MINI PROJECT REPORT

On

DETECTION OF PHISHING WEBSITES (Machine Learning)

Submitted by

AYUSH DWIVEDI
171500070
RISHABH TRIPATHI
171500259
ANIPRIYA RAI
171500057

Department of Computer Engineering & Applications

Institute of Engineering & Technology

GLA University
Mathura- 281406, INDIA

2019

Department of computer Engineering and Applications
GLA University, Mathura
17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,
Mathura – 281406

Declaration

I hereby declare that the work which is being presented in the Mini Project "detection of phishing websites using machine learning**",** in partial fulfillment of the requirements for Summer Training viva voce, is an authentic record of my own work carried under the supervision of "Vivek Kumar Sir" **.**

AYUSH DWIVEDI

171500070

RISHABH TRIPATHI

171500259

ANUPRIYA RAI

171500057

Department of computer Engineering and Applications
GLA University, Mathura
17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,
Mathura – 281406

ABSTRACT

Phishing costs Internet users billions of dollars per year. It refers to luring techniques used by identity thieves to fish for personal information in a pond of unsuspecting internet users. Phishers use spoofed e-mail, phishing software to steal personal information and financial account details such as usernames and passwords. This paper deals with methods for detecting phishing web sites by analyzing various features of benign and phishing URLs by Machine learning techniques. We discuss the methods used for detection of phishing websites based on lexical features, host properties and page importance properties. We consider various data mining algorithms for evaluation of the features in order to get a better understanding of the structure of URLs that spread phishing. The fine-tuned parameters are useful in selecting the apt machine learning algorithm for separating the phishing sites from benign sites.

Department of computer Engineering and Applications
GLA University, Mathura
17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,
Mathura – 281406

CERTIFICATE

It is certified that the work contained in the synopsis entitled "Detection of phishing websites", by AYUSH DWIVEDI, RISHABH TRIPATHI, ANUPRIYA RAI for the mini project, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Mr. Vivek Kumar

Technical Trainer

GLA University, Mathura

Department of computer Engineering and Applications
GLA University, Mathura
17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,
Mathura – 281406

ACKNOWLEDGEMENT

It is our pleasure to acknowledgement the assistance of a number of people without whose help this would not have been possible. First and foremost, we would like to express our gratitude to Mr. VIVEK KUMAR SIR our project guides, for providing invaluable encouragement, guidance and assistance. After doing this project we can confidently say that this experience has not only enriched us with technical knowledge but also has unparsed the maturity of thought and vision. The attributes required being a successful professional.

AYUSH DWIVEDI

ANUPRIYA RAI

RISHABH TRIPATHI

Contents

1. INTRODUCTION

1.1 General Introduction to the topic

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.

Phishing is a criminal mechanism employing both social engineering and technical tricks to steal consumers' personal identity data and financial account credentials. Social engineering schemes use spoofed e-mails, purporting to be from legitimate business and agencies, designed to lead consumers to counterfeit websites that trick recipients into divulging financial data such as username and passwords. Technical subterfuge schemes install malicious software onto computers, to steal credentials directly, often using systems to intercept consumers' online account user names and passwords. The criminals, who want to obtain sensitive data, first create unauthorized replicas of a real website and e-mail, usually from a financial institution or another company that deals with financial information. The e-mail will be created using logos and slogans of a legitimate company. The nature and format of Hypertext Mark-up Language makes it very easy to copy images or even an entire website. While this ease of website creation is one of the reasons that the Internet has grown so rapidly as medium, it also permits the abuse of trademarks, trade names, and other corporate identifiers upon which consumers have come to rely as mechanisms for authentication. Phisher then send the "spoofed" e-mails to as many people as possible in an attempt to lure them in to the scheme. When these e-mails are opened or when a link in the mail is clicked, the consumers are redirected to a spoofed website, appearing to be from the legitimate entity.

1.2 Objective: The primary objectives behind phishing attacks, from an attacker's perspective, are Financial Gain phishers can use stolen banking credentials to their financial benefits Identity Hiding instead of using stolen identities directly, phishers might sell the identities to others whom might be criminals seeking ways to hide their identities and activities Fame and Notoriety phishers might attack victims for the sake of peer recognition

What is phishing?

Fraudsters send fake emails or set up fake web sites that mimic Yahoo!'s sign-in pages (or the sign-in pages of other trusted companies, such as eBay or PayPal) to trick you into disclosing your user name and password. This practice is sometimes referred to as "phishing" — a play on the word "fishing" — because the fraudster is fishing for your private account information. Typically, fraudsters try to trick you into providing your user name and password so that they can gain access to an online account. Once they gain access, they can use your personal information to commit identity theft, charge your credit cards, empty your bank accounts, read your email, and lock you out of your online account by changing your password.

If you receive an email (or instant message) from someone you don't know directing you to sign in to a website, be careful! You may have received a phishing email with links to a phishing website. A phishing website (sometimes called a "spoofed" site) tries to steal your account password or other confidential information by tricking you into believing you're on a legitimate website. You could even land on a phishing site by mistyping a URL (web address).

