Minimal-Feature XSS Detection by SHAP and Importance-Driven Pruning

Abdulkader Hajjouz, Elena Avksentieva

Abstract— **Cross-Site Scripting (XSS) remains a high-impact web threat despite widespread defensive mechanisms. Building on our prior study, we ask: how few features are needed to preserve near-ceiling detection quality? Starting after correlation-based de-redundancy, we iteratively prune features using CatBoost feature importance and SHAP values, retraining at each step on the same stratified, SMOTENC-balanced splits. The process yields an ultra-compact classifier using four features only. On the fixed test set, the model attains Accuracy = 0.984728, MCC = 0.9687, ROC-AUC = 0.9932, AP ≈ 0.99, Precision (macro) = 0.986362, Recall (macro) = 0.982326, and F1 (macro) = 0.98421; the confusion counts are TN = 7023, FP = 25, FN = 159, TP = 4841. Training/validation losses converge smoothly, and SHAP beeswarm plots show that all four retained features contribute consistently across many instances, explaining the strong threshold-free metrics. These results demonstrate that accurate, interpretable, and deployment-ready XSS detection is achievable with a minimal feature budget.**

Keywords— **Cross Site Scripting, XSS, CatBoost, SHAP, feature selection, lightweight models, interpretability.**

I. Introduction

In the ever-changing landscape of cyber threats, Cross-Site Scripting (XSS) continues to challenge web security at scale [1]. By injecting malicious scripts into vulnerable applications, attackers jeopardize data integrity and user safety and can pivot rapidly across systems [2–3]. While Web Application Firewalls (WAFs) mitigate a portion of attacks, gaps persist when unsafe coding patterns reach production or when payloads are deliberately obfuscated to evade rule-based defenses [4]. Incident reports on major platforms repeatedly highlight the operational impact of XSS, and industry surveys continue to rank it among the most reported and severe web vulnerabilities [5–8]. These trends underscore the need for detection strategies that are both precise and operationally efficient.

Machine learning has emerged as a strong candidate for XSS detection, spanning deep-learning pipelines with sequence embeddings and LSTMs, crawler-assisted testing, and classical models such as SVMs, k-NN, Random Forests, and Decision Trees [9–16]. Yet many systems remain feature-rich and computationally heavy, which complicates real-time deployment in resource-constrained environments such as in-line gateways, edge nodes, and large-scale multi-tenant services. This paper addresses that bottleneck by explicitly investigating the minimal signal required for reliable XSS classification.

Our study builds on our previous work [17], which established the dataset, preprocessing pipeline, stratified sampling with SMOTENC balancing, and correlation-based de-redundancy. Here we start after the correlated features have been removed and pursue an iterative, model-driven reduction guided by CatBoost feature importance and SHAP. At each step we retrain and re-evaluate on fixed splits, enabling paired comparisons across the entire reduction path. The central question is whether a drastically reduced feature set can retain high discrimination while remaining interpretable.

The contributions of this paper are threefold. First, we present a transparent reduction path that terminates in a four-feature CatBoost classifier while preserving stability in training and generalization on held-out data. Second, we formalize an evaluation protocol that emphasizes threshold-free assessment and consistency checks under fixed, stratified splits, aligning with deployment realities. Third, we provide interpretable explanations via SHAP that justify the selected signals—Contains Less-Than, ScriptTag, Contains Question-Mark, and Contains Comma—highlighting how compact, semantically meaningful indicators can support simpler, faster, and more portable defenses without sacrificing clarity of decision rationale.

II. METHOLOGY

A. *Continuity with Prior Work*

This study explicitly builds on our previous research [17]. We reuse the same dataset, preprocessing, stratified sampling and SMOTENC balancing, and the hierarchical Spearman-based feature de-redundancy already established and validated in [17]. The present work starts after correlated features have been removed and focuses on deleting features based on model feature importance and SHAP values, followed by grid-searched CatBoost training and a transparent, weighted model-selection stage.

B. *Model Development Process Flow*

The development pipeline follows a systematic, iterative flow aimed at balancing model complexity with discriminative performance. A baseline CatBoost model is first trained using the full feature set to establish an upper bound for accuracy and a reference point for subsequent reductions. Guided by the correlation analysis already

Manuscript received September 21, 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).

