International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 21 No. 2 (2025)

ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 21, No. 2, 2025

Research Paper

ENHANCING PHISHING DETECTION ACCURACY THROUGH MULTI-OBJECTIVE FEATURE OPTIMIZATION AND XGBOOST

Ambala Sukanya[1*], K.Kavitha[2], G.Swathi[2], MD Rehan[2], T.Pavan[2]
[1]Assistant Professor, [2]UG Student, [1,2]Department of Information Technology,
[1,2]Malla Reddy Engineering College and Management Sciences, Kistapur, Medchal, 501401, Telangana, India
*Corresponding author: asukanya.it@mrem.ac.in

ABSTRACT

The rapid proliferation of malicious Uniform Resource Locators (URLs) presents a serious cybersecurity threat, enabling phishing attacks, malware dissemination, and financial fraud. Conventional detection mechanisms, such as blacklisting and rule-based systems, are largely reactive and struggle to keep pace with dynamically generated and zero-day malicious URLs. Blacklisting methods fail to identify newly created threats, while rule-based approaches rely on predefined heuristics that are ineffective against evolving attack patterns. To address these limitations, this research proposes a multi-objective feature selection and XGBoost-based phishing detection framework aimed at improving classification accuracy and detection robustness. A comprehensive dataset containing both benign and malicious URLs is constructed, followed by extensive feature extraction encompassing lexical, host-based, and network-level attributes. A multi-objective optimization strategy is employed to select the most informative feature subset, reducing redundancy while enhancing model performance. The optimized feature set is then utilized to train an Extreme Gradient Boosting (XGBoost) classifier, which effectively captures complex nonlinear patterns associated with phishing URLs. Experimental evaluation demonstrates improved detection accuracy and generalization capability, enabling proactive identification of previously unseen malicious URLs. The proposed approach offers a scalable, adaptive, and efficient solution for modern phishing detection systems.

Keywords: Phishing Detection, Malicious URL Classification, Feature Selection, Multi-Objective Optimization, XGBoost, Machine Learning, Cybersecurity, URL Analysis, Fraud Detection

Received: 02-03-2025 Accepted: 27-04-2025 Published: 06-05-2025

1. INTRODUCTION

In today's interconnected society, web security is paramount in safeguarding sensitive information and maintaining trust in digital interactions. As individuals and organizations increasingly rely on online platforms for communication, commerce, and data storage, the protection of these digital assets from unauthorized access and cyber threats becomes essential. Robust web security measures help prevent data breaches, identity theft, and financial losses, thereby ensuring the stability and resilience of modern digital infrastructures.

The interaction between users and websites hinges on the implementation of effective security protocols. Secure websites employ encryption technologies, such as HTTPS, to protect data transmitted between the user's browser and the web server. Authentication mechanisms, including multi-factor authentication, further enhance security by verifying user identities before granting access to sensitive information. These measures not only protect user data but also bolster confidence in online services, fostering a safer and more trustworthy internet environment.

Websites are categorized based on their security implementations. Secure websites

utilize advanced encryption and authentication protocols to ensure data integrity and confidentiality. Semi-secure websites implement basic security measures but lack comprehensive protections, increasing their susceptibility to cyber threats. Insecure websites fail to employ adequate security practices, leaving user data vulnerable to interception and exploitation. Users must engage with secure websites, especially when handling sensitive information, to mitigate potential risks.

The COVID-19 pandemic has underscored the critical importance of web security in real-time applications. With a significant shift towards remote work and online services, there has been a notable increase in cyberattacks. A survey conducted among IT security professionals worldwide indicates that data exfiltration and leakage have become more prevalent since the onset of the pandemic. Phishing emails have also seen a significant rise, as cybercriminals exploit the increased online presence of individuals and organizations. These statistics highlight the urgent need for enhanced web security measures to protect against evolving cyber threats in a post-pandemic world.

2. LITERATURE SURVEY

Omolara, Abiodun Esther, and Moatsum Alawida, et al. [1] (2025) considered the challenges on malicious URLs was still an open problem. These URLs often hid behind static links in emails or web pages, posing a threat to individuals and organizations. Despite blacklisting services, many harmful sites evaded detection due to inadequate scrutiny or recent creation. Hence, to improve URL detection, a Diverse and Efficient Ensemble (DaE2) machine learning algorithm was developed using four ensemble models, that is, AdaBoost, Bagging, Stacking, and Voting to classify URLs. After preprocessing, the experimental result showed that all models achieved over 80% accuracy, with AdaBoost reaching 98.5% and Stacking offering the fastest runtime. AdaBoost and Bagging also delivered strong performance, with F1 scores of 0.980 and 0.976, respectively.

Jain, Souratn, et al. [2] (2025) discussed the methods of improving anomalous behavior identification, Intrusion Detection Systems (IDS), and real-time threat analysis, especially focusing on supervised, unsupervised, and reinforcement types of learning. Three burgeoning fields of interest, explainable AI (XAI), adversarial machine learning, and incorporating blockchain into AI methodology, were identified as crucial in responding to new challenges like adversarial attacks and data protection. However, AI and ML had limitations, including high computational demand, lack of data, and bias; hence, future work was needed. This paper outlined a possible interdisciplinary research agenda for enhancing AI in cybersecurity involving integrated platforms, technology case data, and an ethical dimension.

Kandasamy, et al. [3] (2025) traditional rule-based methods struggled to cope with the complexity of these attacks, creating a need for more advanced, adaptive intrusion detection systems. This research introduced the AEXB Model, a hybrid deep learning approach that combined the feature extraction capabilities of an AutoEncoder with the classification power of XGBoost. By combining these complementary methods, the model enhanced detection accuracy and significantly reduced false positives. The AEXB Model's methodology encompassed robust preprocessing steps, including data cleaning, scaling, and dimensionality reduction, followed by comprehensive feature engineering and selection techniques, such as Recursive Feature Elimination (RFE) and correlation analysis.

Reyes-Acosta, et al. [4] (2025) reviewed process identified an initial pool of 403 sources (367 white literature, WL, and 36 gray literature, GL), which was refined to a set of 263 sources (247 WL and 16 GL), addressing research questions. Key contributions included: (1) a detailed classification of core technologies influencing smart industry,

enhancing identification of specific security vulnerabilities; (2) a systematic categorization of critical security challenges, including network threats, software threats, tampering and deception attacks, and advanced attacks, with a graphical summary of prevalent threats; (3) an overview of recent advancements in securing IIoT environments, encompassing theoretical frameworks, intrusion detection systems (IDS), intelligent algorithms, and datasets.

