information

MDPI

Article

Machine Learning-Driven Detection of Cross-Site Scripting Attacks

Rahmah Alhamyani * and Majid Alshammari *

College of Computer and Information Technology, Taif University, Taif 26571, Saudi Arabia
* Correspondence: rahooma88@hotmail.com (R.A.); m.alshammari@tu.edu.sa (M.A.)

Abstract: The ever-growing web application landscape, fueled by technological advancements, introduces new vulnerabilities to cyberattacks. Cross-site scripting (XSS) attacks pose a significant threat, exploiting the difficulty of distinguishing between benign and malicious scripts within web applications. Traditional detection methods struggle with high false-positive (FP) and false-negative (FN) rates. This research proposes a novel machine learning (ML)-based approach for robust XSS attack detection. We evaluate various models including Random Forest (RF), Logistic Regression (LR), Support Vector Machines (SVMs), Decision Trees (DTs), Extreme Gradient Boosting (XGBoost), Multi-Layer Perceptron (MLP), Convolutional Neural Networks (CNNs), Artificial Neural Networks (ANNs), and ensemble learning. The models are trained on a real-world dataset categorized into benign and malicious traffic, incorporating feature selection methods like Information Gain (IG) and Analysis of Variance (ANOVA) for optimal performance. Our findings reveal exceptional accuracy, with the RF model achieving 99.78% and ensemble models exceeding 99.64%. These results surpass existing methods, demonstrating the effectiveness of the proposed approach in securing web applications while minimizing FPs and FNs. This research offers a significant contribution to the field of web application security by providing a highly accurate and robust ML-based solution for XSS attack detection.

Keywords: cross-site scripting attacks; machine learning; deep learning; artificial neural networks; web security; web vulnerabilities; attack detection; feature selection

Citation: Alhamyani, R.; Alshammari, M. Machine Learning-Driven Detection of Cross-Site Scripting Attacks. *Information* **2024**, *15*, 420. https://doi.org/10.3390/info15070420

Academic Editor: Leandros Maglaras

Received: 14 June 2024
Revised: 7 July 2024
Accepted: 18 July 2024
Published: 20 July 2024

1. Introduction

The Internet is rapidly gaining a footing and affecting every aspect of life. Through the Internet, everyone can access information at any time and from any location [1]. In daily life, web applications are becoming more common. The ever-increasing reliance on web applications for various aspects of our lives, from online banking to social media, necessitates robust security measures [2]. Cyber protection has become increasingly vital in safeguarding sensitive information stored within web browsers, including user account credentials [3]. Despite efforts to secure web applications, they remain vulnerable to hacking due to inherent vulnerabilities, particularly during the initial deployment phases [4]. The Open Web Application Security Project (OWASP) regularly assesses these vulnerabilities, gathering data from over 200,000 organizations and business professionals [5]. Based on the OWASP, the most common vulnerabilities in web applications are injection attacks like cross-site scripting (XSS) and Structured Query Language (SQL) injection and demand significant attention from researchers [6,7]. Moreover, the staggering global costs of threat operations, exceeding USD 6 trillion by the end of 2021 [8], underscore the urgency of addressing such vulnerabilities.

Every vulnerability offers a different entry point for cyber threats: attackers can change databases using SQL injection and insert malicious scripts into web pages using XSS. EdgeScan's 2023 vulnerability statistics report is shown in Figure 1. It illustrates the most common critical vulnerabilities, where XSS attacks take over 19.10% [9].

Figure 1. Vulnerability statics report 2023 by EdgeScan.

Web application security focuses on identifying and addressing security vulnerabilities at the web application level, along with implementing effective solutions for these flaws [10]. Web security is essential for businesses because websites and web servers are vulnerable to internal and external threats. Strict policy measures must be implemented to avoid manipulation or unwanted access to sensitive data or destruction, which could harm the company's operations or reputation. Online security principles include authentication, authorization, auditing, and logging [11]. Security is essential for a secure web application. State integrity refers to maintaining the application's state, which should be kept untampered; logic correctness means that the logic of the application should be precisely corrected as intended by the developers; input validity means that user input is verified before it is used; and security misconfiguration refers to configuration settings and using secure components. For online applications to guarantee the authenticity and responsiveness of user inputs, input validation is essential. Both client-side and server-side inputs, such as HTML5, post message invocations, POST method, database queries, and HTTP request query strings, should be subject to validation [5].

XSS represents a pervasive threat in the realm of cybersecurity, characterized as a client-side code injection attack that exploits vulnerabilities in both client-side and server-side components of web applications [12]. In such attacks, attackers leverage resources from third-party websites to launch scripts within the victim's browser environment. Typically, attackers directly insert payloads containing Java Script (JS) into the database of a targeted website. Subsequently, when a user requests a page from the compromised website, the malicious script-containing page is delivered to the victim's browser, wherein the attacker's payload is executed as part of the Hypertext Markup Language (HTML) body. This method allows attackers to manipulate user interactions, steal sensitive information, or compromise the integrity of web applications [8]. Furthermore, attackers exploit vulnerabilities that are publicly disclosed before patches are developed and deployed, enabling unauthorized access to systems and the unauthorized alteration or theft of data [13]. Figure 2 illustrates the general XSS attacks.

By inserting additional HTML or client script code into a website or input form, attackers can compromise the security of users' browsers, gaining unauthorized access to sensitive data such as cookies, session tokens, etc. [12,14]. This malicious script, capable of rewriting HTML text, enables attackers to compromise user security, extract sensitive data, or even deploy harmful software [15].

Figure 3 illustrates the two primary types of XSS vulnerabilities [15,16]:

Client-side (Document Object Model (DOM)-based XSS);
Server-side (persistent and non-persistent XSS).

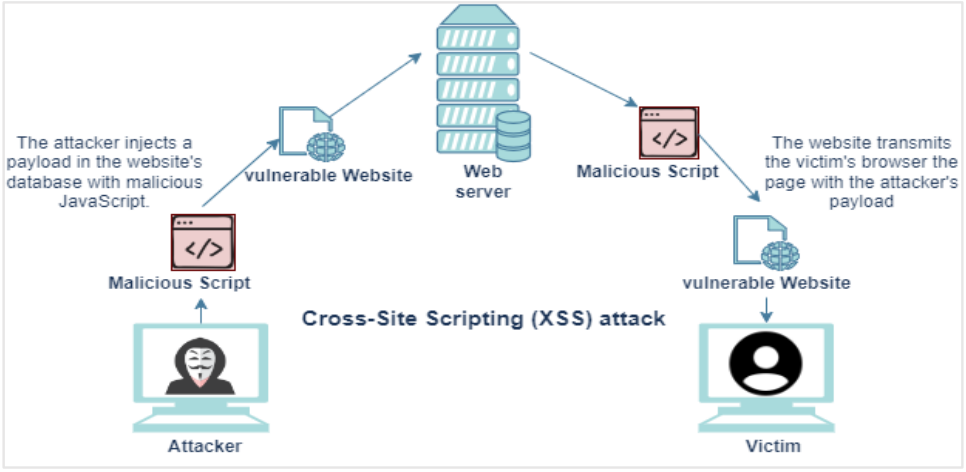

Figure 2. The depiction of the general XSS attack scenario.

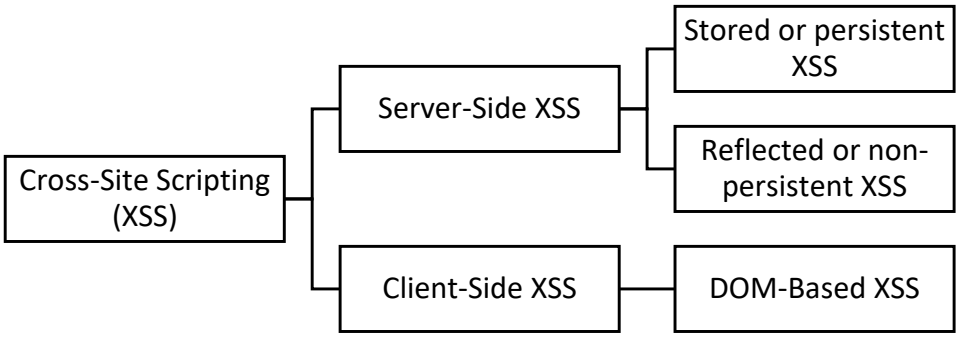

Figure 3. XSS vulnerability taxonomy.

In a stored or persistent XSS attack, malicious scripts are injected into a website, with blogs and forums often being the targets [17]. A database on the website is used to hold scripts that hackers input through forms, posts, or comments. When a victim accesses the compromised website, the script runs, leading the victim to a malicious database. Data belonging to the victim can now be accessed by hackers. Via embedding malicious code that only becomes active when the user visits the infected page, the attack compromises the user's private information by taking advantage of different syntactic symbols [8].

The way a reflected or non-persistent XSS attack works is that a malicious URL is embedded on a website by the attacker, and the user clicks on it. The website poses as a regular browser but really takes the visitor to a malicious page. When the XSS script is parsed by the user, requests are sent to the malicious server. After that, hackers ask the victim to provide their information, which they then take [18].

