    data_set.append(1)

# 2.URL_Length
if len(url) < 54:
    data_set.append(1)
elif len(url) >= 54 and len(url) <= 75:
    data_set.append(0)
else:
    data_set.append(-1)

# 3.Shortening_Service
match=re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.
im|is\.gd|cli\.gs|"yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl

```

```
\.nl|snipurl\.com|"short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|sn  
ipr\.com|fic\.kr|loopt\.us|"doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.l  
y|to\.ly|bit\.do|t\.co|lnkd\.in|"db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\  
com|ow\.ly|bit\.ly|ity\.im|"q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl  
\.com|cutt\.us|u\.bb|yourls\.org|"x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|  
vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|tr\.im|link\.zip\.net',url)
```

```
if match:
```

```
    data_set.append(-1)
```

```
else:
```

```
    data_set.append(1)
```

```
# 4.having_At_Symbol
```

```
if re.findall("@", url):
```

```
    data_set.append(-1)
```

```
else:
```

```
    data_set.append(1)
```

```
# 5.double_slash_redirecting
```

```
list=[x.start(0) for x in re.finditer('/', url)]
```

```
if list[len(list)-1]>6:
```

```
    data_set.append(-1)
```

```
else:
```

```
    data_set.append(1)
```

```
# 6.Prefix_Suffix
```

```
if re.findall(r"https?:/[^\-]+-[^\-]+/", url):
```

```
    data_set.append(-1)
```

```
else:
```

```
    data_set.append(1)
```

```
# 7.having_Sub_Domain
```

```
if len(re.findall(".", url)) == 1:
```

```
    data_set.append(1)
```

```
elif len(re.findall(".", url)) == 2:
```

```
    data_set.append(0)
```

```
else:
```

```
    data_set.append(-1)
```

```
# 8.SSLfinal_State
```

```
try:
```

```

        if response.text:
            data_set.append(1)
    except:
        data_set.append(-1)

# 9.Domain_registration_length
expiration_date = whois_response.expiration_date
registration_length = 0
try:
    expiration_date = min(expiration_date)
    today = time.strftime('%Y-%m-%d')
    today = datetime.strptime(today, '%Y-%m-%d')
    registration_length = abs((expiration_date - today).days)

    if registration_length / 365 <= 1:
        data_set.append(-1)
    else:
        data_set.append(1)
except:
    data_set.append(-1)

# 10.Favicon
if soup == -999:
    data_set.append(-1)
else:
    try:
        for head in soup.find_all('head'):
            for head.link in soup.find_all('link', href=True):
                dots = [x.start(0) for x in re.finditer('\.', head.link['href'])]
                if url in head.link['href'] or len(dots) == 1 or domain in
head.link['href']:
                    data_set.append(1)
                    raise StopIteration
            else:
                data_set.append(-1)
                raise StopIteration
    except StopIteration:
        pass

#11. port

```

```

try:
    port = domain.split(":")[1]
    if port:
        data_set.append(-1)
    else:
        data_set.append(1)
except:
    data_set.append(1)

#12. HTTPS_token
if re.findall(r"^https://", url):
    data_set.append(1)
else:
    data_set.append(-1)

#13. Request_URL
i = 0
success = 0
if soup == -999:
    data_set.append(-1)
else:
    for img in soup.find_all('img', src= True):
        dots= [x.start(0) for x in re.finditer('\.', img['src'])]
        if url in img['src'] or domain in img['src'] or len(dots)==1:
            success = success + 1
        i=i+1

    for audio in soup.find_all('audio', src= True):
        dots = [x.start(0) for x in re.finditer('\.', audio['src'])]
        if url in audio['src'] or domain in audio['src'] or len(dots)==1:
            success = success + 1
        i=i+1

    for embed in soup.find_all('embed', src= True):
        dots=[x.start(0) for x in re.finditer('\.',embed['src'])]
        if url in embed['src'] or domain in embed['src'] or len(dots)==1:
            success = success + 1
        i=i+1

    for iframe in soup.find_all('iframe', src= True):

```

```

dots=[x.start(0) for x in re.finditer('\.',iframe['src'])]
if url in iframe['src'] or domain in iframe['src'] or len(dots)==1:
    success = success + 1
i=i+1

try:
    percentage = success/float(i) * 100
    if percentage < 22.0 :
        data_set.append(1)
    elif((percentage >= 22.0) and (percentage < 61.0)) :
        data_set.append(0)
    else :
        data_set.append(-1)
except:
    data_set.append(1)

