A Sample-Based, Multi-Stage Machine Learning Pipeline for Scalable IoT Threat Detection

Marcelo V. C. Aragão, Tiago de M. Pereira, Felipe A. P. de Figueiredo, and Samuel B. Mafra

***Abstract*—The rapid growth of IoT devices demands scalable and efficient threat detection solutions. This paper introduces a sample-based, multi-stage machine learning (ML) pipeline for IoT threat detection using the CICIoT2023 dataset, integrating feature selection, data balancing, and hyperparameter optimization to improve detection accuracy while reducing the computational overhead associated with training. We evaluate LightGBM, XGBoost, and XGBoostRF across binary, multiclass, and fine-grained tasks, showing that XGBoost with 10% sampling achieves the best trade-off between accuracy and efficiency. Compared to prior methods, our approach eliminates GPU dependence, maintains low latency, and preserves state-of-the-art performance while enabling scalable training for high generalization capacity. Additionally, we provide model selection guidelines based on dataset complexity and computational constraints. The results show that training with a sample-based approach enables effective threat detection on large datasets, producing models that generalize well to diverse IoT attack scenarios, thus ensuring practical applicability in real-world deployments.**

***Index Terms*—IoT Security, Machine Learning, Feature Selection, Data Balancing, Hyperparameter Optimization**

I. Introduction

The rapid growth of IoT devices has created challenges for scalable threat detection. High data volume and dimensionality make real-time analysis computationally expensive, while class imbalance skews model performance, making rare but critical threats difficult to detect. Deploying ML models on resource-limited edge devices requires lightweight solutions, and the evolving nature of IoT attacks necessitates adaptive models that generalize across environments without frequent retraining. These challenges demand scalable and efficient ML-based optimization pipelines based on massive datasets.

ML-based Network Traffic Analysis (NTA) has emerged as a promising solution for IoT security. Using the IoT-23 dataset, decision tree models such as Random Forest (RF) and Extra Trees achieved 99.86% accuracy, outperforming Deep Neural Networks (DNN) and Support Vector Machines (SVM). A pipeline integrating generative and traditional data balancing techniques with Hyperparameter Optimization (HPO) improved AutoML performance by up to 6% [1]. Additionally, a hybrid CNN-LSTM model attained 99.995% accuracy for binary and 99.96% for multiclass classification, demonstrating DL's effectiveness for real-time intrusion detection [2].

Efforts to enhance IoT Intrusion Detection Systems (IDS) have also addressed data balancing and dimensionality challenges. A two-step clustering method with the Gaining-Sharing Knowledge (GSK) optimization algorithm achieved 98.83% accuracy for binary and 97.47% for multiclass classification on CICIoT2023 [3].

The authors are with the National Institute of Telecommunications – Inatel, Av. João de Camargo, 510, Santa Rita do Sapucaí, MG, Brazil (corresponding author's e-mail: marcelovca90@inatel.br).

Manuscript received Month Day, Year; revised Month Day, Year.

The CICIoT2023 dataset [4], with nearly 47 million samples spanning 33 attack types (e.g., DDoS, DoS, Recon, Web-based, Brute Force, Spoofing, and Mirai) across 105 IoT devices, is a key benchmark for ML-based threat detection. It evaluated five ML and DL models – Logistic Regression, Perceptron, Adaboost, RF, and DNN – for binary (benign vs. malicious), multiclass (8 categories), and fine-grained (34 classes) classification. RF achieved the highest F_1 scores: 0.965 for binary, 0.719 for 8-class, and 0.714 for 34-class. While valuable, this study lacks key optimizations such as feature selection (FS), data balancing, and HPO, which are essential for improving detection on imbalanced datasets.

Compared to these methods, our pipeline is designed to achieve high detection performance with significantly lower computational overhead. By integrating FS, data balancing, and multi-stage HPO, our approach enables training on large datasets while maintaining efficiency and enhancing recall and precision without requiring GPU acceleration, thus making it well-suited for real-time, resource-constrained IoT environments. The effectiveness of this optimized pipeline is evaluated in the following sections, which cover the Assessment Methodology (Section II), Experiments and Discussion (Section III), and Conclusion (Section IV).

II. Assessment Methodology

We propose a multi-stage ML pipeline that addresses key challenges such as high data volume, class imbalance, and model optimization to enhance IoT threat detection while maintaining computational efficiency. It includes data sampling, preprocessing, feature selection, data balancing, and HPO, ensuring scalability and robust attack detection. The process is structured into steps depicted in Figure 1 and explained next. The source code implementing the proposed optimization pipeline is publicly available [5].

