Expert Systems With Applications 165 (2021) 113863

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

An improved ELM-based and data preprocessing integrated approach for phishing detection considering comprehensive features

Liqun Yang [a,*], Jiawei Zhang [b], Xiaozhe Wang [c], Zhi Li [a], Zhoujun Li [a,d], Yueying He [b]

[a] School of Computer Science and Engineering, Beihang University, Beijing 100191, China
[b] National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing, 100029, China
[c] Department of Electrical and Computer Engineering, McGill University, Montréal, Quebec H3A 0G4, Canada
[d] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China

ARTICLE INFO

Keywords:
Phishing detection
Extreme learning machine (ELM)
ADASYN
SDAE
Dimension reduction
Non-inverse matrix

ABSTRACT

In this paper, a novel approach based on non-inverse matrix online sequence extreme learning machine (NIOSELM) for phishing detection is presented, which takes into account three types of features to comprehensively characterize a website. For the NIOSELM algorithm, we use Sherman Morriso Woodbury equation to avoid the matrix inversion operation, and introduce the idea of online sequence extreme learning machine (OSELM) to update the training model. In order to reduce the dependence of the detection model on the majority class, we use Adaptive Synthetic Sampling (ADASYN) algorithm to generate the synthetic minority class samples to balance the distribution between the samples of the majority and minority classes. Furthermore, an improved denoising auto-encoder (SDAE) is designed to reduce the dimension of the experimental dataset. The experimental results show the efficiency and feasibility of the proposed detection mechanism. Moreover, the overall detection performance of NIOSELM is better than that of other existing methods, especially in training speed and the detection accuracy.

1. Introduction

Phishing is a common cyber-attack method with attempts to illegally obtain sensitive data by publishing fraud websites pretending to be legitimate. Sensitive data acquired by phishing is mainly social and financial information such as username/password and credit card details, which brings serious financial losses to the victims (Nguyen et al., 2014). Phishing can be initiated in the following ways: 1) randomly disseminate spams containing phishing links to the victims; 2) steal the personal property by disguising fake websites as the existing trustworthy entities such as online banking and stock trading; 3) create fraud e-commerce websites with fake commodity information to defraud money from purchasing; 4) send misleading messages with phishing links to the users by some social software (Xu, Zhan, Xu, & Ye, 2013; Xiang, Hong, Rose, & Cranor, 2011a, 2011b; Moustafa, Misra, & Slay, 2018). According to the report released by Anti-phishing Alliance of China, around 3 billion RMB of loss was caused by phishing attacks and 30% of online shopping customers suffered from phishing (APAC, 2017). The Anti-Phishing Working Group (APWG) reported that 100 million phishing websites were recorded in 2018, and the most affected fields were payment service, software-as-a-service (SaaS) and webmail services occupied (APWG, 2018). The events of phishing attacks show an increasing trend despite the fact that many anti-phishing solutions have been utilized.

The existing research on detecting phishing can be summarized as: Help the users to improve the ability of recognizing phishing websites (Dou et al., 2017). Design the phishing detection techniques for automatic recognition or warning of phishing websites. The phishing detection techniques receive lots of attention from recent work and make great progress (Pham et al., 2018). Blacklisting is the most widely used phishing detection technique, which is utilized in many popular browsers (e.g. IE, Firefox and Chrome). By checking whether the URL of the current web page matches any records in a blacklist, it can make a decision to block the URL existed in the blacklist. In addition, several software and browser plugins such as NetCraft and SiteAdvisor filter phishing websites by loading browser toolbars. However, these blacklist-based approaches require frequently updating the blacklist for getting the newly published phishing websites (Netcraft; Mcafee Labs,

* Corresponding author.
E-mail addresses: lqyang@buaa.edu.cn (L. Yang), zhangjiawei@cert.org.cn (J. Zhang), xiaozhe.wang2@mcgill.ca (X. Wang), limhady@163.com (Z. Li), lizj@buaa.edu.cn (Z. Li), hyy@cert.org.cn (Y. He).

https://doi.org/10.1016/j.eswa.2020.113863
Received 10 October 2019; Received in revised form 15 July 2020; Accepted 7 August 2020
Available online 16 August 2020

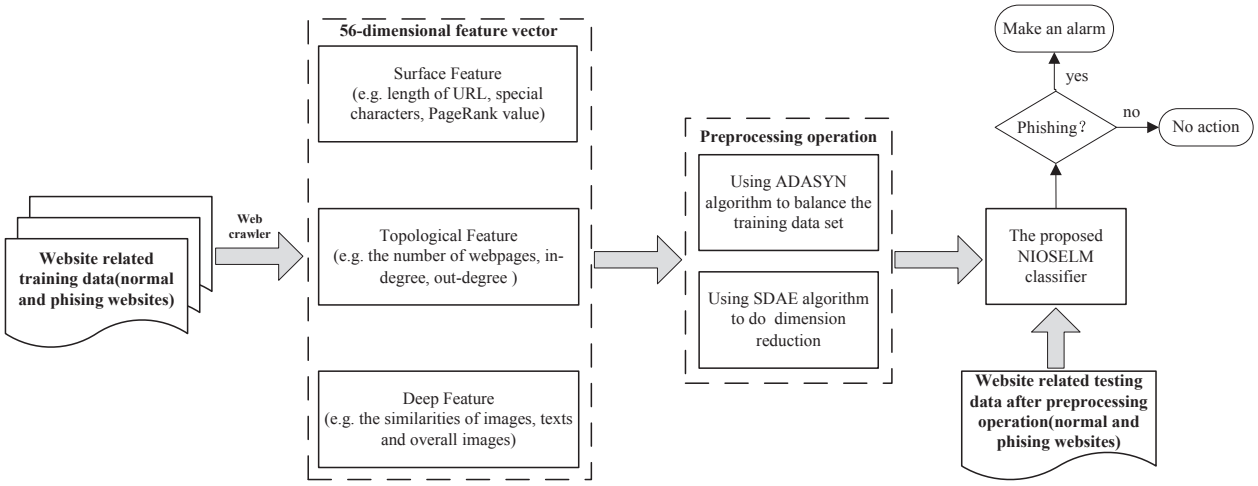


Fig. 1. The overview of the detection framework.

2012). This means that they cannot identify the phishing URL which is not in the blacklist in advance.

Among various detection techniques, machine learning is the most extensive used technique in heuristic-based methods. Most of them take phishing detection as a classification problem. Zhou et al. abstracted the website to a 30-dimensional feature vector and used machine learning method (Zhou et al., 2019) to detect the phishing websites. Liu et al. (Liu et al., 2010) proposed a semantic-link-network-based (SLN) approach, which builds SLN on the target websites and reasons phishing websites through semantic relations. (Sahingoz, Buber, Demir, & Diri, 2019) proposed real-time and language-independent classification algorithms are proposed to detect phishing websites. To enrich the feature vectors, the authors adopt NLP-based method to extract the features. The EMD algorithm was introduced (Fu, Liu, & Deng, 2006) to compute images similarities from HTML files. In addition, some deep learning based techniques have been applied in detecting phishing. The authors in (Halgaš, Agrafiotis & R.C. Nurse) proposed an online phishing detection method based on Recurrent Neural Networks (RNNs). This work is efficient but training RNN is time consuming. Moreover, it just takes few text features into account, which cannot fully characterize the phishing websites. (Smadi, Aslam, and Zhang, 2018) proposed a novel detection framework based on evolving neural network (ENN) and reinforcement learning algorithms. Although this method achieves high detection accuracy with a low false positive rate, it takes a long time for the reinforcement learning process. In addition, (Xiang, Hong, Rose, & Cranor, 2011a, 2011b) compared six machine learning methods for detecting the phishing websites, including SVM, Logistic Regression (LR), Bayesian Network (BN), J48 Decision Tree, Random Forest (RF) and Adaboost, and produced the result that BN performed the best among other classifiers. However, they achieved only about 92% true positive rate on unique testing phishing websites. Although the existing approaches obtain good detection performances, they are time consuming on training detection model and the detection accuracy is not optimal.

To bridge the research gap, in this paper, we propose an improved ELM (Non-inverse matrix extreme learning machine, NIOSELM) as the classifier in terms of a faster training speed and higher detection accuracy. To comprehensively characterize the phishing websites, the surface features, topological features, and deep features are extracted and blended globally. To reduce the strong correlations of the features selected manually, and shorten the time on training the detector or testing the detection model. We adopt Stacked Denoising Auto-Encoder (SADE) to reduce the dimensionality of the experimental data (Abusitta, Bellaiche, Dagenais, & HalabiA, 2019) and Adaptive Synthetic Sampling (ADASYN) algorithm (Baryannis, Dani, & Antoniou, 2019) to generate the synthetic minority class samples to balance the dataset.

The rest of this paper is organized as follows: Section 2 describes the preliminary work and gives an overview of the proposed detection framework. In Section 3, the detailed explanations of 3 kinds of features are introduced, and the data pre-processing algorithms are introduced briefly. A novel phishing detection algorithm is proposed and described in details. In Section 4, lots of experiments are conducted to verify the efficiency and feasibility of the proposed mechanism. Finally, we end this paper with conclusions in Section 5.

2. Preliminary work and overview of our framework

2.1. Preliminary work

Phishing detection can be regarded as a binary classification problem which needs to define the index system to discriminate phishing websites according to the differences between the normal websites and phishing websites (Silva, Feitosa, & Garcia, 2020). We extract the website features from the following three aspects to fully represent the websites.

1) ***Surface features*** which can visibly characterize the phishing websites and the normal ones by the URLs (Uniform Resource Locators) information. Generally, the URLs of the phishing websites are different from that of the normal websites. For example, the phishing URLs may contain special characters and non-standard port numbers.
2) ***Topological features*** which represent the topological structure of a website, and can be extracted from the contents of webpages. Generally, for the sake of saving development cost, phishing websites are designed much simpler than the legitimate ones. It means that the topological information of phishing websites is not rich.
3) ***Deep features*** which can be statistically calculated by the implicit information of images and texts of websites. For example, users can view images and forms from a website, but they cannot observe the pixel size of the images which may be different between the phishing websites and the normal ones. In this paper, we calculate the image similarity and text similarity and the overall image similarity as the deep features.

The features extracted in this paper are explained in details in the next section. We can do statistics based on the URLs information collected by SpoofGuard to obtain the surface features (Likarish, Jung, Dunbar, Hansen, & Hourcade, 2008). For the topological features, we first abstract the webpage contents to the DOM trees, and use a web crawler to get the content information and generate the topological

Table 1
Definition and explanation of the URL feature.

No.	Definition	Value
F1	The length of a URL exceeds	if len(URL) greater than 23 F1 = 1; otherwise F1 = 0
F2	URL contains "@"	if contains F2 = 1; otherwise F2 = 0
F3	URL contains IP address	if contains F3 = 1; otherwise F3 = 0
F4	URL contains "account/login/ ebay/ bank/confirm"	if contains F4 = 1; otherwise F4 = 0
F5	URL uses non-standard port	if uses F5 = 1; otherwise F5 = 0
F6	URL contains "https"	if contains F6 = 1; otherwise F6 = 0
F7	The number of sub-domain in a URL is greater than 5	if the number greater than 5 F7 = 1; otherwise F7 = 0
F8	URL uses redirection operator "//"	if uses F8 = 1; otherwise F8 = 0
F9	The number of "."in a URL	If the number greater than 4 F9 = 1; otherwise F9 = 0
F10	Information entropy of a URL	The value of information entropy
F11	URL contains character"-"	if contains F11 = 1; otherwise F11 = 0
F12	PageRank value of a URL is greater than 0.2	If exceeds F12 = 0; otherwise F12 = 1

Table 2
Definition and explanation of the domain feature.

No.	Definition	Value
F1	The registration date of the domain in a URL	if registration date < 90 days F1 = 1; otherwise F1 = 0
F2	The validity period of the domain in a URL	if validity period < 1 years F2 = 1; otherwise F2 = 0
F3	Domain contains DNS records	if contains F3 = 0; otherwise F3 = 1
F4	The domain of a URL exists in Google Index	if exists F4 = 0; otherwise F4 = 1

features. The deep features are calculated based on deep information of images and texts. In order to improve efficiency, the web crawler is designed using the procedures described in (Li, Yang, & Ding, 2016; Gupta, Gupta, Chaudhary, 2018).

2.2. The overview of our framework

The proposed framework is mainly composed of four parts, which are data acquisition, feature extraction, data preprocessing and phishing detection. The first part uses a web crawler to download webpage contents and record the corresponding URLs and domains. In the second part, we analyse the information of URLs and domains to generate surface features. After parsing the webpages' HTML files to DOM trees, we can generate the topological features by traversing the tree structures. The deep features are calculated based on the deep information. In this paper, we generate 56 features to represent a website. However, high dimension feature data results in longer training time of the detection model and may cause "dimension curse". In the real world, the numbers of phishing websites and the normal ones are extremely unbalanced, the unbalanced data will train a classifier with poor performance. Based on these analyses, we use SDAE and ADASYN algorithms to reduce the dimensionality of data and balance the dataset, respectively. At last, we use the pre-processed data to train the classifier based on NIOSELM. Fig. 1 illustrates the overview of the proposed detection mechanism.

