Applied Soft Computing Journal 95 (2020) 106505

Contents lists available at ScienceDirect

Applied Soft Computing Journal

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

DTOF-ANN: An Artificial Neural Network phishing detection model based on Decision Tree and Optimal Features

Erzhou Zhu *, Yinyin Ju, Zhile Chen, Feng Liu, Xianyong Fang

Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Computer Science and Technology, Anhui University, Hefei 230601, PR China

ARTICLE INFO

Article history:
Received 26 August 2019
Received in revised form 12 April 2020
Accepted 22 June 2020
Available online 30 June 2020

Keywords:
Phishing detection
Feature selection
Neural network
K-medoids clustering

ABSTRACT

Recently, phishing emerges as one of the biggest threats to human's daily networking environments. Phishing attackers disguise illegal URLs as normal ones to steal user's private information with the social engineering techniques, such as emails and SMS, which calls for an effective method of preventing phishing attacks to relieve the loss by them. Neural networks can be used to detect and prevent phishing attacks because of their strong active learning abilities from massive datasets and high accuracy in data classification. However, duplicate points in the public datasets and negative and useless features in the feature vectors will trap the training of the neural networks into the problem of over-fitting, which will make the trained classifier weak when detect phishing websites. This paper proposes DTOF-ANN (Decision Tree and Optimal Features based Artificial Neural Network) to tackle this shortcoming, which is a neural-network phishing detection model based on decision tree and optimal feature selection. First, the traditional K-medoids clustering algorithm is improved with an incremental selection of initial centers to remove the duplicate points from the public datasets. Then, an optimal feature selection algorithm based on the new defined feature evaluation index, decision tree and local search method is designed to prune out the negative and useless features. Finally, the optimal structure of the neural network classifier is constructed through properly adjusting parameters and trained by the selected optimal features. Experimental results have demonstrated that DTOF-ANN exhibits higher performance than many of the existing methods.

1. Introduction

Phishing is a cybercrime which sends malicious links, usually disguised as legal ones, through spams or social networks to induce users to visit and obtain their private information [1], such as usernames, passwords and social security numbers, and then phishing attackers can obtain the money or other benefits. Phishing is now one of the biggest threats to human's daily networking environments. According to the report of the APWG (Anti-Phishing Working Group), in the first quarter of 2019, there are a total of 180768 phishing attacks be detected globally [2]. What is more, the number of phishing attacks is growing rapidly: As the report finds, the number of phishing attacks monthly increased by 3745% on average from 2004 to the first quarter of 2019. Effective methods and techniques for detecting and preventing phishing attacks are urgently needed.

Typical phishing attacks are launched by sending emails that seem to be normal to end users. These emails usually ask the victims to update their private information by the provided URLs. Other kinds of phishing URLs include peer-to-peer file sharing, blogs, forums, instant message and so on [3]. Perhaps the most direct method is to educate end users to recognize and prevent phishing URLs [4]. However, private information may also be leaked by the unaware or careless behaviors of end users [5], especially considering this increasing number of phishing attacks.

Therefore, there is a growing need for automatic methods to protect users from malicious websites. Existing automatic methods can be divided into four types: blacklisting, heuristic detection, visual similarity checking and machine learning [6]. Blacklisting is to phish URLs which are already detected. It does not analyze the content of phishing websites and thus is of high efficiency. However, this type of method is difficult to cope with the newly emerged URLs [7]. The heuristic detection methods generally assign proper signatures on phishing URLs [8], which will raise warnings if malicious behaviors are detected when scanning the assigned signatures of target websites. They can deal with the newly emerged phishing URLs and generate smaller false positive rate than the blacklisting methods. However, phishing attackers can bypass the constructed filters after acquiring the heuristic policies. In this case, the heuristic detection method will

* Corresponding author.
E-mail addresses: ezzhu@ahu.edu.cn (E. Zhu), yinyinju@gmail.com (Y. Ju), zlchen1109@gmail.com (Z. Chen), fengliu@ahu.edu.cn (F. Liu), fangxianyong@ahu.edu.cn (X. Fang).

https://doi.org/10.1016/j.asoc.2020.106505
1568-4946/

be invalidated [9]. For visual similarity checking which visually compares the potential phishing websites with the original legal ones, if this visual similarity is less than a given threshold, the target website is marked as phishing one. This type of methods is accurate in phishing detection. However, its image matching step may incur huge computational load [10].

Machine learning type of methods are now most extensively studied. Common features, such as the information related to the URLs, the structure of the websites and the features of the JavaScript, are collected first to represent the phishing URLs and their related websites. Then, phishing datasets are collected based on the selected features. Subsequently, the underlying classifiers are trained to detect the phishing websites. Existing classifiers include the Bayes model [11], Support Vector Machine (SVM) [12], Association Roles [13], Logistic Regression (LR) [14], neural networks [15] and so on. The machine learning type can automatically learn to classify data robustly, however, it may include huge number of duplicate points in the public phishing datasets with useless and the negative features in the feature vectors, and thus trap the machine learning method into the problem of over-fitting [16]. Excessive features will also enlarge the scale of the final classifiers, while the useless and the negative features will further degrade the detection accuracy of the final classifiers seriously.

This paper proposes DTOF-ANN, a neural network phishing detection model based on decision tree and optimal feature selection, to tackle these shortcomings. In this model, the original phishing dataset is first refined by the improved K-medoids clustering algorithm. Then, the optimal feature selection algorithm that based on the new defined evaluation index, the decision tree and the local search method is designed to construct the optimal feature vector. Finally, the optimal feature vector is fed to the trained neural network classifier to detect the phishing attacks. Generally speaking, our contributions can be summarized as follows:

(1) ***Improve the traditional K-medoids clustering algorithm for refining phishing datasets***. The traditional K-medoids algorithm requires to specify the number of clusters (K) for selecting the corresponding initial clustering centers randomly. However, as an unsupervised method, this number is rarely unknown beforehand for a dataset. Meanwhile, randomly selection of all the initial clustering centers may lead to poor clustering accuracy and huge calculation. We improve the traditional K-medoids algorithm with an incremental method for selecting the initial clustering centers (medoids). All but the first initial centers are incrementally selected based on the Euclidean distances, so that the number of clusters is no longer set at the beginning of the clustering process. This improved K-medoids clustering algorithm can help obtaining a refined set of training instances that can represent the original dataset well.

(2) ***Propose a new feature evaluation index, f_Value***. Existing machine learning-based phishing detection systems generally use phishing features to represent the target URLs and their related websites for training the underlying classifiers. Different features have different effects on the classifier performance, however, useless and the negative features will seriously degrade the detection accuracy of the final classifiers. Therefore, this paper proposes *f_Value*, a new feature evaluation index to evaluate the impact of different features on the phishing detection. *f_Value* is defined based on the Gini coefficient [17] and the decision tree, and can effectively separate positive, useless and negative features.

(3) ***Design a new feature selection algorithm***. Generally speaking, adequate quantity of features and approach of choosing best ones decides the good performance of a machine learning classifier [18]. Many feature collection methods are proposed, for example, the Off-the-Hook [19] collects more than 200 features related phishing URLs and their related websites. However, excessive features will enlarge the scale and bring complex computation of the final classifier. Furthermore, they may contain useless and negative ones which are harmful to the classifier performance. We propose a new optimal feature selection algorithm based on the new index *f_Value*, decision tree and local search method to select the optimal features for the underlying classifier.

The remainder of this paper is organized as follows: Section 2 discusses the related work. Section 3 details DTOF-ANN with Section 4 presenting the experimental results. Section 5 concludes this paper and outlines the future work.

2. Related work

Many anti-phishing methods from different aspects have been proposed. One of the easiest ways to prevent phishing attacks, blacklisting, is largely used in industries. The Google Safe Browsing API [20] is a representative whose blacklist is constructed on the previously detected phishing websites. It does not provide the protection against zero-hour phishing attacks. Any changes to a stored phishing URL may result in no match. Therefore, this method cannot deal with new emerged phishing attacks. This is also why the blacklist of the Google Safe Browsing API constantly updates. In the academic field, Ma et al. [7] used the learning-based method to update the blacklist.

The heuristic method is also used in phishing detection. The Cantina [12] first extracts the top K words from the page content ranked by the term frequency and inverse document frequency (TF–IDF) metric. Then, the selected top K words are compared with the webpage domain keyword in Google. This model works well for the webpage composed of text content. However, its detection accuracy will degrade when the text in the webpage is replaced by the images. The PhishiNet [21] integrates the blacklisting technique into the heuristic method. It produces multiple variations of the same URL via 5 different URL variation heuristics to resolve the growing change of phishing URLs, i.e. replacing the top-level domain, directory structure similarity, IP address equivalence, query string substitution and brand name equivalence.

For visual similarity-based methods, Zhang et al. [22] made use of the informative spatial layout characteristics of web pages, where the spatial layout features extracted from web pages are organized as rectangle blocks to define page similarity. Phishing webpage can be detected by the page similarity degree. Rao et al. [23] proposed an approach based on the combination of the whitelist and the visual similarity techniques to defend the zero-day phishing attacks. This method extracts discriminative features from websites at first, which are then used to compute the similarity between legitimate and suspicious pages. The visual similarity technique is accurate in phishing detection. But this method cannot properly deal with the continue changing phishing webpages.

Machine learning based methods is widely used in phishing detection, thanks to the strong ability of data processing. Gu et al. [11] used the Bayes model and the SVM model to classify phishing websites based on the extracted features from URLs. Pan and Ding [24] designed an SVM-based algorithm for the phishing detection, which is based on the differences among the

ID of the websites, the structure of the websites and the HTTP transactions. He et al. [25] designed the TF–IDF feature extracting method to obtain 12 features related to the websites and then trained the underlying SVM classifier. Similar SVM model was also proposed by Zouina and Outtaj [26]. Martin et al. [27] constructed a neural network model with features extracted from "Anti-Phishing Working Group" and "PhishTank". Mohammad et al. [28] extracted 18 features on phishing websites to training the deep neural network classifier. Nguyen et al. [29] proposed a dynamic phishing detection method based on the single layer artificial neural network.

Most machine learning methods are based on the feature extraction which may contribute to the high accuracy on phishing detection. For example, the Off-the-Hook [19] extracts more than 200 features. However, excessive number of features will increase the scale of the final classifiers and, consequently, may trap the machine learning methods into over-fitting. Some methods target the optimal features for the underlying classifier. Chiew et al. [30] proposed the hybrid ensemble feature selection method (HEFS) to select the most important features. Toolan and Carthy [31] introduced the information gain selection method to select optimal features from 40 features related to phishing detection, while Zabihimayvan and Doran [32] used the fuzzy rough set to select optimal features. OFS-NN [6] integrates the FVV index to evaluate the importance of each feature. Our proposed DTOF-ANN adopts the artificial neural network for the construction of the underlying classifier. The importance of different features is evaluated first with the novel index *f_Value* and then the optimal feature selection algorithm is designed to construct the optimal feature vector for the underlying neural network classifier.

