

WEBSHIELD: SHEILDING YOUR BROWSING EXPERIENCE USING DECISION TREE

Submitted in partial fulfillment of the requirements
for the award .

Bachelor of Engineering degree in Computer Science and Engineering

by

MALLIDI VENKATA SUKUMAR REDDY (Register.No - 40110710)

MANDAPALLI RITHWIK REDDY (Register.No – 40110717)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

CATEGORY -1 UNIVERSITY BY UGC

Accredited with Grade “A++” by NAAC | 12B Status by UGC | Approved by AICTE

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

APRIL - 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of **Mallidi Venkata Sukumar Reddy(Reg.No.40110710)** and **Mandapalli Rithwik Reddy(Reg.No.40110717)** who carried out the Project entitled **"WEBSHIELD : SHIELDING YOUR BROWSING EXPERIENCE USING DECISION TREE"** under my supervision from November 2023 to April 2024.

Internal Guide

Dr. L. SUJITHLEN M.E., Ph.D.,

HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Head of the Department

Dr. L. LAKSHMANAN M.E., Ph.D.
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Jepplaar Nagar, Rajiv Gandhi Salai,
Chennai - 600 119.

Submitted for Viva voce Examination held on 13/04/2024

Internal Examiner

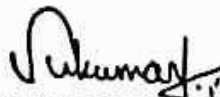
External Examiner

DECLARATION

I, Mallidi Venkata Sukumar Reddy (Reg.No.40110710), hereby declare that the Project Report entitled "WEBSHIELD: SHIELDING YOUR BROWSING EXPERIENCE USING DECISION TREE" done by me under the guidance of Dr. L SUJI HELEN, M.E., Ph.D., is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: 13/04/24

PLACE: CHENNAI



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **The Board of Management of Sathyabama Institute of Science and Technology** for their encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA M.E., Ph. D, Dean**, School of Computing, **Dr. L. LAKSHMANAN, M.E., Ph.D.**, Head of the Department, Department of Computer Science and Engineering for providing me necessary support during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. L SUJI HELEN, M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Phishing sites that expect to take the victims confidential data by diverting them to surf a fake website page that resembles an honest to goodness one is another type of criminal acts through the internet and its one of the especially concerns toward numerous areas including e-managing an account and retailing. Phishing site detection is truly an unpredictable and element issue including numerous components and criteria that are not stable. On account of the last and in addition ambiguities in arranging sites because of the intelligent procedures programmers are utilizing, some keen proactive strategies can be helpful and powerful tools can be utilized, for example, fuzzy, neural system and data mining methods can be a successful mechanism in distinguishing phishing sites. In this model the algorithms that are applies was Random Forest (RF), one of the different types of machine learning based algorithms used for detection of Phishing websites. Finally, we measured and compared the performance of the classifier in terms of accuracy.

TABLE OF CONTENT

Chapter No.	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	viii
1	INTRODUCTION	1
	1.1 Problem Definition	2
	1.2 Scope and Objectives	2
2	LITERATURE SURVEY	3
	2.1 Inferences from Literature Survey	6
	2.2 Motivation	6
	2.3 Open problems in Existing System	7
3	SYSTEM ANALYSIS	10
	3.1 Drawbacks of Existing System	10
	3.2 Software and Hardware Requirements	10
4	DESCRIPTION OF PROPOSED SYSTEM	11
	4.1 Architecture of Proposed System	11
	4.2 Selected Methodology and Process Model	12
	4.3 Data sets	13
	4.3.1 Using the IP address	13
	4.3.2 Using URL Shortening Services	14
	4.3.3 URL's having "@" symbol	15
	4.3.4 Redirecting using "/"	15
	4.3.5 Adding Prefix or Suffix	15
	4.3.6 Sub domain and Multi sub domain	16
	4.3.7 HTTPS	16
	4.3.8 Domain registration length	17
	4.3.9 Favicon	17
	4.3.10 Using non-standard port	17
	4.3.11 The existence of HTTPS token in the domain	19
	4.3.12 Abnormal based features	19
	4.3.13 HTML and JAVASCRIPT based features	22
	4.1.14 Domain based features	23

5	SYSTEM IMPLEMENTATION	31
	5.1 Development and Deployment Setup	31
	5.2 Algorithms	32
	5.3 Testing	33
	5.4 Results and Discussion	41
6	CONCLUSION	48
	6.1 Conclusion	48
	6.2 Future Enhancement	48
/	REFERENCES	50
	APPENDIX	54
	SOURCE CODE	54
	SCREENSHOTS	59

FIGURE NO.	FIGURE NAME	PAGE NO.
2.1	Basics Statistics	14
2.2	Attacks by Industry	16
4.1	System Architecture	21
4.2	Local Host Web Server	38
4.3	Local Host Output Console	39
5.1	Localhost Type Error	45

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Phishing is a type of extensive fraud that happens when a malicious website act like a real one keeping in mind that the end goal to obtain touchy data, for example, passwords, account points of interest, or MasterCard numbers. In spite of the fact that there are a few contrary to phishing programming and methods for distinguishing potential phishing endeavors in messages and identifying phishing substance on sites, phishes think of new and half-breed strategies to go around the accessible programming and systems.

Phishing is a trickery system that uses a blend of social designing what's more, innovation to assemble delicate and individual data, for example, passwords and charge card subtle elements by taking on the appearance of a dependable individual or business in an electronic correspondence. Phishing makes utilization of spoof messages that are made to look valid and implied to be originating from honest to goodness sources like money related foundations, ecommerce destinations and so forth, to draw clients to visit fake sites through joins gave in the phishing email. The misleading sites are intended to emulate the look of a genuine organization site page.

The employing so as to phishing invader's trap clients diverse social building strategies, for example, debilitating to suspend client accounts on the off chance that they don't finish the account upgrade process, give other data to approve their records or a few different motivations to get the clients to visit their satirize page. Supervised learning (Classification Technique) accommodates a vastly improved precision while unsupervised learning accommodates a quick and dependable way to deal with infer information from a dataset. That's why we used supervised learning in our work.

1.2 Problem Definition

Detection of Malicious URLs websites hosting phishing, spam etc. by using Machine Learning.

1.3 Scope and Objectives

- The system should be useful in many e-commercial websites for maintaining the security and reliability of customers and people online.
- The system should be useful in preventing online frauds leading to leakage of important and private user data.
- The scope of using Machine Language over other Traditional Detecting Methods.

Objectives:

- Understanding phishing domain (or Fraudulent Domain) characteristics, its distinguishing features from legitimate domains
- Why it is so important to detect this domain and how they can be detected using machine learning and natural language processing techniques
- Reviewing the state-of-the-art machine learning techniques for malicious URL detection in literature
- Understanding the newly emerging concept of Malicious URL Detection as a service and the principles to be used while designing such a system.

To distinguish the phishing websites from the legitimate websites and ensure secure transactions to users.

CHAPTER 2

LITERATURE SURVEY

M. Aljabri and S. Mirza, "Phishing Attacks Detection using Machine Learning and Deep Learning Models," 2022 7th International Conference on Data Science and Machine Learning Applications (CDMA), Riyadh, Saudi Arabia, 2022, pp. 175-180, doi: 10.1109/CDMA54072.2022.00034..

Aljabri and Mirza has done this in Data science and Machine Learning Applications. One of the most significant concerns is the production and spread of hostile URLs. Newsletters, hyperlinks, pop-up ads, & ads are a few of the common ways that harmful URLs spread. Due to the fact that they host malicious material without permission, their proliferation may lead to monetary loss, personal information being stolen, and even ransomware being downloaded & installed on a computer.

Awasthi and N. Goel, "Phishing Website Prediction: A Comparison of Machine Learning Techniques," in Data Intelligence and Cognitive Informatics, Springer. Singapore. 2021, pp. 637-650.

They has completely studied about the Url Among the two key methods has been utilized to identify fraudulent URLs. The very first method is a blacklisted that is regularly evolving when potentially malicious URLs are discovered. They used only two algorithms in their complete project such as Random forest classification and support vector machine

A. Hashim, R. Medani and T. A. Attia, "Defences Against web Application Attacks and Detecting Phishing Links Using Machine Learning," 2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE), Khartoum, Sudan, 2021, pp. 1-6, doi:10.1109/ICCCEEE49695.2021.9429609..

In this we studied that The very first method is a blacklisted that is regularly evolving when potentially malicious URLs are discovered. Software like Google Secure Browser, Intel Sites Defender, & Content filtering Threat Seeker are just a few

examples of business products that use this method to detecting harmful URLs.

M. A. Syafiq Rohmat Rose, N. Basir, N. F. Nabila Rafie Heng, N. Juana Mohd Zaizi and M. M. Saudi, "Phishing Detection and Prevention using Chrome Extension," 2022 10th International Symposium on Digital Forensics and Security (ISDFS), Istanbul, Turkey, 2022, pp. 1-6, doi: 10.1109/ISDFS55398.2022.9800826..

Researchers have been progressively using this strategy in the past century as a means of fixing the problems encountered by the earlier method. A test set of dangerous and safe URLs is essential for ML.

M. Chatterjee and A. -S. Namin, "Detecting Phishing Websites through Deep Reinforcement Learning," 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), Milwaukee, WI, USA, 2019, pp. 227-232, doi: 10.1109/COMPSAC.2019.10211.

In this research the strategy involves the use of AI-based methods, such as ML different classifiers for threat detection through URLs. Researchers have been progressively using this strategy in the past century as a means of fixing the problems encountered by the earlier method. A test set of dangerous and safe URLs is essential for ML

M. Korkmaz, O. K. Sahingoz, and B. Diri, "Detection of Phishing Websites by Using Machine Learning-Based URL Analysis," in 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2020, pp. 1-7, doi: 10.1109/ICCCNT49239.2020.9225561.

Korkmaz and Sahingoz has said that they had looked at 59 distinct aspects connected to the name, subdomain, & route parts of the URL, with a special emphasis on lexical-based variables. They tested a number of Machine learning models on three sets of data, and found that RF had the best overall performance (with a reliability of up to

96.6% over the datasets) while ANN had the quickest runtime. Site spoofing attempt identification was performed by Hossain.

M. Tubyte and A. Paulauskaite-Taraseviciene, “Research on phishing email detection based on URL parameters using machine learning algorithms,” CEUR Workshop Proc., vol. 2915, pp. 18- 26, 2021.

In order to assess the efficacy of ML models for identifying malicious URLs, Khan. tested them on three different datasets. They found that RF and ANN were superior to other techniques of categorization after their accuracy is determined as 70.2%. After demonstrating that the model's performance was unaffected by the removal of irrelevant characteristics from the fields of data, they found the set of overlapping features present in all three databases

N. Bari and M. Ali Shah, "SECURING DIGITAL ECONOMIES: DETECTION OF PHISHING ATTACKS USING MACHINE LEARNING APPROACHES," Competitive Advantage in the Digital Economy (CADE 2021), Online Conference, 2021, pp. 105-111, doi: 10.1049/icp.2021.2412.

They had studied and Based on findings, SVM and Decision Tree were the most effective methods, with an accuracy of 81.97%. They also noted that the research was limited because of both the amount of data and the quantity of attributes.

R. Mohammad, T. L. McCluskey, and F. Thabtah, “UCI Machine Learning Repository: Phishing Websites Data Set.” <https://archive.ics.uci.edu/ml/datasets/phishing+websites> (accessed Nov. 20, 2021).

Throughout this study, we suggested an ML-based method for identifying malicious URL. When conducting our study, we relied on two different datasets: one that was originally written by Mohamad. and another we downloaded through Kaggle.

UCI phishing collection to conduct our research that can be found on McCluskey. There are 32 qualities and 11,056 occurrences in all. No values are missing from it. The project 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.).

	2H2014	1H2014	2H2013	1H2013	2H2012	1H2012
Phishing domain names	95,321	87,901	82,163	53,685	89,748	64,204
Attacks	123,972	123,741	115,565	72,758	123,476	93,462
TLDs used	272	227	210	194	207	202
IP-based phish (unique Ips)	3,095	2,317	837	1,626	1,981	1,864
Maliciously registered domains	27,253	22,679	22,831	12,173	5,833	7,712
IDN domains	103	112	82	78	147	58
Number of targets	569	756	681	720	611	486

Fig 2.1 Basic Statistics

“Phishers continued to attack Apple, PayPal, and Taobao.com heavily. Each of these three e-commerce giants suffered over 20,000 phishing attacks against their respective services and brands. Together, these top three were the targets of nearly 54 percent of the world’s phishing attacks. The next seven brands were targeted for a

combined 23 percent of all phishing attacks — meaning the top 10 targets accounted for over three quarters of all phishing attacks observed worldwide. The number of times that the targets were attacked follows a long tail. Half of the targets were attacked four or fewer times during the six-month period (up from three times in 1H2014). One hundred and fifty-eight targets were attacked only once each in the period.” [‘Global Phishing Survey: Trends and Domain Name Use in 2H2014’]

Other interesting trends highlighted in the Global Phishing Survey 2H2014 report are:

- New companies are constantly being targeted by phishers. Some phishers are attacking targets where consumers may least expect it.
- The ten companies that are targeted most often by phishers are attacked constantly, sometimes more than 1,000 times per month. Together the top ten targets suffered more than three-quarters of all the phishing attacks observed worldwide.
- The number of domain names used for phishing reached an all-time high.
- Phishing in the new top-level domains started slowly. We expect to see phishing levels in them rise as time goes on.
- Chinese phishers were responsible for 85% of the domain names that were registered for phishing. These phishers started using .CN domains more frequently.
- Phishing attacks were not mitigated as quickly. The median uptime of phishing attacks increased to 10 hours 6 minutes — up from 8 hours and 42 minutes in 1H2014. This means that phishing attacks were not being shut down as efficiently in the critical first hours, when most victims fall prey.
- If attacks are divided by Industry, we can clearly see that the makers involving money are the ones more targeted like it can be seen them in the next chart:

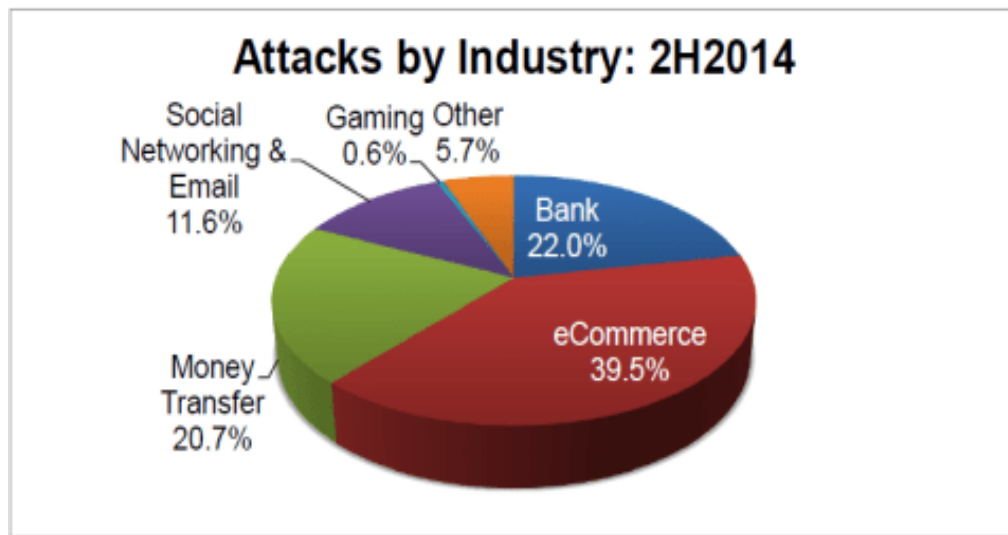


Fig 2.2 Attacks by Industry

2.1 INFERENCES FROM LITREATURE SURVEY

That proves that “These show criminals seeking the credentials of consumers in places where consumers may least expect it. Phishers target wide-ranging targets for several reasons. One is to perform credit card theft, and hitting new targets may lull consumers into a false sense of security. The phishers can also monetize stolen data through reshipping fraud, a tactic that remains popular. Phishers also steal usernames and passwords from one site in order to try those credentials on other sites. Many consumers re-use usernames and passwords, and this poor habit can be costly. If a site is getting phished for the first time, it may have been targeted by a more sophisticated phisher, who had the skill to design a new phishing template.” [‘Global Phishing Survey: Trends and Domain Name Use in 2H2014’]

2.2 Motivation

Malicious URL, a.k.a. malicious website, is a common and serious threat to cyber security. 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. The project 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.).

