Computer Systems Science & Engineering
DOI:10.32604/csse.2022.019300
Article

Tech Science Press

Semantic Based Greedy Levy Gradient Boosting Algorithm for Phishing Detection

R. Sakunthala Jenni* **and S. Shankar**

Department of Computer Science and Engineering, Hindusthan College of Engineering and Technology, Coimbatore, 641032, India
*Corresponding Author: R. Sakunthala Jenni. Email: sakunthalajenni2021@gmail.com
Received: 09 April 2021; Accepted: 14 June 2021

Abstract: The detection of phishing and legitimate websites is considered a great challenge for web service providers because the users of such websites are indistinguishable. Phishing websites also create traffic in the entire network. Another phishing issue is the broadening malware of the entire network, thus highlighting the demand for their detection while massive datasets (i.e., big data) are processed. Despite the application of boosting mechanisms in phishing detection, these methods are prone to significant errors in their output, specifically due to the combination of all website features in the training state. The upcoming big data system requires MapReduce, a popular parallel programming, to process massive datasets. To address these issues, a probabilistic latent semantic and greedy levy gradient boosting (PLS-GLGB) algorithm for website phishing detection using MapReduce is proposed. A feature selection-based model is provided using a probabilistic intersective latent semantic preprocessing model to minimize errors in website phishing detection. Here, the missing data in each URL are identified and discarded for further processing to ensure data quality. Subsequently, with the preprocessed features (URLs), feature vectors are updated by the greedy levy divergence gradient (model) that selects the optimal features in the URL and accurately detects the websites. Thus, greedy levy efficiently differentiates between phishing websites and legitimate websites. Experiments are conducted using one of the largest public corpora of a website phish tank dataset. Results show that the PLS-GLGB algorithm for website phishing detection outperforms state-of-the-art phishing detection methods. Significant amounts of phishing detection time and errors are also saved during the detection of website phishing.

Keywords: Web service providers; probabilistic intersective; latent semantic; greedy levy; divergence; gradient; phishing detection; big data

1 Introduction

Web security is a materializing inclination in novel big data settings. Conventionally, web security is directed by utilizing different methods, such as privacy preservation techniques, hidden Markov models, and reasoning-based strategies. Amid various issues, web phishing is the current pertinent interest. Phishing refers to the process of mimicking an official website of banks and social networking sites.

Phishing detection refers to the process of detecting a phishing activity. Several algorithms have been designed by many researchers.

An optimal feature selection and neural network (OFS-NN) was proposed in [1] to detect phishing websites. A feature validity value (FVV) was initially introduced to measure the significance of sensitive features on detecting phishing websites. On the basis of the FVV value, an algorithm was developed to select optimal features. Thus, the issue related to overfitting in the neural network was solved.

Finally, the features selected were utilized to train the neural network with which phishing websites were detected by means of an optimal classifier, which resulted in the accurate detection of phishing websites. With continuous changes in the sensitive features involved in phishing attacks, optimal feature selection remained an important issue. Optimal feature selection was performed by observing the URL to identify the missing data and, accordingly, eliminate the URL. In this manner, optimal feature selection was ensured even in the presence of continuous changes in the sensitive features.

A lightweight application called CatchPhish was proposed in [2] to predict the authority of the URL without searching the website. The proposed method used the host name of the user, complete URL, and inverse document frequency of the corresponding term frequency. Finally, phish-hinted words were utilized from the suspicious URL for classification via random forest classifier, thereby contributing to the accuracy. However, the significance of the features was not concentrated. To address this issue, only significant features were obtained through preprocessing by applying the probabilistic intersective latent semantic preprocessing (PILSP) model.

A machine learning framework to assist in the comprehensive analysis and detection of web phishing was proposed in [3]. Decision tree algorithms were utilized. Therefore, significant classification, whether the website was phishing or normal, was provided in an exhaustive manner based on certain features, improving the precision and recall factor. Despite improvements observed in terms of precision and recall, the phishing attack detection was less focused. The model of greedy levy divergence gradient (GLDG) website phishing detection was developed to attain optimal detection and improve accuracy rate.

The contributions of the proposed method are as follows:

1. We proposed a latent semantic preprocessing model (i.e., PILSP) to identify missing data in the URL and discard the URL with certain missing data to detect phishing sites.
2. We designed a GLDG model by combining gradient boosting and contrast divergence to increase the detection accuracy of phishing sites.
3. We deployed our method, namely, probabilistic latent semantic and greedy levy gradient boosting (PLS-GLGB), to provide protection and safeguard users from accessing phishing sites even for big data using MapReduce.
4. Experimental evaluations on a complex phish tank data set demonstrate the efficiency of the proposed method in terms of phishing detection time, phishing detection overhead, and efficiency of the detection system using a confusion matrix.