In DOM-based XSS attacks, the malicious script is loaded into the web browser's DOM, allowing injected code to modify the DOM's contents and an object's properties any time a victim visits a susceptible website [8]. This vulnerability is exploited using client-side scripting, including JavaScript, VBScript, AJAX, ActiveX, and JQuery. AJAX queries and image tags can be used by DOM-XSS attackers to target third-party apps, contrary to the common misconception that they improve online security [19]. Attackers are able to obtain cookies and confidential information such as IP addresses and user passwords as a result [8]. DOM-based XSS attacks alter the client's DOM directly as opposed to reflected XSS attacks, which include introducing malicious code into the website's response [17].

Server-side XSS vulnerabilities arise when unsensitized data from the server are incorporated directly into the HTTP response, posing significant risks to all users accessing the compromised website. Conversely, client-side XSS vulnerabilities emerge when JS manipulates page content, allowing attackers to craft malicious Uniform Resource Locators

(URLs) containing unfiltered text that executes upon user interaction. The repercussions of successful XSS payloads include cookie theft, keylogging, session hijacking, and identity theft [16].

Despite the widespread adoption of standardized code development practices, over 60% of websites remain vulnerable to XSS attacks, highlighting the critical need for robust detection and prevention mechanisms [20]. Identifying and thwarting XSS attacks are paramount for safeguarding web applications and protecting sensitive user data. To this end, various analysis techniques have been employed, including static analysis, dynamic analysis, and machine learning (ML). Static analysis involves scrutinizing the source code to detect vulnerabilities, offering assurances regarding specific vulnerability absence but potentially requiring extensive time and yielding limited results [21]. Conversely, dynamic analysis focuses on understanding script behavior during execution, facilitating the identification of unknown vulnerabilities and novel attack types that static analysis may overlook [21].

Traditional methods for XSS detection often suffer from high false-positive (FP) rates, meaning that they flag harmless activity as malicious. Additionally, these methods may struggle to adapt to new attack vectors employed by cybercriminals. ML, on the other hand, leverages existing script data to create classifiers and predict the behavior of new scripts, offering the rapid identification of malicious scenarios, adaptability to evolving attack types, and the ability to operate across diverse application environments without the need for a dedicated analysis environment [21].

The integration of ML into XSS detection frameworks holds significant promise, enabling enhanced threat detection capabilities and proactive defense measures against evolving cyber threats. This paper explores the effectiveness and benefits of ML-based XSS detection methods, highlighting their potential to bolster web application security and mitigate the risk of XSS attacks.

This research has several key objectives:

1. Create an ML-based model: We provide an ML model that greatly enhances the precision and potency of XSS detection in web applications.
2. Identify ideal features: To guarantee the accurate detection of XSS attacks while reducing false alarms, we seek to identify the best traits and data sources for ML model training.
3. Assess the efficacy of ML-based detection systems: We will evaluate the ML-based approach's accuracy, efficiency, and reliability by comparing it with state-of-the-art detection techniques.
4. Examine current methods: We will quickly summarize current ML and deep learning (DL) algorithms that have been applied to XSS detection.

This research addresses these challenges by proposing a novel ML-based system for XSS attack detection. Our aim is to enhance web application security by developing a more accurate, robust, and adaptable detection system. By reducing the number of FPs and effectively identifying XSS attacks, this research contributes significantly to the improvement in web application security practices.

We develop and assess the effectiveness of ML-based methods, including Decision Trees (DTs), Support Vector Machines (SVMs), Random Forest (RF), Logistic Regression (LR), Extreme Gradient Boosting (XGBoost), Multi-Layer Perceptron (MLP), Convolutional Neural Networks (CNNs), Artificial Neural Networks (ANNs), and ensemble learning with feature selection techniques such as Information Gain (IG) and Analysis of Variance (ANOVA) to identify the most relevant features and enhance the accuracy and efficiency of identifying XSS in web applications. The experiment is conducted using the XSS dataset that was published by Mokbal et al. [22]. We selected the top 25 features using IG as a feature selection technique to choose the most informative features.

After a comparative analysis of the ten models' performance, the RF, ensemble model of RF, DTs with Gradient Boost (GB), and ensemble mode of RF with MLP, respectively, scored 99.78%, 99.76%, and 99.65% in terms of accuracy and achieved high results in the

other evaluation metrics. The proposed models provide significant performance improvements compared to other existing state-of-the-art methods as they can detect XSS-based attacks while simultaneously minimizing FP and false-negative (FN) rates. Moreover, the proposed method is compared with other existing state-of-the-art XSS attack detection methods.

The remainder of this paper is organized as follows. Section 2 presents different methods for XSS attack detection, followed by the proposed research methodology in Section 3. Sections 4 and 5 present the results and discussion, respectively. Finally, Section 6 contains the conclusion.

2. Related Work

ML approaches have emerged as a promising avenue for XSS attack detection in web applications. Their ability to learn from data and adapt to evolving attack patterns offers significant advantages over traditional methods [23]. However, selecting the most effective preprocessing techniques is crucial to optimize detection performance [23].

Several studies have explored various ML algorithms for XSS detection. Banerjee et al. [24] implemented ML algorithms for identifying XSS threats. These algorithms include SVMs, KNN, RF, and LR. The LR model was utilized to map the true and false values included in the dataset. The implementation of the suggested model used a dataset with 24 attributes based on JS and URL features. Among these, 1453 samples were benign, while 158 were flagged as malicious. They achieved the highest accuracy of 98% for the RF classifier. Similarly, Habibi and Surantha [12] proposed a method to enhance XSS attack detection performance by using different ML techniques with an n-gram approach to script features. These techniques include SVMs, KNN, and NB. The results demonstrate that SVMs and the n-gram method work together to reach the maximum accuracy and achieve an accuracy of 98%. Kascheev and Olenchikova [21] compared various ML algorithms such as the SVMs, DTs, NB, and LR classifier, with DTs achieving the highest accuracy of 98.81%. While these studies demonstrate the effectiveness of ML, they also highlight the importance of feature selection, as evidenced by the lower performance of LR in Kascheev and Olenchikova's work [21].

Gogoi et al. [25] proposed a hybrid approach combining traditional Web Application Firewalls with ML algorithms like SVMs. Their focus on reducing FPs and FNs while maintaining high precision yielded promising results of 98%. They discovered that SVMs successfully distinguished inputs from XSS attacks and legitimate web applications with a balance between precision and accuracy. Mokbal et al. [22] introduced the XGBXSS framework, utilizing XGBoost and a hybrid feature selection technique consisting of sequential backward selection (SBS) combined with Information Gain (IG) to choose an optimal subset while lowering computing expenses and preserving the good performance of the detector. Also, the study's dataset included 138,569 samples, with 100,000 samples classified as benign. This ensemble learning approach achieved exceptional performance with an accuracy of 99.59%, precision of 99.53%, and a low FP rate of 0.18%.

Research on hybrid models for XSS detection is also gaining traction. Stiawan et al. [26] combined Long Short-Term Memory (LSTM) with Principal Component Analysis (PCA) for dimensionality reduction, achieving 96.85% accuracy. Other studies explored combinations of LSTM and CNNs [27] or CNNs and scanners [28] to achieve high accuracy rates exceeding 99%. Melicher et al. [29] proposed a method integrating three-layer Deep Neural Networks (DNN) with taint tracking for DOM XSS detection, achieving 95% accuracy but with limitations in precision, achieving 26.7%. Alaoui et al. [30] utilized an LSTM Encoder–Decoder with word embeddings including tools like word2vec, FastText, and Glove for XSS attack detection, reaching a precision of 99.09% and recall and accuracy of 99.08% each.

Gupta et al. [31] presented GeneMiner, a system for detecting novel XSS attacks. It consists of GeneMiner-E for extracting new features and GeneMiner-C for classifying input payloads as either malicious or benign. GeneMiner responds to changing patterns of attack

payloads to detect adversarial XSS attacks. They evaluate their classification accuracy by comparing it with NB, RF, LR, SVMs, AdaBoost, MLP, CNNs, and reinforcement learning. Their approach achieved an accuracy of 98.5% in identifying newly crafted malicious payloads. Dawadi et al. [32] conducted a comparative analysis using LSTMs for Distributed Denial of Service (DDoS), XSS, and SQL injection detection, achieving an accuracy of 89.34% for XSS attacks within their layered architecture model.

The relevance of XSS attack detection extends beyond web applications to Internet of Things (IoT) devices and cloud-based services. Tian et al. [33] proposed a CNN-based method for edge devices that can be used with cloud-based web applications. The suggested model used the Continuous Bag of Words (CBOWs) model to vectorize URLs during the data preparation phase. The Rectified Linear Unit (ReLU) function, dropout layers, and pooling layers are used in the CNN architecture to optimize the CNN model, achieving an accuracy of 99.41%. Chaudhary et al. [34] introduced a CNN-based approach for identifying XSS attacks. The suggested approach applies CNNs after two stages of data preparation, specifically decoding and contextual tagging. Their work has been implemented in Fog nodes connected with IoT networks, achieving an accuracy of 99.88%. Luo et al. [35] proposed an Ensemble DL-Based Web Attack Detection System (EDL-WADS) for online attack detection in IoT networks. The suggested model examined URL requests for abnormalities, and it used a combination of three DL-based models, namely Multimodal Residual Networks (MRNs), CNNs, and LSTM, achieving 98% accuracy. Finally, Odun-Ayo et al. [36] explored MLP for real-time XSS detection in cloud-based web applications, achieving an accuracy of 99.47%.