Is that website legitimate? Don't be fooled by a site that looks real. It's easy for phishers to create websites that look like the genuine article, complete with the logo and other graphics of a trusted website.
Important: If you're at all unsure about a website, do not sign in. The safest thing to do is to close and then reopen your browser, and then type the URL into your browser's URL bar. Typing the correct URL is the best way to be sure you're not redirected to a spoofed site.

2 SOFTWARE REQUIREMENT ANALYSIS:

2.1 PROBLEM DEINITION:

Phishing is a fraudulent attempt, usually made through email, to steal your personal information. We define a phishing page as any web page that, without permission, alleges to act on behalf of a third party with the intention of confusing viewers into performing an action with which the viewer would only trust a true agent of the third party.

Phishing is a type of computer attack that communicates socially engineered messages to humans via electronic communication channels in order to persuade them to perform certain actions for the attacker's benefit.

2.2 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.

2.3. Address Bar based Features

2.3.1 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".

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

2.3.2 Long URL to Hide the Suspicious Part

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.

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

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

2.3.3 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".

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

2.3.4 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.

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

2.3.5 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.

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

2.3.6 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/.

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

2.3.7 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.

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

2.3.8 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.

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

2.3.9 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.

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

2.3.10 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.

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

2.3.11 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.

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

Table 1 Common ports to be checked

PORT	Service	Meaning	Preferred Status
21	FTP	Transfer files from one host to another	Close
22	SSH	Secure File Transfer Protocol	Close
23	Telnet	provide a bidirectional 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

2.3.12 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/.

$$\text{Rule: IF} \begin{cases} \text{Using HTTP Token in Domain Part of The URL} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

2.4. Abnormal Based Features

2.4.1 Request URL

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.

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

2.4.2 URL of Anchor

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

1. If the <a> tags and the website have different domain names. This is similar to request URL feature.
2. If the anchor does not link to any webpage, e.g.:
   A. <a href= "#">
   B. <a href= "#content">
   C. <a href= "#skip">
   D. <a href= "JavaScript: void (0)">

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

2.4.3 Links in <Meta>, <Script> and <Link> tags

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

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

2.4.4 Server Form Handler (SFH)

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.

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

2.4.5 Submitting Information to Email

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.

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

2.4.6 Abnormal URL

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

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

2.5. HTML and JavaScript based Features

2.5.1 Website Forwarding

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.

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

2.5.2 Status Bar Customization

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.

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

2.5.3 Disabling Right Click

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.

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

2.5.4 Using Pop-up Window

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.

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

2.5.5 IFrame Redirection

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.

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

2.6. Domain based Features

2.6.1 Age of Domain

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.

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

2.6.2 DNS Record

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”.

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

2.6.3 Website Traffic

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”.

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

2.6.4 PageRank

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”.

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

2.6.5 Google Index

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.

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

2.6.6 Number of Links Pointing to Page

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 pointing to them.

Rule:

IF

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

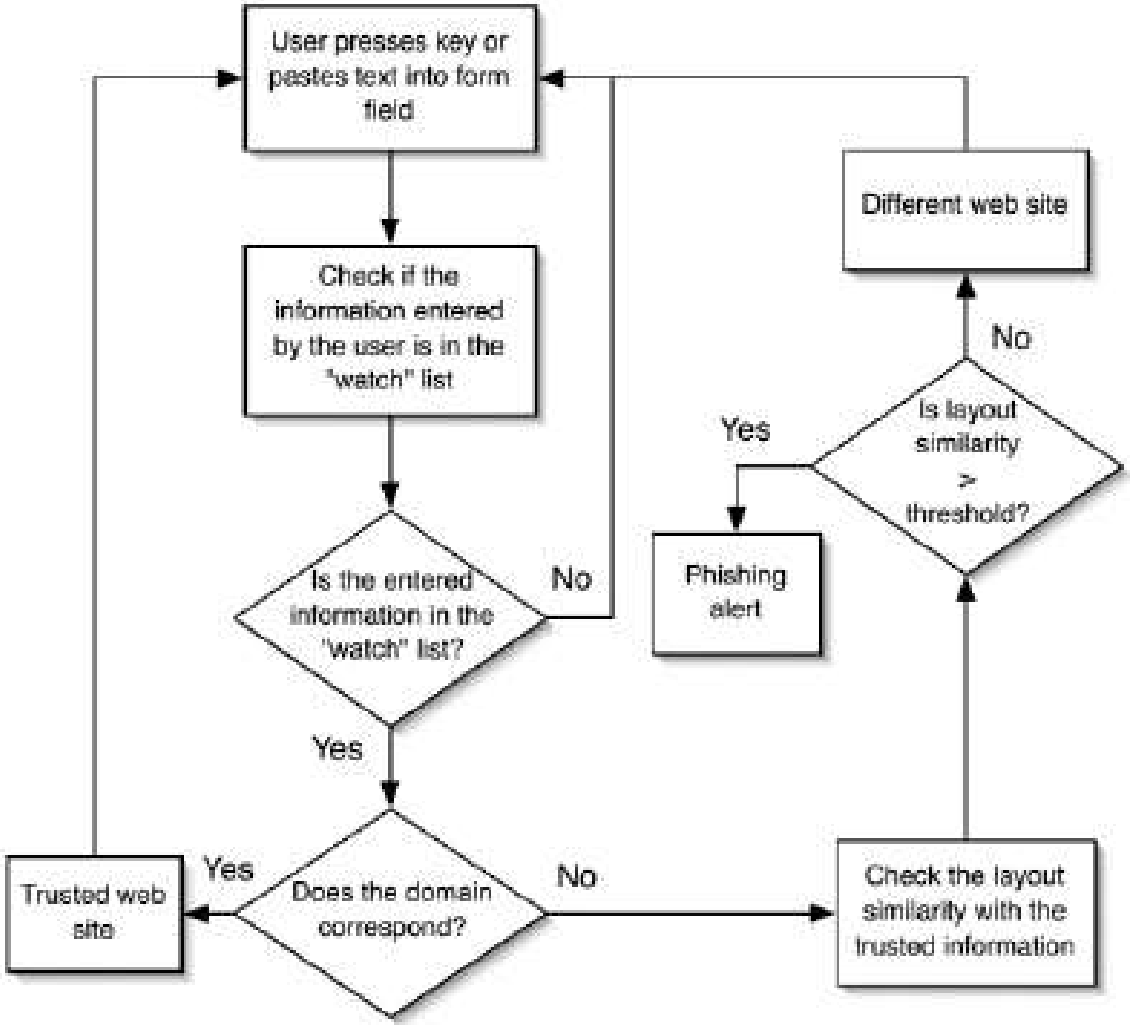
2.6.7 Statistical-Reports Based Feature

Several parties such as Phish Tank (PhishTank Stats, 2010-2012), and Stop adware (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 Phish Tank: “Top 10 Domains” and “Top 10 IPs” according to statistical-reports published in the last three years, starting in January2010 to November 2012. Whereas for “Stop adware”, we used “Top 50” IP addresses.

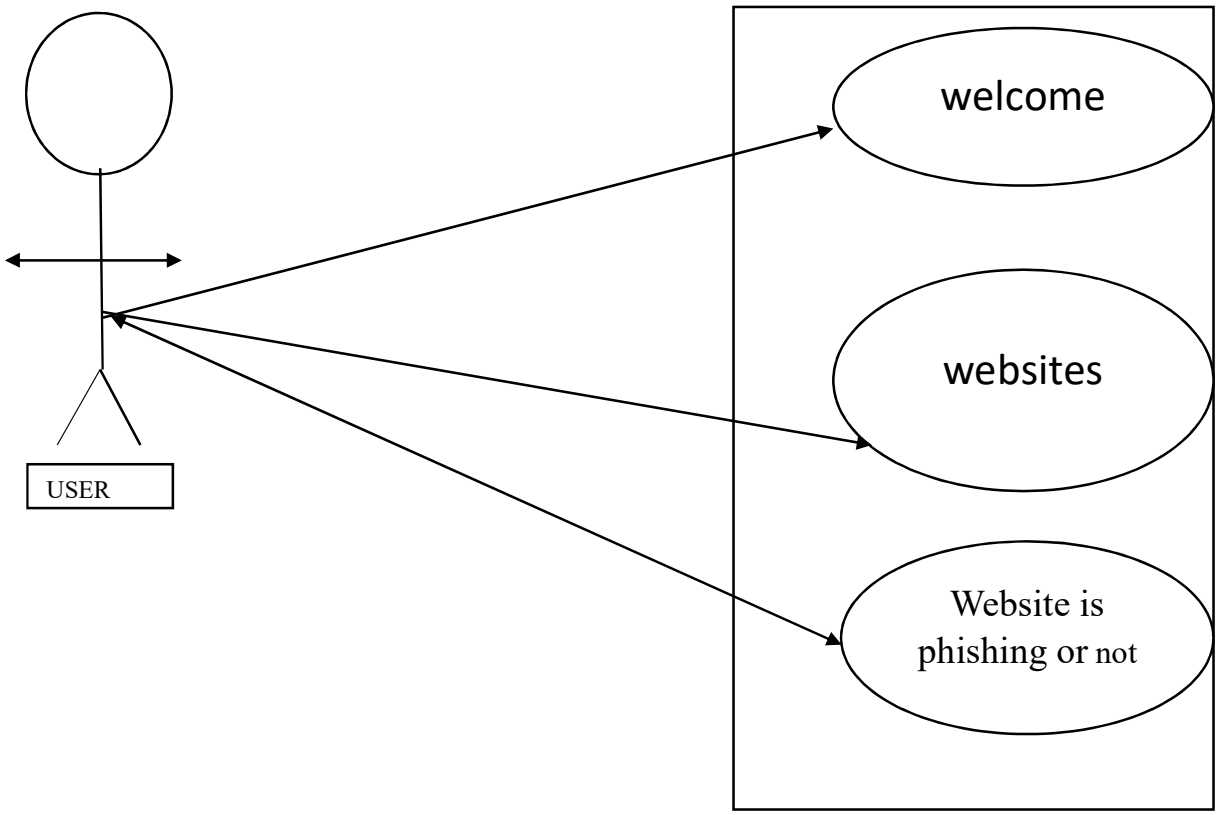
Rule: IF $\begin{cases} \text{Host Belongs to Top Phishing IPs or Top Phishing Domains} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

3 SOFTWARE DESIGN-

3.1. DATA FLOW DIAGRAM:



3.2 UML



4 TESTING:

Testing is a process of executing a program with the aim of finding error. To make our software perform well it should be error free. If testing is done successfully it will remove all the errors from the software.

