An Enhanced Anti-Phishing Technique for Social Media Users: A Multilayer Q-Learning Approach

Asif Irshad Khan[1], Bhuvan Unhelkar[2]
Computer Science Department-Faculty of Computing and Information Technology,
King Abdulaziz University, Jeddah 21589, Saudi Arabi[1]
Muma School of Business, University of South Florida, Sarasota-Manatee Campus,
Sarasota, FL 33620, USA[2]

***Abstract*—As social media usage grows in popularity, so does the risk of encountering malicious Uniform Resource Locator (URLs). Determining the authenticity of a URL can be a highly challenging task, primarily due to the sophisticated attack structure employed by phishing attempts. Phishing exploits the vulnerabilities of computer users, making it difficult to discern between genuine and fraudulent URLs. To address this issue, a self-learning AI framework is required to warn social media users of potentially dangerous links. While several anti-phishing techniques exist, including blacklists, heuristics, and machine learning-based techniques, there is still a need for improvement in terms of detection accuracy. Hence, this study proposed a novel approach to combat phishing attacks using artificial neural networks, and the main aim is to create and validate the anti-phishing technique tool for detection accuracy. Initially, the URL data is collected, followed by preprocessing and then the analysis for malicious activity using the Logistic Bayesian Long Short-Term Memory model (LB-LSTM). The observed malicious URL features are extracted using multilayer Q-learning with the CaspNet and swarm optimization models. Analysis of these features enables the identification of a malicious URL, which is then removed, and the social media user is warned. The proposed technique attained a detection accuracy of 94.33%, Area under the ROC Curve (AUC) of 98.71%, Mean Squared Error (MSE) of 5.67%, Mean average precision of 88.67%, Recall of 98.67%, and F-1 score of 94.34%.**

***Keywords*—Multilayer Q-learning; anti-phishing model; social media users; machine learning; optimization; URLs; logistic Bayesian LSTM model**

I. INTRODUCTION

Phishing is a deceitful online practice that employs social engineering and a particular strategy to deceive individuals using the internet to obtain their personal information or critical online data [1]. Businesses across various sizes and industries increasingly recognize the imperative of investing in anti-phishing solutions to protect their most valuable asset. Phishing is a fraudulent technique used to acquire sensitive customer information, such as classified data, via deceptive emails, counterfeit websites, dubious internet-based advertisements or promotions using forged Short Message Service (SMS) messages purporting to be from reputable service providers or online organizations, and other similar methods. Recent studies have shown that individuals with a sociable and trustworthy personality are more susceptible to phishing schemes, mainly when actively participating in various social media platforms. A single instance of attack occurred on the social media platform Facebook, which enticed individuals to visit fraudulent websites designed to mimic the Facebook login page. The dissemination of the attack afterward extended to Facebook users via the promotion of acquaintances to access the hyperlink present on the original user's profile [2]. As an example, the security reports of Details and Patterns in 2017 revealed that a substantial sum of about $5 billion was stolen during the period spanning from October 2013 to December 2016. This financial loss was attributed to a W-2 phishing attempt, which affected a global population of over 24,000 individuals. W-2 phishing emails have recently been identified as one of the most dangerous kinds of phishing email scams, primarily due to their propensity to facilitate fraudulent tax filings and refund claims. Specialized deception encompasses using harmful code or crime ware, often installed on a personal computer or laptop, without the user's awareness [3]. Phishing may take several forms, including DNS poisoning, keyloggers, capturing meetings, damaging records, injecting information, etc. A new kind of malicious software called "ransomware" has emerged in recent years, allowing cybercriminals to run malicious code on a client's assets, locking them and demanding a payment "ransom" to unlock them. According to the CSO's announcement, 93% of all phishing emails nowadays are "ransomware." [4]. According to this study [5], the vast majority of victims of such crimes pay ransom demands quickly. Using Artificial Intelligence (AI) to glean information from customer files, postings, or social media and provide a timely warning is essential for fighting the phishing scourge. Though AI has promise, most current implementations need extensive client oversight, costly resources, and considerable management effort to be successful. In contrast to traditional detection and warning methods, current deep-learning systems are faster and more effective and don't need client mediation. The recent trend in Deep Learning (DL) based research has focused on extracting key highlights from text rather than conventional facts. The ability of deep-learning algorithms to effectively identify and predict concealed patterns within textual data is the primary reason why a traditional approach may struggle to detect or anticipate such patterns [6]. The highlights of this study are as follows:

- An innovative approach to mitigate phishing attacks targeting social media users through AI techniques.

- The development of a multilayer Q-learning model, in combination with CaspNet (Mul_Q_capsnet), and swarm optimization for the extraction of URL features.
- Utilizes a Logistic Bayesian LSTM model (LB-LSTM) to detect malicious behavior within social networking URLs.
- Provides warnings to social media users regarding potentially dangerous URLs.

Following is an outline of this paper. The related research is described in Section II, Section III outlines the materials and methods, Section IV describes performance analysis, Section V highlights the results and discussion of this research, and finally, final thoughts on the research work, like conclusion and future work, are listed in Section VI.

II. Related Works

In today's environment, phishing poses the Internet with the most prominent challenge it must overcome. Many researchers have worked to create services that protect users from cyberattacks by detecting and blocking phished URLs using artificial intelligence (AI) and deep learning (DL) techniques [7]. Previous studies have developed and implemented two types of phishing detection systems: list-based and AI-based phishing identification systems. The list-based study identified many factors contributing to users' susceptibility to phishing attacks, including the lack of personal computer (PC) knowledge, inadequate understanding of security indicators, susceptibility to visual deception, and limited attention. Also, study in [8] investigated the persistence of successful phishing attacks despite ongoing efforts to mitigate associated risks. The findings of their investigations indicated that even when trained to identify phishing attacks, people were vulnerable at a rate of 53%. A study in [9] indicates that client PC data, client orientation, and the client's educational level are among the primary factors influencing whether clients open phishing emails.

A study in [10] introduced a system that generates a whitelist by logging the IP addresses of websites containing a login interface that a user visits. The system issues a warning if there is any inconsistency in the recorded website information when a user accesses a site. However, this approach raises concerns regarding websites users visit for the first time. In response to this challenge, work in Reference [11] has proposed a strategy to alert users on the web by maintaining an up-to-date whitelist of reputable sites. This strategy consists of two key components: a domain-IP address-matching module and extracting link attributes from the source code. This approach is further discussed in Reference [12]. A collaborative learning approach was employed for detecting phishing attacks in emails. Substituted selection methods were used to eliminate features unrelated to accuracy, achieving nearly 100% accuracy with just 11 selected features.

Study in [13] employed the Phi DMA approach, which used five layers: URL highlight layers, lexical layer, and whitelist layer, and accomplished an accuracy of 92%. In another review [14], the examination of phishing was identified through SVM. Author in [15] proposed an outrageous learning machine, a regulated AI calculation to determine spam accounts in SinaWeibo, Chinese miniature writing for a blogging site. Alberto et al. proposed an internet-based framework to filter comments posted on YouTube [16].