3. The proposed phishing detection mechanism

3.1. Feature definition and extraction

We divide the surface features into the URL features and the domain features by analysing the URLs and domains. Generally speaking, normal URL consists of protocol, domain name, file path, and query parameters. Phishing website has a short life cycle, so it is difficult to find the corresponding DNS record (Ding, Luktarha, Li, & Slamu, 2019).

Table 3
Definition and explanation of the topological feature.

No.	Definition	Value
F1	The number of webpages in a website	F1 = the number of the webpages
F2, F3	The mean and variance of the numbers of images for the webpages in a website	F2 = the mean value of the numbers; F3 = the variance of the numbers
F4, F5	The mean and variance of the numbers of inbound links for webpages in a website	F4 = the mean value of the numbers; F5 = the variance of the numbers
F6, F7	The mean and variance of the numbers of outbound links for webpages in a website	F6 = the mean value of the numbers; F7 = the variance of the numbers
F8, F9	The mean and variance of the numbers of internal links for webpages in a website	F8 = the mean value of the numbers; F9 = the variance of the numbers
F10, F11	The mean and variance of the numbers of CSS files for the webpages in a website	F10 = the mean value of the numbers; F11 = the variance of the numbers
F12, F13	The mean and variance of the numbers of JS files for the webpages in a website	F12 = the mean value of the numbers; F13 = the variance of the numbers
F14, F15	The mean and variance of the numbers of in-degree for the webpages in a website	F14 = the mean value of the numbers; F15 = the variance of the numbers
F16, F17	The mean and variance of the numbers of out-degree for the webpages in a website	F16 = the mean value of the numbers; F17 = the variance of the numbers
F18, F19	The mean and variance of the numbers of "forms" tags for the webpages in a website	F18 = the mean value of the numbers; F19 = the variance of the numbers
F20, F21	The mean and variance of the numbers of "input" tags for the webpages in a website	F20 = the mean value of the numbers; F21 = the variance of the numbers
F22, F23	The mean and variance of the numbers of "password" boxes for the webpages in a website	F22 = the mean value of the numbers; F23 = the variance of the numbers
F24, F25	The mean and variance of the ratios of the numbers of forms to the average out-degree for the webpages in a website	F24 = the mean value of the ratios; F25 = the variance of the ratios
F26, F27	The ratios of the number of static and dynamic webpages to the number of all webpages in a website	F26 = the ratio of the static webpages; F27 = the ratio of the dynamic webpages
F28	The number of "hidden" tags for the webpages in a website	F28 = the number of "hidden" tags

Moreover, the traffic of phishing websites is less than that of the normal ones. We select 12 features related to the URLs and 4 features related to the domains as the surface features. Tables 1 and 2 define the URL features and domain features.

Although phishing websites are visually similar to the normal ones, the topologies of the phishing websites are extremely simple. It is found that thousands of pages and links exist in the normal large websites (Chiew et al., 2019). Phishing websites are developed and deployed within in a short time, so their structures are difficult to be designed as complex as the normal ones. The existing phishing detection technologies are mainly based on the features of single webpage and they ignore the entire topology structure of the website where the webpages are located in. In this paper, we develop 30 topological features of a website by analysing the HTML contents. The definitions of topological features are shown in Table 3.

It is noted that the password box can be indicated by <input> tag with type="password". Inbound link and outbound link refer to the links which points to other websites and is pointed by other websites. In-degree and out-degree refer to the ratio of the number of links pointing to the analysed website to the number of the webpages in the analysed website. The URLs of dynamic webpages end with the suffixes such as php, aspx, asp, jsp, etc., and that of static webpages end with suffixes such as htm and html. In order to get the deep features, we first use a web crawler to collect the webpages' images and the text contents. Then calculate the text similarity, image similarity and overall image

similarity as the deep features. The deep features are explained in the details below.

Definition 1. Use quintuple vector $m_i =< m_{i1}, m_{i2}, ..., m_{i5} >$ to express the features of the *ith* image in a webpage, where $m_{ik}, k = 1, 2, ..., 5$ are the src attribute /the area of image/the colour histogram/the location and the wavelet feature of the *ith* image in this webpage. In this paper, we select one legitimate website as the referenced website to calculate the similarities with the target website.

Set $S_{src}, S_A, S_{CC}, S_W$ as the similarities of the image src attribute, the area of image, the image colour histogram and the image wavelet feature which can be expressed in a compact from as follows:

$$S_{\mathrm{src}}(F, F^{'}) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} S_{src}^{(i),(j)}(F_{src}^{'(j)}, F_{src}^{(i)})}{m^{*}n} \tag{1}$$

$$S_{\mathrm{src}}(A, A^{'}) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\left(1 - \frac{|A_i - A_j^{'}|}{max(A_i, A^{'})}\right)}{m^{*}n} \tag{2}$$

$$S_{\mathrm{CC}}(C_C, C_C^{'}) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{(C_i * C_j^{'})}{|C_j^{'}|*|C_i|}}{m^{*}n} \tag{3}$$

$$S_W(W^{'}, W) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{(W_j^{'} * W_i)}{|W_j^{'}|*|W_i|}}{m^{*}n} \tag{4}$$

where $S_{src}^{(i),(j)}(F_{src}^{'(j)}, F_{src}^{(i)}) = \begin{cases} 0, & F_{src}^{'(j)} = F_{src}^{(i)} \\ 1, & F_{src}^{'(j)} \neq F_{src}^{(i)} \end{cases}$, $F_{src}^{'(j)}$,$F_{src}^{(i)}$ are the *jth* image src attribute of the target website and *ith* image src attribute of the referenced website. $A_i, C_i, W_i, A_j^{'}, C_j^{'}, W_j^{'}$ are the area/the histogram and the wavelet feature of the *ith* image of the referenced websites, and the area/the histogram and the wavelet feature of the *jth* image of the target websites.

Definition 2:. Use six-tuple vector $t_i =< t_{i1}, t_{i2}, ..., t_{i6} >$ to express the features of the *ith* text in a webpage, where $m_{ig}, g = 1, 2, ..., 6$ are the text string/the foreground and background colour/the text font size/the font name/the text location of the *ith* text in the webpage. Assume the referenced website contains text set $T =< t_1, t_2, ..., t_n >$, and the target website contains text set $T^{'} =< t_1^{'}, t_2^{'}, ..., t_m^{'} >$. Set $S_T, S_c, S_{FN}, S_{FS}, S_P$ as the similarities of the text string, the text colour, the font name and font size and the text location which can be expressed in the following compact forms:

$$S_{\mathrm{src}}(T, T^{'}) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\left(1 - \frac{D_{LT}(t_i, t_j^{'})}{max(len(t_i), len(t_j^{'}))}\right)}{m^{*}n} \tag{5}$$

$$S_c(C, C^{'}) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\left(1 - \frac{D_{LC}(C_i, C_j^{'})}{ColorNUM}\right)}{m^{*}n} \tag{6}$$

$$S_{FN}(FN, FN^{'}) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} S_{FN}^{(i),(j)}(FN_j^{'}, FN_i)}{m^{*}n} \tag{7}$$

$$S_{FS}(FS, FS^{'}) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\left(1 - \frac{|FS_i - FS_j^{'}|}{max(FS_i, FS_j^{'})}\right)}{m^{*}n} \tag{8}$$

$$S_P(P, P^{'}) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\left(1 - \frac{D_P(P_i, P_j^{'})}{M_d}\right)}{m^{*}n} \tag{9}$$

where $S_F^{(i),(j)}(F_j^{'}, F_i) = \begin{cases} 0, & F_j^{'} = F_i \\ 1, & F_j^{'} \neq F_i \end{cases}$, $F_j^{'}$, F_i are the *jth* text's font name of the target website and the *ith* font name of the referenced website. D_{LT} is the edit distance,$D_{LC} = |R_i - R_j^{'}| + |G_i - G_j^{'}| + |B_i - B_j^{'}|$, where R, G, B are the number of red, green and blue pixels of the foreground and background. $FN_i, FS_j, FN_j^{'}, FS_j^{'}$ are the font names and font sizes of the *ith* text of the referenced website and the *jth* text of the target website. M_d is the length of the webpage diagonal line. D_P is the Euclidean Distance between the two text position vectors.

Definition 3:. We set $<< C_1, Cent_1, N_{c1} >, < C_2, Cent_2, N_{c2} >, ...< C_{ND}, Cent_{ND}, N_{cND} >>$ as the overall image features. Where $C_1, C_2, ..., C_{ND}$ are the different colours, $C_{cent_i} = \frac{1}{N_{Ci}}\sum_{k=1}^{N_{Ci}} coord(k, i)$, $coord(k, i)$ is the coordinate of the *kth* pixel with colour C_i. $Cent_i$ is the central point coordinate of the *kth* pixel with colour C_i. N_{Ci} is the number of pixels with colour C_i. Set $S_{color}, S_{number}, S_{center}$ as the similarities of the overall colour, the overall position and the number of colours which can be expressed in a compact from as follows:

$$S_{color}(C_i, C_j^{'}) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\sqrt{(C_i - C_j^{'}) \times \left(C_i - C_j^{'}\right)^{T}}}{m^{*}n{*}maxcolor} \tag{10}$$

$$S_{number}(N_{Ci}, N_{Cj}^{'}) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{|N_{Ci} - N_{Cj}^{'}|}{max(N_{Ci}, N_{Cj}^{'})}}{m^{*}n} \tag{11}$$

$$S_{center}(Cent_i, Cent_j^{'}) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\sqrt{(Cent_i - Cent_j^{'}) \times \left(Cent_i - Cent_j^{'}\right)^{T}}}{m^{*}n^{*}\sqrt{(w^2 + h^2)}}$$

where $maxcolor, w, h$ are the maximum number of the colours, the height and the width of an image. Above all, we obtain 12 deep features, 28 topological features and 16 surface features with a total of 56 features.

3.2. Balance feature data using ADASYN

The domain and topological features can be easily captured by the web crawler. However, we cannot get these features of all the phishing websites because they may be unavailable over a period of time. It is hard to capture the domain and topological features of all phishing websites from PhishTank. Moreover, the numbers of phishing websites and normal websites are extremely unbalanced in the real world. Based on the unbalanced dataset, the detection result of a classifier is more inclined to the majority class samples and this may degrade the detection performance. Alternatively, we take the synthetic data into account and use ADASYN algorithm to generate the synthetic phishing feature data. The main steps of ADASYN algorithm are shown in Algorithm 1.

3.3. Feature data dimension reduction via SDAE

Auto-Encoder (AE) is an unsupervised learning neural network, which can learn the features from the original dataset X. The structure of AE is based on the input layer, the hidden layer, and the output layer. AE first randomly initializes the network input weight and bias and then update them iteratively during training through Backpropagation. Based on the encoding and decoding processes, AE can form more abstract high-level representations by combining the low-level data features, and it has been widely used in the field of data dimension reduction (Lin et al., 2019).

However, AE model is prone to over-fitting because it is sensitive to the size of training data and the complexity of the trained model. Denoising Auto-Encoder (DAE) adds the noise to the original data, and uses the new data as the input of the neural network (Tong et al., 2018). At the same time, DAE tries to learn how to remove the noise and reconstruct the uncontaminated data as much as possible to improve the robustness and generation. The structure of DAE is shown as Fig. 2. Where the original data X is disturbed by q_D using a certain probability

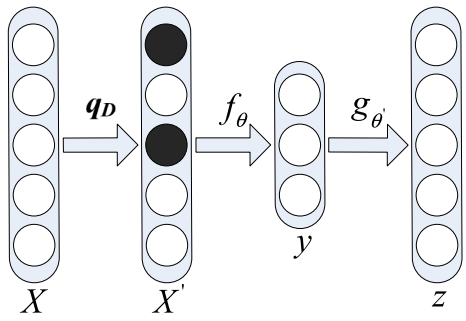


Fig. 2. Schematic diagram of DAE.

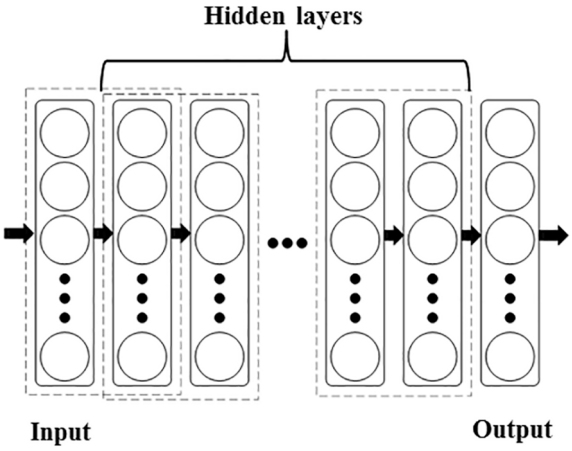


Fig. 3. Schematic diagram of SDAE.

distribution and forming the contaminated data $X^{'}$. After encoding and decoding the output is calculated by the following equations:

$$y = f_\theta(X^{'}) = s(WX^{'} + b) \tag{13}$$

$$z = g_{\theta'}(y) = s^{'}(W^{'}y + b^{'}) \tag{14}$$

where s, $s^{'}$ are the activation functions and θ, $\theta^{'}$ are the fine-tuning parameters.