Furthermore, several recent studies have introduced novel approaches to detecting XSS attacks, leveraging advanced techniques such as attention mechanisms, generative adversarial networks (GANs), and Monte Carlo Tree Search (MCTS) algorithms to enhance detection accuracy and robustness. For instance, [37] proposed the LSTM-attention detection model, integrating an attention mechanism into the LSTM recurrent neural network (RNN) architecture. Achieving remarkable precision and recall rates of 99.3% and 98.2%, respectively, their method demonstrated superior performance in identifying XSS attacks by enhancing the recognition of malicious codes and feature extraction. In a similar vein, [38] introduced the Paths Attention Method (PATS) for detecting reflected XSS vulnerabilities, utilizing syntactic pathways and attention processes to improve training effectiveness. PATS achieved an accuracy rate of 90.25% and an F1-Score of 81.62% while also addressing dataset limitations through the creation of a reliable dataset consisting of 10,000 benign samples from GitHub and 1000 malicious samples from the National Institute of Standards and Technology (NIST). Additionally, [39] employed the MCTS algorithm and GANs to generate adversarial XSS attacks, enhancing the detection rate of adversarial examples. Their method significantly improved the detector's performance by increasing the rate of discovering adversarial samples. Meanwhile, [40] emphasized the susceptibility of DL models to adversarial attacks and proposed a GAN-based approach to automatically generate hostile XSS attacks against an LSTM-based XSS attack detection model. It demonstrated a significant decline in the detection model's performance when tested on modified XSS instances produced by the GAN model and achieved an accuracy of 98%.

Tariq et al. [41] introduced a hybrid methodology combining genetic algorithms, statistical inference, and reinforcement learning (RL) to assess the proximity of each payload to malicious and benign samples, offering a novel approach by merging ML with metaheuristic algorithms like the genetic algorithm, achieving an accuracy of 99.89%. Thajeel et al. [42] addressed the evolving nature of XSS attacks and feature drift using a deep Q-network multi-agent framework (DQN-MAFS) for dynamic feature selection, achieving an accuracy of 98.37%. Their proposed approach, called fair agent reward distribution-based dynamic feature selection (FARD-DFS), outperformed existing methods in terms of accuracy and F1 measure, providing real-time updates and correction of embedded knowledge without the need for offline retraining.

These studies showcase the effectiveness of various ML and DL approaches for XSS attack detection. They highlight the importance of feature selection, ensemble learning techniques, and the exploration of novel architectures like CNNs and LSTMs for achieving high accuracy and adapting to evolving attack vectors. However, challenges such as the lack of standardized datasets, adaptation to emerging attack vectors, and reducing FP rates persist, indicating the need for further research in this field. Our research builds upon this foundation by proposing a novel ML-based model that leverages these insights to further enhance XSS attack detection performance.

3. Research Methodology

This section outlines the methodology employed in conducting research on XSS attack detection utilizing ML techniques and comparison to other state-of-the-art methods. The methodology encompasses data collection, preprocessing, feature selection, model training, and evaluation. Figure 4 illustrates the proposed models' framework.

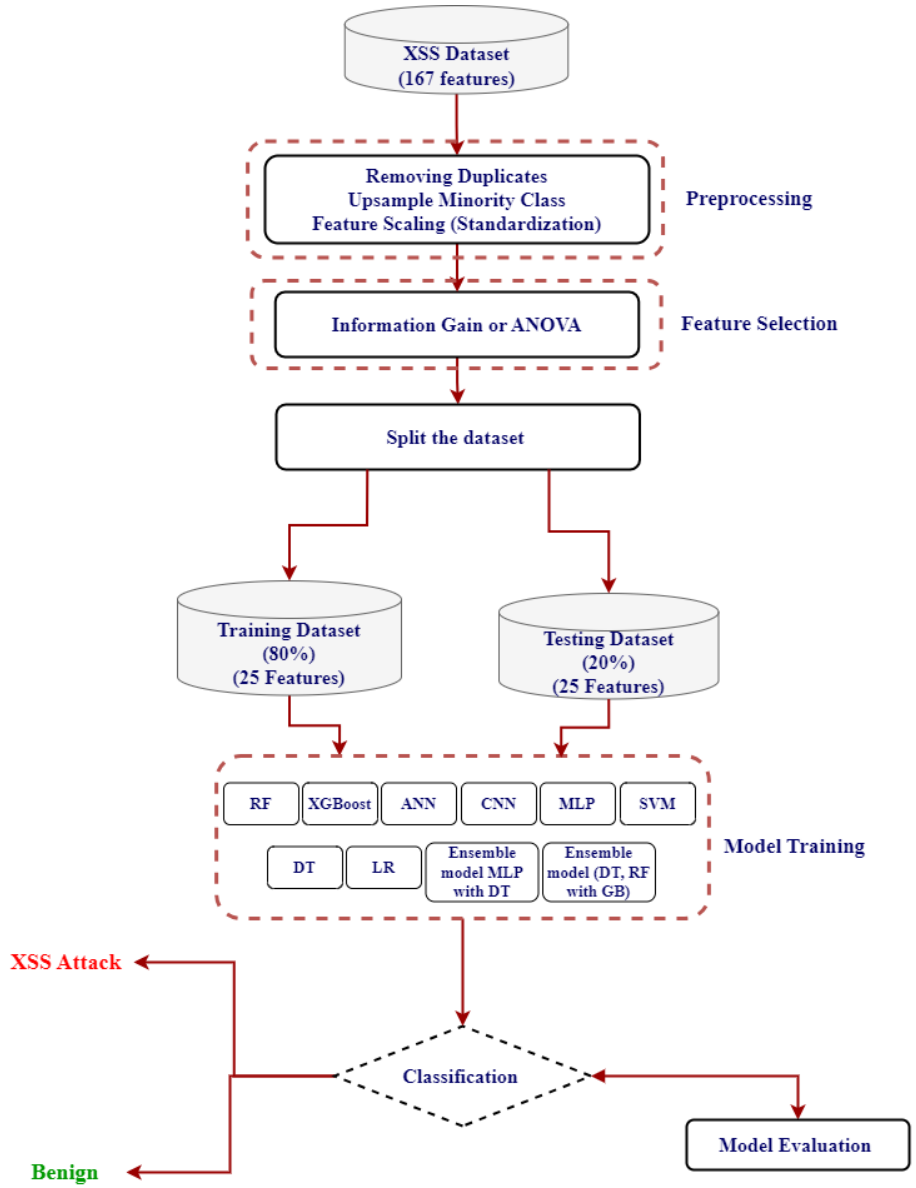

Figure 4. The proposed models' framework.

3.1. Data Collection

The research utilizes the XSS dataset provided by Mokbal et al. [22]. This dataset comprises 138,569 webpages, with 100,000 labeled as benign and 38,569 labeled as malicious. The benign samples in this dataset were produced using an Alexa ranking of the top 50,000 websites. XSSed and Open Bug Bounty are two examples of raw XSS repositories that crawled to gather malicious samples, ensuring a diverse and representative dataset for analysis. The dataset includes 167 features. Recently, this dataset was made accessible online via GitHub [22]. Additionally, the dataset comprises three distinct feature types: HTML, JS, and URL, and we used the whole dataset in the experiments. Table 1 provides summarized information about the dataset obtained for this research.

Table 1. Summarized dataset's information.

Author	No. of Features	Feature Types	No. of Benign Samples	No. of Malicious Samples	Total Number of Samples
Mokbal et al. [22]	167	URL, JS, and HTML	100.000	38.569	138,569

3.2. Preprocessing

Before training the models, the dataset undergoes preprocessing to ensure uniformity and relevance. This step involves cleaning the data, handling missing values, class imbalance, and standardizing formats. We handled the class imbalance by upsampling the minority class, which involves increasing the number of samples in the minority class to balance it with the majority class. By dropping duplicates, the XSS dataset is streamlined and ready for subsequent processing steps, including train–test splitting. Furthermore, we used the standard scalar technique in Python language as feature scaling. Moreover, challenges like the presence of irrelevant features are addressed through feature selection techniques.

3.3. Feature Selection

To enhance the accuracy and efficiency of XSS attack detection, feature selection algorithms such as IG and ANOVA are applied. These algorithms help identify the most discriminative and informative features crucial for accurate detection, thereby reducing noise and improving model performance.

When using an ANOVA for feature selection, the variance between feature groups and the variance within groups are compared to obtain the F-values for each feature. High F-value and low p-value features are deemed important and are included in the model. Feature retention is determined by a significance criterion, often set at $p < 0.05$. This approach increases interpretability, lowers dimensionality, and boosts model performance [43].

IG is a commonly used method to evaluate the importance of features in predicting the target variable (i.e., whether a given input represents a benign or malicious XSS attack). IG measures the reduction in entropy or uncertainty in the target variable that is achieved by splitting the dataset based on the values of a particular feature [44,45].