In study [17], the authors presented a structure for discovering dubious conversations on web-based gatherings using a coordinated help vector machine and particle swarm optimization methodology. The study by [18] focuses on detecting harmful URLs inside web-based social networks using client behavior analysis. The authors propose a research framework that investigates and detects social spam. This framework incorporates characteristics from URLs and online social networks (OSNs), emphasizing user profiles, postings, and URL attributes. The objective is to improve the accuracy of identifying harmful activities. A confirmation of the concept enhancer method was developed in [19], effectively used for the identification of bots, and in [20] identified spam in SMPs and involved the value of features in emphasizing a higher result collection of regulations. AI techniques require an environmental input to be adjusted and moved along.

The authors in [21] highlighted that the current apps, services, and systems are at risk of cyber-attacks like malware and software piracy because of the always-on nature of the Internet. These dangers threaten not just confidentiality but also safety. Malicious software like computer viruses, ransomware, scareware, and Trojan horses, as well as more traditional forms of software piracy like hard-disk loading, client-server overuse, and internet piracy, have the potential to wipe out critical data, resulting in reputational and economic damage. Reference [22] suggests companies can comprehensively enhance their operations by emphasizing roles, processes, individual actions, business strategies, business process modeling, quality assurance, cybersecurity, accountability, and big data.

In research [23], the author used a Neural Network (NN) to examine the blunder level of 4000 bogus and 4000 genuine pictures. With a solid achievement rate, a certified neural network has figured out how to group images as misleading or valid. Feature extraction extracted 17 features from 2500 phishing URLs from the PhishTank archive [24] and divided them into address bar-based highlights, unusual-based elements, and HTML and JavaScript-based highlights. Most parts were automatically separated from the URL and the page's source code without depending on third-party services. However, the WHOIS extracted the domain's age and DNS record [25]. The Alexa database retrieved the page's ranking [26]. Concurrently, the authors outlined an IF-ELSE rule and assigned a weight to each element. The weight of a feature was established by calculating the feature value concerning the total number of phishing links. Each segment's value could be either 1, 0, or 1, representing legitimate, suspicious, or phishing [27].

III. Materials and Methods

This section discusses novel techniques in the anti-phishing model using machine learning techniques for social media users and network optimization. The proposed architecture is shown in Fig. 1.

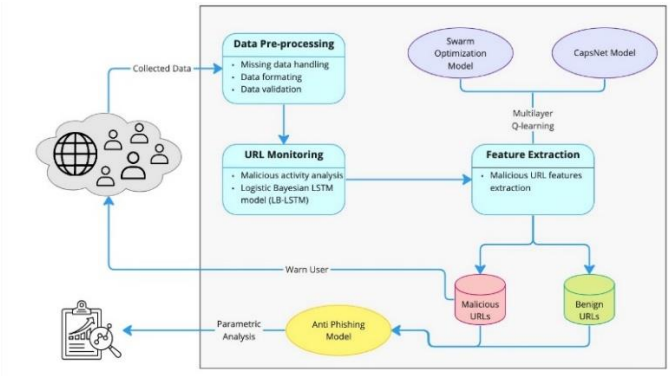


Fig. 1. Proposed Self-Learning framework for detecting malicious URLs.

A. Data Pre-processing

The acquired data must undergo several preprocessing steps before entering each classifier to ensure that the algorithm interprets the information correctly and selects the best approach. One of these preprocessing activities involves organizing and cleaning the data. Formatting is crucial for presenting data in a format that classifiers can understand, such as converting data types into a text file or a tabular form. The cleaning process addresses missing values in a dataset, such as missing names or values in specific data fields. It involves setting properties determined manually or by the majority vote for matching values in different instances and even removing specific examples that could negatively impact the classifier's learning process. Additionally, cleaning details refer to the removal of personal information that could compromise the privacy of specific individuals.

B. Tokenization

Breaking down text into its constituent parts, which can be words, sentences, or even individual characters, is known as tokenization, and the individual units are referred to as tokens. The objective here is to analyze text as a single unit. The list of tokens then becomes an interpretive response or serves as input for further sentence-based analysis. In languages where sentences are divided into segments and computer science, tokenization is crucial for reading text. Proper text tokenization is typically essential at the beginning of any text analysis process. All recognized text analysis methods rely on terms extracted from the dataset. To achieve this, a processor needs to tokenize the data. This can be straightforward when the text is in computer-readable formats. However, specific challenges may arise, such as handling punctuation marks. Other characters, such as parentheses, hyphens, and so on, also need to be managed.

C. Logistic Bayesian LSTM Model (LB-LSTM)- based Malicious Activity Analysis

In logistic regression, the dependent variable typically takes on a binary form, which means it has only two possible values, like 0 or 1, true or false, or yes or no. This characteristic makes logistic regression well-suited for predicting the probability of an event belonging to one of two categories: success or failure. In this scenario, a sigmoid function is commonly employed to describe the connection between the predictor variables and the likelihood of the event happening. The sigmoid function yields output values ranging from 0 to 1, effectively representing probabilities.

Consider a prototypical example with two predictors, A1 and B2. These predictors can be either constant values or binary variables, taking values of 0 or 1. The conversion likelihood W(A B) may be further divided into the acceptance probability A(A B) and the trial proposition probability T(A B), resulting in the equation W(A B) = T(A B) • A(A B). The likelihood of moving from state A to state B and the likelihood of moving in the other direction are related via Eq. (1):

$$P_A \cdot T(A \to B) \cdot A(A \to B) = P_B \cdot T(B \to A) \cdot A(B \to A) \quad (1)$$

For the likelihood for sample structure A to be equivalent to the Boltzmann weight by Eq. (2), the test plan and recognition likelihoods must be carefully selected.

$$P_A = \frac{e^{-\beta E_A}}{Z} \quad (2)$$

where, EA represents structure A's energy, since the conversion likelihoods are available using the proportion of probabilities, data for divider constant Z is unnecessary. Using Eq. (2), the detailed balancing requirement (3 may be rewritten as follows:

$$\frac{T(B\to A)\cdot A(B\to A)}{T(A\to B)\cdot A(A\to B)} = \frac{P_A}{P_B} = e^{-\beta(E_A - E_B)} \quad (3)$$

a process where all pairings of the states A, B satisfy the constraint T(A B) = T(B A). In the case of a design with N spins, for instance, it parallels picking one spin randomly from the matrix,: T(A B) = T(B A) = 1/N. Quickest if A(A B) or A(B A) is equivalent to 1, or if the larger of the two acceptance probabilities. 1 Padd is the chance of not adding an aligned spin. The complete balancing requirement may be expressed as Eq. (4).

$$\frac{T(A\to B)\cdot A(A\to B)}{T(B\to A)\cdot A(B\to A)} = \left(1 - P_{\text{add}}\right)^{m-n} \frac{A(A\to B)}{A(B\to A)} = e^{-\beta(E_B - E_A)} \quad (4)$$