In this paper, the SDAE (Stacked Denoising Autoencoder) is adopted to reconstruct the feature vector with lower dimension by stacking multiple DAEs. After training a DAE, the input vector is mapped to the hidden layer encoded data. Therefore, we can make further dimension reduction via keeping the encoded data as the training result of the DAE. We use the hidden layer encoded data of the last DAE as the input of the next DAE (Chan et al., 2017). So the encoded data of each DAE can be expressed as follows:

$$x_i = f_{\theta_i}(x_{i-1}) \quad i = 1, 2, 3... \tag{15}$$

where x_i is the result encoded by the *ith* DAE. In this paper, we use "Sigmoid" function as the activation function. The structure of SDAE is shown in Fig. 3.

Algorithm 1: ADASYN to balance the dataset
Defination: m_s the number of minority class
m_l the number of majority class
S_l the majority class
S_s the minority class
$r_{1\times m_s}$ percentage of the nearest neighbours in majority class
$G \leftarrow (m_l - m_s) * \beta$ is the number of synthetic examples that need to be generated for the minority class
Syn
Input: the original training set X, the number of nearest neighbors k, desired balanced level $\beta \in [0, 1]$

(*continued on next column*)

(*continued*)

Output: the oversampled training set
for $i \leftarrow 1$ to m_s **do**
  for each i, compute k nearest neighbors and store the indices in the nn
  $r[i] \leftarrow \frac{|nn[i] \cap S_l|}{k}$
end for
  for $i \leftarrow 1$ to m_s **do**
  $\hat{r}[i] \leftarrow \frac{r[i]}{\sum_i (r[i])}$
  $g[i] \leftarrow int(\hat{r}[i] \times G)$
end for
$Syn_{(G)\times m} \leftarrow$ empty array for synesthetic samples
$j = 1$
for $i \leftarrow 1$ to m_s **do**
  $newindex \leftarrow g[i]$
  while $newindex \neq 0$ **do**
    $k_c \leftarrow$ random number between 1 and k
    $diff \leftarrow S_s[nn[i][k_c]] - S_s[i]$
    $Syn[j][] \leftarrow S_s[i][] + diff \times uniform(0, 1)$
    $j + = 1$
    $m_s - = 1$
  end while
end for
Return *X&Syn*

3.4. The proposed non-inverse matrix online sequence extreme learning machine

Extreme Learning Machine (ELM) is one of single hidden layer feedforward neural networks (SLFNs). Compared with the BP neural algorithm, ELM has many advantages over the convergence speed and local optimization, etc. (Yang et al., 2017). Let the number of input layer nodes n, and the number of hidden layer nodes l, and the number of output layer nodes m, then the ELM can be expressed as follows:

$$f_j = \sum_{i=1}^{l} \beta_i g(w_i, b_i, x_j) \tag{16}$$

where f_j is the value of the *jth* output node. g is the "Sigmoid" function, x_j, w_i are the input and the weight between the inputs and the *ith* hidden layer node. b_i is the bias of the *ith* hidden layer node and β_i is the output weight of the *ith* hidden layer node.

According to Eq. (16), when the number of hidden layer nodes is close to infinity, ELM can approximate the nonlinear continuous function with arbitrarily small error to make the precision to be optimum. To reduce the computation of training ELM, Non-inverse matrix extreme learning machine (NIELM) increases the number of hidden layer nodes by calculating the output weight of the $l + 1$ hidden layer node using the output weight of the l hidden layer node. Define the *ith* input and the *ith* output as x_i and y_i. The input weight matrix and bias are W and b.

The output weight with l hidden layer nodes can be expressed as follows:

$$\beta^l = YD^l \tag{17}$$

$$D^l = f^T(W^lX + 1 \otimes b^l)(f(W^lX + 1 \otimes b^l) f^T\left(W^lX + 1 \otimes b^l\right))^{-1} \tag{18}$$

Let $H = f(W^lX + 1 \otimes b^l)$, we obtain

$$D^l = H^T(HH^T)^{-1} \tag{19}$$

Let $Y = [y_1, y_2, ..., y_k] \in R^{m\times k}$. For the identical matrix I, The Eq. (19) can be rewritten as

$$D^l = H^T\left(k^2 I + HH^T\right)^{-1} \tag{20}$$

For the ELM with $l + 1$ hidden layer nodes, its input weight and bias as:

$$W^{l+1} = \begin{bmatrix} W^l \\ w^T \end{bmatrix}, \quad b^{l+1} = \begin{bmatrix} b^l \\ b_{l+1} \end{bmatrix} \tag{21}$$

where W^l and b^l are the input weight and the bias of the l hidden layer nodes.$w \in R^n$ has the same probability distribution with W^l, and $b_{l+1} \in R$ has the same probability distribution with b^l. The output weight of the ELM with $l + 1$ hidden layer nodes can be calculated by the output weight of l hidden layer nodes.

$$\beta^{l+1} = YD^{l+1} \tag{22}$$

Based on Eq. (18) and Eq. (20), we can obtain

$$\begin{aligned} D^{l+1} = & f^T\left(\begin{bmatrix} W^l X + 1 \otimes b^l \\ w^T X + 1 \otimes b_{l+1} \end{bmatrix}\right) \\ & \times \left(k^2 I + f\begin{bmatrix} W^l X + 1 \otimes b^l \\ w^T X + 1 \otimes b_{l+1} \end{bmatrix} f^T\begin{bmatrix} W^l X + 1 \otimes b^l \\ w^T X + 1 \otimes b_{l+1} \end{bmatrix}\right)^{-1} \end{aligned} \tag{23}$$

Set $d = f(X^T w + b^{L+1}1)$, Eq. (23) can be written as:

$$\begin{aligned} D^{l+1} &= \begin{bmatrix} H^T & d \end{bmatrix}\left(k^2 I + \begin{bmatrix} H^T & d \end{bmatrix}^T \begin{bmatrix} H^T & d \end{bmatrix}\right)^{-1} \\ &= \begin{bmatrix} H^T & d \end{bmatrix}\begin{bmatrix} k^2 I + HH^T & Hd \\ d^T H^T & k^2 I + d^T d \end{bmatrix}^{-1} \end{aligned} \tag{24}$$

Based on the Schur Complement equation, we can obtain

$$\begin{bmatrix} k^2 I + HH^T & Hd \\ d^T H^T & k^2 I + d^T d \end{bmatrix}^{-1} = \begin{bmatrix} K & L \\ M & N \end{bmatrix} \tag{25}$$

where

$$K = (k^2 I + HH^T - Hd(k^2 I + d^T d)^{-1} d^T H^T)^{-1} \tag{26}$$

$$L = -KHd(k^2 I + d^T d)^{-1} \tag{27}$$

$$M = -(k^2 I + d^T d)^{-1} d^T H^T K \tag{28}$$

$$\begin{aligned} N = & \left(k^2 I + d^T d\right)^{-1} d^T H^T KHd\left(k^2 I + d^T d\right)^{-1} \\ & + \left(k^2 I + d^T d\right)^{-1} \end{aligned} \tag{29}$$

Based on Eq. (24) and Eq. (25), we can obtain

$$D^{l+1} = \begin{bmatrix} H^T K + dM & H^T L + dN \end{bmatrix} \tag{30}$$

Assume $m = k^2 I + d^T d - d^T H^T (k^2 I + HH^T)^{-1} Hd, m \neq 0$. Eq. (29) can be transferred to the following equation based on Sherman-Morrison equation (Fang, Cao, Zhou, & Wang, 2018).

$$\begin{aligned} K = & \left(k^2 I + HH^T\right)^{-1} Hdm^{-1} d^T H^T \left(k^2 I + H^T H\right)^{-1} \\ & + \left(kI + HH^T\right)^{-1} \end{aligned} \tag{31}$$

With Eq. (14)–(18), the calculation processes of D^{l+1} using D^l are shown as:

$$\begin{aligned} & H^T K + dM = H^T K - d(k^2 I + d^T d)^{-1} d^T H^T K \\ & = (I - d(k^2 I + d^T d)^{-1} d^T)(D^l Hdm^{-1} d^T D^l + D^l) \end{aligned} \tag{32}$$

$$\begin{aligned} & H^T L + dN = H^T(-KHd\left(k^2 I + d^T d\right)^{-1}) + \\ & d(\left(k^2 I + d^T d\right)^{-1} d^T H^T KHd\left(k^2 I + d^T d\right)^{-1} + \left(k^2 I + d^T d\right)^{-1}) \\ & = -(H^T K + dM)Hd\left(k^2 I + d^T d\right)^{-1} + d\left(k^2 I + d^T d\right)^{-1} \end{aligned} \tag{33}$$

According to Eq. (26), Eq. (27) and Eq. (28), we can obtain

$$D^{l+1} = \begin{bmatrix} D_1^{l+1} & D_2^{l+1} \end{bmatrix} \tag{34}$$

$$\begin{aligned} D_1^{l+1} &= (I - d\left(k^2 I + d^T d\right)^{-1} d^T)(D^l Hdm^{-1} d^T D^l + D^l) \\ D_2^{l+1} &= (I - (H^T K + dM)H)d\left(k^2 I + d^T d\right)^{-1} \end{aligned} \tag{35}$$

In order to improve the speed and the accuracy of modelling, we adopt the idea of online sequential extreme learning machine (OSELM) (Scardapane, Comminiello, Scarpiniti, & Uncini, 2015) to avoid resubmitting all the data to update the training model. After adding the new training data, the output matrix H changes to $[H \quad h]$, where h is the hidden layer output matrix of the new added data. We can use the initialized output layer weight $\beta_0 = Y_0 H_0^T (k^2 I + H_0 H_0^T)^{-1}$ to induce the new output layer weight $\beta^{(k+1)} = \beta^{(k)} + (Y_{k+1} - \beta^{(k)} H_{k+1}) H_{k+1}^T P_{k+1}$, where P_{k+1} defined as:

$$\begin{aligned} P_{k+1} &= P_k - P_k H_{k+1}\left(I + H_{k+1}^T P_k H_{k+1}\right)^{-1} H_{k+1}^T P_k \\ P_0 &= (k^2 I + H_0 H_0^T) \end{aligned} \tag{36}$$

Combine the advantage of OSELM over the model updating process without re-inputting all the past samples and the advantage of NIELM over improving the detection precision without too much computation. We propose a non-inverse matrix online sequence extreme learning machine algorithm (NIOSELM) for phishing detection. The proposed algorithm is shown as **Algorithm 2**.

Algorithm 2: The processes of the proposed NIOSELM
Definition: n, m the dimensions of input and output data The initial training input set $X = [x_1, x_2, \ldots, x_{k0}] \in R^{n \times k_0}$
The initial training output set $Y = [y_1, y_2, \ldots, y_{k0}] \in R^{m \times k_0}$
Initial number of hidden layer nodes l_0
The total number of the training samples k_1
The observation size of data batch s
The output matrix of the hidden layer for the newly added samples h
Initial Input weight and output weight and hidden layer bias $W^{l_0} \in R^{n \times k_0}$, $\beta^{l_0} \in R^{m \times l_0}$, $b^{l_0} \in R^{l_0}$
The approximation error $\eta^* > 0$
$W = W^{l_0}, \beta = \beta^{l_0}, b = b^{l_0}$
while $(\eta > \eta^*)$ **do**
  $l \leftarrow l + 1$
  $w \leftarrow$ a random vector in R^n
  $b_0 \leftarrow$ a random scalar
  Update the input weight and bias vector
  $W \leftarrow \begin{bmatrix} W \\ w^T \end{bmatrix}, \quad b \leftarrow \begin{bmatrix} b \\ b_0 \end{bmatrix}$
  $H = f(WX + 1^T \otimes b), d = f(X^T w + b_0 1)$
  $D_1 = (I - d(k^2 I + d^T d)^{-1} d^T)(DHdm^{-1} d^T D + D)$
  $D_2 = -(H^T K + dM)Hd(k^2 I + d^T d) + d(k^2 I + d^T d)^{-1}$
  $D = [D_1 \; D_2]$
  $\beta^{(0)} = YD$ //Update the output weight of hidden layer
  $\eta = MSE(Y - \beta H)$//Calculate the mean square error
end while
$k = 0$
while $(k_0 \leqslant k_1 - 1)$ **do**
  $H \leftarrow [H \; h]$
  $P_{k+1} = P_k - P_k H_{k+1}\left(I + H_{k+1}^T P_k H_{k+1}\right)^{-1} H_{k+1}^T P_k$
  $\beta^{(k+1)} = \beta^{(k)} + (Y_{k+1} - \beta^{(k)} H_{k+1}) H_{k+1}^T P_{k+1}$
  $k += 1$
  $k_0 += s$
end while
Return $\beta^{(k+1)}$

4. Experiments

4.1. Experimental dataset and evaluation index

After collecting the normal and phishing websites, we can extract 56 features for each website. The normal websites is composed of two parts: Alex's top 2000 world-wide websites and 58,000 DMOZ websites covering the banks, internet, games and commercial (https://www.alexa.com, https://dmoz-odp.org). In order to ensure that the phishing websites are up to date, we retrieve the latest 5000 phishing websites from PhishTank (Phishtank, 2018). Our simulation is implemented on a 64-bit computer, and the type of the processor is Intel Core i7 with the processor of 2.9 GHz clock speed. The number of cores is 4. The RAM is LPDDR3 with 16 GB memory and 2133 MHz frequency. The experimental data is collected by a web crawler based on Python 2.7. Using MATLAB_R2017b to train the NIOSELM model and process the

Table 4
Composition of the original experimental data.

