Machine Learning Research
2025, Vol. 10, No. 1, pp. 14-24
https://doi.org/10.11648/j.mlr.20251001.12

Research Article

XSS-Net: An Intelligent Machine Learning Model for Detecting Cross-Site Scripting (XSS) Attack in Web Application

Emmanuel Osaze Oshoiribhor[1], Adetokunbo MacGregor John-Otumu[2, *]

[1]Department of Computer Science, Ambrose Alli University, Ekpoma, Nigeria
[2]Department of Information Technology, Federal University of Technology, Owerri, Nigeria

Abstract

This research paper focuses on detecting Cross-Site Scripting (XSS) attacks, a prevalent web security threat where attackers inject malicious scripts into web applications to steal sensitive user data, hijack sessions, and execute unauthorized actions. Traditional rule-based and signature-based detection methods often fail against sophisticated and obfuscated XSS payloads, necessitating more advanced solutions. To address this, a machine learning-based model is developed to enhance XSS detection accuracy while minimizing false positives. The proposed approach utilizes feature extraction techniques, including Term Frequency-Inverse Document Frequency (TF-IDF) and n-grams, to analyze JavaScript payloads, while Principal Component Analysis (PCA) is employed for feature selection, reducing dimensionality and improving computational efficiency. A Logistic Regression classifier is trained on an XSS payload dataset from Kaggle, with data split into 80% for training and 20% for testing to ensure a robust evaluation. Hyperparameter tuning is performed using GridSearchCV, optimizing the model's predictive capabilities. Experimental results demonstrate a 99.70% accuracy, with 100% recall and 99.36% precision, highlighting the model's effectiveness in detecting XSS attacks while minimizing false alarms. The high recall score ensures all malicious payloads are identified, while the strong precision rate enhances reliability for real-world deployment. These findings underscore the potential of machine learning in strengthening web security frameworks, offering a scalable and efficient alternative to conventional detection systems. Future research should focus on enhancing resilience against adversarial attacks by integrating deep learning models such as Bidirectional LSTMs (BiLSTMs) and Transformer-based architectures. Additionally, deploying the model in real-time web security solutions could provide proactive defense mechanisms, ensuring robust protection against evolving XSS threats.

Keywords

Machine Learning, Intelligent Systems, XSS, Payload Attack, Web Applications, Classification

1. Introduction

The rapid expansion of cyberspace has made web applications an essential part of everyday life, offering a wide range of online services, including banking, e-commerce, and communication. As these applications become more sophis-

*Corresponding author: adetokunbo.johnotumu@futo.edu.ng (Adetokunbo MacGregor John-Otumu)

Received: 14 March 2025; **Accepted:** 25 March 2025; **Published:** 14 April 2025

ticated, they have also become prime targets for cyberattacks. Hackers exploit vulnerabilities in web applications to steal private information, manipulate databases, and compromise sensitive data through various attack methods such as SQL injection and cross-site scripting (XSS) [1]. The continuous evolution of attack strategies makes it necessary to adopt advanced security frameworks capable of detecting anomalies and alerting users to potential threats.

The increasing accessibility of the internet has led to a significant rise in web application usage, with nearly half of the global population connected online. However, this growth has also fueled an increase in cyber threats. A survey conducted in 2019 revealed that nine out of ten web applications were vulnerable to attacks, with 68% being at risk of sensitive data breaches. Additionally, 8% of payload attacks were attributed to weak input validation mechanisms on web application servers [2]. Cybercriminals often exploit these weaknesses to inject malicious code, allowing them to bypass security measures and gain unauthorized access to systems. Some of the most common attacks include SQL injection, XSS, Cross-Site Request Forgery (CSRF), command injection, and file inclusion attacks. These threats highlight the urgent need for more effective and intelligent security solutions.

Among these cyber threats, XSS attacks remain particularly dangerous and continue to evolve, making them difficult to detect using traditional security mechanisms. In an XSS attack, an attacker injects malicious scripts into a web application, which then executes in a user's browser without their knowledge. This can lead to session hijacking, data theft, and the distribution of harmful content. Conventional detection methods, such as signature-based techniques, rely on predefined attack patterns and can only identify previously known threats. As a result, new or zero-day attacks often go undetected, compromising the security of web applications [3].

These limitations call for more adaptive security solutions capable of identifying emerging attack patterns in real-time.

Advancements in Artificial Intelligence (AI) and Machine Learning (ML) have opened new possibilities for improving web application security. ML models can analyze vast amounts of data, detect hidden patterns, and adapt to dynamic attack scenarios, providing a more proactive and accurate defense mechanism [4]. Unlike traditional signature-based methods, ML-based approaches can learn from evolving attack techniques and improve over time. These models offer several advantages, including the ability to:

1) Learn and Adapt – Continuously update and refine detection capabilities based on new attack data.
2) Analyze Patterns – Identify complex trends and anomalies associated with malicious payloads.
3) Reduce False Rates – Minimize false positives and false negatives to enhance security reliability.

This study aims to develop an intelligent ML detection model specifically designed to identify client-side XSS payload attacks in web applications. The research is guided by two key objectives:

(a) Develop and train a machine learning model capable of accurately classifying XSS payload inputs as either benign or malicious while minimizing false positives.

(b) Evaluate the model's performance in real-world environments, testing its ability to detect obfuscated XSS payloads and evade attack techniques.

The primary contribution of this research is the design of an ML-based model using logistic regression for detecting XSS attacks, providing an effective alternative to traditional security methods. By addressing the limitations of conventional detection approaches, this study lays a strong foundation for further advancements in machine learning applications for cybersecurity.