4.1 Principles of Testing: -

(i) All the test should meet the customer requirements

(ii) To make our software testing should be performed by third party

(iii) Exhaustive testing is not possible.As we need the optimal amount of testing based on the risk assessment of the application.

(iv) All the test to be conducted should be planned before implementing it

(v) It follows pareto rule(80/20 rule) which states that 80% of errors comes from 20% of program components.

(vi) Start testing with small parts and extend it to large parts.

4.2.1 White Box Testing

It is defined as the testing of a software solution's internal structure, design, and coding. In this type of testing, the code is visible to the tester. It focuses primarily on verifying the flow of inputs and outputs through the application, improving design and usability, strengthening security.

4.2.2 Black Box Testing

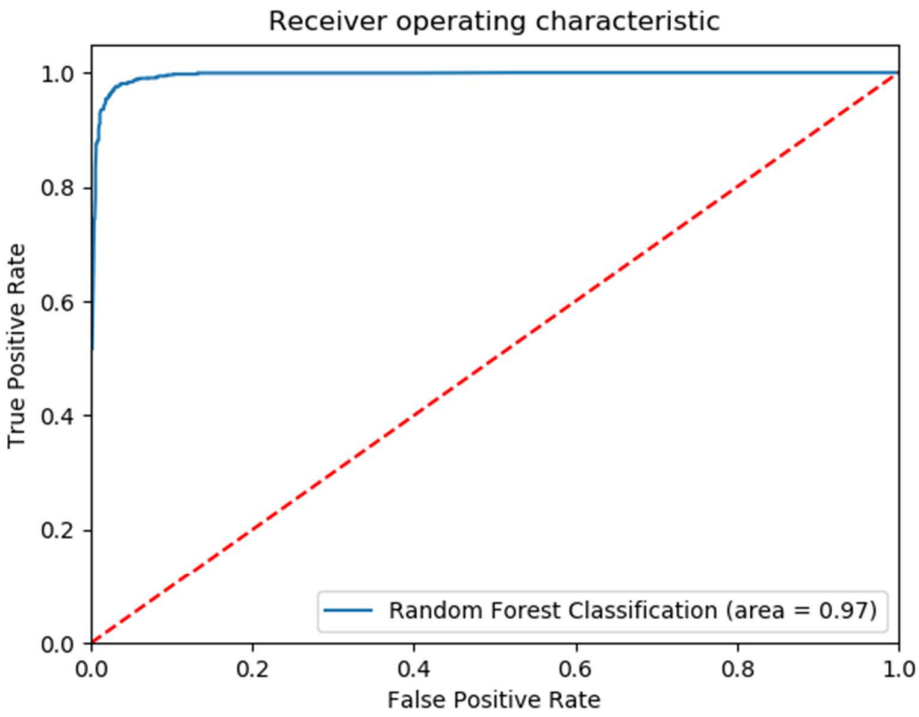
Black box testing is a type of software testing in which the functionality of the software is not known. The testing is done without the internal knowledge of the products.

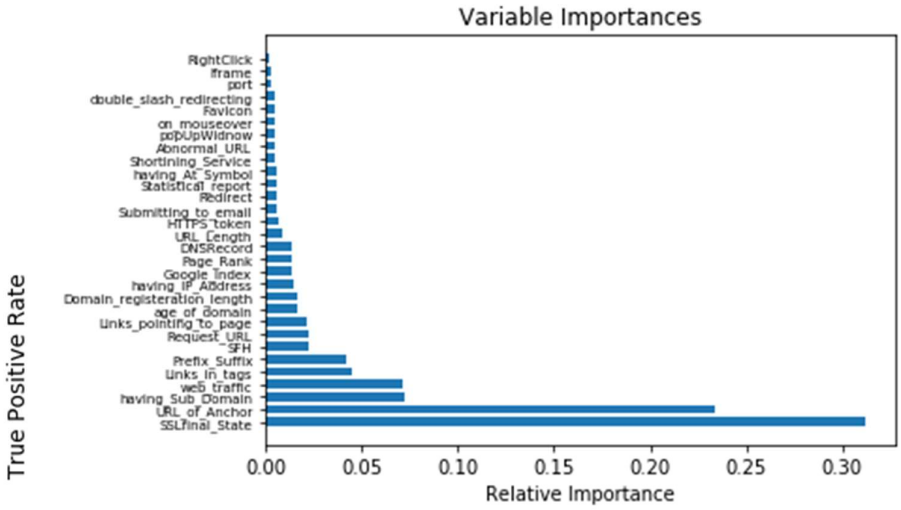

Phishing Website Detection
Write The Website
Predict


Phishing Website Detection
Write The Website
https://www.youtube.com/
Predict
COOL! You can excess the website
https://www.youtube.com/


