Malicious URLs detection using data streaming algorithms

Kayode Sakariyah Adewole, Muiz Olalekan Raheem*), Muyideen Abdulraheem, Idowu Dauda Oladipo, Abdullateef Oluwagbemiga Balogun, Omotola Fatimah Baker

Department of Computer Science, Faculty of Communication and Information Sciences, University of Ilorin P.M.B 1515, Ilorin, Kwara State, Nigeria

Cara sitasi: K. S. Adewole, M. O. Raheem, M. Abdulraheem, I. D. Oladipo, A. O. Balogun, and O. F. Baker, "Malicious URLs detection using data streaming algorithms," *Jurnal Teknologi dan Sistem Komputer*, vol. 9, no. 4, pp. 224-229, 2021. doi: 10.14710/jtsiskom.2021.13965, [Online].

***Abstract** - As a result of advancements in technology and technological devices, data is now spawned at an infinite rate, emanating from a vast array of networks, devices, and daily operations like credit card transactions and mobile phones. Datastream entails sequential and real-time continuous data in the inform of evolving stream. However, the traditional machine learning approach is characterized by a batch learning model. Labeled training data are given apriori to train a model based on some machine learning algorithms. This technique necessitates the entire training sample to be readily accessible before the learning process. The training procedure is mainly done offline in this setting due to the high training cost. Consequently, the traditional batch learning technique suffers severe drawbacks, such as poor scalability for real-time phishing websites detection. The model mostly requires re-training from scratch using new training samples. This paper presents the application of streaming algorithms for detecting malicious URLs based on selected online learners: Hoeffding Tree (HT), Naïve Bayes (NB), and Ozabag. Ozabag produced promising results in terms of accuracy, Kappa and Kappa Temp on the dataset with large samples while HT and NB have the least prediction time with comparable accuracy and Kappa with Ozabag algorithm for the real-time detection of phishing websites.*

***Keywords** – data streaming; phishing; Naïve Bayes; machine learning; Hoeffding Tree*

I. Introduction

The introduction of new digital technology has profoundly impacted business development and promotion across many areas, including e-banking, social networking sites, and others. The World Wide Web (WWW) has steadily grown as its technological advances develop daily. Unfortunately, technological progress comes in conjunction with new, sophisticated thresholds [1]. There are numerous attacks on the network; one of them is phishing, in which the attacker impersonates himself as genuine and takes credentials from the user [2], [3]. An URL originates records and other services on the WWW. It has two essential features: protocol identifier, which reveals the protocol used, and resource name, which specifies the IP address or the domain name of the resource location.

Malicious URLs are compromised links used for cyber-attacks. Choi et al. [4] noted that about one-third of all websites hosted online are potentially malicious, which shows the increase in utilizing malicious URLs to commit cybercrimes. These malicious sites are created to look much like legitimate ones. In addition, a site with several unsolicited contents in the form of phishing, spam, or drive-by download to launch cyber-attacks on unsuspecting users is termed a malicious website.

Unwary visitors of such websites become victims of several scams, including information theft, monetary loss, and malware installation. The drive-by-download is an accidental download of malware upon visiting a website. These attacks are typically carried out by exploiting vulnerabilities in plugins or inserting malicious scripts using JavaScript codes [5]. The goal of phishing and social engineering attacks is to trick legitimate users into revealing sensitive details by masquerading as genuine websites. Spam involves the distribution of unsolicited messages to disseminate advertainment, news, or malicious links. Available statistics showed that these attacks had caused billions of dollars worth of damage [6].

Research effort must be channeled towards developing effective methods to expose these malicious URLs on time. These methods will assist in combating a large number of and a variety of cyber-security attacks. Over the years, researchers have developed different solutions to identify malicious links. The most prevalent method deployed by some antivirus groups is the blacklist-based approach. Blacklists are known databases of malicious URLs confirmed through crowd-sourcing or other annotation means. A typical example of a blacklist database is PhishTank.

Interestingly, the blacklist technique is very fast due to the short time required to execute the simple query for link verification. Therefore, the method is quick to implement. Nevertheless, the major issue with this approach is the difficulty in maintaining the exhaustive list of malicious URLs daily.