The remainder of the paper is organized as follows. A detailed literature of the prevailing antiphishing methods is presented in Section 2. The proposed method is discussed in Section 3. The experimentation details and the results are presented in Section 4. The deployment and effectiveness of the proposed method are discussed in Section 5. Finally, the conclusion is provided in Section 6.

2 Structure

One of the materializing trends in the big data environment is web security. Among several problems in this environment, web phishing has started receiving attention in recent years. In several existing works, web

security has been addressed by different methods, such as privacy preservation techniques and Markov models.

Decision tree algorithms were applied to detect between phishing and nonphishing attacks. Despite several antiphishing mechanisms launched by software companies, such as blacklists, heuristic mechanisms, and machine learning-based approaches, not all phishing attacks can be detected at an early stage. In [4], a real-time antiphishing system utilizing seven classification models was presented to improve accuracy detection.

Compared with conventional visual similarity-based techniques using whitelists, in [5], a lightweight approach using visual similarity at a first-level filter was utilized to detect phishing sites. A survey of different web phishing detection schemes was provided in [6]. Automated page layout-based phishing detection methods were discussed in [7] in addition to learning-based aggregation model, thereby contributing to accuracy. Another model with support vector and naïve Bayes was also designed in [8] to efficiently differentiate between phishing and benign instances.

However, all these aforementioned methods are time consuming and utilize static detection rules. In [9], a PhishLimiter was designed for deep packet inspection and then integrated with software-defined networking to detect activities involving phishing via e-mail and web-based communication in a timely manner.

An FVV [10] was initially identified to obtain significant features in the preliminary stage and then detection mechanism was performed to improve the accuracy rate. In [11], information-based brand authorization techniques were utilized to handle statistical antiphishing. Deep learning techniques were applied in [12] to handle big data.

However, existing antiphishing techniques were designed on the basis of page-related features. For instance, a method involving a fast phishing detection model was designed in [13] to improve the speed of the method and reduce detection time. Another method was proposed in [14] to improve the accuracy rate using principal component analysis and random forest and successfully classify suspicious websites.

A review of machine learning methods for spam detection along with phishing was discussed in [15]. Sine cosine algorithm was utilized in [16] to detect the presence of spam by using artificial neural network and multilayer network perceptron, thereby resulting in minimum error. Deep belief network was applied in [17], which first identified original features and interaction features and efficiently classified between true positive and false positive.

A systematic measure of the infiltration process involved in phishing detection along with the application for chrome extension, namely, Sniff-Phish, was developed in [18], where the computational time involving the resource-intensive model in cloud was designed to detect several types of phishing attacks, thus minimizing the false positive significantly. Text and image watermarking tools [19] were used as means for detecting between phishing attack and normal traffic. Here, watermark was used in the client side and was found to be fool proof.

A companion scheme was introduced in [20] to identify the brands of zero-hour phishing web pages through categorizing the target brand logos in page screenshots. The features of histogram of oriented gradients were used to attain visual representation of target brand logos in a scale-invariant approach.

In [21], systematic benchmarking study and evaluation were carried out with phishing features on diverse and extensive datasets. The imbalanced nature of phishing attacks affected the researchers and the performance of the detection system. New features were required to stop the attackers from fooling detection systems.

Motivated by these works for website phishing detection, we propose a PLS-GLGB for website phishing detection with the objective of reducing phishing detection time, overhead, and misclassification error via a confusion matrix. The description of the proposed method is elaborated in the following sections.

3 PLS-GLGB

The foremost objective of malicious users utilizing website phishing is to break recognition from users. Different materials and methods have been designed to safeguard users from website phishing attacks. PLS-GLGB for website phishing detection is presented in this paper. Latent semantic analysis is the method used for natural language processing. It helps analyze the relationship between a set of features and the objective (i.e., phishing attack detection). Greedy levy gradient boosting is an ensemble learning method used to resolve the exploration/exploitation dilemma. Gradient boosting generates and combines weak prediction models to obtain strong classification output results. The three main processes used for phishing detection technique are preprocessing, feature selection, and classification. Fig. 1 shows the block diagram of the PLS-GLGB method.