2.3 OPEN PROBLEMS IN EXISTING SYSTEM

There are countless domains where phishing attack can occur like online payment sector, webmail, and financial institution, file hosting or cloud storage and many others. The webmail and online payment sector were embattled by phishing more than in any other industry sector. Phishing can be done through email phishing scams and spear phishing hence user should be aware of the consequences and should not give their 100 percent trust on common security application. Machine Learning is one of the efficient techniques to detect phishing as it removes drawback of existing approach. Our technique is not applicable to all types of obfuscated URLs as described in Section II. URLs composed of only a malicious domain, URLs based on shortening services or URLs algorithmically generated are kind of malicious URLs that can bypass our detection technique. This kind of URLs and malicious domains are however widely used in botnet communication (C&C) or spamming activities [29], such activities do not rely on a social engineering process as phishing does. The main part of URLs used for phishing are meaningful and composed of many terms, this is why our technique is relevant in a phishing context.

A limitation of the implementation is that data publicly available through Google Trends and Yahoo Clues is limited. For each requested term only the ten related most popular terms are returned by these tools. Related terms less requested by search engine users do not appear in results while being relevant for intra-URL relatedness computing. For the same reason, some unpopular terms blended in URLs do not match any results. The reason is that Google and Yahoo do not provide data that is not representative enough, i.e., for terms that are not requested enough by their users. These facts limit the accuracy of intra-URL relatedness computing and

is one of the reasons why extra features such as ranking are included in the feature set. However, the data provided by these statistical tools from the search engines rely on searches that have been made by users. To the best of our knowledge this data is not biased by commercial interest but a full access to Web search logs would highly improve the relevancy of intra relatedness metrics and, as a result, the classification performance as well. Despite this limited access to data, the results presented in this paper provide strong hints regarding the relevancy of using search engine query data for phishing detection. A last issue of using on-line tools for inferring intra-URL relatedness is the delay implied by multiple HTTP requests for every term that compose a URL. Even though using streaming analytics techniques such as STORM can lead to a five-fold reduction of the amount of time (0.77 seconds) as presented in Section VI. A solution to speed up the process would be direct access to search engine query data.

S.No	Title	Author	Year	Accuracy	Pros	Cons
1	Phishing Attacks Detection using Machine Learning and Deep Learning Models	Malak Aljabri	2022	80	Develop a real-time system for gathering URL features	Consumes time and space at high rate
2	Phishing Detection and Prevention using Chrome Extension	Mohamad nazim feroz	2022	67	Using only client side features to detect phishing websites	Model is inefficient and inaccurate

S.No	Title	Author	Year	Accuracy	Pros	Cons
3	The Emergence Threat of Phishing Attack and The Detection	Fahmida Islam	2021	75	introduce the concept of intra-URL relatedness	relying on search engine query data.
4	Securing Digital Economies: Detection Of Phishing Attacks	N Bari, M. Ali shah	2021	77	develop a real-time system for gathering URL features	Consumes time and space at high rate

S.No	Title	Aurthor	Year	Accuracy	Pros	Cons
5	Defences Against web Application Attacks and Detecting Phishing Links Using Machine Learning	Aya Hashim	2021	73	derives an internal threat scale	Relying on search query data
6	Detecting Phishing Websites through Deep Reinforcement Learning	Moitrayee Chatterjee	2021	90	deep learning implementing of RL algorithm	Not optimized for real world implementing.

S.No	Title	Author	Year	Accuracy	Pros	Cons
7	Phishing web site prediction: a comparison of ML techniques	N. Goel, Awasthi	2021	68	completely studied about the Url Among the two key methods has been utilized to identify fraudulent URLs	The very first method is a blacklisted that is regularly evolving when potentially malicious URLs are discovered
8	Research on phishing email detection based on url parameters	M. Tubyte and A. Paulauskaite	2021	70.2	develop a real-time system for gathering URL features	relying on search engine query data.

CHAPTER 3

SYSTEM ANALYSIS

3.1 DRAWBACKS OF EXISTING SYSTEM

In this section, we evaluate the performance of our system based on the feature set, Decision Tree, and Random Forests algorithm. Both classifiers are trained using all features. 100,000 legitimate domains sampled randomly from majestic top one million and 97,430 phishing URLs obtained from Phish-Tank are used for generating the example set used for training and testing. All the sampled legitimate domains are checked against Evaluation 66 Safe Browsing and Virus-Total, within which Safe Browsing reports 158 of them as unsafe, while Virus-Total finds 696 of them risky. 94 URLs are reported as malicious by both of them. All the domains that are marked as unsafe by either of them is removed. As a result, 73,654 legitimate examples and 64,657 phishing examples are generated

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

Hardware Requirements

- System : Intel® Core™ i5-9300H CPU @ 2.40GHz.
- Monitor : LED.
- Mouse : Logitech.
- Ram : 8.00 GB or above 8.00 GB
- Hard Disk : 1 TB

Software Requirements:

- Operating System : Windows 10, Kali Linux
- Language : Python 3
- Framework : Flask 2.2.2

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

Trying to gather personal information through deceptive ways is becoming more common nowadays. In order to assist the user to be aware of the access to such websites, the implemented system notifies the user through email and also pop-up, when trying to access a phishing site. This paper proposes an approach of phishing detection system to detect blacklisted URL also known as phishing websites, so that individual can be alerted while browsing or accessing a particular website. Therefore, it can be utilized for identification and authentication and become a legitimate tool to prevent an individual from getting tricked.

4.1 ARCHITECTURE DIAGRAM

In the below fig:4.1 we are going to see about the architecture diagram for the given model. The end user can easily find out about the website which he is going to be used.

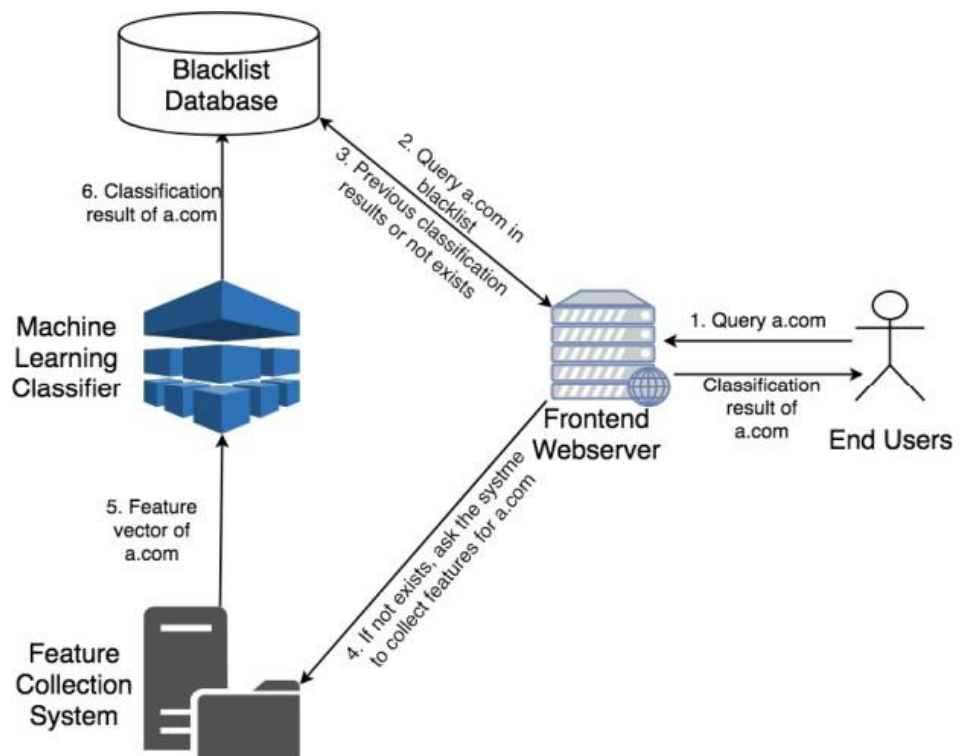


Fig: 4.1 System Architecture

4.2 SELECTED METHODOLOGY OR PROCESS MODEL

Methodology

There are a lot of algorithms and a wide variety of data types for phishing detection in the academic literature and commercial products. A phishing URL and the corresponding page have several features which can be differentiated from a malicious URL. For example; an attacker can register long and confusing domain to hide the actual domain name (Cyber-squatting, Typo-squatted)

Features collected from academic studies for the phishing domain detection with machine learning techniques are grouped as given below.

1. URL-Based Features
2. Domain-Based Features
3. Page-Based Features
4. Content-Based Features

Mainly there is use of Natural Language Processing (NLP) and other machine learning techniques. Moreover, much use more technical features and process them using machine learning algorithms has been imposed.

MODULE DESCRIPTION

Detection Technique

Detection of phishing websites has received a lot of attention recently due to their impact on users' security. Therefore, many techniques have been developed to detect phishing websites varying from communication-oriented techniques, such as authentication protocols, blacklisting, and white-listing, to content-based filtering techniques. The blacklisting and white-listing techniques have not proven though to be sufficiently efficient when used in different domains, and thus they are not commonly used. Meanwhile, the content-based phishing filters have been widely used and have proven to be of high efficiency. In light of this, researches have focused on content-based mechanism and on developing machine learning and data mining techniques based on the header and body of emails.

Phishing Websites Features

One of the challenges faced by our research was the unavailability of reliable training datasets. In fact, this challenge faces any researcher in the field. However, although plenty of articles about predicting phishing websites using data mining techniques have been disseminated these days, no reliable training dataset has been published publically, maybe because there is no agreement in literature on the definitive features that characterize phishing websites, hence it is difficult to shape a dataset that covers all possible features. In this article, we shed light on the important features that have proved to be sound and effective in predicting phishing websites. In addition, we proposed some new features, experimentally assign new rules to some well-known features and update some other features.

4.3 Data Set

One of the challenges faced by our research was the unavailability of reliable training datasets. In fact, this challenge faces any researcher in the field. However, although plenty of articles about predicting phishing websites using data mining techniques have been disseminated these days, no reliable training dataset has been published publically, maybe because there is no agreement in literature on the definitive features that characterize phishing websites, hence it is difficult to shape a dataset that covers all possible features.

In this article, we shed light on the important features that have proved to be sound and effective in predicting phishing websites. In addition, we proposed some new features, experimentally assign new rules to some well-known features and update some other features.

4.3.1 Using the IP Address

If an IP address is used as an alternative of the domain name in the URL, such as "<http://125.98.3.123/fake.html>", users can be sure that someone is trying to steal their personal information. Sometimes, the IP address is even transformed into hexadecimal code as shown in the following link "<http://0x58.0xCC.0xCA.0x62/2/paypal.ca/index.html>".

Rule: IF $\begin{cases} \text{If The Domain Part has an IP Address} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

Phishers can use long URL to hide the doubtful part in the address bar.

For example:

<http://federmacedoadv.com.br/3f/aze/ab51e2e319e51502f416dbe46b773a5e/?cmd= home&dispatch=11004d58f5b74f8dc1e7c2e8dd4105e811004d58f5b74f8dc1e7c2e8dd4105e8@phishing.website.html>

To ensure accuracy of our study, we calculated the length of URLs in the dataset and produced an average URL length. The results showed that if the length of the URL is greater than or equal 54 characters then the URL classified as phishing. By reviewing our dataset we were able to find 1220 URLs lengths equals to 54 or more which constitute 48.8% of the total dataset size.

$$\text{Rule: IF} \begin{cases} \text{URLlength} < 54 \rightarrow \text{feature} = \text{Legitimate} \\ \text{else if URLlength} \geq 54 \text{ and } \leq 75 \rightarrow \text{feature} = \text{Suspicious} \\ \text{otherwise} \rightarrow \text{feature} = \text{Phishing} \end{cases}$$

We have been able to update this feature rule by using a method based on frequency and thus improving upon its accuracy.

4.3.2 Using URL Shortening Services “TinyURL”

URL shortening is a method on the “World Wide Web” in which a URL may be made considerably smaller in length and still lead to the required webpage. This is accomplished by means of an “HTTP Redirect” on a domain name that is short, which links to the webpage that has a long URL. For example, the URL “http://portal.hud.ac.uk/” can be shortened to “bit.ly/19DXSk4”.

$$\text{Rule: IF} \begin{cases} \text{TinyURL} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

4.3.3 URL’s having “@” Symbol

Using “@” symbol in the URL leads the browser to ignore everything preceding the “@” symbol and the real address often follows the “@” symbol.

