Highly Accurate XSS Detection using CatBoost

Abdulkader Hajjouz, Elena Avksentieva

***Abstract* — Cross-Site Scripting (XSS) is still a big security threat to web and user data. We need advanced detection mechanisms to protect web applications. This paper presents a new machine learning framework for XSS detection. We use hierarchical feature selection with Spearman correlation to reduce feature dimension and improve model interpretability, and CatBoostClassifier, a gradient boosting algorithm known for its robustness and performance. We tested our CatBoost-based model on a large dataset, and it achieved 99.88% accuracy, 1.00 ROC AUC, 1.00 Average Precision, and 0.9974 Matthews Correlation Coefficient. Compared to existing XSS detection methods, our proposed framework outperforms the benchmark models on all metrics. Feature importance and SHAP value analysis also show the important features for XSS classification. This paper proves our integrated approach is effective and a good solution for XSS mitigation in web applications.**

***Keywords*— Cross-Site Scripting, Machine Learning, Feature Selection, Web Security.**

I. INTRODUCTION

In the ever-changing landscape of cyber threats, Cross-Site Scripting (XSS) remains a big challenge to web security [1]. These attacks exploit weaknesses in web applications, inject malicious scripts, and put data integrity and users' safety at global risk [2]. The complexity of those attacks ranges from simple scripting exploits to complex, obfuscated injection that evades traditional detection mechanisms [3]. So, we need detection strategies that are both versatile and precise. Hence, we are seeing an increasing reliance on advanced Machine Learning (ML) to mitigate these threats [4]. Web application vulnerabilities are part of cyber threats and affect businesses, governments, and organizations. Despite the widespread deployment of Web Application Firewalls (WAF), these are not foolproof, often because developers are overlooking vulnerabilities at the coding stage—a reflection of inadequate security protocols [2]. Recently, we have seen many incidents where such vulnerabilities were exploited, resulting in high-profile attacks on major social media like Weibo and Twitter, where attackers got many user's identity tokens through XSS vulnerabilities [5–6]. These incidents show how common XSS attacks are and how fast they can propagate across systems, making the security landscape more complex. Although WAF can intercept some network attacks, more than relying on WAF technology for web application security is required. Insights from HackerOne, a leading hacker platform, showed that XSS vulnerabilities were the most reported issue until 2021, around 23% of all vulnerabilities [7]. Also, according to Acunetix's 2021 security report, XSS is in the top 3 high-severity threats, with a 0.5% increase from last year [8]. This proves that XSS is still a big risk to web security, and we need to continue researching in this area. Our research contributes to this by improving XSS detection using machine learning models that balance simplicity and accuracy. To illustrate the different approaches in XSS detection, previous research has tried various methods. For example, authors in [9] used Deep Learning techniques, word2vec, and LSTM. Authors in [10] used a crawler-based approach, combining web crawling with injection testing. Authors in [11] used Machine Learning and Statistical methods: LSTM, CNN, AdaBoost, SVM, and Random Forest. Authors in [12–13] used Machine Learning algorithms: SVM, k-NN, and Random Forest, along with Neural Networks and Explainable AI (XAI). Authors in [14] showed the effectiveness of Decision Trees in Machine Learning. Authors in [16] combined different approaches: Random Forest, Logistic Regression, and k-NN with CSP, IDS, IPS, and WAF. These different results show that machine learning can be used to improve XSS detection.

Manuscript received March 24, 2025.

Abdulkader Hajjouz is with the National Research University ITMO, Saint Petersburg, 191002 Russia (phone: +79693483331; e-mail: hajjouz@itmo.ru).

Elena Avksentieva is with the National Research University ITMO, Saint Petersburg, 191002 Russia (e-mail: eavksenteva@itmo.ru).

II. METHOLOGY

A. Attack Scenarios

XSS detection requires an understanding of all the different attack scenarios, which vary greatly in complexity, exploitability, and impact [2,16–17]. These scenarios include reflected XSS, where malicious scripts execute directly from a URL and pose an immediate threat [18]; stored XSS, where injected scripts persist in a website's database, activate when accessed, and require advanced detection methods because of their latent nature [19]; and DOM-based XSS, which involves real-time manipulation of the client-side DOM without server interaction and can evade traditional security controls [20].

B. Overview Of The Dataset

Our study uses a large dataset, mainly from [12] and augmented with CSIC 2010 data [21], which consists of 43,218 files (28,068 benign, 9,068 plaintext from [23], and 15,150 malicious). This dataset has 65 features (HTML tags, JavaScript events, and metadata for XSS exploits) and is necessary to train XSS detection models that can distinguish between good and bad content. To test thoroughly, the data was divided into non-overlapping training (19,122 instances: 5,150 malicious, 13,972 benign), validation (12,048 instances: 5,000 malicious, 7,048 benign), and testing (12,048 instances: 5,000 malicious, 7,048 benign) sets. This dataset reflects the complexities of real-world scripting and

XSS attacks [8, 23–24] and is necessary to train a model that can work with different XSS scenarios and be resilient to emerging threats.