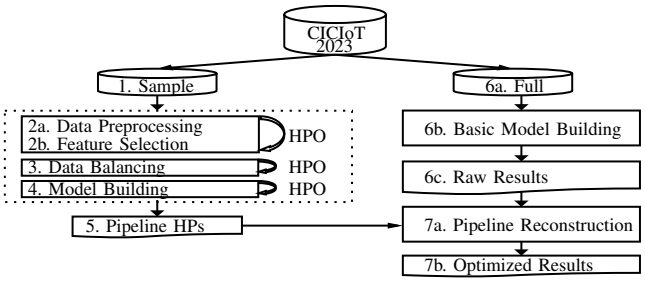

Fig. 1. Proposed Pipeline with Multi-Stage HPO.

1) **Extract Representative Sample**: To reduce computational overhead while preserving data representativeness, a stratified sampling approach is applied. Pandas [6] is used for data handling, while scikit-learn [7] ensures proper stratification. By using a well-balanced subset instead of the full dataset, this step optimizes resource usage while maintaining model performance.

2) **Data Preprocessing and Feature Selection**: High dimensionality is reduced through preprocessing and feature selection, improving efficiency. Scaling methods (MaxAbs, MinMax, Standard, and Robust) are applied via scikit-learn [7]. Feature selection techniques, including Boruta [8], RFE [9], and SHAP [10], retain key features while discarding redundancies. Optuna [11] optimizes preprocessing and FS, while SHAP-Hypetune [12] refines selection to ensure models learn from relevant data.
3) **Data Balancing**: Class imbalance must be addressed to detect rare attacks. HPO is applied to the sample dataset using Random Over- and Undersamplers from imbalanced-learn, optimizing class distribution thresholds via the Tree-structured Parzen Estimator (TPE) Sampler in Optuna. This step ensures adequate representation of minority attack classes, reducing prediction bias.
4) **Model Configuration**: HPO is performed on the stratified sample set to maximize detection accuracy while maintaining efficient deployment. The models considered – LightGBM, XGBoost, and XGBoostRF– are selected for their robustness in handling large-scale datasets and complex patterns typical in IoT security. The TPE Sampler efficiently explores a wide hyperparameter space, balancing model accuracy and computational cost.
5) **Persist Pipeline Hyperparameters**: The best-performing configurations for preprocessing, feature selection, data balancing, and model training are stored to ensure consistency when applying the pipeline to the full dataset, reducing the need for repetitive optimizations.
6) **Full Dataset Raw Performance**: A baseline performance assessment is conducted by training ML models on the full dataset without preprocessing, feature selection, sampling, or HPO. Performance metrics – including accuracy, precision, recall, and macro-averaged F_1 score – are calculated for consistency with [4], providing a reference for later comparisons.
7) **Full Dataset Optimized Performance**: Using optimized preprocessing, feature selection, data balancing, and hyperparameters from previous steps, models are trained on the full dataset. The same evaluation metrics assess classification improvements.
8) **Performance Comparison**: The final optimized results are compared against both the raw performance (Step 6) and results from [4], quantifying the impact of feature selection, data balancing, and HPO in improving detection efficiency and scalability.

III. Experiments and Discussion

As introduced in Section I, the CICIoT2023 dataset includes 33 attack types categorized into DDoS, DoS, Reconnaissance, Brute Force, Spoofing, Web-based, and Mirai-based threats. To assess detection performance at different levels, we define three classification scenarios: (1) **2-class**, which differentiates benign and malicious traffic; (2) **8-class**, where attacks are grouped into broader categories to evaluate generalization; and (3) **34-class**, which distinguishes individual attack types. This structure provides a comprehensive evaluation of model performance across varying levels of attack specificity. The experiments were conducted on a computer with an AMD® Ryzen™ Threadripper 7970X processor, 256 GB of DDR5-4800 RAM, and Ubuntu operating system (v22.04.5).

The pipeline was executed using LightGBM, XGBoost, and XGBoostRF, with computations on the CPU. The minimum class size was set to 5 to ensure representation in the stratified 60/20/20 training, validation, and test splits. Experiments were conducted with 2, 8, and 34 classes, using 1%, 5%, and 10% of the dataset for HPO, feature selection, and data balancing. The full dataset was used only for final inference, as running HPO on the entire dataset would require excessive computational time, making it infeasible. Our results demonstrate that models optimized with just 10% of the dataset generalize effectively to the full dataset while significantly reducing execution time.