Rule: IF $\begin{cases} \text{Url Having @ Symbol} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

4.3.4 Redirecting using “//”

The existence of “//” within the URL path means that the user will be redirected to another website. An example of such URL’s is: “http://www.legitimate.com//http://www.phishing.com”. We examine the location where the “//” appears. We find that if the URL starts with “HTTP”, that means the “//” should appear in the sixth position. However, if the URL employs “HTTPS” then the “//” should appear in seventh position.

Rule: IF $\begin{cases} \text{ThePosition of the Last Occurrence of "://" in the URL} > 7 \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

4.3.5 Adding Prefix or Suffix Separated by (-) to the Domain

The dash symbol is rarely used in legitimate URLs. Phishers tend to add prefixes or suffixes separated by (-) to the domain name so that users feel that they are dealing with a legitimate webpage. For example, <http://www.Confirme-paypal.com/>.

Rule: IF $\begin{cases} \text{Domain Name Part Includes (-) Symbol} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

4.3.6 Sub Domain and Multi Sub Domains

Let us assume we have the following link: <http://www.hud.ac.uk/students/>. A domain name might include the country-code top-level domains (ccTLD), which in our example is “uk”. The “ac” part is shorthand for “academic”, the combined “ac.uk” is called a second-level domain (SLD) and “hud” is the actual name of the domain. To produce a rule for extracting this feature, we firstly have to omit the (www.) from the URL which is in fact a sub domain in itself. Then, we have to remove the (ccTLD) if it exists. Finally, we count the remaining dots. If the number of dots is greater than one, then the URL is classified as “Suspicious” since it has one sub domain. However, if the dots are greater than two, it is classified as “Phishing” since it will have multiple sub domains. Otherwise, if the URL has no sub domains, we will assign “Legitimate” to the feature.

$$\text{Rule: IF } \begin{cases} \text{Dots In Domain Part} = 1 \rightarrow \text{Legitimate} \\ \text{Dots In Domain Part} = 2 \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

4.3.7 HTTPS (Hyper Text Transfer Protocol with Secure Sockets Layer)

The existence of HTTPS is very important in giving the impression of website legitimacy, but this is clearly not enough. The authors in (Mohammad, Thabtah and McCluskey 2012) (Mohammad, Thabtah and McCluskey 2013) suggest checking the certificate assigned with HTTPS including the extent of the trust certificate issuer, and the certificate age. Certificate Authorities that are consistently listed among the top trustworthy names include: “GeoTrust, GoDaddy, Network Solutions, Thawte, Comodo, Doster and VeriSign”. Furthermore, by testing out our datasets, we find that the minimum age of a reputable certificate is two years.

Rule:

$$\text{IF } \begin{cases} \text{Use https and Issuer Is Trusted and Age of Certificate} \geq 1 \text{ Years} \rightarrow \text{Legitimate} \\ \text{Using https and Issuer Is Not Trusted} \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

4.3.8 Domain Registration Length

Based on the fact that a phishing website lives for a short period of time, we believe that trustworthy domains are regularly paid for several years in advance. In our dataset, we find that the longest fraudulent domains have been used for one year only.

$$\text{Rule: IF } \begin{cases} \text{Domains Expires on} \leq 1 \text{ years} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

4.3.9 Favicon

A favicon is a graphic image (icon) associated with a specific webpage. Many existing user agents such as graphical browsers and newsreaders show favicon as a visual reminder of the website identity in the address bar. If the favicon is loaded from a domain other than that shown in the address bar, then the webpage is likely to be considered a Phishing attempt.

Rule: IF $\begin{cases} \text{Favicon Loaded From External Domain} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

4.3.10 Using Non-Standard Port

This feature is useful in validating if a particular service (e.g., HTTP) is up or down on a specific server. In the aim of controlling intrusions, it is much better to merely open ports that you need. Several firewalls, Proxy and Network Address Translation (NAT) servers will, by default, block all or most of the ports and only open the ones selected. If all ports are open, phishers can run almost any service they want and as a result, user information is threatened. The most important ports and their preferred status are shown in Table 2.

Rule: IF $\begin{cases} \text{Port \# is of the Preferred Status} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

POR T	Service	Meaning	Preferred Status
21	FTP	Transfer files from one host to another.	Close
22	SSH	Secure File Transfer Protocol.	Close
23	Telnet	Provide abidirectional interactive text-oriented communication.	Close
80	HTTP	Hyper test transfer protocol.	Open
443	HTTPS	Hypertext transfer protocol secured.	Open

445	SMB	Providing shared access to files, printers, serial ports.	Close
1433	MSSQL	Store and retrieve data as requested by other software applications.	Close
1521	ORACLE	Access oracle database from web.	Close
3306	MySQL	Access MySQL database from web.	Close
3389	Remote Desktop	Allow remote access and remote collaboration.	Close

4.3.11 The Existence of “HTTPS” Token in the Domain Part of the URL

The phishers may add the “HTTPS” token to the domain part of a URL in order to trick users. For example,

<http://https-www-paypal-it-webapps-mpp-home.soft-hair.com/>.

Rule: IF { Using HTTP Token in Domain Part of The URL → Phishing
Otherwise → Legitimate

4.3.12 Abnormal Based Features

Request URL examines whether the external objects contained within a webpage such as images, videos and sounds are loaded from another domain. In legitimate webpages, the webpage address and most of objects embedded within the webpage are sharing the same domain.

$$\text{Rule: IF } \begin{cases} \% \text{ of Request URL} < 22\% \rightarrow \text{Legitimate} \\ \% \text{ of Request URL} \geq 22\% \text{ and } 61\% \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{feature} = \text{Phishing} \end{cases}$$

An anchor is an element defined by the <a> tag. This feature is treated exactly as “Request URL”. However, for this feature we examine:

4.3.12.1 If the <a> tags and the website have different domain names. This is similar to request URL feature.

4.3.12.2 If the anchor does not link to any webpage, e.g.:

4.3.12.2.1

4.3.12.2.2

4.3.12.2.3

4.3.12.2.4

$$\text{Rule: IF } \begin{cases} \% \text{ of URL Of Anchor} < 31\% \rightarrow \text{Legitimate} \\ \% \text{ of URL Of Anchor} \geq 31\% \text{ And } \leq 67\% \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

Given that our investigation covers all angles likely to be used in the webpage source code, we find that it is common for legitimate websites to use <Meta> tags to offer metadata about the HTML document; <Script> tags to create a client-side script; and <Link> tags to retrieve other web resources. It is expected that these tags are linked to the same domain of the webpage.

Rule:

IF

$$\begin{cases} \% \text{ of Links in " < Meta > ", " < Script > " and " < Link > " } < 17\% \rightarrow \text{Legitimate} \\ \% \text{ of Links in " < Meta > ", " < Script > " and " < Link > " } \geq 17\% \text{ And } \leq 81\% \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

SFHs that contain an empty string or “about: blank” are considered doubtful because an action should be taken upon the submitted information. In addition, if the domain name in SFHs is different from the domain name of the webpage, this reveals that the webpage is suspicious because the submitted information is rarely handled by external domains.

$$\text{Rule: IF} \begin{cases} \text{SFH is "about: blank" Or Is Empty} \rightarrow \text{Phishing} \\ \text{SFH Refers To A Different Domain} \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

Web form allows a user to submit his personal information that is directed to a server for processing. A phisher might redirect the user’s information to his personal email. To that end, a server-side script language might be used such as “mail()” function in PHP. One more client-side function that might be used for this purpose is the “mailto:” function.

$$\text{Rule: IF} \begin{cases} \text{Using "mail()" or "mailto:"} \\ \text{Function to Submit User Information} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

This feature can be extracted from WHOIS database. For a legitimate website, identity is typically part of its URL.

$$\text{Rule: IF} \begin{cases} \text{The Host Name Is Not Included In URL} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

4.3.13 HTML and JavaScript based Features

The fine line that distinguishes phishing websites from legitimate ones is how many times a website has been redirected. In our dataset, we find that legitimate websites have been redirected one time max. On the other hand, phishing websites containing this feature have been redirected at least 4 times.

$$\text{Rule: IF } \begin{cases} \text{ofRedirect Page} \leq 1 \rightarrow \text{Legitimate} \\ \text{of Redirect Page} \geq 2 \text{ And } < 4 \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

Phishers may use JavaScript to show a fake URL in the status bar to users. To extract this feature, we must dig-out the webpage source code, particularly the “onMouseOver” event, and check if it makes any changes on the status bar.

$$\text{Rule: IF } \begin{cases} \text{onMouseOver Changes Status Bar} \rightarrow \text{Phishing} \\ \text{It Does't Change Status Bar} \rightarrow \text{Legitimate} \end{cases}$$

Phishers use JavaScript to disable the right-click function, so that users cannot view and save the webpage source code. This feature is treated exactly as “Using onMouseOver to hide the Link”. Nonetheless, for this feature, we will search for event “event.button==2” in the webpage source code and check if the right click is disabled.

$$\text{Rule: IF } \begin{cases} \text{Right Click Disabled} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

It is unusual to find a legitimate website asking users to submit their personal information through a pop-up window. On the other hand, this feature has been used in some legitimate websites and its main goal is to warn users about fraudulent activities or broadcast a welcome announcement, though no personal information was asked to be filled in through these pop-up windows.

$$\text{Rule: IF } \begin{cases} \text{Popoup Window Contains Text Fields} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

IFrame is an HTML tag used to display an additional webpage into one that is currently shown. Phishers can make use of the “iframe” tag and make it invisible i.e., without frame borders. In this regard, phishers make use of the “frameBorder” attribute which causes the browser to render a visual delineation.

$$\text{Rule: IF } \begin{cases} \text{Using iframe} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

4.3.14 Domain based Features

This feature can be extracted from WHOIS database (Whois 2005). Most phishing websites live for a short period of time. By reviewing our dataset, we find that the minimum age of the legitimate domain is 6 months.

$$\text{Rule: IF } \begin{cases} \text{Age Of Domain} \geq 6 \text{ months} \rightarrow \text{Legitimate} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

For phishing websites, either the claimed identity is not recognized by the WHOIS database (Whois 2005) or no records founded for the hostname (Pan and Ding 2006). If the DNS record is empty or not found then the website is classified as “Phishing”, otherwise it is classified as “Legitimate”.

$$\text{Rule: IF } \begin{cases} \text{no DNS Record For The Domain} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

This feature measures the popularity of the website by determining the number of visitors and the number of pages they visit. However, since phishing websites live for a short period of time, they may not be recognized by the Alexa database (Alexa the Web Information Company., 1996). By reviewing our dataset, we find that in worst scenarios, legitimate websites ranked among the top 100,000. Furthermore, if the domain has no traffic or is not recognized by the Alexa database, it is classified as “Phishing”. Otherwise, it is classified as “Suspicious”.

$$\text{Rule: IF } \begin{cases} \text{Website Rank} < 100,000 \rightarrow \text{Legitimate} \\ \text{Website Rank} > 100,000 \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phish} \end{cases}$$

PageRank is a value ranging from “0” to “1”. PageRank aims to measure how important a webpage is on the Internet. The greater the PageRank value the more important the webpage. In our datasets, we find that about 95% of phishing webpages have no PageRank. Moreover, we find that the remaining 5% of phishing webpages may reach a PageRank value up to “0.2”.

$$\text{Rule: IF} \begin{cases} \text{PageRank} < 0.2 \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

This feature examines whether a website is in Google's index or not. When a site is indexed by Google, it is displayed on search results (Webmaster resources, 2014). Usually, phishing webpages are merely accessible for a short period and as a result, many phishing webpages may not be found on the Google index.

$$\text{Rule: IF} \begin{cases} \text{Webpage Indexed by Google} \rightarrow \text{Legitimate} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

The number of links pointing to the webpage indicates its legitimacy level, even if some links are of the same domain (Dean, 2014). In our datasets and due to its short life span, we find that 98% of phishing dataset items have no links pointing to them. On the other hand, legitimate websites have at least 2 external links pointed to them.

$$\text{Rule: IF} \begin{cases} \text{Of Link Pointing to The Webpage} = 0 \rightarrow \text{Phishing} \\ \text{Of Link Pointing to The Webpage} > 0 \text{ and } \leq 2 \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

Several parties such as PhishTank (PhishTank Stats, 2010-2012), and StopBadware (StopBadware, 2010-2012) formulate numerous statistical reports on phishing websites at every given period of time; some are monthly and others are quarterly. In our research, we used 2 forms of the top ten statistics from PhishTank: "Top 10 Domains" and "Top 10 IPs" according to statistical-reports published in the last three years, starting in January 2010 to November 2012. Whereas for "StopBadware", we used "Top 50" IP addresses.

$$\text{Rule: IF} \begin{cases} \text{Host Belongs to Top Phishing IPs or} \\ \text{Top Phishing Domains} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

4.4 PROPOSED ALGORITHMS

- Decision Tree
- Support Vector Machine
- Random Forest Classification

4.4.1 Decision Tree

Decision tree is a supervised machine learning algorithm. It is one of the major classification algorithm. It is of the form of tree like structure which contains parent nodes and child nodes. It converts the data into the form of 0's and 1's respectively. These values will be applied into the decision tree.

4.4.2 Random Forest Classification

Random forest classification is also a supervised machine learning algorithm. This is based upon voting system of decision tree based upon their decisions and values.

4.4.3 Support Vector Machine

Support vector machine is a powerful machine learning algorithm used for classification and regression. Support vector machine can be used for text classification, image classification, gene expression analysis and face detection. Support vector machine is used in high dimensional data and non linear data analysis.

4.4.4 Flask Framework:

Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects. Http protocol is the foundation of data communication in world wide web. Different methods of data retrieval from specified URL are defined in this protocol.

The following table summarizes different http methods –

S.No	Methods & Description
1	GET Sends data in unencrypted form to the server. Most common method.
2	HEAD Same as GET, but without response body
3	POST Used to send HTML form data to server. Data received by POST method is not cached by server.
4	PUT Replaces all current representations of the target resource with the uploaded content.

5	DELETE Removes all current representations of the target resource given by a URL
---	--

By default, the Flask route responds to the **GET** requests. However, this preference can be altered by providing methods argument to **route ()** decorator.

In order to demonstrate the use of **POST** method in URL routing, first let us create an HTML form and use the **POST** method to send form data to a URL.