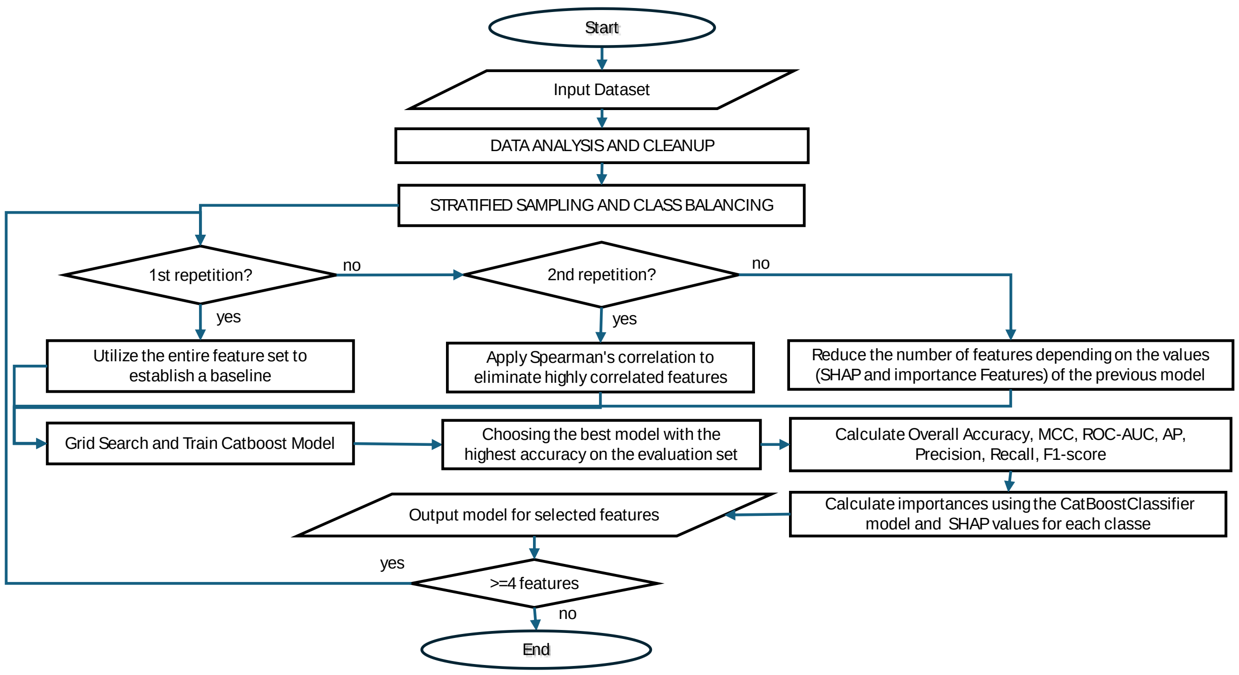
Start
Input Dataset
DATA ANALYSIS AND CLEANUP
STRATIFIED SAMPLING AND CLASS BALANCING
1st repetition?
no
2nd repetition?
no
yes
yes
Utilize the entire feature set to establish a baseline
Apply Spearman's correlation to eliminate highly correlated features
Reduce the number of features depending on the values (SHAP and importance Features) of the previous model
Grid Search and Train Catboost Model
Choosing the best model with the highest accuracy on the evaluation set
Calculate Overall Accuracy, MCC, ROC-AUC, AP, Precision, Recall, F1-score
Output model for selected features
Calculate importances using the CatBoostClassifier model and SHAP values for each classe
yes
>=4 features
no
End

Figure 1: Feature Importances for XSS Detection Model.

performed in [17], features exhibiting strong redundancy are pruned, which yields a streamlined set upon which the present paper operates. The core loop then proceeds in iterations: at each iteration, a CatBoost model is trained and tuned; its evaluation-set accuracy is recorded; SHAP values and model-reported feature importance are computed; and features deemed least influential are eliminated. This train–assess–prune cycle continues while the model's performance is monitored as in [17], and while the evolving importance profiles (Fig. 1) inform the next reduction step. The loop terminates when the retained features are reduced to a compact subset (down to four features), producing the sequence of 14 candidate models referenced throughout this work. Each candidate is subsequently considered in a final selection stage whose objective is to preserve robust XSS detection while minimizing feature count to enable efficient, real-world operation.

C. *Deleting Features Based on Feature Importance and SHAP Values*

Selecting the right features is crucial for distinguishing between benign and malicious inputs while keeping the model simple and efficient. We therefore combine model-reported feature importance with SHAP values to guide an iterative pruning process that preserves interpretability.

We proceed after the correlated features have been removed. At each iteration, a CatBoost model is trained on the current feature set; we then (i) compute feature importance and SHAP values, (ii) identify features with low mean absolute SHAP and low importance rankings (i.e., minimal impact on predictions), (iii) remove those least-influential features, and (iv) retrain the model on the reduced set. After every removal step, we re-evaluate key metrics—accuracy, precision, recall, and F1—to verify that pruning does not degrade discriminative performance. This train–assess–prune cycle continues until a compact subset is obtained (down to four features in our case).

To quantify importance in tree ensembles, we use the split-gain formulation:

$$I(f) = \sum_{t=1}^{T} \Delta G_t(f)$$

where $\Delta G_t(f)$ is the improvement in the objective function (e.g., Gini impurity or entropy) due to splits on feature f in tree t, and T is the total number of trees.

$$\phi_i(f,x) = \sum_{S\subseteq F\setminus\{i\}} \frac{|S|!\,(|F|-|S|-1)!}{|F|!}\,[f(S\cup\{i\}) - f(S)]$$

In this expression, ϕ_i is the SHAP value for feature i, F is the set of all features, x is the input instance, S is a subset of F excluding i, and f(S) is the model output using features in S. Aggregating SHAP values across instances yields mean absolute SHAP profiles and class-specific plots that clarify how features influence benign and attack predictions, which in turn guide the next pruning step.