3. Our proposed DTOF-ANN

Fig. 1 shows the overall workflow of DTOF-ANN, which can be divided into two stages: the training stage and the testing stage. The former stage first introduces the improved K-medoids algorithm to delete the duplicate data points in the open phishing datasets. Then, the optimal feature selection algorithm is designed to construct the optimal feature vector for the underlying neural network classifier, where the new index, *f_Value*, is defined to evaluate the importance of phishing features of data points in the phishing datasets. After, the CART (Classification And Regression Trees) decision tree and the local search method are combined to design the optimal feature selection algorithm. Lastly, datasets with the constructed optimal feature vector for each of its data point is used to train the neural network classifier.

In the latter stage, vectors composed of the optimal features are input to the trained neural network classifier. As shown in Fig. 1, the *Blacklist* module improves the performance, which is constructed by selecting data points from the Alexa website and the PhishTank website. The tested URLs are firstly checked by querying if they are already existed in the Blacklist. The target URLs are directly marked as the phishing ones if so, and, otherwise, the optimal feature vectors of the corresponding URLs are processed by the trained neural network classifier. The detected phishing URLs are also stored in the *Blacklist* to avoid the same kinds of URL in the future processed repeatedly.

3.1. Original phishing datasets refining

Public phishing datasets are usually used to train the underlying classifiers. Here, we adopt the K-medoids clustering algorithm to prune points and thus ease the training workload of the neural network classifier. The number of clusters (K) needs to be set at first in the traditional K-medoids algorithm and then all the corresponding initial clustering centers are randomly selected according to it. However, randomly selection of all the initial clustering centers may bring poor clustering accuracy and high computation expense. The number of clusters in the phishing datasets is usually much bigger than the other kinds of datasets and even a single URL may be taken as a cluster. In addition, public phishing datasets are flooded with duplicate (noise) data points not included in any cluster. So, it is difficult to set K for the phishing datasets before the K-medoids algorithm starts running.

Therefore, we improve the traditional K-medoids algorithm with the incremental method for selecting the initial clustering centers (medoids), i.e., except for the first initial clustering center, the other initial centers are incrementally selected based on the Euclidean distance (Definition 1). Consequently, the number of clusters are no longer set at the beginning of the clustering process.

Definition 1 (*Euclidean Distance*)**.** Suppose the n-data-point phishing sample space $S = \{(X_1, y_1), (X_2, y_2), \ldots, (X_n, y_n)\}$ with each m-attribute data point $X_i = (x_{i1}, x_{i2}, \ldots, x_{im})$ and y_i be the classification result (legal or illegal) of X_i by a phishing website detection tool. The Euclidean distance between point X_i and point X_j can be formulated as

$$d(X_i, X_j) = \sqrt{\sum_p^m \left|x_{ip} - x_{jp}\right|^2} \quad i, j = 1, \ldots, n \ . \tag{1}$$

In the improved algorithm, the first initial clustering center is randomly selected. Then the remaining initial clustering centers are incrementally selected from the existing centers. Specifically, if a data point X_i is taken as a new clustering center, the following condition should to be satisfied:

$$d\left(X_i, Center\right) = max\left\{d\left(X_i, Center_j\right) > 0 | Center_j \in Center\right\}, \tag{2}$$

where: $X_i \in S\text{-}Center$; $Center$ is the set of clustering centers already selected; $d(X_i, Center_j)>0$ for ensuring no duplicate data point is taken as the new clustering center.

The remaining data points can be set to the corresponding clusters by calculating the minimal Euclidean distances with the initial clustering centers of all clusters. Specifically, for a data point X_i belonging to the Cluster C_i, the following condition should to be satisfied:

$$d\left(X_i, Center\right) = min\left\{d\left(X_i, Center_j\right) > 0 | Center_j \in Center\right\}, \tag{3}$$

where $d(X_i, Center_j)>0$, ensuring that no duplicate data point is put to a cluster.

The final center of each cluster is adjusted with the initial clustering center and the remaining data points, where the initial center is replaced with the data point that has the minimal distance to all the other data points of this cluster.

Now the improved K-medoids clustering algorithm (Fig. 2) to refine the original phishing datasets can be obtained from the above analysis. In this algorithm, steps (1) and (2) randomly estimate the first initial cluster center. Step (3) computes the remainder initial centers, where the remainder initial clustering centers are incrementally selected as far as from the existing centers. "$d(X_i, Center)>0$" in the "*if* "condition is imposed so that the remaining duplicate data points of the selected initial centers are not selected as new clustering centers. For example, the distance between X and its duplicate point X' ($d(X, X')$) is 0 if a data point X is selected as the clustering initial center. X' cannot been selected as a new center according to $d(X_i, Center)>0$. Once the cluster centers are specified, step (4) puts the other points to the corresponding clusters. "$d(X_i, Center_i)>0$" in the "*if* "condition of step (4) is set with the same reason as step (3). Finally, step (5) updates the final cluster centers by re-computing the distance of all points in different clusters.

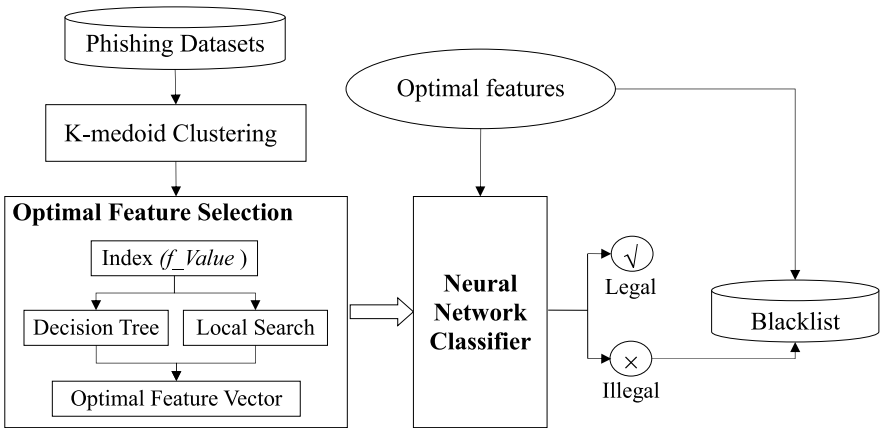

Fig. 1. The overall workflow of DTOF-ANN.

Input: Phishing dataset $S=\{(X_1,y_1), (X_2,y_2), \ldots, (X_n,y_n)\}$, $X_i=(x_{i1}, x_{i2}, \ldots, x_{im})$;
Output: S is divided into K clusters $S=C_1 \cup C_2 \cup \ldots \cup C_K$;
The corresponding clustering centers are $Center=\{Center_1, Center_2, \ldots, Center_K\}$.

(1) Randomly select a data point X (from dataset S) as the first initial clustering center ($Center_1$);
(2) $Center \leftarrow Center \cup \{Center_1\}$; //Put the first cluster center into the set of *Center*. Initially, the *Center* set is empty.
$K \leftarrow 1$; //K specifies the number of clusters.
(3) while (not all n data points been processed) do { //Incrementally find the remainder initial cluster centers.
$max \leftarrow 0$;
for each point X_i in *S-Center* do {//Find point X with the largest distance form *S-Center* to *Center*.
Use Equation (1) and (2) to compute the distance $d(X_i, Center)$;
if ($d(X_i, Center)>0$ and $d(X_i, Center)>max$) {$max \leftarrow \{d(X_i, Center)$; $X \leftarrow X_i$; }
}; //endfor
$Center \leftarrow Center \cup \{X\}$; // Put the newly found initial cluster center into the set of *Center*.
$K \leftarrow K+1$;
}; //endwhile
(4) for each point X_i in *S-Center* do { //Put the remaining data points into the corresponding clusters.
$min \leftarrow \infty$; $Tag \leftarrow 0$; // The "*Tag*" is used to specify which cluster the point X_i should be placed.
for each point $Center_j$ in *Center* do {
use formula (1) to compute $d(X_i, Center_j)$;
if ($0<d(X_i, Center_j)<min$) { $min \leftarrow d(X_i, Center_j)$; $Tag \leftarrow j$; }
}; //endfor
$C_{Tag} \leftarrow C_{Tag} \cup \{X_i\}$; // Put data point X_i into cluster C_{Tag}.
}; //endfor
(5) for each cluster C_k in $\{C_1, C_2, \ldots, C_K\}$ do { // Update the cluster centers.
$min \leftarrow \infty$; $Tag \leftarrow 0$;
for each point X_i in C_k do {
$sum \leftarrow \sum_{X_j \in C_k - \{X_i\}} d(X_i, X_j)$;
// Use Equation (1) to compute the sum of distances between X_i and the remaining point in C_k-$\{X_i\}$.
if $sum < min$ { $min \leftarrow sum$; $Tag \leftarrow i$;} // Find the minimum distance.
}; //endfor
if $min < \sum_{X_j \in C_k - \{Center_k\}} d(Center_k, X_j)$
$Center_k \leftarrow X_{Tag}$; // Update the cluster center C_k;
}. //endfor

Fig. 2. The improved K-medoids algorithm for refining phishing datasets.

3.2. Optimal feature selection

In this part, the new index (*f_Value*) based on the Gini coefficient [17] and CART algorithm is firstly defined to measure the feature importance on the phishing detection. Then, the new algorithm that combines CART and the local search method is designed to construct the optimal feature vector based on this new index. This vector will be taken as the input to the underlying neural network classifier.

3.2.1. Classification of features

We classify phishing features into four categories, the address bar features, the abnormal behavior features, the HTML and JavaScript features, and the domain features. In the training phase of DTOF-ANN, optimal subset of these features is extracted from the datasets and stored in the feature vectors for training the network. In the testing phase, optimal features are fed to the neural network to determine behavior (phishing or legal) of the target website.

For DTOF-ANN, there is no limitation on the number of features as long as they fall into the above four categories. Presently, 30 phishing features from the four categories are collected and stored in the vector of $F = <f_1, f_2, \ldots, f_{30}>$. For each feature f_i ($i \in [1,30]$), its value may be one of the three, -1, 0 and 1, which means legal, potential phishing and most likely phishing respectively.

(1) *Address bar features.* The address bar features are directly or indirectly extracted from the website URLs. Many of the phishing attacks are launched by taking advantages of these features. We collect 12 address bar features, $f_1, f_2, \ldots, f_{12}$, stored in the vector *F*.