#14. URL_of_Anchor
percentage = 0
i = 0
unsafe=0
if soup == -999:
    data_set.append(-1)
else:
    for a in soup.find_all('a', href=True):
        # 2nd condition was 'JavaScript ::void(0)' but we put JavaScript because
        the space between javascript and :: might not be
        # there in the actual a['href']
        if "#" in a['href'] or "javascript" in a['href'].lower() or "mailto" in
a['href'].lower() or not (url in a['href'] or domain in a['href']):
            unsafe = unsafe + 1
            i = i + 1

try:
    percentage = unsafe / float(i) * 100
except:
    data_set.append(1)

if percentage < 31.0:
    data_set.append(1)
elif ((percentage >= 31.0) and (percentage < 67.0)):

```

```

        data_set.append(0)
    else:
        data_set.append(-1)

```

#15. Links_in_tags

```

i=0
success =0
if soup == -999:
    data_set.append(-1)
else:
    for link in soup.find_all('link', href= True):
        dots=[x.start(0) for x in re.finditer('\.',link['href'])]
        if url in link['href'] or domain in link['href'] or len(dots)==1:
            success = success + 1
        i=i+1

    for script in soup.find_all('script', src= True):
        dots=[x.start(0) for x in re.finditer('\.',script['src'])]
        if url in script['src'] or domain in script['src'] or len(dots)==1 :
            success = success + 1
        i=i+1
    try:
        percentage = success / float(i) * 100
    except:
        data_set.append(1)

    if percentage < 17.0 :
        data_set.append(1)
    elif((percentage >= 17.0) and (percentage < 81.0)) :
        data_set.append(0)
    else :
        data_set.append(-1)

```

#16. SFH

```

for form in soup.find_all('form', action= True):
    if form['action'] == "" or form['action'] == "about:blank" :
        data_set.append(-1)
        break
    elif url not in form['action'] and domain not in form['action']:
        data_set.append(0)

```



```

        break
    else:
        data_set.append(1)
        break

#17. Submitting_to_email
if response == "":
    data_set.append(-1)
else:
    if re.findall(r"[mail\(\)|mailto:?}", response.text):
        data_set.append(1)
    else:
        data_set.append(-1)

#18. Abnormal_URL
if response == "":
    data_set.append(-1)
else:
    if response.text == "":
        data_set.append(1)
    else:
        data_set.append(-1)

#19. Redirect
if response == "":
    data_set.append(-1)
else:
    if len(response.history) <= 1:
        data_set.append(-1)
    elif len(response.history) <= 4:
        data_set.append(0)
    else:
        data_set.append(1)

#20. on_mouseover
if response == "":
    data_set.append(-1)
else:
    if re.findall("<script>.+onmouseover.+</script>", response.text):
        data_set.append(1)

```

```

    else:
        data_set.append(-1)

#21. RightClick
if response == "":
    data_set.append(-1)
else:
    if re.findall(r"event.button ?== ?2", response.text):
        data_set.append(1)
    else:
        data_set.append(-1)

#22. popUpWidnow
if response == "":
    data_set.append(-1)
else:
    if re.findall(r"alert\(", response.text):
        data_set.append(1)
    else:
        data_set.append(-1)

#23. Iframe
if response == "":
    data_set.append(-1)
else:
    if re.findall(r"<iframe>|<frameBorder>", response.text):
        data_set.append(1)
    else:
        data_set.append(-1)

#24. age_of_domain
if response == "":
    data_set.append(-1)
else:
    try:
        registration_date = re.findall(r'Registration Date:</div><div class="df-  
value">([^\<]+)</div>', whois_response.text)[0]
        if diff_month(date.today(), date_parse(registration_date)) >= 6:
            data_set.append(-1)
    else:

```

```

        data_set.append(1)
    except:
        data_set.append(1)

#25. DNSRecord
dns = 1
try:
    d = whois.whois(domain)
except:
    dns=-1
if dns == -1:
    data_set.append(-1)
else:
    if registration_length / 365 <= 1:
        data_set.append(-1)
    else:
        data_set.append(1)

#26. web_traffic
try:
    rank =
BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s
&url=" + url).read(), "xml").find("REACH")['RANK']
    rank= int(rank)
    if (rank<100000):
        data_set.append(1)
    else:
        data_set.append(0)
except TypeError:
    data_set.append(-1)

#27. Page_Rank
try:
    if global_rank > 0 and global_rank < 100000:
        data_set.append(-1)
    else:
        data_set.append(1)
except:
    data_set.append(1)

```

