OFS-NN: An Effective Phishing Websites Detection Model Based on Optimal Feature Selection and Neural Network

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

M.Sc COMPUTER SCIENCE

SOUMEN DAS

Reg.No: PCS051822

UNDER THE GUIDANCE OF

Dr. MANOHAR NAIK S.

DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF PHYSICAL SCIENCES, CENTRAL UNIVERSITY OF KERALA, TEJASWINI HILLS PERIYE, KASARAGOD - 671316, KERALA, INDIA APRIL 2018

CENTRAL UNIVERSITY OF KERALA

CERTIFICATE

This is to certify that the thesis entitled, **"OFS-NN: An Effective Phishing Websites Detection Model Based on Optimal Feature Selection and Neural Network"** submitted by **Soumen Das (REG. NO: PCS051822)** in partial fulfillment of the requirements for the award of M.Sc in Computer Science at the Central University of Kerala, is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the report has not been submitted to any other University Institute for the award of any degree.

DATE :

Dr. MANOHAR NAIK S
Assistant Professor
Department of Computer Science
Central University of Kerala
Kasaragod, Kerala - 671316

डॉ. मनोहर नायक. एस / Dr. Manohar Naik S
सहायक आचार्य / Assistant Professor
कंप्यूटर विज्ञान विभाग / Department of Computer Science
भौतिक विज्ञान स्कूल / School of Physical Sciences
केरल केंद्रीय विश्वविद्यालय / Central University of Kerala
तेजस्विनी हिल्स, पेरिया / Tejaswini Hills, Periya P.O
कासरगोड / Kasaragod-671316

CENTRAL UNIVERSITY OF KERALA

CERTIFICATE

This is to certify that the thesis entitled, **"OFS-NN: An Effective Phishing Websites Detection Model Based on Optimal Feature Selection and Neural Network"** is a bonafide work carried out by **Soumen Das (REG. NO: PCS051822)** in partial fulfillment of the requirements for the award of M.Sc in Computer Science at the Central University of Kerala, Kasaragod, during the academic year 2018-2020.

The work is satisfactory to award Master's Degree in Computer Science.

DATE :

Dr. RAJESH R
Head Of the Department
Associate Professor
Department of Computer Science
Central University of Kerala
Kasaragod, Kerala - 671316

EXTERNAL EXAMINER:

DECLARATION

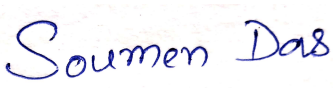
I, SOUMEN DAS, Reg No: PCS051822, student of Fourth Semester M.Sc Computer Science, Central University of Kerala, do hereby declare that the report entitled, **"OFS-NN: An Effective Phishing Websites Detection Model Based on Optimal Feature Selection and Neural Network"**, submitted to the Department of Computer Science is an original record of studies and bona fide work carried out by me from December 2018 to April 2020.

DATE:

PLACE:

SOUMEN DAS

PCS051822



ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success.

I would like to show my greatest appreciation to my guide Dr. MANOHAR NAIK S, Department of Computer science,Central university of kerala . I cannot say thank you enough for her tremendous support and help.I feel motivated and encouraged every time I attend her meeting. Without her encouragement and guidance this project would not have materialized.

I am also grateful to my Head of the Department, Dr. RAJESH R, for providing excellent computing facility and nice atmosphere for completing my project successfully.

I would also like to express my deepest gratitude to my friends and family members for their motivation, emotional support and guidance at various stages of my project

SOUMEN DAS

PCS051822

Abstract

Phishing attack is now a big threat to people's daily life and networking environment.Phishing is one of the most harmful social engineering techniques to subdue end users, where threat actors find a chance to gain access to critical information systems. A common approach in phishing is through the use of e-mail communication with an embedded hyperlink. The detection and mitigation of phishing attacks are a grand challenge due to the complexity of current phishing attacks. Existing techniques are often too time consuming to be used in the real world in terms of detection and mitigation time. By disguising illegal URLs as legitimate ones, attackers may induce users to visit the phishing URLs to get private information and other benefits. There is an urgent need for effective methods to identify the phishing websites to mitigate the risks presented by the phishing attacks.The neural network is commonly used to detect phishing attacks, as the active learning function from large data sets. However, many useless and small influence features at the training data sets stage will trap the neural network model into the over-fitting problem. This problem typically triggers the trained model, which can not detect phishing websites effectively.This paper proposes OFS-NN, an efficient model for the detection of phishing websites based on the optimal feature selection approach and neural network, to mitigate this problem.A new index, the feature validity value (FVV), is implemented in the proposed OFS-NN to measure the effect of sensitive features on the identification of phishing web pages. Then an algorithm is programmed to pick the best features from the phishing websites, based on the latest FVV index. This algorithm will to a large extent ease the over-fitting question of the underlying

neural network.To train the underlying neural network, the selected optimal features are used, and finally, an optimal classifier is designed to detect the phishing websites. The experimental results show that the model OFS-NN is accurate and stable in the identification of many phishing websites. The experimental results have shown that the OFS-NN model provides an effective solution for predicting and detecting phishing websites.

Table of contents

List of figures

List of tables

Chapter 1

INTRODUCTION

1.1 Introduction

Phishing is a form of fraud in which the attacker tries to learn sensitive information such as login credentials or account information by sending as a reputable entity or person in email or other communication channels. Typically a victim receives a message that appears to have been sent by a known contact or organization. The message contains malicious software targeting the user's computer or has links to direct victims to malicious websites in order to trick them into divulging personal and financial information, such as passwords, account IDs or credit card details.Phishing is a fraudulent attempt, usually made through email, to steal your personal information. The best way to protect yourself from phishing is to learn how to recognize a phish.Phishing emails usually appear to come from a well-known organization and ask for your personal information — such as credit card number, social security number, account number or password. Often times phishing attempts appear to come from sites, services and companies with which you do not even have an account. In order for Internet criminals to successfully "phish" your personal information, they must get you to go from an email to a website. Phishing emails will almost always tell you to click a link that takes you to a site where your personal information is requested. Legitimate organizations would never request this information of you via email. A vulnerable hyperlink to a malicious website,

waiting for a user's press, is among a overwhelming pile of unread email messages. This message from a threat actor is commonly called phishing, where the threat actor is waiting for the end User clicking on such a link could compromise an organization or a computing system.

1.2 Background to the project

Generally speaking, the possible values of the phishing websites detection can be classified as a two-way problem: the "phishing websites" or the "legitimate websites". Several methods are suggested to train and educate end users to identify and detect phishing URLs to minimize the threat of phishing attacks. To some extent, these approaches take effect by periodically sending messages to warn end-users of potential phishing threats. But they still depend on the behaviors of the users and knowledge of the underlying systems being used. Due to high accuracy and efficiency, the software based automatic method is widely used to detect phishing attacks. At present, the automatic phishing detection methods can be classified into four categories: the blacklist and white list methods, the heuristics methods, the visual similarity methods and the machine learning methods [12].

Through recording previously detected phishing or legal URLs, IP addresses and keywords, the method of constructing black and white lists can effectively prevent phishing attacks[13]. Due to small workload on analyzing the content of websites, this method has its advantage of requiring small resources on the underlying systems.

The heuristic phishing detection approaches can be taken as the extension of black and white lists [14]. The heuristic approaches for identified phishing attacks are usually based on assigned signatures. Such approaches raise a warning by scanning websites for the assigned signatures, if malicious behaviors are found.

By visually comparing the suspect website with the legitimate target, the method of visual similarity can also achieve the same precision as the technique of blacklist and white list [15].

This method is based on catching snapshots of the appearances of websites in web browsers and sorting the snapshots that have been acquired. Such works can however incur very high costs of time and space.

Given the active learning capability from massive data sets, the machine learning method is widely used to detect the phishing web sites. This method usually uses the feature selection algorithm to extract a vector of sensitive features that may at first help to distinguish phishing and legitimate websites. Then, the underlying machine learning classifiers are equipped to detect the phishing websites based on the extracted features [16]. Typically, the classifiers are designed based on the model of the neural network, the model of the support vector machine, the model of the Naïve Bayes, etc. The method of machine learning is accurate in the detection of phishing websites.

This paper proposes OFS-NN, an effective model for detection of phishing attacks based on optimal feature selection and neural networking. Under this model the new FVV index is first defined to assess the impact of sensitive features on the detection of phishing websites. Then an algorithm is designed, based on the FVV index, to select optimal features from the phishing URLs and their respective websites. Finally, the selected features are used to train the underlying neural network to construct a classifier for phishing attack detection. In general terms this paper's contributions are listed as follows:

1. Defines a new index—FVV: In order to better evaluate the impact of a selected sensitive feature on detecting phishing attacks, this paper presents the FVV index.

2. Designs an optimal feature selection algorithm: This algorithm calculates the FFV values of all features of the input URLs and their relevant websites at first. Then, a threshold is set to select sensitive features to construct an optimal feature vector.

3. Presents the OFS-NN model: Through the selected sensitive features and a large number of experimental analyses, the optimal structure of the neural network is trained and constructed as the final classifier of the OFS-NN model.

1.3 Motivation to the project

Since phishing attacks usually take advantages of users' careless behaviors or ignorance on using networking tools, it is a hard problem to be permanently resolved [17]. Southeast Asia is one of the biggest hub for phishing attacks. Quite lately, Kaspersky revealed a report that there were already 14 million phishing attempts in this region within June 2019. This is the region that cyber criminals target by infecting their devices and networks and various other phishing tricks [18]. Phishing is one of the most harmful social engineering techniques to subdue end users, where threat actors find a chance to gain access to critical information systems. A common approach in phishing is through the use of e-mail communication with an embedded hyperlink. The detection and mitigation of phishing attacks are a grand challenge due to the complexity of current phishing attacks. Existing techniques are often too time consuming to be used in the real world in terms of detection and mitigation time. This paper proposes OFS-NN, an effective model for detection of phishing attacks basedon optimal feature selection and neural networking

1.4 Organization of Thesis

The thesis is organized as follows. Chapter 2 is about the literature reviews on the area of various phishing detection techniques. Chapter 3 discusses about the database acquisition, design of the database and methodologies used in the project (feature extraction techniques, feature selection technique and neural network model). Chapter 4 is about the experimental analysis and results. The results are studied in this chapter to evaluate the system performance.Chapter 5 lists the papers referred for doing this work.

Chapter 2

REVIEW OF LITERATURE

2.1 Introduction

A literature review describes published information in a specific subject area, and sometimes knowledge in a specified subject area within a certain period of time. This chapter gives a description about the previous classification techniques,related work, and review of literature and corresponding author names and publishing year of this paper.

2.2 Related Work

Software-based automatic detection system is the most effective approach to ease the challenge of phishing threats, comparing to teaching and educating end users. The black and white lists[3], as the most direct process, will reduce phishing attacks with relatively low usage of resources. Here, The authors say about effectiveness of blacklist phishing. This paper has published on January, 2009 by S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang. It used 191 fresh phishes to perform two experiments on eight anti-phishing toolbars that were less than 30 minutes old. They had found that 63% of our data set's phishing campaigns lasted less than two hours. Initially, blacklists were

unsuccessful at shielding consumers, as most captured fewer than 20 % of phish at zero hour. They also observed that blacklists were revised at various rates and ranged in scope, as 47%-83% of phishes appeared 12 hours from the initial check on blacklists. They found that two methods that use heuristics to complement blacklists initially captured slightly more phish than those that use blacklists alone.It took a long time though for heuristics-detected phish to appear on blacklists. Finally, They checked the toolbars for false positives on a set of 13,458 legal URLs, and found no instance of mislabeling either for blacklists or for heuristics. We present these results and explore ways to strengthen the anti-phishing tools.The Google Safe Browsing API is a symbolic blacklist function, which Google Inc. retains continuously. [4].The safe browsing Go package can be used with the Google Safe Browsing APIs (v4) to access the Google Safe Browsing lists of unsafe web resources. Inside the cmd sub-directory, you can find two programs: sblookup and sbserver. The sbserver program creates a proxy local server to check URLs and a URL redirector to redirect users to a warning page for unsafe URLs. The sblookup program is a command line service that can also be used to check URLs Client applications can test whether the target URL is in the blacklists by interacting with the defined APIs.

"Advanced white list approach for preventing access to phishing sites"[5] is introducing a method to detect the phishing websites based on the white list.This paper has published on November,2007 by J. Kang and D. Le. In this paper they present the white list-based approach that prevents access to explicit phishing sites and warns by the URL similarity check for phishing-suspicious accesses. We propose a mechanism comparing the results of the DNS query to cope with the Local and DNS Pharming Attacks. Implementation called Phishing Guard and test examples also be added in this paper.This method organizes user access to the site by detecting similarity between the URLs."A phishing sites blacklist generator" [6] is published on April 2008 .The authors Sharifi and Siadati suggest a system for the identification of phishing websites through blacklist generator. It tackles local and

DNS spoofing attacks by contrasting the outcomes of DNS enquiries. This approach decides if it is a website for phishing by comparing the website's domain name and the search results from Google.Sandboxes live phishing tool kits [7] is introduced by X. Han, N. Kheir, and D. Balzarotti on October,2016.it is calculating the effect of blacklisting services on phishing websites when such services are enabled in the beginning. L.H. Lee, K.C. Lee, H.H. Chen, and Y.H. Tseng have proposed "POSTER: Proactive blacklist update for anti-phishing" [8] on November 2014.

In order to mitigate the threats of newly emerged phishing URLs,PhishTrack frame work is used automatically updating the blacklist of phishing sites. The black and white lists consume small resources on the underlying systems.This PhishTrack frame demonstrates the viability of investigating new ones Blacklists to discover Phishing URLs. It's time. A proactive blacklist update process consisting of redirection tracking and type tracking Suspicious URLs. A. Aleroud and L. Zhou has has proposed "Phishing environments techniques" on Jul. 2017 [9]. However,it can not properly deal with the newly emerged phishing attacks.In order to mitigate this shortage,as the work that is presented in this paper,the black and white lists are usually working in collaboration with other methods [10].

