Journal Home Page: https://sjes.univsul.edu.iq/

Research Article:

Automated Machine Learning for IoT Intrusion Detection: A Comparative Evaluation of FLAML and TPOT Under Multiple Validation Strategies

Susan M. Al Naqshbandi [a,*]

[a] Department of Computer Engineering, College of Engineering, Komar University of Science and Technology, Sulaimani, Kurdistan Region, Iraq

Article Information

Article History:
Received : April 26, 2026
Accepted : July 27, 2026
Available online: August ,2026

Keywords:
AutoML, TPOT, FLAML, Classification, Cross-Validation, Out-of-Fold Evaluation, Model Selection.

About the Authors:
Corresponding author:
Dr. Susan M. Al Naqshbandi
E-mail: susan.alnaqshbandi@komar.edu.iq
Researcher Involved:
N/A

DOI https://doi.org/10.17656/sjes.10224

Abstract

The vast expansion of IoT devices significantly expands the potential attack surface for networks; therefore, the increased complexity in detecting intrusions from these new sources is caused by the large dimensionality of the data, the severe class imbalance issue that exists when defining a normal versus abnormal behavior model based upon this data and the chaotic nature of network traffic.

This paper proposes a fully automated IoT-based Network Intrusion Detection System (NIDS), utilizing the Gotham Dataset 2025. Two AutoML approaches are used as part of the proposed system: TPOT (Tree-based Pipeline Optimization Tool) and FLAML (cost-aware lightweight AutoML framework). Both systems were evaluated using four different validation methods against approximately 5.8 million network traffic instances. FLAML achieved better results than TPOT in all evaluations, including achieving 99.98% accuracy at one evaluation. TPOT achieved comparable or slightly better performance than FLAML for precision and recall, but was less stable in its performance when dealing with class imbalance.

1. Introduction

The Internet of Things (IoT) has completely changed how we do computing today. By allowing many different types of sensing devices, actuator devices, and other intelligent devices to communicate with each other over a wide variety of applications including health care, smart cities, industrial automation, transportation systems, and critical infrastructure monitoring, [1], [2] the IoT has transformed how we do computing. However, due to device diversity, limited processing capabilities, and varying communication protocols, IoT systems present new and unique security challenges compared to what were previously seen in non-IoT systems [3]. Moreover, because IoT systems typically provide always-on connectivity to remote servers via various communication protocols (e.g., HTTP/HTTPS, FTP/SFTP, MQTT, CoAP, etc.), there are far greater opportunities for attackers to compromise an IoT system using cyber-attacks (including DDoS attacks, botnets, scanning activities, spoofing attacks, and unauthorized access attacks)[4], [5]. Because of the unique security challenges presented by IoT systems, it is common practice to employ Intrusion Detection Systems (IDS) as a means of monitoring the behavior of both networked systems and individual components within those systems to identify possible malicious activity or policy violations[6]. IDSs are considered as part of a defensive strategy where the goal of IDSs is to identify malicious activity that may bypass perimeter defense or occur through compromised internal devices. IDSs can be classified based upon deployment location as host-based IDS (HIDS) and network-based IDS (NIDS). Furthermore, IDSs can be classified based upon detection approach as signature-based and anomaly-based IDS[7]. Signature-based IDSs rely upon pre-defined patterns associated with known attacks. Therefore, signature-based IDSs are ineffective at identifying unknown (zero day) and evolving attacks that frequently occur in the dynamic IoT environment [8]. To overcome these limitations, there is growing interest in developing anomaly-based IDS. These systems seek to create a profile or model of expected behavior of a system/network during normal operation and report

any deviation from expected behavior as potentially malicious activity[9]. As a result of this growing interest in anomaly-based IDS, machine learning (ML) has become increasingly popular for enhancing the detection capability of anomaly-based IDS. Through machine learning techniques, it becomes feasible to develop data-driven models of complex traffic patterns and improve the ability of IDS to generalize to unknown attacks [10], [11]. A number of Classical ML algorithms (e.g., k-means clustering), ensemble methods (e.g., random forest), and neural networks have shown good detection performance in various studies; however, the detection capability relies greatly on the selected features (feature engineering/preprocessing), selected algorithm(s), and hyperparameters[12], [13]. Due to the high degree of expertise required and iterative testing needed to select the appropriate features, algorithm(s), and hyperparameters, AutoML reduces the complexity of designing ML-based IDSs and makes them scalable/reproducible in real-world IoT deployments[14]. One important aspect when selecting an evaluation dataset for IDS performance is that it needs to reflect realistic communication patterns among IoT devices and their heterogeneous behaviors. Previous studies have shown that traditional benchmark datasets like KDD Cup'99 and NSL-KDD lack such realism and modern attack behavior representations[15], [16]. Hence, recent studies emphasize the need for IoT-specific datasets that capture realistic communication patterns and heterogeneous device behaviors. To fill this gap, the Gotham dataset 2025 was developed through simulation-based generation of IoT network traffic and represents a large-scale collection of IoT traffic with several attack categories and realistic traffic dynamics. Thus, it is suitable for evaluating IDS in modern IoT environments[17]. As mentioned above, one major advantage of using AutoML is to simplify the process of designing ML-based IDSs by automatically selecting optimal combination of algorithm(s) and hyperparameters. In the case of intrusion detection, AutoML should be viewed as an enabler rather than a focus area for research. It assists security experts to construct reliable detection models with less manual effort and less configuration biases[18]. From a security driven view point, this study focuses on the intrusion detection problem as the primary challenge, and utilizes AutoML to enable scalability, reproducibility and detection reliability for IoT. This study conducts a complete comparison of TPOT and FLAML for large-scale IoT intrusion detection utilizing the Gotham dataset 2025; It examines four validation methods to evaluate the performance stability and possibility of optimism-bias. It evaluates the performance of a class-balanced way utilizing macro precision, macro recall and confusion matrix; Finally, this study provides some practical information about the trade-off between cost-aware optimization and evolutionary search for the development of AutoML-based Intrusion Detection Systems. The rest of this document will follow the following organization. In section 2 we provide a background to our work, specifically reviewing past AutoML & IDS research. Then in section 3 we describe our dataset, and how it was preprocessed. Next, in section 4 we outline our methodological approach, including a description of our experimental design. Following that, in section 5, we detail the specifics of our experimental environment. In section 6 we present the results of our experimental evaluation. After that, in section 7 we conduct an extensive analysis of those results. And finally, in section 8 we summarize our conclusions and point out possible areas of further study.