Save the following script as login.html

```
<html>

<body>

<form action="http://localhost:5000/login" method="post">

<p>Enter Name:</p>

<p><input type="text" name="nm"/></p>

<p><input type="submit" value="submit"/></p>

</form>

</body>

</html>
```

Now enter the following script in Python shell.

```
from flask import Flask, redirect, url_for, request

app=Flask(__name__)

@app.route('/success/<name>')
```

```

def success(name):

return 'welcome %s'% name

@app.route ('/login', methods=['POST','GET'])

def login ():

if request.method=='POST':

user=request.form['nm']

return redirect(url_for('success', name= user))

else:

user=request.args.get('nm')

return redirect (url_for ('success', name= user))

if __name__=='__main__':

app.run (debug =True)

```

After the development server starts running, open **login.html** in the browser, enter name in the text field and click **Submit**. As we can see in the below fig:4.2.



Fig: 4.2 Local Host Web Server

Form data is Posted to the URL in action clause of form tag.

http://localhost/login is mapped to the **login ()** function. Since the server has received data by **POST** method, value of 'nm' parameter obtained from the form data is obtained by –

```
user = request. form['nm']
```

It is passed to **'/success'** URL as variable part. The browser displays a **welcome** message in the window.



Fig: 4.3 Local Host Output Console

Change the method parameter to '**GET**' in **login.html** and open it again in the browser. The data received on server is by the **GET** method. The value of 'nm' parameter is now obtained by –

```
User = request.args.get('nm')
```

Here, **args** is dictionary object containing a list of pairs of form parameter and its corresponding value. The value corresponding to 'nm' parameter is passed on to '/success' URL as before.

CHAPTER 5

SYSTEM IMPLEMENTATION

Let's say that in a business environment, you have the perfect model in your hands, with excellent results and ideal everything. Then what? It is indeed tempting to suggest that we just hand over the code to the relevant stakeholders and ask them to run it if they want to see results. But that is not how a business environment works. One solution to this problem is by creating a machine learning web application. With this solution, stakeholders can access and run your machine learning model through a website instead of some code they don't understand. This tutorial will show you how to create a machine learning web application using Python for the machine learning model, Flask for the back-end engine, and HTML for the front-end.

5.1 Development and Deployment Setup

Machine learning is a process that is widely used for prediction. A number of algorithms are available in various libraries which can be used for prediction. In this article, we are going to build a prediction model on historical data using different machine learning algorithms and classifiers, plot the results, and calculate the accuracy of the model on the testing data.

Building/Training a model using various algorithms on a large dataset is one part of the data. But using these models within the different applications is the second part of deploying machine learning in the real world.

To put it to use in order to predict the new data, we have to deploy it over the internet so that the outside world can use it. In this article, we will talk about how we have trained a machine learning model and created a web application on it using-Flask.

We have to install many required libraries which will be used in this model. Use pip command to install all the libraries.

pip install pandas

pip install numpy

pip install sklearn

5.2 Algorithms

Decision Tree is a well-known supervised machine learning algorithm because it is easy to use, resilient and flexible. I have implemented the algorithm on adult dataset from the UCI machine learning repository.

Getting the dataset is not the end. We have to preprocess the data, which means we need to clean the dataset. Cleaning of the dataset includes different types of processes like removing missing values, filling NA values, etc.

```
# importing the dataset
import pandas
import numpy
from sklearn import preprocessing

df = pandas.read_csv('adult.csv')
df.head()
```

Pre-processing the dataset: It consists of 14 attributes and a class label telling whether the income of the individual is less than or more than 50K a year. These attributes range from the age of the person and the working-class label to relationship status and the race the person belongs to. The information about all the attributes can be found [here](#).

At first, we find and remove any missing values from the data. We have replaced the missing values with the mode value in that column. There are many other ways to replace missing values but for this type of dataset, it seemed most optimal.

```
df = df.drop(['fnlwgt', 'educational-num'], axis=1)
col_names = df.columns
for c in col_names:
    df = df.replace("?", numpy.NaN)
df = df.apply(lambda x: x.fillna(x.value_counts().index[0]))
```

The machine learning algorithm cannot process categorical data values. It can only process numerical values. To fit the data into the prediction model, we need to

convert categorical values to numerical ones. Before that, we will evaluate if any transformation on categorical columns is necessary.

Decision Tree:

Accuracy is 83.13031016480704

5.3 Testing

Now, Flask is a Python-based micro framework used for developing small-scale websites. Flask is very easy to make Restful APIs using python. As of now, we have developed a model i.e., model.pkl, which can predict a class of the data based on various attributes of the data. The class label is *Salary >=50K or <50K*.

Now we will design a web application where the user will input all the attribute values and the data will be given to the model, based on the training given to the model, the model will predict what should be the salary of the person whose details have been fed.

HTML Form: We first need to collect the data (new attribute values) to predict the income from various attributes and then use the decision tree model we build above to predict whether the income is more than 50K or less. Therefore, in order to collect the data, we create an HTML form which would contain all the different options to select from each attribute. Here, we have created a simple form using HTML only. If you want to make the form more interactive you can do so as well.

Flask script: Before starting with the coding part, we need to download flask and some other libraries. Here, we make use of a virtual environment, where all the libraries are managed which makes both the development and deployment job easier.

Now let's install Flask.

Create an environment

macOS:

Create a project folder and a venv folder within:

```
$ mkdir myproject
```

```
$ cd myproject
$ python3 -m venv venv
```

Windows:

```
mkdir myproject
> cd myproject
> py -3 -m venv venv
```

Install Flask

Within the activated environment, use the following command to install Flask:

```
$ pip install Flask
```

Application:

A minimal Flask application looks something like this:

```
from flask import Flask
```

```
app = Flask(__name__)
```

```
@app.route("/")
```

```
def hello_world():
```

```
    return "<p>Hello, World!</p>"
```

So, what did that code do?

1. First, we imported the Flask class. An instance of this class will be our WSGI application.
2. Next, we create an instance of this class. The first argument is the name of the application's module or package. `__name__` is a convenient shortcut for this that is appropriate for most cases. This is needed so that Flask knows where to look for resources such as templates and static files.
3. We then use the route() decorator to tell Flask what URL should trigger our function.
4. The function returns the message we want to display in the user's browser. The default content type is HTML, so HTML in the string will be rendered by the browser.

Save it as `hello.py` or something similar. Make sure to not call your application `flask.py` because this would conflict with Flask itself.

To run the application, use the flask command or python -m flask. You need to tell the Flask where your application is with the --app option.

```
$ flask --app hello run
```

```
* Serving Flask app 'hello'
```

```
* Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
```

Application Discovery Behavior

As a shortcut, if the file is named app.py or wsgi.py, you don't have to use --app.

See [Command Line Interface](#) for more details.

This launches a very simple builtin server, which is good enough for testing but probably not what you want to use in production. For deployment options see [Deploying to Production](#).

Now head over to <http://127.0.0.1:5000/>, and you should see your hello world greeting.

If another program is already using port 5000, you'll see OSError: [Errno 98] or OSError: [WinError 10013] when the server tries to start. See [Address already in use](#) for how to handle that.

Externally Visible Server

If you run the server you will notice that the server is only accessible from your own computer, not from any other in the network. This is the default because in debugging mode a user of the application can execute arbitrary Python code on your computer.

If you have the debugger disabled or trust the users on your network, you can make the server publicly available simply by adding --host=0.0.0.0 to the command line:

```
$ flask run --host=0.0.0.0
```

This tells your operating system to listen on all public IPs.

Debug Mode

The flask run command can do more than just start the development server. By enabling debug mode, the server will automatically reload if code changes, and will show an interactive debugger in the browser if an error occurs during a request.

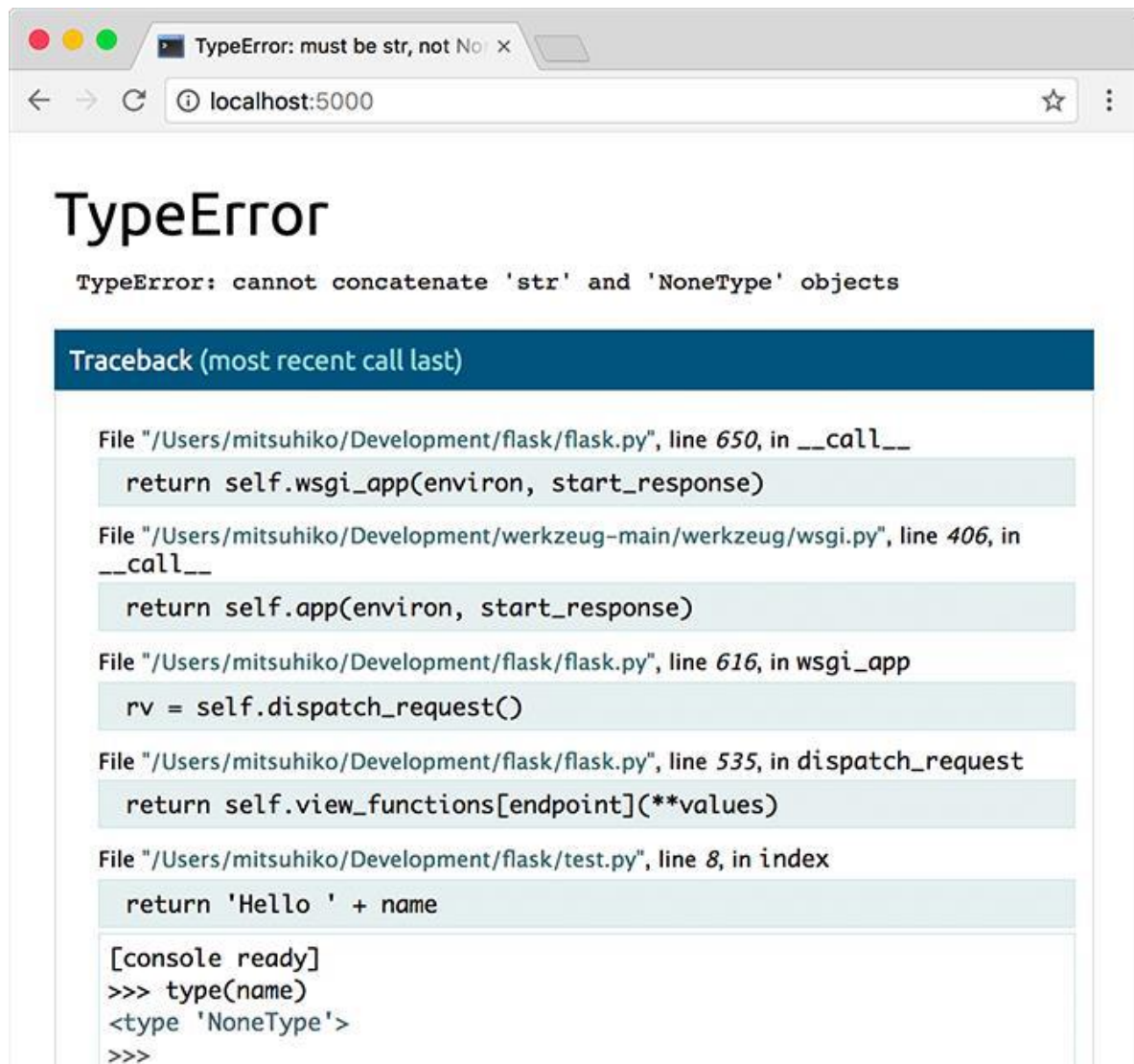


Fig: 5.1 localhost TypeError

To enable debug mode, use the --debug option.

```
$ flask --app hello run --debug
```

```
* Serving Flask app 'hello'
```

```
* Debug mode: on
```

```
* Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
```

```
* Restarting with stat
```

```
* Debugger is active!
```

```
* Debugger PIN: nnn-xxx-xxx
```

HTML Escaping

When returning HTML (the default response type in Flask), any user-provided values rendered in the output must be escaped to protect from injection attacks. HTML templates rendered with Jinja, introduced later, will do this automatically.

`escape()`, shown here, can be used manually. It is omitted in most examples for brevity, but you should always be aware of how you're using untrusted data.

```
from markupsafe import escape
```

```
@app.route("/<name>")
def hello(name):
    return f"Hello, {escape(name)}!"
```

If a user managed to submit the name `<script>alert("bad")</script>`, escaping causes it to be rendered as text, rather than running the script in the user's browser.

`<name>` in the route captures a value from the URL and passes it to the view function.

Routing

Modern web applications use meaningful URLs to help users. Users are more likely to like a page and come back if the page uses a meaningful URL they can remember and use to directly visit a page.

Use the `route()` decorator to bind a function to a URL.

```
@app.route('/')
def index():
    return 'Index Page'
```

```
@app.route('/hello')
def hello():
    return 'Hello, World'
```

Unique URLs / Redirection Behavior

The following two rules differ in their use of a trailing slash.

```
@app.route('/projects/')
def projects():
```

```

    return 'The project page'

@app.route('/about')
def about():
    return 'The about page'

```

URL Building

To build a URL to a specific function, use the `url_for()` function. It accepts the name of the function as its first argument and any number of keyword arguments, each corresponding to a variable part of the URL rule. Unknown variable parts are appended to the URL as query parameters.

Why would you want to build URLs using the URL reversing function `url_for()` instead of hard-coding them into your templates?

1. Reversing is often more descriptive than hard-coding the URLs.
2. You can change your URLs in one go instead of needing to remember to manually change hard-coded URLs.
3. URL building handles escaping of special characters transparently.
4. The generated paths are always absolute, avoiding unexpected behavior of relative paths in browsers.
5. If your application is placed outside the URL root, for example, in `/myapplication` instead of `/`, `url_for()` properly handles that for you.