Due to active learning ability, machine learning methods are extensively studied to detect the phishing websites. Cantina [19] is a phishing attacks detection model built on 27 sensitive features extracted from URLs and their relevant websites. This model makes use of the TF-IDF algorithm to detect phishing attacks. This algorithm can detect many kinds of phishing attacks but at the expense of consuming much time cost of the underlying systems. Meanwhile, some legal websites are reported as phishing ones [20]. CANTINA+[12] is the continuation work of Cantina. Compared with Cantina, CANTINA+ adds 10 more features. Meanwhile, the phishing detection TF-IDF algorithm is replaced with SMV. Through these improvements, shortages of Cantina are mitigated.However,the new CANTINA+has a narrow range of applications [21].

As an automatic phishing detection model,Phish Storm [22] is implemented as an interface between social networking tools and email servers or HTTP proxies. In this model, a random forest classifier is trained by extracting 12 URL relevant features.However,due to small coverage of sensitive features, it cannot detect many types of phishing attacks. Phish Shield[23] is implemented as heuristic phishing detecting tool running on top of personal computers. This tool is able to identify phishing attacks that are designed by changing contents of websites. Because of lacking active learning capacity from phishing attacks, it cannot properly deal with attacks that continuously change their sensitive features.The PhiDMA [24] model is implemented as a multi-filter. It consists of five levels of filters: the auto-upgrade white list layer, the URL feature layer, the lexical signature layer, the string matching layer and the accessibility rating comparison layer. This model detects phishing websites based on accessibility errors (known error, likely error and potential error) comparison. However, in some websites, the model has lacked the ability to encounter all the three errors. Meanwhile, this model only gets 92.72% phishing network detection accuracy. The Off-the-Hook [25] model is built as plugins to certain browsers. Through combining blacklist, machine learning method and 210 features, this model can detect many phishing attacks. However, too many sensitive features seriously degrade the performance of this model.

Chapter 3

METHODOLOGY

a description about the database used for this work and discusses the feature extraction,optimal feature selection, and neural network classification techniques used.The aim is to classify 'phishing websites' or 'legitimate websites'. Dataset plays an important role to assess and validate any machine learning model.

3.1 Data sets

Here, two data sets has been used to build neural network model Two data sets are used to evaluate the performance of the proposed OFS-NN model. Data set 1 is selected form UCI phishing data set [1]. Data set 2 is built on PhishTank records and the Alexa records [2]. A brief description is given on table 3.1:

Table 3.1 Data sets description

Data set	Source	Instance	No of phishing website	No of legitimate website
Data set 1	UCI phishing data set	11055	4899	6156
Data set 2	PhisTank and Alexa record	2117	998	1119

3.2 Optimal Sensitive Feature Generation

In the Internet environment, A Uniform Resource Locator (URL) is another form of addresses for a particular websites. URL is a reference to a web resource that defines its location in the Internet environment and retrieve it what is in the location. The common URL is made up of four components: the protocol, the domain name, the file path and the query parameters. Most web browsers display the URL of a web page above the page in an address bar. A common syntax of URL is describe on the figure 3.1.

Fig. 3.1 A common syntax of URLs

As an example of common URL looks like http://www.exampleurl.com/info/about.html, which indicates a protocol (http), a host name (www.exampleurl.com) includes top level Domain (.com), 2^{nd} level domain (exampleurl) and 3^{rd}, level domain or sub-domain (www.), directory (info) and a file name (about.html). According to the addresses specified by URLs, Internet resources (like documents, images and videos), can be accessed.

Phishing attackers usually mask illegal URLs as legitimate ones. Fortunately, phishing URLs have obvious identifiable features, different from legal URLs. For making a better classifier features data plays a great role. So, First it requires to extract sensitive features from URLs (legitimate and phishing both) and their relevant websites, Through this features extracting algorithm may bring many useless and small influence features. Due to no disturbance from these redundant features, the over fitting problem of the underlying neural network is all deviated. So a optimal features selection algorithm is required to detect optimal features vectors.

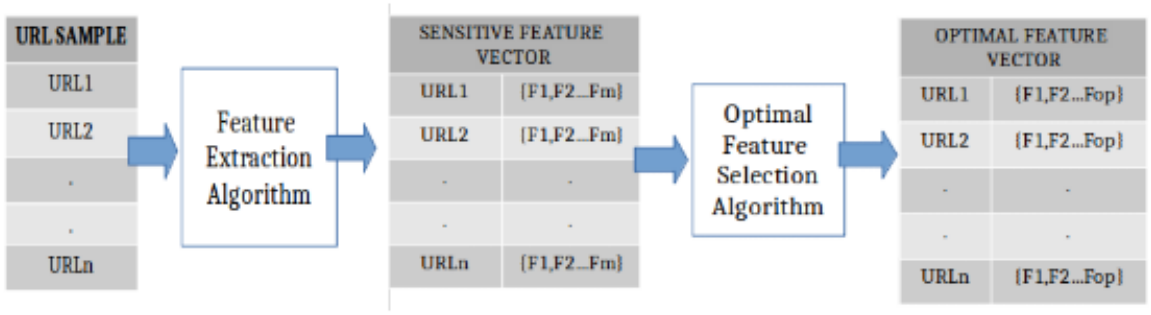


Fig. 3.2 Optimal Sensitive Feature Generation

As shown in Figure 3.2, The extracting algorithm is firstly employed to extract sensitive features from the input sample URLs and their relevant websites. Then, the selecting algorithm is used to generate the optimal features vector.The process of selecting optimal features can reduce the workload of training the underlying neural network classifier. Finally, the refined URL sample set with optimal features will be taken as the input to train the underlying classifier.

3.2.1 Sensitive Feature Extraction from URLs

Illegal URLs are generally disguised as legitimate ones by phishing attackers. By doing so, the attacker can lure users to click to illegal website. Only sensitive features is able to identify the URL is phishing URLs or not. Some identifiable features is shown in tabular form : Fig.3.3 . The sensitive features are categorized into four classes:

- The address bar relevant features
- The abnormal features
- The HTML and JavaScript relevant features
- The domain relevant features.

Each feature is assigned with a number from three values (1, 0 or 1) for each of them.

No	Sensitive Features	Explanation	Conditions
F1	having_IP_Address	URL contains an IP address	If URL contains ,F1 = 1; otherwise,F1 =−1
F2	URL_Length	length of the input URL	If length >75, F2 =1; if length < 54 , F2 = −1; otherwise, F2 = 0
F3	Shortining_Service	URL uses short address	If URL uses short address, F3 =1; else, F3 =1.
F4	having_@_Symbol	URL contains the '@' symbol	If URL contains '@', F4 =1; otherwise, F4 =−1
F5	Double_slash_redirecting	URL uses '//' symbol	If '//' appears after seventh character in URL, F5 =1; else, F5 =−1
F6	Prefix_Suffix	URL contains the '-' symbol	If URL contains '-' , F6= 1; otherwise,F6 = −1
F7	having_Sub_Domain	Number of '.' symbols in an URL	If (Num== 1), F7 =−1; if (Num== 2), F7 = 0;if (Num !<}3), F7
F8	SSL final_State	HTTP protocol and SSL certificate statuses	If (age !< 1 year), F8 =1; otherwise, F8 =1
F9	Domain_registeration_length	Expiry time of domain name	If expiry !> 1 year, F9= 1; otherwise, F9 =0
F10	Favicon	Sources of "favicon/icon" of the URL	If "favicon/icon" is loaded from other domains, F10 =1; else,F10= −1
F11	port	Standard port numbers are used in a website	If port is abnormal, F11=1; else, F11=−1
F12	HTTPS_token	URL has "https" strings in the domain name	If contains "https", F12 =1; else, F12 = −1
F13	Request_URL	Percentage of resources from the same domain	If percentage <22%, F13 =−1; if percentage> 61%, F13 =1 otherwise, F13 = 0
F14	URL_of_Anchor	"anchor" tag in the web page	If <a> tag ratio <31%, F14 =−1; if<67%, F14 =1; else, F14=0
F15	Links_in_tags	links in tags in the HTML page of the website	If link tags ratio<17%, F15 =−1; if>81%, F15= 1; else, F15 = 0
F16	SFH	SFH of web pages	If SFH contains a null character, F16=1;otherwise, F16=−1
F17	Submitting_to_email	Web page contains an email address	If webpage contains email address, F17=1; else, F17= −1
F18	Abnormal_URI	WHOIS" information of the website	If URL exists in WHOIS database, F18=1; otherwise, F18=−1
F19	Page Redirect	Pages redirected by the website	If the number of forwarding<1, F19 = −1; if < 4, F19 =1; otherwise, F19 = 0
F20	on_mouse over	"on Mouse Over" event on the web page	If Mouse over event,F20=1Else,F20=-1
F21	Right Click	Web page prohibits the right click of the mouse	If right click, F21=1; otherwise, F21= −1
F22	pop Up Window	Web page uses a pop-up window	If it is used, F22= 1; otherwise, F22= −1
F23	Iframe	"Iframe" setting in the web page	If Iframe is used, F23= 1; else, F23= −
F24	age_of_domain	Age of a website	If age<6 months, F24 =1; otherwise, F24 =−1
F25	DNS_Record	DNS record of the website	If domain has DNS record, F25=-1other wise,F25 =1
F26	web_traffic	Website traffic	If Page traffic<100000,F26=-1; If Page traffic > 100000 , F26 =0 ; else,F28=1
F27	Page_Rank	Page Rank of a website	If Page rank<0.2,F27=1;else,F27=-1
F28	Google_Index	Google Index of the website	If Page Rank<0.2, F28=1; otherwise,F28=−1
F29	Links_pointing_to_page	Links pointing to web page	If number of links ==0, F29= 1; if number <=2, F29=0; otherwise,F29−1
F30	Statistical_report	Website is in the PhishTank or statistical report	If PhishTank or Stop Badware reports, F30 =1; else, F30 −1.

Fig. 3.3 All Sensitive Features

Address bar features :

It is very important to test if the input URLs contain short addresses, IP addresses, and special characters (e.g. "@', "/", "-" and "'.") to detect phishing attacks accurately. Meanwhile, it is all important to define the lengths of the input URLs.Phishing attackers can use short

addresses to replace legitimate addresses to trick users. Currently, content prior to the "@" character is ignored during the process of a Web browser parsing URLs. Phishing addresses are usually appended after this character by the use of this feature. The character " ." is used to connect multiple sub domains of a URL.These sub-domains can be used to conduct phishing attacks. Additionally, the port status, address bar source icon and SSL certificate status must also be detected via URL. This information on the phishing website should act abnormally, under normal circumstances.Phishing URLs have shorter domain registration time and certificate time than legal ones and also short life cycles. In addition, many phishing websites don't have the SSL certificates.

In figure 3.3, F1 to F12 are the Address bar features.F1 is used to specify if the input URL contained IP address or not in the domain . If the URL is used an IP address,then F1 is set to 1;otherwise, F1 is assigned to 1. F2 is specified the length of a input URL. If the URL contains more than 75 characters, F2 is set to 1; if the corresponding URL contains less than 54 characters, F2 is set to 1; else, F2 is set to 0. F3 defines whether a short address is used in the URL. If the target URL uses a short address, F3 is set to 1; else, F3 is assign to 1. F4 specifies whether URL contains the '@' symbol. If the URL have '@' symbol, F4 is fixed to 1; else, F4 is fixed to 1. F5 is detected whether URL include '//' symbol or not. If the symbol '//' appears after the seventh character in a URL, F5 is assign to 1; else ,F5 is assign to 1. F6 specifies '-' symbol is included in the URL. If the target URL contains the '-' symbol, F6 is assign to 1; else , F6 is assign to 1. F7 is specified the number of '.' symbols in an URL. If this number is equal to 1, F7 is assign to 1; if the number is equal to 2, F7 is assign to 0; if the number is not less than 3, F7 is assign to 1. F8 detects the HTTP protocol and SSL certificate statuses of the target website. If the age of the used "https" by the trusted issuer is no less than 1 year, F8 is fixed to 1; if the used "https" by the untrusted issuer, F8 is fixed to 0; else, F8 is fixed to 1. F9 is specified the expiry time of domain of the target website. If the domain expiry time is no more than 1year,F9 is assign to 1;else,F9 is assign to 0. F10 is used

to denote the sources of "favicon/icon" of the target URL. If the "favicon/icon" is used from other domains, F10 is assign to 1;else, F10 is assign to 1. F11 is specified the situation of standard port numbers (such as 21, 22, 23, 80, 443, 445, 1433, 1521, 3306, etc.) are used in a website.If the port status is not standard, F11 is assign to 1; else, F11 is assign to 1. F12 detects whether the target URL has "https" strings in the domain name . If the domain name contains "https", F12 is assign to 1; else, F12 is assign to 1.

Abnormal features :

Analysing of the page contents of the input URLs can help extract the abnormal features. The relevant features can be obtained by checking the number of requested links on the page, the number of hyperlinked tags, the SFH status information, the "WHOIS" information and whether the page contains the mailbox information. Those features behaves abnormally for most of the phishing websites. The following abnormal behaviors also need to be recorded: the segments between the < a > tag and the website are different; the SFH null value; the tags < Meta >, < Script > and < Link > point to specific web pages; the input URL contains email address.

In fig. 3.3, Features F13 to F18 are known as abnormal features. F13 is specified the percentage of resources from the same domain that is requested by a single URL. If the percentage is less than 22%, F13 is assign to 1; if the percentage is greater than 61%, F13 is assign to 1; else,F13 is assign to 0. F14 is defined to detect pointing direction of the "anchor" tag in the specific web page. An anchor is used as the <a> tag in the HTML. If the ratio of the anchor in the web page is less than 31%, F14 is assign to 1; if this ratio is greater than 67%, F14 is assign to 1; else, F14 is assign to 0.F15 is used for examine links in tags in the HTML page of the website. If the ratio of links in tags (<meta>, <script> and <link>) is less than 17%, F15 is assign to 1; if this ratio is greater than 81%, F15 is assign to 1; else, F15 is assign to 0. F16 is used to get the SFH of web pages. If the SFH contains a null character or

“about:blank”, F16 is assign to 1; if the domain name of the SFH is different from the web page, it is assign to 0; else, it is assign to 1. F17 is define whether the web page contains an email address. If the web page contains the email address, it is assign to 1; else, it is assign to 1. F18 defines the “WHOIS” information of the website. If the URL exists in the WHOIS database, it is assign to 1; else, it is assign to 1.