In short, the dual use of model importance and SHAP provides complementary, transparent signals: global contributions from split gains and local contributions from SHAP. Evaluating after each deletion ensures that the final, minimal feature set maintains the desired balance between interpretability and accuracy.

III. Results

To evaluate the four-feature CatBoost classifier for Cross-Site Scripting (XSS), we examine the test-set confusion matrix (Fig. 2), which aligns predictions with ground truth into true positives (TP), true negatives (TN), false positives (FP; benign requests incorrectly flagged—Type I error), and false negatives (FN; attacks incorrectly accepted—Type II error); on the fixed split the counts are TN = 7023, FP = 25, FN = 159, TP = 4841 (7048 benign, 5000 malicious; total = 12 048), yielding 98.47% overall accuracy, TPR/Recall (attacks) = 96.82% (4841/5000), TNR/Specificity (benign) = 99.65% (7023/7048), FPR = 0.35% (25/7048), FNR = 3.18% (159/5000), Precision (attacks) = 99.49% (4841/(4841+25)), and NPV = 97.79% (7023/(7023+159)); together these figures indicate a very low alarm rate on benign traffic while capturing the vast majority of attacks, and they are consistent with the paper's aggregate metrics (e.g., MCC = 0.9687, macro-Recall = Balanced Accuracy = 98.23%), underscoring that high discrimination is retained despite the aggressive reduction to just four input features.

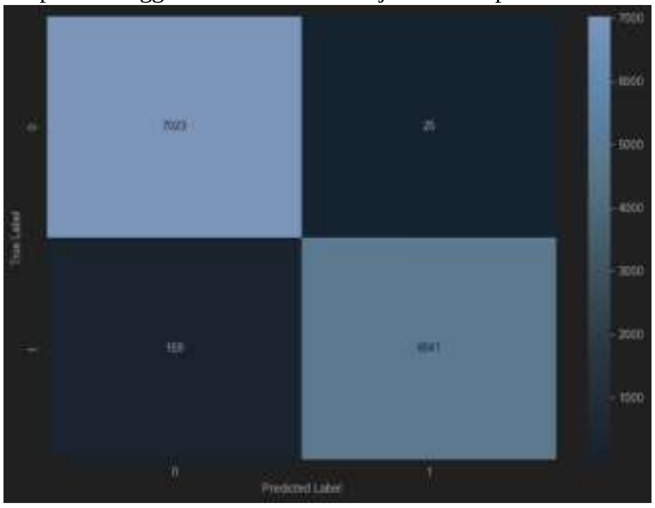

Figure 2: Confusion Matrix of the four feature CatBoostClassifier on the test set.

Figure 3 traces the training and validation loss across boosting iterations. Both curves drop sharply at the start, showing that the model captures the signal quickly; a clear "knee" appears after the early rounds (roughly the first few dozen iterations), after which improvements taper and both curves flatten toward low values. The generalization gap between training and validation remains small and stable throughout, with no late-stage rise in validation loss, indicating stable learning without harmful overfitting under the chosen CatBoost settings. The smooth, non-oscillatory profiles also suggest well-tuned optimization and that an early-stopping checkpoint near the validation minimum would retain performance while limiting computation. Overall, the dynamics confirm that reducing to four features does not impede convergence or stability and aligns with the strong aggregate metrics reported later.

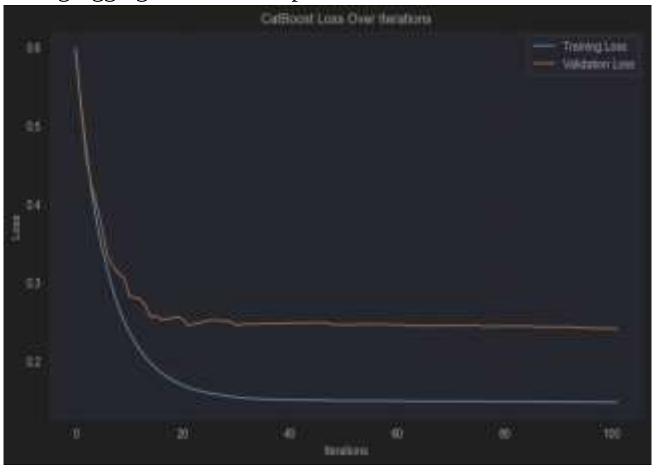

Figure 3: CatBoost Training and Validation Loss Over Iterations.