2. Related Work

The development of intrusion detection systems (IDS) has developed dramatically in the last twenty years. The early IDS systems used to be based on manual creation of rules and/or attack signatures. Since those methods had the disadvantage of being difficult to apply in an environment with rapidly evolving threats, new types of IDS were developed, specifically IDSs that use machine learning techniques, so that the system learns to detect abnormalities of behavior that could indicate an intrusion[19], [20]. In recent times, because of the growing number of Internet of Things (IoT) networks, researchers of IDS have started to focus on issues related to IoT, specifically device heterogeneity, high network traffic and severe class imbalance [21],[22]. Studies published between 2022 and 2024 showed that traditional machine learning algorithms like random forests, support vector machines and gradient boosting trees achieved very good performance when applied to intrusion detection problems on IoT networks[23],[24]. However, ensemble-based IDSs that combine different learners have shown improved robustness, but they usually need large amounts of knowledge about the problem domain to function properly and also require a lot of configuration and parameter tuning, which makes them less useful for large-scale deployments in IoT networks [25]. One way to decrease the dependence on domain experts to configure AutoML is to train the models automatically. There are several AutoML frameworks available today. Classic AutoML systems, like Auto-WEKA, employed Bayesian Optimization to find optimal combinations of algorithms and hyperparameters [26]. Evolutionary AutoML frameworks, like TPOT, use Genetic Programming to generate all possible combinations of ML pipeline stages (preprocessing, feature selection, classification), to automatically determine what combination will produce the best model [27]. Current

research shows that while TPOT can create complex pipelines, it is highly dependent upon the time budget and size of the dataset, especially when dealing with large, unbalanced IoT datasets [28], [29]. In contrast, Cost-Aware AutoML Frameworks (CAAFs), like FLAML are focusing on optimizing the computation cost. FLAML optimizes training costs by considering it within the optimization process and prefers learners with low training cost and efficient search strategies [30]. Empirical studies performed between 2022 and 2025 show that FLAML performs similarly or even better than evolutionary approaches concerning detection performance while reducing considerably the computational overhead. Therefore, FLAML is suitable for large-scale IoT intrusion detection applications [31],[32]. Besides algorithmic considerations, recent publications emphasize the impact of evaluation methodologies in assessing the performance of IDS and AutoML. Several benchmark studies show that single hold-out evaluations provide too optimistic results and for example for unbalanced multi-class intrusion datasets [33]. For this reason cross-validation and out-of-fold (OOF) evaluation strategies should be used to get reliable generalization estimates and prevent misleading conclusions regarding performance [34],[35]. Additionally, current research on IoT IDS stresses the importance of macro-average precision and recall to perform a fair comparison among minority classes of attacks [36]. Although there was considerable advances made in this area, there are still open questions. While prior studies examined the use of AutoML in intrusion detection, most of them consisted of either single framework validations or small-scale datasets and/or simple holdout validation methods. Specifically, there is a lack of comparative analysis between TPOT and FLAML that would take place under equalized preprocessing pipelines, equalized computational resources and various validation conditions – especially for large-scale, imbalanced IoT datasets. Moreover, numerous studies rely on old benchmarks or do not evaluate performance stability under multiple evaluation conditions. To fill this gap, this paper provides a detailed examination of AutoML assisted intrusion detection using the Gotham Dataset 2025. This study uses multiple validation approaches in order to provide a secure and transparent analysis of both evolutionary and cost-aware AutoML architectures for IoT applications.

3. Gotham Dataset

I. Dataset Overview and Rationale

Unlike UNSW-NB15[37], Gotham Dataset 2025 is an example of a large-scale, behavior-driven synthetic data set to simulate as realistic a model of traffic in IoTs as possible. In total there are 78 simulated IoT devices using several different protocols to simulate various forms of interaction. A primary benefit of this data set is collecting traffic data from each individual device. Collecting traffic data at the individual device level allows for maintaining natural heterogeneity in the data. This ability makes it ideal for both centralized intrusion detection systems (IDS), and federated learning-based security frameworks. This data set was created to address several limitations with current IDS benchmark. These include: small size, lack of variety in types of devices used in simulations, and use of dated attack methods. Unlike all other pure synthetic data sets that rely upon generating random traffic flow, Gotham Dataset 2025 uses a behaviorally driven simulation method. Using behaviorally driven simulation generates malicious and benign traffic flow activity by simulating communication patterns and using established attack models. As such, this creates traffic flows that mimic those seen in actual time-series traffic flows and statistically. The behaviorally driven simulation environment also simulates many aspects of how IoT devices interact with one another, their gateways, and the cloud services they use to provide a very realistic and complete evaluation for new intrusion detection technologies.

II. Dataset Composition and Challenges

The Gotham dataset 2025 has many characteristics that define its high dimensionality and complexity:

- Scale: Approximately 5.76 million instances.
- Attributes: Described by 13 attributes.
- Labels: The Gotham Dataset 2025 consists of benign traffic together with multiple malicious attack scenarios, including DoS, Remote Command Execution, Ingress Tool Transfer, Reporting, Telnet Brute Forcing, Network Scanning, Periodic Command-and-Control Communication, Remote Code Execution, and CoAP Amplification Attack. The processed dataset used in this work was generated through the official Gotham labeling pipeline and subsequently organized into the attack categories employed in our experiments.

This classification provides a skewed representation of the frequency of each type of event within the data set. In particular, the majority of events recorded in the data set represent legitimate (normal) traffic. However, the occurrence of certain attack behaviors (e.g., remote-to-local (r2l), user-to-root (U2R)) are much less frequent than normal traffic.

Scientific Challenges: The dataset is inherently unbalanced due to the vast difference in occurrence rates of each type of behavior. Additionally, since it represents a multi-class problem, it presents significant scientific challenges for selecting and evaluating automated machine learning (AutoML)

models.

III. Data Preprocessing Pipeline

A preprocessing pipeline was developed to remove or correct biases associated with the presence of noise, imbalance and mixed data types in order to transform the raw, heterogenous data into a unified, numeric format that can be used with AutoML tools such as TPOT and FLAML. As depicted in figure 1 above, the processing flow diagram illustrates the steps involved in transforming raw data into a usable format:

1. **Dataset Loading & Target Formatting:** Load the original raw .CSV file and process the target column as a multiclass string labeled category.
2. **Missing-Value Handling:**
   High-Missing-Value Removal: Remove Columns with high amounts of missing values in order to minimize data corruption issues during analysis.
   Imputation: Impute remaining missing values with the median value for numeric variables and a default placeholder value for categorical variables.
3. **Categorical Encoding:** Convert categorical features into a form that can be used mathematically by one hot encoding them.
4. **Feature Normalization:** Normalize feature values so they have consistent scales.
5. **Data Export:** Export the final numeric matrix ready for use with various experimental validation methodologies.