HTML and Java script features :

In fig. 3.3 , Features F19 to F23 are the relevant features that can be extracted from the HTML and JavaScript source codes. The corresponding features are: a web page’s redirect and label element numbers, the event "on mouse over" changes the status bar of the input URL and the right-click operation is disabled.

Truly speaking, F19 measures the number of pages that the website redirects. If the number of forwarding websites is less than equal to 1, F19 is assign to 1; if this number is greater than 4, F19 is assign to 1; else, F19 is assign to 0. F20 measures whether there is an "on Mouse Over" event on the web page. If the web page has used the "on Mouse Over" event and the changes to the status bar, it is assign to 1; else, it is assign to 1.F21 measures whether the web page prohibits mouse operation by right-clicking. If operation of the right-click mouse is prohibited, it is assign to 1; else, it is assign to 1. F22 checks whether the web page uses a pop-up window (using pop-up window). If it is used, it is assign to 1; else, it is assign to 1. F23 defines the setting of the "Iframe" in the web page. If the "Iframe" is used, it is assign to 1; else, it is assign to 1.

Domain features :

The phishing websites have short life cycles. A phishing web site can not locate the corresponding DSN record. Statistical indices, such as website traffic, Page Rank and the Google Index, assess the websites’ significance. The phishing websites usually have fewer visitors.

If the traffic is low a URL can be a phishing one.The Page Rank index is a metric for web pages to determine their significance. Phishing URLs usually have Page Rank values zero or much lower than standard ones. Because of short life cycles, Google hardly indexes the phishing URLs. Since the PhishTank usually publishes newly emerging phishing websites, it is important to decide if the target URL is in the PhishTank database.

In the fig. 3.3, Features F24 to F30 are the corresponding features. F24 calculates the age of a website. If the age of the domain is less than 6 months, F24 is assign to 1; else,F24 is assign to 1. F25 defines the DNS record of the website. If the domain has the DNS record, F25 is assign to 1; else,F25 is assign to 1. F26 measures the website traffic. If the websites traffic is less than 100,000, it is assign to 1; if the web page traffic is more than 100,000,it is assign to 0;else, it is assign to 1. F27 define the Page Rank of a website. If the Page Rank value is less than 0.2, it is assign to 1; else, it is assign to 1. F28 defines for the Google Index of the website. If the target URL is included in the Google Index, it is assign to 1; else, it is assign to 1. F29 calculates the number of links pointing to the target web page and specifies whether the link label on the URL pointing to the same domain.If the number of links pointing to the webpage is equal to 0,it is assign to 1; if this number is less than 2, it is assign to 0; else, it is assign to 1. F30 uses to find whether the website is in the PhishTank or Stop-Badware statistical reports. If the URL is in the PhishTank or Stop-Badware reports, it is assign to 1; else, it is assign to1.

SENSITIVE FEATURES EXTRACTION ALGORITHM

It is necessary to extract sensitive features from URLs and their corresponding websites and set them as the input of the proposed OFS-NN [11].

Fig. 3.4 gives the algorithm [11], which one is used to extract the above features from the input URLs and their relevant websites. In Fig. 3.4, $F_v[1...30]$ is the vector for storing 30 sensitive features of an input URL. As shown in Fig. 3.4, the list of URLs is taken as the

Input: UrlList;
Output: The set of F_v;
Each feature is assigned with a value in {1, 0, -1}.

```
(1)   Load(UrlList);
(2)   for each URL in UrlList
(3)     UrlInfo←URL.find();
(4)     PageInfo←URL.request();
(5)     DomainInfo←URL.requestDomain();
(6)     Fv←[0]×30;
(7)     Fv [1…12] ←extract Address bar features from UrlInfo;
(8)     Fv [13…18] ←extract Abnormal  features from PageInfo;
(9)     Fv [19…23] ←extract HTML/JavaScript features from PageInfo;
(10)    Fv [24…30] ←extract Domain features from DomainInfo;
(11)  end for.
```

Fig. 3.4 Sensitive Features Extraction Algorithm

input of the algorithm (Line 1); then, for a given URL, the corresponding feature information is acquired (Line 3 - Line 5); F_v is used to save the extracted sensitive features (Line 7 - Line 10).

3.2.2 Optimal Features Selection from Sensitive Feature

In the Fig.3.2, After features extraction,we have got sensitive features vector.now,next step is optimal feature selection. To train an optimized suitable model for detecting phishing websites, a concisely sensitive feature vector from the sensitive features vector is required. new FVV index is used efficiently to pick optimal sensible features. Through the FVV index, an algorithm can be built for the selection of optimal features to train the underlying optimal neural network model.

FVV Index :

Suppose there is a data set with n sample in a form of $D = [(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), ..., (x^{(n)}, y^{(n)})]$, where x define as features vector and y define as results of classification. For a given feature "A" with the positive value, if it is also predicted as positive value by a classifier, "A" is

called the positive feature. On the other hand, the the value of "A" is negative also predicted as negative by a classifier, then "A" is called the negative feature. As a matter of fact, only positive features and negative features is used to define FVV Index to build feature selection Algorithm .The FVV index is defined in equation (3.1) as follows:

$$FVV = P(A_{(X=positive)and(y=positive)}) + P(A_{(X=negative)and(y=negative)}) \tag{3.1}$$

In the phishing data set 1, $P(A_{(X=negative)and(y=negative)})$ indicates the probability that the specified feature "A" is represented as a phishing website and the detection result (corresponding target value) is also the phishing one; $P(A_{(X=positive)and(y=positive)})$ indicates the probability that the feature "A" is represented as a legitimate website and the detection result (corresponding target value) is also the legitimate one.

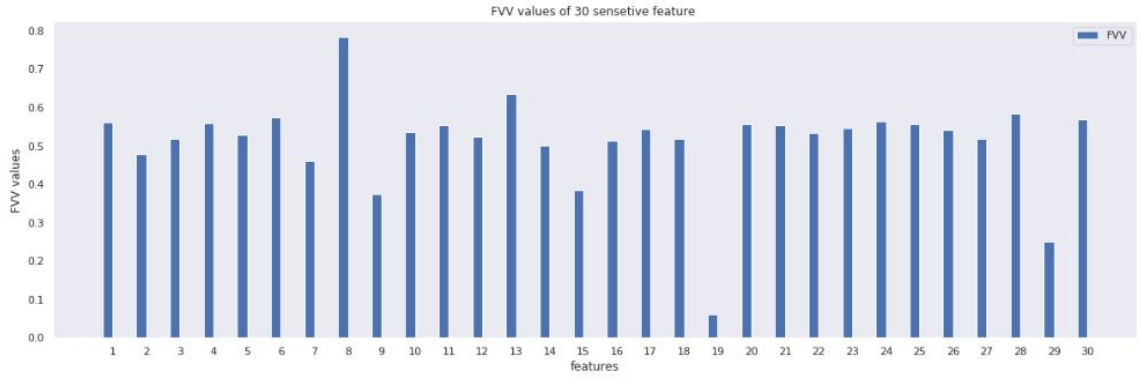


Fig. 3.5 FVV values of 30 sensitive features

In the Fig.3.5, calculates the FVV values of 30 sensitive features for Data set 1. Certain useless and small impact features can be eliminated by calculating the FVV values to reduce the workload of the entire OFS-NN model.

Optimal Sensitive Feature Selection Algorithm :

For selecting optimal features, a threshold is required to eliminate useless and small impact features. Based on the FVV index,the threshold is calculated in equation (3.2) as follows:

$$\rho = \sum_{i=1}^{m} FVV_i/(2 \times m) \tag{3.2}$$

In the formula 3.2, m is the number of sensitive features; is set as half of the average FVV values of all features. For input data set, FVV values of all features of n samples are calculated at first. Then, the threshold can be calculated by comprehensively evaluating all the FVV values. For data set 1, as 30 sensitive features are collected, m is set to 30.

Input: Feature vector $F=(f_1, f_2, \ldots, f_m)$; Threshold ρ;
Data set $D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(n)}, y^{(n)})\}$.
Output: Optimal feature vector $F_{ofs}=(f_1, f_2, \ldots, f_k)$.

(1) Initializes F_{ofs} and ρ;
(2) **for** i=1 to n **do**
(3) **for** j =1 to m **do**
(4) Calculates $P(A_{x=positive \text{ and } y=positive})$ and $P(B_{x=negative \text{ and } y=negative})$
for all features
(5) Calculates FVV values for each feature by equation (1) and stores them in an orderly table;
(6) **end for;**
(7) **end for;**
(8) **for** i=1 to m **do**
(9) **if** the FVV value of a feature > ρ (according to equation (2))
(10) $F_{ofs} \leftarrow (f_1, f_2, \ldots, f_k)$;
(11) **end for.**

Fig. 3.6 Sensitive Features Selection Algorithm

In Fig. 3.6, we define a feature selection algorithm [11]. Based on data set 1,the threshold of FVV is calculated as 0.257 according to the equation 3.2. According to the algorithm (In the fig. 3.5), two small impact features (F19 and F29) can be eliminated by the phishing websites sensitive features extraction to optimal features selection algorithms.

As shown in Fig. 3.5, the FVV values of the two features are less than the threshold. So,the two features from the target data set can be deleted and use only 28 useful features to train neural network classifier. Specifically, for a given input URL, the 28-dimensional optimal features vector is extracted. Since FVV values of useless and small impact features are generally smaller than the average value of all features, they can be deleted for obtaining an optimal features vector.

3.3 Neural Network Classification

The structure of a neural network model is composed of 3 layers: the input layer, the hidden layers and the output layer. Neural network algorithms generally incorporate two phases: the forward propagation and the backward propagation.

The forward propagation starts to work when it receives a positive propagation signal. The backward propagation is invoked when the model detects the occurrence of evident deviation between the calculation result of the output layer and the actual value. Backward propagation uses the gradient descent algorithm to adjust parameters of various neural unit layers to minimize the deviation.

3.3.1 Neural Network Classification Training :

In general terms, more layers will bring better neural network performance. Too many layers, however, will trap the neural network into the over-fitting problem. The over-fitting problem will seriously degrade the detection accuracy of the underlying neural network classifier. For this purpose, the neural network structure obtained must satisfy two criteria, without any over-fitting problem and with greater accuracy in detection. To get an optimal neural network structure, eight models with different layers and hidden units are compared. The eight models include:

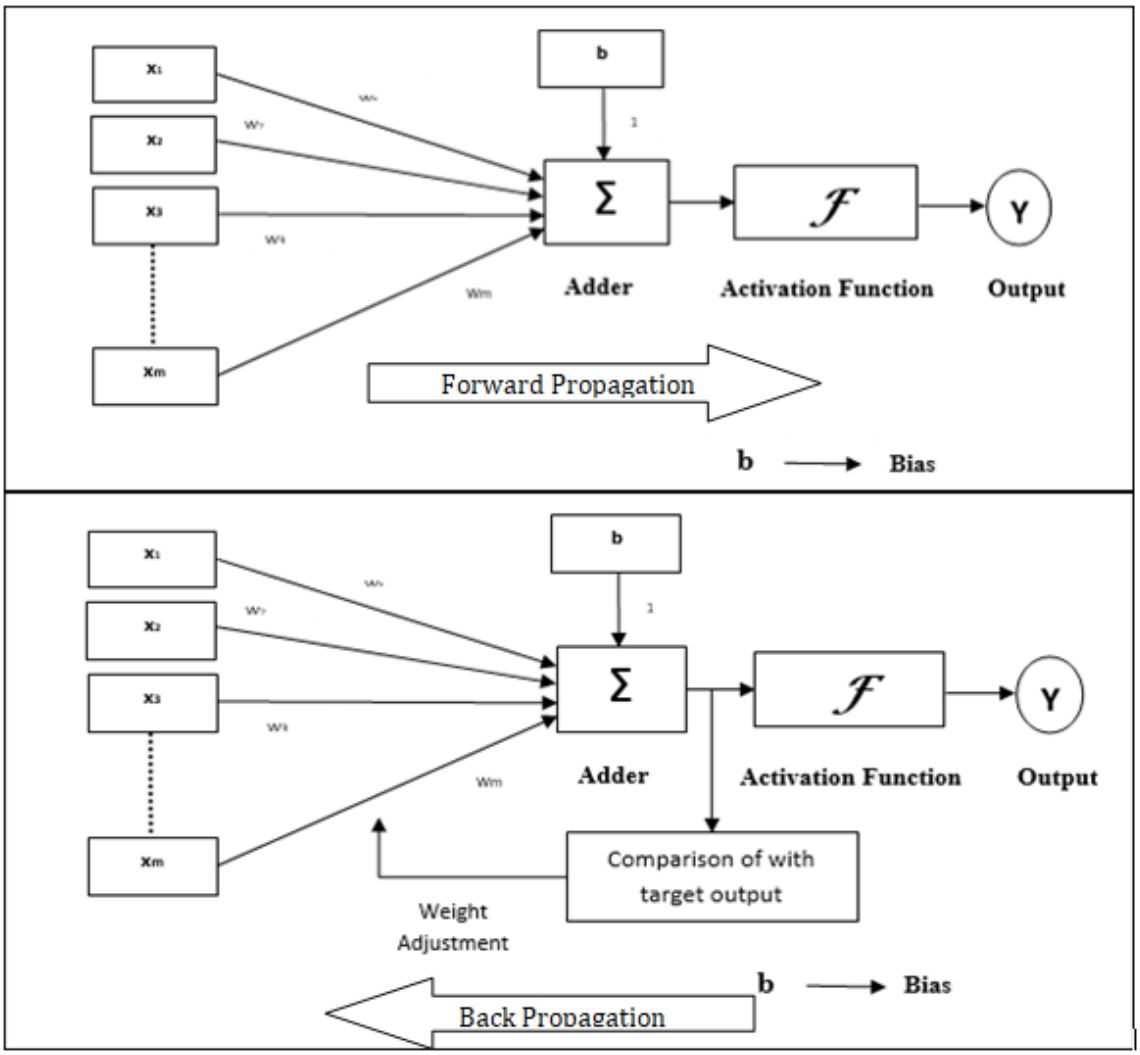


Fig. 3.7 Work Flow of Forward Propagation and Backward Propagation

- 1-layer with no hidden unit
- 2-layer with 5-1 hidden units
- 3-layer with 20-5-1 hidden units
- 4-layer with 50-20-5-1 hidden units
- 5-layer with 100-50-20-5-1 hidden units
- 6-layer with 150-100-50-20-5-1 hidden units