C. Stratified Sampling And Class Balancing

To train and evaluate our models robustly, we used stratified sampling to split our dataset into training, validation, and test sets, each with the same class distribution as the original. Since we knew the class imbalance, especially in the training set, which had 13,972 benign and 5,150 malicious samples, we used the Synthetic Minority Over-sampling Technique for Nominal and Continuous (SMOTENC) to generate synthetic samples for the minority (malicious) class. This balanced the training set to 13,972 benign and 13,972 malicious samples, the validation set to 7,048 benign and 5,000 malicious samples, and the test set remained the same as the original. As recommended in [25], this is important to prevent model bias and to achieve fair and accurate intrusion detection by ensuring both training and testing phases reflect the diversity of the dataset.

D. Hierarchical Feature Selection Using Spearman Correlation

To tackle the problems of high dimensionality and potential multicollinearity in our initial 65 features, we used a rigorous Hierarchical Feature Selection methodology based on Spearman's rank correlation coefficient. This method offers a statistical way to identify and manage feature redundancy by quantifying the monotonic relationship between feature rankings, so it captures non-linear relationships that linear correlation won't. The initial step was to compute the Spearman correlation matrix for all feature pairs in our dataset. This matrix contains the degree and direction of the monotonic relationship between each pair of variables. Then, we conducted Agglomerative Hierarchical Clustering, utilizing the Spearman correlation coefficients (converted into a dissimilarity measure), to group features that have strong statistical similarity. The resulting hierarchical structure is visualized through a dendrogram, which shows the inter-feature relationships based on their rank-order correlations. This hierarchical structure allows us to identify clusters of features where the underlying variability is almost shared. The critical step in this process is to select one feature from each cluster. The selection criterion is to preserve as much information and variance as possible from the whole cluster while reducing the dimensionality of the feature space as much as possible. By focusing the subsequent modeling on a more sparse and independent set of features, this hierarchical selection methodology, based on Spearman correlation, is designed to speed up our machine learning models so they converge faster during training and are faster during inference. Also, the dimensionality reduction can help with generalization by reducing the risk of overfitting to redundant or less informative features, so our XSS detection system will be more robust and accurate.

Figure 1 shows the feature correlation before and after our feature selection. The top heatmap is the Spearman correlation matrix for the initial 65 features. Red is positive correlation and blue is negative correlation, which means potential redundancy. After removing highly correlated features through our hierarchical approach, the bottom heatmap is the correlation matrix for the 46 features. Figure 1 demonstrates that the color-coded cells show a significant reduction in strong correlations (red and blue are less intense and more white areas), which means we successfully reduced the feature redundancy. The heatmap is showing the Spearman correlation coefficients among the remaining features, which means they are independent. This reduction in correlations means we streamlined our model and we have a more efficient and robust final feature set to detect XSS vulnerabilities accurately and reliably without sacrificing the predictive power.

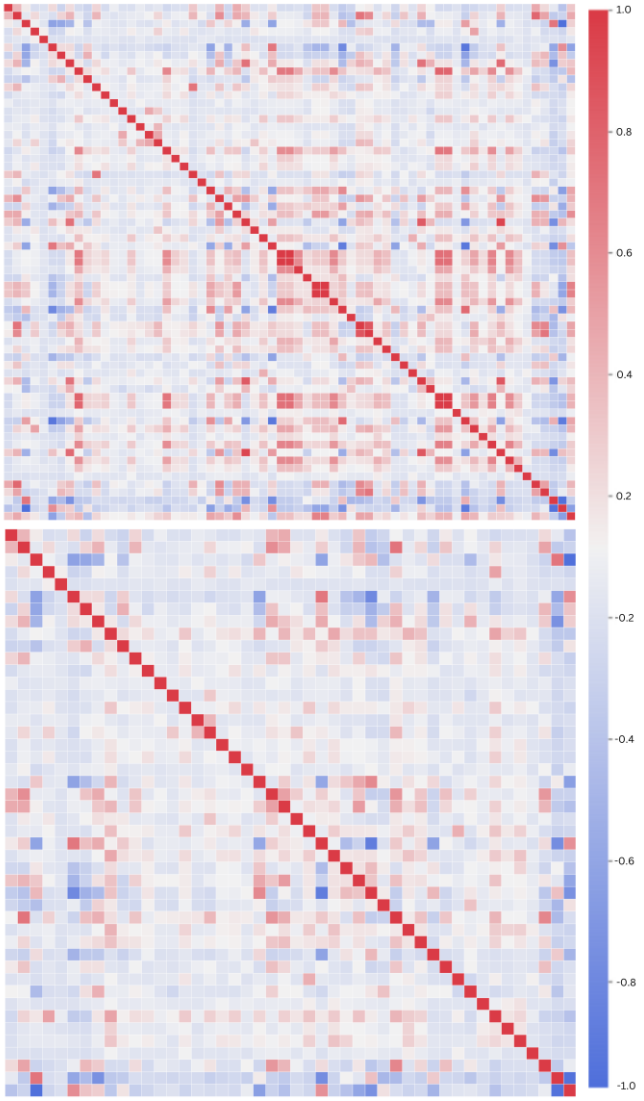

Figure 1: Correlation Heatmaps Before and After Feature Selection. (Top): Initial feature set correlation. (Bottom): Correlation after hierarchical feature selection. Spearman correlation coefficients are visualized with a red-white-blue color scale.