Wang, et al. [5] (2025) entropy sequence of the document was represented as an entropy image, and at the same time, the grayscale level co-occurrence matrix and the texture and spatial information stored in it were converted into a grayscale matrix image. The three types of images were then fused to obtain the GGE color image. The Convolutional Block Attention Module-EfficientNet-B0 (CBAM-EfficientNet-B0) model was used for classification, combining transfer learning and applying the pre-trained model on the ImageNet dataset to the feature extraction process of GGE images.

Dubey, et al. [6] (2025) a hybrid Flame-Sailfish Optimization (HFSO)-based deep learning framework was proposed. It combined the feature extraction capabilities of ResNet50 with the efficiency of MobileNetV2. The HFSO algorithm optimized crucial parameters such as detection thresholds and learning rates so that the model could take full advantage of computing capacity and still operate in real time on devices with limited resources. The model was tested on three datasets—Kaggle Face Mask Detection dataset, Public Places dataset, and Public Videos dataset—achieving up to 97.5% accuracy. It outperformed the previous leader in all cases. The results proved that this framework was reliable and easily applicable for identifying people wearing masks under different conditions.

Khayat, et al. [7] (2025) the latest technologies used in the SOC environment were analyzed to help address different operational and technical challenges and bring out their capabilities. Various methods, ranging from automated incident response and behavioral analytics to neural networks and deep learning, were classified and compared. In addition, an in-depth reference architectural model, which was a blueprint for SOC integrating AI and ML into SOCs, was introduced. The proposed model provided a structured framework for implementation and offered insights into different SOC components and their interactions. Moreover, this systematic review emphasized the benefits of these technologies for enhancing security operations. Finally, a case study was presented to describe the function of ML- and AI-powered SOC components to achieve optimum security.

Zhang, et al. [8] (2025) a deep-learning-based method was proposed for detecting malware using API call sequences. This method transformed the API call sequence into a grayscale image and performed classification in conjunction with sequence features. By leveraging a range of deep-learning algorithms, diverse behavioral information from software was extracted, encompassing semantic details, time-series information, API call frequency data, and more. Additionally, a specialized neural network framework was introduced, and the impact of pixel size on classification effectiveness during the grayscale image-mapping process was assessed. The experimental results showed that the accuracy of the classification method was as high as 99%. Compared with other malware-detection techniques, especially those based on API call sequences, this method mapped API call sequences to gray image analysis and achieved higher detection accuracy.

Al-Shurbaji, et al. [9] (2025) a wide range of researched papers was highlighted, presenting key findings, methodologies, advancements, shortcomings, and challenges in the field. Additionally, a qualitative comparison with existing surveys was performed using author-defined metrics to underscore the uniqueness of this survey. Challenges, limitations, and future research directions were also discussed,

emphasizing the distinctive contributions of this review. Ultimately, this survey served as a guideline for future researchers, contributing to the advancement of botnet detection methods in IoT environments and enhancing security against botnet threats.

Johny, et al. [10] (2025) a novel multimodal fusion algorithm was proposed, leveraging three different visual malware features: Grayscale Image, Entropy Graph, and SimHash Image, with which exhaustive experiments were conducted independently on each feature and combinations of all three using fusion operators such as average, maximum, add, and concatenate for effective malware detection and classification. The proposed strategy had a detection rate of 1.00 (on a scale of 0–1) in identifying malware in the given dataset. Its interpretability was explained with visualization techniques such as t-SNE, SHAP, and Grad-CAM. Experimental results showed that the model worked even for a highly imbalanced dataset. The effectiveness of the proposed method on obfuscated malware was also assessed, achieving state-of-the-art results. Additionally, adversarial attacks were performed on the proposed model using Generative Adversarial Networks (GANs), and adversarial retraining was employed as a defense strategy.

Imtiaz, et al. [11] (2025) a key distinguishing feature of XIoT lay in its emphasis on interpretability, enabling stakeholders to gain insights into the rationale behind its predictions. By integrating explainable AI mechanisms, XIoT delivered accurate classifications of diverse IoT attacks and elucidated the key factors driving its decision-making process. This transparency enhanced trust in the model's outputs and facilitated informed decision-making by cybersecurity analysts and network administrators. Additionally, leveraging the high-speed, low-latency characteristics of optical networks, XIoT's model processed extensive IoT data streams more efficiently, supporting real-time detection in expansive IoT ecosystems.

Mahdi, et al. [12] (2025) the Independent Component Analysis (ICA) was first utilized for isolating relevant features from obfuscated data to be prepared for binary classification. For further data analysis, essential feature selection, and a deeper comprehension of the relationships inside the dataset, the Pearson correlation coefficient was applied to the dataset to be prepared for multiclass classification. This dual scheme improved the feature extraction process depending on the classification type, enhancing the system's versatility and performance.

Hallaji, et al. [13] (2025) aimed to analyze measurements considered when recording network traffic and concluded which features contributed more to detecting APT samples. To do this, we studied the features associated with various APT cases and determined their importance using a machine learning framework. To ensure the generalization of our findings, several feature selection techniques were employed and paired with different classifiers to evaluate their effectiveness. Our findings provided insights into how APT detection could be enhanced in real-world scenarios.

Mishra, et al. [14] (2025) the implementation made use of Gaussian Radial Basis Function (RBF) alongside Reflectional Switch Activation Function (RSWAF). The RBFs allowed the network to capture local non-linear relationships, improving the performance of the model, both in terms of accuracy and computational efficiency. The RSWAF provided a computationally efficient activation mechanism that facilitated faster learning and inference. Our experiments demonstrated that the faster-KAN implementation significantly reduced training and inference times while maintaining high accuracy and robustness in detecting anomalies and attacks.

Hussain, et al. [15] (2025) introduced a novel framework for automating software defect prediction, focusing on eight specific defects: SIGFPE, NZEC, LOGICAL, SYNTAX, SIGSEGV, SIGABRT, SEMANTIC, and

LINKER. Our research involved a specialized dataset comprising nine classes, including eight common programming errors and one error-free class. The goal was to enhance software testing and development processes by identifying defects within code snippets. The proposed framework utilized a CodeBERT-based algorithm for defect prediction, optimizing model hyperparameters to achieve superior accuracy. Comparative analysis against established models such as RoBERTa, Microsoft CodeBERT, and GPT-2 demonstrated that our approach yielded significant improvements in prediction performance, with accuracy gains of up to 20% and 7% respectively in binary and multi-class experimentation.

Wazid, et al. [16] (2025) IoT-enabled ITS systems and devices had to be protected from cyber-attacks for various reasons, including preserving sensitive data, guaranteeing privacy, preventing unauthorized access, and protecting against the risk of interruptions or manipulations. Malware attacks affected the working and performance of the deployed smart IoT devices. We proposed a secure deep learning-enabled malware attack detection for IoT-enabled ITS (in short, SDLMA-IITS). The approach of explainable artificial intelligence (XAI) was utilized for the effective detection of malware.