- 7-layer with 200-150-100-50-20-5-1 hidden units
- 8-layer with 300-200-150-100-50-20-5-1 hidden units

Table 3.2 Accuracy table for all Neural Network model

Model	Training Accuracy	Testing Accuracy
1st	0.9541	0.9527
2nd	0.9309	0.9297
3rd	0.9616	0.9601
4th	0.9812	0.9773
5th	0.9877	0.9834
6th	0.9879	0.9819
7th	0.9879	0.9834
8th	0.9850	0.9795

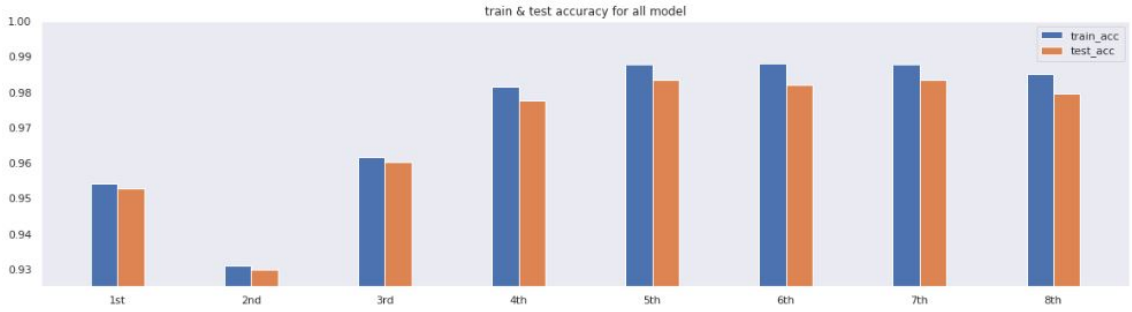


Fig. 3.8 Comparison of Training and Testing Accuracy of Neural Network Models with different layer

It is important to mention that before the stage of training the neural network classifier, the optimal features vector is generated for a given training data set (data set 1).Table 3.2 shows the training and testing accuracy of the eight different models indifferent proportions of URLs. Fig. 3.8 shows the visualization comparison of the eight different models. From this figures, it can be seen that the accuracy drops when the 8-layer neural network is used. The accuracy of the 7-layer neural network is the highest, which can get 98.79% training Accuracy and 98.34% testing accuracy accuracy.

As a matter of fact, in the selection of structure of neural network, the number of neurons in the input layers and the number of neurons in the output layers are set as the number of selected features and the number of classification categories respectively .Meanwhile,with the deepening of the layers of the neural network,the number of the current hidden units has to be ensured that they are bigger than the ones of the former layer.

3.3.2 Optimal Neural Network Model

Based on the above analysis, the 7-layer fully connected neural network is selected as the underlying classifier of our OFS-NN model. Specifically, it has 526 (200+150+100+50+20+5+1) hidden units, 56705 (28×200 + 200×150 + 150×100 + 100×50 + 50×20 + 20×5 + 5×1) weights and 7 biases. Table 2 gives the detailed configuration of the classifier. Fig. 3.9 shows the overall structure of the underlying neural network Classifier of our OFS-NN mode

Input Nodes	Output Nodes	Hidden Layer	Hidden Units	No of Weight	No of Bias
28	1	7	526	56705	7

Fig. 3.9 Configuration of Optimal Neural Network Classifier

Before the Classifier starts to work, the optimal feature selection algorithm is used to transform the original feature vector $(X_1, X_2, ..., X_{ofs})$ into the optimal feature vector $(w_1, w_2, ..., w_{ofs})$ for each input URL. Consequently, the weight parameter sequence will also change to $(w_1, w_2, ..., w_{ofs})$. URLs with the generated optimal feature vector are taken as the input of the Classifier. Meanwhile, the bias b is input to the activation function g(-). The g(-) function is the ReLU activation function. To engage in the process, the value determined by the current layer of the neuron activation function is then inserted into the next layer of the neurons. It calculates those functions iteratively until the seventh layer results are produced. The result of the seventh layer is then put to the g(-) activation function to generate the final result Y of the Classifier. The value of Y is also the detection result of the entire OFS-NN

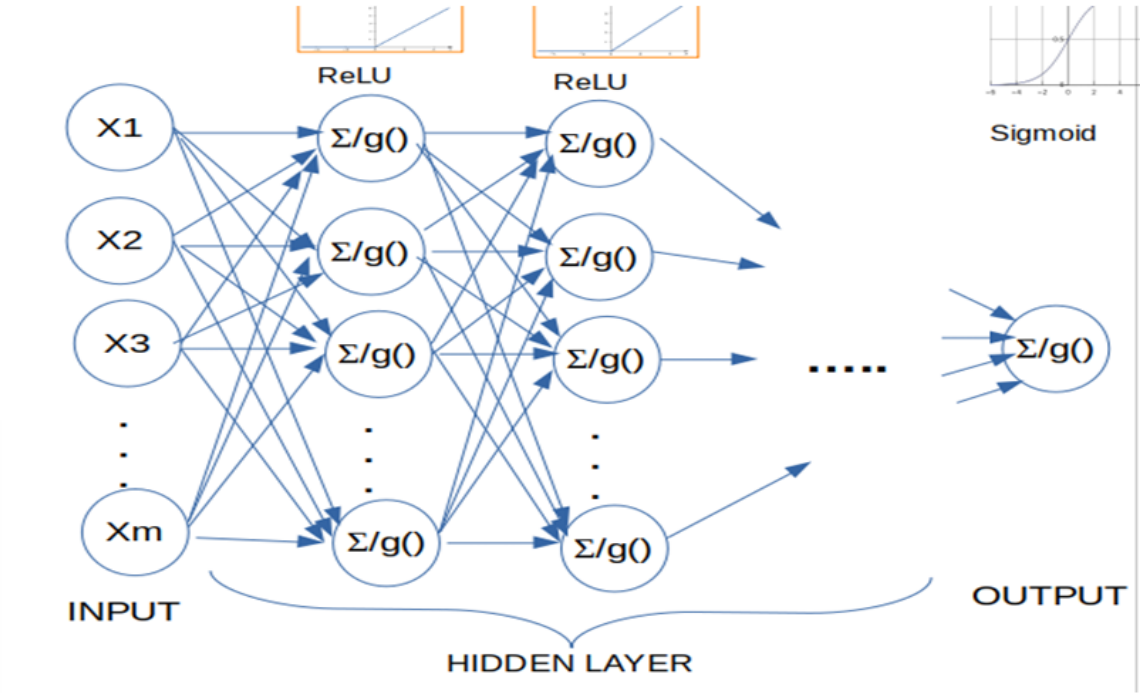


Fig. 3.10 Optimal Neural Network Model

model. Specifically, Fig. 3.11[11] gives the algorithm on training the underlying Classifier. In this algorithm,

Step 1 : initializes the weight sequence $(w_1, w_2, ..., w_{ofs})$ and the value of the offset b. Meanwhile, the appropriate activation functions are also selected. The ReLU function $(g_1(z))$ and Sigmoid function $(g_2(z))$ are selected as the activation functions for the hidden layer and the output layer respectively. The Sigmoid and ReLU are the commonly used activation functions. The Sigmoid function is commonly used in the last layer of the neural network. **Step 2** performs the forward propagation. Through taking the input values $A^{[l-1]}$, the weight W and the bias b, values of $Z^{[l]}$ and $A^{[l]}$ are calculated for each layer. In $A^{[l-1]}$,l indicates the number of layers of the neural network. At the first layer, $A^{[1]}$ is set as the input features X.

Step 3 calculates the loss function. The result of this function indicates the error between the predicted value and the target value. The target of training the neural network model is to reduce the value of this function. In order to alleviate the over-fitting problem in the training

Input: Training data set, Test data set and Feature vector.
Output: Optimal neural network Classifier for detecting phishing attacks.

(1) Initializes the network parameters and selects the *ReLU* function:
$g_1(z) = max(0, z)$, $g_2(z) = 1/(1 + e^{-Z})$.
(2) Performs the forward propagation:
$Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]}$, $A^{[l]} = g^{[l]}(Z^{[l]})$.
(3) Calculates the error of each layer (Adding the L2 regularization term can effectively alleviate the over-fitting problem):
$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} log\left(h_\theta(x^{(i)})\right)_k + (1 - y_k^{(i)}) log\left(1 - h_\theta(x^{(i)})\right)_k\right] + \frac{\lambda}{2m}||W||^2$.
(4) Performs the backward propagation:
$dZ^{[l]} = dA^{[l]} * g^{[l]'}(Z^{[l]})$, $dW^{[l]} = dZ^{[l]} \cdot A^{[l-1]}$, $db^{[l]} = dZ^{[l]}$, $da^{[l-1]} = W^{[l]^T} \cdot dZ^{[l]}$.
(5) Runs the gradient descent algorithm to update parameters of *w* and *b* until the convergence of the loss function:
$W = (1 - \alpha \cdot (\lambda/m))W - \alpha \cdot dW^{[l]}, b = b - db^{[l]}$.
(6) Adjust the parameters (threshold, learning rate, number of layers, hidden layer units) to get the optimal neural network model.

Fig. 3.11 Algorithm of Optimal Neural Network Classifier Training

process,the regularization item L2 is added to the loss function. In this step, m is the sample size; λ is the regularization parameter; $h_\Theta\left(x^{(i)}\right)$ is the prediction value and $y_k^{(i)}$ is the actual value.

Step4 performs the backward propagation. In this step,it is necessary to calculate the gradient values (dA^l, dW^l, anddb^l) for parameters A^l, W^l and b^l at first. Then, these gradient values are used to perform the backward propagation until the first layer is reached.

Step5 runs the gradient descent algorithm to minimize the value of the loss function. In order to minimize this value, Step2-Step4 are repeatedly executed until the loss function converges to a small value and the training result does not have an over-fitting. In this step, α is the learning rate.

Through repeatedly executing Step 2 - Step 5, the corresponding parameters (threshold, learning rate, layers and hidden units) can be continuously adjusted to obtain the optimal neural network Classifier.

3.4 Complete OFS-NN model

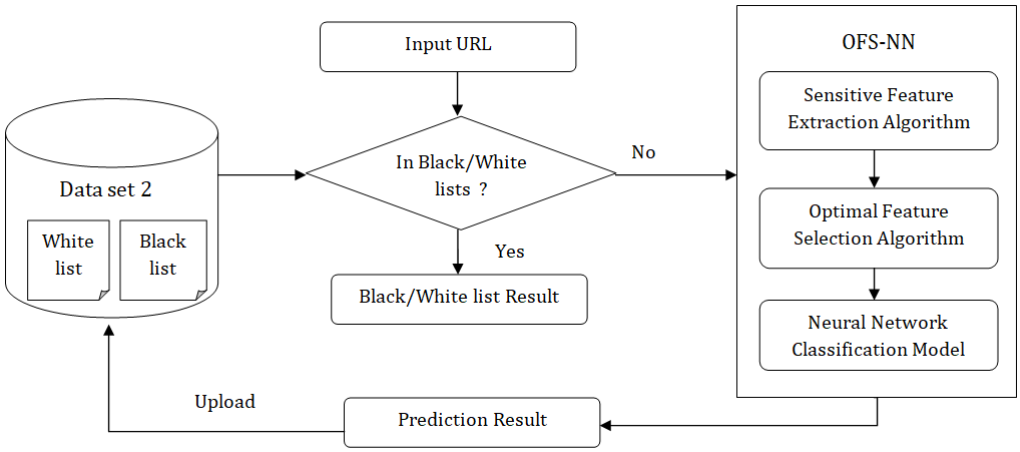


Fig. 3.12 Work Flow of Complete OFS-NN Model and Phishing Websites detection.

Input: F_v: Feature vector; ω: Weights of NN; b: Bias of NN; L: Layers number of NN.
Output: Y: Phishing detection result.

```
(1)   if URL in Lists
(2)     URL in BlackList return -1;  //Legitimate
(3)     URL in WhiteList return 1;   //Phishing
(4)   else NN model ← LoadWeightsandBias(ω,b);
(5)     prediction[0] ← F_v;
(6)     for i in L do
(7)       Y[i − 1] ← reLU(NN model.ω × Y[i − 2] + b[i − 2]);
(8)       Y[i] ← Sigmoid(Y[i − 1]);
(9)     end for
(10)    Y ← prediction[L];
(11)    if Y < 0.5   return 1  //Phishing
(12)    else   return -1  //Legitimate
```

Fig. 3.13 Complete Phishing URL Detection Algorithm

Fig. 3.12 shows the workflow of detecting phishing attacks of the OFS-NN model. As shown in this figure, a module for the black and white list is first added to reduce the time cost of the entire OFS-NN model. The white list is based mainly on the data selected from the Alexa websites. The black list is based mainly on the data selected from the PhishTank websites. It is necessary to check at the beginning of this phase whether the Input URL is in our library.