HPO was performed using Grid Search or TPE Samplers, constrained by a one-hour timeout or 1000 trials for TPE. The macro F_1 score over the validation subset was maximized, and all models utilized available CPU cores with a random seed of 42 for reproducibility.

Table I summarizes the experimental results, detailing the performance metrics of various threat detection models across different sampling rates and scenarios. The first two columns, "Classes" and "Metric," specify the classification scenarios (34, 8, or 2 classes) and evaluation metrics, including accuracy, precision, recall, F_1 score, and latency. Latency represents the time required to classify a single sample. The [4] column reports results from prior work, where RF achieved the highest F_1 score across all scenarios.

The "Raw" column shows baseline results from Step 6, where models were trained on the full dataset without preprocessing, FS, sampling, or HPO. The 1%, 5%, and 10% columns present results from Step 7, where preprocessing, FS, sampling, and HPO were applied before training. The best-performing values are highlighted, and the Δ column quantifies improvements by comparing the best result (from "Raw", 1%, 5%, or 10%) with [4]. No Δ is provided for latency, as it was not reported in [4].

The following subsections analyze model performance in terms of the adopted metrics, with particular attention to the raw F_1 score and its improvements in the Δ column. Additionally, each scenario's Pareto front will be examined to evaluate the trade-offs between execution time and performance, highlighting the configurations offering optimal balance for each case.

A. Performance Metrics Analysis

This subsection evaluates the aforementioned performance metrics, with full experimental results shown in Table I, with Table II summarizing best-performing models, their optimal sampling rates, and underlying factors.

Latency analysis across sampling rates reveals key computational patterns. LightGBM achieves the lowest latency due to efficient histogram-based learning, while XGBoost and XGBoostRF show higher latency at lower sampling rates due to deeper decision paths. While sampling accelerates training, latency is dictated by model structure. Raw (no sampling) configurations can achieve lower latency by optimizing decision boundaries. Moderate sampling (5–10%) balances

TABLE I
MODEL PERFORMANCE ACROSS SAMPLING RATES AND CLASSIFICATION TASKS.

Classes	Metric	[4]	LightGBM				XGBoost				XGBoostRF				Δ
			Raw	1%	5%	10%	Raw	1%	5%	10%	Raw	1%	5%	10%	
34	Accuracy	0.992	0.486	0.326	0.377	0.993	0.994	0.995	0.995	**0.996**	0.991	0.995	0.995	0.995	+0.004
	Recall	0.832	0.168	0.272	0.178	0.757	0.755	0.875	**0.881**	0.841	0.705	0.819	0.819	0.836	+0.049
	Precision	0.704	0.186	0.275	0.175	0.818	0.875	0.887	0.841	0.915	0.858	0.912	0.895	**0.920**	+0.216
	F_1 score	0.714	0.150	0.177	0.132	0.772	0.780	**0.880**	0.857	0.869	0.718	0.845	0.841	0.864	+0.166
	Latency [μs]	–	**0.799**	41.000	13.000	2.000	1.135	183.000	41.000	5.000	1.114	86.000	18.000	20.000	–
8	Accuracy	0.994	0.990	0.982	0.970	0.991	0.995	0.996	**0.997**	0.997	0.993	0.996	0.996	0.996	+0.003
	Recall	0.910	0.741	0.880	0.887	0.897	0.759	**0.909**	0.862	0.866	0.689	0.845	0.851	0.845	-0.001
	Precision	0.705	0.764	0.729	0.654	0.750	0.910	0.805	0.958	0.955	0.936	**0.962**	0.957	0.956	+0.257
	F_1 score	0.719	0.751	0.746	0.696	0.768	0.794	0.834	0.899	**0.901**	0.714	0.889	0.891	0.886	+0.182
	Latency [μs]	–	0.606	6.000	3.000	1.000	**0.268**	28.000	5.000	2.000	0.269	15.000	10.000	6.000	–
2	Accuracy	0.997	0.997	0.997	0.997	0.997	0.997	0.997	0.998	**0.998**	0.995	0.995	0.996	0.996	+0.001
	Recall	0.965	0.979	0.970	0.981	0.985	0.978	**0.994**	0.989	0.986	0.985	0.913	0.914	0.923	+0.029
	Precision	0.965	0.956	0.960	0.957	0.951	0.956	0.950	0.961	0.965	0.912	**0.988**	0.987	0.987	+0.023
	F_1 score	0.965	0.967	0.965	0.968	0.967	0.966	0.971	0.974	**0.976**	0.945	0.947	0.947	0.952	+0.011
	Latency [μs]	–	0.307	2.000	0.605	0.241	**0.037**	4.000	0.743	0.366	0.038	2.000	0.303	0.743	–

TABLE II
BEST PERFORMING MODELS AND UNDERLYING FACTORS INFLUENCING THEIR PERFORMANCE.