Phishing Website Detection
Write The Website
http://appempresarial.net/
Predict
Warning The website is harmfull for you
WARNING

GRAPH: -


Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Random Forest Classification (area = 0.97)


Variable Importances
True Positive Rate
Relative Importance
RightClick
iframe
port
double_slash_redirecting
Favicon
on_mouseover
popUpWidnow
Abnormal_URL
Shortining_Service
having_At_Symbol
Statistical_report
Redirect
Submitting_to_email
HTTPS_token
URL_Length
DNSRecord
Page_Rank
Google_Index
having_IP_Address
Domain_registeration_length
age_of_domain
Links_pointing_to_page
Request_URL
SFH
Prefix_Suffix
Links_in_tags
web_traffic
having_Sub_Domain
URL_of_Anchor
SSLfinal_State
0.00
0.05
0.10
0.15
0.20
0.25
0.30

5 Implementation details

5.1 GUI CODE: -

```
from tkinter import *
import PIL
import sqlite3 as sql
import run
from PIL import ImageTk, Image
from tkinter import filedialog
import os

scr=Tk()
scr.geometry('1200x800+0+0')
def retrieve_input():
    global us
    input1=us.get()
    print(input1)
    out=run.run(input1)
    print(out)
    if(out=="Warning The website is harmfull for you"):
        v1 = StringVar()
        v1.set(out)
        photo = r"C:\Users\admin\Downloads\warning.png"
        #print(v1)
        openfn(photo)

        l2 = Label(f, text='Phishing / Not Phishing', textvariable=v1, font=('times',
20, 'bold'), bg='white', fg='red')
        l2.place(x=500, y=300)
        #print('done')
    else:
        v1 = StringVar()
        v1.set(out)
        photo = r"C:\Users\admin\Downloads\ok_image.png"
        openfn(photo)

        # print(v1)
        l2 = Label(f, text='Phishing / Not Phishing', textvariable=v1, font=('times',
20, 'bold'), bg='white',fg='red')
        l2.place(x=500, y=300)
        #print('done else')



scr.geometry("550x300+300+150")
scr.resizable(width=True, height=True)
```

```
def openfn(y):
    filename = y
    open_img(filename)
def open_img(x):
    img = Image.open(x)
    img = img.resize((250, 250), Image.ANTIALIAS)
    img = ImageTk.PhotoImage(img)
    panel = Label(scr, image=img)
    panel.image = img
    panel.pack()

#btn = Button(root, text='open image', command=open_img).pack()


l=Label(scr,text="Phishing Website
Detection",font=('times',20,'bold'),bg='blue',fg='yellow')
l.pack(side=TOP,fill=X)
f=Frame(scr,bg='orange')
print(dir(Frame))
f.pack(fill=BOTH,expand=12)
v=StringVar()
#v1=StringVar()
#v1.set('lol')
l1=Label(f,text='   Write The Website
',font=('times',20,'bold'),bg='white',fg='black')
l1.place(x=300,y=100)
#l2=Label(f,text='Phishing / Not
Phishing',textvariable=v1,font=('times',20,'bold'),bg='red',fg='yellow')
#l2.place(x=400,y=300)
us=Entry(f,font=('times',20,'bold'),bg='white',fg='black',textvariable=v,widt
h=50)
us.place(x=600,y=100)
#ps=Entry(f,font=('times',20,'bold'),bg='blue',fg='yellow',textvariable=v1)
#ps.place(x=500,y=300)
b1=Button(f,text='Predict',font=('times',20,'bold'),command=retrieve_input)
b1.place(x=700,y=200)
scr.mainloop()
```

5.2 CODE OF RUN.PY:

```
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import GridSearchCV
import numpy as np
import matplotlib.pyplot as plt

dataset = pd.read_csv("phishcoop.csv")
dataset = dataset.drop('id', 1) #removing unwanted column

x = dataset.iloc[ : , :-1].values
y = dataset.iloc[:, -1:].values

x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.25, random_state =0 )
parameters = [{'n_estimators': [100, 700],
    'max_features': ['sqrt', 'log2'],
    'criterion' :['gini', 'entropy']}]

grid_search = GridSearchCV(RandomForestClassifier(),   parameters,cv =5, n_jobs=
-1)
grid_search.fit(x_train, y_train)
#printing best parameters
print("Best Accurancy =" +str( grid_search.best_score_))
print("best parameters =" + str(grid_search.best_params_))
classifier = RandomForestClassifier(n_estimators = 100, criterion = "gini",
max_features = 'log2',   random_state = 0)
classifier.fit(x_train, y_train)

#predicting the tests set result
y_pred = classifier.predict(x_test)

# In[7]:


import Datasetgen as data


# In[8]:



def run(url):

    try:
        #checking and predicting
        checkprediction = data.main(url)
```

```
        #checkprediction=[[0, -1, -1, -1, 1, 1, -1, 1, 0, 0, -1, -1, 1, 1, 0, 0, -1, 0, -1, -1, -1,
-1, 1, 0, 0, 1, 1, 1, -1, -1]]
        prediction = classifier.predict(checkprediction)
        if prediction==1:
            return "Warning The website is harmfull for you"
         else:
            return "COOL! You can excess the website" +"\n"+ url

    except:
        return "The website you entered does not exist"
```