Data type	Normal websites	Phishing websites
Training data	42,000	3500
Testing data	18,000	1500
Total	60,000	5000

Table 5
Evaluation metrics.

Precision	$TP/(TP + FP)$
Recall	$TP/(TP + FN)$
Mcc	$(TP*TN-FP*FN)/((TP + FP)(TP + FN)(TN + FP)(TN + FN))^{0.5}$
Acc	$(TP + TN)/(TP + FN + TN + FP)$
F1	2*Precision*Recall/(Precision + Recall)

experimental data using SDAE and ADASYN. As shown in Table 4, the ratio of the original training data to testing data is 7:3. It is evident that the experimental data is imbalanced because the ratio of the phishing websites to the normal websites is 1:12. We use ADASYN algorithm to generate 45,000 phishing websites in total so that the number of the minority samples is almost equal to the number of the majority samples in the training dataset. In this paper, we normalize the feature data before performing experiments.

In this paper, Precision (Pre), Recall, F1, and Accuracy (Acc) and MCC (Matthews correlation coefficient) are adopted to evaluate the performance of the proposed algorithm. In addition, the true positive, false negative, false positive, and true negative are defined as *TP*, *FN*, *FP*, and *TN*. The details of these metrics are shown in Table 5.

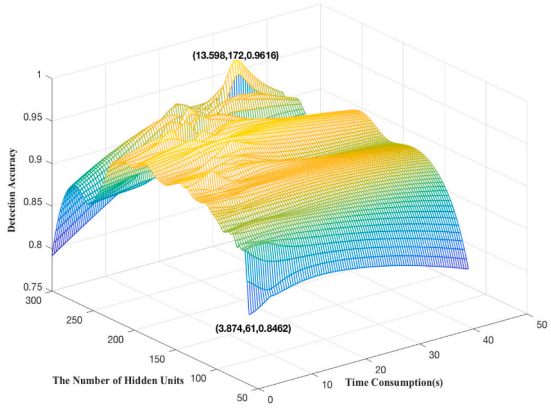
4.2. Experimental results with the number of hidden layer nodes

We first study the impact of the number of hidden layer nodes on the detection results. In this experiment, the structure of ELM contains one single hidden layer, and the number of data samples remains unchanged. The initial number of hidden layer nodes is set as 50. We manually set the approximation error η^* to 1e-4, so the NIOSELM and NIELM can continuously increase the number of hidden layer nodes. The experiment results of NIOSELM, NIELM and OSELM are shown in Fig. 4.

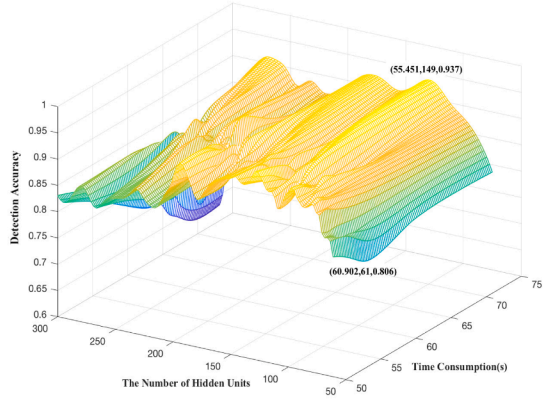
From Fig. 4 we can see that the training time of the three detection models increases and the detection accuracy is in an upward-steady-downward trend with the number of hidden layer nodes continuously increasing. For NIOSELM, OSELM, and NIELM, when the numbers of the hidden layer nodes are greater than 172, 165 and 149, respectively. The detection accuracy decreases with the increase of the training time. According to the above analysis, we can draw the following conclusion: the detection accuracy of the proposed algorithm increases with the number of hidden layer nodes increasing; the classification accuracy is the highest when the hidden layer has 172 nodes; too many hidden layer nodes may lead to over-fitting phenomenon, which reduces the detection accuracy. It is noted that the expected training error η can be calculated according to (Feng, Bao, & Jiao, 1998).

4.3. Experiment results with data preprocessing

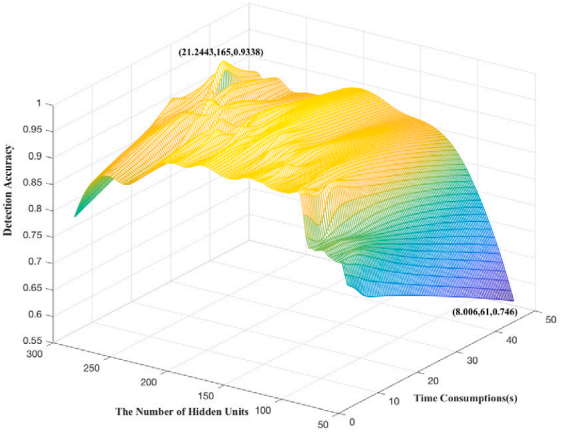
In this section, we mainly explore the effect of the data preprocessing methods on the detection performance of the proposed method, including the effect of using ADASYN and the effect of using SDAE. Firstly, we verify the feasibility of SDAE for dimension reduction. The essence of this experiment is to verify whether the data obtained by SDAE can represent the information of the original data. We choose Mean Squared Error (MSE) as the evaluation criterion to measure the reconstruction error in dimension reduction. MSE is defined as follows:



a. The impact of the number of hidden layer nodes on detection accuracy and training time in NIOSELM.



b. The impact of the number of hidden layer nodes on detection accuracy and training time in OSELM.



c. The impact of the number of hidden layer nodes on detection accuracy and training time in NIELM.

Fig. 4. The impact of the number of hidden layer nodes on detection accuracy and training time.

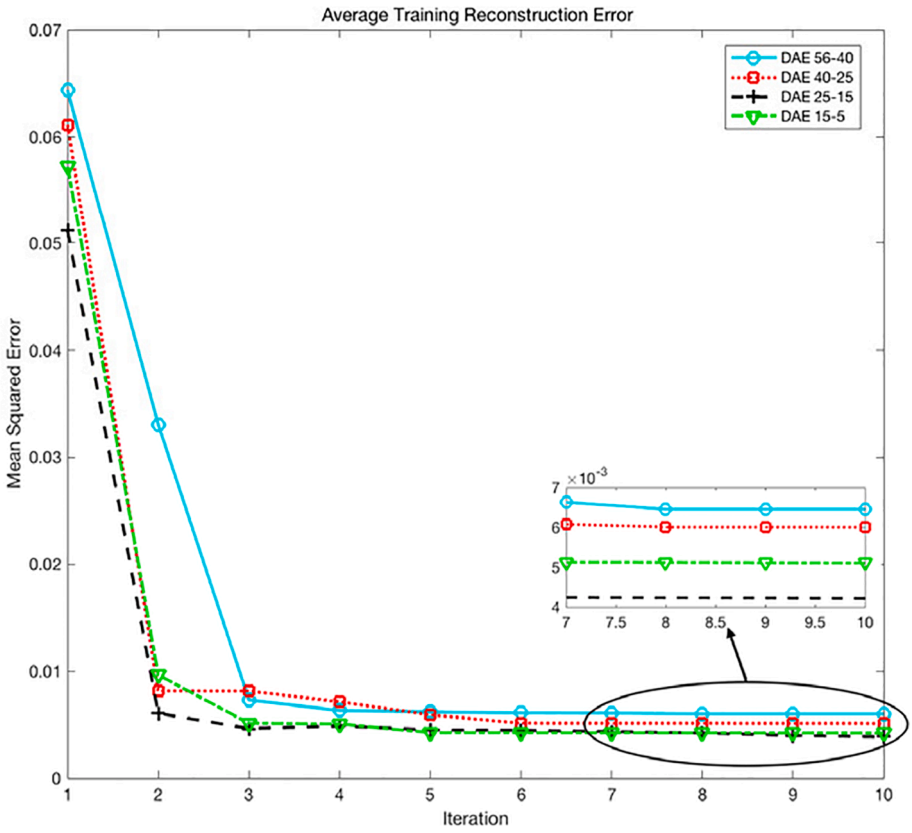


Fig. 5. The reconstruction error of every DAE.

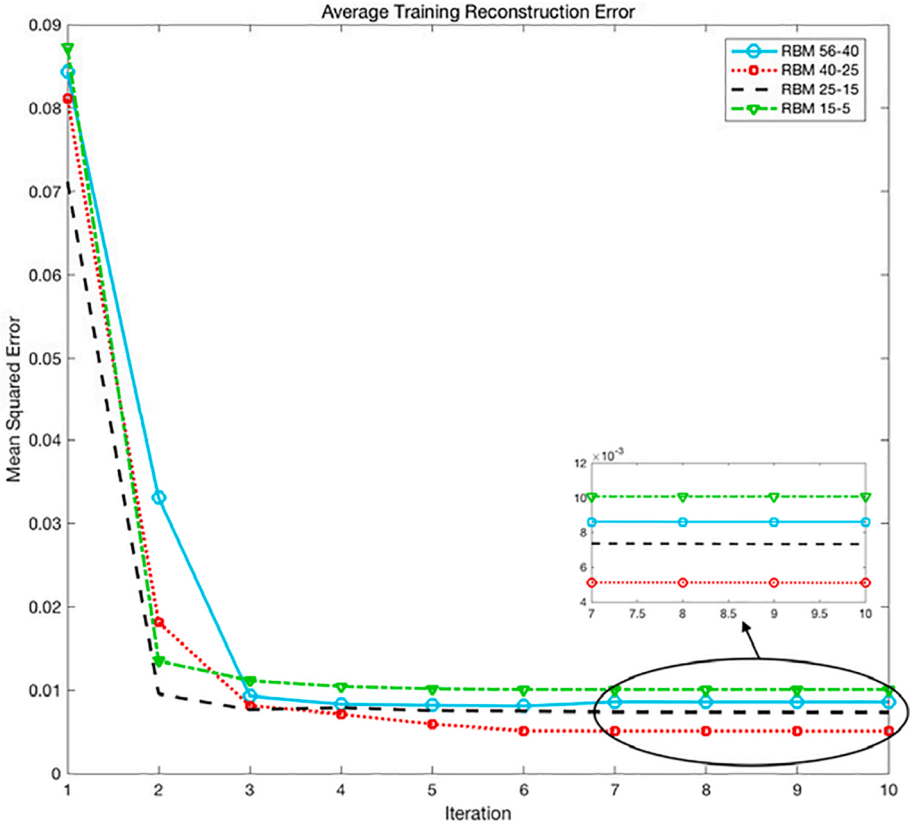


Fig. 6. The reconstruction error of every RBM.

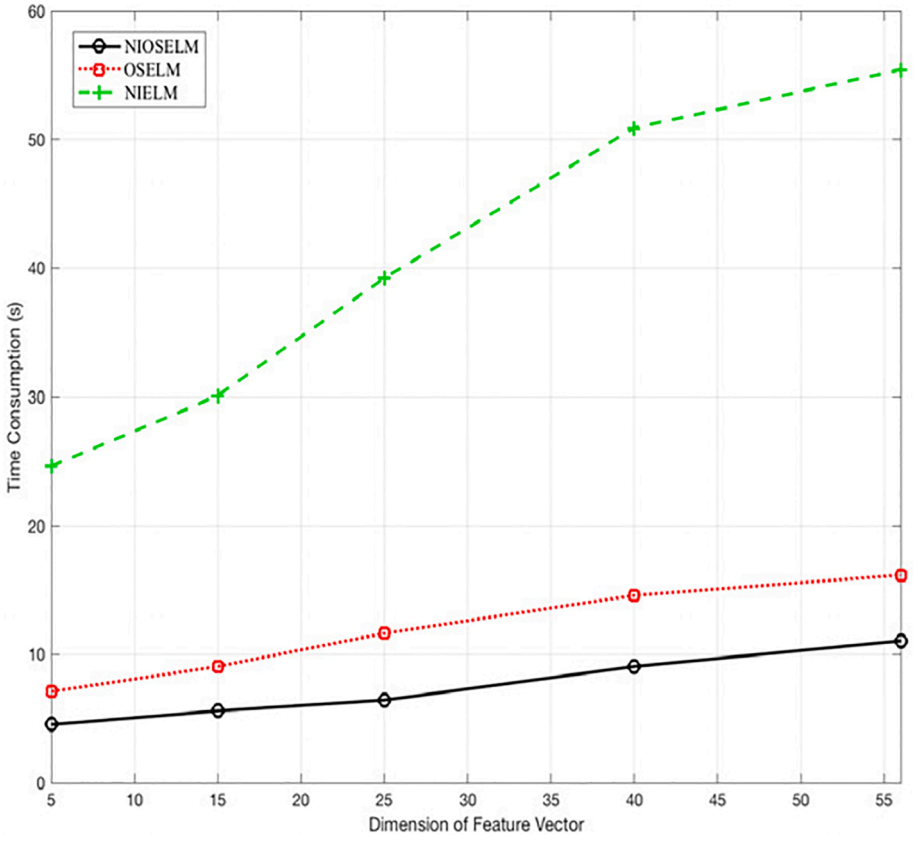


Fig. 7. The impacts of the data dimension on training time.