- *URL containing IP address (f_1).* Phishing attackers usually replace the domain names with the IP addresses to avoid registering domain names. If the domain name of a website's URL is an IP address, this website is a potential phishing one and, consequently, f_1 is set to 1. Otherwise, it is set to -1.
- *Long URL (f_2).* Phishing attackers usually use long URLs to hide the suspicious information. Generally, the length of a typical normal URL is less than 55 characters. So, if the length of the target URL is in the interval [0, 54], f_2 is set to -1. If this length is in the interval [55, 75], f_2 is set to 0. Otherwise, f_2 is set to 1.
- *Short URL (f_3).* Phishing attackers can reduce the length of the URL at first. Then, suspicious websites can be linked by the "http redirection" of the short domain name of the URL. So, if the target URL uses the short address service, f_3 is set to 1. Otherwise, it is set to -1.
- *URL including '@' (f_4).* The URL of a normal website does not include an '@'. If the target URL includes the '@', f_4 is set to 1. Otherwise, it is set to -1.
- *Position of the '//' (f_5).* URLs are started with the strings of "http" or "https". Based on the structures of normal URLs, the '//' is situated at the 7th position of the URLs. So, if the '//' is situated behind the 7th character of the target URL, f_5 is set to 1. Otherwise, it is set to -1.
- *URL including '-' (f_6).* Legal URLs rarely contain a '-'. Phishing attackers usually hide the '-' in the domain names to deceive the users as the normal websites. If the target URL contains the '-' in the domain names, f_6 is set to 1. Otherwise, it is set to -1.
- *Number of '.' in the main domain name (f_7).* The number of '.' in the main domain name specifies the number of subdomains of the target URL. More subdomains mean more suspicious of the target website. If this number is 1, 2, or more than 2, f_7 is set to -1, 0 or 1 respectively.
- *http protocol and SLL certificate statues (f_8).* Each time the sensitive information is transmitted, the legal websites will use the safety domain names. The SLL is the digital certificate in the server. However, phishing websites usually have no SLL certificate. If the target website uses http protocol

and the age of the corresponding SLL certificate issued by the trusted or untrusted person is no less than 1 year, f_8 is set to -1 or 0, respectively. Otherwise, f_8 is set to 1.

- *Expiration time of registered domain name (f_9).* Most of the phishing websites are facing the problem of domain names expiration due to short lifecycles. If the domain expiration time is no more than 1 year, the corresponding website can be taken as a phishing website, i.e., f_9 is set to 1. Otherwise, f_9 is set to -1.
- *Sources of "favicon/icon" (f_{10}).* Icons of a legal website are generally from the same domain. If the icons are loaded from the other domains other than from the domains specified by the target URL, the corresponding website is a potential phishing website with f_{10} set to 1. Otherwise, f_{10} is set to -1.
- *Opening ports of websites (f_{11}).* Normally, most of ports are closed by firewalls, proxy servers and NAT (Network Address Translation) servers with only the selected ports opened. The IDs of the default closed port are: 21 (File Transfer Protocol), 22 (Secure Shell), 23 (Telnet), 445 (Server Message Block), 1433 (MSSQL Database), 1521 (Oracle Database), 3306 (MySQL Database) and 3389 (Remote Desktop). The target website may be a phishing site if the corresponding ports listed above are opened. f_{11} is used to specify this feature: 1 for opened with -1 for closed.
- *Domain names of URL including "https" (f_{12}).* URLs of the target website with domain names contain the "https" are suspicious. If the domain names contain this string, f_{12} is set to 1. Otherwise, f_{12} is set to -1.

(2) *Abnormal behavior features.* Abnormal behavior features can be extracted by analyzing the source codes of the corresponding webpages. 6 abnormal behavior features, f_{13}, f_{14}, …, f_{18}, are extracted and stored in *F*.

- *Source of requested objects (f_{13}).* Most of objects (graphics, videos and sounds) share the same domain in the legal websites. Objects from a greater number of different domains means more dangerous for the corresponding websites. If the ratio of different request objects in a website is smaller than 21% of all domains, f_{13} is set to -1; if this ratio is less than 61%, f_{13} is set to 0; and, otherwise, it is set to 1.
- *Direction of "Anchor" (f_{14}).* An "Anchor" is defined by the <a> tag in the HTML. Generally, a greater number of different domain names of elements in the <a> tags means more dangerous for the target website. If the ratio of "Anchor" is less than 31%, f_{14} is set to -1; if this ratio is greater than 67%, f_{14} is set to 1; and, otherwise, f_{14} is set to 0.
- *Links in tags of HTML documents (f_{15}).* Links in the tags, such as <meta>, <script> and <link>, are from a small number of domains in the normal websites. If the links in these tags point to many different domains, the target website is suspicious. If the ratio of different links in these tags is smaller than 17%, f_{15} is set to -1; if this ratio is greater than 81%, f_{15} is set to 1; and, otherwise, f_{15} is set to 0.
- *Contents of the SFH (f_{16}).* If the SFH (Server Form Handler) includes the null character or the "about:blank" string, the target website is suspicious and consequently, f_{16} is set to 0; if the domain names in the SFH are different from those of the target website, this website is dangerous and then f_{16} is set to 1; and, otherwise, f_{16} is set to -1.
- *Email in the web server script (f_{17}).* Web forms allow the personal information in the server to be submitted to end users. However, phishing attackers may use this feature to steal sensitive information. By embedding functions of "mail()" or "mailto()" in the web server script, sensitive information can be redirected to the emails of the phishing attackers. If the target website uses the two functions, f_{17} is set to 1. Otherwise, it is set to -1.
- *"whois" information (f_{18}).* "whois" information of website inquires the transport protocols of the IP of the domain names and the information of the owners. If the information in the "whois" database matches the ones of the target website, f_{18} is set to -1. Otherwise, it is set to 1.

(3) *HTML and JavaScript features.* These features are extracted from the source codes of the HTML and JavaScript documents. 5 HTML and JavaScript features, f_{19}, f_{20}, …, f_{23}, are extracted and stored in *F*.

- *Number of webpage's redirections (f_{19}).* Phishing attackers can use the page redirections to create the links pointing to the malicious webpages. The number of page redirections in a normal website is less than 2. So, if the number of page redirections is more than 3, this feature (f_{19}) is set to 1; if this number is less than 1, f_{19} is set to -1; and, otherwise, f_{19} is set to 0.
- *Existence of "onMouseOver" event (f_{20}).* Phishing attackers can use JavaScript to display the forged URL in the status bar to cheat users. If the source code of the target website contains the "onMouseOver" event with a changed status bar, this website can be taken as a phishing website and f_{20} is set to 1. Otherwise, f_{20} is set to -1.
- *Right click of mouse operation (f_{21}).* Phishing attackers usually prohibit the mouse right click operation to disguise the source codes of the malicious websites. If the right click operation is prohibited (event.button==2), f_{21} is set to 1. Otherwise, it is set to -1.
- *Usage of "pop-up window" (f_{22}).* Phishing attackers can acquire the private information of end users through the "pop-up window". If it is used, f_{22} is set to 1. Otherwise, it is set to -1.
- *Attribute of "IFrame" (f_{23}).* "IFrame" is a HTML tag used to display the contents of other websites to the current one. Phishing attackers can use the "IFrame" to hide these contents. If this tag is used, f_{23} is set to 1. Otherwise, f_{23} is set to -1.

(4) *Domain features.* The domain features are collected by checking the domain names of URL of the target websites. 7 domain features, f_{24}, f_{25}, …, f_{30}, are collected and stored in *F*.

- *Age of registered domain names (f_{24}).* Phishing websites generally have short lifecycles. After checking the "whois" database of the domain names, we find that the age of domain names of legal websites are more than 6 months. If the age of the registered name is shorter than 6 months, f_{24} is set to1. Otherwise, it is set to -1.
- *DNS record (f_{25}).* For a phishing website, the "whois" database cannot find the corresponding DNS record it declared before. If the DSN record is empty or cannot be found in the "whois" database, the target website is a potential phishing website and then f_{25} is set to 1. Otherwise, f_{25} is set to -1.
- *Traffic of websites (f_{26}).* Due to short lifecycles, the visit number of a phishing website is relatively small. Alexa database records the global comprehensive ranking of website traffics. If the Alexa ranking of the target website is the top 100 000, f_{26} is set to -1; if the target website is in the Alexa database but not the top 100 000 websites, f_{26} is set to 0; and, otherwise, it is set to 1.

- *PageRank of websites (f_{27}).* The PageRank is an important index on evaluating the importance of a website. The value of PageRank is in the interval [0, 1]. The bigger value of the PageRank, the more important of the target website is. Most of the phishing websites have no PageRank. Therefore, for a given website, if the PageRank value is smaller than 0.2, f_{27} is set to 1. Otherwise, f_{27} is set to -1.
- *Google index of websites (f_{28}).* The website included by the Google search engine can be appeared in the search results. However, phishing websites are rarely indexed by the Google due to short lifecycles. If the target website is included in the Google index, f_{28} is set to -1. Otherwise, f_{28} is set to 1.
- *Number of links pointing to target webpages (f_{29}).* Phishing websites generally have no link pointing to its own webpage. If the number of links pointing to the target webpage is 0, f_{29} is set to 1; if this number is smaller than 2, f_{29} is set to 0; and, otherwise, it is set to -1.
- *PhishTank and StopBadware statistical reports (f_{30}).* The PhishTank and the StopBadware periodically publish the newly emerged phishing websites, and thus it is important to check whether the URL of the target website is included the two statistical reports. If the target URL is in the two reports, f_{30} is set to 1. Otherwise, it is set to -1.

3.2.2. f_Value index of features

As an impurity splitting method, the Gini coefficient is widely used in the decision tree algorithms [17]. In this subsection, the Gini coefficient is firstly extended to process phishing features. Then, the decision tree can be generated by CART based on the extended Gini coefficient of different phishing features. Finally, the new index *f_Value* is defined to evaluate the importance of phishing features. The Gini coefficient can be defined as follows:

Definition 2 (*Gini Coefficient*)**.** Suppose the sample space S be divided into K classes and $p_k$be the proportion of data points in class k. The Gini coefficient of the sample space S is defined as

$$Gini\,(S) = 1 - \sum_{k=1}^{K} p_k. \tag{4}$$

For phishing detection, the phishing sample space S is only divided into two classes, the phishing ones and the legal ones. Therefore, the value of K is fixed to 2. The Gini coefficient can be extended to evaluate the importance of phishing features.