5.3 CODE OF DATAGEN.PY:-

```
import requests
import datetime
import re
import socket
import ssl
import urllib.request

import regex
import whois
from bs4 import BeautifulSoup
from tldextract import extract


def generate_data_set(url):
    data_set = []

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

    # Stores the response of the given URL
    try:
        response = requests.get(url)
    except:

        response = ""

    # Extracts domain from the given URL
    domain = re.findall(r"://([^/]+)/?", url)[0]
```

```
    # Requests all the information about the domain
    whois_response = requests.get("https://www.whois.com/whois/" + domain)

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

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

    def url_having_ip(url):
        # using regular function
        symbol = regex.findall(r'(http((s)?)://)((((\d)+).)*)((\w)+)(/((\w)+))?', url)
        if (len(symbol) != 0):
            having_ip = 1    # phishing
        else:
            having_ip = -1    # legitimate
            return (having_ip)
        return 0

    def url_length(url):
        length = len(url)
        if (length < 54):
            return -1
        elif (54 <= length <= 75):
            return 0
        else:
            return 1

    def url_short(url):
        if re.findall("goo.gl|bit.ly", url):
            return 1
        else:
            return -1

    def having_at_symbol(url):
        symbol = regex.findall(r'@', url)
        if (len(symbol) == 0):
            return -1
        else:
            return 1

    def doubleSlash(url):
        if re.findall(r"[^https?:]//", url):
            return -1
```

```
        else:
            return 1

    def prefix_suffix(url):
        subDomain, domain, suffix = extract(url)
        if (domain.count('-')):
            return 1
        else:
            return -1

    def sub_domain(url):
        subDomain, domain, suffix = extract(url)
        if (subDomain.count('.') == 0):
            return -1
        elif (subDomain.count('.') == 1):
            return 0
        else:
            return 1

    def SSLfinal_State(url):
        try:
            # check wheather contains https
            if (regex.search('^https', url)):
                usehttps = 1
            else:
                usehttps = 0
            # getting the certificate issuer to later compare with trusted issuer
            # getting host name
            subDomain, domain, suffix = extract(url)
            host_name = domain + "." + suffix
            context = ssl.create_default_context()
            sct = context.wrap_socket(socket.socket(), server_hostname=host_name)
            sct.connect((host_name, 443))
            certificate = sct.getpeercert()
            issuer = dict(x[0] for x in certificate['issuer'])
            certificate_Auth = str(issuer['commonName'])
            certificate_Auth = certificate_Auth.split()
            if (certificate_Auth[0] == "Network" or certificate_Auth ==
"Deutsche"):
                certificate_Auth = certificate_Auth[0] + " " + certificate_Auth[1]
            else:
                certificate_Auth = certificate_Auth[0]
            trusted_Auth = ['Comodo', 'Symantec', 'GoDaddy', 'GlobalSign',
'DigiCert', 'StartCom', 'Entrust', 'Verizon',
                            'Trustwave', 'Unizeto', 'Buypass', 'QuoVadis',
'Deutsche Telekom', 'Network Solutions',
                            'SwissSign', 'IdenTrust', 'Secom', 'TWCA', 'GeoTrust',
'Thawte', 'Doster', 'VeriSign']
            # getting age of certificate
            startingDate = str(certificate['notBefore'])
```

```
            endingDate = str(certificate['notAfter'])
            startingYear = int(startingDate.split()[3])
            endingYear = int(endingDate.split()[3])
            Age_of_certificate = endingYear - startingYear

            # checking final conditions
            if ((usehttps == 1) and (certificate_Auth in trusted_Auth) and
(Age_of_certificate >= 1)):
                return -1   # legitimate
            elif ((usehttps == 1) and (certificate_Auth not in trusted_Auth)):
                return 0   # suspicious
            else:
                return 1   # phishing

        except Exception as e:

            return 1

    def domain_registration(url):
        try:
            w = whois.whois(url)
            updated = w.updated_date
            exp = w.expiration_date
            length = (exp[0] - updated[0]).days
            if (length <= 365):
                return 1
            else:
                return -1
        except:
            return 0

    def favicon(url):
        # ongoing
        return 0

    def port(url):
        try:
            port = domain.split(":")[1]
            if port:
                return 1
            else:
                return -1
        except:
            return -1

    def https_token(url):
        subDomain, domain, suffix = extract(url)
        host = subDomain + '.' + domain + '.' + suffix
        if (host.count('https')):   # attacker can trick by putting https in domain part
            return 1
```

```
        else:
            return -1

    def request_url(url):
        try:
            subDomain, domain, suffix = extract(url)
            websiteDomain = domain

            opener = urllib.request.urlopen(url).read()
            soup = BeautifulSoup(opener, 'lxml')
            imgs = soup.findAll('img', src=True)
            total = len(imgs)

            linked_to_same = 0
            avg = 0
            for image in imgs:
                subDomain, domain, suffix = extract(image['src'])
                imageDomain = domain
                if (websiteDomain == imageDomain or imageDomain == ''):
                    linked_to_same = linked_to_same + 1
            vids = soup.findAll('video', src=True)
            total = total + len(vids)