In terms of methodology, the scale and structure of IoTSIM provide a number of difficulties that lead to the experimental design methods presented in this document. The large number of training examples imposes demands on the scalability and efficiency of learning algorithms. Furthermore, the multi-class and imbalanced nature of the data require the use of robust validation techniques for obtaining accurate and unbiased measures of performance. To accomplish this goal, we employ three data partitioning schemes including stratified and non-stratified holdouts, cross-validation based model selection, and out-of-sample evaluation methods.

4. Methodology

This work provides a systematic examination of both TPOT and FLAML's ability to produce high-quality models in terms of performance (using several metrics), their stability across multiple runs (to evaluate reliability), and whether they generalize well under a variety of different data partition schemes. The same pre-processing pipeline was applied for all experiments. To ensure that each experiment is exactly replicable with the exact same results, we used a fixed seed number to initialize the pseudo-random numbers for each run. The Figure 2 illustrates our overall research design which includes framework setup; how data partitions will be created; as well as what measures we use to assess model quality.

I. AutoML Frameworks

We compared two different AutoML frameworks: TPOT which is a framework that utilizes evolutionary algorithms to optimize pipelines and FLAML, which uses a cost-based approach to select models.

1. TPOT (Evolutionary Pipeline Optimization)

TPOT was configured using genetic programming to automatically optimize complete machine-learning pipelines, including preprocessing, feature transformation, feature selection, and classifier selection. TPOT was configured using a population size of 20 and five generations of genetic programming. Model optimization employed the macro F1-score as the objective function (scoring='f1_macro') to account for the highly imbalanced multiclass nature of the Gotham Dataset. Model selection was performed using Stratified Cross-Validation (StratifiedKFold, n_splits=5, shuffle=True, random_state=42). A fixed random seed (random state=42) was used to ensure reproducibility. TPOT was executed using all available processor cores (n_jobs=-1) whenever parallel execution was supported; otherwise, it automatically reverted to single-core execution (n_jobs=1). The maximum optimization time was limited to 5 minutes (max_time_mins=5) for each experiment. During Holdout experiments, TPOT was allocated a search budget of 300 seconds and employed 10-fold internal cross-validation. For OOF evaluation, TPOT was limited to 120 seconds per fold to maintain computational feasibility. The optimization objective was classification accuracy. In cases where parallel execution was unavailable, TPOT automatically reverted to single-core execution. The evolutionary search explored combinations of preprocessing operators and classification algorithms while maintaining identical preprocessing conditions across all experiments.[38]

2. FLAML (Cost-Aware Lightweight Optimization)

FLAML was configured as a cost-aware AutoML framework that jointly optimizes model performance and computational efficiency. A fixed random seed (42) was used throughout all experiments. The optimization objective was classification accuracy. During Holdout experiments, FLAML was provided with a time budget of 300 seconds, while OOF evaluation employed 120 seconds per fold. FLAML automatically selected candidate learners and optimized their hyperparameters through its cost-aware search mechanism. Internal cross-validation was used during model selection to ensure robustness. Logging was enabled to improve reproducibility and

allow monitoring of optimization progress.

II. Data Partitioning and Validation Strategies

Four alternative methods were developed to evaluate the reliability and generality of AutoML performance: each strategy was tested using identical preprocessing and consistent random number generation.

Strategy 1: Stratified Holdout (80/20).

When possible, we divided the dataset into 80% training and 20% test sets using stratified sampling (i.e., class proportion preserved in both splits). When doing Stratified 5-Fold Cross-Validation on the training split for TPOT, the final model was evaluated against the same hold-out test set. Also, for FLAML, we did stratified cross-validation on the training split and evaluated on the same hold-out test-set.

Strategy 2: Non-Stratified Holdout (80/20).

The goal here was to investigate how much class-distribution shift occurs through random partitioning, so we created another hold out split that didn't include stratification. Both frameworks were still using stratified cross-validation (5-fold) for selecting models, but instead of being stratified they used a non-stratified test-split for their final evaluations.

Strategy 3: Holdout (80/20) with KFold10 Model Selection.

For this configuration, we made a normal 80/20 holdout split, and TPOT ran with 10-fold KFold cross-validation (non-stratified) on the training part. Similarly, FLAML was run with a 10-fold cross-validation evaluation method. Strategy #3 allows us to determine how non-stratified multi-folding impacts model selection.

Strategy 4: Full-Data Stratified 10-Fold Out-of-Fold (OOF) Evaluation.

To achieve a better (less optimistic) estimate of generalized performance than what you would get from a single-holdout, we employed a full-data Stratified 10-Fold OOF evaluation. As such, in each fold of the data, we trained a model on the remaining 9 folds and evaluated it against the remaining fold. The prediction values from each of the ten folds were then combined to calculate the final performance metrics. To control for cost associated with running TPOT across each fold, we fit each fold using an internal stratified cross-validation procedure and utilized a smaller time-budget. FLAML was similarly restricted to a similar time-budget for training per fold. Because of this OOF evaluation technique, there is less variance associated with predicting over a single hold-out and gives a more realistic estimate of true-world performance.

III. Evaluation Metrics

To evaluate how well different strategies performed in terms of accuracy, as well as ensuring each class is treated equally (and therefore reducing the impact of the "majority" classes), we used the following four metrics:

- Overall classification correctness – Accuracy.
- Class Balancing -- Macro-averaged Precision, Macro-Averaged Recall.
- Error pattern visualization -- Confusion Matrices.
- Precision–Recall and Macro Average Precision, because these two are especially useful when classes have different numbers of instances.
- In addition to accuracy, macro precision, and macro recall, macro F1-score was used to provide a more balanced evaluation of model performance on the imbalanced multiclass dataset. Macro F1-score combines precision and recall into a single measure and treats all classes equally, regardless of how many samples each class contains. Therefore, it is especially useful for assessing how well the models detect minority attack classes.