Definition 3 (*Gini Coefficient of Phishing Features*)**.** Suppose $A = \{a_1, a_2, \ldots, a_m\}$ be the phishing feature set of m data points in the phishing website sample space S. A given feature $a \in A$ has V possible values as $\{a^1, a^2, \ldots, a^V\}$. The CART algorithm will generate V branch nodes if the feature a divides S, where branch v ($v = 1, 2, \ldots, V$) contains a set (marked as S^v) of data points with the attribute a equal to a^v. Consequently, the sample space S is divided into V classes as $\{S^{v^1}, S^{v^2}, \ldots, S^{v^V}\}$. For each element S^v in the set of $\{S^{v^1}, S^{v^2}, \ldots, S^{v^V}\}$, the Gini coefficient of S^v can be computed by Eq. (4). Different branch node contains different number of data points and, therefore, the weight of the branch node v ($v = 1, 2, \ldots, V$) is assigned as $|S^v|/|S|$. Accordingly, a branch node will get the higher weight value if it contains more data points. Then the Gini coefficient of the feature a can be defined as:

$$Gini(S, a) = \sum_{v=1}^{V} \frac{|S^v|}{|S|} Gini(S^v). \tag{5}$$

According to Eq. (5), the Gini coefficients of A can be computed at first. Then, the optimal division feature$a_*$with the smallest Gini coefficient can be:

$$a_* = min_{a \in A}\{Gini\,(S, a)\} \tag{6}$$

Now the decision tree can be generated by CART with the extended Gini coefficient of different phishing features. CART is widely used in data mining as a reprehensive decision tree learning method, which adopts the dichotomous recursive technique to divide the current sample space into two disjoint subsets. Thus, a binary tree is generated with each non-leaf node having two branches. Fig. 3 shows the part of the decision tree that utilizing CART on a phishing dataset (the detailed description of this dataset (Dataset 1) will be described in Section 4.1). As can be seen, the branch node with the biggest extended Gini coefficient is taken as the root of the entire binary tree. We now discuss the extended Gini coefficient-based *f_Value*.

Definition 4 (*Index of Features, f_Value*)**.** To better evaluate the importance of different features in the phishing detecting, the new index, *f_Value*, of a feature a, is defined as follows:

$$f\_Value(a) = (N * Gini\,(S, a) - N_l * Gini_l - N_r * Gini_r)/|S|\,, \tag{7}$$

where: a is the root of the decision tree; N is the number of data points in the root node a; $Gini_l$ and $Gini_r$ are the Gini coefficients of the left child and right child of the root node a respectively; N_l and N_r are the numbers of data points in the corresponding nodes respectively; $|S|$ specifies the total number of data points in the sample space S.

f_Value is taken as the criteria for selecting optimal features. The bigger *f_Value* of a feature, the more important of this feature in the phishing detection. If *f_Value* is 0, this feature is taken as a useless feature. As defined in Eq. (7), the *f_Values* (include those of the negative features) are no less than 0. So, negative features cannot individually be determined by the index values.

Definition 5 (*Index for Negatives, ρ*)**.** The index ρ for the feature selection is defined as follows to effectively find negative features:

$$\rho = \frac{|Times_{Tem} - Times_{ori}|}{acc_{Tem} - acc_{ori}} \tag{8}$$

where: $Time_{ori}$ and Acc_{ori} are the time cost and accuracy of DTOF-ANN when the original features set are utilized respectively; $Time_{tem}$ and Acc_{tem} are the time cost and accuracy of the DTOF-ANN respectively when another feature is put to the original features set. According to Eq. (8), Acc_{tem}-Acc_{ori} will smaller than 0 when a negative feature is put to the original features set.

Combining indexes of *f_Value* and ρ will lead to delete the negative features. As will be verified in Section 4.4, the index values of negative features are much smaller than those of positive features. Therefore, features with relatively small *f_Value* are selected at first. Then, ρ is used to determine whether they are the negative features.

3.2.3. Algorithm for constructing optimal sensitive vector

As discussed previously, some of the useless and small impact features will bring huge computation or even over-fitting to the training of the underlying neural network classifier. In addition, some negative features will degrade the performance of the final classifier. To prune out these features, we design the new algorithm based on the combination of CART and the local search to select the optimal features. The phishing datasets with optimal feature vector for each data points are taken as the input for training and testing the neural network classifier.

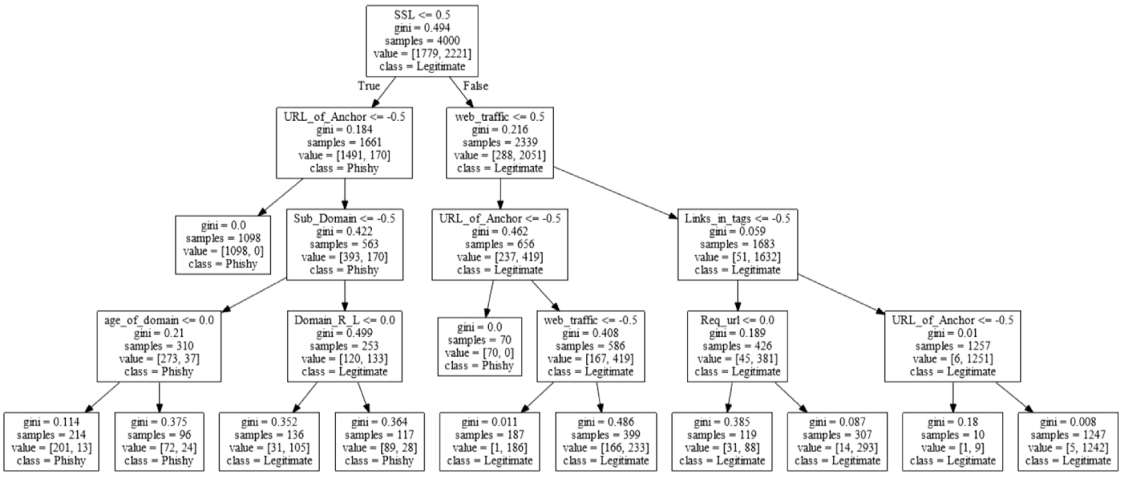

Fig. 3. A decision tree generated by the extended Gini coefficient and CART.

In the new algorithm, CART firstly calculates the index values of the entire collected features based on the *f_Value* Eq. (7). Consequently, a set of features with the highest *f_Value* are selected as the initial optimal features set. Then, the local search method based on the K-Nearest Neighbor (KNN) is used to get the accuracy of the initial optimal features set. The accuracy of the initial optimal features set is taken as the initial solution of the local search algorithm.

The local search algorithm is often used to find the optimal solution to optimization problems, which iterates the adjacent solutions so that the objective function can be optimized step by step until convergence [33]. The local search algorithm constructs the final optimal feature vector for the underlying neural network classifier by sequentially analyzing the remainder collected features. Specifically, the pseudo code of our new algorithm is shown in Fig. 4, which will be discussed below.

Step (1) computes the *f_Values* of all the features of data points in S by CART. Step (2) gets the initial optimal features and stores them in the set X_{ori}. The remainder features are put to the set X_{rem} by the step (3), while step (4) uses the KNN to train the initial optimal features and stores the trained accuracy to acc_{ori}. Step (5) updates the optimal X_{ori} by sequentially selecting positive features from the X_{rem}. If the newly added feature increases the accuracy of the original X_{ori}, this feature is positive and put to X_{ori}. Otherwise, this feature is negative and will not put to X_{ori}. Finally, the optimal feature vector F_{opt} is constructed in step (6) based on the selected optimal feature set. This vector will be taken as the input to the underlying neural network classifier.

3.3. The neural network classifier

The classifier of the DTOF-ANN model is constructed on ANN (Artificial Neural Network) [15] which is generally composed of three parts: the input layer, one or more hidden layers and the out layer. The input layer calculates the weights of the put data with the hidden layers performing the learning process, while the output layer generates the final results of ANN. The learning process of ANN is composed of two procedures: the forward propagation and the backward propagation. Once the activation function detects the input from the input layer, the forward propagation processes the data in the hidden layers and generates the result to the output layer. The backward propagation starts to work when it detects the big deviation between the actual output results and the expectation values. The backward propagation iteratively adjusts parameters of neural nodes in different layers until the actual results approach the expected ones.

Input: Phishing website training set $S=\{(x_1,y_1), (x_2,y_2), \ldots, (x_n,y_n)\}$; Feature set $A=\{a_1, a_2, \ldots, a_m\}$.
Output: Optimal feature vector $F_{opt}=<f_1,f_2,\ldots,f_q>$.

(1) Compute *f_Value*s of all the features of data points in S by CART;
(2) Select 10 features with the highest *f_Value* index values and store them in the set of X_{ori};
(3) Put the remaining features (based on *f_Value*) into the set of X_{rem} subsequently;
(4) $acc_{ori} \leftarrow$ KNN(X_{ori});
// Use the KNN to train the initial optimal features and obtain the trained accuracy acc_{ori}.
(5) for each feature f in X_{rem} do { // Incrementally add the remainder optimal features.
Tempset $\leftarrow X_{ori} \cup \{f\}$;
$acc_{Tem} \leftarrow$ KNN(Tempset);
if $acc_{Tem} > acc_{ori}$ { // Update the accuracy and the optimal feature set.
$Acc_{ori} \leftarrow acc_{Tem}$;
$X_{ori} \leftarrow acc_{Tem}$;
}
} //End for.
(6) $F_{opt} \leftarrow X_{ori}$.
// Convert the final optimal feature set X_{ori} to the vector and obtain the optimal feature vector F_{opt}

Fig. 4. Algorithm of optimal feature vector construction.

Generally, more layers will bring better classification performance. However, more layers may also trap the trained classifier into over-fitting. Therefore, neural network models with different hidden layers and different number of neuron units are tested to train the optimal structure of the underlying classifier. The optimal structure of the neural network for the phishing detection can be obtained with iteratively adjusting parameters. Fig. 5 shows part of the results of the experiments. The x-axis represents the number of neuron units in the corresponding hidden layers with the y-axis being the accuracy (this metric will be defined in Section 4) of the neural network. The tested dataset (Dataset 1) used in this experiment will be presented in Section 4.

The blue polyline in Fig. 5 records the accuracy of the neural network of the first hidden layer when it incorporates 2 to 60 neuron units. The highest accuracy (0.967) is obtained when the first hidden layer incorporates 56 neuron units, therefore, the number of neuron units in this layer is set to 56. The number of neuron units in the second hidden layer is limited to the interval [2, 34] based on the configuration of the first hidden layer. As the orange polyline shown, the highest accuracy (0.969) is obtained when the second hidden layer incorporates 28 neuron units. So, the number of neuron units in this layer is set to 28. Based on the configurations of the first and the second hidden layers, the number of neuron units in the third hidden layer is limited to the interval [2, 20]. As the gray polyline shown in Fig. 5, the highest accuracy (0.971) is obtained when the third hidden layer incorporates 10 neuron units. So, the number of neuron units in this layer is set to 10.