In the last decade, several studies like Zamir et al. [5], Nepali and Wang [2], Adewole et al. [7], Kuyama et al. [8],

*) Corresponding author (Muiz Olalekan Raheem)
Email: raheem069.mo@gmail.com

Submitted: 29 October 2020; Revised: 7 July 2021; Accepted: 9 July 2021; Published: 31 October 2021

and Gugelmann et al. [9] have all applied machine learning (ML) techniques in providing solutions to malicious URLs identification. ML technique employs a set of annotated URLs as training data, and with the use of important features, the method learns predictive function to distinguish malicious URLs from benign links. This learning ability makes it possible to generalize the predictive model to classify new URLs instead of the traditional blacklist approach.

Notwithstanding, many studies have utilized other machine learning techniques such as Extreme Machine Learning (ELM) [10], BART, CART [11] to devise an appropriate and suitable solution for detecting malicious URLs. However, their techniques demand that the entire training dataset be readily available before the learning process, and the training procedure is frequently conducted in an offline environment. Consequently, the significant drawbacks of batch learning approaches include the inability to generalize for large-scale applications due to the need to re-train the models from scratch for new training samples. Therefore, this study proposes the detection of malicious websites using data streaming algorithms for real-time/fast-moving data where URLs are treated as a continuous stream of data. The online learners used include Hoeffding Tree, Naïve Bayes, and Ozabag.

II. Research Methods

This section gives a detailed description of the proposed framework, the techniques used in obtaining the datasets, the streaming algorithms employed in classifying the datasets, and the evaluation metrics used to measure the proposed models' performance. The proposed technique aims to detect phishing websites using streaming techniques, including Hoeffding Tree, Naïve Bayes, and OzaBag. Figure 1 shows the conceptual framework of this study.

A. Phishing datasets

The datasets used in this study were obtained from the prominent UCI machine learning repository. The first dataset, which is later referred to as MaliciousDataset1, has a sum of 1,353 instances of malicious and benign URLs[1]. This dataset consists of ten attributes. The second dataset, denoted as MaliciousDataset2, has a total of 11,055 URLs instances[2]. This dataset is made up of thirty features. The characteristics of the datasets are as presented in Table 1.

Modern datasets are said to grow in three dimensions –features, examples, and cardinality– making dimensionality reduction a mandatory step if standard algorithms are used. Preprocessing procedures that include data reduction perform this generalization by

[1] Can be accessed at https://archive.ics.uci.edu/ml/machine-learning-databases/00379/

[2] Can be accessed at https://archive.ics.uci.edu/ml/machine-learning-databases/00327/

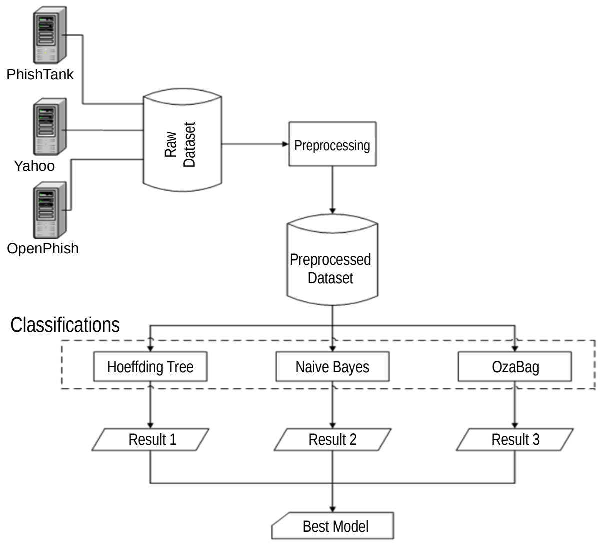


Figure 1. Conceptual framework of the study

Table 1. Details of the data in this research

Name	Type	Features	Instances	Class distribution
Malicious Dataset1	Integer	10	1,353	Malicious (702) Genuine (548) Suspicious (103)
Malicious Dataset2	Integer	30	11,055	Malicious (4898) Genuine (6157)

choosing and removing redundant and noisy features or discretizing complex continuous feature spaces. Therefore, this reduction permits maintaining the original structure and meaning of the input data and, at the same time, obtaining a much more manageable size of data.

B. Classification methods

The Hoeffding Tree, Naïve Bayes, and OzaBag are the proposed techniques to detect phishing websites using streaming techniques in this study.

Hoeffding Tree (HT) is known as the streaming decision tree induction. It was derived from the Hoeffding bound employed in the tree induction. In the data stream setting, where saving all instances seems impossible, the key concern of the decision tree is that the right qualities for division are to be determined by recycling those instances [12]. A distinguishing characteristic of HT generates a tree that converges with sufficiently large data into the tree constructed by a batch learner.

The main idea is that Hoeffding bound provides some level of confidence on the best attribute to split the tree. Therefore, it is possible to build the model using some instances already known. It is often regarded as the decision tree for streaming data. Based on Hoeffding's bound, a confidence interval for the entropy estimation is expressed in (1). The ε is the estimated value, n is the number of samples accumulated at the node, R is the random variable's scope, and δ is the desired chance that the approximation is not within its estimated value. The HT algorithm is as presented in Algorithm 1 [13].

$$\in = \sqrt{\frac{R^2 \ln 1/\partial}{2n}} \tag{1}$$

Naïve Bayes (NB) performs a classical Bayesian prediction with a naive supposition that all inputs are independent. NB is a classification algorithm widely known for its simplicity and low computational cost. Given *n* different classes, the trained NB classifier predicts for every unlabeled instance *I* the class *C* to which it belongs with high accuracy. NB algorithm represents attribute values using vector and probabilistic models. In Bayes classifiers, each attribute in the training data is independent of another. NB is a conditional probability model that represents features using vectors and assigns probabilities to each sample for each of *n* possible outcomes. Using Bayes' theorem, the conditional probability is decayed, which means X_n is the *n*th attribute of a total of *n* attributes. For categorical features, the conditional probability is $P(X_n| C_k)$, which is a tuple of C_k classes. X_n divides tuples of C_k classes into the data set. Algorithm 2 explains the Naïve Bayes algorithm.

Ozabag is a prominent ensemble learning algorithm used in evolving data streams. Its distinguishing characteristic is not only its higher performance than single classifiers but also its ability to add, remove, and update base classifiers when drifts occur. Algorithm 3 explained the Ozabag learner where *S* denotes the data stream, *M* represents the number of base models, *Y* denotes the set of class labels, and *x* denotes the features vector of instances.

Algorithm 1. Hoeffding Tree

```
Input: A stream of labeled data, confidence parameter $\delta$
Output: Tree Model
Let HT be a tree with a single leaf (root)

Initialization: initialize counts $n_{ijk}$ at the root
1: for each data$(x,y)$ in Stream do
2:     HTGrow($(x,y)$, HT, ẟ)
3: HTGrow($(x,y)$, HT, ẟ)
4: sort $(x,y)$ to leaf $l$ using HT
5: update counts $n_{ijk}$ at leaf $l$
6:  if data seen at $l$ so far are not all of the same class then
7:      compute $G$ for each attribute
8:        if $G(best\ attribute) - G(second\ best) > \epsilon$ then
9:            split leaf on best attribute
9:  for each branch do
10:     start a new leaf and initialize counts
```

Algorithm 2. Naive Bayes

```
Input: Training dataset T,  F = (f1,f2,f3,…, fn)
     /* F: value of the predictor feature in testing dataset*/
Output: A class of testing dataset

Initialization: read the training dataset T
1: Calculate the mean and standard deviation of
        predictor variables in each class
   do
2:    Find the probability of fi by the Gauss density
          equation in each class
3: Until the probability of all predictor variables
          (f1,f2,f3,…, fn) has been calculated
4: Calculate the likelihood for each class
5: Get the greatest likelihood
```

Algorithm 3. Ozabag

```
Input: $S$ (Stream data), $M$ (Number of Base Model),
    $Y$ (set of class labels)
Output: predicted class label (Majority vote)

Initialization: initialize M base classifier: $h_1, h_2, \ldots, h_M$
1:  while HasNext($S$) do
2:      $(x, y) \leftarrow$ NextInstance($S$)
3:      for $m \leftarrow 1, \ldots, M$ do
4:              $w \leftarrow$ Poisson(1)
5:              train $h_m$ with instance using weight $w$
6:  return the overall class label through majority vote
```

C. Evaluation metrics.

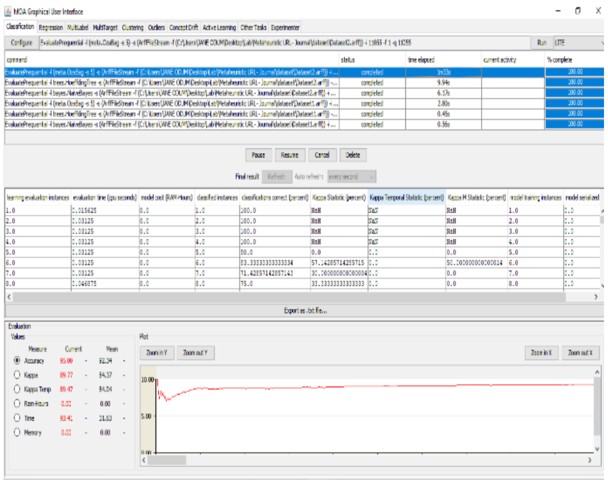
Evaluation metrics are employed to assess the machine learning model(s) performance. In this study, the data streaming models were evaluated using the accuracy, kappa statistics, and CPU time. Accuracy is a used performance measure representing the percentage of correctly classified instances, as presented in (2). The TP expresses a true positive, TN true negative, FP false positive, and FN false negative. CPU time calculates how much the learning and estimation processes would cost depending on the time the computer processor has taken.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2}$$

III. Results and Discussion

This chapter explains the results of the methods utilized in malicious URLs detection based on streams of data. It also shows the results of the experiment performed, the tools used, and the analysis of the results.

For each learner algorithm examined in this study, the datasets were passed using a sampling frequency of a stream per instance to train and test. In addition, the 'evaluatePrequential' class was used in analyzing this learner. This class evaluates a classifier on a data stream by testing, then training with each sample in sequence. Also, the experiment was performed on a system with a Windows 10 Operating System, a Random Access Memory (RAM) of 4 GB, an Intel Core i5 CPU with a processing speed of 2.50 GHz, and a Hard drive capacity of 500 GB.



```
EvaluatePrequential -l (meta.OzaBag -s 5) -s (ArffFileStream -f (C:\Users\JANE ODUM\Desktop\Lab\Metaheuristic URL - Journal\dataset\Dataset2.arff)) -i 11055 -f 1 -q 11055
```

command	status	time elapsed	current activity	% complete
EvaluatePrequential -l (meta.OzaBag -s 5) -s (ArffFileStream -f (C:\Users\JANE ODUM\Desktop\Lab\Metaheuristic URL - Journal\dataset\Dataset2.arff)) -i ...	completed	1m33s		100.00
EvaluatePrequential -l trees.HoeffdingTree -s (ArffFileStream -f (C:\Users\JANE ODUM\Desktop\Lab\Metaheuristic URL - Journal\dataset\Dataset2.arff)) -...	completed	9.94s		100.00
EvaluatePrequential -l bayes.NaiveBayes -s (ArffFileStream -f (C:\Users\JANE ODUM\Desktop\Lab\Metaheuristic URL - Journal\dataset\Dataset2.arff)) -i ...	completed	6.17s		100.00
EvaluatePrequential -l (meta.OzaBag -s 5) -s (ArffFileStream -f (C:\Users\JANE ODUM\Desktop\Lab\Metaheuristic URL - Journal\dataset\Dataset1.arff)) -i ...	completed	2.80s		100.00
EvaluatePrequential -l trees.HoeffdingTree -s (ArffFileStream -f (C:\Users\JANE ODUM\Desktop\Lab\Metaheuristic URL - Journal\dataset\Dataset1.arff)) -...	completed	0.45s		100.00
EvaluatePrequential -l bayes.NaiveBayes -s (ArffFileStream -f (C:\Users\JANE ODUM\Desktop\Lab\Metaheuristic URL - Journal\dataset\Dataset1.arff)) -i ...	completed	0.56s		100.00

learning evaluation instances	evaluation time (cpu seconds)	model cost (RAM-Hours)	classified instances	classifications correct (percent)	Kappa Statistic (percent)	Kappa Temporal Statistic (percent)	Kappa M Statistic (percent)	model training instances	model serialized
1.0	0.015625	0.0	1.0	100.0	NaN	NaN	NaN	1.0	0.0
2.0	0.03125	0.0	2.0	100.0	NaN	NaN	NaN	2.0	0.0
3.0	0.03125	0.0	3.0	100.0	NaN	NaN	NaN	3.0	0.0
4.0	0.03125	0.0	4.0	100.0	NaN	NaN	NaN	4.0	0.0
5.0	0.03125	0.0	5.0	80.0	0.0	0.0	0.0	5.0	0.0
6.0	0.03125	0.0	6.0	83.33333333333334	57.14285714285715	0.0	50.000000000000014	6.0	0.0
7.0	0.03125	0.0	7.0	71.42857142857143	30.000000000000004	0.0	0.0	7.0	0.0
8.0	0.046875	0.0	8.0	75.0	33.33333333333333	0.0	0.0	8.0	0.0

Measure	Current		Mean	
Accuracy	95.00	-	92.34	-
Kappa	89.77	-	84.37	-
Kappa Temp	89.47	-	84.04	-
Ram-Hours	0.00	-	0.00	-
Time	93.41	-	31.03	-
Memory	0.00	-	0.00	-

Figure 2. MOA working environment

A. Performance evaluation and interpretation

The experiment was conducted using Massive Online Analysis (MOA), an open-source framework for real-time learning from data streams. MOA is a machine learning framework that implements several algorithms for online learning based on evolving data streams. It has a collection of offline and online methods alongside suitable tools for evaluation. The framework handles bagging, boosting, and Hoeffding Trees, with and without Naïve Bayes classifiers employed at the leaves.

Also, MOA promotes two-way engagement with the Waikato Environment for Knowledge Analysis (WEKA). Its set of learners and stream generators have several available functional algometries, including Bayesian classifier, decision tree classification, regression, and clustering, which can be accessed from a graphical user interface (GUI) or a command-line interface (CLI). Figure 2 shows a sample screenshot of the MOA working environment, demonstrating how instances are trained and tested during data stream learning.

Hoeffding Tree was employed on both MaliciousDataset1 and MaliciousDataset2. The evaluation of this online algorithm is based on mean accuracy, mean kappa, and mean kappa temp. Based on the experiments conducted on MOA with tieThreshold 0.05 (Table 2), HT had an average accuracy of 77.78 % and 92.31 % on MaliciousDataset1 and MaliciousDataset2. The Kappa and Kappa Temp on MaliciousDataset1 were 61.06 and 57.69. While on MaliciousDataset2, an average kappa and kappa temp of 84.30 and 83.96 were recorded, respectively.

Naive Bayes is a notable instance of incremental learner that was also used on both datasets. The results of this experiment on both MaliciousDataset1 and MaliciousDataset2 are as shown in Table 3. Based on the experimental result, NB achieved 78.61 % and 92.29 % accuracy on MaliciousDataset1 and MaliciousDataset2, respectively.

This Ozabag ensemble type of learner was also used on the datasets. It consists of ten Hoeffding trees with an ensembleSize of 10. The result of the experiment

Table 2. Hoeffding Tree performance on Malicious Dataset1 and MaliciousDataset2

Metrics	MaliciousDataset1	MaliciousDataset2
Accuracy (%)	77.78	92.31
Kappa	61.06	84.30
Kappa Temp	57.69	83.96

Table 3. Naive Bayes performance on MaliciousDataset1 and MaliciousDataset2

Metrics	MaliciousDataset1	MaliciousDataset2
Accuracy (%)	77.61	92.29
Kappa	62.94	84.27
Kappa Temp	59.28	83.92

Table 4. Ozabag Tree performance on MaliciousDataset1 and MaliciousDataset2

Metrics	MaliciousDataset1	MaliciousDataset2
Accuracy (%)	77.56	92.34
Kappa	60.43	84.37
Kappa Temp	57.40	84.04

when Ozabag learner was used on both datasets (MaliciousDataset1 and MaliciousDataset2) is as shown in Table 4. Ozabag had an accuracy of 77.56 % on MaliciousDataset1 with Kappa and Kappa Temp of 60.43 and 57.40, respectively. An accuracy of 92.34% was recorded on MaliciousDataset2 classification.

B. Performance analysis

To further investigate the performance of the streaming algorithms, the enactment of the individual streaming algorithm is compared and analyzed per dataset employed. Figure 3 and Figure 4 show the comparisons among the models. HT produced the best accuracy of 77.78 % while obtaining a very close outcome with NB based on Kappa and Kappa Temp on MaliciousDataset1. Ozabag leads the other two learners in terms of accuracy, Kappa and Kappa Temp achieving an overall accuracy of 92.34 % on MaliciousDataset2. This result is close to the outcome of HT, which produced an accuracy of 92.31 %. It is evident that the proposed models on MaliciousDataset2 surpass [14]-[16] in terms of accuracy, as presented in Table 5.

The streaming learners' performances were also compared based on the time each took in classifying the given stream of data based on MaliciousDataset1 and MaliciousDataset2. On MaliciousDataset1, HT took 0.24secs, Naïve Bayes (NB) took 0.33secs, and Ozabag took 1.29secs. Similarly, on MaliciousDataset2, HT took 3 secs; NB also took 3 secs, and Ozabag took 31.29 secs. The time comparisons are shown in Table 6. This result shows that the least performed learner based on time is Ozabag, as it takes more seconds than the other two learners. HT and NB produced promising results in the area.

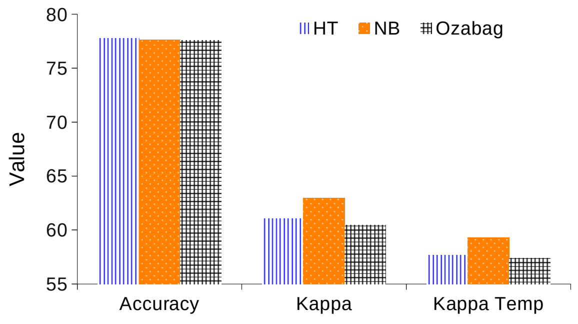


Figure 3. Model comparison on MaliciousDataset1

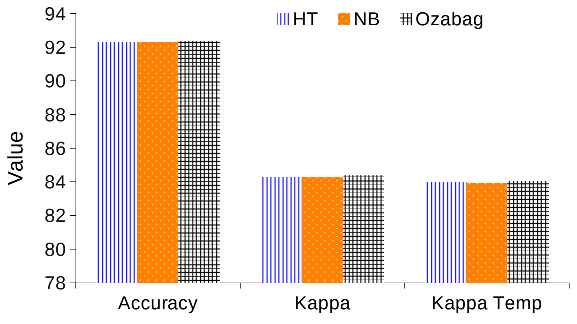


Figure 4. Model comparison on MaliciousDataset2

IV. Conclusion

The data streaming algorithm of HT produces the best accuracy for the malicious websites detection while obtaining a very close outcome with NB based on Kappa and Kappa Temp on MaliciousDataset1. Ozabag leads the other two learners in terms of accuracy, Kappa and Kappa Temp achieving an overall accuracy of 92.34 % on a large dataset of MaliciousDataset2. This result is close to the outcome of HT, which produced an accuracy of 92.31 %. Although Ozabag had promising results on MaliciousDataset2, its overall time consumption is more than HT and NB. Finally, further studies could investigate other data streaming techniques to detect phishing websites and consider more phishing data samples while exploring the impact of dimensionality and class imbalances.

Table 5. Accuracy comparison among models

Reference	Accuracy (%)	Models
Ferreira et al. [14]	87.61	ANN-MLP
Mohammad et al. [15]	92.18	Self-structuring NN
Ali and Ahmed [16]	91.13	GA-DNN
Proposed models	92.34	Ozabag
	92.29	Naive Bayes
	92.31	Hoeffding Tree

Table 6. Time comparison among proposed models

Name	Elapsed time (s)		
	HT	NB	Ozabag
MaliciousDataset1	0.24	0.33	1.29
MaliciousDataset2	3.27	3	31.63

References

[1] D. Sahoo, C. Liu, and S. C. Hoi, “Malicious URL detection using machine learning: A survey,” *arXiv:1701.07179v3 [cs.LG]*, 2019.

[2] R. K. Nepali and Y. Wang, “You look suspicious!!: Leveraging visible attributes to classify malicious short URLs on Twitter,” in *the 49th Hawaii International Conference on System Sciences*, Koloa, USA, Jan. 2016, pp. 2648-2655. doi: 10.1109/HICSS.2016.332

[3] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, “Identifying suspicious URLs: an application of large-scale online learning,” in *the 26th Annual International Conference on Machine Learning*, Quebec, Canada, Jun. 2009, pp. 681-688. doi: 10.1145/1553374.1553462

[4] H. Choi, B. B. Zhu, and H. Lee, “Detecting malicious web links and identifying their attack types,” *WebApps*, vol. 11, pp. 125-136, 2011.

[5] A. Zamir *et al.*, “Phishing web site detection using diverse machine learning algorithms,” *The Electronic Library*, vol. 38, no. 1, pp. 65-80, 2020. doi: 10.1108/EL-05-2019-0118

[6] R. Verma, D. Crane, and O. Gnawali, “Phishing during and after disaster: Hurricane Harvey,” in *2018 Resilience Week*, Denver, USA, Aug. 2018, pp. 88-94. doi: 10.1109/RWEEK.2018.8473509

[7] K. S. Adewole, A. G. Akintola, S. A. Salihu, N. Faruk, and R. G. Jimoh, “Hybrid rule-based model for phishing URLs detection,” in *International Conference for Emerging Technologies in Computing*, London, United Kingdom, Aug. 2019, pp. 119-135. doi: 10.1007/978-3-030-23943-5_9

[8] M. Kuyama, Y. Kakizaki, and R. Sasaki, “Method for detecting a malicious domain by using only well-known information,” *International Journal of Cyber-Security and Digital Forensics*, vol. 5, no. 4, pp. 166-174, 2016. doi: 10.17781/P002212

[9] D. Gugelmann, B. Ager, V. Lenders, and M. Happe, “Towards understanding upstream Web traffic,” in *International Wireless Communications and Mobile Computing Conference*, Dubrovnik, Croatia, Aug. 2015, pp. 538-544. doi: 10.1109/IWCMC.2015.7289141

[10] W. Zhang, Q. Jiang, L. Chen, and C. Li, “Two-stage ELM for phishing web pages detection using hybrid features,” *World Wide Web*, vol. 20, pp. 797-813, 2017. doi: 10.1007/s11280-016-0418-9

[11] H. Y. Abutair and A. Belghith, “Using case-based reasoning for phishing detection,” *Procedia Computer Science*, vol. 109, pp. 281-288, 2017. doi: 10.1016/j.procs.2017.05.352

[12] P. Domingos and G. Hulten, "Mining high-speed data streams," in *the sixth International Conference on Knowledge Discovery & Data*

Mining, Boston, USA, Aug. 2000, pp. 71-80. doi: 10.1145/347090.347107

[13] C. Manapragada, G. I. Webb, and M. Salehi, “Extremely fast decision tree,” in *the 24th International Conference on Knowledge Discovery & Data Mining*, London, United Kingdom, Jul. 2018, pp. 1953-1962. doi: 10.1145/3219819.3220005

[14] R. P. Ferreira et al., “Artificial neural network for websites classification with phishing characteristics,” *Social Networking*, vol. 7, no. 2, pp. 97-109, 2018. doi: 10.4236/sn.2018.72008

[15] R. M. Mohammad, F. Thabtah, and L. McCluskey, “Predicting phishing websites based on self-structuring neural network,” *Neural Computing and Applications*, vol. 25, pp. 443-458, 2014. doi: 10.1007/s00521-013-1490-z

[16] W. Ali and A. A. Ahmed, “Hybrid intelligent phishing website prediction using deep neural networks with genetic algorithm-based feature selection and weighting,” *IET Information Security*, vol. 13, pp. 659-669, 2019. doi: 10.1049/iet-ifs.2019.0006