E. Understanding CatBoost and Grid Search Optimization for Enhanced XSS Detection

CatBoost, a gradient boosting framework over decision trees, is great with complex tabular data and multi-class classification, so it's perfect for high-accuracy tasks like attack detection, especially with imbalanced datasets [26]. Its Ordered Boosting mechanism prevents overfitting and prediction bias by training trees on ordered data, and Ordered Statistics allows to handle categorical features without one-hot encoding. For multi-class problems,

CatBoost minimizes the Cross-Entropy loss function, which measures the difference between predictions and actual values, and iteratively refines the model. The Softmax function converts these predictions into probabilities so that they sum up to one, and the highest probability is the final class. CatBoost performance is very dependent on hyperparameters: number of iterations (trees), learning rate, tree depth, L2 leaf regularization, min data in leaf, random strength, grow policy (e.g., SymmetricTree, Depthwise, Lossguide), overfitting detection parameters (od_type, od_wait), and bootstrap techniques (Bernoulli with subsample rates 0.66 and 0.8, Bayesian). To tune these parameters, we used grid search strategy; we carefully explored a predefined hyperparameter space [27]. We trained and evaluated CatBoostClassifier on GPU for computational efficiency for different combinations of these parameters: limited number of iterations (500) to prevent overfitting, learning rates 0.13 and 0.1 for controlled convergence, depth 3 for simplicity, L2 regularization 1 and 3 for generalization, min data in leaf 1 and 5 for robust node support, random strength 1–10 for learning diversity, and different grow policies to fine-tune tree expansion.

III. Results

To see how our CatBoostClassifier model did with Cross-Site Scripting (XSS), we looked at the confusion matrix in Figure 2. The confusion matrix shows the model predictions against the actual labels of the test dataset, with 4 main metrics: True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN). In our case, True Negatives are the benign samples (no XSS attack) and True Positives are the malicious samples (XSS attacks). False Positives are benign samples classified as malicious (Type I error), and False Negatives are malicious samples classified as benign (Type II error).

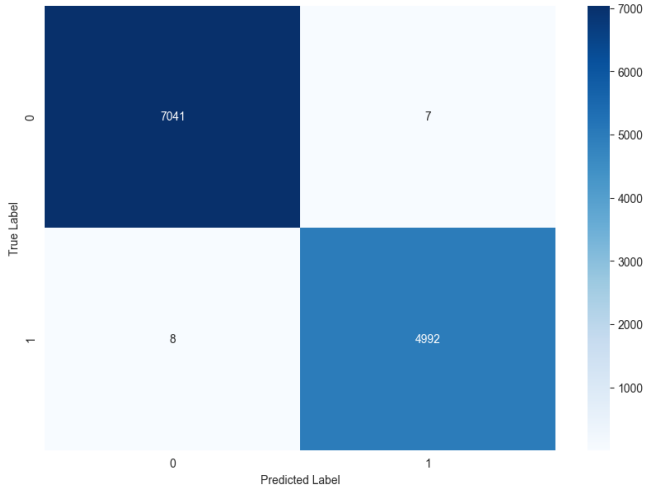

Figure 2: Confusion Matrix of the CatBoostClassifier Model on the Test Set.

In Figure 2, we see the confusion matrix for the CatBoostClassifier on the test set: 7041 TN (labeled benign as benign), 7 FP (labeled benign as malicious), 8 FN (labeled malicious as benign), 4992 TP (labeled malicious as malicious). This gives us an initial view of how the model is doing; it looks like a lot of correct classification for both benign and malicious samples and not many false.

Figure 3 shows training and validation loss curves for CatBoostClassifier training. As we can see, both training loss (blue) and validation loss (orange) go down and converge to a low value with number of iterations. This is a good sign that the model learns from data and generalizes well to unseen data (validation loss is similar to training loss). And validation loss follows training loss closely, which means the model is not overfitting.

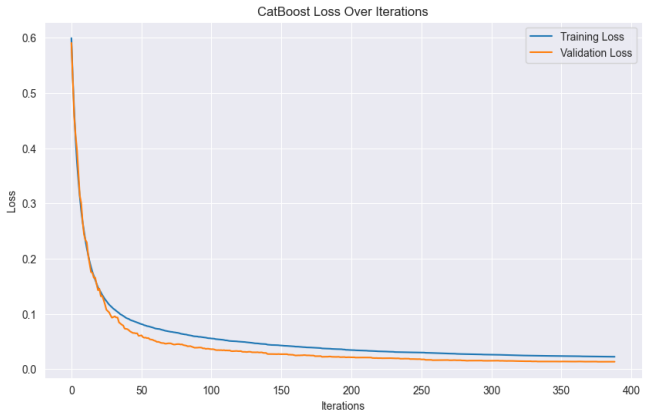

Figure 3: CatBoost Training and Validation Loss Over Iterations.