In the context of using IG and the ANOVA test as feature selection methods for XSS attack detection, the observation that IG yielded better features prompts a discussion about the effectiveness and implications of feature selection techniques. The superiority of IG in selecting features suggests that it successfully identified the most informative predictors for distinguishing between XSS attacks and non-attacks.

By employing IG, the most informative features that contribute significantly to predicting XSS attacks are identified and used to train ML models effectively. This helps improve model performance, reduce computational complexity, and enhance interpretability in XSS detection tasks. Table 2 shows the 25 features selected out of the 167 in the dataset.

Table 2. The selected features using IG (25 out of 167) features.

Feature No.	Feature Name	Feature Description
1	url_length	The length of the URL string in characters.
2	url_special_characters	The count of special characters (e.g., !, @, #, $) present in the URL.
3	url_tag_script	Binary indicator (0 or 1) representing whether the URL contains the <script> tag, which is commonly exploited in XSS attacks.
4	url_cookie	Binary indicator (0 or 1) representing whether the URL contains references to cookies, which may indicate potential security vulnerabilities.
5	url_number_keywords_param	The count of predefined keywords (e.g., signup, login, query) present as parameters in the URL.
6	url_number_domain	The count of domains referenced in the URL, which may indicate redirection or external linking.
7	html_tag_script	Binary indicator (0 or 1) representing whether the HTML content contains the <script> tag, which can execute JS code and potentially lead to XSS vulnerabilities.
8	html_tag_meta	Binary indicator (0 or 1) representing whether the HTML content contains the <meta> tag, which is used for metadata information and can be manipulated for malicious purposes.
9	html_tag_link	Binary indicator (0 or 1) representing whether the HTML content contains the <link> tag, which is used to define relationships between documents and can be exploited in XSS attacks.
10	html_tag_div	Binary indicator (0 or 1) representing whether the HTML content contains the tag, which is commonly used for layout purposes and can be manipulated for XSS attacks.
11	html_tag_style	Binary indicator (0 or 1) representing whether the HTML content contains the <style> tag, which is used to define styles and can be manipulated to execute malicious code.
12	html_attr_background	Binary indicator (0 or 1) representing whether the HTML content contains the background attribute, which can be exploited for XSS attacks.
13	html_attr_href	Binary indicator (0 or 1) representing whether the HTML content contains the href attribute, commonly used for hyperlinks and can be manipulated for XSS attacks.
14	html_attr_src	Binary indicator (0 or 1) representing whether the HTML content contains the src attribute, commonly used to specify the source of external resources and can be manipulated for XSS attacks.
15	html_event_onmouseout	Binary indicator (0 or 1) representing whether the HTML content contains the onmouseout event attribute, which can execute JS code when the mouse leaves an element and may be exploited for XSS attacks.
16	js_file	Binary indicator (0 or 1) representing whether JS files are referenced in the HTML content, which may contain vulnerable code.
17	js_dom_location	Binary indicator (0 or 1) representing whether the JS code accesses the location object, which can manipulate the URL and may lead to XSS vulnerabilities.
18	js_dom_document	Binary indicator (0 or 1) representing whether the JS code accesses the document object, which represents the HTML document and can be manipulated for XSS attacks.
19	js_method_getElementsByTagName	Binary indicator (0 or 1) representing whether the JS code uses the getElementsByTagName() method, which retrieves elements by tag name and may be used in XSS attacks.
20	js_method_getElementById	Binary indicator (0 or 1) represents whether the JS code uses the getElementById() method, which retrieves an element by its ID and may be exploited for XSS attacks.
21	js_method_alert	Binary indicator (0 or 1) represents whether the JS code uses the alert() method, which displays an alert dialog box and may be used for XSS attacks.
22	js_min_length	The minimum length of JS strings in the code.
23	js_min_function_calls	The minimum number of function calls in the JS code.
24	js_string_max_length	The maximum length of JS strings in the code.
25	html_length	The length of the HTML content in characters.

3.4. Model Training

This research utilizes a diverse range of ML algorithms during both the training and evaluation phases. These models include DTs, SVMs, RF, LR, XGBoost, MLP, CNNs, and ANNs. Each model is characterized by distinct algorithms and architectures, contributing to the exploration of varied detection methodologies.

In addition to individual models, two ensemble models are investigated: the first ensemble model combines the MLP classifier with RF, while the second integrates DTs with RF with GB. This ensemble approach aims to enhance detection accuracy and robustness by amalgamating the complementary features of constituent algorithms.

The training phase utilizes standard supervised learning approaches provided by the Sklearn library in Python. Each algorithm is trained using 80% of the dataset, randomly selected for model construction, while the remaining 20% is reserved for testing. This partitioning ensures the evaluation of model generalization and performance on unseen data, thereby validating the efficacy of the proposed detection approach.

3.5. Evaluation

The performance of each trained model is evaluated to determine its effectiveness in detecting XSS attacks. Evaluation metrics such as accuracy, precision, recall, F1-Score, Receiver Operating Characteristic—Area Under the Curve (ROC-AUC) score, and confusion matrix are used to assess the models' performance [23]. The evaluation process provides insights into the strengths and weaknesses of each model, guiding the selection of the most suitable approach for XSS attack detection.

3.5.1. Accuracy

Accuracy is a metric for assessing the potency of a classification model. It shows the proportion of accurately classified samples out of all the samples that have been classified, as shown in Equation (1).

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

3.5.2. Precision

Precision serves as a metric for assessing the ability of a model to predict positive samples. The total number of positive samples indicates the ratio of correctly predicted samples (*TP*), as shown in Equation (2).

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

3.5.3. Recall

Recall is a metric that assesses the ability of a model to identify positive samples. The quantity of positive samples that indicate that the prediction was an accurate true positive (*TP*) divided by the total number of samples in the same real class is what this indicates. It serves as an example of the model's *FN*, as shown in Equation (3).

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

3.5.4. F1-Score

The F1 measure is sometimes referred to as the harmonic mean of recall and precision because it considers both metrics and provides a fair evaluation of their performance. This assessment metric is commonly used when datasets are uneven, meaning that one class has a much higher number of occurrences than the other, as shown in Equation (4).

$$F1\text{-}score = \frac{1}{\propto \cdot \frac{1}{p} + (1 - \alpha) \cdot \frac{1}{R}} \tag{4}$$

3.5.5. ROC-AUC

ROC-AUC is a popular function in ML for assessing the performance of binary classification models. It computes the ROC-AUC. The ROC-AUC score goes from 0 to 1, with 1 representing perfect classification and 0.5 indicating random guessing. A score higher than 0.5 indicates that the model outperforms random.

As XSS attacks evolve in complexity with advancements in web applications, this research acknowledges the challenges posed by obfuscation techniques and semantic reasoning in attack statements. The methodology is designed to address these challenges by employing robust preprocessing, feature selection, and model training techniques to enhance the detection of XSS vulnerabilities in web applications.

4. Results

The experimental results, as depicted in Table 3, underscore the efficacy of employing diverse ML approaches in XSS attack detection, signifying a significant advancement in web security measures. Moreover, the proposed method is benchmarked against existing state-of-the-art XSS attack detection methods, offering a comparative analysis to gauge its performance and efficacy.

Table 3. The experimental results of the proposed models.

Model	Evaluation Metrics				Confusion Matrix			
	Accuracy (%)	**Precision (%)**	**Recall (%)**	**F1-Score (%)**	**TP (%)**	**FP (%)**	**TN (%)**	**FN (%)**
LR	98.28	99.38	94.32	96.79	99.78	0.22	94.32	5.68
SVMs	98.53	99.21	97.84	98.52	99.22	0.78	97.84	2.16
MLP	99.14	99.26	99.02	99.14	99.27	0.73	99.02	0.98
ANNs	99.06	99.08	99.04	99.06	99.08	0.92	99.04	0.96
CNNs	98.82	99.57	98.07	98.81	99.57	0.43	98.07	1.93
XGboost	99.62	99.70	99.54	99.62	99.70	0.30	99.54	0.46
DTs	99.47	99.22	99.72	99.47	99.22	0.78	99.72	0.28
RF	99.78	99.80	99.75	99.78	99.80	0.20	99.75	0.25
Ensemble model (MLP with RF)	99.65	99.59	99.71	99.65	99.59	0.41	99.71	0.29
Ensemble model (DTs, RF with GB)	99.76	99.74	99.77	99.76	99.74	0.26	99.77	0.23

All experiments were meticulously conducted within the Google Colab environment, ensuring consistency and reproducibility in training and testing the models. In the subsequent sections, a detailed exploration of the empirical findings unfolds, shedding light on various aspects, including model performance, feature importance, computational efficiency, and broader implications for bolstering web security against XSS vulnerabilities.

Through this comprehensive analysis, valuable insights are gleaned, paving the way for the development of enhanced detection mechanisms and resilient defense strategies in the dynamic landscape of web security. We applied various ML models, outlined in the following.