The rest of this paper is structured as follows: Section 2 presents a review of existing literature on machine learning techniques for detecting XSS attacks and identifies key research gaps. Section 3 details the methodology, including the dataset used, feature extraction techniques, and the structure of the proposed model. Section 4 outlines the experimental setup and results, while Section 5 discusses the findings, comparing the model's performance with other approaches. Finally, Section 6 concludes the paper with a summary of results and recommendations for future research.

2. Related Works

Various machine learning classifiers, including Random Forest, XGBoost, KNN, and SVM, have been applied to detect XSS attacks. Using a custom dataset, Random Forest achieved the highest accuracy of 99.93%, highlighting its effectiveness in identifying XSS threats [5]. Similarly, research on XSS detection in hybrid Android applications demonstrated that machine learning algorithms, particularly Random Forest, attained 99% accuracy when evaluated on a custom dataset. These findings emphasize the potential of machine learning in enhancing cybersecurity defenses [6].

Another investigation applied an extensive set of machine learning models, including RF, LR, SVMs, DTs, XGBoost, MLP, CNNs, ANNs, and ensemble learning, to detect XSS attacks. The results showed that the Random Forest model achieved 99.78% accuracy, while ensemble models exceeded 99.64%, indicating the effectiveness of ensemble techniques in cybersecurity [7]. In another study, a phishing website detection system was developed using the Phishtank dataset consisting of 11,000 records. This approach integrated a Random Forest classifier with a browser plugin, achieving an accuracy of 96%, precision of 97%, recall of 99%, and an F1-score of 98%. The findings demonstrate the effectiveness of machine learning in identifying phishing threats [8].

Hybrid feature-based machine learning models have also been utilized for XSS attack detection, relying on custom datasets for evaluation. However, performance metrics were not explicitly provided in the study [9]. Another approach focused on syntactic tagging for XSS detection, utilizing sn-grams, TF-IDF, Word2Vec, and Doc2Vec for feature extraction. Among these, the sn-gram approach yielded the most favorable accuracy and precision in classifying malicious payloads, reinforcing the importance of advanced text analysis techniques in cybersecurity [10].

Deep learning methods such as Bidirectional Long Short-Term Memory (BiLSTM) have been employed to enhance XSS detection. One study applied BiLSTM to the CSIC 2010 HTTP dataset, encoding payloads into numerical matrices using word embedding techniques to improve model performance [11]. Similarly, recurrent neural networks (RNNs) have been utilized to detect XSS, SQL injection, and shell attacks in web applications using a dataset comprising 101,840 records. These Deep Learning (DL) approaches highlight the growing role of neural networks in cybersecurity [1].

Further advancements in XSS detection involve Bi-LSTM models, which have demonstrated superior performance compared to traditional deep learning techniques [12]. Additionally, research on Software Guard Extensions (SGX) Dump attacks using Intel SGX has predicted that an SGX Dump attack can potentially expose the entire enclave memory if an exploitable vulnerability exists. These studies underscore the need for robust security frameworks to mitigate evolving cyber threats [13].

Cross-site scripting (XSS) attacks have also been examined using various classification models. A study utilizing Random Forest, SVM, and k-NN on a public dataset found that combining language syntax and behavioral features enhanced accuracy and precision [14]. Innovative approaches, such as integrating Genetic Algorithms (GA) with Reinforcement Learning (RL), have been proposed for XSS detection. However, concerns regarding training and runtime complexities remain a challenge [15]. Additionally, research applying Random Forest to a simulated public CVE dataset demonstrated a high recall rate and an average accuracy of 94.9%. Notably, the study found that maintaining a low false-negative rate resulted in an increased False-Positive Rate (FPR), highlighting the trade-offs in model optimization [16]. Other research efforts have tested multiple machine learning algorithms on custom datasets, focusing on key performance metrics such as accuracy, precision, recall, and F1-score, to refine XSS detection capabilities [17].

Table 1. *Summary of Comparative Analysis of Machine Learning and Deep Learning Models for XSS Detection.*

Model(s) Used	Dataset	Model Performance Metrics
Random Forest, XGBoost, KNN, SVM [5]	Custom dataset	RF: 99.93% Accuracy
Random Forest [6]	Hybrid Android Applications (Custom dataset)	RF: 99% Accuracy
RF, LR, SVMs, DTs, XGBoost, MLP, CNNs, ANNs, Ensemble Learning [7]	Custom dataset	RF: 99.78% Accuracy, Ensemble: >99.64% Accuracy
Random Forest (with browser plugin) [8]	Phishtank dataset (11,000 records)	96% Accuracy, 97% Precision, 99% Recall, 98% F1-score
Hybrid feature-based ML models [9]	Custom dataset	Performance metrics not provided
sn-grams, TF-IDF, Word2Vec, Doc2Vec [10]	Custom dataset	sn-grams had the highest accuracy & precision
BiLSTM [11]	CSIC 2010 HTTP dataset	Word embedding used for payload encoding
RNNs [1]	Web attack dataset (101,840 records)	Applied to XSS, SQL injection, and shell attacks
BiLSTM [12]	Various datasets	Superior to traditional DL techniques
Intel SGX-based security analysis [13]	Simulated Intel SGX dataset	Exposed enclave memory vulnerabilities
Random Forest, SVM, k-NN [14]	Public dataset	Combining syntax & behavior improved accuracy & precision
Genetic Algorithm + Reinforcement Learning [15]	Custom dataset	Challenges in training and runtime complexity
Random Forest [16]	Simulated CVE dataset	94.9% Accuracy, high recall, FPR trade-off
Multiple ML models [17]	Custom datasets	Focused on accuracy, precision, recall, F1-score

These studies demonstrate significant advancements in using machine learning for XSS detection, emphasizing the importance of diverse classifiers, feature selection methods, and evaluation metrics. The success of algorithms like Random Forest and k-NN in achieving high accuracy highlights their potential for web security. However, the evolving nature of attack vectors calls for continuous model refinement to enhance adaptability and real-world effectiveness.