To measure how well our CatBoostClassifier model worked, we calculated key metrics from the confusion matrix, looking at both class-specific and overall performance. For benign samples (Class 0), the model had a precision of 99.89%; this means 99.89% of the samples it classified as benign were actually benign, and we minimized false alarms. The recall for benign samples was 99.90%; the model is very good at finding almost all the actual benign traffic. The F1-score for Class 0 was 99.89%; this means the model is robust in classifying benign traffic, and we had 7048 actual benign samples in the test set. For malicious samples (Class 1), which are XSS attacks, the precision was 99.86%; this is important to minimize false positives and security alerts. The recall for malicious samples was 99.84%; the model is good at finding almost all the real XSS threats, which is very important for security. The F1-score for Class 1 was 99.85%; this means a good balance between precision and recall in malicious sample detection, and we had 5000 actual malicious samples. Overall, the model had an accuracy of 99.88%; this means 99.88% of all the samples in the test set were correctly classified. These very high numbers for precision, recall, and F1-score for both classes and overall accuracy mean the CatBoostClassifier model is very good at detecting XSS vulnerabilities.

To evaluate our CatBoostClassifier more thoroughly, we looked at the Receiver Operating Characteristic (ROC) curve and the Precision-Recall (PR) curve in Figure 4 and Figure 5, respectively. These curves show how the model performs at different threshold settings.

Figure 4 displays the overall ROC curve for our CatBoostClassifier. The ROC curve compares the True Positive Rate (Sensitivity) against the False Positive Rate (1 - Specificity) at different categorisation thresholds. The Area Under the ROC Curve (AUC) is a single statistic that represents the performance of the classifier across all thresholds. An AUC of 1.0 represents a perfect classifier.

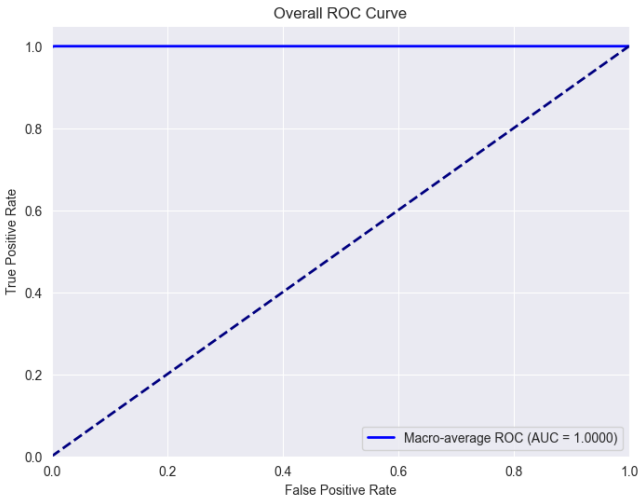

Figure 4: Overall ROC Curve for CatBoostClassifier.

As shown in Figure 4, our model gets a Macro-average ROC AUC of 1.0000. This means the CatBoostClassifier is perfect at separating benign and malicious samples. A perfect AUC of 1.0 means the model can perfectly separate the positive and negative classes (malicious and benign in our case), with no overlapping in the distributions of their predicted scores. In XSS detection terms, this means the model can get perfect balance between detecting actual XSS attacks (high True Positive Rate) and low false alarms (low False Positive Rate) at all thresholds.

Figure 5 shows the Precision-Recall curve for our multi-class classification problem for Class 0 (Benign) and Class 1 (Malicious). The curve plots Precision against Recall at different thresholds. The Average Precision (AP) is the weighted mean of precisions at each threshold, with the increase in recall from the previous threshold as the weight. Higher AP means better; 1.0 is the maximum AP.

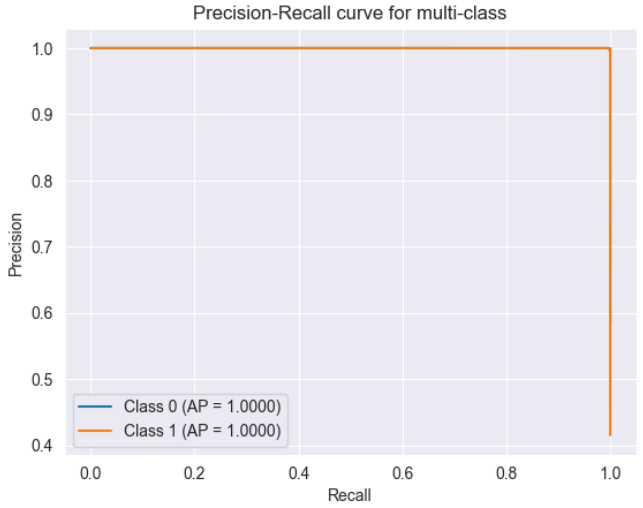

Figure 5: Precision-Recall Curve for CatBoostClassifier.

As shown in Figure 5, our CatBoostClassifier has an Average Precision (AP) of 1.0000 for both Class 0 (Benign) and Class 1 (Malicious). These perfect AP scores are amazing. An AP of 1.0 for both classes means the model is perfect at all thresholds. For XSS detection, this means that the model can have both high precision (minimise false positives) and high recall (minimise false negatives) simultaneously. This is critical for a security system since we need to detect as many real threats as possible (high recall) while minimising the number of false alarms that can disrupt security operations (high precision).

The perfect ROC AUC of 1.0000 and perfect Average Precision scores of 1.0000 for both classes, as seen in the ROC and Precision-Recall curves, is visual proof of the excellent performance of our CatBoostClassifier. These results confirm the findings from the confusion matrix and numerical metrics, we can see our model is very good at detecting XSS with perfect balance of precision and recall. The visual from these curves combined with the numbers is strong evidence of our proposed method for XSS detection.