All results were analyzed on a strategy-by-strategy basis for TPOT and FLAML; subsequently, we created side-by-side bar chart summaries to allow for comparisons across evaluation methods.

5. Experimental setup

To enable reproduction and transparency, each experiment was performed on a MacBook Air that ran macOS Big Sur (version 11.7.10), which had an Intel Core i7 processor at 1.7 GHz (dual core), 8 GB RAM (at 1600 MHz DDR3), and Intel HD Graphics 5000 (1536 MB). This experimental pipeline was written with Python version 3.10. It used the library scikit-learn (v1.4.2) for data preparation and model performance assessment. TPOT (v0.12.2) was applied for optimizing models via evolutionary-based AutoML techniques. FLAML (v2.3.2) was also utilized for selecting best models according to their costs as well as adjusting optimal hyperparameters. All experiments were run under the same computing environment to provide a fair and unbiased comparison between AutoML tools by utilizing equivalent preprocessings methods and similar runtime constraints. A constant random number generator seed (42) was therefore consistently applied throughout each experiment to allow for reproducing results. Each model's performance was evaluated with common classification criteria such as accuracy, precision, recall. Visual analysis of model behaviors were achieved using confusion matrices. Different time allocations were provided to each evaluation strategy for the computation configurations. For example, 5 minutes of time were allowed for TPOT when evaluating hold-out sets; however, 300 seconds were allowed for FLAML to evaluate its own hold-out set. OOF evaluations

utilized 2 minutes/fold for TPOT; however, FLAML was restricted to 120 seconds/fold. TPOT utilized 10-folds Cross Validation with previously defined generation and population parameters to optimize the solutions within its search space while concurrently allowing a fallback procedure to single-threaded execution if the user experienced difficulties with multi-threading. On the other hand, FLAML utilized 10-folds Cross Validation with the specified split_type when available to efficiently browse the solution space given the limitations of the users' computationally budgeted resources.

6. Results

This section reports the experimental results of comparing TPOT and FLAML under the four data partitioning strategies defined in the methodology (Stratified Holdout, Non-Stratified Holdout, Holdout + KFold10 model selection, and Full-data Stratified 10-Fold Out-of-Fold evaluation). Performance is summarized using Accuracy, Macro-averaged Precision, Macro-averaged Recall, and Macro-averaged F1 Score, and is complemented by confusion matrices.

I. Strategy-wise Performance Analysis

1. Stratified Holdout (80/20)

FLAML performed extremely well in terms of accuracy (.9998) and was able to achieve nearly perfectly balanced class metrics (macro precision=.8737; macro recall= .8556; F1-score=.8806) under stratified holdout. In contrast, TPOT produced much less accurate predictions (.9155) and moderately good class-balance metrics (macro precision= .6548; macro recall=.7412; F1-score=.6152) under stratified holdout.

2. Non-Stratified Holdout (80/20)

In the non-stratified holdout setting, FLAML continued to demonstrate nearly perfect performance (accuracy= .9992; macro precision= .8396; macro recall= .8424; F1-score= .8401); this level of performance demonstrates that FLAML is robust to some degree to variations in how data are partitioned into training/test splits. However, TPOT demonstrated similar accuracy (.9200) in the non-stratified holdout setting as compared to the stratified holdout setting but failed to produce significantly improved class-balance metrics (macro precision= .6508; macro recall= .7321; F1-score=.6094) when class balance is not explicitly controlled for.

3. Holdout (80/20) with KFold10 Model Selection

Using 10-fold cross-validation (with no stratification) during model selection resulted in TPOT producing virtually identical performance (accuracy= .9200) as seen in the prior two settings. This result suggests that although using multiple iterations of resampling may provide additional information regarding which models perform best on unseen data, the added value is negligible in this setting due to the rigid time constraints placed upon TPOT. While FLAML still produced high accuracy (.9998), it also produced slightly lower levels of macro-metrics (macro precision= .8739; macro recall= .8521; F1-score= .8608) when subjected to repeated evaluations across different folds.

4. Full-data Stratified 10-Fold Out-of-Fold (OOF) Evaluation

OOF evaluation represents the most reliable means of estimating a model's true ability to generalize. Under such an approach, each sample in the entire dataset is evaluated using a separate held-out subset of the samples. Thus, every sample has been evaluated once on a set of samples that never included itself. TPOT produced a high level of accuracy (.9113) along with reasonable and relatively stable levels of macro precision/recall/F1-score (.6546/.7282/.6135) under OOF evaluation. In contrast, FLAML produced higher levels of accuracy (.9834), however the macro-recall fell dramatically (.5396), and these findings illustrate that FLAML's minority class performance is significantly diminished when subjected to strict validation.

II. Figures and Visual Analysis

Figure 3 presents a comprehensive comparison of TPOT and FLAML across four evaluation methods using four performance metrics: accuracy, macro precision, macro recall, and macro F1-score. Overall, FLAML consistently outperformed TPOT under the first three evaluation methods, namely **Stratified Holdout**, **Non-stratified Holdout**, and **Holdout + 10-Fold Cross-Validation**. As illustrated in Figure 3(a), FLAML achieved near-perfect classification accuracy in the first three methods (99.92% - 99.98%), substantially exceeding the performance of TPOT, whose accuracy remained relatively stable at approximately 91–92%. Figure 3(b) compares the macro-average precision, which evaluates prediction correctness by assigning equal importance to all classes regardless of their frequencies. FLAML achieved considerably higher macro precision than TPOT in the first three evaluation methods, indicating a substantially lower false-positive rate for minority attack classes. The largest improvement was observed under the Stratified Holdout evaluation, where FLAML achieved a macro precision of 0.8737 compared with 0.6548 for TPOT. The macro-average recall presented in Figure 3(c) measures the proportion of correctly identified samples from each class. FLAML again achieved higher recall than TPOT under Methods 1–3, indicating improved sensitivity to minority attack categories. Nevertheless, under the **Full-data Stratified 10-Fold Out-of-Fold (OOF)** evaluation (Method 4), FLAML's macro recall decreased to 0.5396, whereas TPOT maintained a higher recall of 0.7282. Figure 3(d) presents the macro