According to the above analysis, the underlying classifier of the DTOF-ANN model (Fig. 6) is designed as a five-layer structure

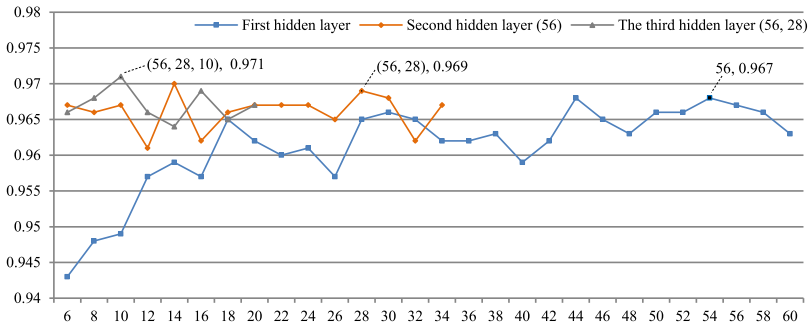

Fig. 5. Accuracy of the neural classifier under different hidden layers.

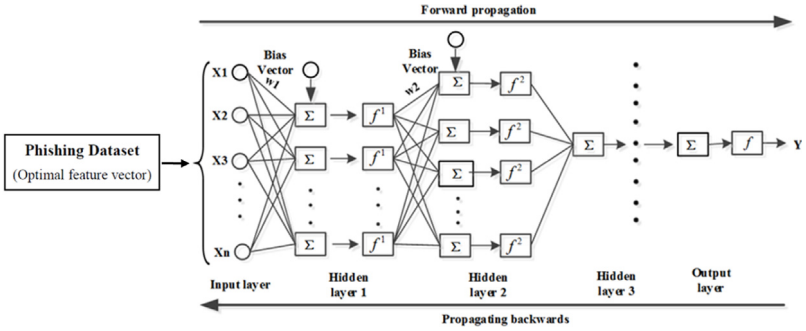

Fig. 6. The architecture of the neural network classifier of the DTOF-ANN model.

(One input layer, three hidden layers and one output layer) of artificial neural network with 56, 28 and 10 neuron units in three different hidden layers respectively. It can be seen that optimal vectors of URLs are taken as the input of the classifier. The main tasks of the forward propagation and the backward propagation of the classifier are as follows:

(1) *Forward propagation.* Phishing datasets with the selected optimal features are input to the neural network. The input datasets are generally divided into two parts: the training set and the testing set. The training set is used to get the optimal structure of the neural network, while the testing set is used to test the performance of the entire phishing website detection model.

Suppose the training set expressed as $S = \{(X_1, y_1), (X_2, y_2),\ldots, (X_n, y_n)\}$, where: $X_i \in R^m$ is the input data point; $y_i \in R^l$ is the corresponding detection result of X_i; H is the number of neuron units in the hidden layers; and the weights and offsets are randomly generated. Once the activation function detects the input from the input layer, the output matrix of the hidden layer neuron units can be calculated as follows:

$$\mathrm{H}_{\mathrm{sum}} = \begin{bmatrix} f(w_1 * X_1 + b_1) & \cdots & f(w_H * X_1 + b_H) \\ \vdots & \ddots & \vdots \\ f(w_1 * X_n + b_1) & \cdots & f(w_H * X_n + b_H) \end{bmatrix}, \tag{9}$$

where: $w_j = (w_{j1}, w_{j2},\ldots, w_{jH})|\ (j = 1,2,\ldots, H)$ is the randomly generated weight vector; $b_j\ (j = 1,2,\ldots, H)$ represents the number of the neuron units in the jth hidden layer; n is the total number of input data points; $f(\blacksquare)$ is the activation function defined as $f = max(0, x)$. The output layer of neural network X_i can be represented as:

$$O_i = \sum_{j=1}^{H} \beta_j f\left(w_j * X_i + b_j\right) \quad i = 1, 2, \ldots, m, \tag{10}$$

where $\beta_j = (\beta_{j1}, \beta_{j2},\ldots, \beta_{jl})$ is the jth weight on connecting the jth neuron unit of the hidden layer and the neuron units of the output layer.

(2) Backward propagation. For a given data point (X_k, y_k) in the training set S, suppose the output of the neural network be $\hat{y}_k = (\hat{y}_1^k, \hat{y}_2^k, \ldots, \hat{y}_l^k)$. The following relationship is

$$\hat{y}_j^k = f\left(\beta_j - b_j\right). \tag{11}$$

Then, the mean square error of the data point (X_k, y_k) in the neural network can be computed as:

$$E_k = \frac{1}{2}\sum_{j=1}^{l}\left(\hat{y}_j^k - y_j^k\right)^2. \tag{12}$$

Subsequently, the weight of the data point (X_k, y_k) is updated as:

$$\begin{aligned} w_j &\leftarrow w_j + \Delta w_j \\ \Delta w_j &= \gamma\left(y_k - \hat{y}_k\right) X_k \end{aligned}, \tag{13}$$

where $\gamma \in (0, 1)$ is the learning rate.

Accordingly, if the neural network gets the correct prediction on the data point (X_k, y_k), i.e. $\hat{y}_k = y_k$, the neural network will not change.

4. Experimental results

The improved K-medoids is firstly tested to verify the effect of removing duplicate data points from the original phishing dataset. Then, the performance of the optimal feature selection algorithm and the overall performance of DTOF-ANN are tested subsequently. The performance of DTOF-ANN is compared with those of several existing phishing detection model. Experiments here are performed on a computer with Intel i5 CPU (3.4 GHz), DDR3 RAM (16 GB) and Windows 10 operation system (64 bit). Meanwhile, experimental programs are written by the Python programming language.

Table 1
Detailed description of phishing features of the two datasets.

Features: Values	Value distributions (Dataset 1)						Value distributions (Dataset 2)					
	−1		0		1		−1		0		1	
	P	*L*	*P*	*L*	*P*	*L*	*P*	*L*	*P*	*L*	*P*	*L*
f_1: {−1, 1}	1926	1867	–	–	2972	4290	638	5665	–	–	838	7441
f_2: {−1, 0, 1}	4079	4881	83	52	736	1224	850	8574	121	348	505	4184
f_3: {−1, 1}	626	818	–	–	4272	5339	468	4156	–	–	1008	8950
f_4: {−1, 1}	837	818	–	–	4061	5339	551	4893	–	–	925	8213
f_5: {−1, 1}	562	867	–	–	4336	5290	227	2016	–	–	1249	11090
f_6: {−1, 1}	4692	4898	–	–	1645	0	1291	11463	–	–	185	1643
f_7: {−1, 0, 1}	1836	1527	2252	1370	810	3260	386	3434	295	2619	795	7053
f_8: {−1, 0, 1}	3051	506	1146	21	701	5630	862	7654	0	0	794	5452
f_9: {−1, 1}	2690	4699	–	–	2208	1458	795	7095	–	–	681	6011
f_{10}: {−1, 1}	909	1144	–	–	3989	5013	378	3356	–	–	1098	7950
f_{11}: {−1, 1}	734	768	–	–	4164	5389	290	2575	–	–	1186	10531
f_{12}: {−1, 1}	715	1081	–	–	4183	5076	314	2788	–	–	1162	10318
f_{13}: {−1, 0, 1}	2675	1820	0	0	2223	4337	736	6535	128	1137	612	5434
f_{14}: {−1, 0, 1}	3246	36	1502	3835	150	2286	847	7521	385	4724	244	861
f_{15}: {−1, 0, 1}	2387	1569	1744	2705	767	1883	688	3419	532	4723	256	4964
f_{16}: {−1, 0, 1}	4238	4202	236	498	397	1457	796	7068	116	1030	564	5008
f_{17}: {−1, 1}	931	1083	–	–	3967	5074	265	2353	–	–	1211	10753
f_{18}: {−1, 1}	604	1025	–	–	4294	5132	433	3845	–	–	1043	9621
f_{19}: {−1, 0, 1}	0	0	4296	5480	602	677	589	5230	716	6356	171	1520
f_{20}: {−1, 0, 1}	657	658	0	0	4241	5499	337	2992	25	988	1114	9126
f_{21}: {−1, 1}	225	251	–	–	4673	5906	573	5476	–	–	903	7630
f_{22}: {−1, 1}	947	1190	–	–	3951	4967	289	2566	–	–	1187	10540
f_{23}: {−1, 1}	443	569	–	–	4455	5588	418	3712	–	–	1058	9394
f_{24}: {−1, 1}	2632	2557	–	–	2266	3600	798	7006	–	–	678	6100
f_{25}: {−1, 1}	1718	1725	–	–	3180	4432	469	4164	–	–	1007	8942
f_{26}: {−1, 0, 1}	1673	982	1718	851	1507	4324	587	5212	245	1095	644	6799
f_{27}: {−1, 1}	3885	4316	–	–	1013	1841	843	7485	–	–	633	5621
f_{28}: {−1, 1}	927	621	–	–	3971	5545	652	5789	–	–	824	7317
f_{29}: {−1, 0, 1}	193	355	2929	3227	1776	2575	265	2353	476	4226	735	6527
f_{30}: {−1, 1}	839	711	–	–	4059	5446	772	6855	–	–	704	6251

4.1. Description of the phishing dataset

At present, the performances of many existing phishing detection models are verified by their own designed or collected datasets (such as the tools listed in Table 8). However, many self-defined features may invalid for newly emerged phishing websites due to short life cycles and fast update speed of phishing websites. For fairness, two public phishing datasets (described in Table 1), marked by Dataset 1 and Dataset 2, are used to test the performance of DTOF-ANN.

Dataset 1 is downloaded from the UCI library collected by Mohammad, Thabtah and McCluskey [34]. This dataset has multiple sources, including PhishTank, MillerSmiles, Google, Yahoo and Starting Point catalogue. It is composed of 11055 data points with 4898 (44.31%) legal websites and 6157 (55.69%) phishing websites. 70% of the data points in this dataset are used for training while the remainder 30% points are used for testing [30]. Dataset 2 are crawled from the PhishTank [35] and Alexa [36] records during the period of December 2017 to June 2018. Since many real data (such as graphics, files and videos) are contained in the original crawled datasets, URLs with the file type suffixes (.jpg, .zip, .mp4, and so on) are deleted. Dataset 2 is constructed after refinement, which is composed of 14582 data points with 1476 (10.12%) phishing points.

Dataset 1 is used to test performance of DTOF-ANN from several aspects, including the performances of the improved K-medoids algorithm, different features on the phishing detection, and the optimal feature selection algorithm. Dataset 2 is used to test the performance of the entire DTOF-ANN model.

Table 1 lists the details of phishing features of the two datasets. The meaning of f_1~f_{30} are introduced in Section 3.2.1 of this paper. The value of each feature can be −1, 0 or 1. The distribution of the feature value is also provided. For example, for $f_2$in Dataset 1, there are 4079 phishing websites (marked by "*P*") and 4881 legal websites (marked by "*L*") when f_2 is −1; there are 83 phishing websites and 52 legal websites when f_2 is 0; and there are 736 phishing websites and 1224 legal websites when f_2 is 1.