To prove the strength and stability of our CatBoostClassifier model, we ran an analysis using the Matthews Correlation Coefficient (MCC), which is a balanced metric for binary classification. Our MCC values are very high: Class 0 (benign samples) and Class 1 (malicious samples/XSS attacks) both are 0.9974, overall MCC is 0.9974. These near-perfect MCC values mean the predicted and actual classifications are almost perfect for both benign and malicious traffic. The model is very good at minimizing errors for both classes. This is also backed up by our previous findings from the confusion matrix, quantitative metrics, and curve analysis; this is another evidence that our proposed CatBoost-based methodology is working well for XSS vulnerability detection.

We did a feature importance analysis to see what's inside our CatBoostClassifier and which features are most important in detecting XSS. CatBoost has a built-in way to calculate feature importances based on how much each feature contributes to the model's decision. Figure 6 shows the top features as per the CatBoostClassifier model. The importance is quantified as a percentage, and the features are ranked in descending order so we can easily see which features are most important in XSS detection.

The feature importance is shown in Figure 6. As we can see, there is a clear hierarchy of features for Cross-Site Scripting (XSS) detection. The top feature, "Contains Less Than," which means the presence of '<', is a strong indicator of XSS attacks, followed by "Contains Question Mark," "ScriptTag," "Contains Comma," and "Numbers Ratio" in descending order. "Contains Less Than" and "ScriptTag" are the HTML and JavaScript injection, which are the basis of most XSS attacks. "Contains Question Mark" is the URL query parameter reflected XSS. "Contains Comma," "Numbers Ratio," "Contains And," "Contains Slash," "Contains Semicolon," "Contains Quotations," and "Contains Percentage" are capturing the syntactic and structural patterns of malicious scripts or encoded payloads. This hierarchy gives us an idea of how the model is working and what are the input characteristics that are indicative of XSS vulnerabilities and what patterns the model is able to detect from common XSS attack vectors. So, these findings have practical implications for feature engineering and where to focus the security efforts towards the most important indicators of malicious activity.

To see how our CatBoostClassifier decides and what features contribute to classifying samples as Class 0