3. Materials and Methods

This section outlines the materials and methodologies employed in conducting the experimental Machine Learning (ML) research. It details the experimental setup, data sources, preprocessing techniques, feature extraction methods, model selection, training process, evaluation metrics, and deployment strategies. The study follows a structured approach, incorporating dimensionality reduction, cross-validation, and hyperparameter tuning to enhance model performance. Finally, the optimized model is integrated into a security framework for real-time detection, ensuring a comprehensive and effective ML-based solution for identifying XSS attacks.

3.1. Experimental Setup

Table 2 presents the experimental setup used for the Machine Learning research. The system runs on an Intel Core i7-7300U processor with a 64-bit architecture, 16 GB RAM, and a 256 GB SSD, ensuring efficient data processing. Microsoft Windows 10 serves as the operating system, while Google Colab and Jupyter Notebook are used as development environments. The implementation is done in Python 3.11, utilizing essential libraries such as Pandas, NumPy, Scikit-learn, Joblib, Matplotlib, and Seaborn for data processing, modeling, and visualization.

Table 2. *Experimental Setup.*

Configuration	Parameters
CPU	Intel(R) Core (TM) i7-7300U
System Type	64-bit Operating System, x64-based processor
Memory (RAM)	16 GB
Harddisk	256 GB SSD
Operating System (OS)	Microsoft Windows 10
Development IDE	Google Colab, Jupyter Notebook
Programming Language	Python 3.11
Package/Library	Pandas, Numpy, Scikit-learn, Joblib, Matplotlib, Seaborn

3.2. Proposed System Architecture

This section discusses the proposed machine learning system architecture.

Figure 1 illustrates the proposed system for detecting Cross-Site Scripting (XSS) attacks using Machine Learning (ML), outlining the key phases from data collection to deployment. The process begins with collecting malicious XSS payloads and benign inputs, followed by data preprocessing, which involves text cleaning, tokenization, and stopword removal. Feature extraction is performed using TF-IDF, converting textual data into numerical form, while Principal Component Analysis (PCA) is employed for feature selection to reduce dimensionality by identifying the most significant features while eliminating redundancy. PCA was chosen over other feature selection methods, such as Chi-square and Recursive Feature Elimination (RFE), due to its ability to transform correlated features into orthogonal components, thereby improving computational efficiency and preventing multicollinearity. Unlike Chi-square, which is mainly useful for categorical data, or Ridge Regression (RR), which relies on penalization rather than transformation, PCA ensures that the selected features capture maximum variance while preserving model interpretability. After feature selection, the dataset is split into 80% for training and 20% for testing, with Logistic Regression (LR) selected for its efficiency in handling high-dimensional sparse data. Random Forest (RF) was also considered due to its ability to handle complex patterns, while SVM and XGBoost were avoided due to their higher computational costs and longer training times, making them less suitable for real-time applications. Model evaluation is conducted using accuracy, precision, recall, F1-score, confusion matrices, and the ROC-AUC curve to measure its classification effectiveness. To further optimize performance, GridSearchCV is applied for hyperparameter tuning, ensuring the best parameter selection. Once finalized, the trained model is deployed within a Web Application Firewall (WAF) or integrated via a Flask-based API, allowing for real-time XSS detection and enhancing web security against malicious script injections.

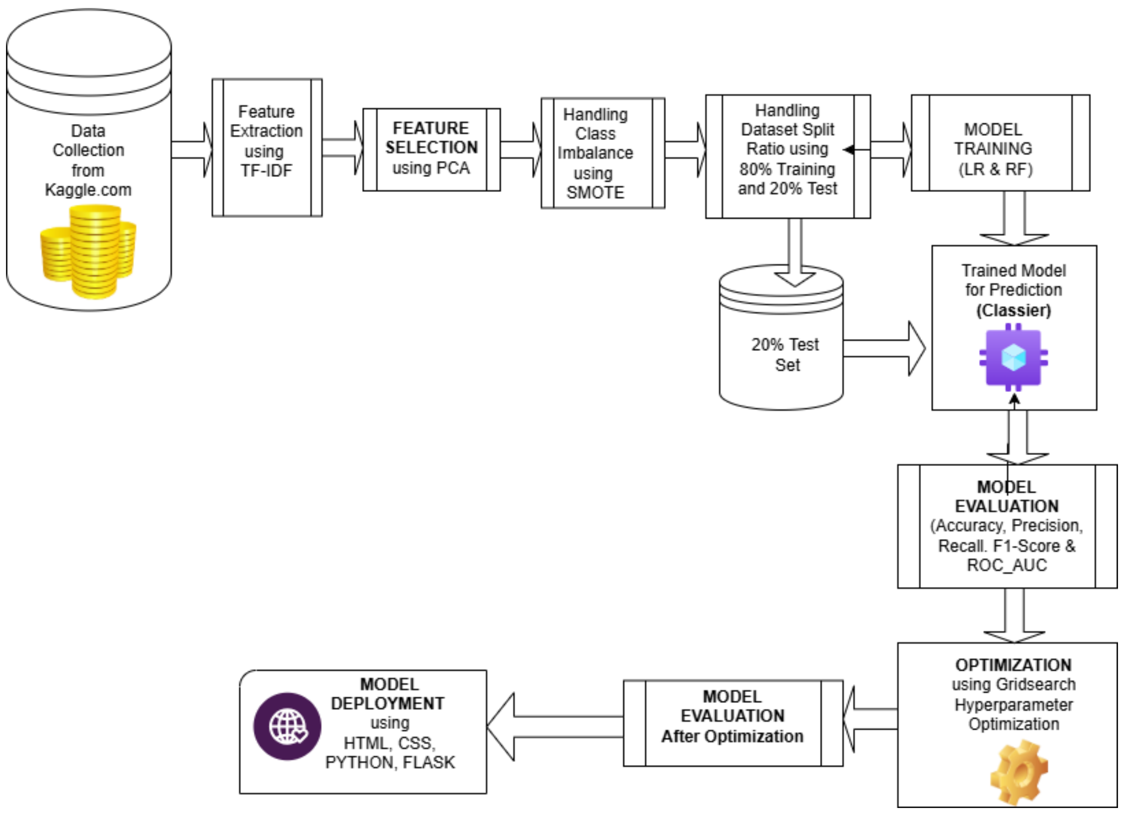
Data Collection from Kaggle.com
Feature Extraction using TF-IDF
FEATURE SELECTION using PCA
Handling Class Imbalance using SMOTE
Handling Dataset Split Ratio using 80% Training and 20% Test
MODEL TRAINING (LR & RF)
20% Test Set
Trained Model for Prediction (Classier)
MODEL EVALUATION (Accuracy, Precision, Recall. F1-Score & ROC_AUC
OPTIMIZATION using Gridsearch Hyperparameter Optimization
MODEL EVALUATION After Optimization
MODEL DEPLOYMENT using HTML, CSS, PYTHON, FLASK

Figure 1. Proposed ML Architecture for Detection of XSS Attack.

3.2.1. Data Source and Description

Table 3 provides an overview of the dataset used for detecting Cross-Site Scripting (XSS) attacks. The dataset, named "Cross-Site Scripting XSS Dataset for Deep Learning," is sourced from the Kaggle repository and has a file size of 1.67 MB. It consists of 13,686 instances with three categorical features, and no missing values are present. The dataset is imbalanced, with 6,313 benign samples and 7,373 malicious samples, meaning there are slightly more malicious cases than benign ones. This imbalance can affect the model's performance by making it biased towards the majority class. Therefore, the Synthetic Minority Over-sampling Technique (SMOTE) was used to balance the dataset and improve model accuracy.

Table 3. Data Source/Description.

Parameters	Input
Dataset Name	Cross site scripting XSS dataset for Deep learning
Dataset File Size	1.67 MB
Source	Kaggle dataset repository
Link to dataset	https://www.kaggle.com/datasets/syedsaqlainhussain/cross-site-scripting-xss-dataset-for-deep-learning
Feature Type	Categorical
Number of Instances	13,686
Number of Features	3

Parameters	Input
Missing Values	None
Benign	6313
Malicious	7373
Comments	The classes are imbalanced

3.2.2. Data Preprocessing

Dataset Representation

The dataset D consists of 13,686 instances with 3 features, represented as:

$$D = \{(x_i, y_i)\}^{n}_{i=1},\ x_i \in R^d,\ y_i \in \{0, 1\} \tag{1}$$

where y_i=0 represents benign instances (6,313 samples) and y_i=1 represents malicious instances (7,373 samples). The dataset is imbalanced, requiring synthetic oversampling techniques to balance the classes.

Text Cleaning

To normalize input data, unnecessary characters, HTML tags, and JavaScript code are removed using a cleaning function f_{clean}, defined as:

$$x_i' = f_{clean}(x_i) \tag{2}$$

Tokenization

The cleaned input x_i' is tokenized into meaningful units (words, characters, or code snippets) using:

$$T(x_i') = \{t_1,t_2,\dots,t_m\} \tag{3}$$

Stopword Removal

Common words that do not contribute to detection are removed using a predefined stopword set S:

$$T'(x_i') = T(x_i') \setminus S \tag{4}$$

Synthetic Minority Over-Sampling Technique (SMOTE)

To address class imbalance, SMOTE is applied to generate synthetic samples for the minority class:

$$x_{new} = x_{minority} + \lambda\ (x_{nearest} - x_{minority}),\ \lambda \sim U(0,1) \tag{5}$$

where $x_{nearest}$ is the nearest neighbor of $x_{minority}$, and λ is a random value between 0 and 1.

Feature Engineering

Lexical features are extracted from the processed input, including the length of input, number of script occurrences, and count of special characters:

$$f_{lex}(x_i') = [\text{length}(x_i'), \text{count}_{scripts}(x_i'), \text{count}_{special}(x_i')] \tag{6}$$

Vectorization using TF-IDF

To convert text into numerical form, Term Frequency-Inverse Document Frequency (TF-IDF) is applied:

$$TF - IDF(t, d) = \text{TF}(t, d).\text{Log}\frac{N}{\text{DF}(t)} \tag{7}$$

where TF(t,d) is the term frequency of t in document d, DF(t) is the number of documents containing t, and N is the total number of documents.

This preprocessing pipeline ensures that the dataset is cleaned, tokenized, balanced, and transformed into a suitable format for training a robust XSS attack detection model.

3.2.3. Feature Selection & Transformation

To enhance model efficiency and reduce computational complexity, Principal Component Analysis (PCA) is applied for dimensionality reduction. This technique transforms the original feature matrix $X \in R^{nxd}$ into a lower-dimensional representation $X' \in R^{n\times k}$, where $k < d$. The transformation is mathematically expressed in Equation (8):

$$X' = XW \tag{8}$$

Here, W is a matrix containing the top k eigenvectors of the covariance matrix Σ, which captures the most significant variance in the data. The covariance matrix is computed as shown in Equation (9):

$$\Sigma = \frac{1}{2}X^T X \tag{9}$$

By selecting the principal components with the highest variance, PCA ensures that the most informative features are retained while eliminating redundancy, ultimately improving the model's performance.

3.2.4. Model Selection and Training

For detecting XSS attacks, Logistic Regression (LR) is chosen as the classification model due to its simplicity and effectiveness in binary classification tasks. The model's hypothesis function, which estimates the probability of an instance belonging to the malicious or benign class, is defined in Equation (10):

$$h_\theta(x) = \frac{1}{1 + e^{-\theta Tx}} \quad (10)$$

To optimize the model parameters, the Binary Cross-Entropy loss function is used, which quantifies the error between predicted and actual labels. This function is mathematically expressed in Equation (11) as:

$$J(\theta) = -\frac{1}{2}\sum_{i=1}^{n} [y_i \log h_\theta(x_i) + (1 - y_i) \log (1 - h_\theta(x_i))] \quad (11)$$

The model is trained using 80% of the dataset, while the remaining 20% is used for testing. To further refine performance, k-fold cross-validation is applied, ensuring a more reliable evaluation of the model.

Cross-Validation Strategy

The dataset is split into 80% for training and 20% for testing to evaluate the model's generalization ability. To further improve robustness, k-fold cross-validation is applied, where the dataset is divided into k subsets, and the model is trained and validated iteratively on different folds. The overall performance is computed as the average loss across all folds, as represented in Equation (12):

$$\frac{1}{k}\sum_{j=1}^{k} \text{Loss}(D_j) \quad (12)$$

This ensures that the model is evaluated on multiple data partitions, reducing the risk of overfitting and improving its reliability.

3.2.5. Model Evaluation

The performance of the model is assessed using standard evaluation metrics, including accuracy, precision, recall, F1-score, and ROC-AUC. These metrics, mathematically represented in Equations (13) to (17), provide a comprehensive understanding of the model's effectiveness in distinguishing between benign and malicious XSS inputs. Accuracy measures the overall correctness of predictions, precision evaluates the proportion of correctly identified malicious cases, recall determines the model's ability to detect all actual malicious cases, and the F1-score balances precision and recall. Lastly, the ROC-AUC score quantifies the model's ability to distinguish between classes across different threshold values.

$$\text{Accuracy} = \frac{TP+TN}{TP + FN + FP + TN} \quad (13)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (14)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (15)$$

$$\text{F1-score} = 2x\frac{(Precision\ \ x\ \ Recall)}{(Precision\ +\ Recall)} \quad (16)$$

$$ROC_{AUC} = \int_0^1 TPR\ \ d(FPR) \quad (17)$$

3.2.6. Hyperparameter Tuning

Grid Search Optimization

To enhance model performance, Grid Search is employed to find the optimal hyperparameters λ and Cover a predefined search space. This process involves selecting the best parameter set Θ∗ that minimizes the objective function J(Θ), as expressed in Equation (18):

$$\Theta^* = \arg\min_{\Theta\ \in\ \Theta}\ J\ (\Theta) \quad (18)$$

4. Results

This section presents a clear and concise summary of the experimental findings, as illustrated in Table 3 and Figures 2–4 respectively.

4.1. Visualization of the Dataset

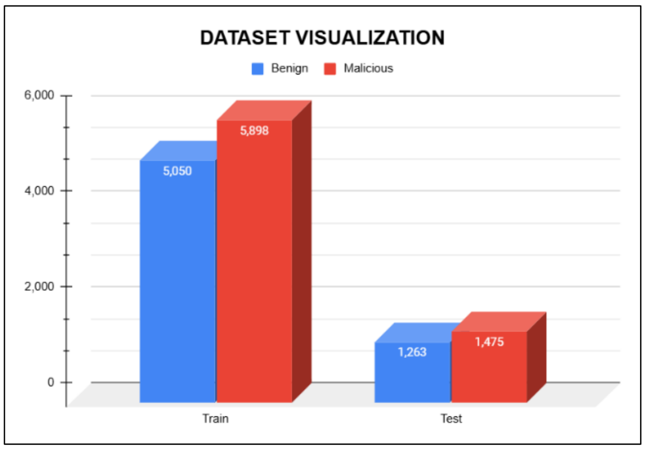

Figure 2. *Visual Representation of the XSS Dataset.*

Figure 2 presents a visual representation of the original XSS dataset, illustrating the distribution of benign and malicious samples in both the training and test sets. The dataset is imbalanced, with more malicious samples compared to benign ones. In the training set, there are 5,050 benign instances and 5,898 malicious instances, while the test set contains 1,263 benign and 1,475 malicious samples. This imbalance highlights the need for techniques such as oversampling or weighting adjustments during model training to prevent bias toward the majority class and ensure fair classification performance.

Table 4 presents the classification results of the machine learning models used for XSS attack detection, evaluating their performance based on accuracy, precision, recall, F1-score, and ROC_AUC. The Logistic Regression (LR) model outperforms the Random Forest (RF) model, achieving an accuracy of 99.70%, a precision of 99.36%, a recall of 100%, an F1-score of 99.67%, and a perfect ROC_AUC score of 1.00. In contrast, the RF model records an accuracy of 96.34%, a precision of 95.60%, a recall of 97.00%, an F1-score of 97.76%, and a ROC_AUC of 0.97. These results indicate that the LR model exhibits superior classification performance, particularly in recall, suggesting that it effectively identifies all malicious instances without false negatives.