4.1. Logistic Regression (LR)

We employed IG to select the top 25 features essential for XSS attack detection. Additionally, we fine-tuned the LR model by specifying a maximum of 1000 iterations. The LR model demonstrated promising performance, achieving an accuracy of 98.28%, with precision and recall rates of 99.38% and 94.32%, respectively. The F1-Score, a harmonic mean of precision and recall, was calculated at 96.79%. Furthermore, the ROC-AUC score,

indicative of the model's ability to distinguish between classes, stood at 97%, highlighting its robustness in discriminating between benign and malicious instances. The confusion matrix provides additional insights into the LR classifier's performance, revealing a minimal misclassification rate of 0.22% for benign instances and 5.68% for malicious instances.

4.2. Support Vector Machine (SVM)

We utilized IG to select the top 25 features crucial for XSS attack detection. Leveraging the SVC with the Radial Basis Function (RBF) kernel, we set the gamma parameter to 'scale' for optimal performance.

The SVM classifier exhibited impressive performance, achieving an accuracy of 98.53%, with precision and recall rates of 99.21% and 97.84%, respectively. The F1-Score, a balanced measure of precision and recall, was calculated at 98.52%. Furthermore, a high ROC-AUC score of 99% underscores the model's robustness in distinguishing between benign and malicious instances. The confusion matrix provides additional insights into the SVM classifier's performance, revealing a minimal misclassification rate of 0.78% for benign instances and 2.16% for malicious instances.

4.3. Multi-Layer Perceptron (MLP)

We employed IG to select the top 25 features crucial for XSS attack detection. The MLP classifier, a powerful neural network model, was configured with 100 neurons in the hidden layer, utilizing the ReLU activation function and the Adam optimizer for efficient training.

The MLP classifier demonstrated exceptional performance, achieving an accuracy of 99.14% with precision and recall rates of 99.26% and 99.02%, respectively. The F1-Score, a balanced measure of precision and recall, was calculated at 99.14%, indicating the classifier's robustness in accurately identifying XSS attacks. Furthermore, a high ROC-AUC score of 99% underscores the model's ability to discriminate between benign and malicious instances effectively. The confusion matrix provides additional insights, revealing a minimal misclassification rate of 0.73% for benign instances and 0.98% for malicious instances.

4.4. Artificial Neural Networks (ANNs)

We utilized the ANOVA F-test to select the top 25 features essential for detecting XSS attacks. We implemented a sequential model in the ANN architecture, incorporating specific configurations to optimize performance. The model comprises an input layer with the ReLU activation function, followed by two hidden layers with tanh and ReLU activation functions, respectively, and an output layer with sigmoid activation function. Additionally, we employed the Adam optimizer and the binary cross-entropy loss function to facilitate efficient training.

Furthermore, to ensure convergence and robustness, we set the batch size to 32 and trained the model for 50 epochs. The ANN model demonstrates outstanding performance, achieving an accuracy of 99.06%, with precision and recall rates of 99.08% and 99.04%, respectively. The F1-Score, a balanced measure of precision and recall, was calculated at 99.06%, indicating the model's effectiveness in accurately identifying XSS attacks.

Moreover, a high ROC-AUC score of 99.94% underscores the model's exceptional ability to discriminate between benign and malicious instances. The confusion matrix provides additional insights, revealing a minimal misclassification rate of 0.92% for benign instances and 0.96% for malicious instances.

4.5. Convolutional Neural Networks (CNNs)

We employed IG to select the top 25 features crucial for detecting XSS attacks. The CNN model was initialized as a sequential model, specifying the number of features. ReLU activation functions were applied to the layers, accompanied by MaxPooling to enhance feature extraction. Additionally, we utilized the Adam optimizer and the binary cross-entropy loss function for efficient training.

The CNN model demonstrates robust performance, achieving an accuracy of 98.82%, with precision and recall rates of 99.57% and 98. 07%, respectively. The F1-Score, a balanced measure of precision and recall, was calculated at 98.81%, indicating the model's effectiveness in accurately identifying XSS attacks.

Moreover, a high ROC-AUC score of 99.92% underscores the model's exceptional ability to discriminate between benign and malicious instances. The confusion matrix provides additional insights, revealing a minimal misclassification rate of 0.43% for benign instances and 1.93% for malicious instances.

4.6. Extreme Gradient Boosting (XGBoost)

We employed IG to select the top 25 features crucial for detecting XSS attacks. Furthermore, we meticulously defined the parameters for XGBoost, a renowned gradient-boosting algorithm known for its efficacy in classification tasks. These parameters play a pivotal role in configuring the behavior and performance of the XGBoost model. Specifically, we specified the objective function as binary classification using LR and utilized the classification error rate as the evaluation metric during training. Moreover, we set the maximum depth of each tree to 6, the learning rate to 0.3, and the subsample and colsample_bytree parameters to 1. The evaluation metrics for the XGBoost model on the selected dataset showcased outstanding performance, with an accuracy of 99.62%, precision of 99.70%, recall of 99.54%, and an F1-Score of 99.62%. Notably, an ROC-AUC score of 100% underscores the model's exceptional ability to distinguish between benign and malicious instances. The confusion matrix provides additional insights, revealing a minimal misclassification rate of 0.30% for benign instances and 0.46% for malicious instances.

4.7. Decision Tree (DT)

We leveraged the top 25 features selected based on IG to train a DT model. We configured the model with a criterion set to "gini" and a minimum samples leaf equal to one to optimize its performance. The DT model exhibited strong performance, achieving an accuracy of 99.47%, with precision and recall rates of 99.22% and 99.72%, respectively. The F1-Score, a balanced measure of precision and recall, was calculated at 99.47%, indicating the model's effectiveness in accurately identifying XSS attacks. Furthermore, a high ROC-AUC score of 99% underscores the model's ability to discriminate between benign and malicious instances. The confusion matrix provides additional insights, revealing a minimal misclassification rate of 0.78% for benign instances and 0.28% for malicious instances.

4.8. Random Forest (RF)

We utilized the top 25 features selected through IG to train an RF classifier. We configured the RF model with the n_estimators parameter set to 120 to ensure robustness and accuracy. The RF model demonstrated exceptional performance, achieving an accuracy of 99.78%, with precision and recall rates of 99.80% and 99.75%, respectively. The F1-Score, a balanced measure of precision and recall, was calculated at 99.78%, indicating the model's effectiveness in accurately identifying XSS attacks.

Moreover, a high ROC-AUC score of 100% underscores the model's exceptional ability to discriminate between benign and malicious instances. The confusion matrix provides additional insights, revealing a minimal misclassification rate of 0.20% for benign instances and 0.25% for malicious instances.

4.9. Ensemble Model of MLP Classifier and RF

We employed a Voting Classifier that combines the strengths of the MLP classifier and RF to enhance predictive performance. The MLP classifier excels at capturing complex nonlinear relationships within the data, leveraging its layered structure and activation functions to learn intricate patterns and generalize well on unseen data. Conversely, RF is a robust ensemble learning method that aggregates predictions from multiple DTs, offering resistance against overfitting and providing insights into feature importance.

By leveraging the complementary strengths of these models, our ensemble approach aims to improve predictive performance and model generalization across various datasets and applications. We selected the top 25 features using IG. For the RF classifier, we set the n_estimators parameter to 100, while for the MLP classifier, we specified a maximum of 1000 iterations and defined two hidden layers with sizes of 170 and 50 neurons, respectively.

The ensemble model of RF and MLP demonstrates outstanding performance, achieving an accuracy of 99.65%, with precision and recall rates of 99.59% and 99.71%, respectively. The F1-Score, a balanced measure of precision and recall, was calculated at 99.65%, indicating the model's effectiveness in accurately identifying XSS attacks.

Moreover, a high ROC-AUC score of 100% underscores the model's exceptional ability to discriminate between benign and malicious instances. The confusion matrix provides additional insights, revealing a minimal misclassification rate of 0.41% for benign instances and 0.29% for malicious instances.

4.10. Ensemble Model of DTs, RF, and GB

We employed a Voting Classifier combining DT and RF with GB in XSS detection attacks. This ensemble technique utilizes hard voting to make final predictions, leveraging the diversity of these algorithms to enhance the overall performance of the XSS detection system.

DTs are powerful models capable of capturing complex relationships through hierarchical decisions, providing interpretability to the ensemble. RF, an ensemble method, aggregates predictions from multiple DTs, reducing overfitting and improving robustness through randomness in the training process. GB constructs an ensemble of weak learners, typically DTs, sequentially focusing on correcting errors, leading to enhanced predictive accuracy.

By combining these three algorithms, we aim to exploit their complementary strengths: DTs for interpretability, RF for robustness, and GB for predictive accuracy. This ensemble approach has the potential to achieve superior performance in detecting XSS attacks by leveraging diversity and ensemble learning techniques. By using a Voting Classifier, we combine the predictions of these classifiers, benefiting from their diversity and leading to a more accurate and reliable detection of XSS attacks.

We selected the top 25 features using IG. For the RF and GB classifiers, we set the n_estimators parameter to 100. The ensemble model of DT and RF with GB demonstrates outstanding performance, achieving an accuracy of 99.76%, with precision and recall rates of 99.74% and 99.77%, respectively. The F1-Score, a balanced measure of precision and recall, was calculated at 99.76%, indicating the model's effectiveness in accurately identifying XSS attacks. Moreover, a high ROC-AUC score of 100% underscores the model's exceptional ability to discriminate between benign and malicious instances. The confusion matrix provides additional insights, revealing a minimal misclassification rate of 0.26% for benign instances and 0.23% for malicious instances.