For example, here we use the `test_request_context()` method to try out `url_for()`. `test_request_context()` tells Flask to behave as though it's handling a request even while we use a Python shell. See [Context Locals](#).

```
from flask import url_for
```

```

@app.route('/')
def index():
    return 'index'

```

```

@app.route('/login')
def login():
    return 'login'

@app.route('/user/<username>')

```

```
def profile(username):
    return f'{username}'s profile'

with app.test_request_context():
    print(url_for('index'))
    print(url_for('login'))
    print(url_for('login', next='/'))
    print(url_for('profile', username='John Doe'))
```

HTTP Methods

Web applications use different HTTP methods when accessing URLs. You should familiarize yourself with the HTTP methods as you work with Flask. By default, a route only answers to GET requests. You can use the `methods` argument of the `route()` decorator to handle different HTTP methods.

```
from flask import request
```

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        return do_the_login()
    else:
        return show_the_login_form()
```

Static Files

Dynamic web applications also need static files. That's usually where the CSS and JavaScript files are coming from. Ideally your web server is configured to serve them for you, but during development Flask can do that as well. Just create a folder called `static` in your package or next to your module and it will be available at `/static` on the application.

To generate URLs for static files, use the special `'static'` endpoint name:

```
url_for('static', filename='style.css')
```

The file has to be stored on the filesystem as `static/style.css`.

Redirects and Errors

To redirect a user to another endpoint, use the [redirect\(\)](#) function; to abort a request early with an error code, use the [abort\(\)](#) function:

```
from flask import abort, redirect, url_for
```

```
@app.route('/')
```

```
def index():
```

```
    return redirect(url_for('login'))
```

```
@app.route('/login')
```

```
def login():
```

```
    abort(401)
```

```
    this_is_never_executed()
```

5.4 RESULTS AND DISCUSSION DETAILS

Trying to gather personal information through deceptive ways is becoming more common nowadays. In order to assist the user to be aware of the access to such websites, the implemented system notifies the user through email and also pop-up, when trying to access a phishing site. This paper proposes an approach of phishing detection system to detect blacklisted URL also known as phishing websites, so that individual can be alerted while browsing or accessing a particular website. Therefore, it can be utilized for identification and authentication and become a legitimate tool to prevent an individual from getting tricked.

We deployed RF, and DT for feature selection more to enhance the classification accuracy of the deployed models. Overall, RF performed the best in both circumstances. RF and SVM achieved perfect accuracy on the first dataset owing to the original characteristics study as well as the associated aspects study, but the second dataset did not fare as well. Because it was found that the initial dataset had more associated traits that were lacking in the second. For this reason, we conducted an extensive series of studies to learn what factors influence the recognition of the intelligent models while making the determination of whether or not a site is phishing.

In this modern world, the technology has become a major threat to our daily lives. Staying secure from the cyber-attacks is considered as the most important task. Being able to say that whether a particular website is phishing or not with the help of same technology is considered as the safest way available.

CHAPTER 6

CONCLUSION

6.1 Conclusion

An increasing number of phishing websites and internet users have necessitated innovative methods for detecting these malicious domains. Numerous studies have been conducted on the topic of distinguishing phishing websites from legal ones by using vast datasets and clever approaches. In this post, we've focused mostly on features development as we worked to create a number of modeling software and identify the aspects that influence these systems' ability to identify malicious urls.

6.2 Future Work

We recommend further research into other datasets and the use of deeper DL models to improve and compare performance, as well as the identification of additional essential factors in detecting phishing sites.

Future work will aim to develop a system that can learn by itself about new types of phishing attacks by adding a more enhanced feature to the detection process. The scope of this approach not only helps in adding more enhanced features but also updating the existing features to improve its importance level to make detection more efficient and reduce the false positive rate to a large extent. Another further work should include deploying this approach into a web extension to make the detection more robust to the user.

Another further work should include deploying this approach into a web extension to make the detection more robust to the user.

REFERENCES

- [1] A. Awasthi and N. Goel, "Phishing Website Prediction: A Comparison of Machine Learning Techniques," in *Data Intelligence and Cognitive Informatics*, Springer. Singapore. 2021, pp. 637-650.
- [2] A. Hashim, R. Medani and T. A. Attia, "Defences Against web Application Attacks and Detecting Phishing Links Using Machine Learning," 2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE), Khartoum, Sudan, 2021, pp. 1-6, doi:10.1109/ICCCEEE49695.2021.9429609..
- [3] M. Aljabri and S. Mirza, "Phishing Attacks Detection using Machine Learning and Deep Learning Models," 2022 7th International Conference on Data Science and Machine Learning Applications (CDMA), Riyadh, Saudi Arabia, 2022, pp. 175-180, doi: 10.1109/CDMA54072.2022.00034..
- [4] : M. A. Syafiq Rohmat Rose, N. Basir, N. F. Nabila Rafie Heng, N. Juana Mohd Zaizi and M. M. Saudi, "Phishing Detection and Prevention using Chrome Extension," 2022 10th International Symposium on Digital Forensics and Security (ISDFS), Istanbul, Turkey, 2022, pp. 1-6, doi: 10.1109/ISDFS55398.2022.9800826..
- [5] M. Chatterjee and A. -S. Namin, "Detecting Phishing Websites through Deep Reinforcement Learning," 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), Milwaukee, WI, USA, 2019, pp. 227-232, doi: 10.1109/COMPSAC.2019.10211.
- [6] M. Korkmaz, O. K. Sahingoz, and B. Diri, "Detection of Phishing Websites by Using Machine Learning-Based URL Analysis," in 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2020, pp. 1- 7, doi: 10.1109/ICCCNT49239.2020.9225561.

- [7] M. Tubyte and A. Paulauskaite-Taraseviciene, "Research on phishing email detection based on URL parameters using machine learning algorithms," CEUR Workshop Proc., vol. 2915, pp. 18- 26, 2021.
- [8] N. Bari and M. Ali Shah, "SECURING DIGITAL ECONOMIES: DETECTION OF PHISHING ATTACKS USING MACHINE LEARNING APPROACHES," Competitive Advantage in the Digital Economy (CADE 2021), Online Conference, 2021, pp. 105-111, doi: 10.1049/icp.2021.2412.
- [9] R. Mohammad, T. L. McCluskey, and F. Thabtah, "UCI Machine Learning Repository: Phishing Websites Data Set." <https://archive.ics.uci.edu/ml/datasets/phishing+websites> (accessed Nov. 20, 2021).
- [10] S. Abeysooriya, "Phishing Detection Dataset | Kaggle," 2021. <https://www.kaggle.com/sandunabeysooriya/phishing-detectiondataset> (accessed Nov. 17, 2021).
- [11] S. A. Khan, W. Khan, and A. Hussain, "Phishing Attacks and Websites Classification Using Machine Learning and Multiple Datasets (A Comparative Analysis)," in International Conference on intelligent Computing Methodologies, Springer. Cham. 2020, pp. 301-313.
- [12] S. P. Ripa, F. Islam and M. Arifuzzaman, "The Emergence Threat of Phishing Attack and The Detection Techniques Using Machine Learning Models," 2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI), Rajshahi, Bangladesh, 2021, pp. 1-6, doi: 10.1109/ACMI53878.2021.9528204.
- [13] S. Hossain, D. Sarma, and R. J. Chakma, "Machine learning-based phishing attack detection," Int. J. Adv. Comput. Sci. Appl., vol. 11, no. 9, pp. 378-388, 2020, doi: 10.14569/IJACSA.2020.0110945.
- [14] T. Olowookere, A. O. Adetunmbi, and J. O. Ajayi, "Phishing dataset." 2021. (accessed Nov. 17, 2021)

[15] R. Kiruthiga and D. Akila, "Phishing websites detection using machine learning," *Int. J. Recent Technol. Eng.*, vol. 8, no. 2 Special Issue 11, pp. 111-114, 2019, doi: 10.35940/ijrte.B1018.0982S1119. Akila, "Phishing websites detection using machine learning," *Int. J. Recent Technol. Eng.*, vol. 8, no. 2 Special Issue 11, pp. 111-114, 2019, doi: 10.35940/ijrte.B1018.0982S1

APPENDIX

A. SOURCE CODE

```
def f1_having_IP_Address(domain):
    #1. checking hex digits
    domainhex = domain
    domainhex = domainhex.replace(".", "")
    domainhex = domainhex.replace("0x", "0")
    count = 0
    for dig in domainhex:
        if dig in string.hexdigits:
            count+=1
    if(count>=len(domainhex)):
        return -1
    #2. checking direct IP address
    try:
        ipaddress.ip_address(domain)
        return -1
    except:
        return 1
def f2_URL_Length(url):
    if len(url)>=54:
        if len(url)>75:
            return -1
        return 0
    return 1
def f3_Shortening_Service(url):
    services = ['bit.ly', 'tinyurl.com', 'goo.gl', 'tr.im', 'is.gd', 'cli.gs', 'yfrog.com',
'migre.me', 'rebrand.ly', 't.co', 'youtu.be', 'ow.ly', 'w.wiki', 'ff.me', 'tiny.cc',
'ur14.eu', 'twurl.nl', 'snipurl.com', 'short.to' ]
    ind = url.find("https://")
    newurl = url
    if(ind!=-1):
        newurl = url[(ind+8):]
```

```

index = newurl.find("/")
if(index==-1):
    return 1
service = newurl[:index]
if service in services:
    return -1
else:
    return 1
def f4_having_At_Symbol(url):
    if "@" in url:
        return -1
    return 1
def f5_double_slash_redirecting(url):
    pos = url.rfind("//")
    if (pos > 5):
        if (pos > 6):
            return -1
        else:
            return 1
    else:
        return 1
def f6_Prefix_Suffix(domain):
    if "-" in domain:
        return -1
    return 1
def f7_having_sub_domain(domain):
    subdom = domain[3: domain.rfind(".")]
    count = 0
    for pt in subdom:
        if pt==".":
            count+=1
    if count>1:
        if count>2:
            return -1

```

```

        else:
            return 0
    else:
        return 1
def f9_Domain_registration_length(domain):
    try:
        w = whois.whois(domain)
    except Exception:
        return -1
    else:
        whois_info = whois.whois(domain)
        exp_date = whois_info.expiration_date
        try:
            today = datetime.datetime.now()
            length = exp_date[0] - today
            if length>datetime.timedelta(days=365):
                return 1
            else:
                return -1
        except:
            return -1
def f10_Favicon(domain):
    href = []
    for icon in soup.find_all("link", rel="shortcut icon"):
        href.append(icon.get("href"))
    for h in href:
        if(h=="favicon.ico"):
            return 1
        else:
            dom = urlparse(h).netloc
            if (dom!=domain and dom!=" and dom!=b"):
                return -1
    return 1
def f11_HTTPS_token(domain):

```

```

    if ("https" in domain) or ("http" in domain):
        return -1
    return 1
def checkperc(oth, cnt):
    perc = 0
    if(cnt!=0):
        perc = (oth*100)/cnt
    return perc
def f12_Request_URL(domain):
    percimg = f121_findsrcdomain('img', domain)
    percvid = f121_findsrcdomain('video', domain)
    percsound = f121_findsrcdomain('embed', domain)
    perc = percimg + percvid + percsound
    if perc>=21:
        if perc>=61:
            return -1
        else:
            return 0
    else:
        return 1
def f121_findsrcdomain(tag, domain):
    srcs = []
    invalidhref = 0
    count = 0
    for t in soup.find_all(tag):
        srcs.append(t.get('src'))
    for src in srcs:
        dom = urlparse(src).netloc
        if (dom!=domain and dom!=" and dom!=b"):
            invalidhref+=1
    count=len(srcs)
    perc = checkperc(invalidhref, count)
    return (perc)
def f13_URI_of_Anchor(domain):

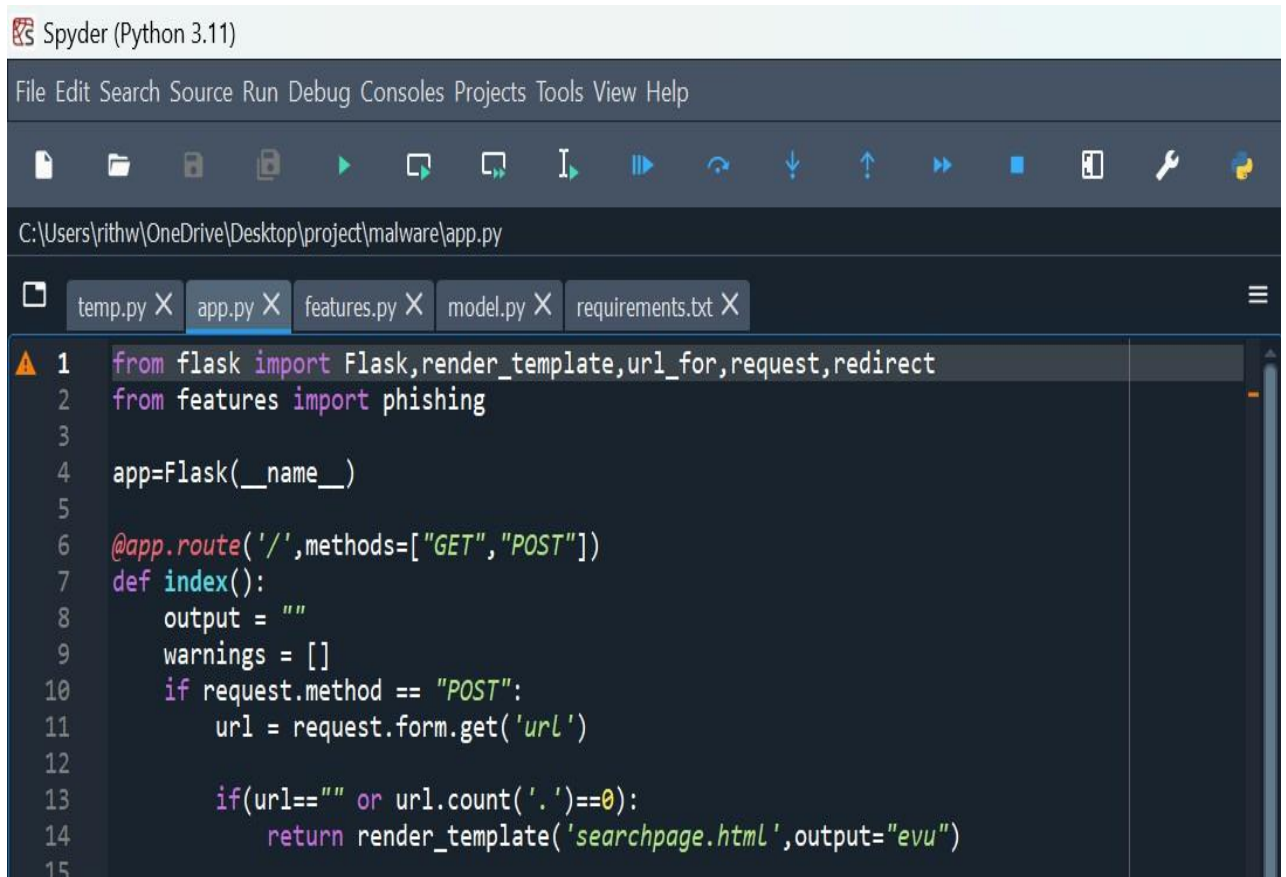
```

```

href = []
count = 0
invalidhref = 0
for a in soup.find_all('a'):
    href.append(a.get('href'))
nullweb = ['#', '#content', '#skip', 'JavaScript ::void(0)']
for h in href:
    if h in nullweb:
        invalidhref+=1
    else:
        dom = urlparse(h).netloc
        if (dom!=domain and dom!=" and dom!=b"):
            invalidhref+=1
count=len(href)
perc = checkperc(invalidhref, count)
if perc>=31:
    if perc>67:
        return -1
    else:
        return 0
return 1

```

B. SCREENSHOTS



The screenshot displays the Spyder Python IDE interface. The title bar indicates 'Spyder (Python 3.11)'. The menu bar includes 'File', 'Edit', 'Search', 'Source', 'Run', 'Debug', 'Consoles', 'Projects', 'Tools', 'View', and 'Help'. The toolbar contains various icons for file operations and execution. The file explorer shows the path 'C:\Users\rithw\OneDrive\Desktop\project\malware\app.py'. The editor window has tabs for 'temp.py', 'app.py', 'features.py', 'model.py', and 'requirements.txt'. The code in 'app.py' is as follows:

```
1 from flask import Flask,render_template,url_for,request,redirect
2 from features import phishing
3
4 app=Flask(__name__)
5
6 @app.route('/',methods=["GET","POST"])
7 def index():
8     output = ""
9     warnings = []
10    if request.method == "POST":
11        url = request.form.get('url')
12
13        if(url=="" or url.count('.')==0):
14            return render_template('searchpage.html',output="evu")
15
```

```

15
16     val, warnings = phishing(url)
17     val = int(val)
18     if val==1:
19         output = "Legitimate"
20     elif val==0:
21         output = "Suspicious"
22     elif val==404:
23         output="evu"
24     else:
25         output = "Phishing"
26     return render_template('searchpage.html', output = output, warnings = warnings)
27
28 @app.after_request
29 def add_header(r):
30     """
31     Add headers to both force latest IE rendering engine or Chrome Frame,
32     and also to cache the rendered page for 10 minutes.
33     """
34     r.headers["Cache-Control"] = "no-cache, no-store, must-revalidate"
35     r.headers["Pragma"] = "no-cache"
36     r.headers["Expires"] = "0"
37     r.headers['Cache-Control'] = 'public, max-age=0'
38     return r
39
40 if __name__ == "__main__":
41     app.run(debug=True)
42

```

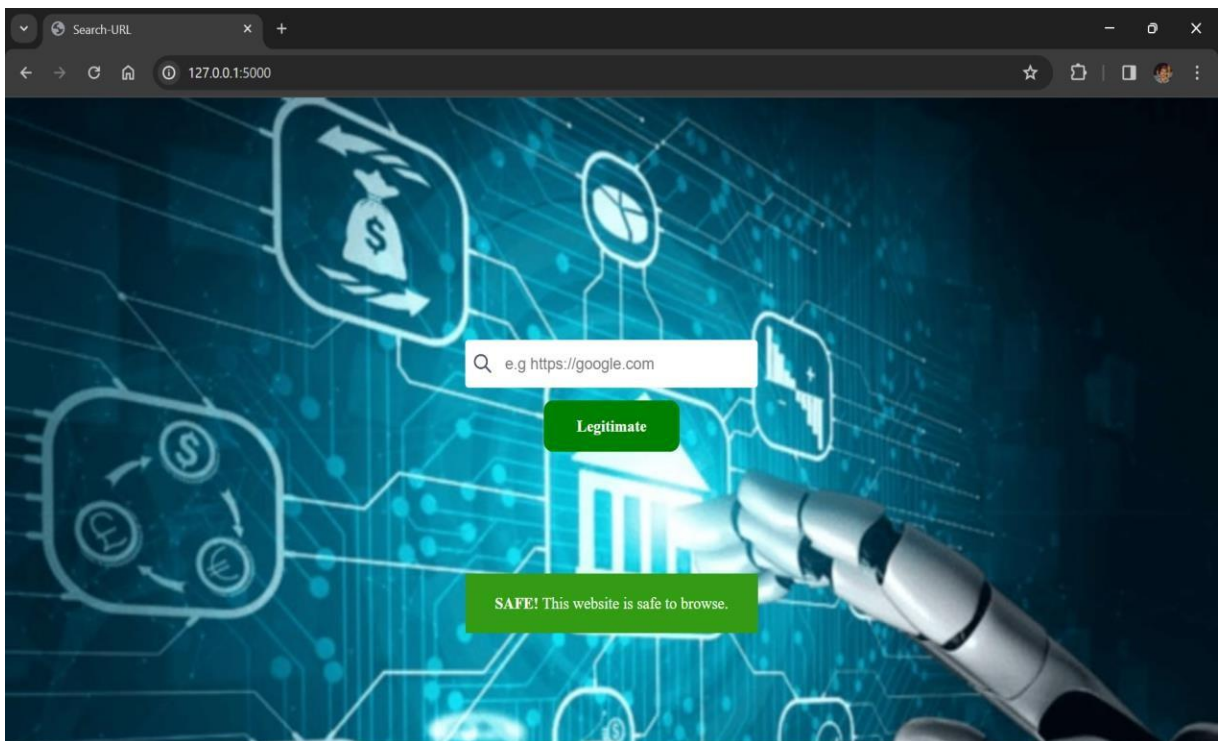
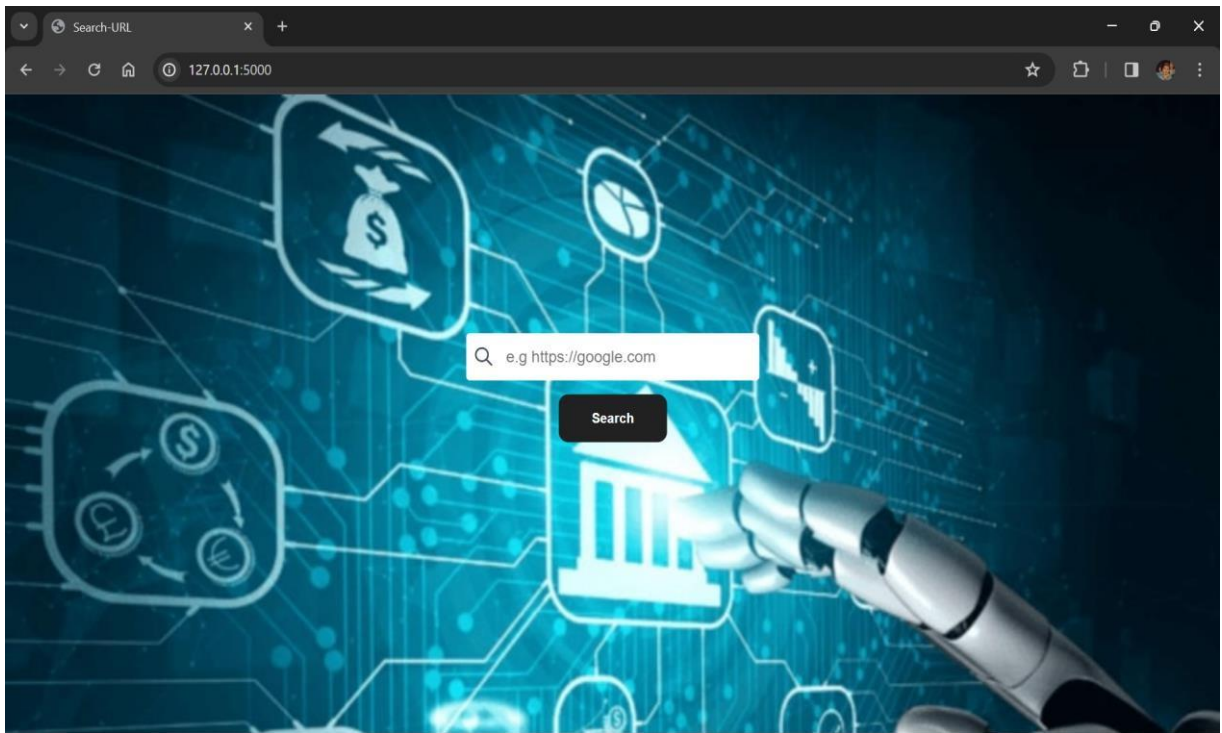
```