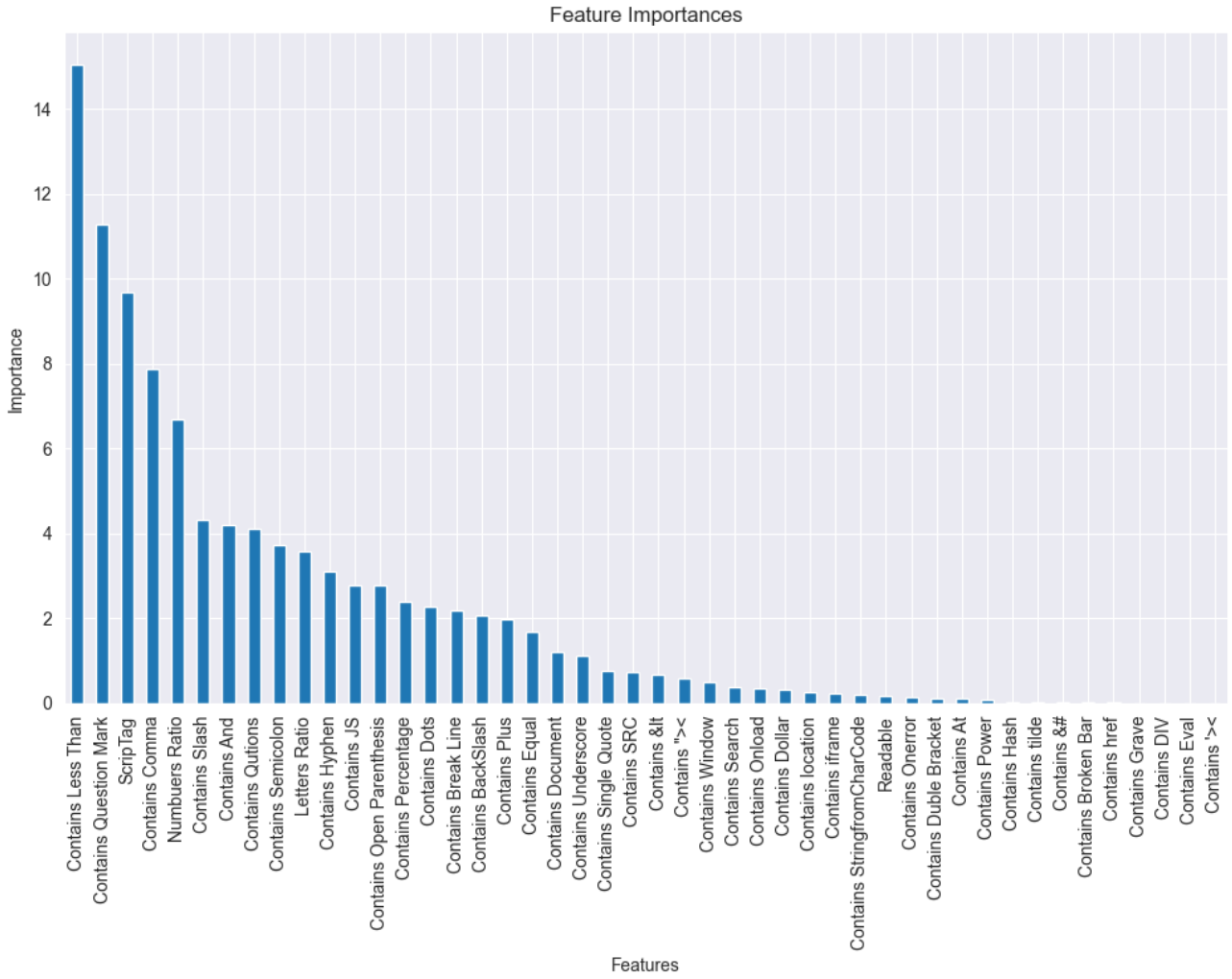

Figure 6: Feature Importances for XSS Detection Model.

(benign) or Class 1 (malicious), we did a SHAP (SHapley Additive exPlanations) value analysis. SHAP values provide a unified measure of feature importance, quantifying the marginal contribution of each feature to the model's output for individual predictions.

Figure 7 shows the SHAP values for features that affect benign samples (Class 0). Features are sorted by importance, the x-axis is SHAP value (impact on output), points are colored by feature value (red = high, blue = low).

Figure 8 shows the SHAP values for features that affect malicious samples (Class 1, XSS attacks). Same as Figure 7, features are sorted by importance, points are colored by feature value.

For Class 0 (Benign Samples) (Figure 7), the SHAP analysis shows that features like "Contains Less Than," "Contains Question Mark," and "Contains Comma" are important. For "Contains Less Than" and "Contains Question Mark," the higher feature values (red points) have negative SHAP values (to the left of the vertical center line). So, when these features are more present, they tend to decrease the chances of the sample being classified as benign, which might seem counterintuitive at first. But it could be because, while these characters are common in benign text, their absence is even more characteristic of benign content in this dataset. Features like "ScriptTag" and "Contains JS" also seem to have an impact, with varying SHAP values depending on their presence.

For Class 1 (Malicious Samples - XSS Attacks) (Figure 8), the SHAP plot is different. Features like "Contains Broken Bar", "Letters Ratio", and "Numbers Ratio" are among the most important. In this plot, the direction of feature impact is not clear from this snippet, but a full SHAP plot would show how high or low values of these features affect the prediction towards the malicious class. Note that features that are important for Class 1 might not be the opposite of those for Class 0, because distinguishing malicious from benign content is complex, and the model learns nuanced patterns.

The SHAP value analysis gives us a fine-grain view of feature contributions. SHAP plots show how each feature affects the prediction for individual samples and for each class. The fact that features like "Contains Less Than" and "Contains Question Mark" have a negative SHAP value for benign class prediction means it's a complex relationship. It's possible that for benign samples in this dataset, the absence or lower frequency of these characters is more indicative of benign content than their presence. For malicious samples, the presence of different sets of features, perhaps related to encoding or obfuscation techniques (as "Contains Broken Bar," if it refers to special characters used in encoding), becomes more important.

We compared the performance of our CatBoostClassifier

model to several existing XSS detection models as reported in previous studies, as shown in Table 1. This allows us to see how our approach fares in the bigger picture of XSS detection.

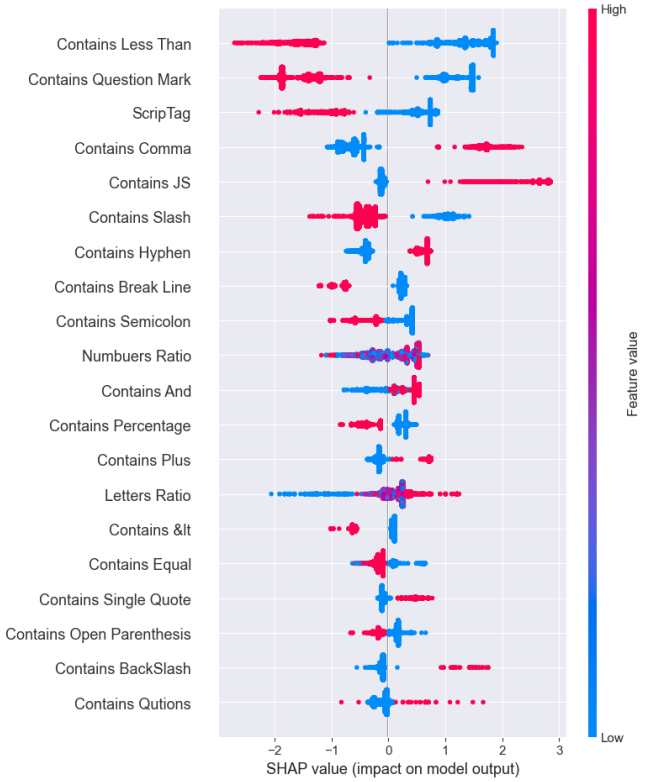

Figure 7: SHAP Value Plot for Benign Samples.