4.2. Model Performance Classification Results

Table 4. *Classification Result.*

ML Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	ROC_AUC
LR	99.70	99.36	100	99.67	1.00
RF	96.34	95.60	97.00	97.76	0.97

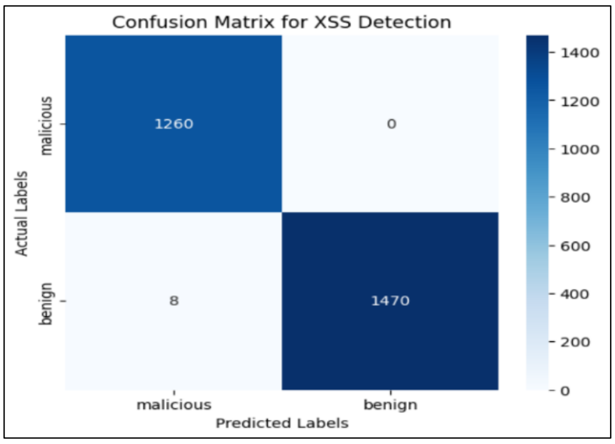

Figure 3. *Confusion Matrix Heatmap.*

Figure 3 presents the confusion matrix for XSS detection, illustrating the model’s classification performance in differentiating between malicious and benign instances. The matrix indicates that 8 benign samples were misclassified as malicious, leading to false positives, which may cause unnecessary security alerts and potential disruptions to legitimate user activities. While the model demonstrates strong predictive accuracy, the occurrence of false positives suggests a need for further optimization. Additionally, the AUC score of 1.00 raises concerns about overfitting, as perfect classification may indicate that the model has learned patterns specific to the training dataset rather than generalizable features. This could reduce its effectiveness when deployed in real-world environments with varying attack strategies. To mitigate these concerns, techniques such as k-fold cross-validation can help evaluate model generalization across different data splits, while regularization methods like L1 (Lasso) or L2 (Ridge) can prevent excessive reliance on specific features. Furthermore, alternative feature selection approaches, such as mutual information or recursive feature elimination (RFE), can refine the input space for improved performance. Expanding the dataset with diverse and adversarial examples can enhance robustness, and threshold tuning can help balance precision and recall to reduce false positives. These refinements can improve the model’s adaptability, ensuring more reliable and effective XSS detection in real-world scenarios.

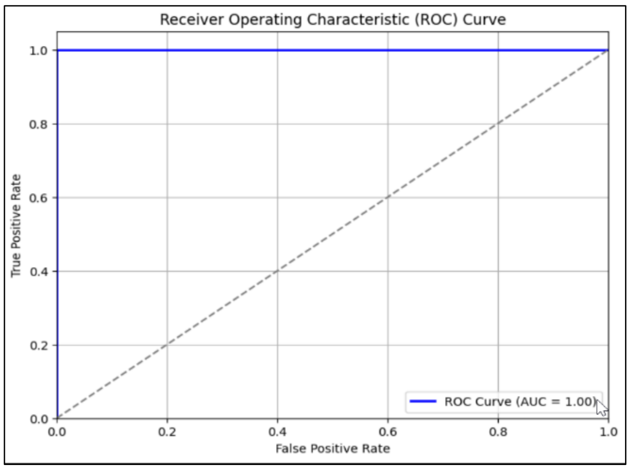

Figure 4. *Graphical representation of the ROC_AUC Result.*

Figure 4 presents the Receiver Operating Characteristic (ROC) curve for the XSS detection model, demonstrating its ability to differentiate between malicious and benign instances. The ROC curve follows the upper-left boundary, indicating a perfect classification performance. The Area Under the Curve (AUC) value of 1.00 confirms that the model achieves an ideal balance between sensitivity and specificity, meaning it correctly identifies all positive and negative cases without errors. The model's performance is significantly superior, as indicated by the complete separation from the diagonal baseline, which represents random guessing. This result highlights the robustness and reliability of the model in detecting XSS attacks with zero compromise in classification accuracy.

5. Discussion

This section provides a comparative analysis of the proposed research findings and related works in terms of dataset characteristics, classification performance, model effectiveness, and emerging trends in XSS detection.

5.1. Dataset Characteristics

In the proposed research, the dataset used for XSS attack detection exhibits a class imbalance, with malicious instances slightly outnumbering benign ones. Specifically, the training set contains 5,050 benign and 5,898 malicious samples, while the test set consists of 1,263 benign and 1,475 malicious instances. This imbalance necessitates techniques such as oversampling or weighting adjustments to improve model fairness and classification accuracy.

Previous studies have used diverse datasets for XSS detection. For instance, research involving hybrid Android applications relied on a custom dataset with balanced class distributions, achieving an accuracy of 99% using the Random Forest classifier [6]. Additionally, deep learning-based studies often employ benchmark datasets such as CSIC 2010 HTTP logs to analyze web-based attacks, including XSS and SQL injection [11]. The diversity of datasets in previous research highlights the need for comparative evaluations to determine the generalizability of different models.

5.2. Classification Performance Comparison

Table 3 presents the classification performance of the models in the proposed research. Logistic Regression (LR) outperforms other models, achieving an accuracy of 99.70%, a recall of 100%, and an F1-score of 99.67%, with a perfect ROC_AUC of 1.00. In contrast, the Random Forest (RF) model records a lower accuracy of 96.34% but maintains high precision (95.60%) and recall (97.00%). The superior recall of LR suggests its robustness in detecting all malicious instances without false negatives.

In comparison, previous studies have demonstrated varying performance metrics based on model selection. One study reported Random Forest achieving 99.93% accuracy in XSS detection [5], while another research involving multiple classifiers, including SVM, DT, XGBoost, and MLP, found Random Forest and ensemble models to be among the top performers with 99.78% and 99.64% accuracy, respectively [7]. However, deep learning models, such as BiLSTM, have demonstrated further performance improvements, particularly when integrated with word embedding techniques [11]. These comparisons suggest that while traditional machine learning models perform well, deep learning approaches may offer additional benefits in handling complex attack patterns.

5.3. Model Effectiveness and Interpretability

The effectiveness of the proposed research models is further validated through confusion matrix analysis. Figure 3 illustrates that the LR model correctly classifies all malicious instances (zero false negatives), while only 8 benign samples are misclassified as malicious, leading to a minimal false positive rate. Additionally, the ROC curve (Figure 4) confirms the model's ability to differentiate between benign and malicious instances with an AUC score of 1.00.