Classes	Metric	Best Configuration				Underlying Factors
		Model	Sampling	Best Result	Δ	
34	Accuracy	XGBoost	10%	0.996	+0.004	Handles complex 34-class decision boundaries with moderate sampling.
	Recall	XGBoost	5%	0.881	+0.049	Captures diverse attack patterns while minimizing false negatives.
	Precision	XGBoostRF	10%	0.920	+0.216	Forest ensembling reduces variance and lowers false positives.
	F_1 score	XGBoost	1%	0.880	+0.166	Early feature selection plus tuning yields balanced performance.
	Latency	LightGBM	Raw	0.799 μs	–	Histogram-based learning remains efficient on the full dataset.
8	Accuracy	XGBoost	5%	0.996	+0.003	Moderate sampling ensures strong coverage for 8-class tasks.
	Recall	XGBoost	1%	0.909	-0.001	Key features are retained even with minimal data, boosting detection.
	Precision	XGBoostRF	1%	0.962	+0.257	Ensemble approach mitigates overfitting to maximize precision.
	F_1 score	XGBoost	10%	0.901	+0.182	Larger sample refines class boundaries for a higher F_1.
	Latency	XGBoost	Raw	0.268 μs	–	No sampling overhead yields very low latency.
2	Accuracy	All Models	Raw	0.997	+0.001	Binary classification is simpler, so all models converge near max.
	Recall	XGBoost	1%	0.994	+0.029	High capture rate of attacks with minimal false negatives.
	Precision	XGBoostRF	1%	0.988	+0.023	Reduced false alarms through variance-limiting random forests.
	F_1 score	XGBoost	10%	0.976	+0.011	More training data further balances recall and precision.
	Latency	XGBoost	Raw	0.037 μs	–	Single-step detection latency remains exceptionally low for binary tasks.

performance and efficiency, whereas extreme sampling (1%) increases variability. Results confirm that latency depends more on model architecture and optimization than dataset size.

Overall, XGBoost achieves the highest F_1 **scores**, particularly in complex 34-class and 8-class tasks, showcasing its ability to handle intricate decision boundaries. XGBoostRF excels in **precision**, minimizing false positives, especially at 1% sampling. LightGBM performs well in binary classification but sacrifices recall and precision due to its efficiency-focused histogram learning. Feature selection, data balancing, and HPO significantly improve performance, validating the sample-based approach. Despite latency trade-offs, the pipeline remains well-suited for real-time IoT threat detection.

While F_1 scores and latency provide valuable insights, they alone do not offer a complete evaluation. Full execution time, including training and prediction, must also be considered alongside the trade-offs between speed and accuracy. To address this, a Pareto front analysis identifies configurations that optimally balance these metrics, providing a holistic assessment of model effectiveness.

B. Pareto Front Analysis

To evaluate total execution (i.e., training and prediction) time versus performance trade-offs, we analyze the Pareto front across models and sampling strategies (Figure 2). Table III summarizes the best-performing models in three key categories: highest F_1 score, lowest execution time, and the optimal balance between performance and efficiency, while also reporting CPU usage, RAM consumption, and model size to consider their feasibility for deployment on resource-constrained devices.

For 34-class detection, XGBoost at 1% sampling maximizes F_1 score by capturing key features, while LightGBM at 10% minimizes latency via efficient histogram-based learning. XGBoost at 10% best balances accuracy and speed.

In the 8-class scenario, more evenly distributed features allow XGBoost at 10% to achieve the highest F_1, while LightGBM at 1% delivers the lowest latency. XGBoostRF at 1% provides a well-balanced compromise between accuracy and computational efficiency.

For 2-class classification, simpler decision boundaries enable XGBoost at 10% to lead in F_1, with XGBoostRF at 5% maintaining precision through variance reduction. Lower sampling rates, such as LightGBM at 1%, excel when efficiency is the priority.