To evaluate threshold behavior, we examine the Receiver Operating Characteristic (ROC), which traces the True Positive Rate (TPR/sensitivity) against the False Positive Rate (FPR = 1 − specificity) as the decision threshold varies. The macro-averaged ROC in Fig. 4 hugs the upper-left corner and sits far above the no-skill diagonal, yielding AUC = 0.9932. This implies that, over random pairs of attack vs. benign requests, the classifier ranks the attack higher ~99.3% of the time. The steep early rise at very low FPR indicates the model can operate in conservative regimes—with minimal false alarms—while retaining high sensitivity, consistent with the low FPR observed in the confusion-matrix analysis. Overall, the ROC confirms strong ranking quality independent of any single operating threshold.

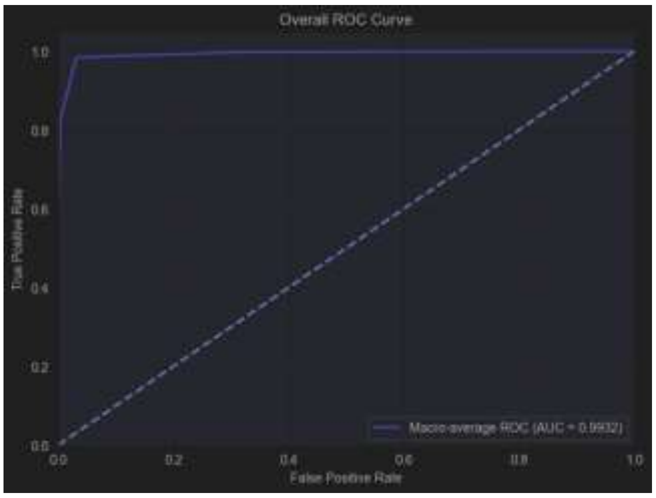

Figure 4: Overall ROC Curve for the four feature CatBoostClassifier

Precision–Recall (PR) curves characterize performance under class imbalance by showing how precision varies with recall as the decision threshold moves; unlike ROC, PR focuses on the quality of positive predictions, which is critical in XSS detection where both missed attacks and spurious alerts are costly. In Fig. 5, both class-wise curves cling to the upper-right region and remain essentially flat at high precision across a broad recall range, with a sharp precision drop only at extreme recall values. The area under each curve is high—AP = 0.9910 for Class 0 (benign) and AP = 0.9889 for Class 1 (attack)—yielding a macro-AP ≈ 0.9900. This shape indicates that the model supports conservative operating points (very low false alarms) as well as high-recall regimes with minimal precision penalty, consistent with the confusion-matrix summary (Precision_attack ≈ 99.49%, Recall_attack ≈ 96.82%). Overall, the PR analysis confirms strong ranking quality for both classes and reinforces the detector's suitability for deployment where precision at high recall is required.

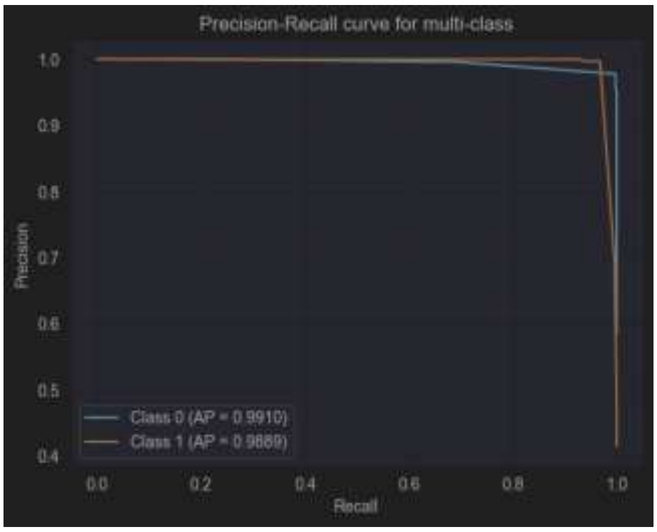

Figure 5: Precision–Recall Curves for the four feature CatBoostClassifier.

To understand which inputs drive decisions in such a compact model, we examine the global importances and local explanation signals along the reduction path. Figure 6 charts CatBoost's split-based feature importances per candidate model (2→14). As features are removed, importance mass is reallocated among the survivors; weak indicators quickly sink toward the baseline and are dropped, while a small core steadily rises into the top tier. Notably, the four signals that remain in the final detector—Contains Less-Than, ScriptTag, Contains Question Mark, and Contains Comma—progressively dominate the ranking as the set shrinks, which is exactly what we expect if they carry most of the discriminative signal. Because importances are relative within each model, the most reliable way to read this plot is by rank order and its stability across models rather than by comparing raw magnitudes.