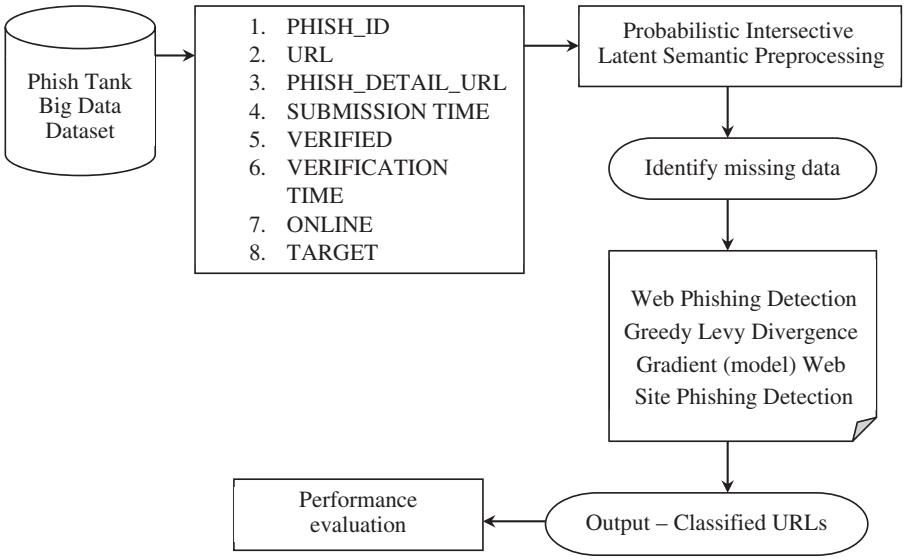

Figure 1: Block diagram of the PLS-GLGB method

As illustrated in Fig. 1, two steps are involved in the processing of the PLS-GLGB method for timely and accurate web phishing detection, namely, preprocessing and phishing detection. Preprocessing is performed to identify missing data in the URL provided as input from the Phish tank dataset. It is performed in our work by applying the PILSP model.

With this preprocessing model, missing data in the URL are identified and eliminated by means of data maximum likelihood, which ensures data quality during phishing detection. Subsequently, with the preprocessed features, the GLDG model is applied for accurate website phishing detection in an optimal manner. The elaborate description of the proposed method is presented in the following section.

3.1 PILSP Model Text Layout

The first step involved in the design of web phishing detection is to preprocess the given input dataset (i.e., missing data) with the objective of ensuring data quality. A PILSP model is used to obtain computationally efficient preprocessed features that analyze missing data in the URL path. Thus, only

data pertaining to the entire information are used for web phishing detection while missing data in a certain path are eliminated for further processing. This approach is performed by mapping high-dimensional vector parts of a URL to a lower-dimensional vector of phish identifier. The PILSP model is shown in Fig. 2.

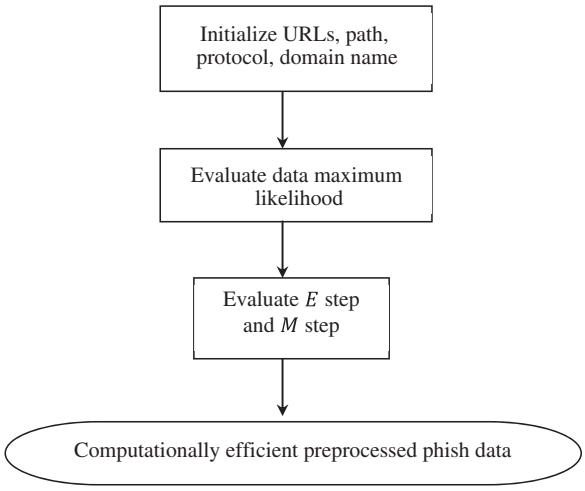

Figure 2: Flow diagram of the PILSP model

A collection of URLs, $URL = U_1, U_2, \ldots, U_n$ and a set of three parts $\{P, DN, Path\}$ representing protocol P, domain name DN, and path $Path$ which occur in those URLs, $Path = P_1, P_2, \ldots, P_n$ (with only the path used for analyzing the missing data to ensure data quality), are considered. The model then links a latent phish identifier variable $p = p_1, p_2, \ldots, p_n$ with the contingency of each path $Path = P_1, P_2, \ldots, P_n$ in a particular URL. Thus, the PILSP for the path–URL contingency is indicated by means of the probability intersection function (PIF) as shown in the following section.

$$Prob\ (U_i,\ P_j) = Prob\ (U_i) \sum_{i,j,k=1}^{n} Prob\ (P_j|p_k) Prob\ (p_k|U_i) \tag{1}$$

In Eq. (1), $Prob\ (U_i)$ is the probability that a path is observed in a given URL U_i, $Prob\ (P_j|p_k)$ is the probability of a specific URL governed on latent phish identifier variable p_k, $Prob\ (p_k|U_i)$ is the probability diffusion of a particular URL over the latent phish identifier variable space, and k refers to the number of phish IDs.

The probability $Prob\ (P_j|p_k)$ corresponds to URLs that make up a given phish ID, and the probability $Prob\ (p_k|U_i)$ corresponds to phish IDs that a given URL belongs to. With the aid of PIF, the parameters $Prob(P_j|p_k)$ and $Prob\ (p_k|\ U_i, P_j)$ are evaluated by means of data maximum likelihood ML and formulated as follows:

$$ML = \sum_{U_i \in URL} \sum_{P_j \in Path} n(U_i,\ P_j) \log Prob(U_i,\ P_j) \tag{2}$$