By analyzing Table III, we observe a trade-off between F_1 score optimization, inference latency, and balanced configurations across classification tasks. XGBoost achieves high F_1 scores but demands more CPU, peaking at 208.33% in the 34-class task. LightGBM is the most efficient for low-latency scenarios, using less RAM and CPU while maintaining a compact size, suggesting suitability for embedded deployment. In binary classification, all models have low RAM usage, but XGBoostRF consumes nearly twice the CPU of LightGBM when optimized for latency.

Sampling strategies optimize performance and execution time based on deployment needs, with larger samples improving complex models' decision boundaries and smaller samples favoring latency-sensitive tasks. XGBoost at 10% sampling offers the best trade-off between accuracy, efficiency, and scalability, making it a strong baseline for IoT threat detection.

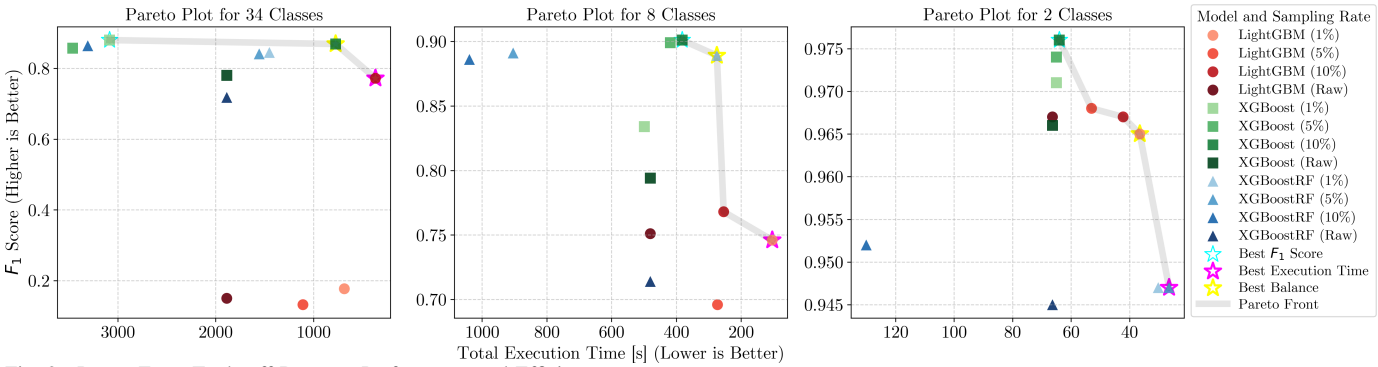

Fig. 2. Pareto Front Trade-off Between Performance and Efficiency.

TABLE III
BEST CONFIGURATIONS AND COMPUTATIONAL RESOURCE USAGE.

Classes	Best F_1 Score Configuration					Best Latency Configuration					Best Balanced Configuration				
	Model	Sampling	CPU$_{\%}$	RAM$_{MB}$	Size$_{MB}$	Model	Sampling	CPU$_{\%}$	RAM$_{MB}$	Size$_{MB}$	Model	Sampling	CPU$_{\%}$	RAM$_{MB}$	Size$_{MB}$
34	XGBoost	1%	89.52	13.13	5.01	LightGBM	10%	166.83	8.98	5.70	XGBoost	10%	208.33	127.50	7.54
8	XGBoost	10%	164.93	35.70	2.34	LightGBM	1%	56.70	1.54	2.53	XGBoostRF	1%	121.53	3.00	2.50
2	XGBoost	10%	82.47	10.80	0.34	XGBoostRF	5%	96.60	5.44	0.40	LightGBM	1%	4.34	0.88	0.33

C. Comparison with State-of-the-Art Approaches

While recent IoT threat detection methods achieve high accuracy, they face limitations: CNN-LSTM models [2] require GPU acceleration, restricting deployment options, and GSK-based optimization [3] performs well in binary classification but underperforms in multiclass scenarios. Our pipeline achieves comparable or superior recall and precision at lower cost. As shown in Tables I, II, and III, our integration of FS, data balancing, and HPO enhances performance across all tasks while reducing CPU usage and memory consumption. This efficiency makes the approach more suitable for deployment in resource-constrained environments.