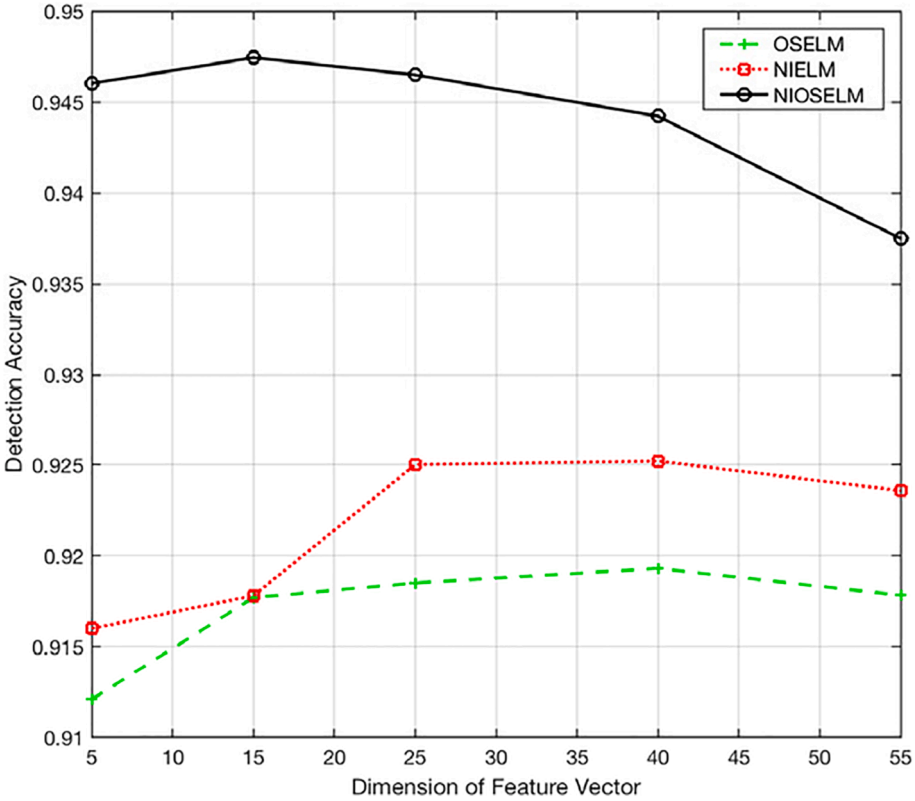


Fig. 8. The impacts of data dimension on detection accuracy.

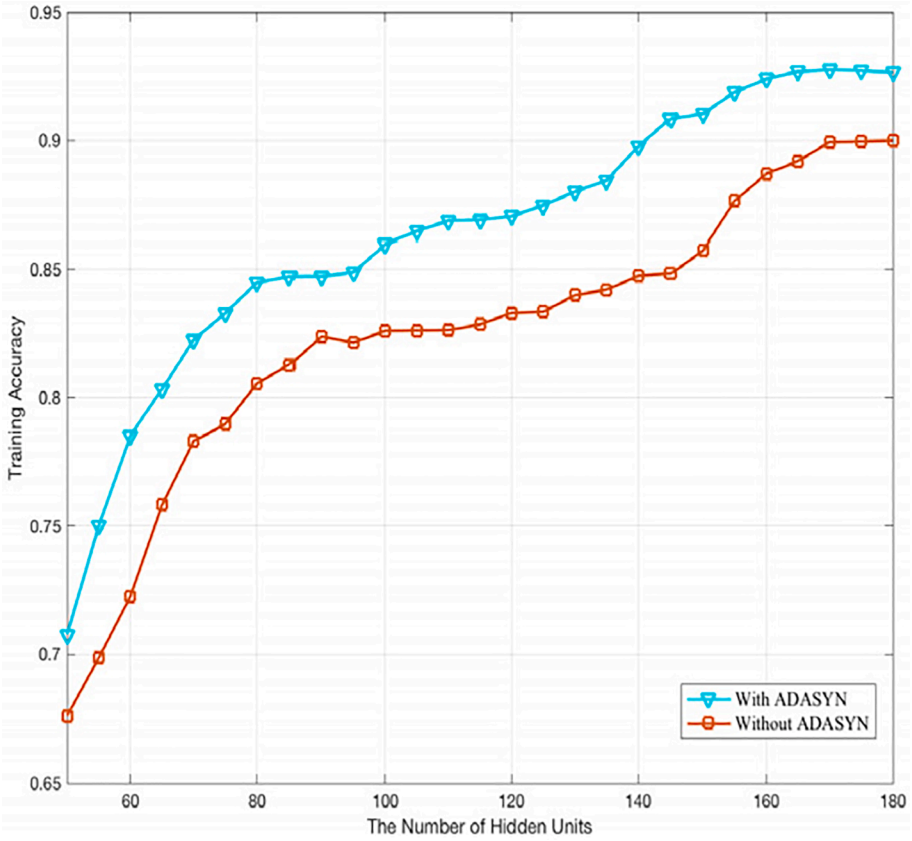


Fig. 9. Training accuracy of NIOSELM.

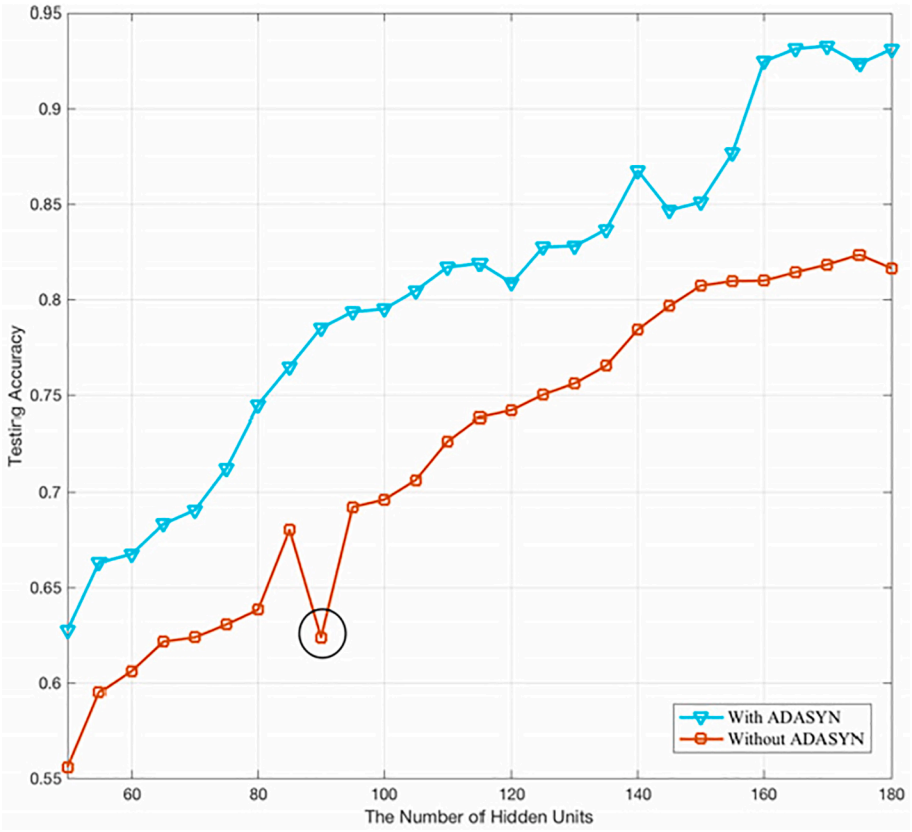


Fig. 10. Testing accuracies of NIOSELM.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_{data} - y_{recon})^2$$

where y_{data} is the training sample or the testing sample, and y_{recon} is the reconstructed sample. N is the number of training samples or testing samples. We use the MSE as the reconstruction errors of SDAE and DAE, and compare the changing curves of MSEs to verify the correctness of using SDAE as the dimension reduction method. The figures below show the changing curves of MSE.

From Figs. 5 and 6, we can see that the MSEs of each DAE in SDAE are <0.008 after 5 iterations. For the RBM-based DAE, when the training time reaches 5, the MSEs of each RBM are about 0.01. SDAE has smaller reconstruction error than DAE, but DAE can get into the steady state faster than SDAE. This is because SDAE consists of multi-layered DAEs. SDAE will get into a stable state only after all the DAEs reach a stable state. According to the above experiments, the correctness of SDAE as a dimension reduction method is proved. It is noted that the purpose of using SDAE is mainly to reduce the model training and detection time. The following figures show the training time and detection accuracy of NIOSELM, NIELM and OSELM models using different dimensional data. The number of hidden layer nodes is set as 150.

As shown in Figs. 7 and 8, the detection accuracy of the proposed method based on different dimensional data changes. The model training time is 11.065 s when using 56-dimensional data, while the training time is 4.544 s with 5-dimensional data, which drops by about 58.9%. The corresponding detection accuracy increases by 0.54% from 0.9375 to 0.9426. Therefore, the dimension reduction method adopted in this paper can greatly reduce the model training time under the premise of maintaining the detection accuracy.

We would like to clarify that the average time consumption of analysing and getting the Domain features for each website is 36.7 ms. For the topological features, we first use web crawler to parse HTML text into the DOM tree, then traverse the structure of the DOM tree and calculate the topological features. This is a time-consuming work that it takes about 2877.4 s to get the topological features for all the normal websites. Moreover, we record the time consumption on SDAE and NIOSELM correctively. The average time consumption on reducing dimensionality for each testing data is 18.77 microseconds, and the average time consumption in SDAE and NIOSELM is 137.69 microseconds.

To validate the effectiveness of ADASYN, we explore the training accuracy and the testing accuracy of the detection model constructed by the pre-processed balanced data and the unprocessed data. Figs. 9 and 10 show the results in the two cases. We can conclude that the detection model trained by the balanced data has relatively higher training accuracy 0.927 and testing accuracy 0.931. Using the imbalanced data to train the detection model, the training accuracy and test accuracy are not high and there is an oscillation 0.624 in the testing accuracy. This is because the imbalanced data cannot cover all the data features to train a detection model with a good generation.

To validate the effectiveness of ADASYN, we explore the training accuracy and the testing accuracy of the detection model constructed by the pre-processed balanced data and the unprocessed data. Figs. 9 and 10 show the results in the two cases. We can conclude that the detection model trained by the balanced data has relatively higher training accuracy 0.927 and testing accuracy 0.931. Using the imbalanced data to train the detection model, the training accuracy and test accuracy are not high and there is an oscillation 0.624 in the testing accuracy. This is because the imbalanced data cannot cover all the data features to train a detection model with a good generation.

4.4. Evaluation of the proposed classifier on the synthetic data

In this subsection, the NIOSELM classifier is trained after performing ADASYN and SDAE in different execution order. Table 6 shows the results of F1, Recall, Acc, Pre and MCC calculated by the experiments. We use the standard deviations of the evaluation indicators to explore the stability of the proposed classifier.

Table 6
Detection results with different execution order of SDAE and ADASYN.

Algorithm	F1	Recall	Acc	Pre	MCC
SDAE + NIOSELM	0.839	0.880	0.799	0.802	0.741
ADASYN + NIOSELM	0.933	0.942	0.930	0.924	0.847
SDAE + ADASYN + NIOSELM	0.899	0.923	0.864	0.876	0.810
ADASYN + SDAE + NIOSELM	0.954	0.951	0.946	0.957	0.879
NIOSELM	0.882	0.935	0.829	0.835	0.773

The written orders of the algorithms in the Table 5 and Fig. 11 represent the execution orders of the algorithms. For example, ADASYN + SDAE + NIOSELM means that execution order of the algorithms is ADASYN-SDAE-NIOSELM. According to the experimental results, we can draw the following conclusions: (i) ADASYN can effectively improve the performance of the proposed detection classifier. For example, in ADASYN + NIOSELM and NIOSELM, the different value of F1 between them is 0.0172, and the precision and the detection accuracy differ by 0.0411 and 0.0354, which indicates that the classifier integrating ADASYN algorithm is more robust. (ii) Using SDAE to reduce the dimensionality of the imbalanced data will result in a significant decrease in the detection performance. This is because the dimension reduction operation reduces the data diversity. The noise is added to SDAE, so the classifier is trained with noisy input and it improves the classification performance. (iii) When the classifier integrates ADASYN and SDAE, the classification performance of the classifier is improved, and the value of F1 is increased by more than 5.7% compared with NIOSELM based classifier. (iv) Because the newly generated data by ADASYN is bound with errors, after executing ADASYN, SDAE not only reduces the dimensionality of the data, but also reduces the noise, which significantly improves the classification performance. Therefore, the classification performance of the ADASYN + SDAE + NIOSELM is much better than that of the SDAE + ADASYN + NIOSELM.