```

#28. Google_Index
site=search(url, 5)
if site:
    data_set.append(1)
else:
    data_set.append(-1)

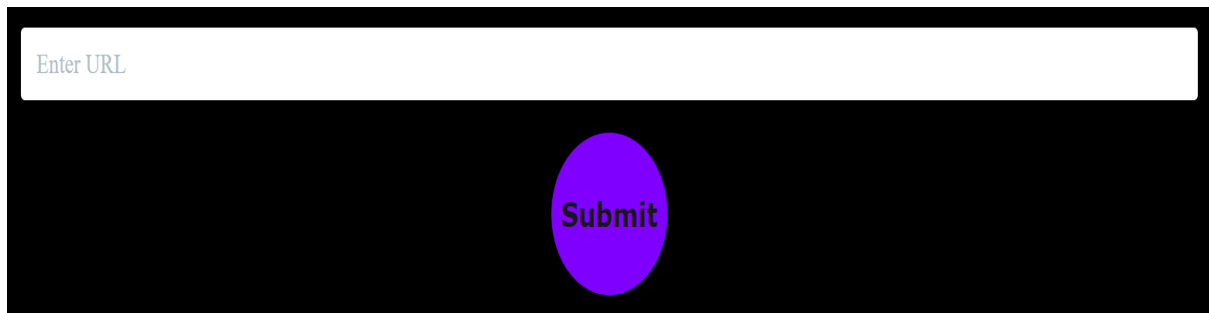
#29. Links_pointing_to_page
if response == "":
    data_set.append(-1)
else:
    number_of_links = len(re.findall(r"<a href=", response.text))
    if number_of_links == 0:
        data_set.append(1)
    elif number_of_links <= 2:
        data_set.append(0)
    else:
        data_set.append(-1)

#30. Statistical_report
url_match=re.search('at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|h
ol\.es|sweddy\.com|myjino\.ru|96\.lt|ow\.ly',url)
try:
    ip_address=socket.gethostbyname(domain)
    ip_match=re.search(IP',ip_address)
    if url_match:
        data_set.append(-1)
    elif ip_match:
        data_set.append(-1)
    else:
        data_set.append(1)
except:
    data_set.append(0)
    # return data_set
    print ('Connection problem. Please check your internet connection!')
print (data_set)
return data_set