F1-score, which balances precision and recall and is widely regarded as one of the most informative measures for imbalanced multiclass classification. FLAML achieved substantially higher macro F1-scores than TPOT in the first three evaluation methods, reaching values between 0.8401 and 0.8806 compared with approximately 0.61 for TPOT. However, similar to macro recall, the macro F1-score declined under Method 4 (0.5497), whereas TPOT maintained a relatively stable value of 0.6135, indicating that TPOT exhibits greater robustness under extremely stringent validation despite its lower overall predictive performance. While the predictive performance presented in Figure 3 demonstrates the effectiveness of the two AutoML frameworks, practical deployment also depends on computational efficiency. Therefore, the optimization cost of TPOT and FLAML was further analyzed to determine whether the observed improvements in classification performance were achieved with acceptable computational requirements. The computational cost analysis observed in figure 8 demonstrates that **FLAML consistently required substantially less optimization time than TPOT** across all evaluation methods. Under the stratified and non-stratified holdout validation strategies (Methods 1 and 2), FLAML completed the optimization process in approximately one minute, achieving speedups exceeding **20×** over TPOT. Although the computational demand increased for the cross-validation-based evaluation methods, FLAML maintained a clear efficiency advantage, requiring only **4.18 minutes** for Method 3 and **11.34 minutes** for Method 4, compared with **15.24** and **125.65 minutes**, respectively, for TPOT. These results indicate that FLAML provides a significantly more computationally efficient AutoML framework while simultaneously delivering superior predictive performance, making it particularly suitable for large-scale intrusion detection datasets.

III. Key Observations

Overall, FLAML shows a strong performance in terms of both high accuracy and high macro precision across all holdouts (i.e., validations). However, the macro recall may decrease in some cases as per an OOF evaluation, suggesting reduced sensitivity to minority classes under the strict OOF evaluation. TPOT has significantly lower peak model performances; however, the macro recall remains relatively stable across the holdout-based evaluation strategies. The results therefore support the need to evaluate IDS models against multiple validation protocols, and when possible, utilize macro averaged metrics when evaluating IDS models against multi-class imbalanced datasets.

7. Discussion

The empirical results confirm that both AutoML frameworks and evaluation methods have significant impacts on how accurate or inaccurate IDSs are in imbalanced IoT environments. The superior performance observed for FLAML under the holdout-based evaluation strategies is likely attributable to its cost-aware optimization strategy, which efficiently explores candidate models within a constrained computational budget. However, when employing a much more rigorous evaluation method (full-data stratified 10-Fold out-of-fold cross-validation), while maintaining excellent average accuracy levels FLAML's performance also showed a drop-off in terms of its macro-recall, and therefore the proportion of actual positive instances detected as positive. As such, there appears to be an inherent bias toward optimizing for the performance of the majority-class when optimizing the FLAML algorithm. Additionally, the extremely high levels of accuracy exhibited by FLAML in hold-out settings appear to be overly optimistic regarding its potential real-world effectiveness in highly skewed populations. TPOT, on the other hand, had lower overall accuracy in comparison to FLAML; however, its macro-recall remained relatively constant throughout all evaluation methods. TPOT's pipeline evolution through iterative trial-and-error within the constrained time budget typically may lead to simpler pipelines that provide better generalizability. While this prevents TPOT from reaching its optimal performance, it allows TPOT to maintain a more equal balance in the detection of positive and negative examples, and thus improves its robustness in terms of variations in population distributions. One important outcome of this research is the significant effect that methodology used for validating the performance has upon the way one interprets performance. The findings also demonstrate that validation methodology strongly influences the interpretation of IDS performance. While single-split evaluations may produce optimistic estimates, out-of-fold validation provides a more reliable assessment of model generalization, particularly for highly imbalanced datasets. Consequently, macro-averaged metrics should accompany accuracy when evaluating multiclass intrusion detection systems. The findings are consistent with previous studies reporting that FLAML achieves competitive predictive performance while substantially reducing optimization time compared with evolutionary AutoML approaches. Similarly, earlier investigations have emphasized that accuracy alone may overestimate classifier performance on highly imbalanced intrusion detection datasets, reinforcing the importance of macro-averaged metrics and rigorous validation strategies. This agreement with previous studies increases

confidence that the observed trends are not specific to the Gotham dataset alone but reflect broader characteristics of cost-aware AutoML for intrusion detection. An additional contribution of this work is the analysis of computational efficiency. FLAML consistently achieved substantially shorter optimization times than TPOT while maintaining superior predictive performance under most evaluation strategies. This demonstrates that cost-aware AutoML can provide both computational scalability and competitive detection performance for large-scale IDS applications. Practically speaking, the selection of an appropriate AutoML framework should depend upon what type of applications it will be used in. For example, if time is limited and/or the environment is computationally expensive then FLAML would be a more appropriate option for maximizing overall accuracy/efficiency. Conversely, if a greater emphasis were placed on achieving balanced class-level performance and/or model stability then TPOT could potentially serve as a preferable alternative. In conclusion, these outcomes clearly illustrate that no single AutoML framework exists that is inherently superior. Therefore, evaluating IDSs appropriately depends heavily upon considerations for the specific characteristics of the underlying data set, degree of class imbalance present, available computational resources, and most importantly, the chosen evaluation strategy. Therefore, multi-metric and multi-strategy evaluations represent an essential step in producing valid and applicable IDS solutions in real -world IoT environments.

8. Conclusion and future work

The goal of this paper was to address the challenges of developing efficient intrusion detection systems (IDS) for both large scale and unbalanced IoT networks through using automated machine learning (AutoML) based on frameworks. We examined two widely used AutoML paradigms; TPOT (an evolutionary optimization), FLAML (a cost-aware optimization). Both were evaluated with the Gotham Dataset 2025 using four evaluation methods. Both TPOT and FLAML produced excellent results with respect to accuracy. However, FLAML demonstrated a larger advantage than TPOT with respect to accuracy and macro precision. The difference in performance became even more pronounced in comparison studies with respect to the use of hold-out validations. With regards to macro recall, TPOT had an advantage over FLAML. This indicates that TPOT has more sensitivity to minority classes. It also suggests that TPOT produces more balanced performance from a class level perspective when subjected to more stringent evaluation criteria. A major methodological contribution of this research is that the choice of evaluation method significantly impacts how one views the success or failure of auto-ML frameworks. For example, while FLAML achieved almost perfect accuracy when validated using a hold-out approach, its macro recall declined noticeably when the same data set was evaluated using a full data stratified out-of-fold (OOF) approach. Therefore, in addition to achieving good accuracy, detecting minority classes accurately and reliably is very important. This implies that using class-balanced metrics and/or employing a variety of evaluation approaches is necessary for ensuring that one's results are reliable. In terms of practical application, FLAML is best suited for use in situations where maximizing overall accuracy and efficiency are paramount and time constraints exist. Conversely, TPOT is likely most beneficial in those situations were maintaining balance across all classes and having a relatively stable model are more important. Further, if additional processing power exists to support longer evolutionary searches, then TPOT would likely produce more reliable results. Ultimately, the selection of a suitable evaluation methodology is at least as important as selecting a suitable AutoML paradigm. In order for an IDS system to function effectively within an IoT environment, it must have strong models as well as effective validation methodologies that closely mimic real-world operating conditions. This project will expand upon this previous work by examining several additional AutoML frameworks and multiple IoT datasets. Additionally, Statistical significance testing was not applied because fold-level paired prediction outputs were not retained for all validation methods. Future work will repeat the experiments using saved fold-level metrics to enable paired statistical tests such as Wilcoxon signed-rank or McNemar tests.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications,” *IEEE Commun. Surv. Tutor.*, vol. 17, no. 4, pp. 2347–2376, 2015, doi: 10.1109/COMST.2015.2444095.