Fig.3.13 [11] shows the algorithm of detecting phishing attacks of the OFS-NN model. If the input URL is not in the black and white lists, the qualified neural network classifier will be used to process this URL.First, feature extraction and selection algorithms will be used to produce optimum feature vector for this URL. Then finished URL with optimum responsive features is taken as the classifier's input. Parameters of weights and offsets are loaded into the neural network in the underlying classifier. For a given layer, the weight matrix w is multiplied by the ReLU function with the previous layer value. And the sum of the product of multiplication and the offset b form the output of the current layer. The output (Y) of the last layer is the end result of the whole classifier. As the algorithm shows in Fig. 3.11 [11], If Y<0.5, returns the "1" algorithm and marks this URL as a phishing one. Otherwise, the algorithm returns "1" which is classified as a legitimate one. Additionally, the detection results are stored for future use in the black or white lists.

3.5 Conclusion

The methods used for this work has been discussed above.A description about the feature extraction techniques,feature selection techniques and classification techniques has been given.The results and analysis will be given in the next chapter.

Chapter 4

RESULTS AND DISCUSSIONS

4.1 Introduction

In this chapter,the results of various classifications using different neural network model with various combinations of features has been discussed.

4.2 Implementation Details

Experiments of this section are performed on a computer with Intel i3 CPU (1.7 GHz), RAM (4 GB) and linux 18.4 operation system (64bit). Meanwhile, experimental programs are written by the Python programming language.

This work is implemented in ANACONDA platform (Jupyter editor) Using the python language.In additions,tensorflow,sklearn,pandas,numpy,matplotlib etc package have been used to data analysis and classifies the phishing URLs. We have added multilayer perceptrons neural network,sequential neural network model. It uses ReLU,Sigmoid activation function.

In addition, We have used scikit-learn for confusion matrix, precision, accuracy, recall, F1-score to analysing the performance and comparing the performances with others.

4.3 Experimental Results

4.3.1 Description of Tested Datasets :

Two data sets are used to assess the performance of the proposed OFS-NN model, as listed in Table 3.1. Data Set 1 consists of 11055 samples with 55.69% legal URLs, and 44.31% phishing URLs. Accordingly, there are 1926 phishing websites and 1867 legal websites where F1 has the value of 1; there are 2972 phishing websites and 4290 legal websites where F1 has the value of 1. In the meantime, 70% of the total 11055 samples are used to train the Classifier, and 30% of the samples are tested to evaluate the OFS-NN model performance.As a matter of fact, a small number of training samples will trap the ultimate classifier into under-fitting problems and weak generalization ability. On the contrary, when all samples of the data set are used for training, the classifier will fall into problems of over-fitting and poor classification result.

Data set 2 (composed of 2117 samples with 47.14% phishing URLs 52.26% legitimate URLs) is used to evaluate the overall performance between our OFS-NN model and some existing phishing detection models.

Table 4.1 lists the distribution of data objects of Data set 1. For a given feature "Frequency" gives the number of data objects belonging to each value of this feature. As there are 11055 instances in Data set 1, the total number of data objects belonging to all values of a feature is 11055. As a example,for feature F1 there are 3793 objects where F1 has the value of 1 and 7262 objects where F1 has the value of 1 (shown in Table 4.1). Fig. 4.1 visualizes

Table 4.1 Distributions of sample in Data set 1

Category	No.	Values	Frequency		
			-1	0	1
Address bar Features	F1	(-1,1)	3793	–	7262
	F2	(-1,0,1)	8960	135	1960
	F3	(-1,1)	1444	–	9611
	F4	(-1,1)	1655	–	9400
	F5	(-1,1)	1429	–	9626
	F6	(-1,1)	9590	–	1465
	F7	(-1,0,1)	3363	3622	4070
	F8	(-1,0,1)	3557	1167	6331
	F9	(-1,1)	7389	–	3666
	F10	(-1,1)	2053	–	9002
	F11	(-1,1)	1502	–	9553
	F12	(-1,1)	1796	–	9259
Abnormal Features	F13	(-1,1)	4495	–	6560
	F14	(-1,0,1)	3282	5337	2436
	F15	(-1,0,1)	3956	4449	2650
	F16	(-1,0,1)	8440	761	1356
	F17	(-1,1)	2014	–	9041
	F18	(-1,1)	1629	–	9426
HTML and JavaScript Features	F19	(-1,0,1)	0	9776	1279
	F20	(-1,1)	1315	–	9740
	F21	(-1,1)	476	–	10579
	F22	(-1,1)	3137	–	8918
	F23	(-1,1)	1012	–	10043
Domain Features	F24	(-1,1)	5189	–	5866
	F25	(-1,1)	3443	–	7612
	F26	(-1,0,1)	2655	2659	5741
	F27	(-1,1)	8201	–	2854
	F28	(-1,1)	1539	–	9516
	F29	(-1,0,1)	548	6156	4351
	F30	(-1,1)	1550	–	9505

the distribution of legitimate and phishing websites .

4.3.2 Metrics used in Evaluation :

The confusion matrix is used for assessing OFS-NN model performance. The Fig. 4.2 [11], the matrix consists of four indexes: TP (true positive), TN (true negative), FP (fake positive),

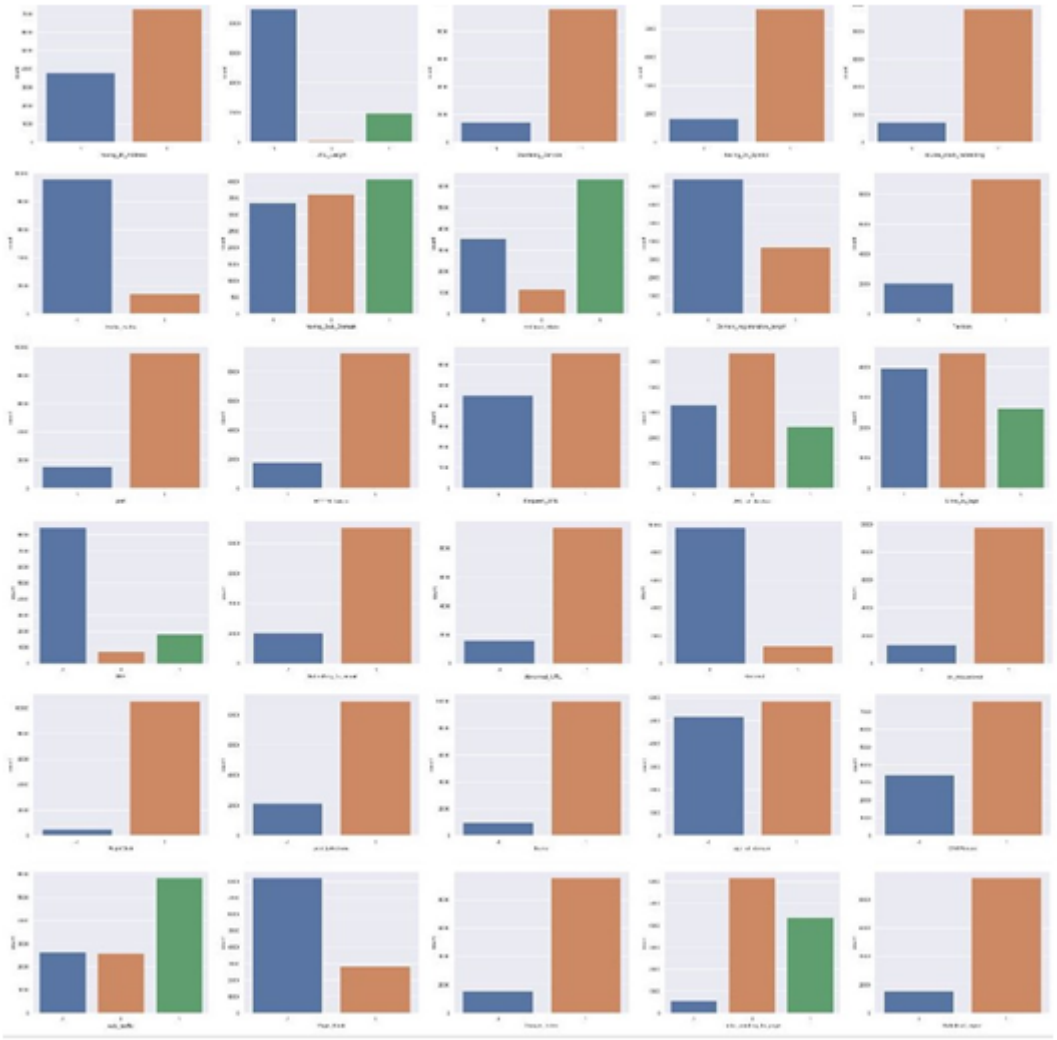


Fig. 4.1 Visualization Distribution of Data set 1

Actual value / Predict value	Phishing pages	Legitimate pages
Predicted as phishing	TP	FP
Predicted as legal	FN	TN

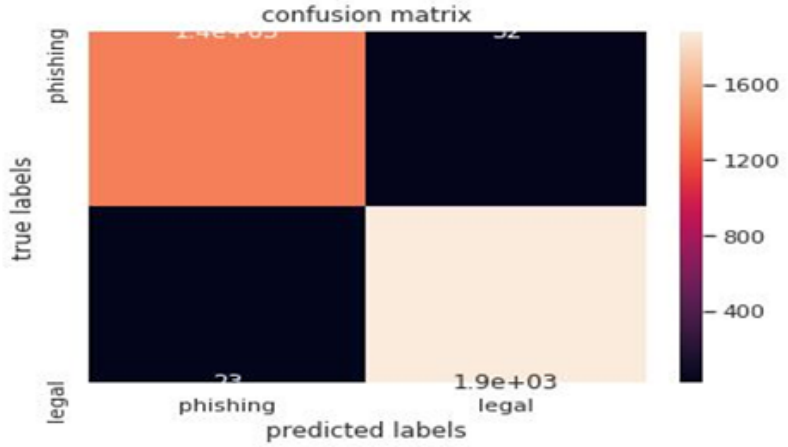
Fig. 4.2 The Structure of Confusion Matrix

and FN (false negative). The phishing websites are seen as positive samples in this paper; while the legitimate websites are taken as negative samples.By this designation TP is the number of phishing websites correctly predicted as phishing websites; TN is the number of

legal websites that are correctly predicted as legal ones; FP is the number of legal websites that are wrongly predicted as phishing ones; FN is the number of phishing websites that are wrongly predicted as legal ones. The Fig. 4.3 (A), the matrix is the confusion matrix for

Actual value / Predict value	Phishing pages	Legitimate pages
Predicted as phishing	1382	32
Predicted as legal	23	1879

(A)



(B)

Fig. 4.3 The Confusion Matrix

Optimal training Model and Fig. 4.3 (B) is visualization of confusion matrix in the respect of Data Set 1.

The accuracy rate is defined as the ratio of the number of correctly predicted URLs (TP + TN) to the number of all instance.

$$Accuracy = (TP+TN)/(TP+TN+FP+FN) \tag{4.1}$$

The precision rate is defined as the ratio of the number of phishing websites that are correctly predicted to the number of all the predicted phishing websites (TP+FP).

$$Precision = TP/(TP+FP) \tag{4.2}$$

The recall rate is defined as the ratio of the number of phishing websites that are correctly predicted to the number of all phishing URLs (TP+FN).

$$Recall = TP/(TP+FN)) \tag{4.3}$$

$F1-score$ is defined as the harmonic average of the recall rate and the precision rate.

$$F1-score = 2 \times (Precision \times Recall)/(Precision + Recall) \tag{4.4}$$

Among the four metrics defined by equation (4.1) - equation (4.4), the most important one is the accuracy as it reflects the overall performance of a classifier. The F1-score is a comprehensive judgment metric. The value of the F1-score is in the interval of [0, 1]. In this interval, the bigger value of F1-score,the better performance of the underlying classifier.

4.3.3 Performance Evaluation :

In this subsection the performance evaluation consists of three parts, the performance of the new specified FVV index, the performance of the optimum responsive feature selection method and the performance of the entire OFS-NN.

Performance of FVV :

"Information entropy" is a common metric used to measure a data set 's purity. Assuming the proportion of the sample i^{th} in data set D is $P(xi)$, the entropy of D is described below.

$$E(D) = \sum_{i=1}^{n} P(x_i) \cdot \log_2 P(x_i) \tag{4.5}$$

Information gain (Gain) is a selection metric features to evaluate a classification algorithm 's efficiency. In general , the greater the Benefit value, the better the classification algorithm's selectivity efficiency.

If an attribute a (a is supposed to have V possible values) is used to divide data set D,then V branch nodes are generated. Where, the V branch nodes contain all the values in data set D on attribute a. As in equation 4.5, the Gain indicator is calculated by the difference between the entropy of the sample set to be classified and the conditional entropy of a selected feature.

$$Gain(D,a) = E(D) - \sum_{v=1}^{V} |D_v|/|D| \cdot E(D^v) \tag{4.6}$$

In the formula, V represents the value range of the attribute a; D^v represents that all samples of sample data set D in the v^{th} branch node take values on attribute a;$|D_v|/|D|$ is the weight of the branch node.

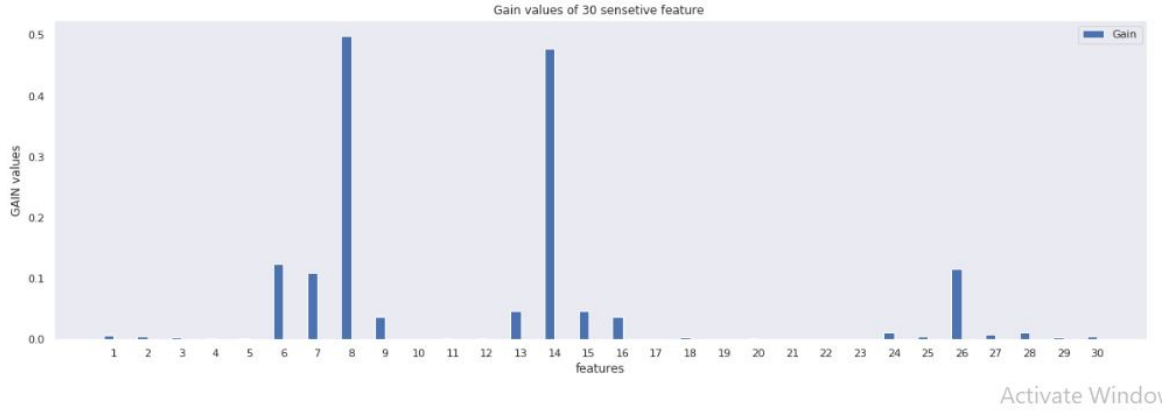


Fig. 4.4 "Information gain" (Gain) for 30 features of Data set 1

Fig. 4.4 shows the values of the Gain indicator corresponding to the 30 sensitive features in Data set 1. This paper compares the performance of the "FVV index" and the "Information gain (Gain)" on the optimal features selection. In the case of the same average threshold for FVV index is 0.257 and threshold for Gain is 0.034.