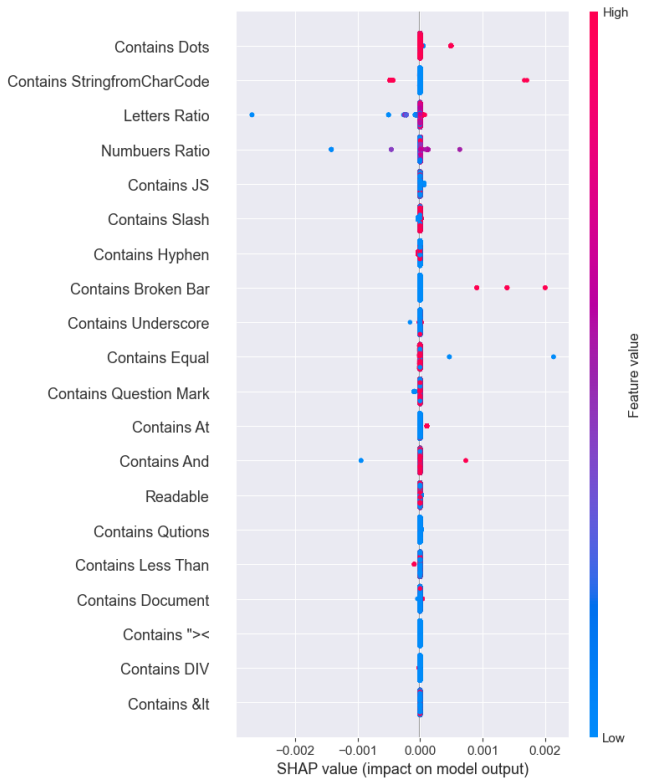

Figure 8: Feature Importances for Malicious Samples.

Table 1: Performance Comparison with Existing XSS Detection Models.

Model	**Authors**	**Accuracy**	**MCC**	**ROC AUC**	**AP**	**Precision**	**Recall**	**F1-score**
DeepXSS	[9]	N/A	N/A	0.98	N/A	99.5	0.979	98.7
ADTree		N/A	N/A	N/A	N/A	93.8	0.936	93.6
AdaBoost		N/A	N/A	N/A	N/A	94.1	0.939	93.9
Linear	[12]	96.32	N/A	N/A	N/A	98.33	94.53	N/A
Polynomial		99.60	N/A	N/A	N/A	99.69	99.22	N/A
k-NN		99.75	N/A	N/A	N/A	**99.88**	99.61	N/A
RandomForest		99.50	N/A	N/A	N/A	99.84	99.15	N/A
FNN-16	[13]	99.78	N/A	N/A	N/A	99.94	99.53	N/A
FNN-34		99.88	N/A	N/A	N/A	99.98	99.75	N/A
Decision tree	[14]	98,81	N/A	N/A	N/A	99,19	93,70	95,90
Naive Bayes		65,27	N/A	N/A	N/A	30,30	80,68	55,46
Logistic regression		83,03	N/A	N/A	N/A	39,68	0,20	39
SVM		71,37	N/A	N/A	N/A	27,82	43,21	48,06
CatBoost	**Ours**	**99,88**	**99,74**	**1,00**	**1,00**	**99,88**	**99,88**	**99,88**

Table 1 is a comparison of our CatBoostClassifier model to the other models. We used Accuracy, MCC, ROC AUC, AP, Precision, Recall, and F1-score as metrics. The best results are in bold.

As shown in Table 1, our CatBoostClassifier model beats all the other models in all the metrics. And our CatBoost model has perfect ROC AUC and Average Precision (AP), 1.00. That's the ideal score, better than the other models which don't report ROC AUC and AP, but unlikely to reach 1.00. Our model's Matthews Correlation Coefficient (MCC) is also very high, 0.9974; that means it's robust and balanced. This comparison shows our proposed method works well with hierarchical feature selection and CatBoostClassifier.

IV. CONCLUSION

In conclusion, we have tackled the long-standing issue of Cross-Site Scripting (XSS) detection by introducing a new machine learning framework that combines hierarchical feature selection with CatBoostClassifier. Our method, using Spearman correlation-based feature selection to speed up and reduce dimensionality, together with the robust CatBoost algorithm, has achieved excellent results in detecting XSS attacks. The CatBoostClassifier model got

amazing performance metrics: 99.88% accuracy, perfect ROC AUC and Average Precision 1.00, and Matthews Correlation Coefficient 0.9974, clearly outperforming the state-of-the-art XSS detection methods as shown in our comparison. This paper not only proves the effectiveness of gradient boosting methods, especially CatBoost, in cybersecurity but also sets a new standard for XSS detection. Feature and SHAP value analysis gives us more insights into the most important features for XSS vulnerabilities. Future work could be to test the model against evolving XSS attack vectors and in real-world deployment scenarios to make it more practical for web application security.

REFERENCES

[1] Nair, S. S. (2024). Securing Against Advanced Cyber Threats: A Comprehensive Guide to Phishing, XSS, and SQL Injection Defense. Journal of Computer Science and Technology Studies, 6(1), 76-93.

[2] Rodríguez, G. E., Torres, J. G., Flores, P., & Benavides, D. E. (2020). Cross-site scripting (XSS) attacks and mitigation: A survey. Computer Networks, 166, 106960.

[3] Hannousse, A., Yahiouche, S., & Nait-Hamoud, M. C. (2024). Twenty-two years since revealing cross-site scripting attacks: a systematic mapping and a comprehensive survey. Computer Science Review, 52, 100634.