Overall, the ensemble approach combining DT and RF with GB showcases promising results, highlighting its potential as a valuable tool for enhancing web security measures and mitigating cyber threats.

5. Discussion

In summary, the comprehensive evaluation of all the proposed models in our research, as depicted in Figures 5 and 6, underscores the effectiveness of various ML techniques for XSS attack detection. Among these models, the RF model emerges as the top-performing one, exhibiting the highest accuracy and balanced performance across all evaluation metrics. Nevertheless, other models such as XGBoost also demonstrate competitive performance, showcasing the versatility and efficacy of diverse ML approaches in addressing XSS vulnerabilities. The ensemble models, combining multiple classifiers, further accentuate the potential benefits of leveraging ensemble techniques to enhance predictive performance. These findings offer valuable insights for both researchers and practitioners in the cyber-

security domain, guiding the development of more robust and effective XSS detection systems to bolster web security measures and mitigate cyber threats effectively.

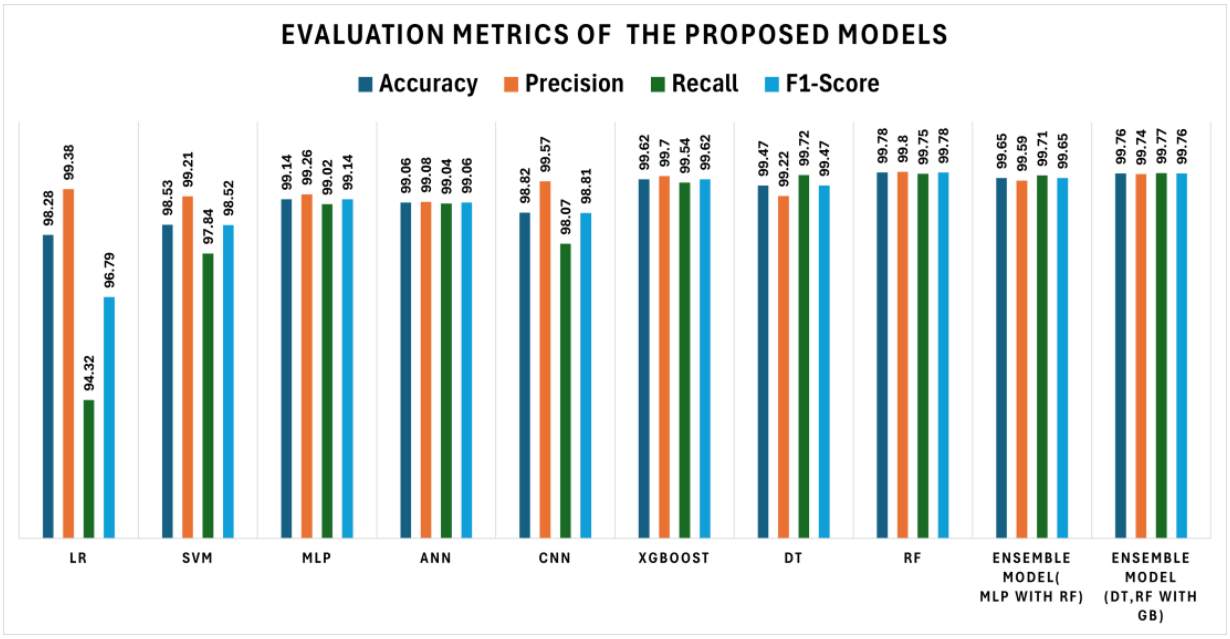

Figure 5. Evaluation metrics of the proposed models.

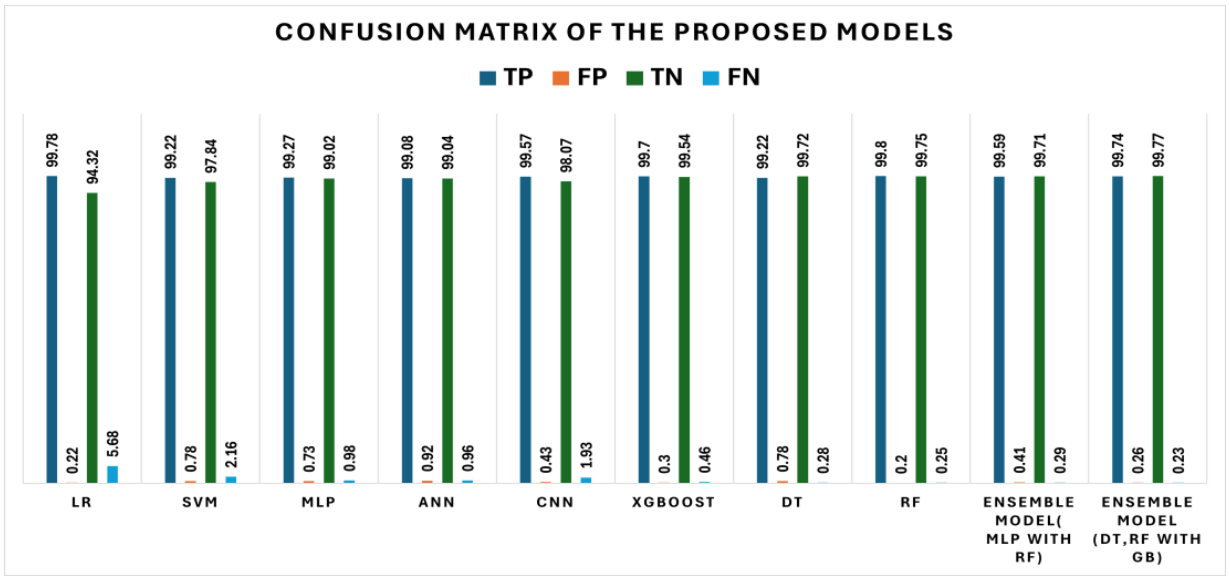

Figure 6. Confusion matrix of the proposed models.

5.1. Comparison with Other State-of-the-Art Methods

To verify the performance of our proposed method, we conducted a comparative analysis with several recent XSS attack detection methods, all using the XSS dataset by Mokbal et al. [22]. Mokbal et al. [22] achieved an accuracy of 99.59% using XGBoost with feature selection based on IG and sequential backward selection (SBS), utilizing 30 features. Although their accuracy was commendable, it is noteworthy that our models achieved comparable accuracy with a reduced feature set of 25, highlighting the efficiency of our approach. Thajeel et al. [42] employed a DT model with dynamic selection based on RL, showcasing the potential of dynamic selection techniques. However, their model utilized

167 features and exhibited lower performance compared to ours, suggesting potential limitations in their approach or dataset. Additionally, they re-implemented a combination of genetic algorithm, statistical inference, and RL introduced by Tariq et al. [41] for XSS attack detection.

Our experimental results, as depicted in Table 4 and Figure 7, demonstrate that our proposed methods, along with the method presented by Mokbal et al. [22], achieved the highest accuracy and precision rates. Furthermore, our proposed methods, including RF, ensemble learning models RF and GB with DTs, and MLP with RF, as well as the method from Tariq et al. [41], attained the best recall rates, ranging from 99.54% to 99.77%. These results signify the robustness and effectiveness of our proposed methods in detecting XSS attacks. Among the three proposed models, RF achieved exceptional results in terms of accuracy, precision, and F1-Score; however, in terms of recall, the ensemble learning models RF and GB with DTs achieved higher results. In general, the three proposed models yielded higher results in all evaluation metrics than the existing studies.

Table 4. Results comparison with state-of-the-art methods.

Author	Methodology	Features		Evaluation Metrics			
	Algorithms	Feature Selection Method	No. of Selected Features	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Mokbal et al. [22] (2021)	XGBoost	Hybrid (IG and SBS)	30	99.59	99.50	99.01	99.27
Thajeel et al. [42] (2023)	DTs	Dynamic	167	98.81	98.16	97.70	97.84
Tariq et al. [41,42] (2023)	Genetic algorithm, statistical inference, and reinforcement learning	-	167	95.38	95.93	99.54	95.20
Our proposed models	RF DT and RF with GB MLP with RF	IG	25	99.78 99.76 99.65	99.80 99.74 99.59	99.75 99.77 99.71	99.78 99.76 99.65