Figure 7 complements this with mean absolute SHAP values for Class 0 (benign) across iterations. Whereas Fig. 6 summarizes global split usage, SHAP aggregates instance-wise contributions. The same core features persist with high

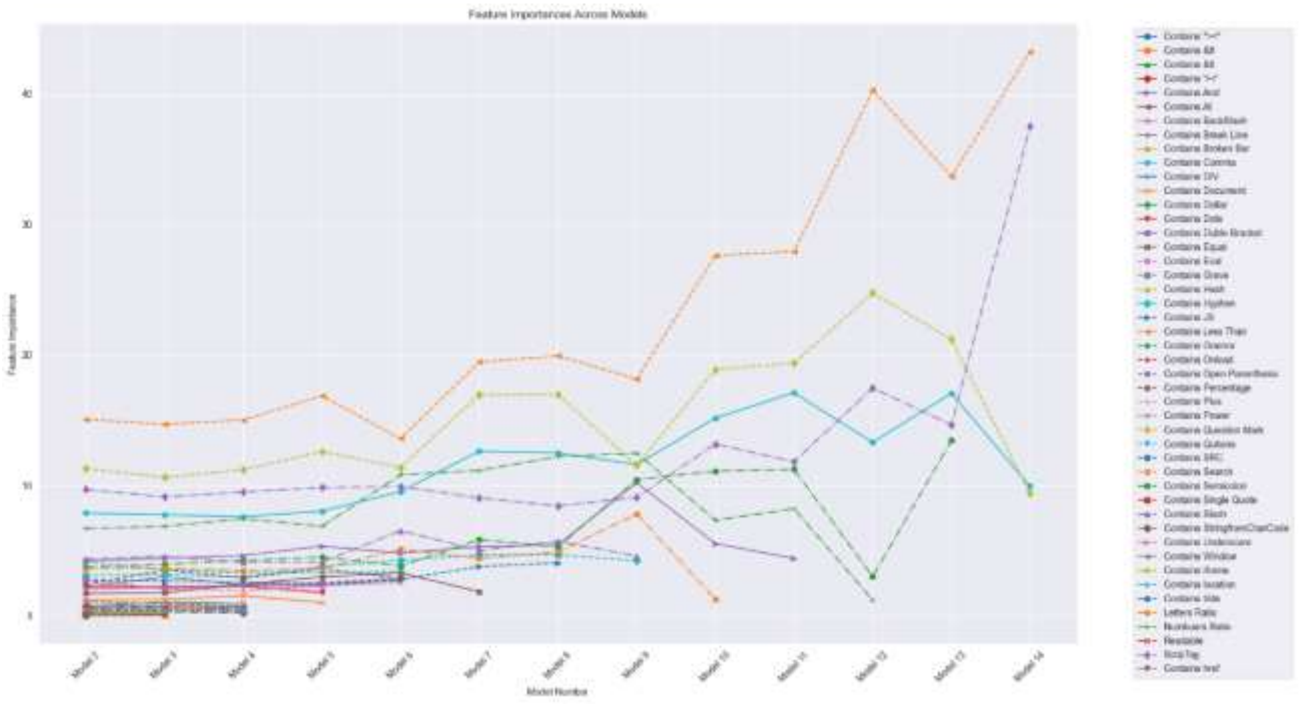

Figure 6: Feature importances across models (2–14).