Al-Haija, et al. [17] (2025) ensured security in these IoT settings was crucial for protecting against malicious activities and safeguarding data. Real-time identification and response to potential intrusions and attacks were essential, and intrusion detection systems (IDS) were pivotal in this process. However, the dynamic and diverse nature of the IoT environment presented significant challenges to existing IDS solutions, which were often based on rule-based or statistical approaches. Deep learning, a subset of artificial intelligence, had shown great potential to enhance IDS in IoT. Deep learning models could identify complex patterns and characteristics by utilizing artificial neural networks, automatically building hierarchical representations from data.

Amissah, et al. [18] (2025) methodology involved rigorous experimentation with four classifiers: K-Nearest Neighbors (KNN), XGBoost, Decision Tree, and Multi-Layer Perceptron (MLP). The results demonstrated that dimensionality reduction techniques, particularly UMAP, improved the performance of KNN and Decision Tree classifiers while adversely affected the performance of XGBoost and MLP. Notably, KNN consistently outperformed the other classifiers across different scenarios, indicating its robustness in handling reduced feature spaces. This study concluded that the effectiveness of dimensionality reduction was contingent upon the specific characteristics of the classifiers employed.

Delavari, et al. [19] (2025) proposed an innovative Deep Reinforcement Learning (DRL)-based technique to enhance the detection and mitigation of DDoS attacks in SDN environments. By leveraging the adaptive learning capabilities of DRL, the proposed model continuously learned and adapted to new attack patterns, providing a robust defense mechanism. The model utilized a combination of Autoencoder (AE) and Bidirectional Gated Recurrent Unit (BGRU) to effectively analyze traffic patterns and detect anomalies. Experimental results, conducted using a comprehensive dataset from real network traffic, demonstrated the superior accuracy, higher detection rate, and reduced false-positive rates of our approach compared to existing methods.

Balasubramanian, et al. [20] (2025) proposed an efficient platform capable of processing compute-intensive data pipelines, based on cloud computing, for real-time detection, collection, and sharing of CTI from various online sources. We developed a prototype platform (TSTEM) with a containerized microservice architecture that used Tweepy, Scrapy, Terraform, Elasticsearch, Logstash, and Kibana (ELK), Kafka, and Machine Learning Operations (MLOps) to

autonomously search, extract, and index indicators of compromise (IOCs) in the wild. Moreover, the provisioning, monitoring, and management of the platform were achieved through infrastructure as code (IaC). Custom focus-crawlers collected web content, processed by a first-level classifier to identify potential IOCs. Relevant content advanced to a second level for further examination. State-of-the-art natural language processing (NLP) models were used for classification and entity extraction, enhancing the IOC extraction methodology.

3. PROPOSED SYSTEM

The rapid expansion of digital platforms transformed how individuals and organizations communicated, conducted commerce, and stored data. This increased reliance on online services underscored the critical need for robust web security measures to protect sensitive information and maintain trust in digital interactions. Effective security protocols, such as HTTPS encryption and multi-factor authentication, became essential in preventing data breaches, identity theft, and financial losses, thereby ensuring the stability and resilience of modern digital infrastructures.

The proposed system architecture started with an input URL dataset and allowed new URL entries. It then extracted features from the URLs for further processing. After that, it applied Min-Max scaling to normalize the extracted features. The dataset was then split into training and testing subsets.

The proposed XGBoost model was used for classification or prediction. Once the model was trained, its performance was evaluated based on certain metrics. The system then categorized the results into normal and malicious URLs. Finally, an optional chatbot feature was included to assist users in interacting with the system.

The dataset was split into training and testing sets for model evaluation. The proposed XGBoost algorithm was then applied to classify URLs based on the extracted features. The model's performance was evaluated, and a comparison was made with normal models to assess its effectiveness. Additionally, an optional chatbot was integrated to provide real-time assistance or feedback based on the classification results.

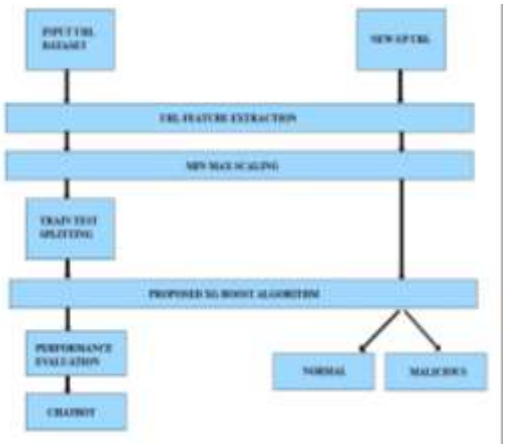

Fig. 1: Proposed System Architecture

Figure 2 provided illustrates the operation of an XGBoost Classifier, a powerful machine learning algorithm often used for tasks like malicious URL classification. XGBoost (Extreme Gradient Boosting) is an ensemble method that builds a series of decision trees iteratively, optimizing them using gradient-based techniques to improve prediction accuracy. In malicious URL classification, XGBoost starts with URL features, initializes a weak learner, and iteratively builds decision trees to correct prediction errors. It uses gradient-based optimization to focus on misclassified URLs, minimizes a loss function with regularization, and updates the model cautiously with a learning rate. The final ensemble of trees provides a robust prediction, effectively distinguishing malicious URLs from benign ones by learning complex patterns in the data while avoiding overfitting. This makes XGBoost a powerful tool for such tasks, often outperforming simpler models like logistic regression or single decision trees.

Step 1: Input Features: The process begins with the Input Features, which are the characteristics of the URLs to be classified as malicious or benign. For malicious URL classification, these features might include the URL's length, the presence of suspicious keywords (e.g., "login," "free"), the number of

subdomains, the use of HTTPS, the domain age, or the presence of special characters (e.g., "@" or "%"). These features are extracted from the dataset and fed into the XGBoost Classifier as numerical or categorical values. The input data is typically a matrix where each row represents a URL and each column represents a feature. This matrix serves as the starting point for the model to learn patterns distinguishing malicious URLs from benign ones.

Step 2: Initialize a Single Weak Learner: The next step is to Initialize a Single Weak Learner, which is the starting point of the XGBoost model. In XGBoost, a weak learner is typically a decision tree that performs slightly better than random guessing. For the first iteration, this weak learner is initialized with a simple prediction, often the average of the target variable (e.g., for binary classification of URLs, where malicious = 1 and benign = 0, the initial prediction might be the log-odds of the classes, such as 0.5 if the dataset is balanced). This initial model makes a baseline prediction for each URL in the dataset, which will be iteratively improved in subsequent steps.