By applying the epistemological rule, the E step involved in the data maximum likelihood ML is mathematically expressed as follows:

$$Prob\ (p_k|\ U_i,\ P_j) = \frac{Prob(p_k) Prob\ (P_j|p_k) Prob(U_i|p_k)}{\sum_{ML\ \in K} Prob(p_{ML}) Prob\ (P_j|p_{ML}) Prob(U_i|p_{ML})} \tag{3}$$

The M step acquired by maximizing the expected data maximum likelihood is given by the following dual expression:

$$Prob(P_j|p_k) = \frac{\sum_{U_i \in URL} n(U_i,\ P_j) Prob(p_k|U_i,\ P_j)}{\sum_{U_i \in URL} \sum_{P_M \in Path} (U_i,\ P_M) Prob(p_k|U_i,\ P_M)} \tag{4}$$

$$Prob(p_k|U_i) = \frac{\sum_{P_j \in Path} n(U_i,\ P_j) Prob(p_k|U_i,\ P_j)}{n(U_i)} \tag{5}$$

With the aid of these functions, the pseudo code representation of probabilistic latent preprocessing is as follows:

Algorithm 1: Probabilistic Latent Preprocessing

Input: URLs, $URL = U_1, U_2, \ldots., U_n$, Path $Path = P_1, P_2, \ldots, P_n$, latent phish identifier variable $p = p_1, p_2, \ldots, p_n$

Output: Computationally efficient preprocessed phish data $PURL = PU_1, PU_2, \ldots, PU_n$

1: **Begin**

2: **For** each URLs, the *URL* with Path *Path* and latent phish identifier variable p

3: Obtain the PIF using Eq. (1)

4: Evaluate the data ML function using Eq. (2)

5: Evaluate the expected data maximum likelihood using Eq. (3)

6: Maximize the expected data maximum likelihood using Eqs. (4) and (5)

7: **Return** (*PURL*)

8: **End for**

9: **End**

In this probabilistic latent preprocessing algorithm, for each URL with its corresponding path and phish identifier variable as input, the objective remains to identify the path with a certain amount of missing data because phishing detection becomes simpler by identifying the missing data. This method is performed by means of PIF and data maximum likelihood. The parameters are evaluated in an iterative manner by varying the E step and M step when all the missing data in the phishing IDs are identified. Thus, the web phishing detection process is performed in a significant manner to improve data quality.

3.2 GLDG (Model) Website Phishing Detection

In preprocessed phish data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, x_i belongs to feature space X, and y_i belongs to label set $Y = \{-1, +1\}$. Here, the feature space corresponds to the preprocessed URLS $(x_1 \rightarrow PU_1, x_2 \rightarrow PU_2, \ldots, x_n \rightarrow PU_n)$ with a weak hypothesis, $wh_t{:}X \rightarrow \{-1, +1\}$. With the given assumptions, a GLDG model for website phishing detection is used to detect web phish in an accurate pattern. Gradient boosting generates an ensemble of weak prediction models.

The prediction process initiates with a weak model. Contrast divergence attempts to master the data space and is enhanced in an iterative manner by the succeeding model that minimizes the error of the

preceding model. The objective of gradient boosting remains to integrate weak learning models into a single strong model as follows:

$$F(PURL) = \sum_{j=1}^{m} \alpha_j CD_j(PURL) \tag{6}$$

Generally, CD_j is the contrast divergence of a specified depth that is ceaselessly enhanced over j assessments, and m refers to the regression parameter for that specific iteration.

$$CD_j(\theta,\ v^0) = -\sum_{PURL} Prob(PURL|v^0)\frac{\partial E(v^0,\ PURL)}{\partial \theta} + \sum_{PURL} Prob(PURL|v^j)\frac{\partial E(v^j,\ PURL)}{\partial \theta} \tag{7}$$

At each iteration, the model is updated as follows:

$$F_{j+1}(PURL) = F_j(PURL) + \alpha_{j+1} CD_{j+1}(PURL) \tag{8}$$

CD_{j+1} is selected to reduce the loss function L involved in the current model fitting of a preprocessed URL $PURL$ in an optimal manner using the sine–cosine function, which is mathematically expressed as follows:

$$MSE = \sum_{i=1}^{n} \frac{(x_i - x_i')^2}{n} \tag{9}$$

where mean square errors (9) x_i and x_i' refer to the actual class of preprocessed URL sample and projected class of preprocessed URL samples, respectively. A preprocessed URL sample vector for reducing the objective function is examined as an optimal preprocessed URL sample vector; it is utilized to detect web phish. When a preprocessed URL sample vector is defined as PU^i, certain URL vectors are generated in a random manner to structure the primary population as follows:

$$PU^i = PU^1,\ PU^2,\ \ldots,\ PU^n \tag{10}$$

Each preprocessed URL sample vector is rationalized via sine and cosine association. Although rationalization is achieved by means of sine and cosine association, the domain of the local optimum is obtained with the lack of self-learning potentiality. To address this issue, in addition to the association factor, a technique based on greedy levy is proposed for optimum website phish detection with respect to individual URLs. In this manner, the primary individual population is jumped out of optimality via the greedy levy operation. The preprocessed URL with association and greedy levy operation is mathematically formulated as follows:

$$PU_i^{t+1} = \begin{cases} PU_i^t + r_1 SINE(r_1) + \theta(j) * levy \\ PU_i^t + r_1 COSINE(r_1) + \theta(j) * levy \end{cases} \tag{11}$$

From the above Eq. (11), r_1 refers to the ratio of current iterations to the maximum number of iterations, and $\theta(j)$*$levy$ corresponds to the coefficient of self-learning factor of the subsequent greedy level for the preprocessed URL. Finally, the evaluated classifier is as follows:

$$H(PU_i^{t+1}) = SIGN\left(\sum_{t\in n} MSE * wh_t(PU_i^{t+1})\right) \tag{12}$$

The pseudo code representation of GLDG website phishing detection is as follows:

Algorithm 2: GLDG Website Phishing Detection

Input: URLs, $URL = U_1, U_2, \ldots, U_n$, Path $Path = P_1, P_2, \ldots, P_n$, latent phish identifier variable $p = p_1, p_2, \ldots, p_n$, preprocessed phish data $PURL = PU_1, PU_2, \ldots, PU_n$.

Output: Accurate website phish detection

1: **Initialize** $\theta = U, P, p$

2: **Begin**

3: Generate gradient boosting formulation using Eq. (6)

4: **For** specific iteration, measure contrast divergence using Eq. (7)

5: Perform updating using Eq. (8)

6: Measure loss function by means of MSE using Eq. (9)

7: Structure primary population using Eq. (10)

8: **For** i in PU, do

9: Update preprocessed URL sample vector via sine cosine function using Eq. (11)

10: Evaluate classifier using Eq. (12)

11: **End for**

12: **Let** $Y \rightarrow Prediction\ (PU)$

13: **If** $Y < 0.5$, return -1 //Phishing

14: **Else** return $+1$//Legitimate

15: **End if**

16: **End for**

17: **End**

With the preprocessed URLs (i.e., removing missing data) as the input, the objective here remains to accurately detect website phishing in an optimal manner. Two factors are considered for these objectives, namely, sine–cosine and greedy levy. Rationalization is initially achieved, followed by individual URL optimality.

3.3 Map Reduce Phase

For each mapper, the subsequent mapper ID (phish_ID) corresponds to the input *key*, and the input *value* refers to a list of values. Two elements constitute the values; the first element identifies the value type, and the second element refers to the data itself. During each epoch, the value *value* corresponds to the output of the reducer in the previous epoch; it includes the updated U, P, p, and their accumulated approximate gradients.

The input dataset obtained from phish tank (i.e., http://data.phishtank.com/data/online-valid.csv) is partitioned into a number of disjoint subsets that are stored as grids on a Hadoop Distributed File System (HDFS). After obtaining each key–value pair, each mapper loads one subset from the HDFS into memory. Each mapper emits three types of intermediate keys, β_U, β_P, and β_p, that denote the increments of U_n, P_n, and p_n, respectively.

Three reducers are used to train the GLDG model. Each reducer reads one type β_U_n, β_P_n, or β_p_n of the intermediate key–value pairs as input and applies the reduce function to initially calculate the increments (i.e., computationally efficient feature) and the update parameter (i.e., website phishing detection). The reducer then obtains the mapper ID as the output *key* and the resulting website phishing detected value as the output.

4 Evaluation

The PLS-GLGB technique attains minimum phishing detection overhead compared with other existing methods because of the application of the probabilistic latent preprocessing algorithm. Two functions, namely, probability intersection and data maximum likelihood, are used to determine the missing data in the phish tank dataset by applying this algorithm. Web phishing is processed only after the missing data are determined, subsequently reducing the overhead incurred in web phishing.

In this section, the PLS-GLGB for website phishing detection is proposed. The method is implemented in JAVA MapReduce parallel programming language with CloudSim simulator. Version 1.1.2 is adopted for the Hadoop cluster. The performance is assessed by the boosting technique to evaluate the effectiveness of our method. Four metrics, precision, recall, time, and overhead, are used to analyze the results of our method.

Misclassification rate is evaluated by means of a confusion matrix. Comparisons are made with two state-of-the-art methods, namely, OFS-NN [1] and CatchPhish [2], using the phish tank dataset. Tab. 1 presents the features and the corresponding description of the phish details used for simulation.

Table 1: Phish tank dataset details

S. No	Features	Description
1	Phish_id	The ID number by which a phish tank is referred to
2	Phish_detail_url	Phish tank detail URL for the phish
3	url	The phish URL
4	Submission_time	The date and time at which the phish was reported to the phish tank
5	Verified	Whether or not the phish has been verified
6	Verification time	The date and time at which the phish was verified
7	Online	Whether or not the phish is online and operational
8	Target	The name of the company or brand the phish is impersonating

5 Discussion

In this section, the performance measure of three parameters, namely, phishing detection time, phishing detection overhead, and confusion matrix, using the proposed PLS-GLGB method is compared with that of the existing OFS-NN [1] and CatchPhish [2] methods. Details are provided in the following sections with the aid of table value and graphical representation.

5.1 Performance Measure of Phishing Detection Time

Phishing detection time refers to the time consumed in detecting the website phishing. A considerable amount of time is consumed to detect website phishing, and it is mathematically formulated as follows:

$$PD_T = \sum_{i=1}^{n} p_i * Time\ H(PU_i^{t+1}) \tag{13}$$

In Eq. (13), phishing detection time PD_T is inferred from the latent phish identifier variable p_i and the time consumed in detecting $Time\ H(PU_i^{t+1})$. It is measured in terms of milliseconds (ms). Tab. 2 shows the results of our feature selection. The total time for detecting web phishing with 50 phish_id is 6.75 ms. This result is exceptionally lower than the total times of 7.75 and 9.25 ms for [1] and [2], respectively.

Fig. 3 shows the phishing detection time with respect to different numbers of latent phish identifier variables ranging between 50 and 150. The graph suggests that the phishing detection time is directly proportional to the number of phish_id. An increase in the number of phish_id increases the features considered for phishing detection, therefore increasing phishing detection time. However, a comparison made with 50 phish identifiers indicated that using PLS-GLGB consumes 6.75 ms while [1] and [2] consume 7.75 and 9.25 ms, respectively.

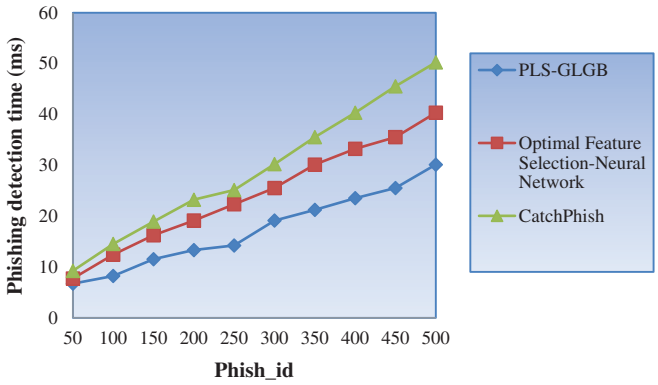

Figure 3: Graphical representation of phishing detection time

This result indicates that the comparison of PLS-GLGB with two other methods show that PLS-GLGB consumes minimum phishing detection time compared with [1] and [2]. This finding is attributed to the application of the PILSP model. By applying this model, in certain path information, data are missing, the path information are discarded from being processed, and only the prevailing path with sufficient information is used for detecting web phishing. Therefore, the web phishing detection time using PLS-GLGB is reduced by 28% compared with [1] and 40% compared with [2].

5.2 Performance Measure of Phishing Detection Overhead

Phishing detection overhead refers to the memory time incurred during the detection of website phishing. A significant amount of memory is incurred while detecting website phishing, and it is mathematically expressed as follows:

$$PD_O = \sum_{i=1}^{n} p_i * Mem\ H(PU_i^{t+1}) \tag{14}$$

In Eq. (14), phishing detection overhead PD_O is inferred from the latent phish identifier variable p_i and the memory incurred in detecting $Mem\ H(PU_i^{t+1})$. It is measured in terms of kilobytes (KB). Tab. 3 shows the results of our feature selection in terms of overhead consumed.

Table 2: Phishing detection time

Phish_id	Phishing detection time (ms)		
	PLS-GLGB	Optimal Feature Selection-Neural Network	CatchPhish
50	6.75	7.75	9.25
100	8.25	12.45	14.55
150	11.55	16.26	18.95
200	13.35	19.15	23.25
250	14.25	22.35	25.15
300	19.15	25.55	30.25
350	21.25	30.15	35.55
400	23.55	33.25	40.35
450	25.55	35.55	45.55
500	30.15	40.35	50.25

Table 3: Phishing detection overhead

Phish_id	Phishing detection overhead (KB)		
	PLS-GLGB	Optimal Feature Selection-Neural Network	CatchPhish
50	100	150	200
100	200	300	400
150	300	450	600
200	400	600	700
250	500	650	800
300	700	800	950
350	850	900	1100
400	1000	1250	1350
450	1150	1400	1500
500	1300	1550	1700

Fig. 4 shows the phishing detection overhead for 500 different identifiers collected at different submission times and possessing different URLs. The figure shows that the phishing detection overhead is directly proportional to the number of phish identifiers considered for conducting simulations. Thus, increasing the number of phish identifiers increases the number of URLs to be processed and the targets to be reached, evidently increasing the overhead incurred.