# Importing model
loaded_model = pickle.load(open("./xgb_30.pkl", "rb"))

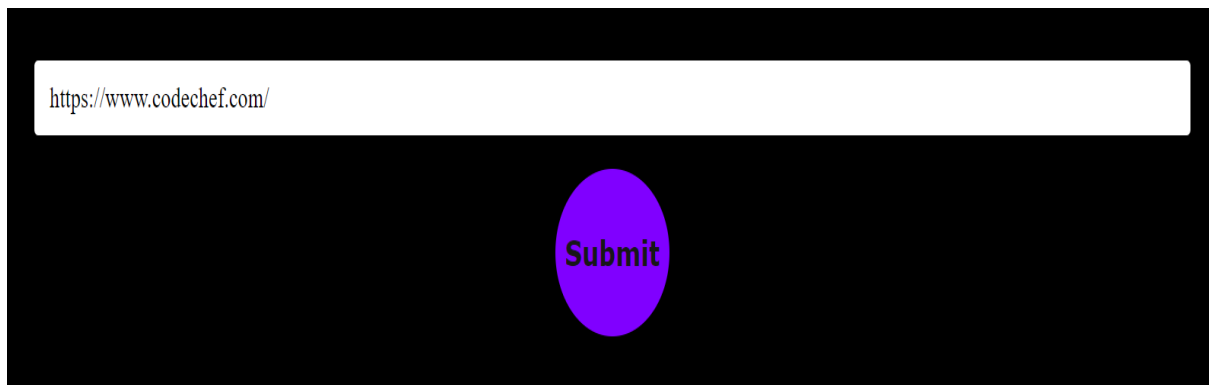
```

7.2 SNAPSHOTS



A screenshot of a web form with a black background. At the top, there is a white rectangular input field containing the placeholder text "Enter URL". Below this field, centered on the black background, is a large, solid blue oval button with the word "Submit" written in white text.

Fig 7.2.1 URL Entry



A screenshot of the same web form as in Fig 7.2.1, but with the input field now containing the URL "https://www.codechef.com/". The "Submit" button remains centered below the field.

Fig 7.2.2 Submitting Legitimate URL

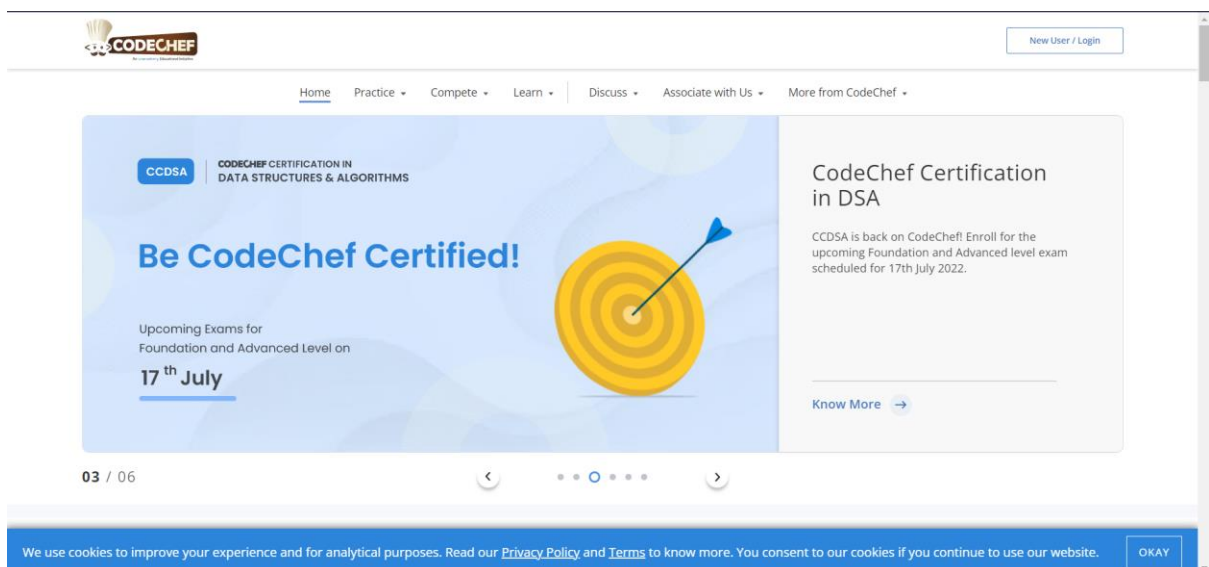


Fig 7.2.3 Redirection - Not Phishing

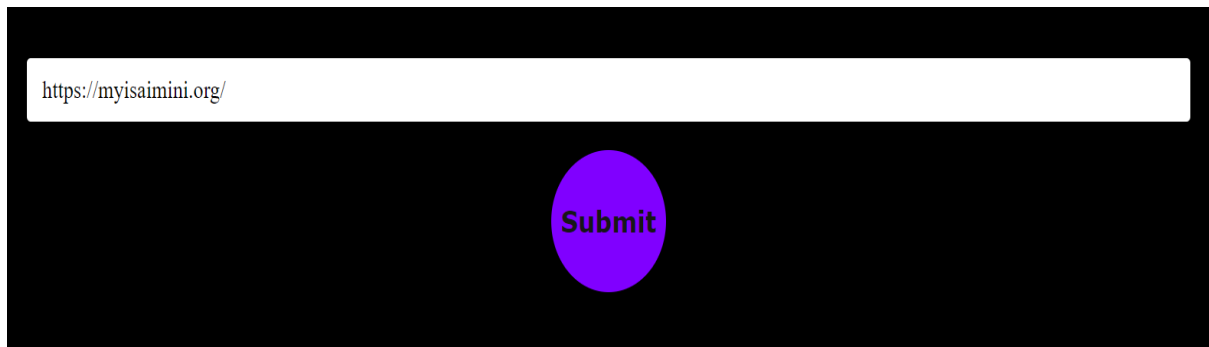


Fig 7.2.4 Submitting fake/Phishing URL

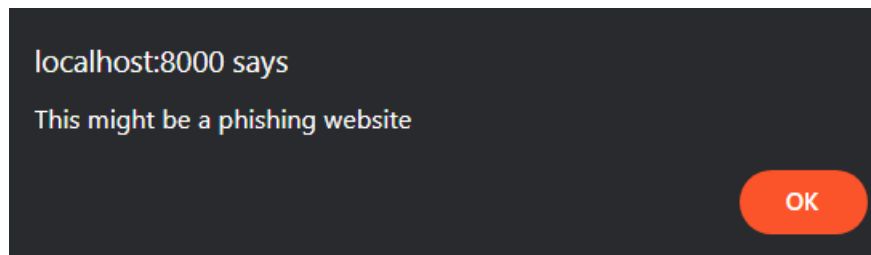


Fig 7.2.5 Warning



Fig 7.2.6 Prediction - Phishing

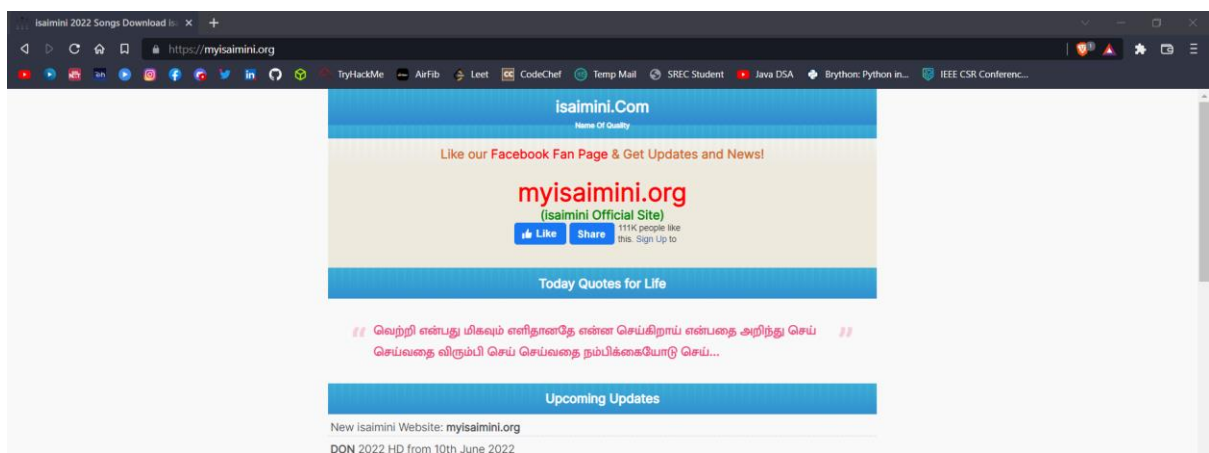


Fig 7.2.7 Continue with RISK

REFERENCES

- [1] Abdallah Moubayed, Mohammad Injadat, Ali Bou Nassif (2018).
“Challenges and Research Opportunities Using Machine Learning & Data Analytics”, July 2018 IEEE Access PP (99):1-1.
- [2] Alwyn Roshan Pais, Routhu Srinivasa Rao “An Enhanced Blacklist Method to Detect Phishing Websites”. Information Systems Security. ICISS 2017. Lecture Notes in Computer Science, vol 10717.