13:26:23] [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.15.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/rithw/OneDrive/Desktop/project/malware/
app.py', wdir='C:/Users/rithw/OneDrive/Desktop/project/malware')
* Serving Flask app 'app'
* Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a
production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with watchdog (windowsapi)
INFO:werkzeug: * Restarting with watchdog (windowsapi)

```



A1

X

✓

f_x

having_IP_Address

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	
1	having_IP_Address	URL_Length	ShortURL	having_A_double_slash	Prefix_Suffix	having_SSID	Final_Domain	Favicon	port	HTTPS_to_HTTP	Request_URL	of_A_Links_in_SPH	Submitter	Abnormal	Redirect	on_mouse_click	Clicked	popup	Wiframe	age_of_dnsrecord	web_traffic	Page_Rank	Google_Links	pool	Statistics	Result									
2	-1	1	1	1	-1	-1	-1	-1	-1	1	1	-1	1	-1	1	-1	-1	-1	0	1	1	1	1	-1	-1	-1	-1	1	1	-1	-1				
3	1	1	1	1	1	-1	0	1	-1	1	1	-1	1	0	-1	-1	1	1	0	1	1	1	1	-1	-1	0	-1	1	1	1	-1	-1			
4	1	0	1	1	1	-1	-1	-1	-1	1	1	-1	1	0	-1	-1	-1	-1	0	1	1	1	1	1	-1	1	-1	1	0	-1	-1				
5	1	0	1	1	1	-1	-1	-1	-1	1	1	1	-1	-1	0	0	-1	1	1	0	1	1	1	1	-1	-1	1	-1	1	-1	1	-1			
6	1	0	-1	1	1	-1	1	1	-1	1	1	1	1	0	0	-1	1	1	0	-1	1	-1	1	-1	-1	0	-1	1	1	1	1	1			
7	-1	0	-1	1	-1	-1	1	1	-1	1	1	-1	1	0	0	-1	-1	-1	0	1	1	1	1	1	1	1	1	-1	1	-1	-1	1			
8	1	0	-1	1	1	-1	-1	-1	1	1	1	1	1	-1	-1	0	-1	-1	-1	0	1	1	1	1	1	-1	-1	-1	1	0	-1	-1			
9	1	0	1	1	1	-1	-1	-1	-1	1	1	1	-1	-1	0	-1	-1	1	1	0	1	1	1	1	-1	-1	0	-1	1	0	1	-1			
10	1	0	-1	1	1	-1	1	1	-1	1	1	-1	1	0	1	-1	1	1	0	1	1	1	1	1	1	-1	1	1	1	0	1	1			
11	1	1	-1	1	1	-1	-1	1	-1	1	1	1	1	0	1	-1	1	1	0	1	1	1	1	1	1	-1	0	-1	1	0	1	-1			
12	1	1	1	1	1	-1	0	1	1	1	1	1	1	-1	0	0	-1	-1	-1	0	1	1	1	1	-1	1	1	1	1	-1	-1	1			
13	1	1	-1	1	1	-1	1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	-1	0	1	1	1	1	-1	-1	-1	-1	1	0	-1	-1			
14	-1	1	-1	1	-1	-1	0	0	1	1	1	-1	-1	-1	-1	1	-1	1	0	-1	1	-1	1	1	-1	-1	-1	-1	1	0	1	-1			
15	1	1	-1	1	1	-1	0	-1	1	1	1	1	-1	-1	-1	-1	1	1	0	1	1	1	1	1	-1	-1	0	-1	1	1	1	-1			
16	1	1	-1	1	1	1	-1	1	-1	1	1	-1	1	0	1	1	1	1	1	0	1	1	1	1	1	-1	-1	-1	1	-1	1	1	1		
17	1	-1	-1	-1	1	-1	0	0	1	1	1	1	-1	-1	0	-1	1	1	0	1	1	1	1	1	1	-1	-1	-1	1	0	1	-1			
18	1	-1	-1	1	1	-1	1	1	-1	1	1	-1	1	0	-1	-1	-1	-1	0	1	1	1	1	1	1	-1	0	-1	1	1	-1	-1			
19	1	-1	1	1	1	-1	-1	0	1	1	-1	1	1	0	-1	-1	-1	-1	0	1	1	1	1	1	-1	1	1	-1	1	1	-1	-1			
20	1	1	1	1	1	-1	-1	1	1	1	1	-1	-1	0	-1	-1	-1	-1	0	1	1	1	1	1	1	-1	-1	1	1	-1	-1	1			
21	1	1	1	1	1	-1	-1	1	-1	1	1	1	1	0	0	-1	-1	-1	0	-1	-1	-1	-1	-1	1	-1	0	-1	1	0	-1	1			
22	1	0	-1	1	1	-1	0	1	-1	1	1	1	1	0	0	-1	-1	-1	0	-1	1	-1	1	-1	1	-1	1	1	-1	-1	1	-1	-1		
23	1	0	1	1	1	-1	0	1	1	1	1	-1	-1	0	-1	-1	-1	-1	0	1	1	1	1	1	-1	1	-1	-1	1	0	-1	1			
24	1	1	1	1	1	-1	-1	-1	-1	1	1	-1	1	0	0	-1	1	1	0	1	1	1	1	1	1	1	1	0	-1	1	-1	1	1		
25	1	1	1	1	1	-1	1	0	-1	1	1	1	1	0	0	-1	1	1	0	1	1	1	1	1	1	1	1	-1	1	-1	1	1			
26	1	-1	-1	-1	1	-1	1	1	-1	1	1	-1	-1	0	0	-1	1	1	0	1	1	1	1	1	1	1	-1	-1	1	0	1	1			
27	1	-1	1	1	1	-1	0	1	-1	1	1	1	1	1	0	-1	1	1	0	1	1	1	1	1	-1	1	1	-1	1	0	1	1			
28	1	-1	1	1	1	-1	0	-1	1	1	1	-1	-1	-1	-1	-1	-1	-1	0	1	1	1	1	1	1	1	0	-1	1	-1	-1	-1			
29	1	-1	-1	1	1	1	-1	1	1	1	1	1	-1	1	0	-1	-1	-1	0	1	1	1	1	1	1	-1	0	-1	1	0	-1	1			
30	1	-1	-1	1	-1	1	-1	1	-1	1	1	1	1	1	0	-1	1	1	1	1	1	1	1	1	-1	-1	1	-1	1	-1	1	1			
31	1	-1	1	1	1	-1	-1	1	-1	1	1	1	1	1	0	-1	1	1	0	1	1	1	1	1	-1	1	1	1	1	0	1	1			
32	1	-1	1	1	1	-1	-1	1	-1	1	1	-1	1	0	1	-1	1	1	0	1	1	1	1	1	-1	-1	-1	-1	1	0	1	1			
33	1	-1	1	1	1	-1	-1	1	-1	-1	1	-1	1	0	-1	-1	-1	-1	0	-1	1	-1	-1	-1	1	-1	1	1	1	0	-1	1			
34	1	-1	1	1	1	1	1	1	-1	1	1	1	1	1	1	1	-1	-1	0	1	1	1	1	1	1	1	-1	-1	1	-1	-1	1			
35	1	0	1	1	1	-1	-1	1	-1	1	1	1	1	0	1	-1	-1	-1	0	1	1	1	1	1	1	1	1	-1	1	0	-1	1			
36	1	0	1	1	1	-1	0	0	1	1	1	1	1	-1	-1	-1	-1	-1	0	1	1	1	1	1	1	-1	1	-1	1	1	-1	-1			
37	1	0	1	1	1	-1	-1	-1	-1	1	1	-1	1	-1	0	-1	1	1	0	1	1	1	1	1	-1	-1	-1	-1	1	1	1	-1			
38	1	0	1	1	1	-1	-1	1	-1	-1	1	-1	-1	-1	0	-1	1	1	0	-1	1	-1	-1	-1	-1	-1	0	-1	1	0	1	-1			
39	1	0	1	1	1	-1	-1	-1	1	-1	1	1	-1	-1	0	-1	1	1	0	-1	1	-1	-1	-1	-1	-1	0	-1	1	0	1	-1			
40	1	1	1	1	1	1	-1	1	-1	1	1	1	1	1	1	-1	-1	-1	0	1	1	1	1	1	1	-1	1	-1	-1	1	-1	1			
41	1	-1	1	1	1	-1	0	-1	-1	1	1	-1	-1	1	0	-1	1	1	0	1	1	1	1	1	-1	-1	1	-1	1	0	1	1			

<

>

dataset

+

</

WEB SHIELD: Shielding your Browsing Experience using Decision Tree

Mallidi Venkata Sukumar Reddy

Department of CSE
Sathyabama Institute of
Science and Technology, Chennai

Mandapalli Rithwik Reddy

Department of CSE
Sathyabama Institute of
Science and Technology, Chennai

L. Sujihelen

Department of CSE
Sathyabama Institute of Science and Technology,
Chennai

sujihelen.cse@sathyabama.ac.in

Abstract— Phishing websites aim to steal the victims' private information by tricking them into visiting a fraudulent website that looks real. This is a different kind of online crime and is particularly concerning for a number of industries, such as retailing and e-managing accounts. Getting hold of supersensitive information, such bank account information and passwords, is the ultimate aim. Spoof messages are used by malicious URLs to appear genuine. resources include monetary foundations, online shopping sites, presents, discounts, and more. Phishing attacks can happen in a wide range of domains, including but not limited to online payment, financial institutions, webmail, file hosting, and cloud storage. Detecting malicious URLs with high accuracy is the aim of the proposed system. Decision tree algorithms are utilised in the suggested approach to identify phishing websites. Lastly, we assessed and contrasted the classifier's accuracy performance.

Keywords—*Machine learning, phishing websites, Deep learning, Random forest.*

I. INTRODUCTION

Since the Internet is widely used in our connected world and network technology is advancing, more complex cyberattacks are possible, which emphasises the need for robust cybersecurity [1]. The creation and dissemination of malicious URLs via pop-up ads, hyperlinks, newsletters, and advertisements is a major concern. These malicious URLs carry significant hazards, such as the possibility for ransomware to infect computer systems, financial losses, and illegal access to private data. In order to detect fake URLs, two main techniques are frequently employed [2][3]. The first approach makes use of an ever-changing blacklist that gets updated whenever new URLs that might be harmful are found. Products that use this technique include Content Filtering Threat Seeker and Google Secure Browser [4]. This approach works well against known fraudulent URLs. By creating signatures based on behavioral traits, this approach shows enhanced generalization abilities compared to blacklist-based methods. However, since it depends on predetermined signatures, intruders can potentially bypass these systems, making machines using this strategy unable to recognize the new malicious URLs. Another strategy will be using AI-based methods for the detection, such as Machine Learning, with different classifier for detecting the threat through the URLs. This approach has gained popularity to

suppress the limitations of earlier method. ML implementation requires a test set with dangerous and safe URLs, using various characteristics that are associated with the URLs to create a training-set. Trained ML models can then categorize new URLs based on the acquired information. This method we are going to study, the propose an ML-based technique for the detection of the malicious URLs, using two datasets: one compiled by Mohamad [5][6]. As of the date of the initial dataset, 11,550 records had been included. In the latter dataset, there were 14,903 records. Lexical, domain, and functional categories are found in both samples. Feature selection was conducted on both datasets, and various machine learning techniques, including SVM, RF, and DT, were assessed for quality. Among the models which were tested based on the initial dataset, RF and SVM both achieved a accurate rate on detecting the spamming. Additional features in the initial dataset, focusing on domain-based aspects such as page prominence in operation, contributed to these superior outcomes. Attributes like Google indexes, subdomain attributes indicating webpage inclusion in Google's anchor tags, search tags, and access data elements were found to impact website usability, influencing the models' ability to identify malicious sites. This article analyzes the factors influencing models in identifying phishing from non-phishing websites and investigates optimal feature selections which enhances ML's and DL's performance.

II. LITERATURE REVIEW

Researchers have explored diverse AI-based approaches, this will be including and Machine Learning (ML) Deep Learning (DL) algorithms, to discern harmful URLs. These methods mainly relay on properties associated with URLs, broadly categorized into lexical based, domain based and material functionalities. URLs statistically, such as duration and special character count, length, and encryption status. "Information features" pertain to webpage characteristics related to content, encompassing the elements in HTML files and the metadata links. Within the domain of phrases, "property features" were directly associate to the domain of

website.

The gauge effectiveness to the ML models in which it identifies the malicious URLs, Khan [7] has tested them with 3 diverse datasets, highlighting Random Forest, Artificial Neural Network as superior techniques with an efficiency of over 97.2%. Overlapping features across all three databases showcased the resilience of the model's performance. Awasthi [8] compared various ML algorithms. Kokmaz [9] scrutinized about 59 distinct aspects associated with the name, sub-domain, and parts of the URL, emphasizing lexical-based variables. Testing multiple ML model's based on the three datasets, they found that Random Forest demonstrated the best overall performance. Hossain [10] identified site spoofing attempts using a balanced dataset for 10,000 records which has 46 characteristics that relates to URLs' linguistics and site properties. Validating the effectiveness of the Random Forest algorithm that detects the phishing websites which has an F-score of 0.98, their research achieved an accuracy of 92% or above with a limited dataset. Additionally, the algorithm was able to achieve the best accuracy of 95% from both datasets. Kiruthigah[11] [12], focusing on the same two groups of characteristics and employing fewer attributes for website categorization (safe, doubtful, phishing), utilized the dataset comprising nine features and about 1,352 URLs[13][14]. Their findings indicated Support Vector Machine and most effective method was the Decision Tree with an accuracy of 91%. The study acknowledged limitations due to data quantity and attribute quantity. This research enriches the field by comparing the effectiveness of various ML models designed for identifying deceptive websites[15]. The impact in variable selection and accuracy is scrutinized, emphasizing three significant methods of the characteristics.

III Proposed System

In this proposed study aiming to differentiate and categorize phishing websites from legitimate ones. To achieve this goal, two initially published phishing datasets [5][6] were utilized. The examination of these datasets involved a detailed analysis of their structure, size, and limitations. Subsequently, preprocessing techniques were applied to address class imbalance issues. Feature selection focused on identifying attributes with the highest correlation. The final phase encompassed the implementation of classification models, and the resulting outcomes underwent a comprehensive analysis. The entire process is visually represented in Figure 1 below.

A. Dataset Description

In our experimentation, we utilized two datasets and conducted a comparative analysis of their outcomes. Initially, our research made use of Mohammad's UCI phishing collection [5], which is accessible through Olowookere . This dataset comprises 32 attributes, totaling 11,056 instances with no missing values. It represents a binary value indicating whether or not the website is legitimate or phishing. Specifically, 6158 instances are categorized as legitimate, while 4898 instances are identified as phishing attempts. The second dataset, consisting of 14,093 instances, also maintains the phishing/legitimate status as a binary value. In this dataset, 7044 instances are classified as fraudulent and phishing attempts, and 7049 instances are marked as legitimate. While sharing similar characteristics with the original dataset, it lacks 5 features present in the original. Notably, both datasets include features related to lexicon, #

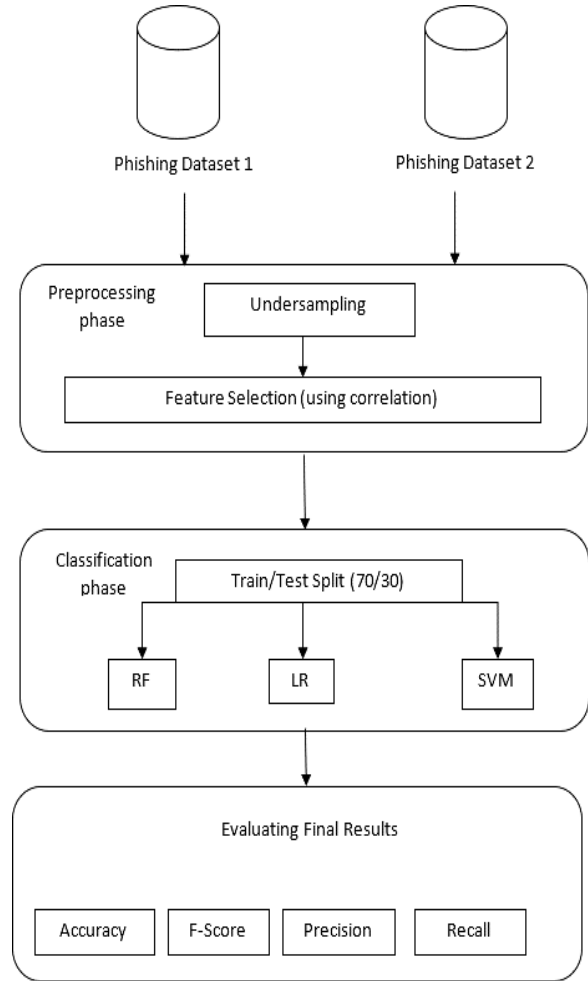


Fig 1: Workflow of phishing classification

domain, and content.

B. Preprocessing

To address the issue of class imbalance, we implemented under-sampling. In the initial dataset, the genuine classification had only 6152 occurrences, while the fraudulent subclass had 4890 examples, creating an imbalance. The technique used to mitigate this is known as "randomized under-sampling," where the number of dominant classifiers is reduced until it is approximately equal's the number of minor classifiers among the dataset. Even after the sampling, the deceptive category still had about 4890 occurrences', and the legitimate category also contains 4890 occurrences. Similarly, in the second dataset has 7044 occurrences each. Representing the random under sampling method applies to the first dataset that is illustrated in Figure 2.

C. Feature Selection

This selection method plays a pivotal role in the improving of model performance by identifying the most significant traits that contribute to enhanced accuracy. In this study, the chosen method for feature selection was correlation analysis is shown in figure 3. It is a statistical technique used to determine the relationship between the independent variables and the dependent attribute. A total of 16 characteristics from both datasets underwent analysis using correlation analysis. This process facilitated the development of intelligent models, and their performance was assessed based on the selected features.