            for video in vids:
                subDomain, domain, suffix = extract(video['src'])
                vidDomain = domain
                if (websiteDomain == vidDomain or vidDomain == ''):
                    linked_to_same = linked_to_same + 1
            linked_outside = total - linked_to_same
            if (total != 0):
                avg = linked_outside / total

            if (avg < 0.22):
                return -1
            elif (0.22 <= avg <= 0.61):
                return 0
            else:
                return 1
        except:
            return 0

    def url_of_anchor(url):
        try:
            subDomain, domain, suffix = extract(url)
            websiteDomain = domain

            opener = urllib.request.urlopen(url).read()
            soup = BeautifulSoup(opener, 'lxml')
            anchors = soup.findAll('a', href=True)
            total = len(anchors)
```

```
            linked_to_same = 0
            avg = 0
            for anchor in anchors:
                subDomain, domain, suffix = extract(anchor['href'])
                anchorDomain = domain
                if (websiteDomain == anchorDomain or anchorDomain == ''):
                    linked_to_same = linked_to_same + 1
            linked_outside = total - linked_to_same
            if (total != 0):
                avg = linked_outside / total

            if (avg < 0.31):
                return -1
            elif (0.31 <= avg <= 0.67):
                return 0
            else:
                return 1
        except:
            return 0

    def Links_in_tags(url):
        try:
            opener = urllib.request.urlopen(url).read()
            soup = BeautifulSoup(opener, 'lxml')

            no_of_meta = 0
            no_of_link = 0
            no_of_script = 0
            anchors = 0
            avg = 0
            for meta in soup.find_all('meta'):
                no_of_meta = no_of_meta + 1
            for link in soup.find_all('link'):
                no_of_link = no_of_link + 1
            for script in soup.find_all('script'):
                no_of_script = no_of_script + 1
            for anchor in soup.find_all('a'):
                anchors = anchors + 1
            total = no_of_meta + no_of_link + no_of_script + anchors
            tags = no_of_meta + no_of_link + no_of_script
            if (total != 0):
                avg = tags / total

            if (avg < 0.25):
                return -1
            elif (0.25 <= avg <= 0.81):
                return 0
            else:
                return 1
        except:
```

```
            return 0

    def sfh(url):
        # ongoing
        return 0

    def email_submit(url):
        try:
            opener = urllib.request.urlopen(url).read()
            soup = BeautifulSoup(opener, 'lxml')
            if (soup.find('mailto:')):
                return 1
            else:
                return -1
        except:
            return 0

    def abnormal_url(url):
        return 0;

    def redirect(url):
        if len(response.history) <= 1:
            return -1
        elif len(response.history) <= 4:
            return 0
        else:
            return 1

    def on_mouseover(url):
        if re.findall("<script>.+onmouseover.+</script>", response.text):
            return 1
        else:
            return -1

    def popup(url):
        if re.findall(r"alert\(", response.text):
            return 1
        else:
            return -1

    def iframe(url):
        if re.findall(r"[<iframe>|<frameBorder>]", response.text):
            return 1
        else:
            return -1

    def age_of_domain(url):
        try:
            w = whois.whois(url)
            start_date = w.creation_date
```

```
            current_date = datetime.datetime.now()
            age = (current_date - start_date[0]).days
            if (age >= 180):
                return -1
            else:
                return 1
        except Exception as e:
            print(e)
            return 0

    def dns(url):
        # ongoing
        return 0

    def web_traffic(url):
        try:
            if global_rank > 0 and global_rank < 100000:
                return -1
            else:
                return 1
        except:
            return 1

    def page_rank(url):
        try:
            if global_rank > 0 and global_rank < 100000:
                return -1
            else:
                return 1
        except:
            return 1

    def google_index(url):
        try:
            if global_rank > 0 and global_rank < 100000:
                return -1
            else:
                return 1
        except:
            return 1

    def links_pointing(url):
        number_of_links = len(re.findall(r"<a href=", response.text))
        if number_of_links == 0:
            return 1
        elif number_of_links <= 2:
            return 0
        else:
            return -1
```

```
    def rightClick(url):
        if re.findall(r"event.button ?== ?2", response.text):
            return 1
        else:
            return -1

    def statistical(url):
        # ongoing
        return -1

    check = [[url_having_ip(url), url_length(url), url_short(url),
having_at_symbol(url),
                doubleSlash(url), prefix_suffix(url), sub_domain(url),
SSLfinal_State(url),
                domain_registration(url), favicon(url), port(url), https_token(url),
request_url(url),
                url_of_anchor(url), Links_in_tags(url), sfh(url), email_submit(url),
abnormal_url(url),
                redirect(url), on_mouseover(url), rightClick(url), popup(url), iframe(url),
                age_of_domain(url), dns(url), web_traffic(url), page_rank(url),
google_index(url),
                links_pointing(url), statistical(url)]]
    return check
def main(url):
    return generate_data_set(url)
```