Step 3: Gradient-Based Error Calculation: Once the initial weak learner makes its predictions, the Gradient-Based Error Calculation step evaluates how well these predictions match the actual labels (malicious or benign). XGBoost uses a loss function to measure the error—typically, for binary classification, this is the log-loss (or cross-entropy loss). The gradient of the loss function with respect to the predictions is calculated for each URL in the dataset. These gradients represent the direction and magnitude of the error: a positive gradient indicates the prediction is too low (e.g., a malicious URL predicted as benign), while a negative gradient indicates the prediction is too high. Additionally, XGBoost computes the second-order gradients (Hessians) to better approximate the loss landscape, which helps in optimizing the model more efficiently.

Step 4: Decision Trees Creation: Using the gradients calculated, XGBoost proceeds to Decision Trees Creation. This step involves building a series of decision trees (labeled as Decision Tree 1, Decision Tree 2, ..., Decision Tree N in the diagram). Each tree is constructed to correct the errors of the previous model by focusing on the residuals (errors) indicated by the gradients. For malicious URL classification, a decision tree might split the data based on a feature like "URL length > 50" or "contains keyword 'login' = yes/no." The tree is built by recursively splitting the feature space to minimize a loss function, which now includes the gradients and Hessians from the previous step. XGBoost uses a unique approach called "approximate greedy algorithm" or "histogram-based learning" to efficiently find the best splits, ensuring scalability for large datasets like those in URL classification.

Step 5: Fitting a New Weak Learner: In the Fitting a New Weak Learner step, each new decision tree is added to the model as a weak learner. The tree is fitted to the residuals (errors) of the current ensemble, weighted by the gradients. In the context of malicious URL classification, this means the tree focuses on URLs that were misclassified in the previous iteration—for example, if a malicious URL was predicted as benign, the tree will try to adjust the prediction by assigning a higher weight to that URL's features. The tree's output is a small correction to the previous predictions, ensuring that the model improves incrementally without overfitting. This process repeats for a predefined number of iterations (or trees), as indicated by Decision Tree 1 to Decision Tree N in the diagram.

Step 6: Minimized Loss Function: As each new tree is added, the Minimized Loss Function step ensures that the overall loss of the model decreases. XGBoost combines the predictions of all trees to minimize the loss function, which includes both the classification error (log-loss) and a regularization term to prevent overfitting. The regularization term penalizes overly complex trees by adding

constraints on the tree depth, the number of leaves, and the magnitude of the leaf weights. For malicious URL classification, this step ensures that the model doesn't overfit to noise in the data (e.g., irrelevant features like specific but rare keywords) while still capturing meaningful patterns (e.g., long URLs with suspicious characters being more likely to be malicious).

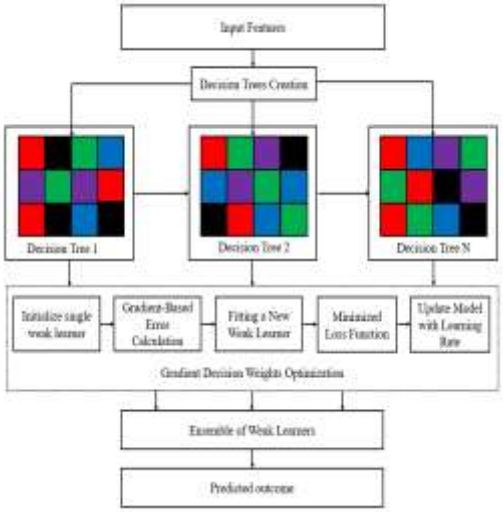

Fig. 2: XGBoost Classifier Block Diagram.

Step 7: Update Model with Learning Rate: The Update Model with Learning Rate step combines the predictions of the new tree with the existing ensemble. XGBoost introduces a learning rate (also called shrinkage), which scales the contribution of each new tree. For example, if the learning rate is 0.1, the new tree's predictions are multiplied by 0.1 before being added to the ensemble. This slow, cautious update prevents the model from making drastic changes that could lead to overfitting. In the context of URL classification, this step ensures that the model gradually learns to distinguish malicious URLs from benign ones, refining its predictions with each iteration while maintaining stability.

Step 8: Gradient Decision Weights Optimization: The Gradient Decision Weights Optimization step ties together the previous steps by optimizing the weights of the decision trees using the gradients and Hessians. XGBoost adjusts the leaf values of each tree to minimize the loss function more effectively, using a second-order Taylor approximation of the loss. This optimization ensures that the trees focus on the most significant errors in the predictions. For malicious URL classification, this means the model prioritizes correcting misclassifications of URLs that are harder to classify (e.g., phishing URLs that mimic legitimate ones) while de-emphasizing URLs that are already correctly classified.

Step 9: Ensemble of Weak Learners: After iterating through the above steps for a specified number of trees, the result is an Ensemble of Weak Learners. This ensemble is the final XGBoost model, which combines the predictions of all decision trees to produce a robust classifier. Each tree contributes a small correction to the overall prediction, and the combined output is a weighted sum of the individual tree predictions. For malicious URL classification, the ensemble might output a probability score for each URL, indicating the likelihood of it being malicious (e.g., a score of 0.9 suggests a high probability of being malicious, while 0.2 suggests benign).

Step 10: Predicted Outcome: Finally, the Predicted Outcome is generated by the ensemble model. For binary classification of URLs, XGBoost outputs a probability score, which is then thresholded (typically at 0.5) to classify the URL as malicious (1) or benign (0). For example, a URL with a long length, suspicious keywords, and no HTTPS might receive a high probability of being malicious and be classified as such. The final model is highly accurate because it has iteratively learned from its errors, focused on hard-to-classify examples, and balanced complexity with generalization through regularization and the learning rate.

4. RESULTS AND DISCUSSION

Figure 2 represents the user interface (UI) screen of the research work, though no visual or detailed description of the UI is provided. Based on the context, this UI likely serves as the front-end for a client-side tool designed to detect phishing attacks in real-time. The UI would typically allow users to input URLs and display the classification results (e.g., "Phishing" or "Safe"). It might include fields

for entering a URL, a button to trigger the analysis, and an output section showing the prediction along with a confidence score or additional details like the features used for classification. Given the research's focus on a hybrid approach using classification algorithms (SVM, RFC, and XGBoost), the UI could also provide options to select the model for prediction or display comparative performance metrics. Additionally, as seen in later figures (e.g., Figure 9), the UI might integrate with a chatbot application to deliver real-time phishing alerts, enhancing user interaction and accessibility.

Fig. 2: UI Screen of Research Work.