D. Guidelines for Model and Data Sample Selection

Selecting the optimal model and training data size depends on dataset complexity, class diversity, and computational constraints. For highly diverse datasets, models benefiting from feature selection and tuning, such as XGBoost at 10% sampling, yield the best performance. In moderate-class scenarios, XGBoostRF at 1% balances accuracy and efficiency, while LightGBM at 1% is well-suited for binary classification. When feature importance is crucial, tree-based models like XGBoost and XGBoostRF are preferable, whereas lower-dimensional datasets benefit from LightGBM's efficiency at lower sampling rates. To optimize cost, 1–5% sampling enables efficient HPO, while 10% strikes a balance between performance and time.

IV. CONCLUSION

This study introduced a scalable, sample-based, multi-stage pipeline for IoT threat detection using the CICIoT2023 dataset. By integrating sampling, FS, data balancing, and HPO with LightGBM, XGBoost, and XGBoostRF, the pipeline significantly improved precision and F_1 scores, particularly in multiclass classification, while reducing computational overhead.

Sampling enabled training with large datasets without excessive resource usage while maintaining accuracy, ensuring optimal trade-off between time and performance. Experimental results showed that XGBoost achieved the highest F_1 scores, XGBoostRF excelled in precision, and LightGBM had the lowest latency, making it well-suited for real-time detection.

The pipeline generates optimized models with varying computational requirements. Lighter configurations, such as LightGBM with 1% sampling, require minimal resources, making them promising for embedded devices, while higher-performing models like XGBoost with 10% sampling are more demanding and better suited for edge devices. We suggest XGBoost with 10% sampling as the baseline for IoT threat detection, though performance may vary by deployment scenario.

Future work will explore adaptive sampling and AutoML to enhance generalizability and assess edge device deployment.

ACKNOWLEDGMENTS

This work was partially funded by CNPq (Grant Nos. 403612/2020-9, 311470/2021-1, and 403827/2021-3), by São Paulo Research Foundation (FAPESP) (Grant No. 2021/06946-0), by Minas Gerais Research Foundation (FAPEMIG) (Grant Nos. PPE-00124-23, APQ-04523-23, APQ-05305-23, and APQ-03162-24), Brasil 6G project (01245.020548/2021-07), supported by RNP and MCTI, and by the projects XGM-AFCCT-2024-2-5-1, and XGM-AFCCT-2024-9-1-1 supported by xGMobile – EMBRAPII-Inatel Competence Center on 5G and 6G Networks, with financial resources from the PPI IoT/Manufatura 4.0 from MCTI grant number 052/2023, signed with EMBRAPII.

REFERENCES

[1] M. V. C. Aragão, M. F. Carvalho, T. M. Pereira *et al.* (2024, May) Enhancing AutoML Performance for Imbalanced Tabular Data Classification: A Self-Balancing Pipeline. [Online]. Available: https://www.researchsquare.com/article/rs-4469436/v1

[2] S. Yaras and M. Dener, "IoT-Based Intrusion Detection System Using New Hybrid DL Algorithm," *Electronics*, vol. 13, no. 6, p. 1053, 2024.

[3] H. Q. Gheni and W. L. Al-Yaseen, "Two-Step Data Clustering for Improved IDS Using CICIoT2023 Dataset," *SSRN 4762201*, 2024.

[4] E. C. P. Neto, S. Dadkhah, R. Ferreira *et al.*, "CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment," *Sensors*, vol. 23, no. 13, p. 5941, 2023.

[5] M. V. C. Aragão, T. M. Pereira, F. A. P. Figueiredo *et al.*, "Source code for sample-based multi-stage ML pipeline," Feb. 2025. [Online]. Available: https://github.com/marcelovca90/CICIoT2023-Sampling-Pipeline

[6] T. pandas development team. (2020, Feb.) pandas-dev/pandas: Pandas.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort *et al.*, "Scikit-learn: ML in Python," *J. Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[8] M. B. Kursa and W. R. Rudnicki, "Feature selection with the Boruta package," *J. Statistical Software*, vol. 36, no. 11, pp. 1–13, 2010.

[9] I. Guyon, J. Weston, S. Barnhill *et al.*, "Gene selection for cancer classification using SVMs," *Mach. Learn.*, vol. 46, pp. 389–422, 2002.

[10] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, ser. NIPS'17. Curran Associates Inc., 2017, p. 4768–4777.

[11] T. Akiba, S. Sano, T. Yanase *et al.*, "Optuna: A Next-generation HPO Framework," in *ACM KDD 2019*, 2019, pp. 2623–2631.

[12] M. Cerliani. (2024, Feb.) shap-hypetune. [Online]. Available: https://github.com/cerlymarco/shap-hypetune