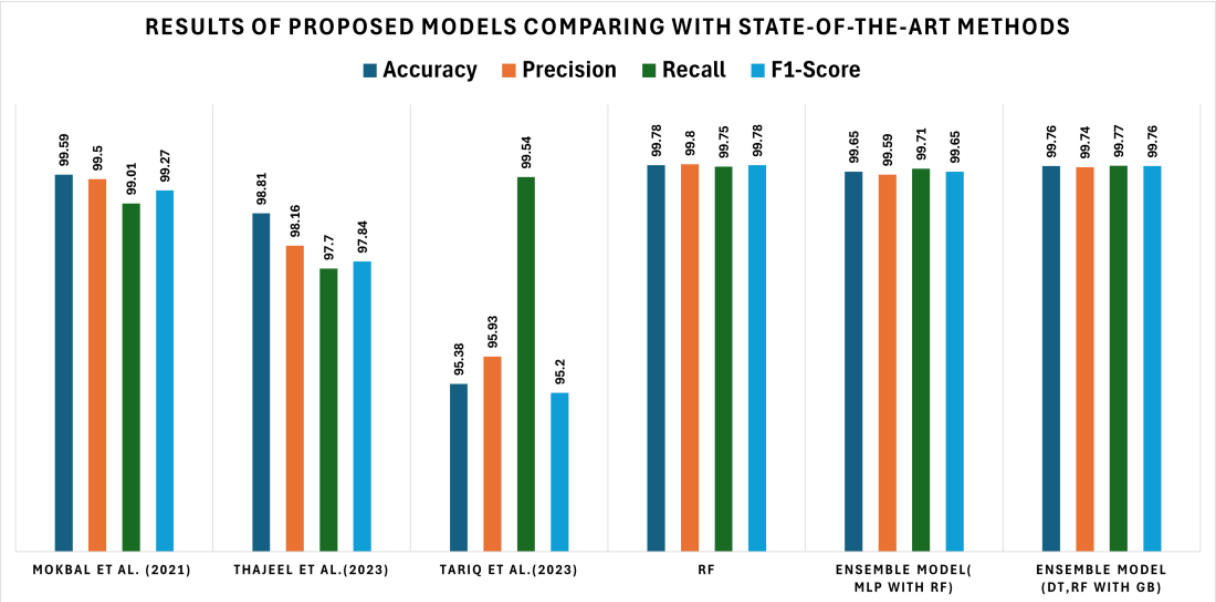

Figure 7. Results comparison with state-of-the-art methods [22,41,42].

Overall, our findings underscore the advancements made in XSS attack detection through ML techniques, emphasizing the importance of algorithm selection and feature

engineering in developing accurate and robust detection models. By achieving superior performance compared to existing methods, our research contributes significantly to enhancing web security measures and mitigating cyber threats effectively.

5.2. Practical Implementation Challenges in Real-World Systems

The dynamic nature of XSS attack vectors typically exceeds the capabilities of traditional rule-based detection systems, requiring the adoption of more advanced techniques like ML. However, applying ML models to identify XSS attacks in real-world systems has unique practical challenges. This section delves further into these challenges and offers several ways to deal with them.

It is challenging to collect high-quality labeled data due to data scarcity and unbalanced datasets; hence, crowdsourcing, data augmentation, and synthetic generation of data are used for labeling. Additionally, due to the dynamic nature of XSS attacks, feature engineering has to evolve to include regular updates, the study of user behavior, and advanced techniques such as natural language processing and deep learning.

Moreover, model complexity and performance must be balanced during model training and selection, and overfitting must be controlled via cross-validation and regularization. Furthermore, real-time detection has latency and scalability challenges, which may be addressed by model enhancement, edge computing, and distributed processing. Also, for minimal operational disturbance, modular design, API-based solutions, and gradual deployment are necessary for integrating detection models with current systems.

Additionally, the mitigation of adversarial threats requires ongoing learning and adversarial training. Moreover, managing FP and FN includes modifying judgment thresholds, creating post-processing rules, and incorporating anomaly detection. Finally, it is imperative to handle regulatory and privacy concerns by means of data anonymization, consent management, and frequent audits to ensure compliance with legal frameworks.

Overall, to design successful XSS detection systems, a comprehensive approach that combines technological solutions, ongoing monitoring, and regulatory compliance is necessary.

6. Conclusions

In conclusion, the escalating complexity of XSS attacks underscores the urgency for robust detection mechanisms. As attackers increasingly employ obfuscation techniques to evade detection, traditional methods struggle to keep pace. However, leveraging ML proves to be a potent strategy in combating this evolving threat landscape. ML models offer a more resilient and dynamic protection against malicious code injections because they can learn from enormous volumes of data, adjust to changing attack patterns, and generalize to uncommon circumstances.

This research proposes a unique framework model for XSS attack detection using a comprehensive suite of ML algorithms. Our experimentation encompassed DTs, SVMs, RF, LR, XGboost, MLP, CNNs, ANNs, and ensemble learning techniques. Through rigorous evaluation of a dataset of recent real-world traffic, augmented by feature selection methods like IG and ANOVA, we identified the most effective models.

Among the ten models examined, the RF model emerged as the top performer, achieving an accuracy score of 99.78%. Additionally, ensemble models combining RF with DTs and GB, as well as ensemble models integrating RF with MLP, demonstrated high accuracy scores of 99.76% and 99.65%, respectively, alongside robust performance across various evaluation criteria. Notably, these proposed models outperformed previous state-of-the-art methods, effectively detecting XSS-based attacks while minimizing FPs and FNs.

Looking ahead, our future work will focus on further enhancing these top-performing models to detect other types of web application attacks, such as SQL injection. By continuing to innovate and refine our approach, we aim to fortify web security measures and stay ahead of emerging cyber threats in an ever-evolving digital landscape. The ML models will be implemented using a large-scale dataset which will improve overall data protection, decrease FPs, and improve real-time monitoring with their smart, adaptable,

and effective detection capabilities. They maximize resource use, promote creativity, and may be incorporated into larger security policies. In the end, the ML models are a big step forward in safeguarding sensitive data and web applications from SQL injection attacks and other online threats.

Author Contributions: Conceptualization, R.A. and M.A.; methodology, R.A. and M.A.; formal analysis, R.A. and M.A.; investigation, R.A. and M.A.; writing—original draft, R.A.; visualization, R.A.; supervision, M.A.; funding acquisition, M.A. All authors have read and agreed to the published version of the manuscript.

Funding: The authors extend their appreciation to Taif University, Saudi Arabia, for supporting this work through project number (TU-DSPP-2024-286).

Data Availability Statement: This work utilizes the freely accessible XSS dataset that can be found in [22].

Acknowledgments: The authors extend their appreciation to Taif University, Saudi Arabia, for supporting this work through project number (TU-DSPP-2024-286).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Sotnik, S.; Shakurova, T.; Lyashenko, V. Development Features Web-Applications. 2023. Available online: www.ijeais.org/ijaar (accessed on 13 June 2024).
2. Prasetio, D.A.; Kusrini, K.; Arief, M.R. Cross-site Scripting Attack Detection Using Machine Learning with Hybrid Features. *J. Infotel* **2021**, *13*, 1–6. [CrossRef]
3. Bielova, N. Survey on JavaScript security policies and their enforcement mechanisms in a web browser. *J. Log. Algebr. Program.* **2013**, *82*, 243–262. [CrossRef]
4. Dasgupta, D.; Akhtar, Z.; Sen, S. Machine learning in cybersecurity: A comprehensive survey. *J. Def. Model. Simul.* **2022**, *19*, 57–106. [CrossRef]
5. Chaudhari, G.R.; Vaidya, M.V. A Survey on Security and Vulnerabilities of Web Application. 2014. Available online: www.ijcsit.com (accessed on 13 June 2024).
6. Parashar, P.; Srivastava, P. An Analysis of XSS Vulnerabilities and Prevention of XSS Attacks in Web Applications. Available online: https://www.researchgate.net/publication/371724261_An_Analysis_of_XSS_Vulnerabilities_and_Prevention_of_XSS_Attacks_in_Web_Applications (accessed on 3 January 2024).
7. Nir, O. "OWASP Top Ten 2023—The Complete Guide", Reflectiz. Available online: https://www.reflectiz.com/blog/owasp-top-ten-2023/ (accessed on 9 October 2023).
8. Kaur, J.; Garg, U.; Bathla, G. Detection of cross-site scripting (XSS) attacks using machine learning techniques: A review. *Artif. Intell. Rev.* **2023**, *56*, 12725–12769. [CrossRef]
9. Edgescan. Vulnerability Statistics Snapshot. January 2022. Available online: https://www.edgescan.com/january-2022-vulnerability-statistics-snapshot/ (accessed on 10 August 2023).
10. Erşahin, B.; Erşahin, M. Web application security. *South Fla. J. Dev.* **2022**, *3*, 4194–4203. [CrossRef]
11. Awad, M.; Ali, M.; Takruri, M.; Ismail, S. Security vulnerabilities related to web-based data. *Telkomnika (Telecommun. Comput. Electron. Control)* **2019**, *17*, 852–856. [CrossRef]
12. Habibi, G.; Surantha, N. *XSS Attack Detection with Machine Learning and n-Gram Methods*; Institute of Electrical and Electronics Engineers: Los Alamitos, CA, USA, 2020.
13. Sarker, I.H. Multi-aspects AI -based modeling and adversarial learning for cybersecurity intelligence and robustness: A comprehensive overview. *Secur. Priv.* **2023**, *6*, e295. [CrossRef]
14. Stency, V.S.; Mohanasundaram, N. A Study on XSS Attacks: Intelligent Detection Methods. In *Journal of Physics: Conference Series, Volume 1767, International E-Conference on Data Analytics, Intelligent Systems and Information Security & ICDIIS 2020, Pollachi, India, 11–12 December 2020*; IOP Publishing Ltd.: Bristol, UK, 2021. [CrossRef]
15. Marashdih, A.W.; Zaaba, Z.F.; Suwais, K.; Mohd, N.A. Web application security: An investigation on static analysis with other algorithms to detect cross site scripting. *Procedia Comput. Sci.* **2019**, *161*, 1173–1181. [CrossRef]
16. Cheah, C.S.; Selvarajah, V. A Review of Common Web Application Breaching Techniques (SQLi, XSS, CSRF). In Proceedings of the 3rd International Conference on Integrated Intelligent Computing Communication & Security (ICIIC 2021), Bangalore, India, 6–7 August 2021.
17. Liu, M.; Zhang, B.; Chen, W.; Zhang, X. A Survey of Exploitation and Detection Methods of XSS Vulnerabilities. *IEEE Access* **2019**, *7*, 182004–182016. [CrossRef]
18. Rodríguez, G.E.; Torres, J.G.; Flores, P.; Benavides, D.E. Cross-site scripting (XSS) attacks and mitigation: A survey. *Comput. Netw.* **2020**, *166*, 106960. [CrossRef]