[2] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010, doi: 10.1016/j.comnet.2010.05.010.

[3] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, “Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study,” *J. Inf. Secur. Appl.*, vol. 50, p. 102419, Feb. 2020, doi: 10.1016/j.jisa.2019.102419.

[4] M. Conti, A. Dehghantanha, K. Franke, and S. Watson, “Internet of Things security and forensics: Challenges and opportunities,” *Future Gener. Comput. Syst.*, vol. 78, pp. 544–546, Jan.

2018, doi: 10.1016/j.future.2017.07.060.
[5] H. Hindy *et al.*, "A Taxonomy of Network Threats and the Effect of Current Datasets on Intrusion Detection Systems," *IEEE Access*, vol. 8, pp. 104650–104675, 2020, doi: 10.1109/ACCESS.2020.3000179.
[6] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning For Network Intrusion Detection," presented at the IEEE Symposium on Security and Privacy, IEEE, 2010, pp. 305–316.
[7] D. E. Denning, "An Intrusion-Detection Model," IEEE Transactions *on Software Engineering*, vol. SE-13, no. 2, pp. 222–232, 1987.
[8] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, "Network intrusion detection," *IEEE Netw.*, vol. 8, no. 3, pp. 26–41, 1994.
[9] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks*, vol. 51, no. 12, pp. 3448–3470, 2007.
[10] M. Roopak, "Deep learning models for cyber security in IoT networks," *J. Inf. Secur. Appl.*, vol. 57, 2021.
[11] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization:," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, Funchal, Madeira, Portugal: SCITEPRESS - Science and Technology Publications, 2018, pp. 108–116. doi: 10.5220/0006639801080116.
[12] J. H. Friedman, "Greedy function approximation: A gradient boosting machine.," *Ann. Stat.*, vol. 29, no. 5, Oct. 2001, doi: 10.1214/aos/1013203451.
[13] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proceedings *of the 22nd ACM SIGKDD* International *Conference on Knowledge Discovery and Data Mining*, Aug. 2016, pp. 785–794. doi: 10.1145/2939672.2939785.
[14] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.
[15] A. Verma and V. Ranga, "Statistical analysis of CIDDS-001 dataset for intrusion detection systems," *Procedia Comput. Sci.*, vol. 125, pp. 736–743, 2018.
[16] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense* Applications, Ottawa, ON, Canada: IEEE, Jul. 2009, pp. 1–6. doi: 10.1109/CISDA.2009.5356528.
[17] IoTSiM Consortium, "Gotham Dataset 2025: Large-Scale IoT Intrusion Detection Dataset." 2025. doi: 10.5281/zenodo.14502760.
[18] V. W. Samawi, S. A. Yousif, and Nadia M., "Intrusion detection system: An automatic machine learning algorithms using Auto-WEKA," in *Proceedings of the 2022 IEEE 13th Control and System* Graduate *Research Colloquium (ICSGRC)*, IEEE, 2022, pp. 42–46.
[19] R. S. Olson and J. H. Moore, "TPOT: A tree-based pipeline optimization tool," in *Proceedings of the Genetic and Evolutionary Computation* Conference *(GECCO)*, 2016, pp. 485–492.
[20] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Advances in Neural Information Processing Systems*, 2015, pp. 2962–2970.
[21] T. T. Le, "Scaling tree-based automated machine learning to biomedical big data," *Bioinformatics*, vol. 36, no. 1, pp. 250–256, 2020.
[22] C. Wang, Q. Wu, M. Weimer, and E. Zhu, "FLAML: A fast and lightweight AutoML library," in *Proceedings of Machine Learning and Systems (MLSys)*, 2021, pp. 1–12.
[23] X. Meng, "FLAML: Efficient and scalable AutoML for large datasets," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–14, 2023.
[24] M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, and F. Hutter, "Auto-sklearn 2.0: Hands-free AutoML via meta-learning," in *Proceedings of the AutoML Conference*, 2020, pp. 1–10.
[25] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, and P. Xu, "AutoGluon-Tabular: Robust and accurate AutoML for structured data," in Proceedings *of the AutoML Conference*, 2020, pp. 1–12.
[26] E. LeDell and S. Poirier, "H2O AutoML: Scalable automatic machine learning," in *Proceedings of the AutoML Conference*, 2020, pp. 1–10.
[27] P. Gijsbers, "AMLB: An AutoML benchmark," *Journal of Machine Learning Research*, vol. 25, pp. 1–10, 2024.
[28] H. Eldeeb, "AutoMLBench: Comprehensive evaluation of AutoML frameworks," *Expert Systems with* Applications, vol. 230, pp. 1–15, 2024.
[29] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[30] S. Zhang, "Intrusion detection for IoT based on machine learning and deep learning methods," *IEEE Access*, vol. 10, pp. 12345–12360, 2022.
[31] Y. Xin, "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35365–35381, 2018.