Figure 3 provides a glimpse of the dataset used in the research, which contains URLs and their corresponding labels for phishing detection. The dataset includes a "url" column and a "label" column, where the label 1.0 indicates a phishing URL. Five sample entries are shown:

- Row 0: nobell.it/70ffb52d079109dca5664cce6f317373782/... with label 1.0 (phishing)
- Row 1: www.dghjdgf.com/paypal.co.uk/cycgi-bin/webscrc... with label 1.0 (phishing)
- Row 2: serviciosbys.com/paypal.cgi.bin.get-into.herf.... with label 1.0 (phishing)
- Row 3: mail.printakid.com/www.online.americanexpress.... with label 1.0 (phishing)
- Row 4: thewhiskeydregs.com/wp-content/themes/widescre... with label 1.0 (phishing)

These URLs exhibit characteristics typical of phishing, such as mimicking legitimate services (e.g., "paypal.co.uk" in Row 1, "americanexpress" in Row 3) or using long, obfuscated paths (e.g., Row 0). The dataset is likely preprocessed to extract features like URL length, presence of keywords (e.g., "paypal"), or suspicious characters, which are then used to train the classification models. The labels (all 1.0 in this sample) suggest this snippet focuses on phishing examples, but the full dataset would include both phishing (1.0) and legitimate (0.0) URLs for balanced training.

Fig. 3: Sample Dataset.

Figure 4 describes a count plot of the "Phishing" column in the dataset, though the visual plot itself is not provided. This plot would typically show the distribution of the two classes: "Phishing" (label 1.0) and "Legitimate" (label 0.0). Given the dataset size from Figure 5 (50,000 total records) and the test set classification reports (approximately 5,015 phishing and 4,985 legitimate URLs in a 10,000-record test set), the dataset appears to be nearly balanced. A count plot would likely show two bars: one for phishing URLs (around 25,150, assuming a 50.3% phishing proportion as seen in the test set) and one for legitimate URLs (around 24,850). This balance is crucial for training machine learning models to avoid bias toward one class, ensuring that the classifiers (SVM, RFC, and XGBoost) can learn to distinguish between phishing and legitimate URLs effectively.

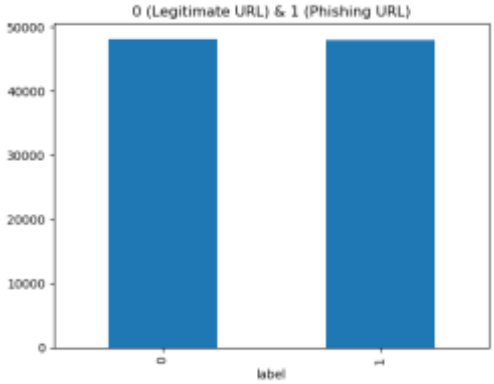

Fig. 4: Count plot of Phishing column in dataset.

Figure 5 refers to the selected features used for training the models, though specific features are not listed. In the context of phishing URL detection, feature selection is critical to identify the most discriminative characteristics of URLs. Common features might include URL length, number of subdomains, presence of HTTPS, domain age, number of special characters (e.g., "@", "%"), presence of suspicious keywords (e.g., "login," "paypal"), and the number of redirects. Feature selection techniques like correlation analysis, mutual information, or recursive feature elimination might have been applied to reduce dimensionality and improve model performance. For example, a feature like "presence of HTTPS" might be selected because legitimate sites are more likely to use HTTPS, while phishing sites often do not. These selected features are then used to train the SVM, RFC, and XGBoost models, ensuring that only the most relevant attributes contribute to the classification process

Fig. 5: Selected Features.

Figure 6 details the train-test splitting of the dataset, which is essential for evaluating the models' performance. The dataset contains a total of 50,000 records. These are split as follows:

- Total records to train: 40,000 (80% of the dataset)
- Total records to test: 10,000 (20% of the dataset)

This 80-20 split is a standard practice in machine learning to ensure the model is trained on a substantial portion of the data while reserving a separate portion for unbiased evaluation. The training set (40,000 records) is used to fit the SVM, RFC, and XGBoost models, allowing them to learn patterns distinguishing phishing URLs from legitimate ones. The test set (10,000 records) is then used to assess the models' generalization to unseen data, as shown in the performance metrics in Figures 5, 6, and 7. The balanced nature of the test set (4,985 legitimate and 5,015 phishing URLs, as seen in the classification reports) ensures a fair evaluation of the models' ability to handle both classes.

Fig. 6: Train Test Splitting.

Figure 7 also provides the performance metrics of the existing Support Vector Machine (SVM) model on the test set of 10,000 records. The SVM results are as follows:

- **Accuracy**: 96.95% (the proportion of correctly classified URLs)
- **Precision**: 97.04% (the proportion of predicted phishing URLs that are actually phishing)
- **Recall**: 96.96% (the proportion of actual phishing URLs correctly identified)
- **F1-Score**: 96.95% (the harmonic mean of precision and recall)
- **Sensitivity**: 99.20% (the true positive rate, i.e., recall for the phishing class)

- **Specificity**: 94.72% (the true negative rate, i.e., the proportion of legitimate URLs correctly identified)

The classification report breaks down the performance by class:

- **Legitimate (0.0)**: Precision 0.95, Recall 0.99, F1-Score 0.97, Support 4,985
- **Phishing (1.0)**: Precision 0.99, Recall 0.95, F1-Score 0.97, Support 5,015
- **Accuracy**: 0.97 (97%)
- **Macro Avg**: Precision 0.97, Recall 0.97, F1-Score 0.97
- **Weighted Avg**: Precision 0.97, Recall 0.97, F1-Score 0.97

The SVM performs well, with high accuracy (96.95%) and balanced precision and recall for both classes. However, its specificity (94.72%) is relatively lower, indicating that it misclassifies some legitimate URLs as phishing (false positives). The high sensitivity (99.20%) shows that it is very effective at detecting phishing URLs, which is critical for a phishing detection system.

Fig. 7: Existing SVM Performance

Figure 8 presents the performance of the existing Random Forest Classifier (RFC) on the same test set of 10,000 records. The RFC results are:

- **Accuracy**: 98.56%
- **Precision**: 98.56%
- **Recall**: 98.56%
- **F1-Score**: 98.56%
- **Sensitivity**: 99.12%
- **Specificity**: 98.01%

The classification report for RFC is:

- **Legitimate (0.0)**: Precision 0.98, Recall 0.99, F1-Score 0.99, Support 4,985
- **Phishing (1.0)**: Precision 0.99, Recall 0.98, F1-Score 0.99, Support 5,015
- **Accuracy**: 0.99 (98.56%)
- **Macro Avg**: Precision 0.99, Recall 0.99, F1-Score 0.99
- **Weighted Avg**: Precision 0.99, Recall 0.99, F1-Score 0.99

The RFC outperforms the SVM, achieving a higher accuracy (98.56%) and better balance between sensitivity (99.12%) and specificity (98.01%). This indicates fewer false positives and false negatives compared to the SVM, making RFC a more robust model for phishing detection. The near-identical precision, recall, and F1-scores for both classes highlight RFC's ability to handle the balanced dataset effectively.