19. Hickling, J. What Is DOM XSS and Why Should You Care? *Comput. Fraud Secur.* **2021**, *4*, 6–10. [CrossRef]
20. Panwar, P.; Mishra, H.; Patidar, R. An Analysis of the Prevention and Detection of Cross Site Scripting Attack. *Int. J. Emerg. Trends Eng. Res.* **2023**, *11*, 30–34. [CrossRef]
21. Kascheev, S.; Olenchikova, T. The Detecting Cross-Site Scripting (XSS) Using Machine Learning Methods. In Proceedings of the 2020 Global Smart Industry Conference, GloSIC 2020, Chelyabinsk, Russia, 17–19 November 2020; Institute of Electrical and Electronics Engineers Inc.: Los Alamitos, CA, USA, 2020; pp. 265–270. [CrossRef]
22. Mokbal, F.M.M.; Dan, W.; Xiaoxi, W.; Wenbin, Z.; Lihua, F. XGBXSS: An Extreme Gradient Boosting Detection Framework for Cross-Site Scripting Attacks Based on Hybrid Feature Selection Approach and Parameters Optimization. *J. Inf. Secur. Appl.* **2021**, *58*, 102813. [CrossRef]
23. Thajeel, I.K.; Samsudin, K.; Hashim, S.J.; Hashim, F. Machine and Deep Learning-based XSS Detection Approaches: A Systematic Literature Review. *J. King Saud Univ.—Comput. Inf. Sci.* **2023**, *35*, 101628. [CrossRef]
24. Banerjee, R.; Baksi, A.; Singh, N.; Bishnu, S.K. Detection of XSS in web applications using Machine Learning Classifiers. In Proceedings of the 2020 4th International Conference on Electronics, Materials Engineering and Nano-Technology, IEMENTech 2020, Kolkata, India, 2–4 October 2020; Institute of Electrical and Electronics Engineers Inc.: Los Alamitos, CA, USA, 2020. [CrossRef]
25. Gogoi, B.; Ahmed, T.; Saikia, H.K. Detection of XSS Attacks in Web Applications: A Machine Learning Approach. *Int. J. Innov. Res. Comput. Sci. Technol.* **2021**, *9*, 1–10. [CrossRef]
26. Stiawan, D.; Bardadi, A.; Afifah, N.; Melinda, L.; Heryanto, A.; Septian, T.W.; Idris, M.Y.; Subroto, I.M.; Budiarto, R. An Improved LSTM-PCA Ensemble Classifier for SQL Injection and XSS Attack Detection. *Comput. Syst. Sci. Eng.* **2023**, *46*, 1759–1774. [CrossRef]
27. RKadhim, W.; Gaata, M.T. A hybrid of CNN and LSTM methods for securing web application against cross-site scripting attack. *Indones. J. Electr. Eng. Comput. Sci.* **2020**, *21*, 1022–1029. [CrossRef]
28. Buz, B.; Gülçiçek, B.; Bahtiyar, Ş. A Hybrid Machine Learning Model to Detect Reflected XSS Attack. *Balk. J. Electr. Comput. Eng.* **2021**, *9*, 235–241. [CrossRef]
29. Melicher, W.; Fung, C.; Bauer, L.; Jia, L. Towards a lightweight, hybrid approach for detecting DOM XSS vulnerabilities with machine learning. In Proceedings of the Web Conference 2021—Proceedings of the World Wide Web Conference, WWW 2021, Ljubljana, Slovenia, 12–16 April 2021; Association for Computing Machinery, Inc.: New York, NY, USA, 2021; pp. 2684–2695. [CrossRef]
30. Lamrani Alaoui, R.; Habib Nfaoui, E. Cross Site Scripting Attack Detection Approach Based on LSTM Encoder-Decoder and Word Embeddings. 2023. Available online: www.ijisae.org (accessed on 13 June 2024).
31. Gupta, C.; Singh, R.K.; Mohapatra, A.K. GeneMiner: A Classification Approach for Detection of XSS Attacks on Web Services. *Comput. Intell. Neurosci.* **2022**, *2022*, 3675821. [CrossRef]
32. Dawadi, B.R.; Adhikari, B.; Srivastava, D.K. Deep Learning Technique-Enabled Web Application Firewall for the Detection of Web Attacks. *Sensors* **2023**, *23*, 2073. [CrossRef]
33. Tian, Z.; Luo, C.; Qiu, J.; Du, X.; Guizani, M. A Distributed Deep Learning System for Web Attack Detection on Edge Devices. *IEEE Trans. Ind. Inf.* **2020**, *16*, 1963–1971. [CrossRef]
34. Chaudhary, P.; Gupta, B.B.; Chang, X.; Nedjah, N.; Chui, K.T. Enhancing big data security through integrating XSS scanner into fog nodes for SMEs gain. *Technol. Forecast. Soc Chang.* **2021**, *168*, 120754. [CrossRef]
35. Luo, C.; Tan, Z.; Min, G.; Gan, J.; Shi, W.; Tian, Z. A Novel Web Attack Detection System for Internet of Things via Ensemble Classification. *IEEE Trans. Ind. Inf.* **2021**, *17*, 5810–5818. [CrossRef]
36. Odun-Ayo, I.; Toro-Abasi, W.; Adebiyi, M.; Alagbe, O. An implementation of real-time detection of cross-site scripting attacks on cloud-based web applications using deep learning. *Bull. Electr. Eng. Inform.* **2021**, *10*, 2442–2453. [CrossRef]
37. Lei, L.; Chen, M.; He, C.; Li, D. XSS Detection Technology Based on LSTM-Attention. In Proceedings of the 2020 5th International Conference on Control, Robotics and Cybernetics, CRC 2020, Wuhan, China, 16–18 October 2020; Institute of Electrical and Electronics Engineers Inc.: Los Alamitos, CA, USA, 2020; pp. 175–180. [CrossRef]
38. Tan, X.; Xu, Y.; Wu, T.; Li, B. Detection of Reflected XSS Vulnerabilities Based on Paths-Attention Method. *Appl. Sci.* **2023**, *13*, 7895. [CrossRef]
39. Zhang, X.; Zhou, Y.; Pei, S.; Zhuge, J.; Chen, J. Adversarial Examples Detection for XSS Attacks Based on Generative Adversarial Networks. *IEEE Access* **2020**, *8*, 10989–10996. [CrossRef]
40. Alaoui, R.L.; Nfaoui, E.H. Generative Adversarial Network-Based Approach for Automated Generation of Adversarial Attacks Against a Deep-Learning Based XSS Attack Detection Model. 2023. Available online: www.ijacsa.thesai.org (accessed on 13 June 2024).
41. Tariq, I.; Sindhu, M.A.; Abbasi, R.A.; Khattak, A.S.; Maqbool, O.; Siddiqui, G.F. Resolving cross-site scripting attacks through genetic algorithm and reinforcement learning. *Expert Syst. Appl.* **2021**, *168*, 114386. [CrossRef]
42. Thajeel, I.K.; Samsudin, K.; Hashim, S.J.; Hashim, F. Dynamic feature selection model for adaptive cross site scripting attack detection using developed multi-agent deep Q learning model. *J. King Saud Univ.—Comput. Inf. Sci.* **2023**, *35*, 101490. [CrossRef]
43. Van Den Bergh, D.; van Doorn, J.; Marsman, M.; Draws, T.; van Kesteren, E.-J.; Derks, K.; Dablander, F.; Gronau, Q.F.; Kucharský, Š.; Gupta, A.R.K.N.; et al. A tutorial on conducting and interpreting a bayesian ANOVA in JASP. *Annee Psychol.* **2020**, *120*, 73–96. [CrossRef]

44. Omuya, E.O.; Okeyo, G.O.; Kimwele, M.W. Feature Selection for Classification using Principal Component Analysis and Information Gain. *Expert Syst. Appl.* **2021**, *174*, 114765. [CrossRef]
45. Khyat, J.; Chitra, S. Feature Selection Methods for Improving Classification Accuracy-A Comparative Study. *UGC Care Group I Listed J.* **2020**, *10*, 1.