Performance of Optimal feature selection :

This part evaluates the impacts of different categories of features on the performance of the phishing websites detection. The experimental results in Table7 have shown that the highest category (Address bar relevant features) achieves 90.75% accuracy, while the worst category (HTML and JavaScript relevant features) only has 56.43% accuracy. The last line of

Table 4.2 Performance of Phishing detection under 4 categories of features and optimal features

Features Category	Accuracy	Precision	Recall	F1_score
Address bar	0.9079	0.9284	0.9036	0.9158
Abnormal	0.8749	0.8378	0.9619	0.8955
HTML/JavaScript	0.5643	0.5614	0.9943	0.7177
Domain	0.7480	0.7439	0.8229	0.7844
Optimal	0.9834	0.9832	0.9879	0.9856

this table also gives the performance of the phishing websites detection by utilizing optimal feature (selection method (selecting optimal features from all the four categories of features). Through using the optimal feature selection method, the accuracy is improved to 98.34%. This experiment shows that the phishing websites detection performs poorly by features extraction of a single category. Therefore,for the better performance,it is necessary to select optimal sensitive features from all categories.

Table 4.3 lists the results of the average time cost comparisons between “NN” (our model without incorporating the optimal feature selection method) and “OFS-NN” (our neural network model that incorporates the optimal feature selection method).

Table 4.3 Time cost evaluation of the Optimal Feature Selection and comparison of OFS-NN and Normal NN

Items	OFS-NN	NN
Feature Selection Time	369.46 s	–
Training time	87.113 s	94.292 s
Detection Time	0.086 s	0.125 s

4.3.4 Performance of the OFS-NN :

This subsection evaluates the performance of the OFS-NN model under different scales of the tested data set at first. In this experiment, the proposed model is established as a function for different scale of Data set 1. It can be seen from Table9,as the number of samples increases , the performance of the OFS-NN model is improved. In the case of small number of samples, the errors occur frequently. As the number of samples increasing, the occurrences of errors become smaller. When the changing of errors occurrences closed, the trained neural network structure is the optimal one.This structure has no problem of over-fitting or under-fitting.

Table 4.4 Performance of OFS-NN under different scales of Data set 1

Data set	Accuracy	Precision	Recall	F1_score
10%	0.9837	0.9852	0.9853	0.9852
20%	0.9842	0.9840	0.9880	0.9860
30%	0.9834	09800	0.9901	0.9850
40%	0.9810	0.9794	0.9866	0.9830
50%	0.9833	0.9803	0.9899	0.9851
60%	0.9829	0.9816	0.9882	0.9849
70%	0.9822	0.9801	0.9881	0.9841
80%	0.9832	0.9809	0.9889	0.9849
90%	0.9832	0.9809	0.9889	0.9849
100%	0.9846	0.9840	0.9888	0.9864

The receiver operating characteristic (ROC) is a curve in which the false positive rate (FPR) is plotted on the abscissa and the true positive rate (TPR) is plotted on the ordinate. The ROC curve formed by the (0, 0) and (1, 1) connections represents a random classifier. If the curve drawn by the classification model is above the random classifier curve, it means that

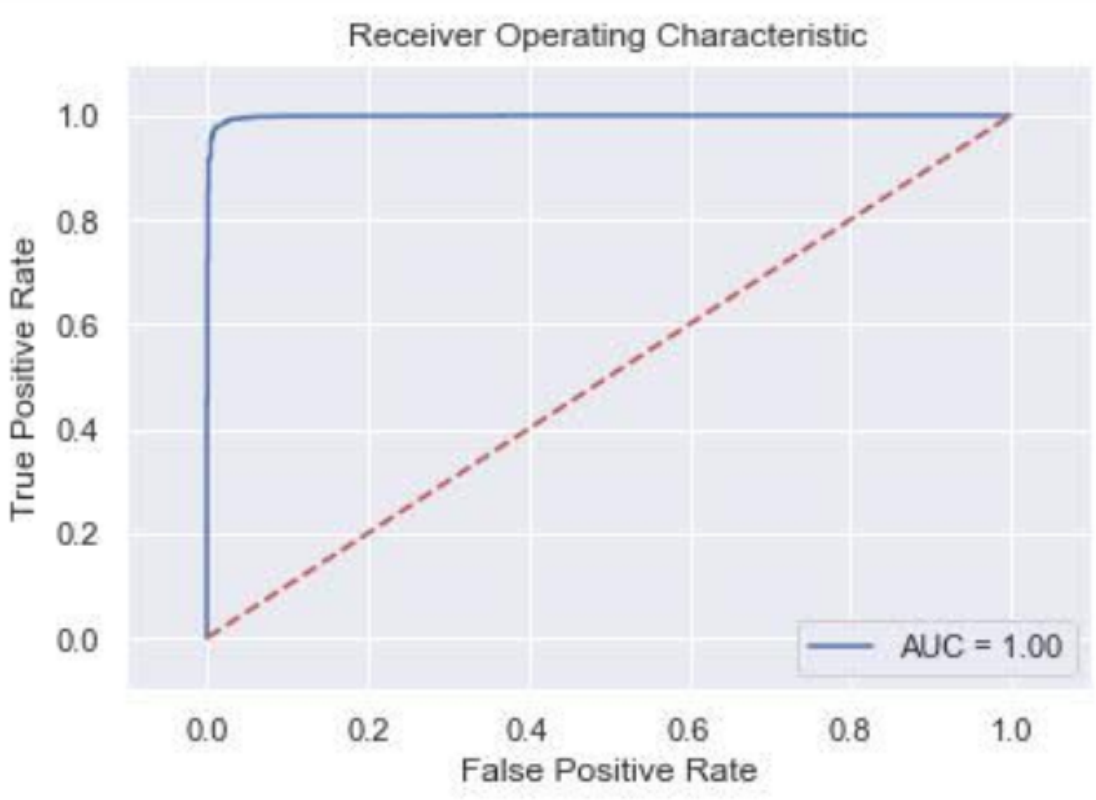


Fig. 4.5 "Information gain" (Gain) for 30 features of Data set 1

the classifier performs better. The area under ROC curve (AUC) value is a standard used to measure the quality of a classification model. The AUC value is the area covered by the ROC curve. Obviously, the larger the AUC value, the better performance of the classification effect of the tested model. If the AUC value is 1,the target classifier is the perfectone. However,in most cases of prediction, there is no perfect classifier. As a matter of fact,if the AUC value is 0.5, the prediction result of the tested classification model follows the probability of machine guessing and this model has no predictive value. When the AUC value is in the interval of (0.5, 1), the performance tested classification model is better than random guess and this model has predictive value.

Fig. 4.5 shows the ROC curve of the OFS-NN model. From this figure, it can be seen that, the AUG value of the OFS-NN is 1.00.This means the OFS-NN model has good performance for classification.

Table 4.5 Performance Comparison between the OFS-NN and some existing Phishing Detection Models.

Models	Test data sets		Phishing Ratio	Accuracy	Precision	Recall	F1_Score
	Legitimate	Phishing					
Cantina	2100	19	0.89%	0.969	0.212	0.89	0.342
Cantina+	1868	940	26.40%	0.955	0.964	0.964	0.964
Jain	405	1120	73.44%	0.894	0.993	0.861	0.922
Xiang	7906	3543	30.94%	0.955	0.957	0.900	0.928
Varshney	2500	500	16.66%	0.936	0.723	0.995	0.837
Kausar	71	89	55.62%	0.875	0.888	0.887	0.887
PhishStrom	48009	48009	50.00%	0.949	0.984	0.913	0.947
PhishShield	250	1600	86.48%	0.966	0.999	0.971	0.985
Off-the-Hook	200000	2000	0.99%	0.999	0.975	0.951	0.963
OFS-NN	6156	4899	44.31%	0.983	0.982	0.987	0.985

Lastly, the overall performance of the OFS-NN model is compared with some existing phishing websites detection models. In Table 4.5,the data source of Cantina [19] is selected from “Alexa/PhishTank”; the data source of CANTINA+[20] is selected from “Alexa/3Sharp”; the data source of Jain and Gupta [26] is selected from “Alexa”; the data source of Xiang and Hong [27] is selected from “PhishTank/Alexa/3Sharp/ Yahoo”; the data source of Varshney et al [28] is selected from “Alexa”; the data source of Kausar [**?**] is selected from “PhishTank”; the data source of PhishStorm [22] is selected from “DMOZ”; the data source of PhishShield [23] is selected from “PhishTank”;the data source of Off-the-Hook [25]is selected from “PhishTank/Intel Security”; and the data source of our OFS-NN model is selected from “Alexa/ PhishTank”.

From Table 4.5, It can be seen that the OFS-NN model has a higher accuracy than the other models except for the Off-the Hook model. The Off-the-Hook model has the highest accuracy but the recall rate is lower than our OFS-NN model. Meanwhile the OFS-NN model’s time cost is much lower than the Off-the-Hook model’s. As a matter of fact, the Off-the-Hook model collects 210 features (without optimal feature selection) from URLs and related websites. Too many features however significantly degrade this model’s performance.

4.4 Conclusion and Future Work

Here, it describes an efficient phishing attacks detection model OFS-NN based on the optimal set of features and the neural network.It has proven highly accurate and has low false negatives and lower false positives rate.In this model, we have shown that the FVV index is used to calculate the value of the feature and to determine the effect on phishing detection of sensitive features.Then the optimal feature selection algorithm is built to construct the optimal feature vector for training the underlying neural network classifier, based on the new FVV index. This algorithm could manage the issues of large number of phishing sensitive features and continuous feature changes properly.Consequently it can mitigate the neural network classifier's over-fitting problem.Finally, the optimal neural network classifier is equipped through repeated experimental experiments to identify phishing attacks.

As the continuous shift of phishing attacks' sensitive features, more features need to be collected in the future in order to perform optimum selection of features.Future development of our approach can include applying deep learning to web page code extraction and web page text extraction functionality. However, we expect to incorporate our web browser embedding approach into a plugin.

Chapter 5

BIBLIOGRAPHY

References

[1] UCI Machine Learning Repository. Center for Machine Learning and Intelligent Systems. Accessed: 2019. [Online]. Available: http://archive.ics.uci.edu/ml/index.php

[2] Phishtank. Out of the Net, Into the Tank. Accessed: 2019. [Online]. Avail- able: https://www.phishtank.com/

[3] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang, "An empirical analysis of phishing blacklists," in Proc. 6th Conf. Email Anti-Spam (CEAS), Mountain View, CA, USA, Jul. 2009, pp. 1–20.

[4] GitHub. Implementation for the Usage of Google Safe Browsing APIs(v4).Accessed:2019.[Online].Available:https://github.com/google/ safebrowsing.

[5] J. Kang and D. Lee, "Advanced white list approach for preventing access to phishing sites," in Proc. Int. Conf. Converg. Inf. Technol. (ICCIT), Gyeongju, South Korea, Nov. 2007, pp. 491–496.

[6] M. Sharifi and S. Siadati, “A phishing sites blacklist generator,” in Proc. 6th ACS/IEEE Int. Conf. Comput. Syst. Appl. (AICCSA), Doha, Qatar, Mar./Apr. 2008, pp. 840–843.

[7] X. Han, N. Kheir, and D. Balzarotti, “PhishEye: Live monitoring of sandboxed phishing kits,” in Proc. 23rd ACM Conf. Comput. Commun. Secur. (CCS), Vienna, Austria, Oct. 2016, pp. 1402–1413.

[8] L.-H. Lee, K.-C. Lee, H.-H. Chen, and Y.-H. Tseng, “POSTER: Proactivebblacklist update for anti-phishing,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS), Scottsdale, AZ, USA, Nov. 2014, pp. 1448–1450.

[9] A.Aleroud and L. Zhou, “Phishing environments, techniques, and countermeasures: A survey,” Comput. Secur., vol. 68, pp. 160–196, Jul. 2017.

[10] L. Shi, D. Lin, C. V. Fang, and Y. Zhai, “A hybrid learning from multibehavior for malicious domain detection on enterprise network,” in Proc. IEEE 15th Int. Conf. Data Mining Workshops (ICDMW), Atlantic City, NJ, USA, Nov. 2015, pp. 987–996.

[11] OFS-NN: An Effective Phishing Websites Detection Model Based on Optimal Feature Selection and Neural Network E Zhu, Y Chen, C Ye, X Li, F Liu - IEEE Access, 2019 - ieeexplore.ieee.org

[12] E.-S. M. El-Alfy, “Detection of phishing websites based on probabilistic neural networks and K-medoids clustering,” Comput. J., vol. 60, no. 12, pp. 1745–1759, 2017.

[13] C. Huang, S. Hao, L. Invernizzi, Y. Fang, C. Kruegel, and G. Vigna, “Gossip: Automatically identifying malicious domains from mailing list discussions,” in Proc. ACM Asia Conf. Comput. Commun. Secur. (ASIA CCS), Abu Dhabi, United Arab Emirates, Apr. 2017, pp. 494–505

[14] F. Vanhoenshoven, G. Nápoles, R. Falcon, K. Vanhoof, and M. Köppen, "Detecting malicious URLs using machine learning techniques," in Proc. IEEE Symp. Ser. Comput. Intell. (SSCI), Dec. 2016, pp. 1–8.