We would like to clarify that the purpose of SDAE is to transform the original data (features) with noise into a clean data without noise. The training process of SDAE should first add the noise to the original data (features), and then use the noisy data (features) to train the SDAE to reconstruct the original data by removing the noise. By doing so, it can make the reconstructed data (features) be more robust, and result in improving the classification performance.

At the end of the experiments, we use the public data mining platform WEKA to generate different classifiers and compare them with the proposed method (Holmes, Donkin, & Witten, 1995). Since the proposed method belongs to an improved algorithm, the superiority of the proposed method is verified by comparing it with the improved SVM (BVM) proposed in the literature (Li, Yang, & Ding, 2016). Finally, we also demonstrate the performance of the proposed method by comparing with a deep learning algorithm (DBN) (Hinton et al., 2006). In this experiment, the classifiers are all trained by the synthetic balanced data without dimensionality reduction.

From Fig. 12, we can see that the AUC (Area Under ROC Curve) of the proposed method is bigger than other ELM-based methods. It can be seen from Fig. 13 that comparing with SVM and RF (Random Forest algorithm) (Patil, & Patil, 2015; Breiman, 2001), the proposed method has obvious advantages of the detection performance, and SVM is more suitable for phishing detection than RF (Fig. 13). In Fig. 14, it is evident that DBN shows good performance in phishing detection, but it is still weaker than NIOSELM. Moreover, from Fig. 15 we can see that the proposed method has a better overall performance than the improved SVM (BVM).

In the previous experiments, we calculate the evaluation metrics on the testing dataset, which is described in Section 4.1. To further validate the performance of our proposed method, we perform 10-fold cross validation on the training dataset to test the detection performance in

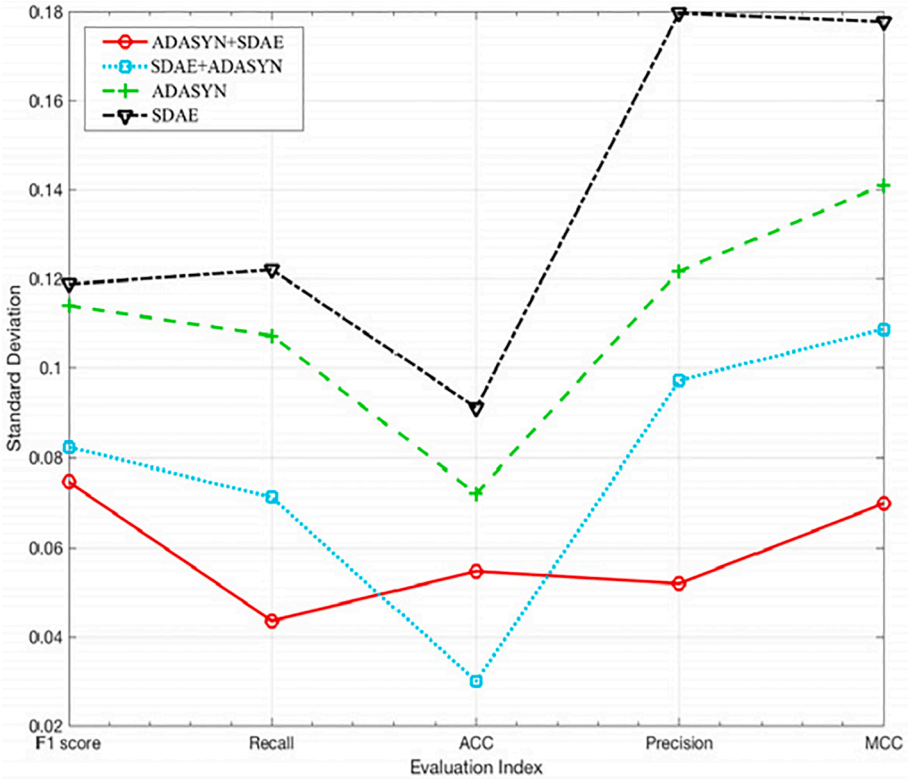


Fig. 11. The standard deviations of the evaluation index values.

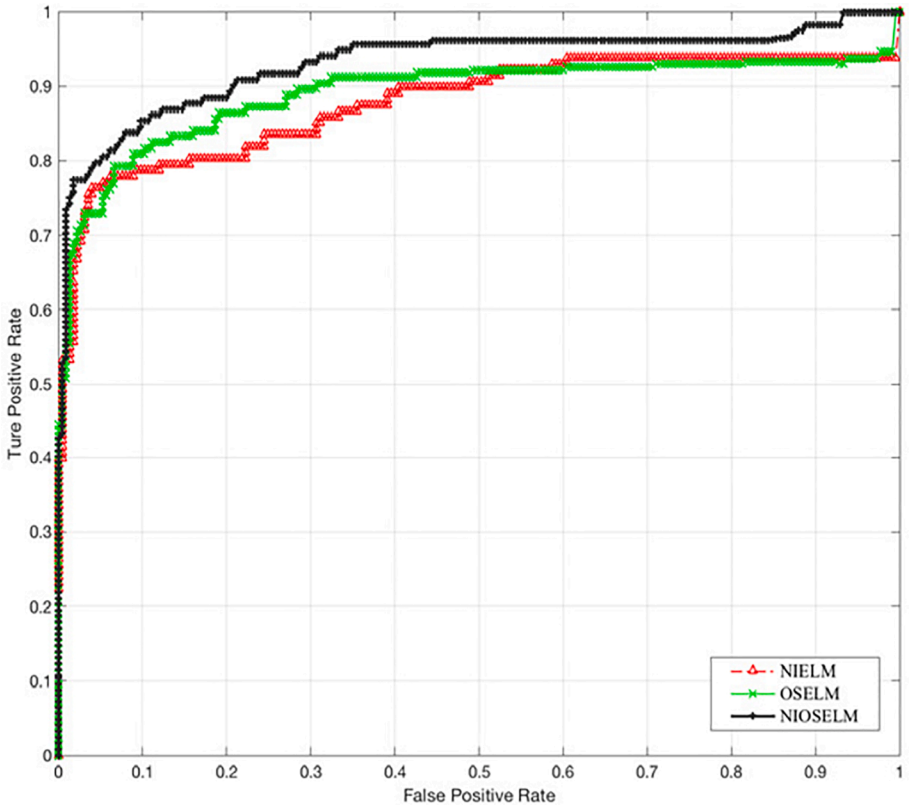


Fig. 12. The ROC curves of different ELM-based methods.

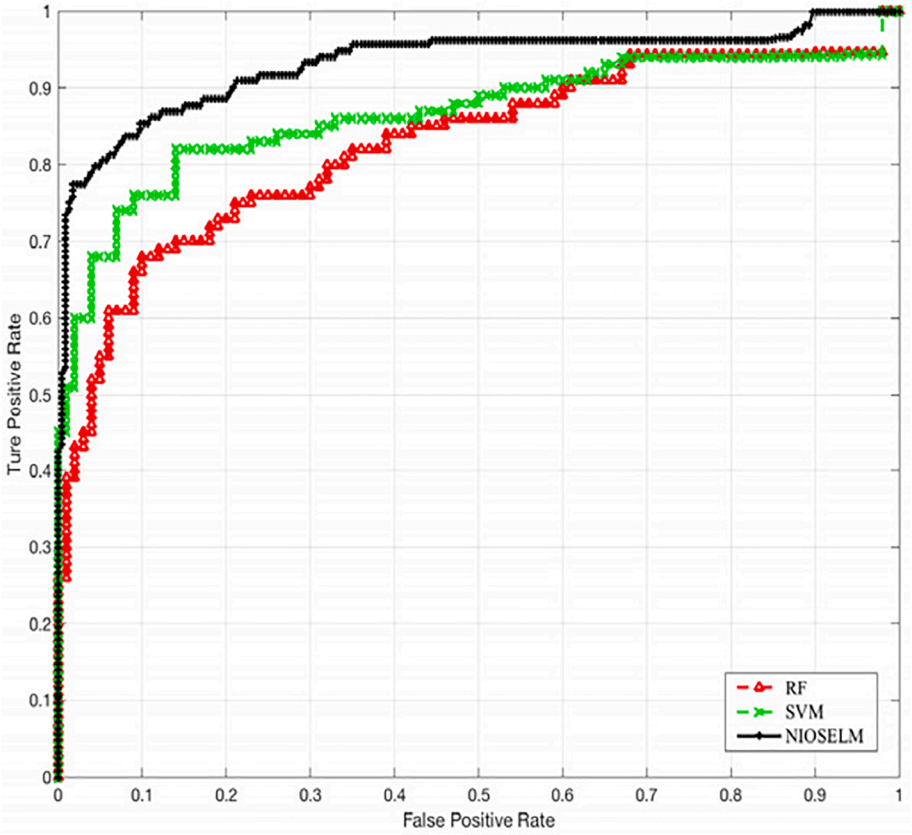


Fig. 13. The ROC curves of RF, SVM and NIOSELM.

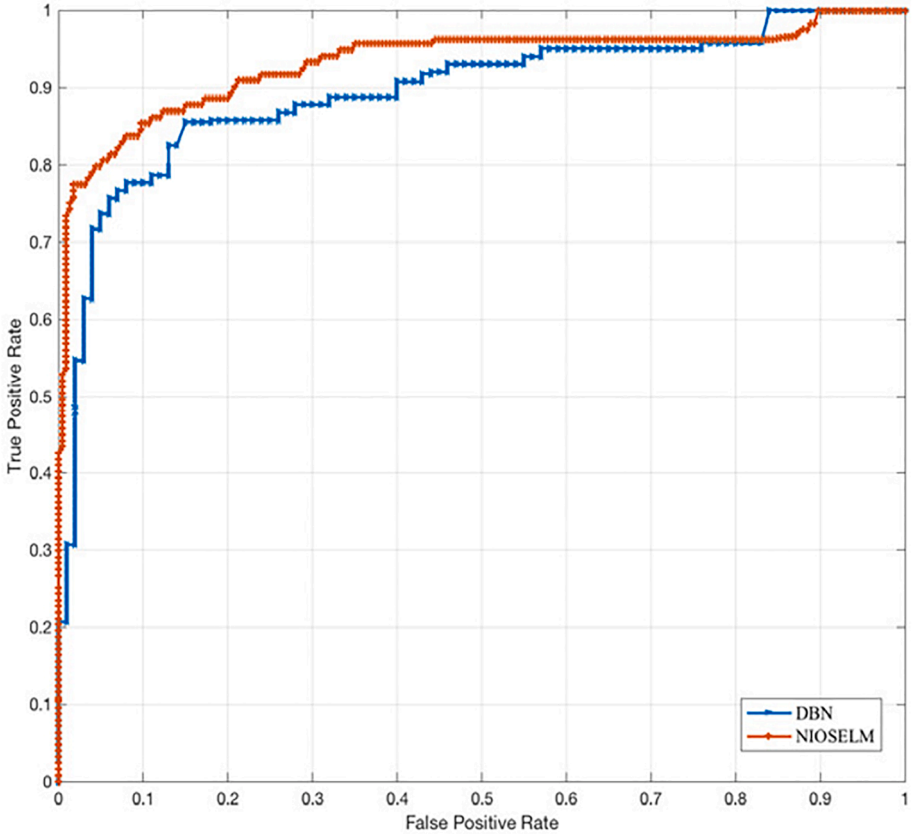


Fig. 14. The ROC curves of NIOSELM and DBN.

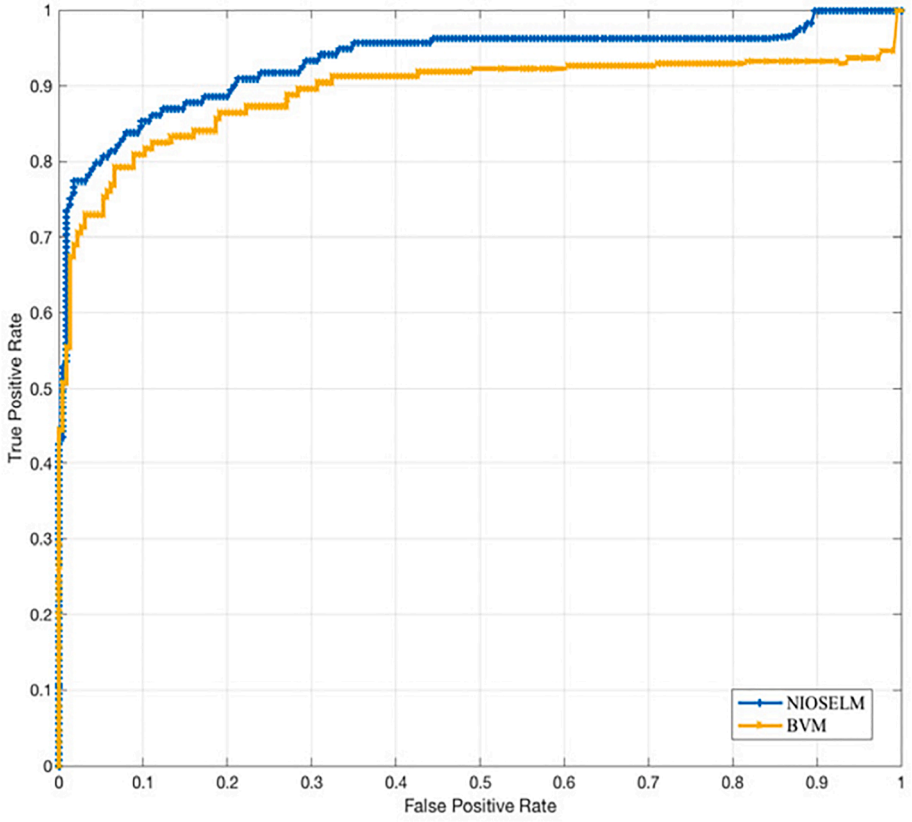


Fig. 15. The ROC curves of NISELM and BVM.