[4] Kaur, J., Garg, U., & Bathla, G. (2023). Detection of cross-site scripting (XSS) attacks using machine learning techniques: a review. Artificial Intelligence Review, 56(11), 12725-12769.

[5] Chinese Twitter hit by XSS worm. 2022. https://news.softpedia.com/news/ Chinese-Twitter-Hit-by-XSS-Worm-209292.shtml. (accessed on 2 January 2024).

[6] Digging Experience | constructing twitter XSS worm from twitter's XSS vulnerability. 2022. https://www.freebuf.com/vuls/203052.html. (accessed on 2 January 2024).

[7] The 2021 hacker report. 2022. https://www.hackerone.com/resources/reporting/ the-2021-hacker-report.(accessed on 25 December 2023).

[8] Acunetix. 2021. The Invicti AppSEC Indicator Spring 2021 edition: Acunetix Web Vulnerability Report. Acunetix. Retrieved from https://www.acunetix.com/white-papers/acunetix-web-application-vulnerability-report-2021/. (accessed on 3 January 2024).

[9] Fang, Y., Li, Y., Liu, L., & Huang, C. (2018, March). DeepXSS: Cross site scripting detection based on deep learning. In Proceedings of the 2018 international conference on computing and artificial intelligence (pp. 47-51).

[10] Guan, H., Li, D., Li, H., & Zhao, M. (2022, December). A Crawler-Based Vulnerability Detection Method for Cross-Site Scripting Attacks. In 2022 IEEE 22nd International Conference on Software Quality, Reliability, and Security Companion (QRS-C) (pp. 651-655). IEEE.

[11] Kumar, J. H., & Ponsam, J. G. (2023, January). Cross site scripting (XSS) Vulnerability detection using machine learning and statistical analysis. In 2023 International Conference on Computer Communication and Informatics (ICCCI) (pp. 1-9). IEEE.

[12] Mereani, F. A., & Howe, J. M. (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.

[13] Mereani, F., & Howe, J. M. (2019). Exact and approximate rule extraction from neural networks with Boolean features. In Proceedings of the 11th International Joint Conference on Computational Intelligence (Vol. 1, pp. 424-433). SCITEPRESS-Science and Technology Publications.

[14] Kascheev, S., & Olenchikova, T. (2020, November). The detecting cross-site scripting (XSS) using machine learning methods. In 2020 global smart industry conference (GloSIC) (pp. 265-270). IEEE.

[15] Chen, H. C., Nshimiyimana, A., Damarjati, C., & Chang, P. H. (2021, January). Detection and prevention of cross-site scripting attack with combined approaches. In 2021 International conference on electronics, information, and communication (ICEIC) (pp. 1-4). IEEE.

[16] Rodríguez-Galán, G., & Torres, J. (2024). Personal data filtering: a systematic literature review comparing the effectiveness of XSS attacks in web applications vs cookie stealing. Annals of Telecommunications, 1-40.

[17] Liu, M., Zhang, B., Chen, W., & Zhang, X. (2019). A survey of exploitation and detection methods of XSS vulnerabilities. IEEE access, 7, 182004-182016.

[18] Alenzi, K. F., & Abbase, O. A. B. (2022). A Defensive Framework for Reflected XSS in Client-Side Applications. Journal of Web Engineering, 21(7), 2209-2229.

[19] Anagandula, K., & Zavarsky, P. (2020, June). An analysis of effectiveness of black-box web application scanners in detection of stored SQL injection and stored XSS vulnerabilities. In 2020 3rd International Conference on Data Intelligence and Security (ICDIS) (pp. 40-48). IEEE.

[20] Bensalim, S., Klein, D., Barber, T., & Johns, M. (2021, April). Talking about my generation: Targeted dom-based xss exploit generation using dynamic data flow analysis. In Proceedings of the 14th European Workshop on Systems Security (pp. 27-33).

[21] Giménez, C. T., Villegas, A. P., & Marañón, G. Á. (2010). HTTP data set CSIC 2010. Information Security Institute of CSIC (Spanish Research National Council), 64, 07.

[22] Wang, H., Lu, Y., & Zhai, C. (2011, August). Latent aspect rating analysis without aspect keyword supervision. In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 618-626).

[23] Rustam, F., Raza, A., Ashraf, I., & Jurcut, A. D. (2023, June). Deep ensemble-based efficient framework for network attack detection. In 2023 21st Mediterranean Communication and Computer Networking Conference (MedComNet) (pp. 1-10). IEEE.

[24] OWASP Top Ten. OWASP Foundation. Retrieved from https://owasp.org/www-project-top-ten/. (accessed on 2 January 2024).

[25] Mukherjee, M., & Khushi, M. (2021). SMOTE-ENC: A novel SMOTE-based method to generate synthetic data for nominal and continuous features. Applied system innovation, 4(1), 18.

[26] Dorogush, A. V., Ershov, V., & Gulin, A. (2018). CatBoost: gradient boosting with categorical features support. arXiv preprint arXiv:1810.11363.

[27] Syarif, I., Prugel-Bennett, A., & Wills, G. (2016). SVM parameter optimization using grid search and genetic algorithm to improve classification performance. TELKOMNIKA (Telecommunication Computing Electronics and Control), 14(4), 1502-1509.