[15] L. Wu, X. Du, and J. Wu, "Effective defense schemes for phishing attacks onmobilecomputingplatforms,"IEEETrans.Veh.Technol.,vol.65,no.8, pp. 6678–6691, Aug. 2016.

[16] R. Gowtham and I. Krishnamurthi, "A comprehensive and efficacious architecture for detecting phishing webpages," Comput. Secur., vol. 40, pp. 23–37, 2014.

[17] M. Khonji, Y. Iraqi, and A. Jones, "Phishing detection: A literature survey," IEEE Commun. Surveys Tuts., vol. 15, no. 4, pp. 2091–2121, 4th Quart., 2013.

[18] SOUTHEAST ASIA – THE HUB FOR PHISHING ATTACKS [Online]. available:https://hooksecurity.co/blog/southeast-asia-the-hub-for-phishing-attacks/

[19] Y.Zhang,J.I.Hong,andL.F.Cranor,"Cantina:Acontent-basedapproach to detecting phishing Web sites," in Proc. 16th Int. Conf. World Wide Web (WWW), Banff, AB, Canada, May 2007, pp. 639–648

[20] X.Li,G.Geng,Z.Yan,Y.Chen,andX.Lee,"Phishingdetectionbasedon newly registered domains," in Proc. IEEE Int. Conf. Big Data (BigData), Washington DC, USA, Dec. 2016, pp. 3685–3692.

[21] L. Anh, T. Nguyen, B. L. To, H. K. Nguyen, and M. H. Nguyen, "Anefficientapproachforphishingdetectionusingsinglelayerneuralnetwork,"inProc.Int.Conf.Adv.Technol.Commun.(ATC),Hanoi,Vietnam, Oct. 2014, pp. 435–440.

[22] S. Marchal, J. FranÃğois, R. State, and T. Engel, “PhishStorm: Detecting phishing with streaming analytics,” IEEE Trans. Netw. Service Manage., vol. 11, no. 4, pp. 458–471, Dec. 2014.

[23] R. S. Rao and S. T. Ali, “PhishShield: A desktop application to detect phishing Webpages through heuristic approach,” Procedia Comput. Sci., vol. 54, pp. 147–156, Aug. 2015.

[24] G. Sonowal and K. S. Kuppusamy, “PhiDMA—A phishing detection model with multi-filter approach,” J. King Saud Univ.–Comput. Inf. Sci., to be published. doi: 10.1016/j.jksuci.2017.07.005.

[25] S. Marchal, G. Armano, T. Gröndahl, K. Saari, N. Singh, and N. Asokan, “Off-the-hook: An efficient and usable client-side phishing prevention application,” IEEE Trans. Comput., vol. 66, no. 10, pp. 1717–1733, Oct. 2017.

[26] A. K. Jain and B. B. Gupta, “A novel approach to protect against phishing attacks at client side using auto-updated white-list,” EURASIP J. Inf. Secur., vol. 2016, no. 1, Dec. 2016, Art. no. 9.

[27] G. Xiang and J. I. Hong, “A hybrid phish detection approach by identity discoveryand-keywordsretrieval,”inProc.18thInt.Conf.WorldWideWeb (WWW), Madrid, Spain, Apr. 2009, pp. 571–580.

[28] G.Varshney,M.Misra,andP.K.Atrey,“Aphishdetectorusinglightweight search features,” Comput. Secur., vol. 62, pp. 213–228, Sep. 2016.

[29] F. Kausar, B. Al-Otaibi, A. Al-Qadi, and N. Al-Dossari, “Hybrid client sidephishing-websitesdetectionapproach,”Int.J.Adv.Comput.Sci.Appl., vol. 5, no. 7, pp. 132–140, 2014.

Screenshots of Results

```
In [5]: data
```

Out[5]:

	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSLfinal_State	Dor
0	-1	1	1	1	-1	-1	-1	-1	
1	1	1	1	1	1	-1	0	1	
2	1	0	1	1	1	-1	-1	-1	
3	1	0	1	1	1	-1	-1	-1	
4	1	0	-1	1	1	-1	1	1	
...	...	...	...	...	...	...	...	...	
11050	1	-1	1	-1	1	1	1	1	
11051	-1	1	1	-1	-1	-1	1	-1	
11052	1	-1	1	1	1	-1	1	-1	
11053	-1	-1	1	1	1	-1	-1	-1	
11054	-1	-1	1	1	1	-1	-1	-1	

11055 rows × 31 columns

Fig. 5.1 Data set

```
In [7]: data.describe()
```

Out[7]:

	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSLfinal_State	Dor
count	11055.000000	11055.000000	11055.000000	11055.000000	11055.000000	11055.000000	11055.000000	11055.000000	
mean	0.313795	-0.633198	0.738761	0.700588	0.741474	-0.734962	0.063953	0.250927	
std	0.949534	0.766095	0.673998	0.713598	0.671011	0.678139	0.817518	0.911892	
min	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	
25%	-1.000000	-1.000000	1.000000	1.000000	1.000000	-1.000000	-1.000000	-1.000000	
50%	1.000000	-1.000000	1.000000	1.000000	1.000000	-1.000000	0.000000	1.000000	
75%	1.000000	-1.000000	1.000000	1.000000	1.000000	-1.000000	1.000000	1.000000	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

8 rows × 31 columns

```
In [8]: #from matplotlib import pyplot as plt
        #data.plot(subplots=True,layout=(4,5))
        #data.plot.bar(subplots=True,grid=True)
        #plt.show()
```

Fig. 5.2 Data set Description

we have total 30 features ,Now we apply FVV for optimal feature selection

```
In [10]: data1 = np.loadtxt('phishing_Dataset.csv', delimiter=',',)

         FVV=[]
         for k in range(0,30):
             ncount=0
             pcount=0
             for i in range(0,11055):
                 #print(data1[i,0],data1[i,30])
                 if (data1[i,k]==1 and data1[i,30]==1):
                     #print("ok")
                     pcount=pcount+1
             for i in range(0,11055):
                 if (data1[i,k]==-1 and data1[i,30]==-1):
                     ncount=ncount+1
             FVV.append((pcount+ncount)/11055)

         thresold=sum(FVV)/(2*30)

In [11]: print("Threold=",thresold)

         Threold= 0.2565973164480627

In [12]: from matplotlib import pyplot as plt
         #X = list(range(1,31))
         X=np.arange(1,31)
         fig ,ax=plt.subplots(figsize=(20,6))
         width=0.30
```


Fig. 5.3 FVV Index and Threshold calculation

```
fig ,ax=plt.subplots(figsize=(20,6))
width=0.30
p=ax.bar(X,FVV,width)

ax.set_xticks(X + width/2)
ax.set_xticklabels(X)
ax.set_title("FVV values of 30 sensetive feature ")
plt.xlabel("features")
plt.ylabel("FVV values")
plt.legend(["FVV"])
plt.grid()
plt.show()
```

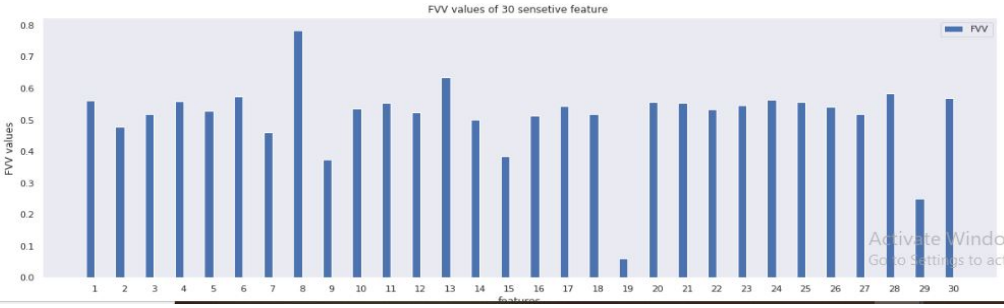
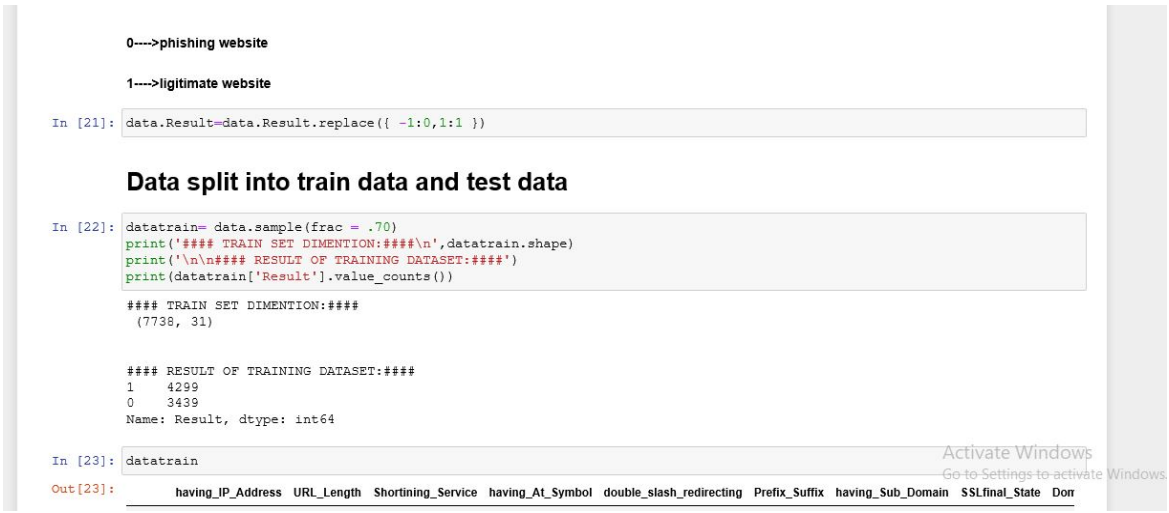


Fig. 5.4 FVV visualization

```
In [13]: Fofs={}
         nonFofs={}
         for k in range(0,30):
             if (FVV[k]>=thresold):
                 Fofs[k+1]=Feature[k+1]
             else:
                 nonFofs[k+1]=Feature[k+1]

In [14]: print("optimal feature are")
         print(Fofs)

         optimal feature are
         {1: 'having_IP_Address', 2: 'URL_Length', 3: 'Shortining_Service', 4: 'having_At_Symbol', 5: 'double_slash_redirectin
         g', 6: 'Prefix_Suffix', 7: 'having_Sub_Domain', 8: 'SSLfinal_State', 9: 'Domain_registeration_length', 10: 'Favicon',
         11: 'port', 12: 'HTTPS_token', 13: 'Request_URL', 14: 'URL_of_Anchor', 15: 'Links_in_tags', 16: 'SFH', 17: 'Submittin
         g_to_email', 18: 'Abnormal_URL', 20: 'on_mouseover', 21: 'RightClick', 22: 'popUpWidnow', 23: 'Iframe', 24: 'age_of_d
         omain', 25: 'DNSRecord', 26: 'web_traffic', 27: 'Page_Rank', 28: 'Google_Index', 30: 'Statistical_report'}

In [15]: print("Non optimal feature are .....delete that")
         print(nonFofs)

         Non optimal feature are .....delete that
         {19: 'Redirect', 29: 'Links_pointing_to_page'}
```

DELETE THE NON OPTIMAL

Fig. 5.5 List of Optimal and Non-optimal features



0---->phishing website

1---->ligitimate website

In [21]:
```
data.Result=data.Result.replace({ -1:0,1:1 })
```

Data split into train data and test data

In [22]:
```
datatrain= data.sample(frac = .70)
print('#### TRAIN SET DIMENTION:####\n',datatrain.shape)
print('\n\n#### RESULT OF TRAINING DATASET:####')
print(datatrain['Result'].value_counts())
```

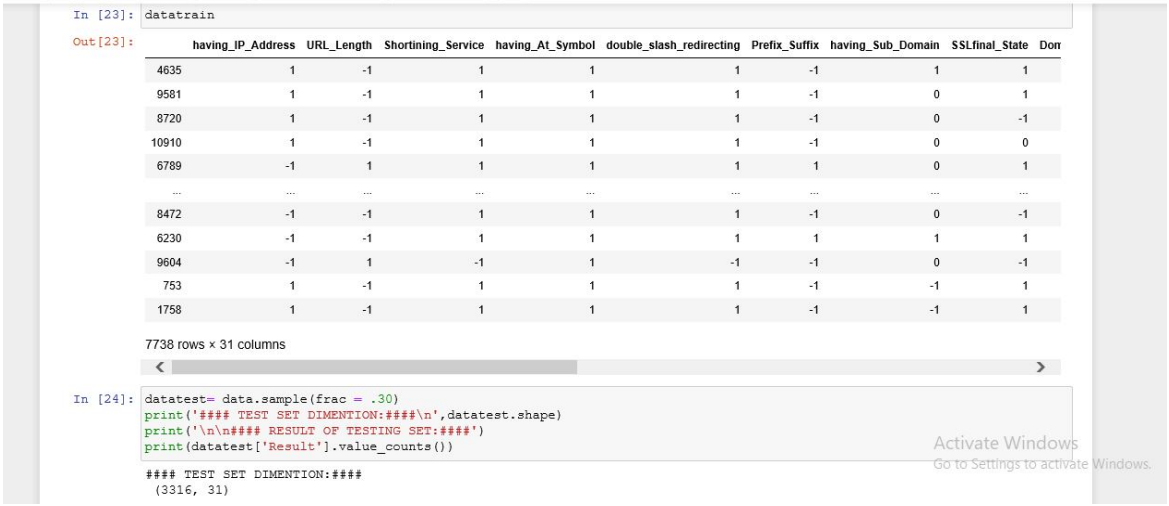
```
#### TRAIN SET DIMENTION:####
 (7738, 31)


#### RESULT OF TRAINING DATASET:####
1    4299
0    3439
Name: Result, dtype: int64
```