Table 7
Confusion matrix of detection results.

	Classified Phishing	Classified Normal
Phishing	TP (57976)	FN (2024)
Normal	FP (1628)	TN (58372)

the situation of the detector is trained by the synthetic balanced data. That is to say there are 60,000 normal data and 5000 phishing data in the initialized training dataset. For the training dataset, we use ADASYN to generate 55,000 more phishing data so that the training dataset is balanced. Then use SDAE to reduce the dimension. In this experiment, the confusion matrix (İş and Tuncer, 2019) is analysed which is shown in Table 7.

Based on Tables 5 and 7, the Precision and Accuracy can be calculated with the values 97.27% and 96.96%, respectively. At the same time, we can calculate that the values of Recall and F1 are 96.63% and 96.95%. We can conclude that the 10-fold cross validation is a more effective way to validate the detection performance. Examining the confusion matrix, the detection performance of our proposed mechanism is good.

4.5. Evaluation of the proposed classifier using the real data

In this subsection, we continue to numerically evaluate the impact of the balanced real data on the detection performance of our proposed method. To guarantee the experimental data up to date, instead of the synthetic phishing websites, the phishing data is extracted from "PhishTank" from January 1st, 2020. So, there are 60,000 phishing samples and 60,000 normal samples in this experiment. In this subsection, we would like to clarify that in this subsection all the testing results are the mean of the results of performing 10-fold cross validation for 10 times.

Due to the dimensionality reduction method is proved to be effective in the above experiments, and our proposed mechanism is a combination of ADASYN, SDAE and NIOSELM, so we should consider the effectiveness of the dimensionality SDAE for the real balanced data. We first calculate the Precision, Accuracy and Recall based on different dimensional dataset. As shown in Fig. 16, we can see that the detection performance with reduced dimensional dataset is better than that with higher dimensional dataset. We can also see that the detection performance is the best when the dimension is 15. To intuitively show the - performance of our proposed method, we use the Missing Rate (1-Recall) to evaluate our proposed method. From Fig. 17, we can see that the proposed method has the lowest missing rate when the dimension is 15. Moreover, as analysed in Section 4.3, the low dimension data can reduce the time cost. Therefore, we can conclude that the proposed NIOSELM is effective to detect the real-world dataset, and SDAE is effective to reduce the dimension of the data.

To study the robustness of our detection method to the environment noise. We add the Additive White Gaussian Noise (AWGN)$\tilde{N}(0,\sigma)$ with the standard deviation σ ranging from 0.5 to 3, to the training data. We would like to clarify that in the following experiments, the training data is firstly reduced to 15, and we use this dataset to train the NIOSELM-based, OSELM-based and NIELM-based detectors. Comparing the detection accuracy of the three detection models, from Fig. 18, we can see that the NIOSELM can achieve the highest detection accuracy among the three different methods. Furthermore, as the standard deviation σ increases the detection accuracy decreases and the proposed detection mechanism can achieve the accuracy of detection above 72.42%. This demonstrates that our proposed mechanism is robustness to the environment noise.

We continually investigate the advantages of our proposed detection mechanism on the real-world data. In this experiment, we compare the proposed NIOSELM with NIELM and NIOSELM using ROC curve and P-R

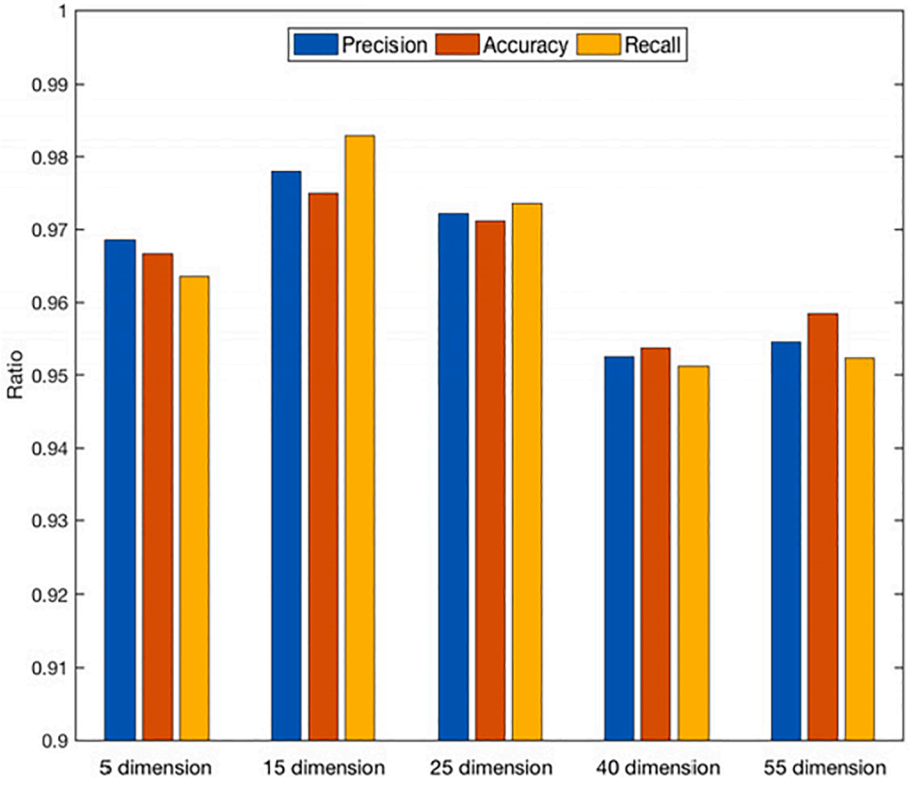


Fig. 16. Detection performance of NIOSELM for the real data.

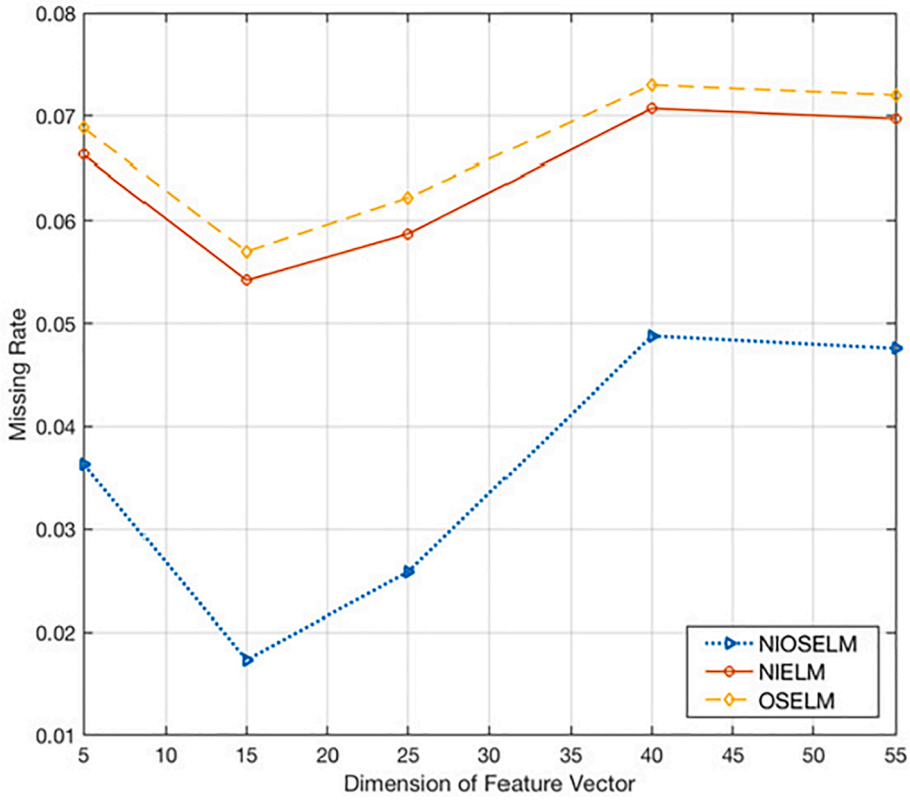


Fig. 17. Impact of dimension on missing rate.

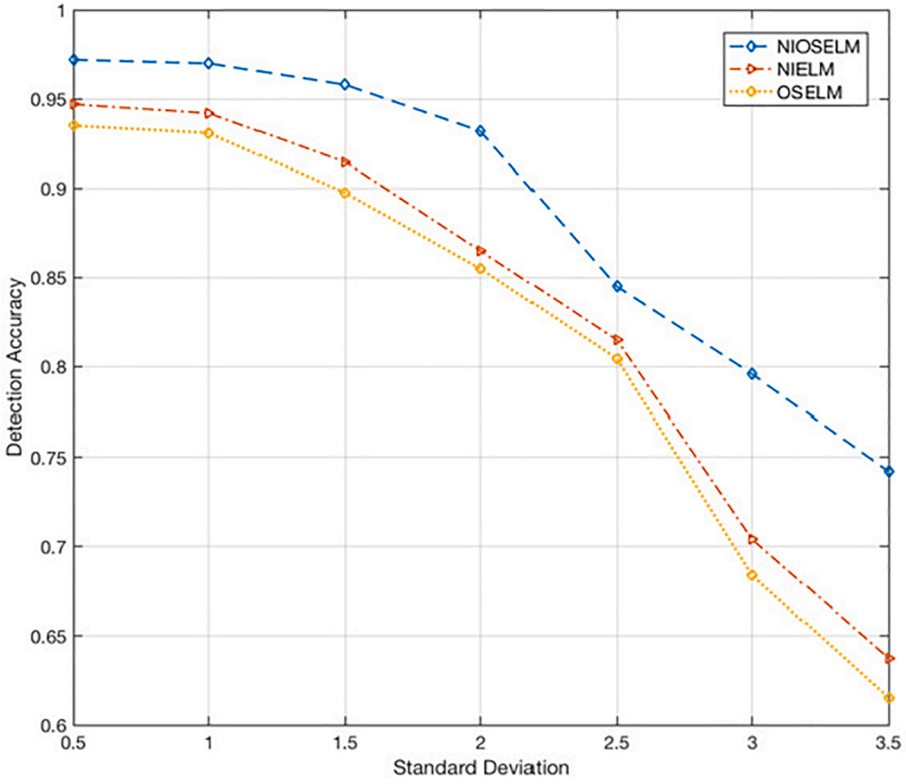


Fig. 18. Detection accuracy with different noise level.

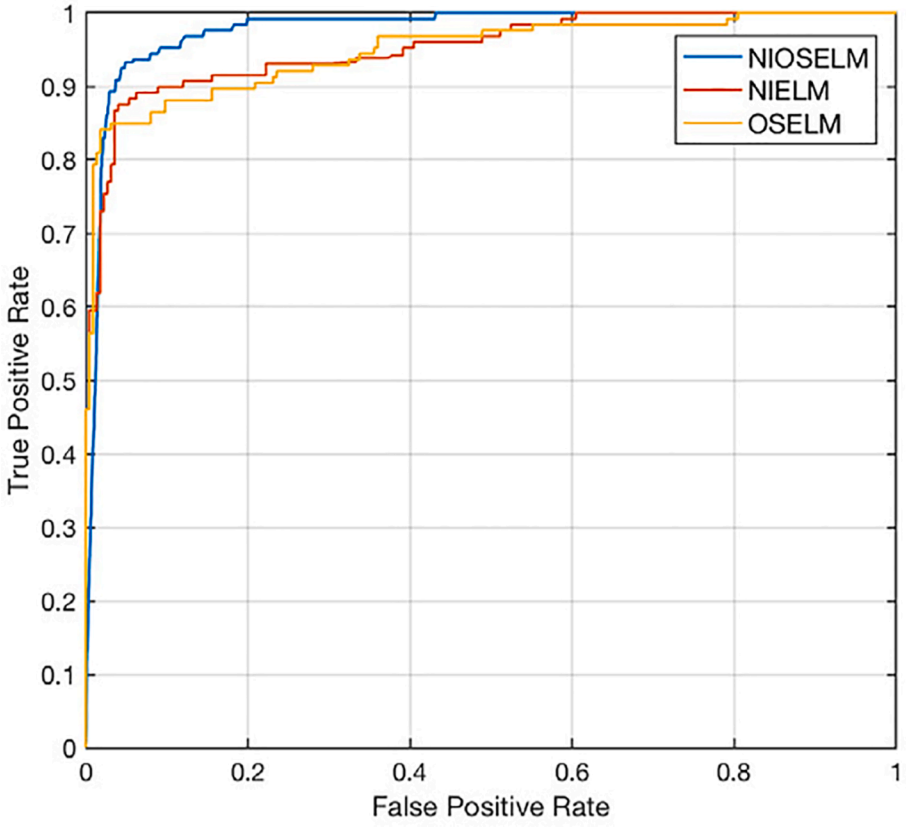


Fig. 19. The ROC curve of NIOSELM, NIELM and OSELM.