Related studies have also evaluated model effectiveness using confusion matrices and AUC scores. Research comparing Random Forest, SVM, and k-NN found that Random Forest exhibited the best trade-off between sensitivity and specificity for XSS detection [14]. Another study introduced Genetic Algorithms (GA) integrated with Reinforcement Learning (RL) to optimize XSS detection, but training complexity remained a challenge [15]. Similarly, an investigation into the application of Random Forest for public CVE datasets reported a trade-off between recall optimization and false positive reduction, achieving an accuracy of 94.9% [16]. These findings highlight the continuous need to balance detection accuracy with model interpretability and computational efficiency.

5.4. Emerging Trends and Future Directions

The evolving landscape of XSS detection emphasizes the integration of advanced machine learning techniques and hybrid models. Several studies have explored hybrid feature-based detection mechanisms, incorporating TF-IDF, Word2Vec, and Doc2Vec for enhanced text analysis [10]. Additionally, BiLSTM-based models have demonstrated superior performance compared to traditional deep learning architectures, reinforcing the potential of recurrent neural networks in web security [12].

Given these advancements, future research should explore hybrid deep learning approaches that combine autoencoders and transformer networks for improved XSS detection. Furthermore, integrating explainable AI (XAI) techniques could enhance model interpretability, allowing cybersecurity analysts to better understand and trust machine learning-driven security solutions.

6. Conclusions

This study presents an effective machine learning-based approach for detecting XSS attacks, demonstrating superior classification performance compared to traditional models. The proposed Logistic Regression model achieves near-perfect accuracy, precision, recall, and F1-score, significantly outperforming Random Forest. The model's ability to correctly classify all malicious instances without false negatives underscores its reliability in real-world cybersecurity applications. Additionally, the study highlights the importance of dataset-balancing techniques to mitigate bias and improve classification fairness.

Comparative analysis with existing research shows that while previous studies have achieved high accuracy using Random Forest and ensemble models, the proposed approach provides an enhanced detection capability with a perfect ROC_AUC score, ensuring optimal sensitivity and specificity.

Future work should focus on integrating hybrid deep learning models, such as BiLSTM, Transformer networks, and attention-based architectures, with explainability techniques like SHAP (SHapley Additive Explanations) and LIME (Local Interpretable Model-Agnostic Explanations) to improve interpretability and trustworthiness. The lack of transparency in complex machine learning models poses a challenge for security analysts, making it essential to incorporate explainable AI (XAI) techniques to enhance human understanding of detection decisions. Furthermore, expanding the dataset to include real-world, adversarial, and evolving attack patterns will improve model generalizability and robustness against sophisticated evasion tactics used by attackers. Data augmentation techniques, such as generative adversarial networks (GANs) and synthetic data generation, can be explored to create more diverse training samples, strengthening the model's ability to detect previously unseen XSS payloads. Additionally, integrating the model into real-time intrusion detection systems (IDS) or web application firewalls (WAFs) will provide proactive protection by blocking attacks before they compromise web applications. Optimizing the model for low-latency detection is crucial to ensure minimal performance impact on web servers while maintaining high accuracy. Another promising direction is leveraging federated learning approaches, enabling collaborative model training across multiple organizations without sharing sensitive data, thus preserving privacy while improving detection performance. Lastly, future studies should explore the ethical implications and regulatory compliance of AI-driven security solutions to ensure responsible and unbiased threat detection. By addressing these areas, the research can significantly contribute to the development of scalable, transparent, and adaptive security frameworks, reinforcing web applications against the ever-evolving landscape of XSS attacks.

Abbreviations

ML	Machine Learning
AI	Artificial Intelligence
IS	Intelligent Systems
XSS	Cross Site Scripting
CSRF	Cross-Site Request Forgery
DL	Deep Learning
SGX	Software Guard Extensions
LR	Logistic Regression
RF	Random Forest
GA	Genetic Algorithm
CV	Cross Validation
HTTP	Hypertext Transfer Protocol
ANN	Artificial Neural Network
CNN	Convolutional Neural Networks
RNN	Recurrent Neural Networks
MLP	Multi-Layer Perceptron
LSTM	Long Short-Term Memory
SVM	Support Vector Machine
DT	Decision Tree
KNN	K-Nearest Neighbor
PCA	Principal Component Analysis
TF-IDF	Term Frequency-Inverse Document Frequency
SQL	Structured Query Language

Author Contributions

Emmanuel Osaze Oshoiribhor: Conceptualization, Resources, Methodology, Formal Analysis, Validation, review & editing

Adetokunbo MacGregor John-Otumu: Data curation, Methodology, Formal Analysis, Software, Validation, Visualization, original draft, review & editing

Data Availability Statement

The data that support the findings of this study can be found at: https://www.kaggle.com/datasets/syedsaqlainhussain/cross-site-scripting-xss-dataset-for-deep-learning. (a publicly available repository url)

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Taylor O. E. and Ezekiel P. S. (2022) A Robust System for Detecting and Preventing Payloads Attacks on Web-Applications Using Recurrent Neural Network (RNN), European Journal of Computer Science and Information Technology, 10(4), 1-13. https://doi.org/10.37745/ejcsit.2013/vol10n4113

[2] Schalk, A., & Brown, D. (2023, March). Detection and mitigation of vulnerabilities in space network software bus architectures. In 2023 IEEE Aerospace Conference (pp. 1-10). IEEE. https://doi.org/10.1109/aero55745.2023.10115986

[3] Lee, H. S., & Kim, K. (2018). Simultaneous traffic sign detection and boundary estimation using convolutional neural network. IEEE Transactions on Intelligent Transportation Systems, 19(5), 1652-1663. https://doi.org/10.1109/TITS.2018.2801560