- [3] Ding Y, Luktarhan N, Slamun W (2019). A keyword-based combination approach for detecting phishing webpages” Computers & security, 84, 256-275.
- [4] Doyen Sahoo, Chenghao Liu, Steven (2019) – “Malicious URL Detection using Machine Learning: A Survey”. last revised 21 Aug 2019 (version, v3).
- [5] Gunter Ollmann “Securing against the ‘threat’ of instant” Published by Network Security Volume 2004, Issue 3, March 2004, Pages 8-11.
- [6] Joseph Johnson “Worldwide digital population as of April 2022” Published by, May 9, 2022.
- [7] Miramirkhani N, Antonakakis M (2017). “Hiding in Plain Sight: A Longitudinal Study of Combosquatting Abuse. Association of Computer Machinery's”, Computer and Communications Security (ACM CCS) 2017.
- [8] Rao, R. S., & Pais, A. R. (2019). “Jail-Phish: An improved search engine-based phishing detection system”, Computers & Security, 83, 246-267.
- [9] Sagar Patil, Yogesh Shetye (2020). “Detecting Phishing Websites Using Machine Learning”. IRJET e-ISSN: 2395-0056 Volume: 07 Issue: Feb 2020.
- [10] Sophiya Shikalgar , Dr. S. D. Sawarkar, Mrs. Swati Narwane (2019) – “Detection of URL based Phishing Attacks using Machine Learning“. IJERT ISSN: 2278-0181 Vol. 8 Issue 11, November-2019.
- [11] Suzanne Widup, Alex Pinto, Gabriel Bassett, David Hylender (2021) Verizon Data Breach Investigations Report, May 2021.
- [12] Widup Suzanne, Widup Marc, Spitler David Hylender Gabriel (2018). “Verizon Data Breach Investigations Report 2018”. April 2018.