In [23]:
```
datatrain
```

Out[23]:

having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSLfinal_State	Dom

Fig. 5.6 Data split into training data and testing data



In [23]:
```
datatrain
```

Out[23]:

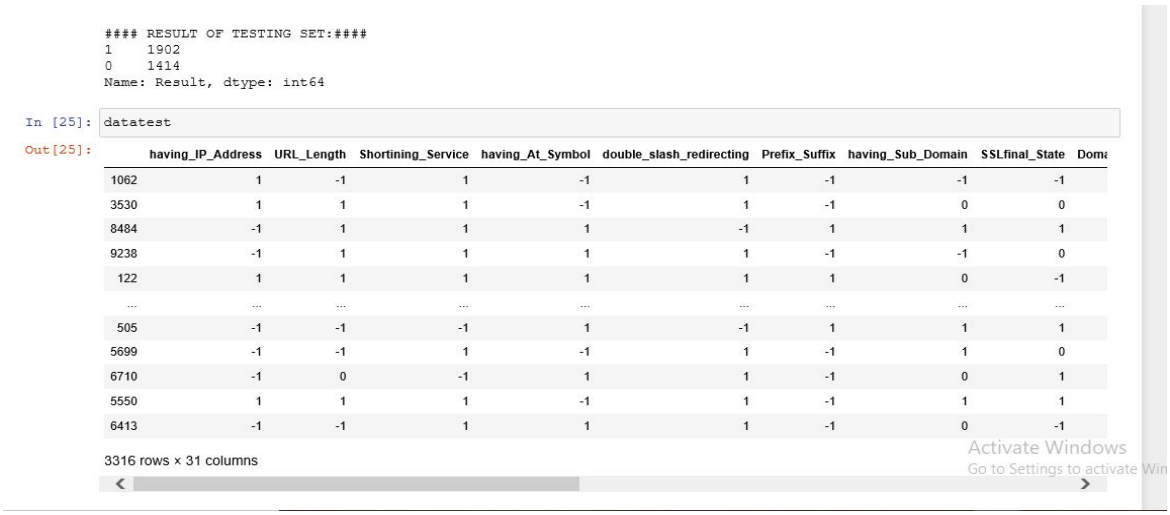
	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSLfinal_State	Dom
4635	1	-1	1	1	1	-1	1	1	
9581	1	-1	1	1	1	-1	0	1	
8720	1	-1	1	1	1	-1	0	-1	
10910	1	-1	1	1	1	-1	0	0	
6789	-1	1	1	1	1	1	0	1	
...	...	...	...	...	...	...	...	...	
8472	-1	-1	1	1	1	-1	0	-1	
6230	-1	-1	1	1	1	1	1	1	
9604	-1	1	-1	1	-1	-1	0	-1	
753	1	-1	1	1	1	-1	-1	1	
1758	1	-1	1	1	1	-1	-1	1	

7738 rows × 31 columns

In [24]:
```
datatest= data.sample(frac = .30)
print('#### TEST SET DIMENTION:####\n',datatest.shape)
print('\n\n#### RESULT OF TESTING SET:####')
print(datatest['Result'].value_counts())
```

```
#### TEST SET DIMENTION:####
 (3316, 31)
```


Fig. 5.7 Training Data set



```
#### RESULT OF TESTING SET:####
1    1902
0    1414
Name: Result, dtype: int64
```

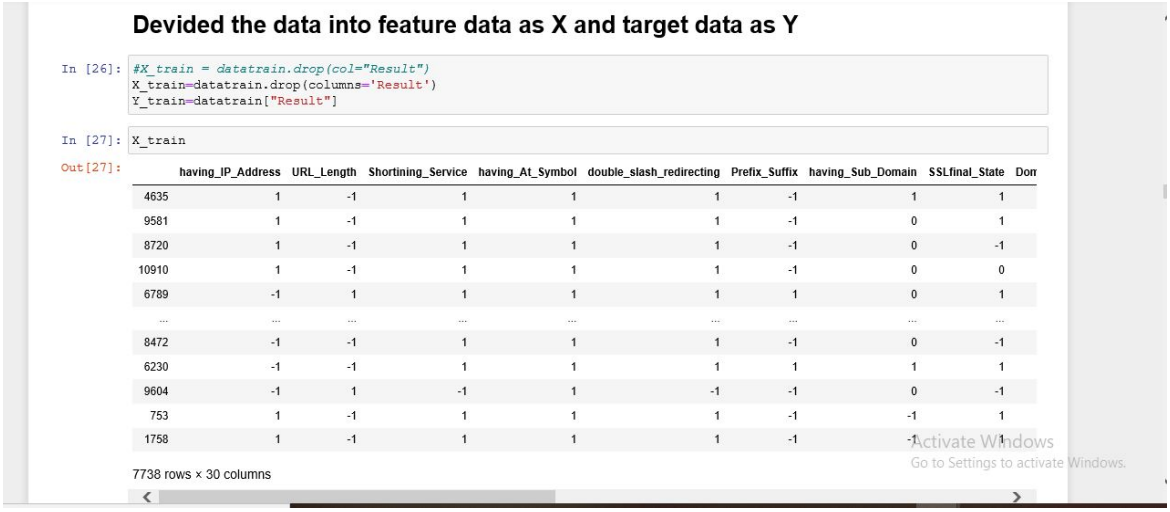
In [25]:
```
datatest
```

Out[25]:

	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSLfinal_State	Doma
1062	1	-1	1	-1	1	-1	-1	-1	
3530	1	1	1	-1	1	-1	0	0	
8484	-1	1	1	1	-1	1	1	1	
9238	-1	1	1	1	1	-1	-1	0	
122	1	1	1	1	1	1	0	-1	
...	...	...	...	...	...	...	...	...	
505	-1	-1	-1	1	-1	1	1	1	
5699	-1	-1	1	-1	1	-1	1	0	
6710	-1	0	-1	1	1	-1	0	1	
5550	1	1	1	-1	1	-1	1	1	
6413	-1	-1	1	1	1	-1	0	-1	

3316 rows × 31 columns

Fig. 5.8 Testing Data set



Devided the data into feature data as X and target data as Y

```
In [26]: #X_train = datatrain.drop(col="Result")
         X_train=datatrain.drop(columns='Result')
         Y_train=datatrain["Result"]

In [27]: X_train
```

Out[27]:

	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSLfinal_State	Dom
4635	1	-1	1	1	1	-1	1	1	
9581	1	-1	1	1	1	-1	0	1	
8720	1	-1	1	1	1	-1	0	-1	
10910	1	-1	1	1	1	-1	0	0	
6789	-1	1	1	1	1	1	0	1	
...	...	...	...	...	...	...	...	...	
8472	-1	-1	1	1	1	-1	0	-1	
6230	-1	-1	1	1	1	1	1	1	
9604	-1	1	-1	1	-1	-1	0	-1	
753	1	-1	1	1	1	-1	-1	1	
1758	1	-1	1	1	1	-1	-1	1	

7738 rows × 30 columns

Fig. 5.9 Dividing the training data into features as X and target data as Y

```
Iteration 14, loss = 0.20371427
Iteration 15, loss = 0.19696210
Iteration 16, loss = 0.19153725
Iteration 17, loss = 0.18703274
Iteration 18, loss = 0.18335903
Iteration 19, loss = 0.17979475
```

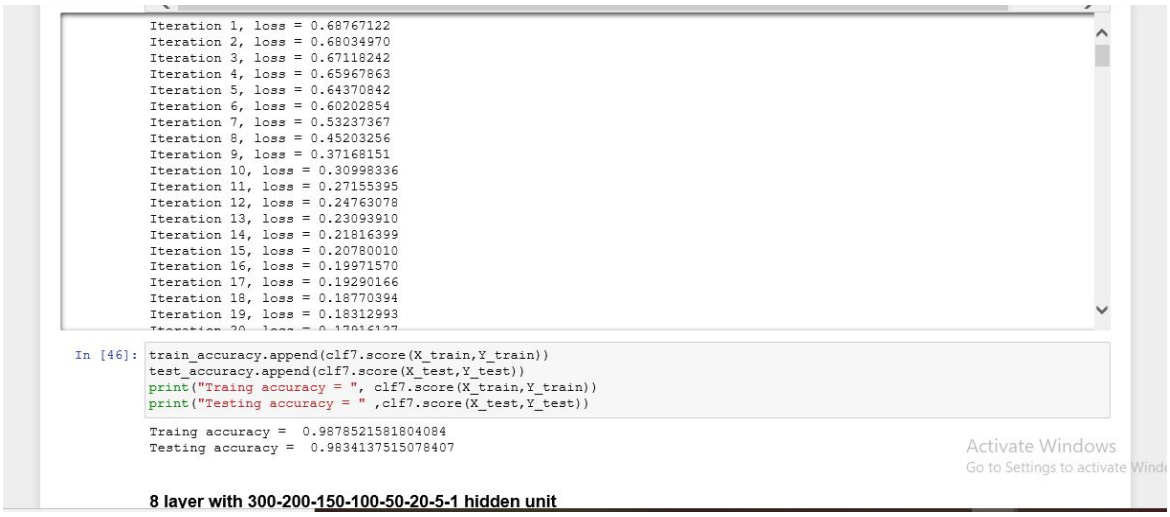
```
In [44]: train_accuracy.append(clf6.score(X_train,Y_train))
         test_accuracy.append(clf6.score(X_test,Y_test))
         print("Traing accuracy = ", clf6.score(X_train,Y_train))
         print("Testing accuracy = " ,clf6.score(X_test,Y_test))

         Traing accuracy =  0.9879813905401913
         Testing accuracy =  0.9819059107358263
```

7 layer with 200-150-100-50-20-5-1 hidden unit

Execution time using time() function for NN

```
In [45]: import time
         from sklearn.neural_network import MLPClassifier

         clf7 = MLPClassifier(solver='sgd', alpha=1e-5,hidden_layer_sizes=(200,150,100,50,20,5),verbose=100, max_iter=1000,randor
         t0=time.time()
         clf7.fit(X_train,Y_train)
         NN_training_time=round(time.time()-t0,3)
```


Fig. 5.10 Neural Network Model : Training



```
Iteration 1, loss = 0.68767122
Iteration 2, loss = 0.68034970
Iteration 3, loss = 0.67118242
Iteration 4, loss = 0.65967863
Iteration 5, loss = 0.64370842
Iteration 6, loss = 0.60202854
Iteration 7, loss = 0.53237367
Iteration 8, loss = 0.45203256
Iteration 9, loss = 0.37168151
Iteration 10, loss = 0.30998336
Iteration 11, loss = 0.27155395
Iteration 12, loss = 0.24763078
Iteration 13, loss = 0.23093910
Iteration 14, loss = 0.21816399
Iteration 15, loss = 0.20780010
Iteration 16, loss = 0.19971570
Iteration 17, loss = 0.19290166
Iteration 18, loss = 0.18770394
Iteration 19, loss = 0.18312993
```

```
In [46]: train_accuracy.append(clf7.score(X_train,Y_train))
         test_accuracy.append(clf7.score(X_test,Y_test))
         print("Traing accuracy = ", clf7.score(X_train,Y_train))
         print("Testing accuracy = " ,clf7.score(X_test,Y_test))

         Traing accuracy =  0.9878521581804084
         Testing accuracy =  0.9834137515078407
```


8 layer with 300-200-150-100-50-20-5-1 hidden unit

Fig. 5.11 Neural Network Model : Testing

so model 7 is the best model, save the model

```
In [57]: saved_model=pickle.dumps(clf7)

In [58]: #saved_model
         [coef.shape for coef in clf7.coefs_]

Out[58]: [(30, 200), (200, 150), (150, 100), (100, 50), (50, 20), (20, 5), (5, 1)]

In [59]: import time
         #load model and predict
         t0=time.time()
         clf_from_pickle=pickle.loads(saved_model)
         pred=clf_from_pickle.predict(X_test)
         NN_testing_time=round(time.time()-t0,3)

In [71]: import time
         #load model and predict
         t0=time.time()
         clf_from_pickle=pickle.loads(saved_model)
         pred1=clf_from_pickle.predict(X_test.head(1))
         NN_testing_time1=round(time.time()-t0,3)

In [60]: pred

Out[60]: array([0, 0, 1, ..., 1, 1, 0])

In [73]: import time
```


Fig. 5.12 Configuration of Neural Network model and saving the model

```
In [78]: import seaborn as sns
         import matplotlib.pyplot as plt
         ax=plt.subplot()
         sns.heatmap(matrix,annot=True,ax=ax)
         ax.set_xlabel("predicted labels")
         ax.set_ylabel("true labels")
         ax.set_title('confusion matrix')
         ax.xaxis.set_ticklabels(['phishing','legal'])
         ax.yaxis.set_ticklabels(['phishing','legal'])

Out[78]: [Text(0, 0.5, 'phishing'), Text(0, 1.5, 'legal')]
```

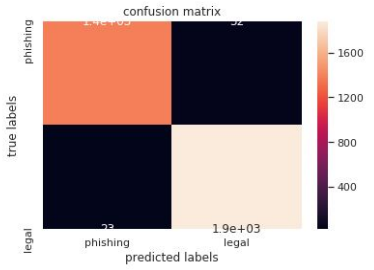


Fig. 5.13 Confusion Matrix calculation

```
In [80]: from sklearn.metrics import accuracy_score
         from sklearn.metrics import precision_score
         from sklearn.metrics import recall_score
         from sklearn.metrics import f1_score

         accuracy=accuracy_score(Y_test,pred)
         print("accuracy = ",accuracy)
         print("\n")
         precision=precision_score(Y_test,pred)
         print("precision = ",precision)
         print("\n")
         recall=recall_score(Y_test,pred)
         print("recall = ",recall)
         print("\n")
         f1_score=f1_score(Y_test,pred)
         print("f1_score = ",f1_score)

         accuracy =  0.9834137515078407


         precision =  0.9832548403976975


         recall =  0.9879074658254469


         f1_score =  0.985575662208235
```


Optimal Model is applied on 10% of data

Fig. 5.14 Accuracy, Precision, Recall, F1_score calculation