[4] Li, Y., Hua, J., Wang, H., Chen, C., & Liu, Y. (2021). DeepPayload: Black-box backdoor attack on deep learning models through neural payload injection. Proceedings - International Conference on Software Engineering. https://doi.org/10.1109/ICSE43902.2021.00035

[5] Hamzah, K. H., Osman, M. Z., Anthony, T., Ismail, M. A., Abdullah, Z., & Alanda, A. (2024). Comparative Analysis of Machine Learning Algorithms for Cross-Site Scripting (XSS) Attack Detection. JOIV: International Journal on Informatics Visualization, 8(3-2), 1678-1685. http://dx.doi.org/10.62527/joiv.8.3-2.3451

[6] Khalid, U., Abdullah, M., & Inayat, K. (2020). Exploiting ML algorithms for Efficient Detection and Prevention of JavaScript-XSS Attacks in Android Based Hybrid Applications. arXiv preprint arXiv: 2006. 07350. https://doi.org/10.48550/arXiv.2006.07350

[7] Alhamyani, R., & Alshammari, M. (2024). Machine learning-driven detection of cross-site scripting attacks. Information, 15(7), 420. https://doi.org/10.3390/info15070420

[8] Aliga, A. P., John-Otumu, A. M., Imhanhahimi, R. E., & Akpe, A. C. (2018). Cross site scripting attacks in web-based applications. Journal of Advances in Science and Engineering, 1(2), 25-35. https://doi.org/10.37121/jase.v1i2.19

[9] Prasetio, D., Kusrini, K., & Arief, M. R. (2021). Cross-site scripting attack detection using machine learning with hybrid features. *INFOTEL, 13(1),* 1–6. https://doi.org/10.20895/infotel.v13i1.606

[10] Talib, N. A., & Kyung-Goo Doh, K, (2022). Run-time Detection of Cross-site Scripting: A Machine-Learning Approach Using Syntactic-Tagging N-Gram Features, International Journal of Computer Science and Security (IJCSS), 16(2), 9 - 27.

[11] Farea, A. A. R., Amran, G. A., Farea, E., Alabrah, A., Abdulraheem, A. A., Mursil, M., & Al-Qaness, M. A. A. (2023). Injections Attacks Efficient and Secure Techniques Based on Bidirectional Long Short Time Memory Model. Computers, Materials and Continua, 76(3). https://doi.org/10.32604/cmc.2023.040121

[12] Hao, S., Long, J., & Yang, Y. (2019). BL-IDS: Detecting Web Attacks Using Bi-LSTM Model Based on Deep Learning. *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. https://doi.org/10.1007/978-3-030-21373-2_45

[13] Sovet, Y. G., & Kokkoz, M. M. (2022). Detection of xss attacks in web applications using machine learning. Вестник Алматинского Университета Энергетики и Связи, (2). https://doi.org/10.51775/2790-0886_2022_57_2_157

[14] Howe, J. M., & Mereani, F. A. (2018, January). Detecting cross-site scripting attacks using machine learning. In International conference on advanced machine learning technologies and applications (pp. 200-210). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-74690-6_20

[15] Tariq, I., Sindhu, M. A., Abbasi, R. A., Khattak, A. S., Maqbool, O., & Siddiqui, G. F. (2021). Resolving cross-site scripting attacks through genetic algorithm and reinforcement learning. Expert Systems with Applications, 168, 114386. https://doi.org/10.1016/j.eswa.2020.114386

[16] Lu, J., Wei, Z., Qin, Z., Chang, Y., & Zhang, S. (2022). Resolving cross-site scripting attacks through fusion verification and machine learning. Mathematics, 10(20), 3787. https://doi.org/10.3390/math10203787

[17] Kumar, A., & Sharma, I. (2023, April). Performance evaluation of machine learning techniques for detecting cross-site scripting attacks. In 2023 11th International Conference on Emerging Trends in Engineering & Technology-Signal and Information Processing (ICETET-SIP) (pp. 1-5). IEEE. https://doi.org/10.1109/icetet-sip58143.2023.10151468

Biography

Emmanuel Osaze Oshoiribhor is a Senior Lecturer in the Department of Computer Science at Ambrose Alli University. He earned his PhD in Computer Science from Ambrose Alli University in 2017, following his Master of Science and Bachelor of Science degrees from the University of Benin. As an active member of several professional organizations, including the Nigeria Computer Society (NCS), he has contributed to the academic community through participation in multiple international conferences. His research interests focus on Investigative Data Mining, Data Science, and Machine Learning.

Adetokunbo MacGregor John-Otumu is a Lecturer and Researcher in the Department of Information Technology at the Federal University of Technology Owerri (FUTO). He was a Postdoctoral Research Fellow at Morgan State University, USA, from 2021 to 2022. He earned a Ph.D. in Computer Science (AI) from Ebonyi State University in 2018, along with master's degrees in Computer Science (Ambrose Alli University) and Information Technology (National Open University of Nigeria). Dr. John-Otumu is a member of professional bodies such as NCS, CPN, IEEE, ACM, AAAI, and the Internet Society. He leads the Machine Learning Research Group in his Department, at FUTO and contributes to global research collaborations. In 2024, he was a Keynote Speaker at the Climate Change Summit (NCS, Imo State) and served as a Technical Committee Member and Session Chair at FUTO's first International Conference on ICT. He is also an editorial board member for several academic publications.

Research Field

Emmanuel Osaze Oshoiribhor: Investigative Data Mining, Machine Learning, Data Science, Intelligent Software Engineering

Adetokunbo MacGregor John-Otumu: Computer Vision, Deep Learning, Machine Learning, NLP, Multi-Agent Systems