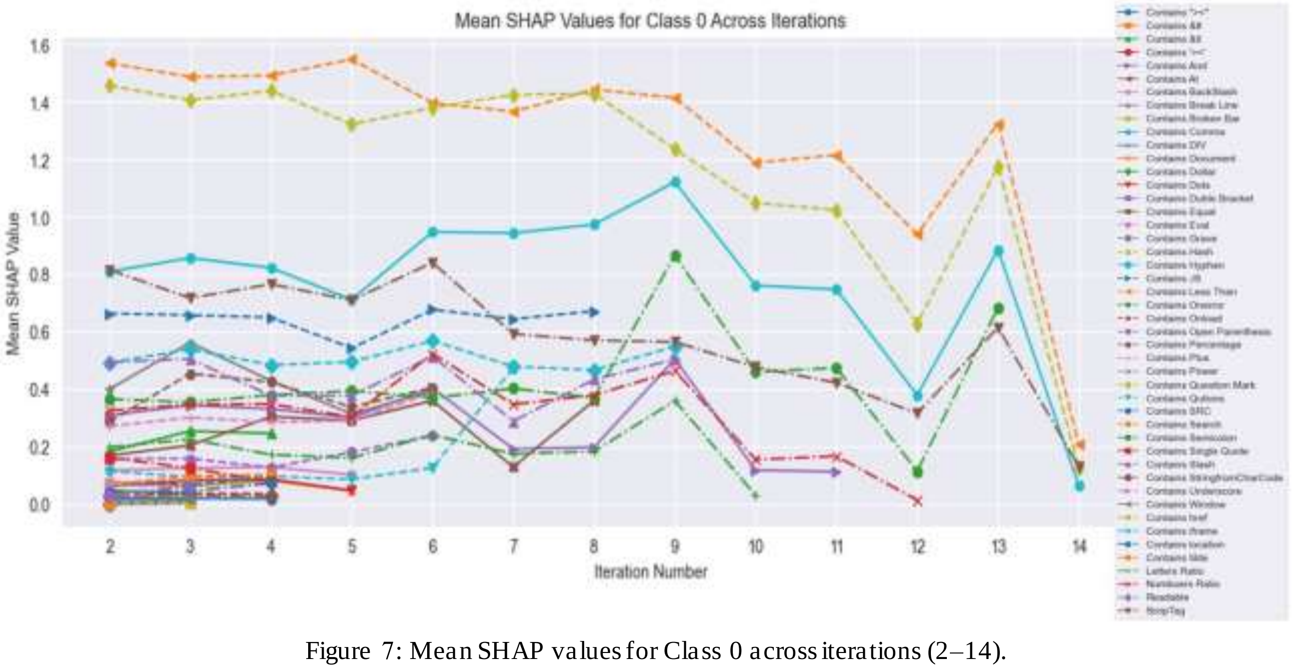

Figure 7: Mean SHAP values for Class 0 across iterations (2–14).

mean SHAP across the path, indicating consistent local influence over many examples rather than a few outliers. Features that are eventually pruned show diminishing SHAP profiles prior to removal, and once eliminated their traces drop out in subsequent iterations—evidence that the pruning loop is guided by genuine contribution rather than noise. Together, the two views (global importance and local SHAP) converge on the same compact feature set, explaining why the model retains high discrimination even with a minimal budget.

Figure 8 provides class-wise TreeSHAP explanations for the final four-feature model, clarifying how each signal shapes decisions. SHAP values are in log-odds: points to the right of zero support the plotted class, points to the left oppose it; color reflects the raw feature value (warm = high, cool = low). In the benign panel (top), high Contains Less-Than and ScriptTag values yield consistently negative contributions clustered left of zero, matching their role as attack markers. Contains Question Mark and Contains Comma show smaller, bidirectional effects near zero, reflecting punctuation that appears in both benign queries and obfuscated payloads.

The attack panel (bottom) mirrors this pattern: Less-Than and ScriptTag shift decisively right, contributing strongly across many instances, while punctuation features add positive evidence mainly when co-occurring with the structural markers—suggesting learned interactions rather than isolated cues. Two properties aid deployment: high sign consistency for the structural features, which makes explanations stable and auditable, and a noticeable mass near zero for punctuation, which helps maintain low FPR while preserving recall. The ordering by mean |SHAP| also aligns with the global importance trend (Fig. 8), indicating that both global and local views single out the same core signals. In practice, operators can tune thresholds for conservative modes while monitoring drift by tracking per-feature mean |SHAP| over time. Together, strong, semantically grounded markers plus low-variance modulators explain how the compact model stays both discriminative and interpretable. Edge cases where punctuation dominates without structural cues are rare and tend to cluster near the decision boundary, making them natural candidates for manual review or margin-based alerting.

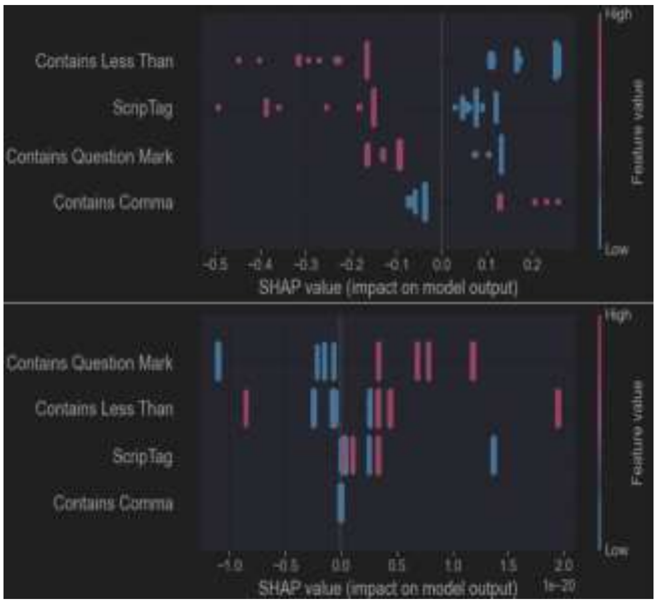

Figure 8: SHAP Value Plots for the four feature model — top: Class 0 (benign), bottom: Class 1 (attack).

For reference, Table 1 enumerates the features retained at each step of the reduction sequence; the final Model 14 uses indices 41, 2, 25, and 39, which correspond respectively to

Table 1: Feature indices retained by each XSS detection model along the reduction path.