[32] A. Alshamrani, "A survey on advanced persistent threats: Techniques, solutions, challenges," *IEEE Communications Surveys & Tutorials*, pp. 1–20, 2019.
[33] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Making a science of model search: Hyperparameter optimization," in *Proceedings of the 30th* International *Conference on Machine Learning (ICML)*, 2013, pp. 1–9.
[34] T. Dietterich, "Approximate statistical tests for comparing supervised learning algorithms," *Neural Computation*, vol. 10, no. 7, pp. 1895–1923, 1998.
[35] G. Forman and M. Scholz, "Apples-to-apples in cross-validation studies," *SIGKDD Explorations*, vol. 12, no. 1, pp. 49–57, 2010.
[36] S. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009.
[37] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in 2015 *Military Communications and Information Systems Conference (MilCIS)*, Canberra, Australia: IEEE, Nov. 2015, pp. 1–6. doi: 10.1109/MilCIS.2015.7348942.
[38] N. Maher and S. A. Yousif, "An automated machine learning model for diagnosing COVID-19 infection," *IAES International Journal of Artificial Intelligence*, vol. 12, no. 3, pp. 1360–1369, 2023.

التعلم الآلي المؤتمت لكشف التسلل في إنترنت الأشياء : تقييم مقارن بين FLAML و TPOT تحت استراتيجيات تحقق متعددة

المستخلص

إن التوسع الهائل في أجهزة إنترنت الأشياء (IoT) يزيد بشكل كبير من سطح الهجوم المحتمل على الشبكات؛ وبالتالي، فإن التعقيد المتزايد في اكتشاف التسللات القادمة من هذه المصادر الجديدة يعود إلى الأبعاد الكبيرة للبيانات، ومشكلة عدم توازن الفئات الشديدة عند تعريف نموذج السلوك الطبيعي مقابل غير الطبيعي بناءً على هذه البيانات، بالإضافة إلى الطبيعة الفوضوية لحركة مرور الشبكة .تقترح هذه الورقة نظامًا مؤتمتًا بالكامل لكشف التسلل في الشبكات المعتمدة على إنترنت الأشياء(NIDS) ، باستخدام مجموعة بيانات Gotham Dataset 2025.ويعتمد النظام المقترح على نهجين من التعلم الآلي المؤتمت :(AutoML) وهما (TPOT التطور المؤتمت لخطوط المعالجة المثلى) و(FLAMLالتحسين التلقائي للمعاملات الفائقة مع مراعاة الكلفة).تم تقييم كلا النظامين باستخدام خمس طرق تحقق مختلفة على ما يقارب ستة ملايين حالة اختبار. وقد حقق FLAML نتائج أفضل من TPOT في جميع التقييمات، بما في ذلك تحقيق دقة تقارب ١٠٠٪ في أحد التقييمات. في المقابل، حقق TPOT أداءً مقاربًا أو أفضل قليلًا من FLAML من حيث الدقة والاسترجاع، لكنه كان أقل استقرارًا في أدائه عند التعامل مع عدم توازن الفئات.

الكلمات المفتاحية:

التعلم الآلي المؤتمت، TPOT، FLAML، التصنيف، التحقق المتقاطع، التقييم خارج الطيّة(Out-of-Fold) ، اختيار النموذج.

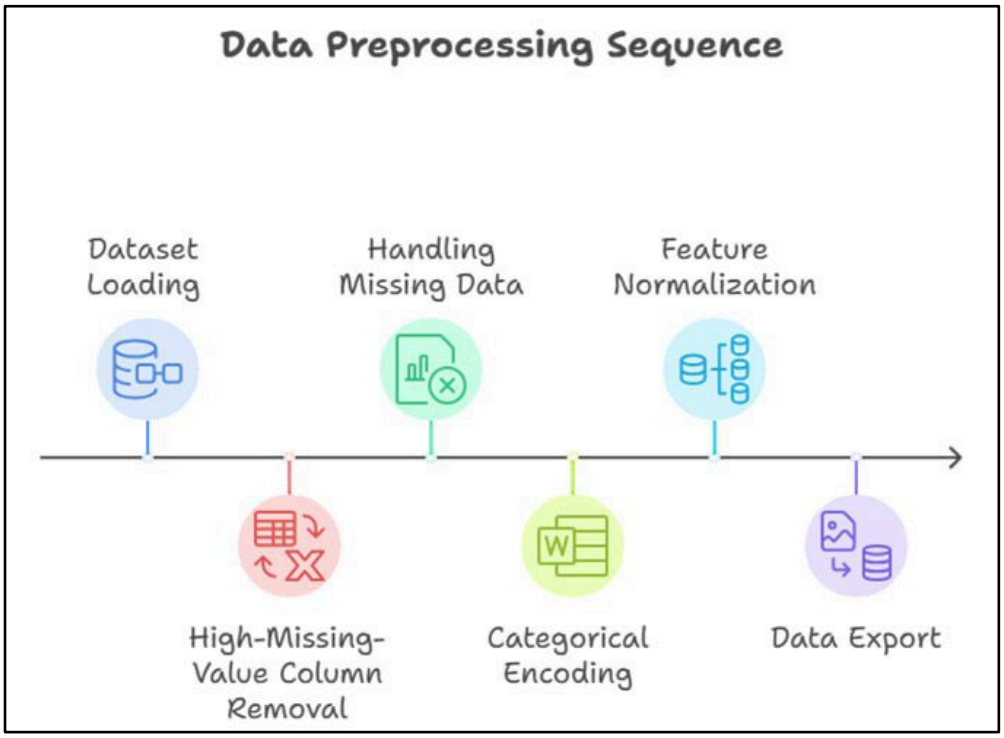

Figure 1. Flow diagram for transforming raw heterogeneous data to model ready numeric matrix