Noticing that EA − EB = 2J (n − m), it follows that by Eq. (5):

$$\frac{A(A\to B)}{A(B\to A)} = \left[\left(1 - P_{\text{add}}\right)e^{2\beta J}\right]^{n-m} \quad (5)$$

Due to the constraint that a Bayesian network connected to Gi can have a maximum in-degree equal to n classes, there may be conditional probability tables with an exponential number of items in the nth category. However, this becomes impractical when dealing with issues that involve multiple types. Therefore, to address this challenge, it is necessary to reduce the maximum in-degree through the application of a structural learning technique. In real-world applications, we evaluate the Gi optimization as follows in order to shorten the arcs connecting classes to features arcs: $\mathcal{G}_i^* = \arg\max_{G,CGC} \log P(\mathcal{G} \mid \mathcal{D})$, where it's important to consider set inclusions among graphs in the arcs space, as specified by Eq. (6).

$$\log P(\mathcal{G} \mid \mathcal{D}) = \sum_{i=1}^{n} \psi_a[C_i, \text{Pa}(C_i)] + \sum_{j=1}^{m} \psi_a\left[F_j, \text{Pa}\left(F_j\right)\right] \quad (6)$$

Pa(Fj) represents Fj's paternities consistent with G, and Pa(Ci) defines Ci's parentages. Moreover, ψα is BDEu score

with equivalent model size α. For illustration, the score ψα[Fj ,Pa(Fj)] is$\sum_{i=1}^{|\mathrm{Pa}(F_j)|}\left[\log\frac{\Gamma(\alpha_j)}{\Gamma(\alpha_j+N_{ji})}+\sum_{k=1}^{|F_j|}\log\frac{\Gamma(\alpha_{ji}+N_{jik})}{\Gamma(\alpha_{ji})}\right]$.

Finally, while j = P i ji, ji is equal to divided by the sum of the (joint) states of the parents of Fj and the number of states of Fj. All class variables share the same parents across all the graphs within the search space when considering a network associated with a specific class event, as the connections linking the class events are predetermined. This implies that the initial sum in Eq. (4) remains constant on the right side. Consequently, by concentrating solely on the attributes, we can achieve the optimization described in Eq. (3). A feature's parent set may be selected from any subset of C, which simplifies the problem into m distinct local optimizations. In reality, Eq. (7) by G establishes who Fj's parents are.

$$Bold\mathbf{C}_{F_j}=\arg_{\mathrm{Pa}(F_j)\subseteq \mathrm{C}}\max_a\left[F_j,\mathrm{Pa}(F_j)\right] \tag{7}$$

For each time j = 1, m. This is made practical by bipartite partition of class occasions as well as elements, yet much of the time, coordinated cycles might be found in the chart that expands every neighborhood score. Let k be the quantity of blend parts (i.e., the quantity of C qualities), with X being the arrangement of inquiry factors, Z being different factors. The marginal distribution of X may be calculated using Eq. (8) by adding together C and Z:

$$\begin{aligned}P(X=x)&=\sum_{c=1}^{k}\sum_z P(C=c,X=x,Z=z)\\&=\sum_{i=1}^{k}\sum_z P(c)\prod_{i=1}^{\prod}P(x_i\mid c)\prod_{j=1}^{H}P(z,\mid c)\\&=\sum_{c=1}^{k}P(c)\prod_{i=1}^{!1}P(x_i\mid c)\prod_{j=1}^{4}\sum_{z_j}P(z_j\mid c)\\&=\sum_{c=1}^{k}P(c)\prod_{i=1}^{M}P(x_i\mid c)\end{aligned} \tag{8}$$

where past equality is valid since, for any j, j 1 zj P c z like this, it is easy to dismiss non-inquiry factors Z while figuring P(X = x), and calculation of P(X = x) takes O(|X|k), paying little heed to |Z|. Bayesian organization inference, conversely, is the most pessimistic scenario dramatic in |Z|. Restrictive probabilities, successfully assessed as proportions of peripheral probabilities, should likewise be considered. P(X=x,Y=y) = P(X=x,Y=y)/(Y=y) The combination of trees, where every variable in each bunch is permitted to have one extra parent notwithstanding C, gives somewhat more extravagant model than naïve Bayes while as yet taking into consideration productive inference.

The computation involved is about the probability $(P_1^-,P_2\ldots P_n)$ when $X(x_1,x_2\ldots x_n)$ belongs to $C_1,C_2\ldots C_n$, with $P_jX(x_1,x_2\ldots x_n)$ being the probability when $X(x_1,x_2\ldots x_n)$ belongs to C_j. Then max $(P_1,P_2\ldots P_n)$ is demanded result.

$$P\left(C_j\mid x_1,x_2\ldots,x_n\right)=P(x1,x2,\ldots,x_n\mid C)P\left(C_j\right) \tag{9}$$

According to this formula, P(Cj) is the posterior probability that Cj includes text vector (x1,x2... xn) when the text to be categorized belongs to Cj and P(x1,x2... xn|Cj) is the prior probability when text belongs to Cj. As a result, max (P1,P2,...,Pn) is the highest value possible for the following Eq. (10):

$$\arg\max_{x_t}P\left(C_j\mid x_1,x_2\ldots,x_n\right) \tag{10}$$

The qualities (x1,x2,...,xn) are assumed by Bayes to be independent of one another. The product of the probabilities for each attribute is then the joint probability. Thus, Eq. (11) is the final classification function.

$$\arg\max_{x_t}P(x1,x2,\ldots,x_n\mid C)P\left(C_j\right) \tag{11}$$

In this formula, $\frac{N(X_i=x_i,C=C_j)+1}{N(C=C_i)+M}$ is amount of text in training set that belongs to Cj, and N is total amount of text in training set. $P\left(x_i\mid C_j\right)=\frac{N(X_i=x_i,C=C_j)+1}{N(C=C_i)+M}N(X_i=x_iC=C)$ is the amount of text that has Cj's attribute xi, N(C=Cj) is amount of text that belongs to Cj, and M is text vector's dimension. Naive Bayes classifier's core design process, where two areas have been enhanced, is described above.

Typically, when presented with a microarray picture, we are unsure of which category it belongs to, hence the fairness principle demands that the text obtain the same prior probability for each group. Since there are differing amounts of text types in training sets, treating the prior probability differently is unjust and illogical. As a result, it makes sense to evaluate prior probability computation and apply the same prior probability instead. The classification function Eq. (12) shown below can therefore be obtained:

$$C_j\in Ci=1\prod_{i=1}^{n}P\left(x_i\mid C_j\right) \tag{12}$$

Omitting the computation of prior probability can significantly accelerate the calculation, but it does not affect the final grouping result because the maximum probability is required. The proposed model uses a two-valued constant to store the subsequent likelihood. Sometimes, the sentence that needs to be categorized is lengthy, which increases the sentence's dimension vector and decreases the subsequent likelihood. Additionally, reproducing the probabilities of all potentials may lead to inaccurate transmission. This can be corrected by reproducing the subsequent likelihood of each feature property. This will not affect the experiment's findings since what matters in the end is comparing probabilities between categories, and multiplying probabilities is logical. Initially, we expanded ten times, but throughout the research, we found that the subsequent likelihood would occasionally fall outside the range of the two-valued constant, significantly impacting the experiment's outcomes. We added an enlargement feature K to the function to reduce the impact of inaccurate propagation. The following optimization Eq. 13 and Eq. 14 are optimized to evaluate the load and bias values.

$$\min_{w,b,\xi}\frac{1}{2}ww^T+C\sum_{i=1}^{n}\xi_i \tag{13}$$

$$\text{subject to}\begin{cases}y_i(w^T\phi(x_i)+b)\geq 1-\xi_i\\ \xi_i\geq 0,i=1,\ldots.,n\end{cases} \tag{14}$$

$\phi(x_i)$ – other dimension data points have been transferred. ξi- represent slack variable and these variables direct observations in the direction of the margin. Regularization is defined by C.

In this study, LSTM networks are utilized to create base models. Fig. 2 displays the construction of our base learner models using LSTM. Our deep LSTM network consists of multiple hidden layers, with a sequential input layer and two LSTM hidden layers added between the input and output layers. Each hidden layer consists of multiple memory cells and fully connected dense layers. We incorporate multiple LSTM and dense layers to construct a more precise, robust, expressive deep network for URL classification. The LSTM strategy takes URLs as input without requiring manual element extraction.

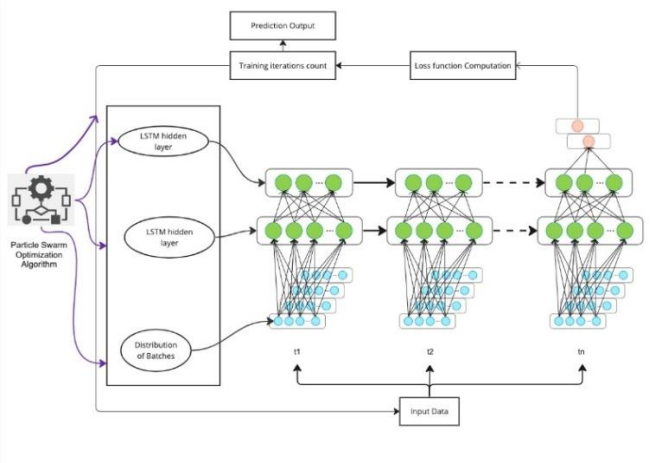


Fig. 2. Structure of the proposed LSTM model of the system.

The LSTM algorithm automatically processes the sequence of characters found in the URL. URLs are initially subjected to a tokenization process, where each unique character in the sequence is assigned a specific number, thereby converting the URL into tokens. These resulting token lists serve as inputs to the LSTM base models. The output layer makes predictions regarding the final outcome, specifically whether the URL is associated with phishing. We conduct experiments to determine the optimal number of LSTM units, the number of neurons in each dense layer, and the number of dense layers in each network. The models are fine-tuned by varying the number of epochs.

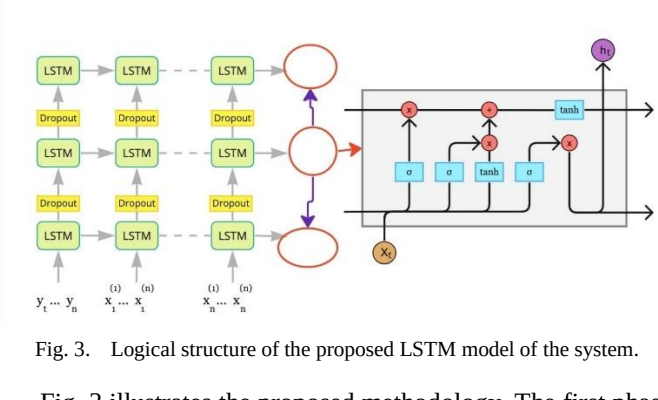


Fig. 3. Logical structure of the proposed LSTM model of the system.

Fig. 3 illustrates the proposed methodology. The first phase involves creating and training n LSTM models, each using a distinct subset of the training data. In the subsequent stage, a collection of records is supplied as input to each model for testing purposes. The models are tasked with predicting the classification of each respective test record. The proposed ensemble LSTM model utilizes a voting approach for generating predictions. Essentially, the class assigned by the ensemble model is based on the majority of votes received from the individual models for a given test record.

For instance, if five LSTM models trained on distinct subsets of training data are provided with a particular test record, and four of them predict the record as "phishing" while the remaining one predicts it as "legitimate", the ensemble method would predict it as "phishing" using the voting approach since it was the majority prediction among the individual models. Algorithm of the proposed model is given below.

Algorithm 1: Phishing Detection with LB-STM, CaspNet, and Swarm Optimization

1: Input: Social media URL data
2: Output: Warning for potentially dangerous links
3: Initialize:
- URL data collection module.
- LB-LSTM model for malicious activity analysis.
- Multilayer Q-learning with CaspNet and swarm optimization models for feature extraction.
- Anti-phishing tool for social media users.

4: Main Loop (URL Analysis):
5: for all URL ∈ SocialMediaURLData **do**
6: Preprocess URL data:
preprocessed data ← Preprocess(URL)
7: Extract features using multilayer Q-learning, CaspNet, and swarm optimization:
q features ←
MultilayerQlearning(preprocessed_data)
caspNet features ←
CaspNet(preprocessed_data)
swarmOpt_features ←
SwarmOptimization(preprocessed_data)
8: Combine extracted features:
Combined features ←
[q_features, caspNet_features,
swarmOpt_features]
9: Analyze malicious activity using LB-LSTM model:
maliciousness ←
LB-LSTM(combined features)
10: if maliciousness > Threshold **then**
11: Remove the URL and warn the social media user
12: end if
13: end for

D. Swarm Network Optimization

The Particle Swarm Optimization (PSO) is a system that utilizes particles traveling at a certain speed in a swarm or population to explore potential solutions and address each competitor arrangement. The system is made up of two stages: the preparation stage and the location stage. During the preparation stage, various legitimate and phishing websites are utilized to create and construct a sophisticated recognition model. Initially, 11055 websites are divided into twelve distinct categories based on the characteristics derived from the website's address bar. Additionally, six categories are established based on anomalies, five categories pertaining to HTML, and classifications dependent upon JavaScript.

Furthermore, the website's domain is used to classify seven distinct groups.

To determine the weightings to apply to specific website qualities, the suggested technique employs Particle Swarm Optimization (PSO). This practice is used due to the unique importance and varied contributions of each feature in identifying phishing websites. The PSO method accurately identifies phishing websites by computing the weighted sum of webpage properties. This practice guarantees that essential information is included in the artificial intelligence computations. In order to optimize the efficacy of artificial intelligence (AI) models, the technique of component biasing is used. This methodology assigns diminished weights to traits of lesser influence while attributing more significance to pivotal features. This stands in contrast to component selection techniques that completely neglect less relevant features.

It is anticipated that component weights will be represented as real numbers falling within the [0, 1] range. In accordance with the Particle Swarm Optimization (PSO) algorithm's recommended feature weighting, these values will indicate the relative importance of website characteristics. The number of features to which weights are assigned in the PSO is determined by the dimensionality of each particle.

Consider a scenario where 'n' feature weights are encoded in PSO particles according to the suggested PSO-based feature weighting method. In this setting, the 'nth' component and any supplementary features have a disproportionate weight, whereas the principal feature has less sway. However, the 'third' and 'n-1st' properties are deemed unnecessary when developing reliable forecasting models. A novel method for feature weighting, employing Particle Swarm Optimization (PSO), has been introduced. PSO entails a swarm of particles exploring diverse feature weight configurations. Each particle moves with random speeds and angles, characterized by a unique position representing the assigned weights for various features. These weights are encoded as real values within the range of zero to one.

Once the initial particle swarm is established, each particle's fitness is assessed based on the accuracy of feature characterization. PSO evaluates the well-being of each particle by initially determining the characterization accuracy and utilizing the weighted attributes of the particle for constructing the AI model from the training data. The objective of the PSO's health evaluation is to identify the best position for each element (pbest) as well as the best position for the entire swarm (gbest). If the values of interest surpass the individual health benefits of previous pbest and gbest, the current pbest and gbest will be updated. Each particle then appropriately adjusts its speed and position according to the updated pbest and gbest. This process is iterated until PSO optimizes the most prominent features. Ultimately, PSO yields the gbest, which contains some of the best feature weights achievable within the swarm.

Next, six commonly used AI algorithms are generated using a training dataset with features weighted through PSO's feature weighting process. The dataset, enriched with components weighted by PSO, is harnessed to construct various machine learning models, including Backpropagation Neural Networks, Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), Decision Trees (DT), Random Forests (RF), and Naïve Bayes classifiers (NB). These pre-trained models are developed and stored according to the specifications necessary for the effortless identification of new phishing websites.

A thoroughly evaluation is conducted using a recently collected dataset to gauge the effectiveness of the developed methods for detecting phishing websites. The primary objective is to determine the most vital components within the testing dataset. These elements include properties related to HTML and JavaScript, characteristics of the address bar, attributes based on geographical regions, and features linked to anomalies. The rigorous method of extracting features is crucial in developing accurate website models.

Subsequently, weights are applied to the characteristics in the testing dataset based on the optimum values generated from the PSO algorithm during the preparation step. This critical step significantly improves the accuracy of phishing website detection. PSO-weighted characteristics are added as input variables into the phishing site detection models created in the preliminary phase when considered necessary. These models are then used to determine if a website fits the requirements for being classified as a phishing site.

In conclusion, the performance of phishing site detection models incorporating proposed PSO-based feature weighting is evaluated and compared to standalone models.

IV. Performance Analysis

The main goal of this research is to evaluate how effectively organizations can combat phishing attacks by utilizing artificial neural networks. Our objective is to design a machine-learning-based anti-phishing technique that can be easily integrated into social media platforms. To achieve this, we gather social media URLs and analyze them using the logistic Bayesian LSTM model (LB-LSTM) to identify any malicious activity. We use a combination of multilayer Q-Q learning with the CaspNet (Mul_Q_capsnet) and swarm optimization models to extract features that are indicative of malicious URLs. Any URLs that exhibit these features are then flagged and removed to ensure the safety of users.

Testing the proposed framework presents a significant challenge, as a globally recognized dataset is not readily available. To tackle this issue, we have undertaken the task of generating our own dataset. The dataset included in this study consists of 73,575 URLs, selected based on their considerable size and lack of device restrictions. The dataset contains 37,175 URLs classified as phishing and 36,400 URLs classified as legal. In the experimental configuration, we used a random assignment strategy to allocate 75% of the dataset for training purposes, while the remaining 25% was put aside exclusively for classification testing. To conduct a thorough assessment of the efficacy of our methodology, we used a range of measures, including accuracy, recall, and F-measure. It is important to note that each classification Algorithm was developed and tested independently for user-based features. In the same way, each classifier is trained and tested separately for content-based characteristics.

A. Dataset

As previously discussed, in our efforts to obtain a reliable dataset for assessing the proposed framework, we faced challenges in finding one that fulfilled our standards. Thus, it was crucial to create a strong and extensive dataset. To accomplish this, we concentrated on gathering two distinct sets of URLs - legitimate and phishing - to ensure a well-rounded and thorough dataset. For the phishing URLs used in this research, our primary source was PhishTank (2018). However, it's worth noting that PhishTank does not provide a free dataset. As a result, we implemented a script to systematically acquire a substantial volume of rogue website addresses. At the same time, we collected authentic websites. The Yandex Search API proved to be invaluable in this regard (YandexXMLYandex Innovations, 2013).

To obtain a collection of web pages with minimal risk of phishing, we first created a specific query_word_list and then utilized the Yandex Search API to retrieve the top-ranked pages. This method was chosen due to the transient nature of malicious URLs, resulting in lower rankings by search engines. Our efforts resulted in a comprehensive dataset of 73,575 URLs, now available online for fellow researchers. This dataset includes 37,175 phishing URLs and 36,400 legitimate URLs, ensuring a well-rounded evaluation.

B. Feature Analysis Based on NLP (Natural Language Processing)

By utilizing information preprocessing as detailed in previous sections, separating a few unmistakable features can be made simple. These features are extracted using Natural Language Processing (NLP) techniques that rely on the English language. For optimal efficiency, features are currently extracted based on the English language but can easily be adapted to any language. The selection and design of these features are minor issues, and most works focused on phishing detection use various feature lists based on their algorithms. Our chosen feature list primarily includes the need to parametrize the URL of the webpage, meaning that the text-based web address should be broken down into the words it contains. However, this process is not a simple task as a web address can contain several concatenated texts, making it difficult to find each word. To address this, the URL is parsed, and certain unique characters are considered, such as ("?", "/", ".", "=", and "and"), to rot the text.

The last character is particularly favored by attackers to convince victims that it is a legitimate webpage. Attackers use various tactics to deceive users, including using publicly-known brand names like Apple, Google, Gmail, Oracle, or specific keywords such as login, secure, account, server, etc., depending on the type of attack or targeted website. Therefore, in addition to the features proposed in previous literature, we defined several additional elements for detecting phishing websites. Although this number is not excessive, it is necessary to apply a feature reduction tool when using NLP, either alone or in combination with other techniques.

C. Word Vectors

Converting words into vectors is a popular approach for identifying key features, such as text handling or text mining techniques. Our system connects with the URL of a webpage, which is essentially a message composed of many words. Rather than manually modifying these words, a programmed vectorization technique is preferred. To accomplish this, we use the "StringtoWordVector" feature of Weka to convert each URL into a word vector. Once the linked vectors are obtained, the selected AI algorithm can easily utilize them. In the suggested framework, we tested 73,575 URLs and extracted 1,701 word highlights during the vectorization process.

To reduce the number of highlights, we utilized a component reduction system that employs the "CfsSubsetEval" method - an algorithm for element determination that utilizes the best first pursue technique. This lowering tool has reduced the number of necessary highlights from 1,701 to 102 in the rundown.

V. Results and Discussion

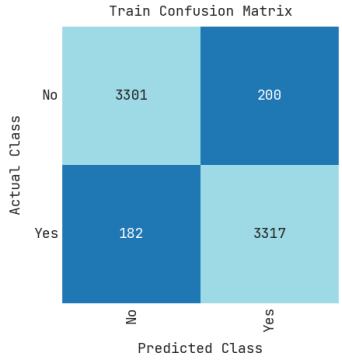
A. Results

Finding a widely recognized dataset was one of the most difficult challenges we encountered when evaluating our suggested methodology. We couldn't find one, so we made our own. This dataset has 73,575 URLs and was selected owing to its massive size and absence of a test gadget restriction. The collection contains 37,175 phishing URLs and 36,400 trustworthy URLs. Our tests were conducted on a MacBook Pro with a 2.7 GHz Intel Core i5 CPU and 8 GB of 1867 MHz DDR3 RAM. We used Weka and numerous readymade libraries to test our system. We used 10-fold Cross-Validation and the default limit possible gains of all computations during the testing. In addition, each test set was run using seven distinct simulated intelligence algorithms. We created a confusion matrix for the tested learning algorithms and eventually found the optimal test type, whether NLP-based features, Word Vectors, or Hybrid. Table I shows the parametric study results of the suggested anti-phishing model in terms of Detection accuracy, AUC, MSE, Mean average precision, Recall, and F-1 score. Both the training and test data are analysed when the parameters are adjusted. Fig. 3 is a confusion matrix illustrating the identification of malicious URL detection.

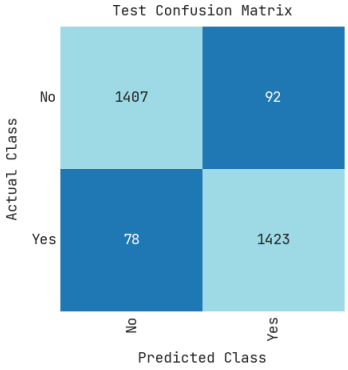
Table I shows the parametric study results of the suggested anti-phishing model in terms of Detection accuracy, AUC, MSE, Mean average precision, Recall, and F-1 score. Both the training and test data are analysed when the parameters are adjusted. Fig. 4 is a confusion matrix illustrating the identification of malicious URL detection.

TABLE I. Proposed Anti-Phishing Model-based Parametric Analysis

Metrics	Training results	Testing results
Detection accuracy	94.54	94.33
AUC	98.55	98.71
MSE	5.46	5.67
Mean average precision	89.09	88.67
Recall	98.54	98.67
F-1 score	94.54	94.34



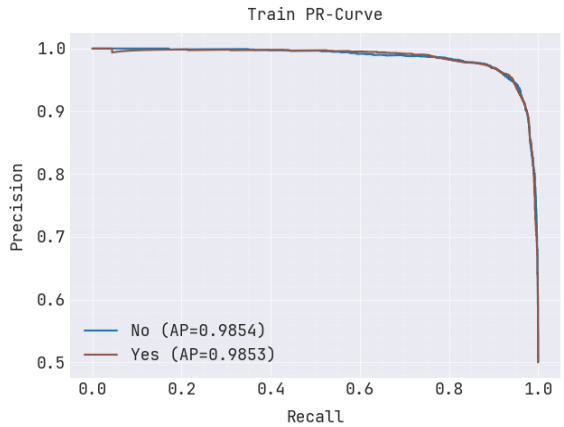
(a) Training result-based confusion matrix.



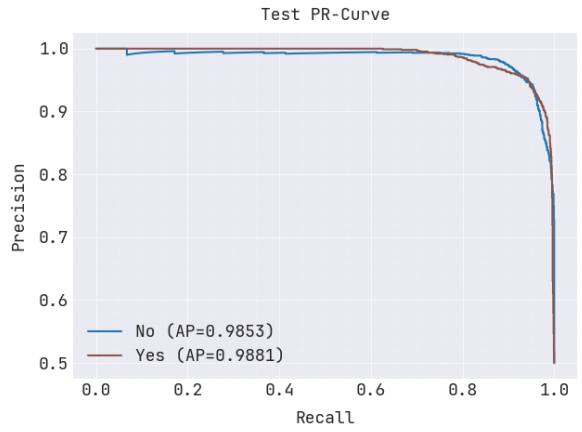
(b) Test result-based confusion matrix.

Fig. 4. Proposed model-based malicious URL detection based on confusion matrix for (a) training results, (b) Test results.

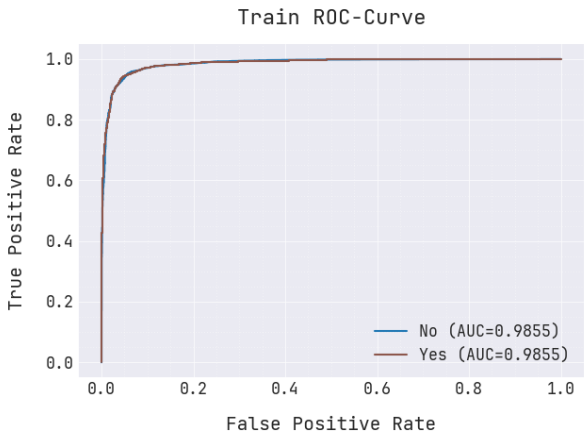
The precision-recall and ROcurve result for the proposed model is shown in Fig. 5, as determined by both training and testing. These results are analyzed based on the true positive and false positive rates, which are determined by the model's confusion matrix.



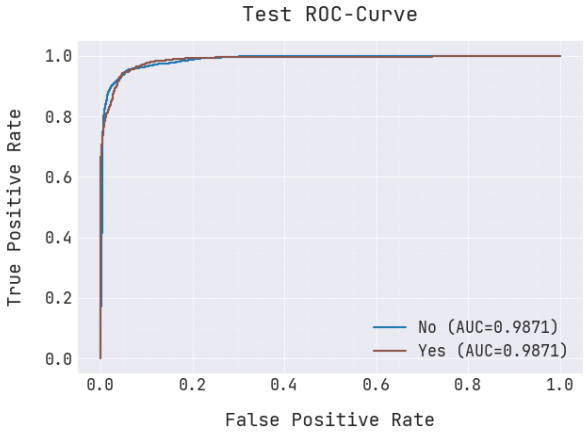
(a) Training PR- curve.



(b) Testing PR- curve.



(c) Training ROC curve.



(d) Testing ROC curve.

Fig. 5. Training and testing result analysis based on PR and ROC curve.

The proposed analysis, utilizing the training and testing results is shown in Fig. 6.

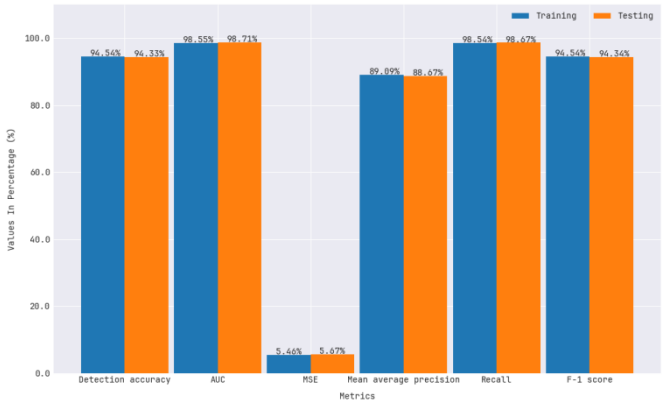


Fig. 6. Proposed training and test result analysis.

From the above figure, the proposed technique attained Detection accuracy of 94.54%, AUC 98.55%, MSE of 5.46%, Mean average precision 89.09%, Recall of 98.54% and F-1 score of 94.54% for training results; for testing results proposed technique attained Detection accuracy of 94.33%, AUC 98.71%, MSE of 5.67%, Mean average precision 88.67%, Recall of 98.67% and F-1 score of 94.34%.

TABLE II. ANALYSIS BETWEEN PROPOSED AND EXISTING TECHNIQUE

Metrics	SVM_RNN	DBM_SAE	Proposed_ LB-LSTM_ Mul_Q_capsnet
Detection accuracy	87.23	91.8	94.33
AUC	92.1	95.2	98.71
MSE	10.56	8.74	5.67
Mean average precision	81.45	86.23	88.67
Recall	91.34	95.22	98.67
F-1 score	87.66	92.33	94.34

Table II shows analysis of the anti-phishing model in terms of Detection accuracy, AUC, MSE, Mean average precision, Recall, and F-1 score.

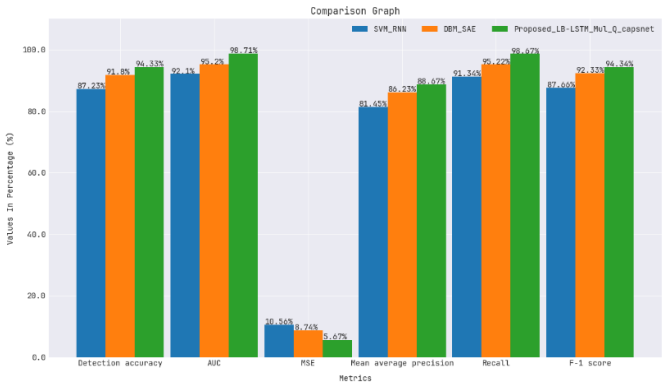


Fig. 7. Comparative analysis for anti-phishing model.

Fig. 7 analysis is shown. The proposed technique attained a Detection accuracy of 94.33%, AUC of 98.71%, MSE of 5.67%, and Mean average precision of 88.67%, Recall of 98.67%, and F-1 score of 94.34%, while existing SVM_RNN attained Detection accuracy of 87.23%, AUC 92.1%, MSE of 10.56%, Mean average precision 81.45%, Recall of 91.34% and F-1 score of 87.66%; DBM_SAE attained Detection accuracy of 91.8%, AUC 95.2%, MSE of 8.74%, Mean average precision 86.23%, Recall of 95.22% and F-1 score of 92.33%.

B. Discussion

The anti-phishing technique presented here has shown impressive results, highlighting significant progress in terms of detection accuracy, Area under the ROC Curve (AUC), Mean Squared Error (MSE), Mean Average Precision, Recall, and F-1 score. Upon closer analysis of the methodology and results, several important factors come to light. Although the proposed solution based on Natural Language Processing (NLP) is generally effective, there is a concern regarding its ability to detect pages with a single-space name, like "www.testbank.com." Phishing attacks frequently take advantage of differences in URLs, and the model's inability to address this particular situation could limit its efficacy in real-world scenarios. In a typical phishing attack, the page is designed to appear legitimate, and attackers attempt to conceal their extended URL by using unusual words to deceive customers. Customers who are already aware of phishing attacks tend to look for shorter URLs. Tests show that straight or probabilistic AI models like support vector machines and linear regression perform poorly overall. On the other hand, tree-based models significantly improve the identification of phishing URLs and produce highly effective and significant results.

The study recognises the ever-changing nature of phishing attacks, but it does not extensively explore how well the model can adapt to these evolving tactics. Phishing techniques are always evolving, so it's important to evaluate how well the suggested technique can adjust to new threats and consistently detect them accurately in the long run. The study does not thoroughly address the importance of user awareness and education in relation to the proposed anti-phishing tool. Although the model is effective, it is crucial to prioritize user education to combat phishing attacks effectively. Considering this aspect would offer a more comprehensive approach to cybersecurity. The success of the model is greatly influenced by the calibre and variety of the training data. If the training data lacks diversity or does not accurately reflect various phishing scenarios, the model may face challenges in effectively applying its knowledge to real-world situations.

Staying Ahead of Changing Threats: Phishing techniques are always changing, and attackers are constantly finding new ways to get around detection mechanisms. It is essential for the model to be able to adjust to new phishing trends and variations in order to ensure its long-term effectiveness.

Implementing deep learning models, especially those with intricate architectures such as Logistic Bayesian Long Short-Term Memory (LB-LSTM), can require significant computational resources. Deployment on platforms with limited resources, such as mobile devices, can present challenges. False Positives and User Experience: Excessively sensitive models can produce false positives, incorrectly identifying valid URLs as malicious. This may result in a subpar user experience and a decrease in trust in the system.

Striking a balance between achieving accurate detection and minimising false positives can be quite challenging.

VI. Conclusion

Phishing attack is a serious threat to any organization, as it preys on individuals' independent decision-making. Responding promptly and effectively to phishing attacks is crucial for maintaining a secure business environment. Since employees are often the primary targets of phishers due to their unpredictable online behavior, sophisticated attackers know how to bypass logical reasoning and take a more deceptive approach. This paper investigates current strategies for detecting phishing web pages using machine learning. Furthermore, this study aims to enhance the efficiency and effectiveness of phishing datasets by employing feature selection methods. Feature selection is used to optimize an up-to-date phishing dataset and expedite the model-building process. The study utilizes the logistic Bayesian LSTM model and feature extraction to assess malicious behavior, with Multilayer Q-learning and the CaspNet (Mul_Q_capsnet) model with swarm optimization applied in the process. The proposed method achieves impressive results, including an F-1 score of 94.34%, an AUC of 98.71%, an MSE of 5.67%, an MPP of 88.67%, an RPP of 98.67%, and a detection accuracy of 94.33%. Future research could focus on changes in user behavior and evaluating the significance of account suspensions as a parameter by collecting new data once these regulations have reached a stable state. Additionally, researchers could categorize the various Arabic dialects used on social media platforms, and our methodology could be extended to other popular Online Social Networks (OSNs) such as Facebook and Instagram. We will further explore the ability of the proposed technique to handle large datasets and real-time scenarios. Discover various deployment strategies for seamlessly integrating the model across multiple social media platforms, ensuring comprehensive protection for all users.

Authors' Contributions

Each author made valuable contributions to the conception, validation, formal analysis, and initial draft writing of the study. A.I.K. and B.U. were responsible for the methodology, investigation, and visualization, with B.U. also providing oversight and contributing to the writing, review, and editing. All authors have reviewed and approved the final version of the manuscript for publication.

Data Availability Statement

Data used for this article were collected by the research team and will be given to other researchers upon request.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] J. D. Rosita P and W. S. Jacob, "Multi-Objective Genetic Algorithm and CNN-Based Deep Learning Architectural Scheme for effective spam detection," International Journal of Intelligent Networks, vol. 3, pp. 9–15, 2022.

[2] B. Prabhu Kavin et al., "Machine Learning-Based Secure Data Acquisition for Fake Accounts Detection in Future Mobile Communication Networks," Wireless Communications and Mobile Computing, vol. 2022, pp. 1–10, Jan. 2022.

[3] A. S. Alhassun and M. A. Rassam, "A Combined Text-Based and Metadata-Based Deep-Learning Framework for the Detection of Spam Accounts on the Social Media Platform Twitter," Processes, vol. 10, no. 3, p. 439, Feb. 2022.

[4] R. Ghanem, H. Erbay, and K. Bakour, "Contents-Based Spam Detection on Social Networks Using RoBERTa Embedding and Stacked BLSTM," SN computer science, vol. 4, no. 4, May 2023.

[5] I. H. Sarker, Y. B. Abushark, F. Alsolami, and A. I. Khan, "IntruDTree: A Machine Learning Based Cyber Security Intrusion Detection Model," Symmetry, vol. 12, no. 5, p. 754, May 2020.

[6] R. Ghanem and H. Erbay, "Spam detection on social networks using deep contextualized word representation," Multimedia Tools and Applications, Jul. 2022.

[7] A. Almomani et al.,"Phishing Website Detection with semantic features Based on Machine learning Classifiers-A Comparative Study," International Journal on Semantic Web and Information Systems, vol. 18, no. 1, Jan. 2022.

[8] Z. Zhang, R. Hou and J. Yang, "Detection of Social Network Spam Based on Improved Extreme Learning Machine," in *IEEE Access*, vol. 8, pp. 112003-112014, 2020.

[9] N. Ahmed, R. Amin, H. Aldabbas, D. Koundal, B. Alouffi, and T. Shah, "Machine Learning Techniques for Spam Detection in Email and IoT Platforms: Analysis and Research Challenges," Security and Communication Networks, vol. 2022, pp. 1–19, Feb. 2022.

[10] B. Prabhu Kavin et al., "Machine Learning-Based Secure Data Acquisition for Fake Accounts Detection in Future Mobile Communication Networks," Wireless Communications and Mobile Computing, vol. 2022, pp. 1–10, Jan. 2022.

[11] S. Rao, Anil Kumar Verma, and T. Bhatia, "Hybrid ensemble framework with self-attention mechanism for social spam detection on imbalanced data," vol. 217, pp. 119594–119594, May 2023.

[12] M. Alshehri, A. Abugabah, A. Algarni, and S. Almotairi, "Character-level word encoding deep learning model for combating cyber threats in phishing URL detection," Computers and Electrical Engineering, vol. 100, p. 107868, May 2022.

[13] Q. Zhang, Z. Guo, Y. Zhu, P. Vijayakumar, A. Castiglione, and B. B. Gupta, "A Deep Learning-based Fast Fake News Detection Model for Cyber-Physical Social Services," Pattern Recognition Letters, vol. 168, pp. 31–38, Apr. 2023.

[14] W. Khan and M. Haroon, "An unsupervised deep learning ensemble model for anomaly detection in static attributed social networks," International Journal of Cognitive Computing in Engineering, vol. 3, pp. 153–160, Jun. 2022.

[15] E. Dubasova, A. Berdashkevich, G. Kopanitsa, P. Kashlikov and O. Metsker, "Social Network Users Profiling Using Machine Learning for Information Security Tasks," *2022 32nd Conference of Open Innovations Association (FRUCT)*, Tampere, Finland, 2022, pp. 87-92.

[16] K. Hayawi, S. Saha, M. M. Masud, S. S. Mathew, and M. Kaosar, "Social media bot detection with deep learning methods: a systematic review," Neural Computing and Applications, Mar. 2023.

[17] M. Bhattacharya, S. Roy, S. Chattopadhyay, A. K. Das, and S. Shetty, "A comprehensive survey on online social networks security and privacy issues: Threats, machine learning-based solutions, and open challenges," Security And Privacy, Oct. 2022.

[18] M. Senthil. Raja and L. Arun. Raj, "Fake news detection on social networks using Machine learning techniques," Materials Today: Proceedings, Mar. 2022.

[19] V. Niranjani, Y. Agalya, K. Charunandhini, K. Gayathri and R. Gayathri, "Spam Detection for Social Media Networks Using Machine Learning," *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, 2022, pp. 2082-2088.

[20] P. Manasa *et al.*, "Tweet Spam Detection Using Machine Learning and Swarm Optimization Techniques," in *IEEE Transactions on Computational Social Systems*.

[21] S. K. Sharma, B. Bhushan, B. Unhelkar, "Security and trust issues in internet of things: Blockchain to the rescue," CRC Press, 2020.

[22] B. Unhelkar, T. Gonsalves, "Artificial Intelligence for Business Optimization: Research and Applications," CRC Press, 2021.

[23] A. Mughaid et al., "A novel machine learning and face recognition technique for fake accounts detection system on cyber social networks," Multimedia Tools and Applications, vol. 82, no. 17, pp. 26353–26378, Jan. 2023.

[24] Y. A. Alsariera, V. E. Adeyemo, A. O. Balogun and A. K. Alazzawi, "AI Meta-Learners and Extra-Trees Algorithm for the Detection of Phishing Websites," in *IEEE Access*, vol. 8, pp. 142532-142542, 2020.

[25] C. Singh and Meenu, "Phishing Website Detection Based on Machine Learning: A Survey," *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, 2020, pp. 398-404.

[26] M. A. El-Rashidy, "A Smart Model for Web Phishing Detection Based on New Proposed Feature Selection Technique," Menoufia Journal of Electronic Engineering Research, vol. 30, no. 1, pp. 97–104, Jan. 2021.

[27] B. B. Gupta, K. Yadav, I. Razzak, K. Psannis, A. Castiglione, and X. Chang, "A novel approach for phishing URLs detection using lexical based machine learning in a real-time environment," Computer Communications, vol. 175, pp. 47–57, Jul. 2021.