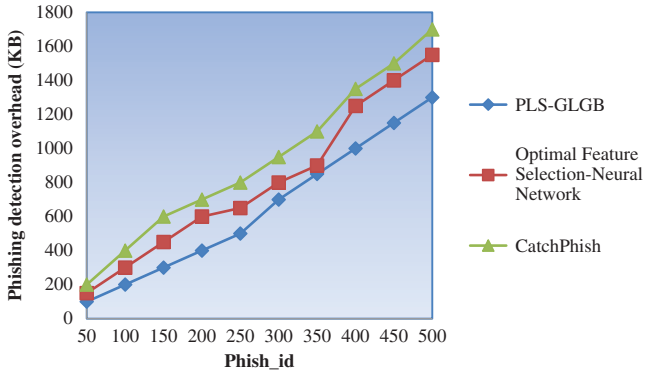

Figure 4: Graphical representation of phishing detection overhead

Table 4: Example of a confusion matrix

N = 500 (phish IDs)	Predicted: NO	Predicted: YES
Actual: NO	25	45
Actual: YES	20	410

Table 5: Updated confusion matrix

N = 500 (phish IDs)	Predicted: NO	Predicted: YES	
Actual: NO	TN = 25	FP = 45	70
Actual: YES	FN = 20	TP = 410	430
	45	455	

Table 6: Misclassification rate (using the confusion matrix)

N = 500	PLS-GLGB	Optimal Feature Selection-Neural Network	CatchPhish
Misclassification rate	0.13	0.16	0.20

However, the simulations conducted with 50 phish IDs indicated that the overhead values incurred using PLS-GLGB were 100 KB and 150 and 200 KB when applied using [1] and [2]. Therefore, the overhead is comparatively higher using PLS-GLGB than [1] and [2] because of the application of the probabilistic latent preprocessing algorithm. Two functions, namely, PIF and data maximum likelihood, are used to determine the missing data in the phish tank dataset. Web phishing is processed only after the missing data are determined. Thus, the overall overhead incurred in web phishing using PLS-GLGB is reduced by 23% compared with [1] and 35% compared with [2].

5.3 Performance Measure of the Confusion Matrix

The confusion matrix corresponds to a table that is utilized to describe the performance of the GLDG classifier on a set of test data for which the true values are known. Tab. 4 shows an example of a confusion matrix.

The table (i.e., matrix) shows two probable predicted classes, namely, YES and NO; YES predicts the presence of a phishing attack, and NO indicates no attack. A total of 500 phish IDs are considered for prediction (i.e., 500 phish IDs were tested for the presence of phish attack). Among the 500 cases, the classifier predicted YES 455 times and NO 45 times. In reality, 430 phish IDs in the sample had an attack and 70 phish IDs did not. Tab. 5 represents the updated confusion matrix.

Several metrics, such as accuracy, misclassification rate, true positive rate, false positive rate, true negative rate, and precision, are usually evaluated from the confusion matrix for a binary classifier. In our work, misclassification rate is used and mathematically expressed as follows:

$$E = \frac{FP + FN}{T} \tag{15}$$

In Eq. (15), $E(using\ \text{PLS} - \text{GLGB}) = \frac{45+20}{500} = 0.13$, E(using [1]) $= \frac{60+20}{500} = 0.16$, and E(using [2]) $= \frac{80+20}{500} = 0.20$. Tab. 6 shows the final misclassification rate using the confusion matrix.

Misclassification rate using the proposed PLS-GLGB was 0.13 and 0.16 and 0.20 using [1] and [2], respectively. The results indicate that the misclassification rate using the PLS-GLGB method is less than [1] and [2] because of the application of the GLDG website phishing detection algorithm. Contrast divergence function is utilized to master the data space, which in turn minimizes the error. In addition, sample vector is rationalized via sine and cosine association, where the primary individual population jumps out of the optimality via the greedy levy operation. Thus, misclassification using the PLS-GLGB method is less than that when using [1] and [2].

6 Conclusion

A website phishing attack is a very customary social engineering model that attacks an institution of the end users. It is conceived to be one of the most dangerous attacks in recent years. Several studies on detecting and alleviating phishing attacks have been conducted. Conventional methods focus on the utilization of neural network models to address phishing attacks. We proposed PLS-GLGB for website phishing detection by using MapReduce as a novel solution to prevent website phishing attacks. PLS-GLGB has the ability to handle network traffic dynamics containing phishing attacks. It can provide a computationally efficient phishing detection mechanism because it utilizes probabilistic latent preprocessing. Specifically, we initially obtained computationally efficient preprocessed phish data using this preprocessing model. Then, we built a trustworthy system using GLDG website phishing detection to minimize the misclassification rate generated via confusion matrix. We theoretically and experimentally evaluated PLS-GLGB, and experiments showed minimum phishing detection time and overhead at minimum misclassification rate compared with the performance of state-of-the-art works. We performed phishing detection in websites. However, time consumption and overhead during phishing detection in websites were not reduced at the required level. A deep learning method can be introduced in the future to further enhance the phishing detection performance.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare no conflicts of interest to report.