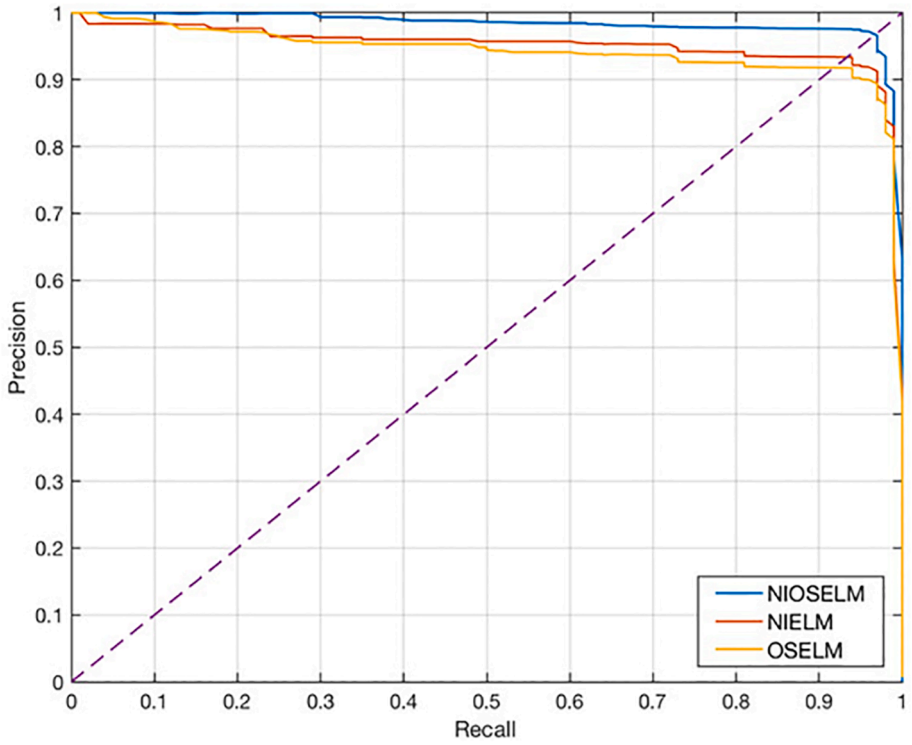


Fig. 20. The P-R curves of NIOSELM, NIELM and OSELM.

Table 8
Comparison of methods.

Method	F1	Recall	ACC	Pre
NIOSELM	0.973	0.983	0.975	0.978
KNN	0.945	0.941	0.950	0.953
LR	0.937	0.952	0.946	0.939
XGBoost	0.955	0.959	0.961	0.965

curve (Precision Recall). The ROC curve and P-R curve are plotted in Figs. 19 and 20. From Fig. 19, we can observe that the proposed mechanism can achieve the best performance, and the AUC is 0.9786. This validates the excellent performance of our detection scheme. From Fig. 20, we can observe that the proposed scheme has the highest precision under the same Recall. Therefore, we can conclude that the proposed method NIOSELM has clear advantage.

At the end of this paper, we compare the proposed method with other methods such as KNN (K-Nearest Neighbor), LR (Logistic Regression), and XGBoost (Li et al., 2019). In Table 8, it is evident that our proposed method has the best detection performance over other methods. Followed by XGBoost, KNN and LR.

5. Conclusions

A novel method for phishing detection based on NIOSELM is proposed. From the respects of surface, topology and inherence of a website, we select 56 features to comprehensively characterize the website. Before training the detection model, preprocessing methods including ADASYN algorithm and SDAE algorithm are adopted to balance dataset and reduce the data dimension. The proposed approach is tested and verified by many experiments which show the feasibility and effectiveness of the detection model. Moreover, the proposed method is also tested on the balanced real data.

Although our method can achieve fairly good detection performance, especially the time efficiency and the detection accuracy, the detection accuracy may not be the best compared with the existing methods. In our future work, we will consider using more efficient data synthesizing methods to balance the dataset. We think there are also other potentially promising works to do for further exploration.

CRediT authorship contribution statement

Liqun Yang: Investigation, Writing - original draft. **Jiawei Zhang:** . **Xiaozhe Wang:** . **Zhi Li:** Writing - review & editing. **Zhoujun Li:** Writing - review & editing. **Yueying He:** Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China (No. U1636211, No. 61672081, No. 61370126, No. 61906075, No. 61932010), Beijing Advanced Innovation Center for Imaging Technology (Grant No. BAICIT-2016001), Fund of the State Key Laboratory of Software Development Environment (Grant No. SKLSDE-2019ZX-17), Natural Science Foundation of Guangdong Province, China (No. 2019A1515011920).

References

Nguyen, L. A. T., To, B. L., Nguyen, H. K., & Nguyen, M. H. (2014). A novel approach for phishing detection using URL-based heuristic. In *Proceedings of international conference on computing, management and telecommunications* (pp. 298–303). Da Nang, Vietnam.

Xu, L., Zhan, Z., Xu, S., & Ye, K. (2013). Cross-layer detection of malicious websites. In *Proceedings of the 3rd ACM conference on data and application security and privacy* (pp. 141–152).

Xiang, G., Hong, J., Rose, C. P., & Cranor, L. (2011a). Cantina+: A feature-rich machine learning framework for detecting phishing web sites. *ACM Transactions on Information System Security, 14*(2), 21.

Moustafa, N., Misra, G., & Slay, J. (2018). *Generalized outlier Gaussian mixture technique based on automated association features for simulating and detecting web application attacks*. IEEE Transactions on Sustainable Computing. Early Access.

APAC, Briefing on Handling of Phishing Websites in 2017, http://en.apac.cn/Briefing_on_Handling_of_Phishing_Websites/201710/P020171026615980923388.pdf,http://www.cac.gov.cn/files/pdf/cnnic/CNNIC-2014qqzwdiaoyuwzqsEN.pdf.

APWG, Phishing activity trends report for the 1st Quarter of 2018, https://docs.apwg.org/reports/apwg_trends_report_q1_2018.pdf.

Dou, Z., Khalil, I., Khreishah, A., Fuqaha, A. A., & Guizani, A. A. (2017). Systematization of Knowledge (SoK): A Systematic Review of Software-Based Web Phishing Detection. *IEEE Communications Survey & Tutorials, 19*(4), 2797–2819.

Pham, C., Nguyen, L. A. T., Tran, N. H., Huh, E. N., & Hong, C. S. (2018). Phishing-Aware: A Neuro-Fuzzy Approach for Anti-Phishing on Fog Networks. *IEEE Transactions Network and Service Management, 15*(3), 1076–1089.

Labs, M.. Mcafee threats report: Third quarter 2012. Accessed: Nov 2012. http://www.mcafee.com/aulresources/reports/ rp-quarterly-threat-q3-20 12.pdf.

Zhou, E., Chen, Y., Ye, C., Li, X., & Liu, F. (2019). OFS-NN: An Effective Phishing Websites Detection Model Based on Optimal Feature Selection and Neural Network. *IEEE Access, 7*, 73271–73284.

Liu, W., Fang, N., Quan, X., Qiu, B., & Liu, G. (2010). Discovering phishing target based on semantic link network. *Future Generation Computer. Systems, 26*(3), 381–388.

Fu, A. Y., Liu, W., & Deng, X. (2006). Detecting Phishing Web Pages with Visual Similarity Assessment Based on Earth Mover's Distance (EMD). *IEEE Transactions on Dependable Security and Computing, 3*(4), 301–311.

Abusitta, A., Bellaiche, M., Dagenais, M., & HalabiA, T. (2019). deep learning approach for proactive multi-cloud cooperative intrusion detection system. *Future Generation Computer Systems, 98*, 308–318.

Baryannis, G., Dani, S., & Antoniou, G. (2019). Predicting supply chain risks using machine learning: The trade-off between performance and interpretability. *Future Generation Computer Systems, 101*, 993–1004.

Silva, C. M. R. D., Feitosa, E. L., & Garcia, V. C. (2020). Heuristic-based strategy for Phishing prediction: A survey of URL-based approach. *Computer & Security, 88*.

Likarish, P., Jung, E., Dunbar, D., Hansen, T. E., & Hourcade, J. P. (2008). B-APT: Bayesian Anti-Phishing Toolbar. In *Proceedings of international conference on communications* (p. 23). Beijing, China.

Li, Y., Yang, L., & Ding, J. (2016). A minimum enclosing ball-based support vector machine approach for detection of phishing websites. *Optik, 127*(1), 345–351.

Gupta, S., Gupta, B. B., & Chaudhary, P. (2018). Hunting for DOM-Based XSS vulnerabilities in mobile cloud-based online social network. *Computer & Security, 79*(1), 319–336.

Ding, Y., Luktarha, N., Li, K., & Slamu, W. (2019). A keyword-based combination approach for detecting phishing webpages. *Computer & Security, 84*, 256–275.

Chiew, K. L., Tan, C. L., Wong, K., Yong, K. S. C., & Tiong, W. K. (2019). A new hybrid ensemble feature selection framework for machine learning-based phishing detection system. *Information Sciences, 484*, 153–166.

Lin, X., Tang, Y., Tianfile, H., Qian, F., & Zhong, W. (2019). A novel approach to reconstruction based saliency detection via convolutional neural network stacked with auto-encoder. *Neurocomputing, 349*, 145–155.

Tong, C., Lang, C., Kong, F., Niu, J., & Rodrigues, J. J. P. C. (2018). An efficient deep model for day-ahead electricity load forecasting with stacked denoising auto-encoders. *Journal of Parallel Distributed Computing, 117*, 267–273.

Chan, P. P. K., Lin, Z., Hu, X., Tsang, E. C. C., & Yeung, D. S. (2017). Sensitivity based robust learning for stacked auto-encoder against evasion attack. *Neurocomputing, 267*, 572–580.

Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. (2019). Machine learning based phishing detection from URLs. *Expert Systems with Applications, 117*, 345–357.

Smadi, S., Aslam, N., & Zhang, L. (2018). Detection of online phishing email using dynamic evolving neural network based on reinforcement learning. *Decision Support Systems, 107*, 88–102.

Xiang, G., Hong, J., Rose, C. P., & Cranor, L. (2011b). CANTINA+: A feature-rich machine learning framework for detecting phishing Web sites. *ACM Transactions on Information System Security, 14*, 1–28.

Yang, L., Li, Y., & Li, Z. (2017). Improved-ELM method for detecting false data attack in smart grid. *International Journal of Electrical Power and Energy Systems, 91*, 183–191.

Fang, X., Cao, Q., Zhou, Y., & Wang, Y. (2018). Multiscale Compressed and Spliced Sherman-Morrison-Woodbury Algorithm With Characteristic Basis Function Method. *IEEE Transactions on Electromagnetic Compatibility, 60*(3), 716–724.

Scardapane, S., Comminiello, D., Scarpiniti, M., & Uncini, A. (2015). Online Sequential Extreme Learning Machine With Kernels. *IEEE Transactions on Neural Networks and Learning Systems, 26*(9), 2214–2220.

Phishtank, Phishtank developer information. http://www.phishtank.com/developer_info.php, accessed Feb26, 2018.

Feng, D., Bao, Z., & Jiao, L. (1998). Total least mean squares algorithm. *IEEE Transactions on Signal Processing, 46*(8), 122–2130.

Holmes, G., Donkin, A., & Witten, I. H. (1995). WEKA: A machine learning work- bench. *Proceedings of Australian New Zealnd intelligent information systems conference*. Brisbane, Queensland, Australia.

Hinton, G. E., Osindero, S., & The, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation, 18*(7), 1527–1554.

Patil, R. C., & Patil, D. R. (2015). Web spam detection using SVM classifier. *Proceedings of international conference on intelligent systems and control*. Coimbatore, India.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.

İş, H., & Tuncer, T. (2019). Interaction-Based Behavioral Analysis of Twitter Social Network Accounts. *Applied Science, 9*, 4448.

Li, Y., Yang, Z., Chen, X., Yuan, H., & Liu, W. (2019). A stacking model using URL and HTML features for phishing webpage detection. *Future Generation Computer Systems, 94*, 27–39.