Fig. 8: Existing RFC Performance

Figure 9 showcases the performance of the proposed XGBoost Classifier, which is the focus of the research. The XGBoost results on the test set are:

- **Accuracy**: 99.16%
- **Precision**: 99.16%
- **Recall**: 99.16%
- **F1-Score**: 99.16%
- **Sensitivity**: 99.50%
- **Specificity**: 98.82%

The classification report for XGBoost is:

- **Legitimate (0.0)**: Precision 0.99, Recall 0.99, F1-Score 0.99, Support 4,985
- **Phishing (1.0)**: Precision 0.99, Recall 0.99, F1-Score 0.99, Support 5,015
- **Accuracy**: 0.99 (99.16%)
- **Macro Avg**: Precision 0.99, Recall 0.99, F1-Score 0.99
- **Weighted Avg**: Precision 0.99, Recall 0.99, F1-Score 0.99

The XGBoost model achieves the highest performance among the three models, with an accuracy of 99.16%, a sensitivity of 99.50%, and a specificity of 98.82%. These metrics indicate that XGBoost is exceptionally effective at detecting phishing URLs while minimizing false positives. Compared to SVM (96.95% accuracy) and RFC (98.56% accuracy), XGBoost's superior performance demonstrates the effectiveness of its gradient boosting approach, as explained in the previous block diagram, in capturing complex patterns in the URL data.

Fig. 9: Proposed XGBoost Performance

Figure 10 provides examples of predictions made by the XGBoost model on test URLs, showcasing its practical application. The predictions are:

- https://www.education-online.nl/Cliquez.ici.cas.inria.fr.cas.login/login.html → Predicted as **Phishing**
- http://www.revistas-academicas.com → Predicted as **Phishing**
- http://www.google.com → Predicted as **Safe**
- http://www.yahoo.com → Predicted as **Safe**
- http://www.horizonsgallery.com/js/bin/ssl1/_id/www.paypal.com/fr/cgi-bin/webscr/cmd=_registration-run/login.php?cmd=_login-run&dispatch=1471c4bdb044ae2be9e2fc3ec514b88b1471c4bdb044ae2be9e2fc3ec514b88b → Predicted as **Phishing**
- http://www.phlebolog.com.ua/libraries/joomla/results.php → Predicted as **Phishing**
- http://www.docs.google.com/spreadsheet/viewform?formkey=dE5rVEdSV2pBdkpSRy11V3o2eDdwbnc6MQ → Predicted as **Phishing**

These predictions align with expected patterns. For instance, well-known domains like google.com and yahoo.com are correctly classified as safe, while URLs with suspicious characteristics—such as long paths, keywords like "login" or "paypal," or unusual domains—are flagged as phishing. The model's ability to correctly classify these URLs reflects its high accuracy (99.16%) and its effectiveness in real-world scenarios.

Fig. 10: Prediction From Test Data.

Figure 11 highlights the integration of the phishing detection system into a chatbot application, demonstrating its real-time functionality. An example is provided:

- **Test Case 3**: URL http://www.revistas-academicas.com
- **Result**: Identified as **Phishing**

The chatbot successfully triggers a phishing alert for this URL, which matches the prediction in Figure 10. This integration aligns with the research's objective to design a client-side tool for real-time phishing detection. The chatbot likely operates by taking a URL input from the user, running it through the XGBoost model, and delivering an alert if the URL is classified as phishing. This feature enhances the usability of the system, allowing users to receive immediate feedback in a conversational format, which is particularly useful for non-technical users who need quick and clear phishing warnings.

International Journal of Engineering Research and Science & Technology

Fig. 11: Chatbot Application.

Table 1 compares the performance of the SVM, RFC, and XGBoost models specifically for the Normal (Legitimate) class, which represents URLs that are not phishing (labeled as 0.0). The metrics are derived from the classification reports provided in Figures 5, 6, and 7. For SVM, the precision is 0.95, meaning 95% of URLs predicted as legitimate are actually legitimate, but this is the lowest among the three models, indicating a higher rate of false positives (phishing URLs misclassified as legitimate). Its recall is 0.99, showing that 99% of actual legitimate URLs are correctly identified, and the F1-score (harmonic mean of precision and recall) is 0.97. RFC improves on this, with a precision of 0.98, recall of 0.99, and F1-score of 0.99, reflecting better balance and fewer misclassifications. XGBoost performs the best, with a precision of 0.99, recall of 0.99, and F1-score of 0.99, demonstrating near-perfect classification of legitimate URLs. The support (number of legitimate URLs in the test set) is 4,985 for all models, confirming the consistency of the test set across evaluations. This table highlights XGBoost's superior ability to correctly classify legitimate URLs while minimizing false positives, which is crucial for ensuring safe URLs are not flagged unnecessarily in a phishing detection system.

Table 1. Normal Class Comparison

Model	Precision	Recall	F1-Score	Support
Existing SVM	0.95	0.99	0.97	4985
Existing RFC	0.98	0.99	0.99	4985
Proposed XGBoost	0.99	0.99	0.99	4985

Table 2 focuses on the performance of the SVM, RFC, and XGBoost models for the Phishing class (labeled as 1.0), which represents malicious URLs. The metrics are sourced from the classification reports in Figures 5, 6, and 7. For SVM, the precision is 0.99, meaning 99% of URLs predicted as phishing are indeed phishing, but its recall is 0.95, indicating that 5% of actual phishing URLs are missed (false negatives), which is a concern for a phishing detection system. Its F1-score is 0.97, reflecting a decent balance. RFC improves significantly, with a precision of 0.99, recall of 0.98, and F1-score of 0.99, showing better detection of phishing URLs with fewer misses. XGBoost again performs the best, with a precision of 0.99, recall of 0.99, and F1-score of 0.99, indicating it detects nearly all phishing URLs while maintaining high precision. The support for the phishing class is 5,015 across all models, confirming the balanced nature of the test set. This table underscores XGBoost's effectiveness in identifying phishing URLs, which is critical for the research's objective of real-time phishing detection, as missing phishing URLs could lead to security risks for users.