References

[1] E. Zhu, Y. Chen, C. Ye, X. Li and F. Liu, "OFS-Nn: An effective phishing websites detection model based on optimal feature selection and neural network," *IEEE Access*, vol. 7, pp. 73271–73284, 2019.

[2] R. S. Rao, T. Vaishnavi and A. R. Pais, "Catchphish: Detection of phishing websites by inspecting URLs," *Ambient Intelligence & Humanized Computing*, vol. 11, pp. 813–825, 2020.

[3] A. Cuzzocrea, F. Martinelli and F. Mercaldo, "A machine learning framework for supporting intelligent web-phishing detection and analysis," in *Proc. IDEAS*, New York, NY, USA, pp. 1–3, 2019.

[4] O. K. Sahingoz, E. Buber, O. Demir and B. Diri, "Machine learning based phishing detection from URLs," *Expert Systems with Applications*, vol. 117, pp. 345–357, 2019.

[5] R. S. Rao and A. R. Pais, "Two level filtering mechanism to detect phishing sites using lightweight visual similarity approach," *Ambient Intelligence & Humanized Computing*, vol. 11, pp. 3853–3872, 2020.

[6] G. Varshney, M. Misra and P. K. Atrey, "A survey and classification of web phishing detection schemes," *Security & Communication Networks*, vol. 9, no. 18, pp. 6266–6284, 2016.

[7] J. Mao, J. Bian, W. Tian, S. Zhu, T. Wei *et al.*, "Phishing page detection via learning classifiers from page layout feature," *EURASIP Journal on Wireless Communications & Networking*, vol. 2019, no. 43, pp. 1–14, 2019.

[8] A. A. Orunsolu, A. S. Sodiya and A. T. Akinwale, "A predictive model for phishing detection," *Journal of King Saud University-Computer and Information Sciences*, pp. 1–16, 2019.

[9] T. Chin, K. Xiong and C. Hu, "Phishlimiter: A phishing detection and mitigation approach using software-define networking," *IEEE Access*, vol. 6, pp. 42516–42531, 2018.

[10] E. Zhu, Y. Chen, C. Ye, X. Li and F. Liu, "OFS-Nn: An effective phishing website detection model based on optimal feature selection and neural network," *IEEE Access*, vol. 7, pp. 73271–73284, 2019.

[11] G. G. Geng, X. D. Lee and Y. M. Zhang, "Combating phishing attacks via brand identity and authorization features," *Security & Communication Networks*, vol. 8, no. 6, pp. 888–898, 2014.

[12] T. Feng, and C. Yue, "Visualizing and interpreting RNN models in URL-based phishing detection," *Proc. SACMAT*, Barcelona, Spain, pp. 13–24, 2020.

[13] W. Wang, F. Zhang, X. Luo and S. Zhang, "PDRCNN: Precise phishing detection with recurrent convolutional neural networks," *Security & Communication Networks*, vol. 2019, pp. 1–15, 2019.

[14] R. S. Rao and A. R. Pais, "Detection of phishing websites using an efficient feature-based machine learning framework," *Neural Computing & Applications*, vol. 31, pp. 3851–3873, 2018.

[15] T. Gangavarapu, C. D. Jaidhar and B. Chanduka, "Applicability of machine learning in spam and phishing email filtering: Review and approaches," *Artificial Intelligence Review*, vol. 53, pp. 5019–5081, 2020.

[16] R. T. Pashiri, Y. Rostami and M. Mahrami, "Spam detection through feature selection using artificial neural network and sine–cosine algorithm," *Mathematical Sciences*, vol. 14, pp. 193–199, 2020.

[17] P.Yi, Y. Guan, F. Zou, Y. Yao, W. Wanget *et al.*, "Web phishing detection using a deep learning framework," *Wireless Communications & Mobile Computing*, vol. 2018, pp. 1–10, 2018.

[18] M. Satheesh Kumar, K. G. Srinivasagan and J. Ben-Othman, "Sniff phish: A novel framework for resource intensive computation in cloud to detect email scam," *Transactions on Emerging Telecommunications Technologies*, vol. 30, no. 6, pp. e3590, 2019.

[19] M. Hajiali, M. Amirmazlaghani and H. Kordestani, "Preventing phishing attacks using text and image watermarking," *Concurrency & Computation: Practice & Experience*, vol. 31, no.13, pp. e5083, 2018.

[20] A. S. Bozkir and M. Aydos, "LogoSENSE: A companion HOG based logo detection scheme for phishing webpage and E-mail brand recognition," *Computer & Security*, vol. 95, pp. 101855–18, 2020.

[21] A. Aassal, S. Baki, A. Das and R. M. Verma, "An in-depth benchmarking and evaluation of phishing detection research for security needs,", *IEEE Access*, vol. 8, pp. 22170–22192, 2020.