Model	Feature Index
1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65
2	41, 25, 2, 39, 29, 8, 40, 62, 27, 64, 6, 7, 10, 63, 1, 31, 24, 36, 9, 54, 11, 18, 44, 46, 49, 35, 45, 12, 3, 13, 23, 16, 4, 14, 34, 43, 30, 60, 47, 59, 21, 55, 52, 15, 20, 5
3	41, 25, 39, 2, 29, 62, 40, 8, 7, 27, 64, 6, 10, 1, 63, 31, 36, 24, 9, 46, 54, 49, 18, 11, 3, 44, 45, 13, 35, 23, 16, 12, 4, 14, 43, 34, 30, 60, 15, 47
4	41, 25, 39, 2, 29, 40, 62, 7, 27, 8, 64, 6, 31, 10, 1, 63, 9, 36, 46, 54, 18, 49, 11, 24, 45, 44, 13, 23, 12, 4
5	41, 25, 39, 2, 29, 40, 27, 8, 7, 62, 64, 6, 31, 10, 36, 63, 9, 54, 49, 11
6	41, 25, 39, 2, 29, 40, 64, 8, 7, 6, 62, 10, 27, 31, 63, 36, 54
7	25, 41, 39, 29, 2, 40, 27, 64, 8, 6, 63, 31
8	41, 25, 39, 29, 2, 40, 8, 64, 27, 6, 63
9	41, 25, 39, 27, 2, 40, 6, 8, 64, 63
10	41, 25, 39, 2, 27, 64, 6, 63.
11	41, 25, 39, 27,2, 64, 6
12	41, 25, 39, 2, 27, 64
13	41, 25, 39, 27, 2
14	41, 2, 25, 39

Index and feature

1 Contains &lt, 2 ScripTag, 3 Readable, 4 Contains "><, 5 Contains ><, 6 Contains And, 7 Contains Percentage, 8 Contains Slas h, 9 Contains BackSlash, 10 Contains Plus, 11 Contains Document, 12 Contains Window, 13 Contains Onload, 14 Contains Onerror, 15 Contains DIV, 16 Contains iframe, 17 Contains img, 18 Contains SRC, 19 Containss Var, 20 Contains Eval, 21 Contains href, 22 Contains Cookie, 23 Contains StringfromCharCode, 24 Contains Single Quote, 25 Contains Question Mark, 26 Contains Exclamation Mark, 27 Contains Semicolon, 28 Contains HTTP, 29 Contains JS, 30 Contains Hash, 31 Contains Equal, 32 Contains Open Bracket, 33 Contains Close Bracket, 34 Contains Duble Bracket, 35 Contains Dollar, 36 Contains Open Parenthesis, 37 Contains Close Parenthesis, 38 Contains Asterisk, 39 Contains Comma, 40 Contains Hyphen, 41 Contains Less Than, 42 Contains Greater Than, 43 Contains At, 44 Contains Underscore, 45 Contains location, 46 Contains Search, 47 Contains &#, 48 Contains Colon, 49 Contains Dots, 50 Contains Open Brace, 51 Contains Close Brace, 52 Contains tilde, 53 Contains Spase, 54 Contains Qutions, 55 Contains Grave, 56 Contains Duble Equals, 57 Contains Duble Slash, 58 Contains Vertical Bar, 59 Contains Power, 60 Contains Broken Bar, 61 Contains Alert, 62 Contains Break Line, 63 Letters Ratio, 64 Numbuers Ratio, 65 Symbols Ratio

Contains Less-Than, ScriptTag, Contains Question-Mark, and Contains Comma. Table 2 reports the full set of test-set metrics for all models. The four-feature classifier attains Accuracy = 0.984728, MCC = 0.9687, ROC-AUC = 0.9932, AP = 0.9910, Precision = 0.986362, Recall = 0.982326, and F1 = 0.98421, confirming that near-ceiling discrimination is preserved despite aggressive pruning.

Table 2 contais comparative test-set performance across models. Accuracy, Precision, Recall, and F1 are macro-averaged. The AP column reports macro-AP; for the four-feature model the per-class APs are 0.9910 and 0.9889, yielding a macro-AP of approximately 0.9900.

Table 2: Comparative test-set performance across models. Accuracy, Precision, Recall, and F1 are macro-averaged.

Model	Number Of Features	Accuracy	MCC	ROC AUC	AP	Precision	Recall	F1-score
1	65	0.999751	0.9995	1	1	0.999758	0.999729	0.99974
2	46	0.998755	0.9974	1	1	0.998733	0.998703	0.99872
3	40	0.998589	0.9971	1	1	0.998533	0.998561	0.99855
4	30	0.998672	0.9973	1	1	0.998633	0.998633	0.99863
5	20	0.998506	0.9969	1	1	0.998520	0.998403	0.99846
6	17	0.999087	0.9981	1	1	0.999104	0.999016	0.99906
7	12	0.997510	0.9949	1	1	0.997670	0.997204	0.99744
8	11	0.997759	0.9954	0.9999	0.9999	0.997942	0.997445	0.99769
9	10	0.997427	0.9947	0.9999	0.9999	0.997166	0.997540	0.99735
10	8	0.996680	0.9932	0.9984	0.9988	0.996814	0.996348	0.99658
11	7	0.996431	0.9927	0.9984	0.9987	0.996572	0.996078	0.99632
12	6	0.994273	0.9882	0.9975	0.9975	0.994467	0.993739	0.99410
13	5	0.991783	0.9832	0.9952	0.9949	0.993006	0.990158	0.99151
14	4	0.984728	0.9687	0.9932	0.9910	0.986362	0.982326	0.98421