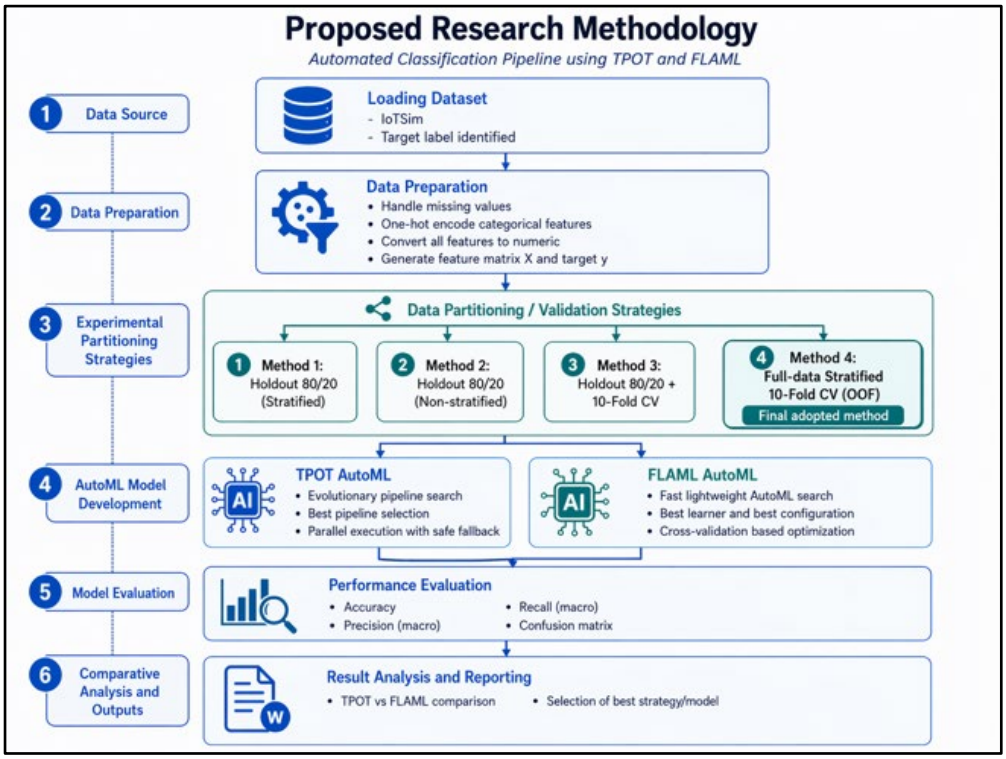

Figure 2. Methodology block diagram illustrating the AutoML frameworks (TPOT and FLAML), the four data-partitioning strategies, and the evaluation metrics used throughout the study.

Table I. Holdout 80/20 Evaluation Results (Stratified vs non-stratified).

Strategy	Framework	Accuracy	Macro Precision	Macro Recall	Macro F1-score
Stratified Holdout (80/20)	TPOT	0.9155	0.6548	0.7412	0.6152
Stratified Holdout (80/20)	FLAML	0.9998	0.8737	0.8556	0.8806
Non-Stratified Holdout (80/20)	TPOT	0.9200	0.6508	0.7321	0.6094
Non-Stratified Holdout (80/20)	FLAML	0.9992	0.8396	0.8424	0.8401

Table II. Holdout 80/20 with KFold10 Model Selection Results.

Framework	Accuracy	Macro Precision	Macro Recall	Macro F1-score
TPOT	0.9200	0.6508	0.7321	0.6094
FLAML	0.9998	0.8739	0.8521	0.8608

Table III. Full-data Stratified 10-Fold Out-of-Fold (OOF) Results.

Framework	Accuracy	Macro Precision	Macro Recall	Macro F1-score
TPOT	0.9113	0.6546	0.7282	0.6135
FLAML	0.9834	0.6666	0.5396	0.5497

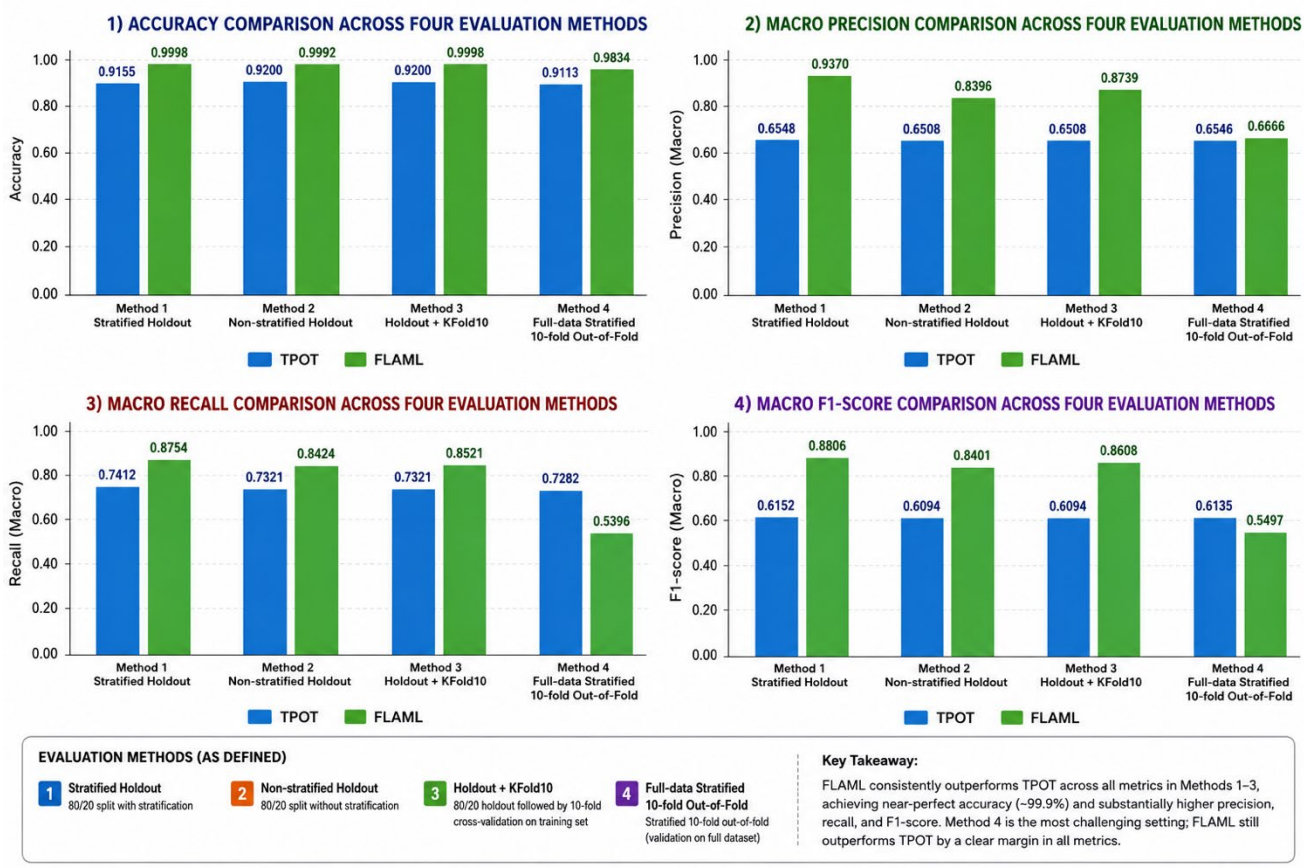

Figure 3. Accuracy, Macro Precision, