PhisNet: Deep learning-based Hybrid and Ensemble Multi-level Approach for the detection of phishing websites

Jayesh Soni

jsoni@fiu.edu

Florida International University

Nagarajan Prabakar

Florida International University

Himanshu Upadhyay

Florida International University

Research Article

Keywords: Phishing Website Detection, Variational AutoEncoder, One Class Classifiers, Neural Network

Posted Date: April 18th, 2024

DOI: https://doi.org/10.21203/rs.3.rs-4283476/v1

Additional Declarations: The authors declare no competing interests.

PhisNet: Deep learning-based Hybrid and Ensemble Multi-level Approach for the detection of phishing websites

Jayesh Soni[1], Nagarajan Prabakar[2], Himanshu Upadhyay[3]

[1]*Applied Research Center, Florida International University, Miami 33174, USA.*
[2]*Knight Foundation School of Computing and Information Sciences, Florida International University, Miami 33174, USA.*
[3]*Electrical and Computer Engineering, Florida International University, Miami 33174, USA.*

Abstract

Phishing is a cyber-attack that intends to trick individuals into providing an attacker with sensitive information, such as login authorizations or business information. One tactic that attackers use is creating fake websites that mimic legitimate ones to fool victims into entering their information. A learning-based model can be used to analyze patterns in website content, structure, behavior, email text, sender, and links. These models can help identify phishing attempts and protect individuals and organizations from falling victim to these scams. In this research, we study and experiment with the deep learning-based algorithm in classifying phishing webpages from legitimate webpages that can be generalized across multiple domains. We used the open-source benchmark dataset, "Phishing Dataset for Machine Learning," available on Kaggle. We propose a hybrid and a multi-level ensemble approach for phishing website detection. Several one-class classifiers are trained on the first level, and the Variational Autoencoder (VAE) is used to reduce the size of feature vectors. Each one class classifiers have its own strength and limitations, and thus an adaptive weightage approach is applied. At the second level, a multilayer neural network is trained for classification. Further, the training time is reduced due to feature reduction in the latent space. Our experimental results classify phishing websites from legitimate websites with 98% accuracy.

Keywords: Phishing Website Detection, Variational AutoEncoder, One Class Classifiers, Neural Network

1. INTRODUCTION

Social engineering is a form of attack wherein hackers use different ways to deceive an individual, usually through phishing. Phishing websites are one common technique used by attackers to execute their attacks. Research activities in information security have marked the detection of phishing websites as significant in preventing unauthorized access. Identifying websites is a crucial part of cybersecurity that aims to safeguard people and organizations from being damaged by malicious attacks such as phishing. The damage caused by these attacks can be economic and reputationally damaging because they typically result in the theft of critical data such as login credentials and financial information.

One way of combating phishing is by using anti-phishing software, browser extensions, and other blocking technologies to prevent it from happening. This reduces the risk of being a victim of such attacks that can go unnoticed by other detection methods. A high level of security is also required because GDPR and all other industry standards in data protection demand these steps based on how much private and financial information about countless people should be out there. Some companies still need to meet requirements so that they might incur penalties, including fines for non-compliance. However, they implement measures to keep personal and financial data always under lock and key.

Phishing is a serious danger and must be controlled in a way that addresses regulatory compliance and meets industry standards. While it is of the utmost importance to shield the organization from such incidents, other internal measures, like implementing control systems for digital security, must also be enforced to prevent data theft that could result in unnecessary damage to its reputation.

The method advised to tackle the issue suggests the use of blacklists and whitelists; by creating a list of identified phishing pages; these methods demonstrate how data could be withheld from users by not allowing access or issuing warnings [1]. However, there is a limitation to this approach as it does not prevent new sites where criminals bypass manual systems or sites that are not widely known. Another method uses a heuristic and rules-based approach to identify unique features or patterns in phishing websites so that they can be blocked [2]. In addition, email filtering [3] is also used as a measure against phishing emails. In their process, such filters use various techniques to establish and block the phishing mails. To verify the email's authenticity, the content and sender details must be checked. Sometimes, human authentication can also prove to be an efficient means of doing this. Phishing websites are challenging to identify in many cases. For example, a company with an exceptional team regularly searches the Internet for such sites. Browser extensions and plugins [4] can also be considered. Various tools can detect and hinder phishing attempts by comparing the site URL against a list of known phishing sites and generating a warning if a match is found. Two-factor authentication [5] is also a good security measure. While impersonating someone is not permitted, the two-factor user verification process incorporates two components to authenticate the identity. Machine learning-based methods [6] for phishing detection have also been proposed. Phishing detection methods employ algorithms to recognize patterns and features that characterize phishing websites so that these factors can be used to identify newer ones. Some well-known machine learning models used for this purpose include Naive Bayes, Decision Trees (DT), Random Forest, Support Vector Machine (SVM), Artificial Neural Network, and Deep Learning.

One way they are superior to traditional methods is that, unlike older systems, which remain stuck in their knowledge base unless updated manually by humans, they can evolve with modern-day attack strategies and stop unknown or new phishing websites. Hossain et al. [7] have used URL lexical-based and domain-based features and have trained the Random Forest algorithm. Kiruthiga et al. [8] classified websites into legitimate, suspicious, and phishing websites using the trained DT and SVM to obtain the highest accuracy of 90.97%. In the meanwhile, some recent research has shifted the focus to deep learning approaches using Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), as well as Long Short-term Memory (LSTM) Networks. These methods have been identified as good approaches to detecting phishing websites based on analyzing the visual and textual features of the web page. Bahnsen et al. [9] used statistical and syntax features extraction from URL with LSTM [10] for classification. The detection of a phishing website is hard; the present restriction results from high dimensionality, which necessitates an inclusive way to address it.

A hybrid, multi-level, and deep learning system is designed to detect phishing websites. We train several weak classifiers in the first level based on one class and VAE. These algorithms transform the dataset into a feature space with fewer dimensions; hence, we get a final output of hybrid features. Multiple weak classifiers have their strengths and limitations. Thus, we perform an adaptive weighting approach for each weak classifier. In the second level, a neural network is trained to classify the website as either phishing or legitimate. This approach is highly accurate due to its reduced features.

2. METHODS

This section will discuss phishing website detection using a Hybrid and Ensemble Multi-level Approach. The proposed framework is shown in Figure 1. It involves two levels: Hybrid Feature Extraction and Phishing Website Detection.

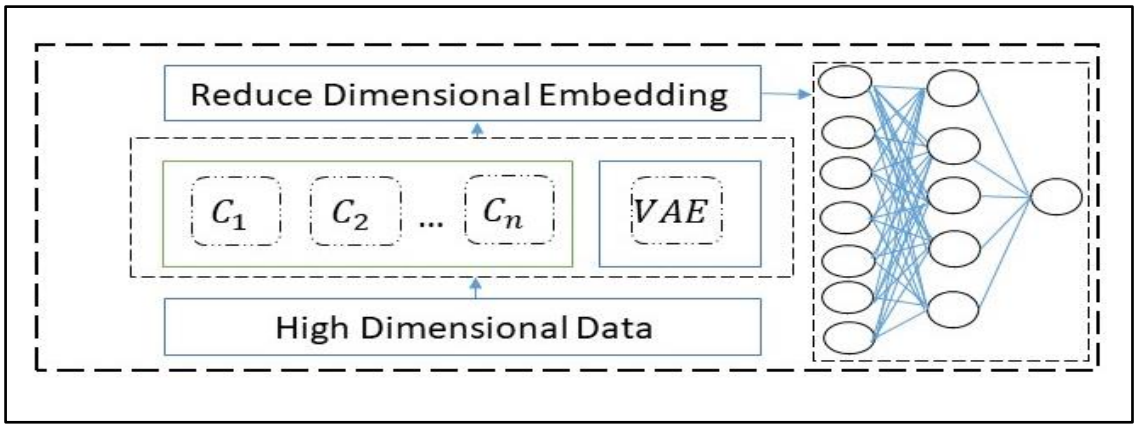

Figure 1. Phishing Website Detection Framework

2.1. Hybrid Feature Detection

The hybrid feature detection approach uses the strengths of many one-class classifiers to detect features in data. A one-class classifier is a machine learning model trained with data having only one class or category. Detecting features is usually aimed at determining whether some specific characteristics are present within the data or not. The following one-class classifiers as depicted in Figure 2 were trained on legitimate websites features: Isolation Forest [11], Local Outlier Factor [12], Mahalanobis Classifier [13], Elliptical Envelope [14], Minimum Covariance Determinant [15], and One-Class Support Vector Machine (OCSVM) [16]. These individual weak classifiers were represented as C_i for each weak classifier. These weak classifiers are combined at the end, deciding whether an attribute is present in the data set. One commonly used method of integration is voting, and here, the most basic way to combine the output of classifiers is to take a vote.

Voting: In this scenario, each classifier predicts whether or not a specific feature is present in the data by returning a binary decision, and then all these decisions are counted. The case with the most votes becomes the winner.

Weighted Voting: With weighted voting, a weight is assigned to each classifier that reflects the importance of accuracy in prediction. These weights help determine how much each classifier contributes towards shaping the ultimate decision.

To use adaptive weighting, a 10-fold cross-validation technique is utilized. This involves using the error-accumulated number of False Positives produced by each algorithm, as shown in **Equation 1**.

$$Avg.\ FP(C_i) = \frac{\sum_{i=1}^{k} FP(C_i)_k}{|Val\ Data| * k} \quad (1)$$

The adaptive weightage of each classifier is calculated as shown in **Equation 2**:

$$AdaptiveWeight_{Classifier} = \ = 1 - Avg.\ FP(C_1) \quad (2)$$

Finally, each class classifier's output produces a unique set of features.

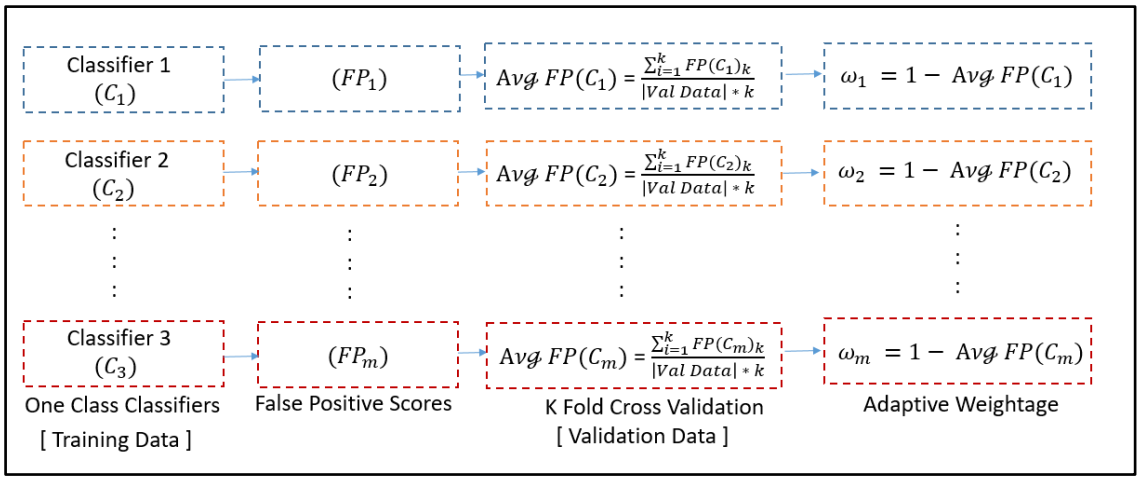

Figure 2. One Class Classifier Feature Detection

Another collection of features is produced by training a Variational AutoEncoder [17]. VAEs constitute a generative model in deep learning, and they attempt to define the internal structure of high-dimensional data with dimensionality reduction methods. This is accomplished through encoding input data into an efficient, smaller-dimensional vector named latent space; where, the code is decoded back into an approximately complete representation of the initial dataset. The VAE task aims to minimize the dissimilarity between the original input data and its reconstruction to learn an appropriate compact representation that carries all essential information about the data. With the use of VAEs and their dimensionality reduction method, it is easier to understand various datasets that may be very complicated.

A variational autoencoder comprises an encoder and a decoder, two major components in the architecture. The encoder takes the data and converts it into the latent space, while the decoder works with latent representation, transforming it back to the input data, as demonstrated in Figure 3. They minimize the reconstruction loss described in Equation 3 based on original data versus their rebuilt counterpart using regular training, which combines supervised and unsupervised learning techniques for VAEs.

In the encoding phase, the VAE is trained to extract information from an observed data set through regular learning. In the decoding stage, it learns how to recreate the primary data using an unsupervised model of learning. The combined approach of supervised and unsupervised methods enables VAEs to learn a compressed and meaningful data representation for subsequent analysis or processing.

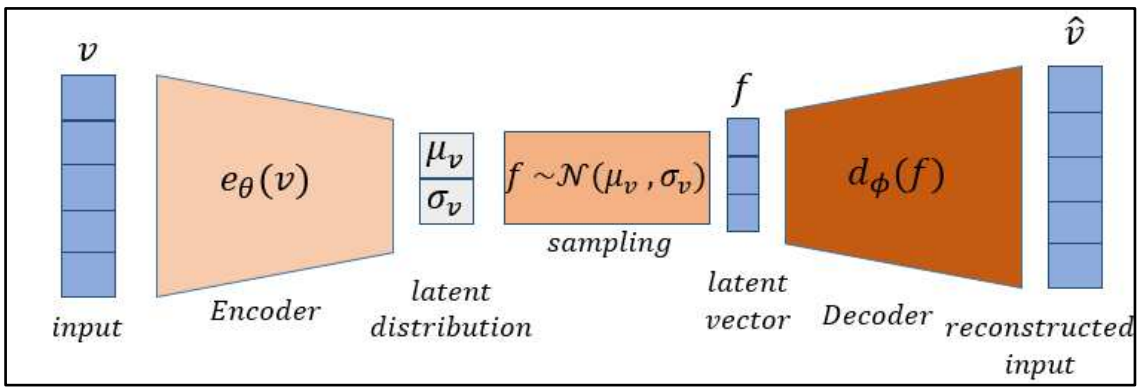

Figure 3. VAE for latent space

$$R - Loss = \| v - \hat{v} \|_2 = \| v - d_\phi(f) \|_2 = \| v - d_\phi(\mu_v + \sigma_v \epsilon) \|_2 \quad (3)$$

Featuring the hybrid attributes derived from several single-classifier VAEs, the multilayer neural network is used to classify multi-class data. **Algorithm 1** shows the steps for the proposed framework.

Algorithm 1 Phishing Website Detection Algorithm

Input: High Dimensional Features of Legitimate and Phishing Websites

Output: Legitimate or Phishing Website

P = Number of Data Points
C_{Out} = Train multiple One Class Classifiers and Produce Output Prediction
FP = False Positives on the $Vali_{Data}$
$Avg.\ FP(C_1) = \frac{\sum_{i=1}^{k} FP(C_1)_k}{|Val\ Data| * k}$
$AdaptiveWeight_{Classifier} = 1 - Avg.\ FP(C_1)$
$W_F = C_{Out} * AdaptiveWeight_{Classifier}$
VAE_F = Prediction from *VAE*
$Total_{FS} = W_F$ U VAE_F
NN = Neural Network on $Total_{FS}$
for *j* in the range *0 to P* **do**
    Out_{NN} = *NN* prediction for P_i
    if $Out_{NN} > Ad_{Thres}$ **then**
        Phishing Website Detected
    else
        Legitimate Website Detected
    end if
end for

2.2. Phishing Website Detector

This is the second level of the proposed approach, which takes the input generated from the first level. A deep neural network is trained with a single output predicting the probability that a website is legitimate. To compute the value of the adaptive threshold, we leverage the K-Fold Cross Validation mechanism. The adaptive threshold finally determines whether the website is legitimate or a phishing website.

3. RESULTS

The experimental results derived from the open-source dataset [18] from the proposed framework are discussed in this section. There are 10000 rows, where 5000 rows belong to a phishing website, and 5000 rows belong to legitimate websites. Each row has 48 features. Below, we discuss the optimization techniques for both levels.

3.1. First Level of the Proposed Framework

Each one-class classifier's hyperparameters are tuned using the K-Fold Cross Validation approach. Below, we discuss different parameter ranges, and the model is trained.

3.1.1 OCSVM

- Nu: The upper bound on the fraction of training errors.
- Kernel: The type of nonlinear function.
- Gamma: The kernel coefficient.
- Tolerance: The value for stopping criterion.
- Max_iter: The total number of iterations for the solver to converge.
- Decision_function_shape: Specifies how the final decision function should be calculated.

Table 1 shows the hyperparameter ranges for OCSVM.

Table 1. OCSVM Parameters

Parameter	Range	Best Hyperparameter
Nu	**[0.1, 0.3]**	**0.1**
Kernel	**[linear, rbf, poly, sigmoid]**	**Poly**
Gamma	**[0.1, 10]**	**10**
Tolerance	**[0.0001, 0.001]**	**0.001**
Max_iter	**[1000, 10000]**	**1000**
Decision_function_shape	**[ovr, ovo]**	**ovr**

3.1.2 Isolation Forest

- n_estimators: The total number of trees in the forest.
- max_samples: The total number of samples for training each tree.
- Contamination: The contamination amount in the data set.
- max_features: The top features to be used for the best split.
- Bootstrap: Whether or not to bootstrap the samples when building trees.

Table 2 shows the hyperparameter ranges for Isolation Forest.

Table 2. Isolation Forest Parameters

Parameter	Range	Best Hyperparameter
n_estimators	**[100, 1000]**	**100**
max_samples	**[0.1, 1.0]**	**1.0**
contamination	**[0.05, 0.25]**	**0.25**
max_features	**[1.0, "auto"]**	**1.0**
bootstrap	**[True, False]**	**True**

3.1.3 Mahalanobis Distance Metric

- store_covariance: The flag indicates whether the covariance matrix (needed for Mahalanobis distance computation) should be stored.

Table 3 shows the hyperparameter ranges for the Mahalanobis Distance Metric.

Table 3. Mahalanobis Distance Metric Parameters

Parameter	Range	Best Hyperparameter
store_covariance	**[True, False]**	**False**

3.1.4 Local Outlier Factor

- n_neighbors: The total number of neighbors to use for k neighbors queries.
- Algorithm: The algorithm for nearest neighbors search.
- Metric: The distance metric for the tree.
- Contamination: The contamination amount in the data set.

Table 4 shows the hyperparameter ranges for the Local Outlier Factor.

Table 4. Local Outlier Factor Parameters

Parameter	Range	Best Hyperparameter
n_neighbors	**[5, 30]**	**30**
algorithm	**[auto, kd_tree]**	**kd_tree**
metric	**[minkowski, euclidean]**	**Minkowski**
contamination	**[0.05, 0.25]**	**0.25**

3.1.5 Elliptical Envelope

- contamination: The proportion of observations that are expected to be outliers.
- Ellipse: The type of ellipse to be used. Possible values include "raw" (which uses the covariance matrix), "outlier" (which uses the robust covariance matrix), or "robust" (which uses the robust covariance matrix but adjusts the threshold based on the contamination rate).

Table 5 shows the hyperparameter ranges for Elliptical Envelope.

Table 5. Elliptical Envelope Parameters

Parameter	Range	Best Hyperparameter
contamination	**[0, 0.5]**	**0.5**
ellipse	**[raw, outlier, robust]**	**Robust**

3.1.6 Minimum Covariant Determinant

- support_fraction: The fraction of the data set that should be used to estimate the covariance matrix.

Table 6 shows the hyperparameter ranges for the Minimum Covariant Determinant.

Table 6. Minimum Covariant Determinant Parameters

Parameter	Range	Best Hyperparameter
support_fraction	**[0.1, 0.5, 1.0]**	**0.5**

Figure 4. Benign Score for One Class Classifier

As shown in **Figure 4**, OCSVM has the highest benign score of 0.97 and thus will be given higher weightage than other classifiers. Next, we discuss the hyper-parameter tuning of the VAE Model.

3.1.7 Variational AutoEncoder

- Learning rate: The rate at which the optimizer updates the model parameters.
- Batch size: The number of samples used in each iteration of the training process.
- The number of epochs: The number of times the model is trained on the entire data set.
- Activation function: The function used to introduce nonlinearity into the model.
- Optimizer: The algorithm used to update the model parameters.

Table 7 shows the hyperparameter ranges for the Variational AutoEncoder.

Table 7. Variational AutoEncoder Parameters

Parameter	Range	Best Hyperparameter
Learning Rate	**[0.001, 0.01]**	**0.001**

BatchSize	**[16, 32, 64, 128, 256]**	**128**
Number of epochs	**[10, 30, 50,70, 80, 100]**	**80**
Activation function	**[sigmoid, tanh, ReLU, and LeakyReLU]**	**LeakyReLU**
Optimizer	**[Adam, SGD, RMSprop, and Adadelta]**	**Adam**

Leaky ReLU is a popular activation function in Variational Autoencoders (VAEs) and other deep-learning models. Its popularity is due to its ability to solve the problem of "dying ReLU" by allowing a slight negative slope for inputs below zero. In a standard ReLU activation function, all inputs below zero are set to zero, leading to the vanishing gradient problem, where the gradients of the parameters become very small or even zero, causing the model to stop learning. One way of addressing this problem is with Leaky ReLU, which permits a slight gradient for negative inputs and can aid learning without stagnation. Also, Leaky ReLU is good because it improves the stability and speed of highly nonlinear deep learning models, particularly during the training phase; it can also help combat overfitting, as a mild negative slope may serve as regularization.

One can configure the batch size hyperparameter in three different ways: Batch Gradient Descent (BGD), Stochastic Gradient Descent (SGD), and Mini-Batch Gradient Descent (MBGD).

BGD is computationally expensive, but it will guarantee that the model parameters are taking an update with the correct gradient. SGD updates the parameters after every single data point. SGD is the fastest of the three, but it can also lead to more inconsistent updates of the parameters, especially if the data is noisy. MBGD is a compromise between BGD and SGD. MBGD divides the training set into smaller "mini-batches," and the parameter weights are updated. MBGD is faster than BGD but still provides some stability compared to SGD. **Figure 5** shows the reconstruction loss for each batch size for VAE, whereas **Figure 6** depicts the loss at each epoch. The final loss achieved is 0.08 for a batch size of 128.

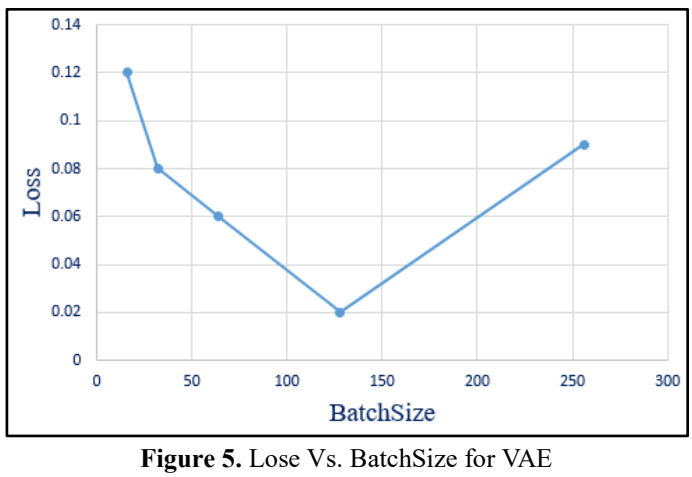

Figure 5. Lose Vs. BatchSize for VAE

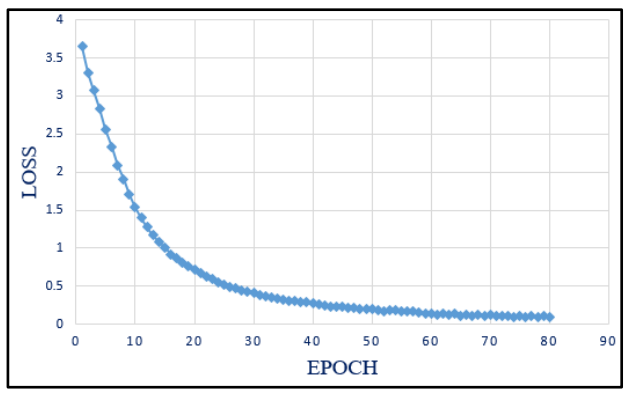

Figure 6. Loss Vs Epoch for VAE

3.2. Second Level of the Proposed Framework

The Phishing Website Detector trained at the second level has the hyperparameters optimized, as shown in **Table 8**. **Figure 7** shows the loss and accuracy value at each epoch. The accuracy achieved is 0.98, with a loss of 0.015. **Figure 8** depicts the loss at each batch size.

Table 8. Website Phishing Detector Parameters

Parameter	Range	Best Hyperparameter
Learning Rate	**[0.001, 0.01]**	**0.001**
BatchSize	**[16, 32, 64, 128, 256]**	**64**
Number of epochs	**[10, 30, 50, 70, 80, 100]**	**100**
Activation function	**[sigmoid, tanh, ReLU, and LeakyReLU]**	**LeakyReLU**
Optimizer	**[Adam, SGD, RMSprop, and Adadelta]**	**Adam**

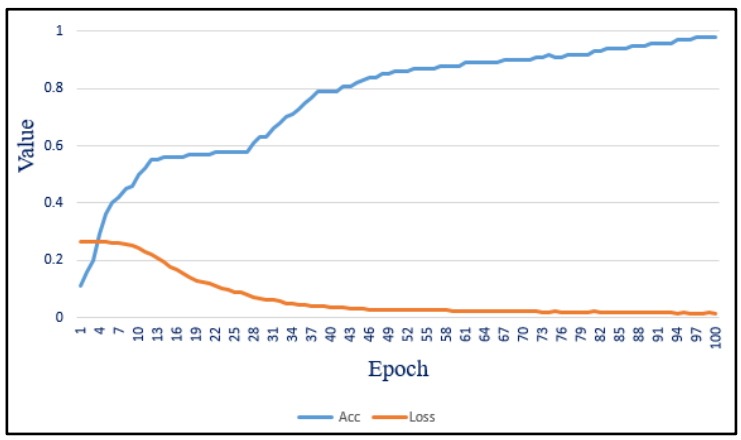

Figure 7. Lose Vs. Epoch for Phishing Website Detector

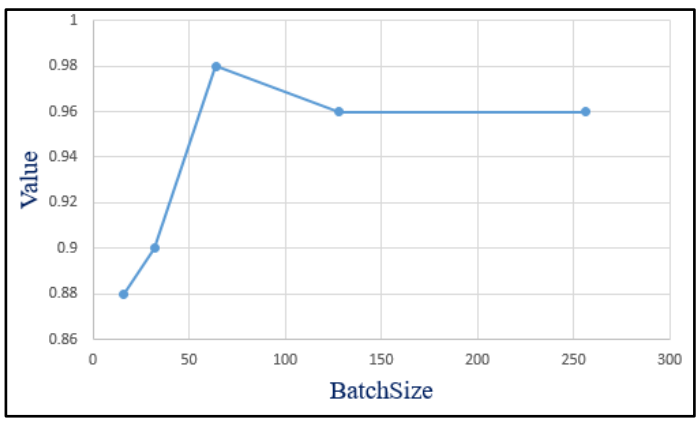

Figure 8. Lose Vs. BatchSize for Phishing Website Detector

Table 9 shows the metrics evaluated in the proposed framework. The following Evaluation Metrics are used:

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

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

Where TP: True Positive, TN: True Negative, FP: False Positive, FN: False Negative

Table 9. Evaluated Metrics on the Proposed Framework

Model	Recall	Precision	F1 Score	Accuracy
Proposed Framework	**0.962**	**0.978**	**0.964**	**0.989**

Table 10. Comparative Analysis

Technique	Accuracy
James et al. [18]	**89.75**
Abdelhamid et al. [19]	**93.5**
Pandey et al. [20]	**94.0**
Mao et al. [21]	**97.31**
Subasi et al. [22]	**97.36**
Proposed Approach	**98.9**

Table 10 lists the accuracy of different approaches. James et al. [19] trained several machine learning classifiers such as Naïve Bayes, Support Vector Machine (SVM), and C4.5 and achieved the highest accuracy of 89.75. Abdelhamid et al. [20] implemented an enhanced dynamic rule induction (eDRI) approach. Their approach sets the two thresholds at different levels and uses only the solid features for training the model. Their model achieved the highest accuracy of 93.5. Pandey et al. [21] implemented a hybrid Random Forest and SVM model and achieved the highest accuracy of 94.0. Mao et al. [22] compare the page's design to distinguish the legitimate from the phishing webpage. They developed the guidelines and built a phishing page classifier consisting of two classifiers: SVM and DT. Their model achieved the highest accuracy of 97.31. Subasi et al. [23] compared Multiboosting and Adaboost algorithms for detecting phishing web pages. The accuracy achieved with the usage of a boosting-based algorithm was 97.61%. Generally, conventional methodologies utilized in this kind of two-class problem build a binary classifier using a tiny subset of features, if any. The performance of our new algorithm reaches 98.9% accuracy compared to the baseline model.

4. DISCUSSION

Phishing is a significant security concern that can result in unauthorized access to systems and networks and loss of sensitive data such as passwords and credit card numbers. Phishing for attacks and soliciting website scams have been a widespread problems, and with artificial intelligence algorithms and machine learning models becoming more innovative every day, it is challenging for any individual to figure out and keep away from such sites. Hence, effectively detecting phishing websites is essential to prevent individuals from engaging in this behavior. Those entities can offer protection to their customers and clients and insure themselves against the loss suffered due to these acts. People can also protect themselves from falling victim to these. To prevent phishing attacks, detecting websites can help organizations and individuals remain safer. Machine learning methods have been used to predict a phishing site's location significantly. Detecting phishing websites has been successful with the use of a one-class classifiers. They aim to distinguish the suspicious from the benign websites by discovering average web profiles and red-flagging deviations. This is followed by verification through VAE, which

involves comparing reconstructed inputs with actuals. As a result, these two techniques give a more reliable and precise phishing detection system.

Since phishing techniques are constantly changing, monitoring and updating the models are essential in enhancing effectiveness. Furthermore, these strategies must complement other security measures, ensuring one is completely protected against phishing attacks. Many currently available systems for detecting phishing websites are based on already existing datasets, which they rely upon when it comes to identifying such sites. However, the real-time detection of new phishing websites is crucial because knowing about them is instrumental in keeping a step ahead with ever-changing tactics used by phishers and attackers alike. Future research could focus on developing real-time phishing website detection systems that can quickly identify and flag new phishing sites as they appear. Many phishing websites use dynamic content, such as pop-up windows and dynamic URLs, to evade detection. Future research could also explore integrating blockchain technology [24] to securely store and verify the authenticity of data to enhance trust and transparency in the process.

Authors' contributions

Made substantial contributions to the conception and design of the study, performed data analysis, machine learning, and deep learning modeling and interpretation: Soni J, Prabakar N;
Performed data acquisition and technical support: Upadhyay H.

Availability of data and materials

"Not applicable."

Financial support and sponsorship

None.

Conflicts of interest

"All authors declared that there are no conflicts of interest.".

Ethical approval and consent to participate

"Not applicable."

REFERENCES

1) Huh, J. H., & Kim, H. (2011). Phishing Detection with Popular Search Engines: Simple and Effective. FPS, 11, 194-207.
2) Raja, A. S., Pradeepa, G., & Arulkumar, N. (2022, May). Mudhr: Malicious URL detection using a heuristic rules-based approach. In AIP Conference Proceedings (Vol. 2393, No. 1, p. 020176). AIP Publishing LLC.
3) Bergholz, A., De Beer, J., Glahn, S., Moens, M. F., Paaß, G., & Strobel, S. (2010). New filtering approaches for phishing email. Journal of computer security, 18(1), 7-35.
4) Tsalis, N., Mylonas, A., & Gritzalis, D. (2016). An intensive analysis of security and privacy browser add-ons. In Risks and Security of Internet and Systems: 10th International Conference, CRiSIS 2015, Mytilene, Lesbos Island, Greece, July 20-22, 2015, Revised Selected Papers 10 (pp. 258-273). Springer International Publishing.
5) Sun, Y., Zhu, S., Zhao, Y., & Sun, P. (2022, October). A User-Friendly Two-Factor Authentication Method against Real-Time Phishing Attacks. In 2022 IEEE Conference on Communications and Network Security (CNS) (pp. 91–99). IEEE.
6) Sirigineedi, S. S., Soni, J., & Upadhyay, H. (2020, March). Learning-based models to detect runtime phishing activities using URLs. In Proceedings of the 2020 4th International Conference on Compute and Data Analysis (pp. 102–106).
7) S. Hossain, D. Sarma, and R. J. Chakma, "Machine learning-based phishing attack detection," Int. J. Adv. Comput. Sci. Appl., vol. 11, no. 9, pp. 378–388, 2020.

8) R. Kiruthiga and D. Akila, "Phishing websites detection using machine learning," Int. J. Recent Technol. Eng., vol. 8, no. 2, pp. 111–114, 2019.
9) A. C. Bahnse, E. C. Bohorquez, S. Villegas, J. Vargas, and F. A. Gonález, ‘‘Classifying phishing URLs using recurrent neural networks,’’ in Proc. IEEE APWG Symp. Electron. Res. (eCrime), Apr. 2017, pp. 1–8.
10) Soni, J., Prabakar, N., & Upadhyay, H. (2019, May). Deep learning approach to detect malicious attacks at system level: poster. In Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks (pp. 314-315).
11) Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008, December). Isolation forest. In 2008, eighth IEEE International Conference on Data Mining (pp. 413–422). IEEE.
12) Cheng, Z., Zou, C., & Dong, J. (2019, September). Outlier detection using isolation forest and local outlier factor. In Proceedings of the conference on research in adaptive and convergent systems (pp. 161-168).
13) McLachlan, G. J. (1999). Mahalanobis distance. Resonance, 4(6), 20-26.
14) Ashrafuzzaman, M., Das, S., Jillepalli, A. A., Chakhchoukh, Y., & Sheldon, F. T. (2020, December). Elliptic envelope-based detection of stealthy false data injection attacks in smart grid control systems. In 2020 IEEE Symposium Series on Computational Intelligence (SSCI) (pp. 1131–1137). IEEE.
15) Rousseeuw, P. J., & Driessen, K. V. (1999). A fast algorithm for the minimum covariance determinant estimator. Technometrics, 41(3), 212-223.
16) Soni, J., Peddoju, S. K., Prabakar, N., & Upadhyay, H. (2021). Comparative analysis of LSTM, one-class SVM, and PCA to monitor real-time malware threats using system call sequences and virtual machine introspection. In International Conference on Communication, Computing and Electronics Systems: Proceedings of ICCCES 2020 (pp. 113-127). Springer Singapore.
17) Peddoju, S. K., Upadhyay, H., Soni, J., & Prabakar, N. (2020). Natural language processing based anomalous system call sequences detection with virtual memory introspection. International Journal of Advanced Computer Science and Applications, 11(5).
18) Tan, Choon L. (2018). "Phishing Dataset for Machine Learning: Feature Evaluation," Mendeley Data, V1, doi: 10.17632/h3cgnj8hft.1
19) James, J., Sandhya, L., & Thomas, C. (2013). Detection of phishing URLs using machine learning techniques. In 2013 International Conference on Control Communication and Computing (ICCC) (pp. 304–309). IEEE.
20) Abdelhamid, N., Thabtah, F., Abdel-jaber, H. (2017). Phishing detection: A recent intelligent machine learning comparison based on models content and features. In 2017 IEEE International Conference on Intelligence and Security Informatics (ISI) (pp. 72–77). IEEE.
21) Pandey, A., Gill, N., Nadendla, K. S. P., & Thaseen, I. S. (2018). Identification of phishing attacks in websites using random forestsvm hybrid model. In International conference on intelligent systems design and applications (pp. 120–128). Springer.
22) Mao, J., Bian, J., Tian, W., Zhu, S., Wei, T., Li, A., et al. (2018). Detecting phishing websites via aggregation analysis of page layouts. Procedia Computer Science, 129, 224–230.
23) Subasi, A., Molah, E., Almkallawi, F., & Chaudhery, T. J. (2017). Intelligent phishing website detection using random forest classifier. In 2017 International conference on electrical and computing technologies and applications (ICECTA) (pp. 1–5). IEEE.
24) Zhao, W., Aldyaflah, I. M., Gangwani, P., Joshi, S., Upadhyay, H., & Lagos, L. (2023). A blockchain-facilitated secure sensing data processing and logging system. IEEE Access, 11, 21712-21728.