As can be seen from this table, for some features, the distributions of data points vary significantly for different values. For example, f_{21} of Dataset 1 can be −1 or 1 with only 476 (225+251) data points being −1 and 10579 (4673 + 5906) data points being 1. Compared with the number of data points valued 1, the number of data points valued −1 is negligible. Therefore, we can conclude that this feature has little effect on the final classification results. This table also shows that a single feature cannot completely distinguish legal websites and phishing websites.

4.2. Metrics for the performance evaluation

The confusion matrix is used to evaluate the performance of DTOF-ANN as usual. The actual values and the prediction values can be compared with the confusion matrix which is defined as Table 2, considering that the phishing detection in this paper can be represented as a 2-way problem (the phishing websites and the legal websites). Here, the phishing websites are taken as positive samples with the legal websites as negative ones.

Metrics *Precision*, *Recall*, *F1_score* and *Accuracy* can be defined for comprehensively evaluating the performance of DT-NN with the *TP*, *TN*, *FP* and *FN* shown in Table 2. *Precision* Eq. (14) is defined as the number correctly detected phishing websites (*TP*) divided by the number of all detected phishing websites (*TP*+*FP*).

$$Precision = TP/(TP + FP). \tag{14}$$

Recall Eq. (15) is defined as the number of correctly detected phishing websites (*TP*) divided by the number of all phishing websites (*TP*+*FN*).

$$Recall = TP/(TP + FN). \tag{15}$$

Table 2
The confusion matrix for evaluation the performance of the phishing detection.

Actual value	Predict value	
	Positive (Phishing websites)	Negative (Legal websites)
Positive (Phishing websites)	**TP**: The number of phishing websites that are correctly detected as phishing ones.	**FN**: The number of phishing websites that are wrongly detected as legal ones.
Negative (Legal websites)	**FP**: The number of legal websites that are wrongly detected as phishing ones.	**TN**: The number of legal websites that are correctly detected as legal ones.

Table 3
Performance of the improved K-medoids algorithm on Dataset 1.

Points	Precision	Recall	F1_score	Accuracy	Time cost (s)
500	0.8989	0.9057	0.8972	0.8900	0.0876
1000	0.9000	0.8911	0.8955	0.8950	0.2274
1500	0.9257	0.9200	0.9217	0.9333	0.2753
2000	0.9466	0.9406	0.9435	0.9427	0.3723
2500	0.9286	0.9286	0.9286	0.9320	0.5087
3000	0.9588	0.9309	0.9446	0.9500	0.4978
3500	0.9371	0.9391	0.9228	0.9300	0.6049
4000	0.9552	0.9537	0.9519	0.9523	1.0178
4500	0.9517	0.9367	0.9441	0.9456	0.8160
5000	0.9403	0.9561	0.9481	0.9500	1.4735
5500	0.9481	0.9431	0.9456	0.9445	0.9939
5780	0.9557	0.9548	0.9571	0.9558	1.4790
All data	0.9563	0.9563	0.9563	0.9561	4.7461

*F*1_*score* Eq. (16) is defined as the harmonic average of the *Precision* and the *Recall*, which is a comprehensive metric for evaluating the performance of a classifier. The value of this metric is in the interval [0, 1]. The larger value of this metric is, the better performance of the classifier will be.

$$F1\_score = 2\times(Precision\times Recall)/(Precision + Recall). \quad (16)$$

Accuracy Eq. (17) is defined as the number of all phishing websites (*TP*+*TN*) divided by the number of all websites (*TP*+*TN*+*FP*+*FN*).

$$Accuracy = (TP + TN)/(TP + TN + FP + FN). \quad (17)$$

4.3. Performance of the original phishing dataset refining

The improved K-medoids clustering algorithm can remove the duplicate data points from the original phishing dataset. It decreases the complexity of the training procedure of the neural network classifier, where a refined set of data points similar to those in phishing detection as the original dataset is acquired.

Table 3 lists the experimental results of the improved K-medoids algorithm with selected different numbers of points. The original dataset (Dataset 1) without optimal feature selection performs this experiment to verify the effectiveness of this algorithm. In this table, each time the subset of data points (without duplicate data points) is acquired, performance of DTOF-ANN from five different perspectives are tested.

The improved K-medoids algorithm processes the data points of the Dataset 1 one after another until all points are processed. It can be seen that the number of points of the biggest subset of the Dataset 1 is 5780. When the number of data points reaches 5780, the improved K-medoids algorithm terminates. This means there are nearly half of duplicate data points in the original dataset.

The last line of this table gives the performance of DTOF-ANN when all data points in the original dataset are tested. From this table, we can see that the refined dataset (with 5780 data points) has almost the same performance as the original dataset when metrics of "*Precision*" "*Recall*", "*F*1_*sore*" and "*Accuracy*" are tested. However, it takes much less time cost than the ones of the original dataset.

Table 4
Performance comparisons between the traditional and the improved K-medoids algorithm.

Medoids	Traditional K-medoids		Improved K-medoids	
	Accuracy	Times (s)	Accuracy	Times (s)
500	0.8801	0.0156	0.9311	0.0156
1000	0.8803	0.0312	0.9350	0.0468
1500	0.8671	0.1092	0.9167	0.1092
2000	0.9125	0.1560	0.9231	0.1404
2500	0.9140	0.1404	0.9442	0.2340
3000	0.9283	0.2028	0.9283	0.2184
3500	0.9112	0.3432	0.9414	0.2562
4000	0.9114	0.3432	0.9231	0.3432
4500	0.9267	0.3900	0.9278	0.4056
5000	0.9152	0.5772	0.9313	0.5616
5500	0.9236	0.6084	0.9373	0.6084
5780	0.9204	0.6864	0.9403	0.6240

Table 4 compares the performance (accuracy and time cost) between the traditional K-medoids algorithm and the improved K-medoids algorithm. For fairness and implementation simplicity, this experiment uses the KNN as the underlying classifier. Specifically, Dataset 1 (without optimal features selection) is firstly processed by the two algorithms with the same procedures as ones listed in Table 3. Then, two refined datasets by the traditional and improved K-medoids algorithms are fed to the KNN classifier respectively. We can see that the performance of the improved K-medoids algorithm is better than the performance of the traditional K-medoids algorithm.

4.4. Effectiveness of the optimal feature selection

This part evaluates the effectiveness of the optimal feature selection algorithm that combines CART and the local search methods, where useless and negative features can be deleted. In this experiment, Dataset 1 tests the proposed algorithm.

The values of the new defined index *f_Value* for the 30 features of the Dataset 1 are computed first (Fig. 7). Some features, such as f_8, f_{14} and f_{26}, have much higher *f_Value*s than the other features. However, there are also some features have much smaller *f_Value*s. Features with small *f_Value*s may have small or even negative impacts on the performance of the underlying classifier and, therefore, they are deleted them from the original features set. Apparently, if *f_Value* of a feature is 0, this feature is taken as a useless feature. ρin Eq. (8) is the index for finding the negative features having relatively small *f_Value*s. Consequently, negative feature can be deleted by combining *f_Value* and ρ.

Fig. 8 shows the accuracy of the classifier which uses the single feature to detect the phishing websites for the Dataset 1 (without refining data points). Figs. 7 and 8 show that features with *f_Value* bigger than 0.1 also have higher accuracy than the ones with *f_Value* smaller than 0.1. For example, the biggest *f_Value* of f_8 has the highest detection accuracy and thus f_8 has the positive impact on the phishing detection.

Figs. 7 and 8 also show that the detection accuracy of some features is the same when their *f_Value* are less than 0.1. That is, when *f_Value* is less than 0.1, the features with higher *f_Value*

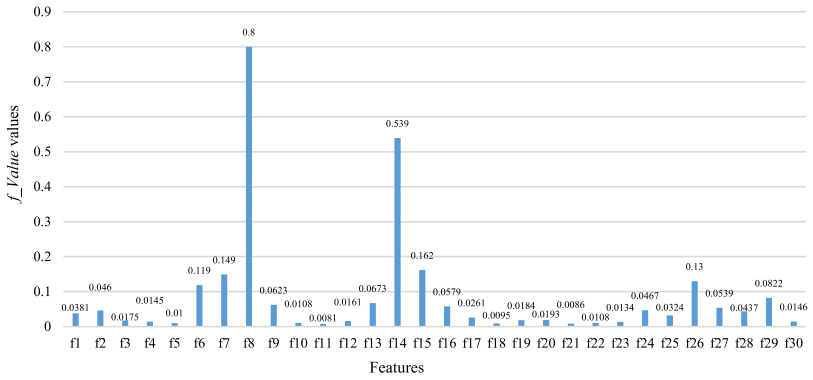

Fig. 7. *f_Value* values of different features of the Dataset 1.

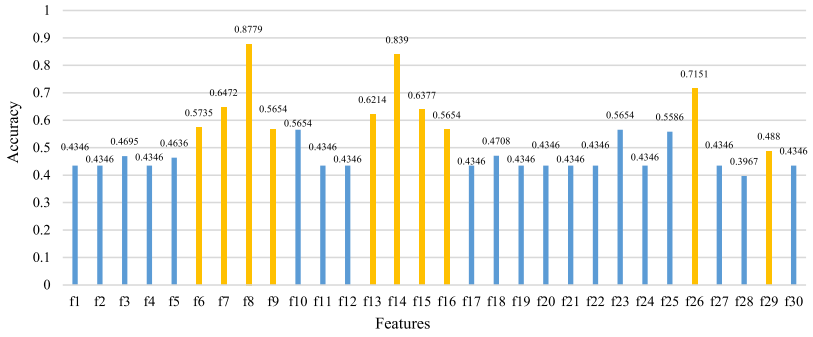

Fig. 8. The accuracy of the classifier based on each single feature.

exhibit smaller detection accuracy than the ones with smaller *f_Value*. Therefore, we can conclude that the features with high *f_Value*s have negative impacts on the phishing websites detection. For example, when the feature set {f_8, f_{14}, f_{15}, f_7, f_{26}, f_6, f_{29}, f_{13}, f_9, f_{16}} are used to by the classifier, the detection accuracy is 93.89%. However, when f_{27} ($f\_Value(f_{27}) = 0.0539$) is included in this set, the detection accuracy is decreased to 93.71%. Therefore, f_{27} has the negative impact on the detection accuracy. The negative features are the ones to be deleted from the original features set for the high detection accuracy.

For the algorithm of optimal feature selection, ten features with the highest *f_Value*s are selected as the initial feature set. As shown in Fig. 8, the feature set {$f_8, f_{14}, f_{15}, f_7, f_{26}, f_6, f_{29}, f_{13}, f_9$, f_{16}} corresponding to the yellow bars are selected as the initial features for the tested dataset. Then, the remainder features with positive impacts are sequentially added to this set based on their *f_Value*s. Specifically, if the detection accuracy is increased, the corresponding positive feature is put to the selected feature set. This process is continued until the detection accuracy barely increased or does not grow any more. This method can prune out the small impact and useless features from the original feature set.

Table 5 gives the detection accuracy (without refining data points) when different positive impact features are added to the initial feature set. As discussed above, since f_{27} (although it has the highest *f_Value* among the remaining features) has the negative impact on the detection accuracy, this feature is not included in the optimal feature set. However, f_{24} (it has the second highest *f_Value* among the remaining features) is put into the optimal feature set, as shown in the second line of Table 5, because it has positive impact on the detection accuracy.

Table 5 shows that the optimal feature set of the tested dataset is F_{14} which has less features (24) than the original ones (30). Meanwhile, it gets higher detection accuracy (95.76%) than the ones of the initial feature set (93.89%). The last line of Table 5 gives the detection accuracy with all features of the tested dataset. The detection accuracy (94.02%) of utilizing all features is smaller than that (95.76%) of utilizing the selected optimal feature set due to negative impact and useless features.

In Table 6, line 2 - line 4 give the experimental results with 4 different categories of features (as listed in Table 1). Here, the original dataset (Dataset 1) is refined by the improved K-medoids algorithm. It can be seen from this table that single category of features cannot obtain the required detection performance. Therefore, combining all the 4 categories of features is necessary. Line 5 and line 6 in Table 6 shows that the results with all of the 30 features and selected optimal feature set (F_{14}in Table 5) respectively. The accuracy of utilizing optimal feature set is better than those with all the features. These results further demonstrate the effectiveness of combining the methods of refining dataset and selecting optimal features.

4.5. Overall performance comparisons among different methods

The performance of DTOF-ANN is also compared with those of some classical classifiers and some existing methods.

Table 7 lists the phishing detection results of DTOF-ANN and some commonly used classifiers. These classifiers are NB (Naïve Bayes), LR (Logistic Regression), GBDT (Gradient Boosting + Decision Tree), SVM (Support Vector Machine), ANN (Artificial Neural Network), RF (Random Forest) and DT (Decision Tree). For fairness, all the classifiers are used to process the two datasets listed in Table 1, with the results being the averages of ten repetitive experiments on the two datasets. We can see that, due to strong learning ability and high classification accuracy, the performances of neural network classifiers (ANN and DTOF-ANN) are better than the other classifiers. In addition, over-fitting and high computational complexity of the DTOF-ANN are alleviated because the negative and useless features are pruned by the optimal

Table 5
Detection accuracy of different feature sets (without data refining).

Feature set names	Specific features	Accuracy
Initial set	$f_8, f_{14}, f_{15}, f_7, f_{26}, f_6, f_{29}, f_9, f_{16}, f_{13}$	0.9389
F_1	$f_8, f_{14}, f_{15}, f_7, f_{26}, f_6, f_{29}, f_9, f_{16}, f_{13}, f_{24}$	0.9421
F_2	$f_8, f_{14}, f_{15}, f_7, f_{26}, f_6, f_{29}, f_9, f_{16}, f_{13}, f_{24}, f_5$	0.9428
F_3	$f_8, f_{14}, f_{15}, f_7, f_{26}, f_6, f_{29}, f_9, f_{16}, f_{13}, f_{24}, f_5, f_{12}$	0.9448
F_4	$f_8, f_{14}, f_{15}, f_7, f_{26}, f_6, f_{29}, f_9, f_{16}, f_{13}, f_{24}, f_5, f_{12}, f_{18}$	0.9462
F_5	$f_8, f_{14}, f_{15}, f_7, f_{26}, f_6, f_{29}, f_9, f_{16}, f_{13}, f_{24}, f_5, f_{12}, f_{18}, f_3$	0.9471
F_6	$f_8, f_{14}, f_{15}, f_7, f_{26}, f_6, f_{29}, f_9, f_{16}, f_{13}, f_{24}, f_5, f_{12}, f_{18}, f_3, f_{21}$	0.9479
F_7	$f_8, f_{14}, f_{15}, f_7, f_{26}, f_6, f_{29}, f_9, f_{16}, f_{13}, f_{24}, f_5, f_{12}, f_{18}, f_3, f_{21}, f_1$	0.9493
F_8	$f_8, f_{14}, f_{15}, f_7, f_{26}, f_6, f_{29}, f_9, f_{16}, f_{13}, f_{24}, f_5, f_{12}, f_{18}, f_3, f_{21}, f_1, f_{23}$	0.9516
F_9	$f_8, f_{14}, f_{15}, f_7, f_{26}, f_6, f_{29}, f_9, f_{16}, f_{13}, f_{24}, f_5, f_{12}, f_{18}, f_3, f_{21}, f_1, f_{23}, f_{22}$	0.9534
F_{10}	$f_8, f_{14}, f_{15}, f_7, f_{26}, f_6, f_{29}, f_9, f_{16}, f_{13}, f_{24}, f_5, f_{12}, f_{18}, f_3, f_{21}, f_1, f_{23}, f_{22}, f_{17}$	0.9538
F_{11}	$f_8, f_{14}, f_{15}, f_7, f_{26}, f_6, f_{29}, f_9, f_{16}, f_{13}, f_{24}, f_5, f_{12}, f_{18}, f_3, f_{21}, f_1, f_{23}, f_{22}, f_{17}, f_{11}$	0.9557
F_{12}	$f_8, f_{14}, f_{15}, f_7, f_{26}, f_6, f_{29}, f_9, f_{16}, f_{13}, f_{24}, f_5, f_{12}, f_{18}, f_3, f_{21}, f_1, f_{23}, f_{22}, f_{17}, f_{11}, f_2$	0.9566
F_{13}	$f_8, f_{14}, f_{15}, f_7, f_{26}, f_6, f_{29}, f_9, f_{16}, f_{13}, f_{24}, f_5, f_{12}, f_{18}, f_3, f_{21}, f_1, f_{23}, f_{22}, f_{17}, f_{11}, f_2, f_{20}$	0.9570
F_{14}	$f_8, f_{14}, f_{15}, f_7, f_{26}, f_6, f_{29}, f_9, f_{16}, f_{13}, f_{24}, f_5, f_{12}, f_{18}, f_3, f_{21}, f_1, f_{23}, f_{22}, f_{17}, f_{11}, f_2, f_{20}, f_{10}$	0.9576
All features	f_1……f_{30}	0.9402

Table 6
Performance of different feature sets (after data refining).

Feature sets	Precision	Recall	F1_score	Accuracy
Address Features (f_1–f_{12})	0.8918	0.8751	0.8834	0.9037
Abnormal Features (f_{13}–f_{18})	0.8874	0.8117	0.8479	0.8734
HTML and JavaScript Features (f_{19}– f_{23})	0.7742	0.0499	0.0938	0.5087
Domain Features (f_{24}–f_{30})	0.7099	0.7180	0.7139	0.7499
All features (f_1–f_{30})	0.9563	0.9563	0.9563	0.9561
Optimal Feature Set (F_{14} in Table 5)	0.9776	0.9736	0.9761	0.9780

Table 7
Performance comparisons between DTOF-ANN and some classical classifiers.

Classifiers	Precision	Recall	F1_score	Accuracy
NB	0.5189	0.9979	0.6828	0.5970
LR	0.9313	0.9022	0.9165	0.9285
GBDT	0.9573	0.9324	0.9447	0.9525
SVM	0.9601	0.9261	0.9420	0.9512
ANN	0.9653	0.9563	0.9563	0.9661
RF	0.9688	0.9620	0.9654	0.9688
DT	0.9460	0.9640	0.9549	0.9588
DTOF-ANN	0.9776	0.9736	0.9761	0.9780

Table 8
Performance comparisons between DTOF-ANN and some existing methods.

Models	Datasets		Precision	Recall	F1_score	Accuracy
	Legal	Phishing				
Ma	15000	20500	0.9980	0.9240	0.9510	0.9550
Cantina	2100	19	0.2120	0.8920	0.3420	0.9690
Zhang	100	100	0.9100	0.9792	0.9150	0.9150
Off-the-Hook	20000	2000	0.9750	0.9510	0.9630	0.9990
Cantina+	1868	940	0.9640	0.9640	0.9640	0.9550
DTOF-ANN	18004	7633	0.9776	0.9736	0.9761	0.9780

feature selection. Consequently, *Accuracy* of DTOF-ANN is better than the traditional ANN classifiers.

Table 8 lists the experimental results of DTOF-ANN and some existing models on detecting phishing attacks. For the generality of comparisons, four kinds of phishing detection methods are selected in this experiment, including the phishing detection system proposed by Ma [7] (abbreviated as "*Ma*" in this table) based on the blacklist; the Cantina [12] based on the heuristic method; the tool proposed by Zhang [22] (abbreviated as "*Zhang*" in this table) based on the visual similarity method; the Off-the-Hook [19] and Cantina+ [37] based on the machine learning method.

The tested dataset used by "*Ma*" is collected from the "DMOI Open Directory Project", the "Yahoo directory" the "PhishTank" and the "Spamscatter". The source dataset of the Cantina is constructed based on the "Alexa" and the "PhishTank". The tested dataset used by "*Zhang*" is selected from the "PhishTank". The tested dataset of the Off-the-Hook tool is selected from the "PhishTank" and the "Intel Security". The test dataset of the Cantina+ is built on the "Alexa" and the "3Sharp".

The tested datasets of DTOF-ANN are composed of Dataset 1 and Dataset 2 (Table 1). As can be seen, the performance of the DTOF-ANN is superior to the others except for *Accuracy* of the off-the-Hook which collects more than 200 phishing features. More features contribute to high accuracy of this tool. However, it will also bring extra complex computation.

Table 8 gives the performances of these tools are evaluated by different datasets. For fairness, the performance of DTOF-ANN is also compared with five other machine learning phishing detection models. Table 9 lists the experimental results, where two datasets listed in Table 1 are used to test these models. The performances of different models are averages of ten repetitive experiments.

In these models, the BN&SVM [11] is based on both the Naive Bayesian and the Support Vector Machine classifiers; the K-medoids&PNN [16] is based on both the K-medoids clustering and probabilistic neural networks; the Hybrid Model [38] is built based on several machine learning classifiers such as the Bayes Net, Naïve Bayes, Random Forest, SVM and so on; FACA [13] is built on the associative classification (AC) which is an effective supervised learning approach targeting to predict unseen instances; and the K-means&SVM [39] is built on both the K-means clustering and SVM.

Table 9
Performance comparisons among different machine learning phishing detection models.

Models	Precision	Recall	F1_score	Accuracy
BN&SVM	0.8817	0.9265	0.90035	0.9178
K-medoids&PNN	0.9666	0.9626	0.9203	0.9607
Hybrid Model	0.9547	0.9617	0.9585	0.9623
FACA	1.0000	0.7022	0.8251	0.8694
K-means&SVM	0.9441	0.5693	0.7102	0.9432
DTOF-ANN	0.9776	0.9736	0.9761	0.9780

We can see from Table 9 that *Precision* of the FACA is best among all the tested models. However, *Recall*, the *F1_score* and *Accuracy* of this model is worse than most of the tested models. So, this model is unstable in detecting phishing attacks. As the bad *Recall* and *F1_score*, the K-means&SVM model is also unstable. In general, the performance of DTOF-ANN is better than other models.

5. Conclusion and future work

This paper proposes DTOF-ANN, a neural network phishing detection model based on decision tree and optimal feature selection. In this model, the traditional K-medoids clustering algorithm is firstly improved to refine the public phishing datasets. Then, a new index, *f_Value*, based on the Gini coefficient, is defined to evaluate the impact of different features on the performance of phishing detection. The optimal feature selection algorithm based on *f_Value*, which integrates the decision tree and local search method, is designed to construct the optimal feature vector for the underlying neural network classifier. Finally, the optimal structure of the neural network that suits to process the problem of phishing detection is constructed through the iterative parameter adjustment. As the continuously change of the phishing attacks, effective methods of optimal feature selection are urgently needed, which will be our future work.

CRediT authorship contribution statement

Erzhou Zhu: Conceptualization, Methodology, Project administration, Funding acquisition, Writing - review & editing. **Yinyin Ju:** Software, Writing - original draft, Investigation. **Zhile Chen:** Formal analysis, Validation. **Feng Liu:** Supervision, Resources, Data curation. **Xianyong Fang:** Conceptualization, Formal analysis.

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.

Acknowledgments

We are grateful for Professor Yun Yang from Swinburne University of Technology in Australia for English proofreading. This work was supported by the University Natural Science Research Project of Anhui Province (P.R. China) (Grant No. KJ2018A0022) and the Natural Science Foundation of Anhui Province (P.R. China) (Grant No. 2008085MF188).

References

[1] Kang Leng Chiew, Kelvin Sheng Chek Yong, Choon Lin Tan, A survey of phishing attacks: their types, vectors and technical approaches, Expert Syst. Appl. 106 (2018) 1–20.
[2] APWG, Phishing activity trends Report: 1st Quarter 2019. Accessed: 2019. [Online]: https://docs.apwg.org/reports/apwg_trends_report_q1_2019.pdf.
[3] Neda Abdelhamid, Aladdin Ayesh, Fadi Thabta, Phishing detection based associative classification data mining, Expert Syst. Appl. 41 (13) (2014) 5948–5959.
[4] Mohamed Alsharnouby, Furkan Alaca, Sonia Chiasson, Why phishing still works: user strategies for combating phishing attacks, Int. J. Hum.-Comput. Stud. 82 (2015) 69–82.
[5] Mahmoud. Khonji, Youssef. Iraqi, Andrew Jones, Phishing detection: A literature survey, IEEE Commun. Surveys Tutor. 15 (4) (2013) 2091–2121.
[6] Erzhou Zhu, Yuyang Chen, Chengcheng Ye, Xuejun Li, Feng Liu, OFS-NN: An effective phishing websites detection model based on optimal feature selection and neural network, IEEE ACCESS 7 (2019) 73271–73284.
[7] Justin Ma, Lawrence K. Saul, Stefan Savage, Geoffrey M. Voelker, Beyond blacklists: Learning to detect malicious web sites from suspicious URLs, in: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD 2009, Paris, France, 2009, pp. 1245-1254.
[8] Frank Vanhoenshoven, Gonzalo Nápoles, Rafael Falcon, Koen Vanhoof, Mario Köppen, Detecting malicious URLs using machine learning techniques, in: Proceedings of the 2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016, Athens, Greece, 2016, pp. 1-8.
[9] Yogesh Joshi, Samir Saklikar, Debabrata Das, Subir Saha, PhishGuard: A browser plug-in for protection from phishing, in: Proceedings of the 2nd International Conference on Internet Multimedia Services Architecture and Applications, IMSAA 2008, Bangalore, India, December 10-12, 2008, pp. 1-6.
[10] Anthony Y. Fu, Liu Wenyin, Xiaotie Deng, Detecting phishing web pages with visual similarity assessment based on earth mover's distance (EMD), IEEE Trans. Dependable Secure Comput. 3 (4) (2006) 301–311.
[11] Xiaoqing Gu, Hongyuan Wang, Tongguang Ni, An efficient approach to detecting phishing web, J. Comput. Inform. Syst. 9 (14) (2013) 5553–5560.
[12] Yue Zhang, Jason I. Hong, Lorrie F. Cranor, Cantina: A content-based approach to detecting phishing web sites, in: Proceedings of the 16th international conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, 2007, pp. 639-648.
[13] Wa'el Hadi, Faisal Aburub, Samer Alhawari, A new fast associative classification algorithm for detecting phishing websites, Appl. Soft Comput. 48 (2016) 729–734.
[14] Mohammed Nazim Feroz, Susan Mengel, Examination of data, rule generation and detection of phishing URLs using online logistic regression, in: Proceedings of the 2014 IEEE International Conference on Big Data, Big Data 2014, Washington, DC, USA, 2014, pp. 241-250.
[15] Kanchan Hans, Laxmi Ahuja, S.K. Muttoo, Detecting redirection spam using multilayer perceptron neural network, Soft Comput. 21 (13) (2017) 3803–3814.
[16] M. El-Sayed, El-Alfy, Detection of phishing websites based on probabilistic neural networks and K-medoids clustering, Comput. J. 60 (12) (2017) 1745–1759.
[17] S. Sivagama sundhari, A knowledge discovery using decision tree by Gini coefficient, in: Proceedings of the 2011 International Conference on Business, Engineering and Industrial Applications, ICBEIA 2011, Kuala Lumpur, Malaysia, 2011, pp. 232-235.
[18] Hiba Zuhair Zeydan, Ali Selamat, Mazleena Salleh, Feature selection for phishing detection: A review of research, Int. J. Intell. Syst. Technol. Appl. 15 (2) (2016) 147–162.
[19] Samuel Marchal, Giovanni Armano, Tommi Grondahl, Kalle Saari, Nidhi Singh, N. Asokan, Off-the-hook: An efficient and usable client-side phishing prevention application, IEEE Trans. Comput. 66 (10) (2017) 1717–1733.
[20] Google Developer, Google Developer. Safe Browsing API. Accessed: 2019. [Online]: https://developers.google.com/safe-browsing/.
[21] Pawan Prakash, Manish Kumar, Ramana Rao Kompella, Minaxi Gupta, PhishNet: Predictive blacklisting to detect phishing attacks, in: Proceedings of the 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2010, San Diego, CA, USA, March 2010, pp. 346-350.
[22] Weifeng Zhang, Hua Lu, Baowen Xu, Hongji Yang, Web phishing detection based on page spatial layout similarity, Informatica (Slovenia) 37 (3) (2013) 231–244.
[23] Routhu Srinivasa. Rao, Syed Taqi Ali, A computer vision technique to detect phishing attacks, in: Proceedings of the 5th International Conference on Communication Systems and Network Technologies, CSNT 2015, Gwalior, India, April 2015, pp. 596-601.
[24] Ying Pan, Xuhua Ding, Anomaly based web phishing page detection, in: Proceedings of the 22nd Annual Computer Security Applications Conference, ACSAC 2006, Miami Beach, FL, USA, December 2006, pp. 381-390.
[25] Mingxing He, Shi-Jinn Horng, Pingzhi Fan, Muhammad Khurram Khan, Ray-Shine Run, Jui-Lin Lai, Rong-Jian Chen, Adi Sutanto, An efficient phishing webpage detector, Expert Syst. Appl. 38 (10) (2011) 12018–12027.
[26] Mouad Zouina, Benaceur Outtaj, A novel lightweight URL phishing detection system using SVM and similarity index, Human-Centric Comput. Inf. Sci. 7 (1) (2017) 98.
[27] A. Martin, Na.Ba. Anutthamaa, M. Sathyavathy, Marie Manjari Saint Francois, Prasanna Venkatesan, A framework for predicting phishing websites using neural networks, Int. J. Comput. Sci. Issues 8 (2) (2011) 330–336.
[28] Rami M. Mohammad, Fadi Thabtah, Lee McCluskey, Predicting phishing websites based on self-structuring neural network, Neural Comput. Appl. 25 (2) (2014) 443–458.
[29] Luong Anh Tuan Nguyen, Ba Lam To, Huu Khuong Nguyen, Minh Hoang Nguyen, An efficient approach for phishing detection using single-layer neural network, in: Proceedings of the International Conference on Advanced Technologies for Communications, ATC 2014, 2014, Hanoi, Vietnam October 2014, pp. 435-440.

[30] Kang Leng Chiew, Choon Lin Tan, KokSheik Wong, Kelvin S.C. Yong, Wei King Tiong, A new hybrid ensemble feature selection framework for machine learning-based phishing detection system, Inform. Sci. 484 (2019) 153–166.
[31] Fergus Toolan, Joe Carthy, Feature selection for spam and phishing detection, in: Proceedings of the 2010 eCrime Researchers Summit, eCrime 2010, Dallas, TX, USA, 2010. pp. 1-12.
[32] Mahdieh Zabihimayvan, Derek Doran, Fuzzy rough set feature selection to enhance phishing attack detection, in: The 2019 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2019, New Orleans, LA, USA, 2019. (Preprint, arXiv:1903.05675).
[33] Y. Kilani, A. Alsarhan, M. Bsoul, A.F. Otoom, Local search algorithms for solving the combinatorial optimization and constraint satisfaction problems, in: Proceedings of the 6th International Workshop Soft Computing Applications, SOFA 2014, Timisoara, Romania July 2014, pp. 199-211.
[34] Rami M. Mohammad, Fadi Thabtah, Lee McCluskey, Intelligent rule-based phishing websites classification, IET Inf. Secur. 8 (3) (2014) 153–160.
[35] Phishtank, Out of the net into the tank, Accessed: 2018. [Online]: https://www.phishtank.com/.
[36] Alexa, Find, Reach, and Convert Your Audience with Marketing That Works. Accessed: 2018, [Online]: https://www.alexa.com/.
[37] Guang Xiang, Jason I. Hong, Carolyn Penstein Rosé, Lorrie Cranor, Cantina+: A feature-rich machine learning framework for detecting phishing web sites, ACM Trans. Inf. Syst Secur. 14 (2) (2011) Article No. 21.
[38] M. Amaad Ul Haq Tahir, Sohail Asghar, Ayesha Zafar, Saira Gillani, A hybrid model to detect phishing-sites using supervised learning algorithms, in: Proceedings of the International Conference on Computational Science and Computational Intelligence, CSCI 2016, Las Vegas, NV, USA, December 2016, pp. 1126-1133.
[39] Nadir Omer Fadl Elssied, Othman Ibrahim, Ahmed Hamza Osman, Enhancement of spam detection mechanism based on hybrid k-mean clustering and support vector machine, Soft Comput. 19 (11) (2015) 3237–3248.