The retained signals align with practical web-programming semantics. ScriptTag marks explicit \<script occurrences, whereas Less-Than flags entry into any HTML-tag context; not only scripts (e.g., <img onerror=…>, <svg onload=…>, <a href="javascript:…">). Question-Mark commonly appears in query-string payloads and obfuscation patterns, and Comma frequently arises in JavaScript argument lists and String.fromCharCode(…) constructs. SHAP analyses (Fig. 8) show that punctuation features contribute only minor, bidirectional evidence near zero and become discriminative chiefly when co-occurring with the structural markers, which explains the low FPR alongside solid recall. Together, these patterns indicate the model is not restricted to <script>-based injections but captures broader tag- and event-handler-based vectors.

IV. CONCLUSION

This paper demonstrated that ultra lightweight XSS detection is feasible without a material loss in discriminative power. Starting from the pipeline established in [17] and operating after correlation-based pruning, we iteratively removed features using CatBoost importance and SHAP guidance until reaching a four feature model. On the fixed test split, the classifier achieved Accuracy = 0.984728, MCC = 0.9687, ROC AUC = 0.9932, AP = 0.9910, Precision = 0.986362, Recall = 0.982326, and F1 = 0.98421, with confusion counts TN = 7023, FP = 25, FN = 159, TP = 4841. Training and validation losses converged smoothly, and SHAP analyses confirmed that each of the four retained features contributes meaningfully and consistently—evidence that the predictive signal concentrates in a compact, interpretable subset.

The practical implications are direct: a smaller feature footprint simplifies data collection and preprocessing, reduces memory and latency, and facilitates deployment in resource constrained or real time settings without sacrificing robustness. While the present study focuses on the established dataset and fixed splits from [17], future work can broaden external validity by testing on heterogeneous web applications, evolving payloads, and adversarially crafted inputs, and by exploring probability calibration and domain adaptation. Overall, the results validate a clear message: with careful, model guided reduction, four features are enough to deliver accurate, transparent, and deployable XSS detection.

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] Steffens, M., Rossow, C., Johns, M., & Stock, B. (2019a). Don't Trust The Locals: Investigating the Prevalence of Persistent Client-Side Cross-Site Scripting in the Wild.

[4] Wang, Q., Chen, J., Jiang, Z., Guo, R., Liu, X., Zhang, C., & Duan, H. (2024). Break the Wall from Bottom: Automated Discovery of Protocol-Level Evasion Vulnerabilities in Web Application Firewalls. 2024 IEEE Symposium on Security and Privacy (SP), 185–202. https://doi.org/10.1109/SP54263.2024.00129

[5] 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. https://doi.org/10.1016/j.cosrev.2024.100634

[6] Caturano, F., Perrone, G., & Romano, S. P. (2021). Discovering reflected cross-site scripting vulnerabilities using a multiobjective reinforcement learning environment. Computers & Security, 103, 102204. https://doi.org/10.1016/j.cose.2021.102204

[7] 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. https://doi.org/10.1007/s10462-023-10433-3

[8] Buyukkayhan, A. S., Gemicioglu, C., Lauinger, T., Oprea, A., Robertson, W., & Kirda, E. (2020). What's in an Exploit? An Empirical Analysis of Reflected Server {XSS} Exploitation Techniques. 107–120.

[9] Fang, Y., Li, Y., Liu, L., & Huang, C. (2018). DeepXSS: Cross site scripting detection based on deep learning. 47–51.

[10] Guan, H., Li, D., Li, H., & Zhao, M. (2022). A Crawler-Based Vulnerability Detection Method for Cross-Site Scripting Attacks. 651–655.

[11] Kumar, J. H., & Ponsam, J. G. (2023). Cross site scripting (XSS) Vulnerability detection using machine learning and statistical analysis. 1–9.

[12] Mereani, F. A., & Howe, J. M. (2018). Detecting cross-site scripting attacks using machine learning. 200–210.

[13] Mereani, F., & Howe, J. M. (2019). Exact and approximate rule extraction from neural networks with Boolean features. 1, 424–433.

[14] Kascheev, S., & Olenchikova, T. (2020). The detecting cross-site scripting (XSS) using machine learning methods. 265–270.

[15] Chen, H.-C., Nshimiyimana, A., Damarjati, C., & Chang, P.-H. (2021). Detection and prevention of cross-site scripting attack with combined approaches. 1–4.

[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, 79(11), 763–802.

[17] Hajjouz, A., & Avksentieva, E. (2025). Highly Accurate XSS Detection using CatBoost. International Journal of Open Information Technologies, 13(6), 125–131.