6 Conclusion

In this work, we implemented four classifiers using MATLAB scripts, which are the decision tree, Naïve Bayes' classifier, Support Vector Machine (SVM), and the Neural Network. The classifiers were used to detect phishing URLs. In detecting phishing URLs, there are two steps. The first step is to extract features from the URLs, and the second step is to classify URLs using the model that has been developed with the help of the training set data. In this work, we used the data set that provided the extracted features. The data set, from The University of California, Irvine Machine Learning Repository, contained nine features. One of the main concerns in the decision tree classifiers is over fitting. Generally, the decision tree classifies the training set data very well but yields poor results with a testing dataset. It is often required to prune the decision tree to work well with testing data. The pruned decision tree provided the highest classification accuracy 90.39 percent. with more features in the data set it may be possible to obtain higher accuracy. In addition, the accuracy may be improved by using an ensemble of trees. It can be seen from Table 2 that the neural network classifier yielded the lowest accuracy for this data set. One of the reasons that it did not perform well is that feature vector values were discrete which results in non-smooth decision boundaries that separate the three classes. However, it is possible to use more number of units in the hidden layer or use deep learning techniques such as adding a number of hidden layers to improve the performance of the network.

6.1 Future Prospects: This system can be used by many E-commerce or other websites in order to have good customer relationship. User can make online payment securely. Data mining algorithm used in this system provides better performance as compared to other traditional classifications algorithms. With the help of this system user can also purchase products online without any hesitation.

REFERENCE/BIBLIOGRAPHY

DATASET: - https://archive.ics.uci.edu/ml/machine-learning-databases/00327/

Other references are: -

- github.com
- Manohar Swamynathan (auth.) - Mastering Machine Learning with Python in Six Steps_ A Practical Implementation Guide to Predictive Data Analytics Using Python (2017, Après)
- Google collaborates.
- www.kaggale.com
- https://www.phishtank.com/phish_search.php?page=1&active=y&verified=u

7 APPENDICES

CODE: - LOGESTIC REGRESSION CODE:-

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import
train_test_split
from sklearn.metrics import confusion_matrix


# In[2]:


dataset = pd.read_csv("phishcoop.csv")
dataset = dataset.drop('id', 1) #removing unwanted
column

x = dataset.iloc[ : , :-1].values
y = dataset.iloc[:, -1:].values


# In[4]:


x_train, x_test, y_train, y_test =
train_test_split(x,y,test_size = 0.25, random_state
=0 )

#fitting logistic regression
classifier = LogisticRegression(random_state = 0)
classifier.fit(x_train, y_train)

#predicting the tests set result
y_pred = classifier.predict(x_test)
```

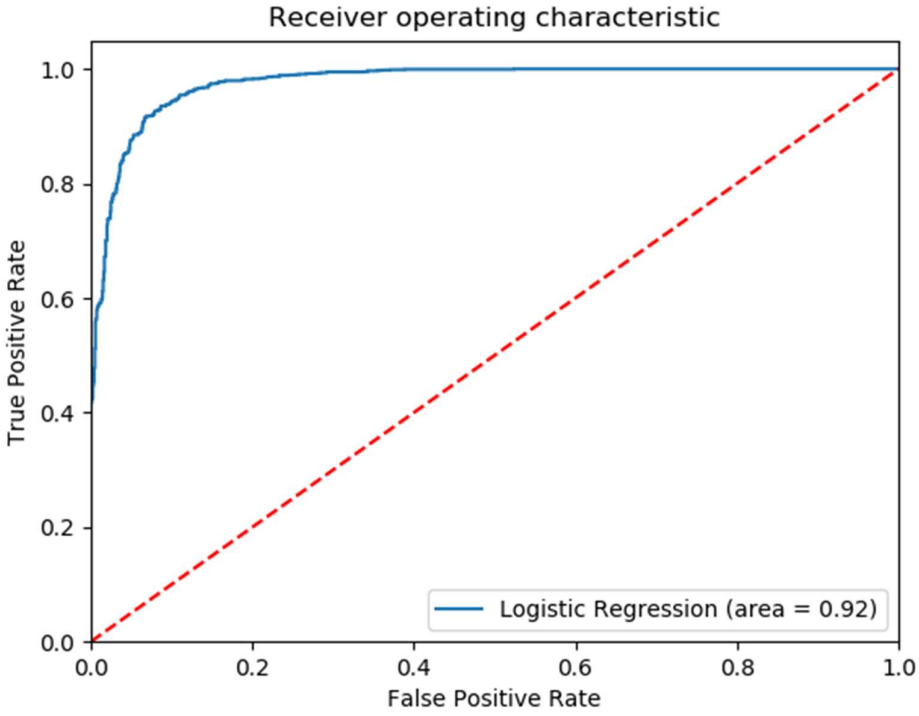
```
# In[6]:



cm = confusion_matrix(y_test, y_pred)
print(cm)
score = classifier.score(x_test, y_test)
print(score*100,"%")

from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test,
classifier.predict(x_test))
fpr, tpr, thresholds = roc_curve(y_test,
classifier.predict_proba(x_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area
= %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```

GRAPH: -


Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Logistic Regression (area = 0.92)