Table 2. Phishing Class Comparison

Model	Model	Precision	Recall	F1-Score	Support
Existing SVM	SVM	0.99	0.95	0.97	5015
Existing RFC	RFC	0.99	0.98	0.99	5015
Proposed XGBoost	XGBoost	0.99	0.99	0.99	5015

Table 3 provides an overall comparison of the SVM, RFC, and XGBoost models across multiple performance metrics, as reported in Figures 5, 6, and 7. SVM achieves an accuracy of 96.95%, with a precision of 97.04%, recall of 96.96%, and F1-score of 96.95%. Its sensitivity (true positive rate for phishing) is high at 99.20%, but its specificity (true negative rate for legitimate) is the lowest at 94.72%, indicating a higher rate of false positives. RFC improves across all metrics, with an accuracy of 98.56%, precision of 98.56%, recall of 98.56%, and F1-score of 98.56%. Its sensitivity is 99.12%, and specificity rises to 98.01%, showing better balance in handling both classes. XGBoost outperforms both, with an accuracy of 99.16%, precision of 99.16%, recall of 99.16%, and F1-score of 99.16%. Its sensitivity is the highest at 99.50%, and specificity is 98.82%, demonstrating exceptional performance in detecting both phishing and legitimate URLs. This table confirms XGBoost as the most effective model for the research's goal of designing a client-side tool for real-time phishing detection, as it minimizes both false positives and false negatives while achieving the highest overall accuracy on the test set of 10,000 records.

Table 3. Overall Comparison

Model	Accuracy	Precision	Recall	F1-Score	Sensitivity	Specificity
Existing SVM	96.95%	97.04%	96.96%	96.95%	99.20%	94.72%
Existing RFC	98.56%	98.56%	98.56%	98.56%	99.12%	98.01%
Proposed XGBoost	99.16%	99.16%	99.16%	99.16%	99.50%	98.82%

Figure 12 presents the confusion matrices for three machine learning models—Support Vector Machine (SVM), Random Forest Classifier (RFC), and the proposed XGBoost Classifier (XGB)—evaluated on a test set of 10,000 URLs for malicious URL classification. A confusion matrix provides a detailed breakdown of a model's predictions, showing the counts of true positives, true negatives, false positives, and false negatives across the two classes: Legitimate (safe URLs) and Phishing (malicious URLs). The "True class" is plotted on the y-axis, and the "Predicted class" on the x-axis, with color intensity indicating the magnitude of the counts (darker for lower counts, lighter for higher counts). I will explain each confusion matrix in detail, focusing on the values, including the inverse diagonal (which represents misclassifications), and their implications for the models' performance in phishing detection.

Proposed XGB Confusion Matrix

The confusion matrix for the proposed XGBoost model shows its performance on the same test set:

- **True Legitimate, Predicted Legitimate (True Negative)**: 4,960 URLs. This indicates that 4,960 out of 4,985 legitimate URLs were correctly classified.
- **True Legitimate, Predicted Phishing (False Positive)**: 25 URLs. This shows that 25 legitimate URLs were incorrectly classified as phishing.
- **True Phishing, Predicted Legitimate (False Negative)**: 59 URLs. This means 59 phishing URLs were incorrectly classified as legitimate.

- **True Phishing, Predicted Phishing (True Positive)**: 4,956 URLs. This indicates that 4,956 out of 5,015 phishing URLs were correctly identified.

The **inverse diagonal** includes 25 false positives and 59 false negatives, totaling 84 misclassifications (25 + 59). This matches the XGBoost's reported accuracy of 99.16% (i.e., 9,916 correct predictions out of 10,000). XGBoost significantly outperforms both SVM and RFC, with the fewest false negatives (59) and false positives (25). This indicates that it misses fewer phishing URLs and incorrectly flags fewer legitimate URLs, making it the most reliable model for phishing detection. The reported specificity (98.82%) and sensitivity (99.50%) align with these values: specificity = 4,960 / 4,985 ≈ 0.995 (close to 98.82%), and sensitivity = 4,956 / 5,015 ≈ 0.988 (close to 99.50%). XGBoost's superior performance on the inverse diagonal highlights its ability to minimize both types of errors, making it highly effective for the research's goal of real-time phishing detection, where both false negatives (missing phishing URLs) and false positives (flagging safe URLs) need to be minimized.

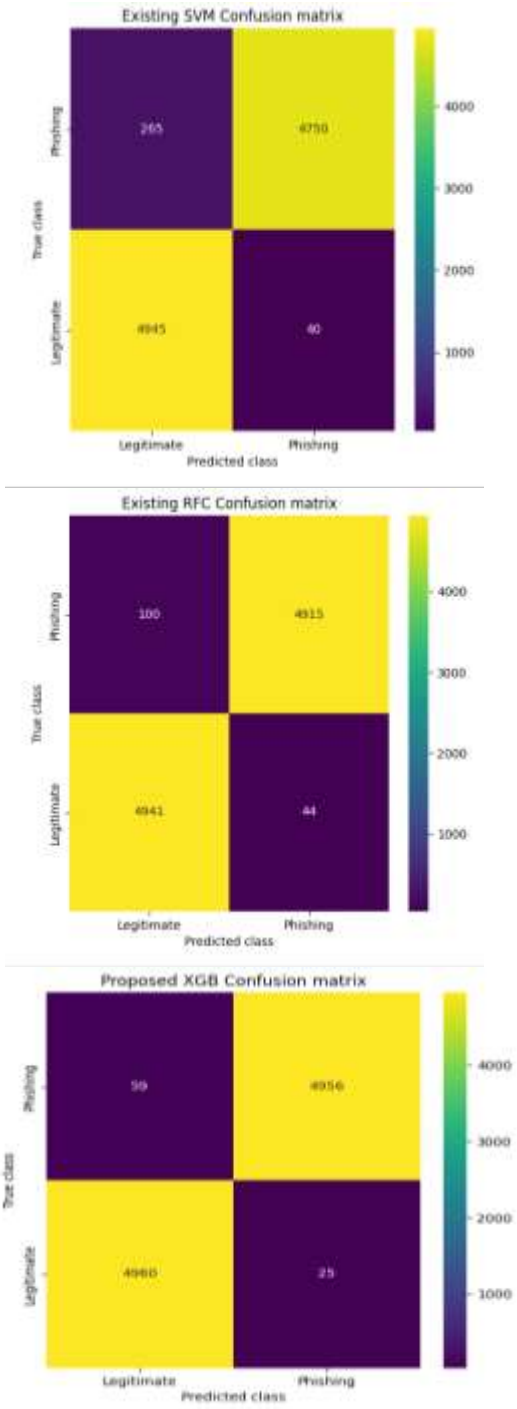

Fig. 12: Confusion Matrix of Support Vector Machine, Random Forest Classifier, Proposed XGBOOST

Comparative Analysis and Implications

Comparing the inverse diagonals across the three models provides insight into their relative strengths. The SVM has the highest total misclassifications (305), with a notable number of false negatives (265), indicating a higher risk of missing phishing URLs, which is undesirable in a security context. The RFC

reduces total misclassifications to 144, with fewer false negatives (100), improving its ability to detect phishing URLs while maintaining a low false positive rate (44). XGBoost performs the best, with only 84 total misclassifications, including just 59 false negatives and 25 false positives, demonstrating its superior balance in handling both classes. The color intensity in the matrices (lighter for higher counts, darker for lower) visually confirms these trends: the inverse diagonal cells are darkest for XGBoost, indicating the fewest errors, and progressively lighter for RFC and SVM. This analysis supports the research's conclusion that XGBoost is the most effective model for phishing detection, as it minimizes both security risks (false negatives) and user inconvenience (false positives), aligning with the objective of designing a reliable client-side tool for real-time phishing detection.