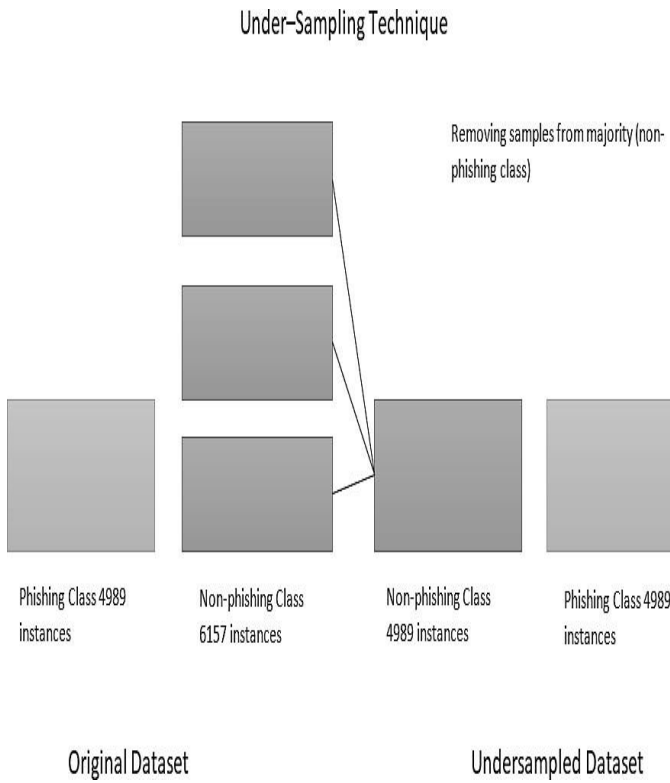


Fig. 2: Under-sampling technique

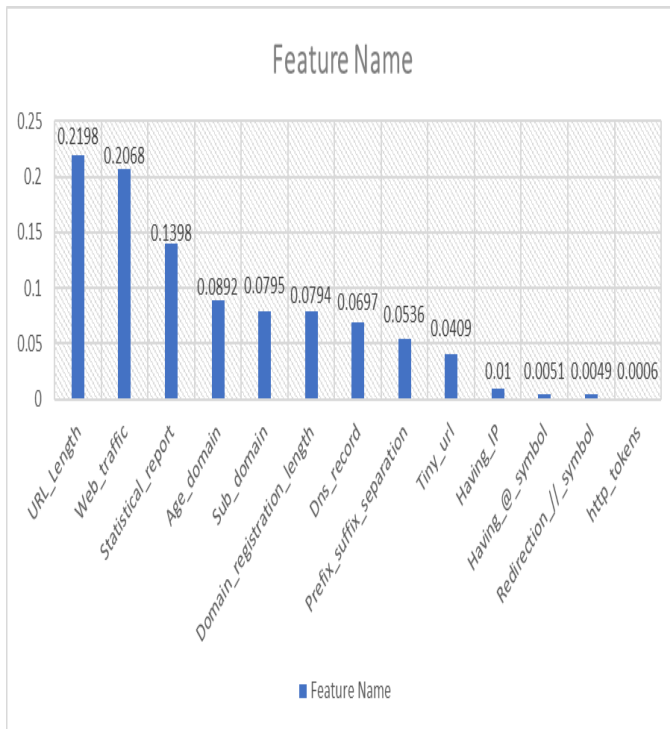


Figure. 3 Correlation values in the two datasets

Table I : First Dataset Information Attribute

#	Attributes	Description	Feature Type
1	Having an IP	Checking URL for IP address	Lexical
2	Length_URL	Length of URL	Lexical
3	Tiny URL's	Checks if URL's is short in it	Lexical
4	Consisting of _@_symbol.	Checks for "@" is in dataset	Lexical
5	Prefix/suffix _separation.	Checks if the URL has a "- "	Lexical
6	Sub domain.	Checks sub domains are more than 2	Lexical
7	Protocols.	http or https protocols	Domain
8	domain_registration_length	Comparing the present length and registration length of link	Domain
9	Path	URL path which directs to another website	Content
10	Redirecting	Checks the meta tags that redirects to another website	Content
11	Age domain	The age of the domain name	Domain
12	Dns record	Checking the domain is registered based on WHOIS site	Domain
13	Web traffic	Checks frequently visiting the website	Domain
14	Page Ranking	Rank of website in the internet.	Domain
15	Label	Checks whether the website is Google Indexed.	Domain
16	Statistical report	Checks the URL is in phishing list(Phish Tank)	Domain

Upon scrutinizing the correlation coefficients, it becomes apparent in the initial dataset contains some vital property that is not present in the second dataset. For the second dataset, our approach was to add the features from the primary dataset along with the existing ones and missing ones. The characteristics shared between the two datasets are listed in Table I along with those unique to the primary dataset. As shown in table II, attributes from dataset2 are different from those from dataset1.

TABLE II: Additional Information attribute's in dataset 2 than in dataset 1

#	Attributes	Description	Feature Type
1	Third party object	Different and abnormal behavior in the	Content
2	anchor tag	Different behavior in the < a > tag	Content

In table III it shows the similar features in dataset1 and dataset2.

TABLE III Similar feature in dataset1 and Dataset 2

#	Attributes	Feature presence in the dataset
1	Having an Ip Address	Features selected in both datasets
2	URL's Length	
3	tiny URL's	
4	Having "@" symbol	

5	Prefix suffix separation	
6	Sub-domains	
7	Protocol	
8	Domain registration length	
9	Links in tags	
10	Path	
11	Redirecting	
12	Age-domain	
13	Dns record	
14	Web-traffic	
15	Statistical report	
16	Page Rank	
17	External objects	
18	Anchor tags	
		Features selected only in the second dataset

B. Applied Algorithms

Random Forest: A machine learning classification technique that constructs many interdependent decision trees. The final output is determined by the tree with the highest votes.

Decision Tree: It is in the form of tree structure contains branches, root nodes, leaf nodes and internal nodes.

SVM: The plots n-dimensional data points on the scatter plot, n represents the total characteristics. To separate two groups, it constructs hyperplane.

Evaluation Metrics

For the evaluating the performance the main tool used is confusion matrix, which is further classified into four terms such as: True Positive (TP), False Positive (FP), True Negative (TN) and the False Negative (FN). The four elements are used to calculate the metrics, recall, precision, accuracy and F1 score that provides a comprehensive understanding in the model’s performance.

IV RESULTS AND DISCUSSION

After pre-processing the datasets, intelligent models were Intel Core CPUs and 8 GB of RAM were used to construct intelligent models after preprocessing the datasets. ML and Deep Learning (DL) models were developed using Python and Jupyter Notebook respectively. Each model was tested on a 70:30 split. The models were then assessed using a range of measures, including accuracy, recall, and precision. Comprehensive details on the parameter values for every category are given in Table IV.

TABLE IV Classifying the parameters

Model	Parameter	Value
Random Forest	Number of Trees	90
SVM	Kernel	Linear
Decision tree	Maximum number of iterations	90

Table V The following section presents the characteristics best in the feature selection method and the help of result’s

obtained by employing the features present in the primary dataset.

TABLE V Analyzing results for Dataset 1

Utilizing All Features				
Algorithm	Accuracy	Precision	Recall	F-Score
RF	95.56	96.4	94.9	95.65
SVM	90.95	91.27	90.45	91.17
DT	91.00	92.20	90.15	91.20
Utilizing Beast Features				
RF	99.98	99.98	99.98	99.98
SVM	98.08	97.91	98.30	98.11
DT	99.98	99.98	99.98	99.98

Table VI shows describes the performance of different classifiers (Decision Tree, Random Forest and Support Vector Machine) on a dataset using the best-selected features. The reported accuracy of 99.98% for both Support Vector Machine (SVM) and Random Forest(RF) is quite high and suggests that these models are performing exceptionally well on your dataset. Additionally, you mentioned the Decision Tree model achieving an accuracy of 98.11%, which is also a good performance but slightly lower than RF and SVM.

You also referred to recall and F-score, indicating that not only did these models achieve high accuracy, but they also performed well in terms of precision, recall, and a balance between recall and precision (F-score). This is important in classification tasks as it provides a more comprehensive evaluation of a model's performance beyond its accuracy.

TABLE VI Analyzing results for dataset 2

Utilizing All Features				
Algorithm	Accuracy	Precision	Recall	F-Score
RF	91.15	92.10	89.85	91.00
SVM	82.85	83.45	82.00	82.80
DT	85.75	89.30	81.32	85.19
Utilizing Beast Features				
RF	91.83	92.71	89.67	91.15
SVM	83.29	83.60	82.66	80.15
DT	85.24	89.26	81.57	85.25

The Random Forest (RF) achieved the highest accuracy at 91.83%, along with Support Vector Machine (SVM) following at 85.25%. Additionally, that RF had the greatest recall and F-score, and it also achieved the best accuracy at 92.71%. In contrast to the initial dataset, the performance of the model did not exhibit substantial improvement in the second dataset when compared to the selected features. Notably, the Support Vector Machine (SVM) demonstrated inferior performance in comparison to the Random Forest (RF), despite being considered closely related in all subsets. The observed variations in SVM performance could be attributed to either the dataset size or the dataset design. Furthermore, the previous statistics highlight that the feature selection strategy used to change the classification accuracy across the board. This underscores the significance of feature engineering methods in enhancing classifier efficacy.

Considering that reputable websites indexed by Google and with a longer existence are more likely to be legitimate, they serve as indicators for identifying potential phishing attempts. Two content-based features, namely URL of Request and Anchor URL, analyze inline Text and HTML meta tags, respectively, searching for signs of malicious links. This analysis is crucial in determining whether a site is a phishing site or not. Moreover, in this feature selection step, the four characteristics demonstrated highly significance for the primary dataset, which indicates their role in identifying phishing websites. However, when these are compared to the results present in Table VI, as indicated by the numbers in Table VII, there was no substantial improvement. This lack of improvement might be attributed to contextual factors and link IDs, two of these must be removed from the second sample of their poor correlation values.

TABLE VII Analyzing features in the two datasets

Utilizing All Features				
Algorithm	Accuracy	Precision	Recall	F-Score
RF	83.32	87.80	82.87	88.35
SVM	82.63	86.19	81.65	82.67
DT	88.25	89.76	86.37	89.30

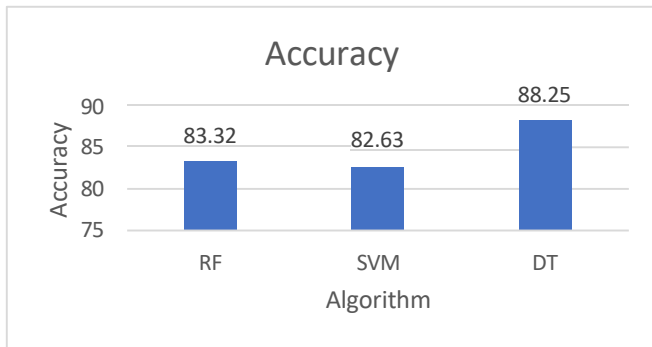


Fig.5 Accuracy

Fig.5 shows about the accuracy of the different algorithms. The decision tree performs 88.25% accuracy when compared with the other algorithms RF and SVM.

III. CONCLUSIONS AND FUTURE WORK

The surge in phishing websites and the escalating number of internet users underscore the imperative need for innovative methods to detect these malicious domains. many studies are involved in distinguishing these phishing websites from legitimate ones, leveraging extensive datasets and intelligent approaches. In this discussion, our primary focus has been on the development of features as we endeavored to create any modeling software and identify the factors influencing these systems' capability to recognize malicious URLs. We employed Support Vector Machine (SVM) and Random Forest(RF) for feature selection, aiming to enhance the classification accuracy of the deployed models. In both instances, RF emerged as the top-performing model. SVM and RF exhibited flawless accuracy in the primary dataset, attributed to the comprehensive study of original characteristics and associated aspects. However, the second dataset did not yield similar results, primarily because the initial dataset contained more associated traits absent in the second. We propose further research encompassing

additional dataset to the utilization of the deeper Deep Learning (DL) models to compare and improve performance. Identifying additional crucial factors in detecting phishing sites is another avenue for exploration. Future work could involve implementing this new approach into the web extension to bolster detection, providing an extra robust defense for users. This extension could offer an added layer of protection, enhancing user security while navigating the online landscape.

REFERENCES

- [1] B. Eshete, A. Villafiorita and K. Weldemariam, "Malicious Website Detection: Effectiveness and Efficiency Issues", 2011 First SysSec Workshop, 2011. Available: 10.1109/syssec.2011.9.
- [2] S. Kumi, C. Lim, and S.-G. Lee, "Malicious URL Detection Based on Associative Classification," *Entropy*, vol. 23, no. 2, p. 182, Jan. 2021, doi: 10.3390/e23020182.
- [3] Könighofer, B., Rudolf, J., Palmisano, A. et al. Online shielding for reinforcement learning. *Innovations Syst Softw Eng* **19**, 379–394 (2023).
- [4] Chow Y, Nachum O, Duenez-Guzman E, et al (2018) A Lyapunov-based approach to safe reinforcement learning. In: *NIPS*, pp 8103–8112
- [5] Kaissis, G.A.; Makowski, M.R.; Rückert, D.; Braren, R.F. Secure, privacy-preserving and federated machine learning in medical imaging. *Nat. Mach. Intell.* **2020**, *2*, 305–311.
- [6] Pan, Y.; Sun, F.; Teng, Z.; White, J.; Schmidt, D.C.; Staples, J.; Krause, L. Detecting web attacks with end-to-end deep learning. *J. Internet Serv. Appl.* **2019**, *10*, 1–22.
- [7] McIntosh, T.; Jang-Jaccard, J.; Watters, P.; Susnjak, T. The inadequacy of entropy-based ransomware detection. In *Proceedings of the International Conference on Neural Information Processing*, Sydney, Australia, 12–15 December 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 181–189.
- [8] M., Iyatomi H. Web application firewall using character-level convolutional neural network; *Proceedings of the 2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA)*; Penang, Malaysia. 9–10 March 2018; pp. 103–106.
- [9] Huang H.C., Zhang Z.K., Cheng H.W., Shieh S.W. Web application security: Threats, countermeasures, and pitfalls. *Computer*. 2017; **50:81–85**.
- [10] M. Tubyte and A. Paulauskaite-Taraseviciene, "Research on phishing email detection based on URL parameters using machine learning algorithms," *CEUR Workshop Proc.*, vol. 2915, pp. 18- 26, 2021.
- [11] Alaoui, R.L.; Nfaoui, E.H. Deep Learning for Vulnerability and Attack Detection on Web Applications: A Systematic Literature Review. *Future Internet* **2022**, *14*, 118.
- [12] Rusyaidi, M.; Jaf, S.; Ibrahim, Z. Detecting distributed denial of service in network traffic with deep learning. *Int. J. Adv. Comput. Sci. Appl.* **2022**, *13*, 34–41.
- [13] Dawadi, Babu R., Bibek Adhikari, and Devesh Kumar Srivastava. 2023. "Deep Learning Technique-Enabled Web Application Firewall for the Detection of Web Attacks" *Sensors* **23**, no. 4: 2073.
- [14] M. Chatterjee and A. -S. Namin, "Detecting Phishing Websites through Deep Reinforcement Learning," 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), Milwaukee, WI, USA, 2019, pp. 227-232, doi: 10.1109/COMPSAC.2019.102
- [15] Zhu, Tingshao & Ning, Yue & Li, Ang & Xu, Xinguo. (2011). Using decision tree to predict mental health status based on web behavior. *IEEE Symposium on Web Society*. 27-31. 10.1109/SWS.2011.610126.



JEPPIAAR
ENGINEERING COLLEGE

(A Christian Minority Institution)

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600119



IEEE
SB JEC



9th IEEE SPONSORED INTERNATIONAL CONFERENCE ON SCIENCE, TECHNOLOGY, ENGINEERING AND MATHEMATICS

ICONSTEM 2024

"The Role of Emerging Technologies in Digital Transformation"

CERTIFICATE OF PARTICIPATION

Paper ID: ICON24T1110

This is to certify that **Mallidi Venkata Sukumar Reddy**

has actively participated and presented a research paper in **ICONSTEM 2024** titled
" WEB SHIELD: Shielding your BrowsingExperience using Decision Tree "

in the 9th IEEE Sponsored International Conference on Science, Technology,
Engineering and Mathematics (**ICONSTEM 2024**) organized by Jeppiaar
Engineering College on 4th & 5th April, 2024.

Dr. J. Jebastine, M.E., Ph.D.
Vice - Principal
Convenor, ICONSTEM 2024
Jeppiaar Engineering College

Dr. K. Senthil Kumar, M.E., Ph.D., PE
Principal
Jeppiaar Engineering College

Dr. Shaleesha A. Stanley, M.Tech, Ph.D., D.Sc.
Dean - Academics
Jeppiaar Engineering College

Dr. M. Regeena J Murali, B.Tech, M.B.A., Ph.D.
Founder & Chancellor, Jeppiaar University
Chairman and Managing Director
Jeppiaar Group of Institutions