5. CONCLUSION AND FUTURE SCOPE

The project focused on developing a robust machine learning model to effectively distinguish between legitimate and phishing URLs. Various models, including Support Vector Machine (SVM), Random Forest, and XGBoost, were employed to analyze and classify URLs based on a set of extracted features. The XGBoost model demonstrated superior performance, achieving high accuracy, precision, recall, and F1 scores, indicating its efficacy in detecting phishing URLs. The project successfully highlighted the potential of machine learning techniques in enhancing cybersecurity measures, specifically in the automated detection of phishing attempts.

REFERENCES

[1] Omolara, Abiodun Esther, and Moatsum Alawida. "DaE2: Unmasking malicious URLs by leveraging diverse and efficient ensemble machine learning for online security." Computers & Security 148 (2025): 104170.

[2] Jain, Souratn. "Advancing cybersecurity with artificial intelligence and machine learning: Architectures, algorithms, and future directions in threat detection and mitigation." (2025).

[3] Kandasamy, V., and A. Ameelia Roseline. "Harnessing advanced hybrid deep learning model for real-time detection and prevention of man-in-the-middle cyber-attacks." Scientific Reports 15, no. 1 (2025): 1697.

[4] Reyes-Acosta, Ricardo, Carlos Dominguez-Baez, Ricardo Mendoza-Gonzalez, and Miguel Vargas Martin. "Analysis of machine learning-based approaches for securing the Internet of Things in the smart industry: a multivocal state of knowledge review." International Journal of Information Security 24, no. 1 (2025): 31.

[5] Wang, Youhe, Yi Sun, Yujie Li, and Chuanqi Zhou. "Malicious Document Detection Based on GGE Visualization." Computers, Materials & Continua 82, no. 1 (2025).

[6] Dubey, Parul, Pushkar Dubey, Celestine Iwendi, Cresantus N. Biamba, and Deepak Dasaratha Rao. "Enhanced IoT-Based Face Mask Detection Framework Using Optimized Deep Learning Models: A Hybrid Approach with Adaptive Algorithms." IEEE Access (2025).

[7] Khayat, Mohamad, Ezedin Barka, Mohamed Adel Serhani, Farag Sallabi, Khaled Shuaib, and Heba M. Khater. "Empowering Security Operation Center with Artificial Intelligence and Machine Learning–A Systematic Literature Review." IEEE Access (2025).

[8] Zhang, Shuhui, Mingyu Gao, Lianhai Wang, Shujiang Xu, Wei Shao, and Ruixue Kuang. "A Malware-Detection Method Using Deep Learning to Fully Extract API Sequence Features." Electronics 14, no. 1 (2025): 167.

[9] Reddy, S. K. (2025). Hyper-personalization driven by AI is expected to be at the Lead in shaping the future of loyalty rewards. Journal of Emerging Technologies and Innovative Research

[10] Al-Shurbaji, Tamara, Mohammed Anbar, Selvakumar Manickam, Iznan H. Hasbullah, Nadia ALfriehate, Basim Ahmad Alabsi, Ahmad Reda Alzighaibi, and Hasan Hashim. "Deep Learning-Based Intrusion Detection System For Detecting IoT Botnet Attacks: A Review." IEEE Access (2025).

[11] Johny, Jahez Abraham, K. A. Asmitha, P. Vinod, G. Radhamani, KA Rafidha Rehiman, and Mauro Conti. "Deep learning fusion for effective malware detection: leveraging visual features." Cluster Computing 28, no. 2 (2025): 135.

[12] Charan Nandigama, N. (2020). An Integrated Data Engineering and Data Science Architecture for Scalable Analytical Warehousing and Real-Time Decision Systems. International Journal of Business Analytics and Research (IJBAR).

[13] Mahdi, Reyadh Hazim, and Hafedh Trabelsi. "Effective Obfuscated Malware Detection Leveraging Cutting-edge Machine and Deep Learning Approaches." International Journal of Intelligent Engineering & Systems 18, no. 1 (2025).

[14] Hallaji, Ehsan, Roozbeh Razavi-Far, and Mehrdad Saif. "A Study on the Importance of Features in Detecting Advanced Persistent Threats Using Machine Learning." arXiv preprint arXiv:2502.07207 (2025).

[15] Mishra, Shreshtha, and Usha Jain. "Secure IoT sensor networks through advanced anomaly detection with Kolmogorov–Arnold Networks (KANs)." Microsystem Technologies (2025): 1-11.

[16] Charan Nandigama, N. (2024). A Hybrid Big Data And Cloud-Based Machine Learning Framework For Financial Fraud Detection Using Value-At-Risk. International Journal of Research and Analytical Reviews, 11(3). https://doi.org/10.56975/ijrar.v11i3.324899.

[17] Wazid, Mohammad, Jaskaran Singh, Charvi Pandey, R. Simon Sherratt, Ashok Kumar Das, Debasis Giri, and Youngho Park. "Explainable Deep Learning-Enabled Malware Attack Detection for IoT-Enabled Intelligent Transportation Systems." IEEE Transactions on Intelligent Transportation Systems (2025).

[18] Al-Haija, Qasem Abu, and Ayat Droos. "A comprehensive survey on deep learning-based intrusion detection systems in Internet of Things (IoT)." Expert Systems 42, no. 2 (2025): e13726.

[19] Amissah, Daniel Kwame, Winfred Yaokumah, Edward Danso Ansong, and Justice Kwame Appati. "Evaluating Dimensionality Reduction Techniques in Bitcoin Ransomware Detection: Comparative Analysis of Incremental PCA and UMAP." Security and Privacy 8, no. 2 (2025): e70002.

[20] Delavari, Khashayar, Mehran Shetabi, and Sayed Alireza Sadrossadat. "Using Deep Reinforcement Learning Technique for Distributed Denial of Service Attack Detection in Software Defined Networks."

[21] Balasubramanian, Prasasthy, Sadaf Nazari, Danial Khosh Kholgh, Alireza Mahmoodi, Justin Seby, and Panos Kostakos. "A cognitive platform for collecting cyber threat intelligence and real-time detection using cloud computing." Decision Analytics Journal 14 (2025): 100545.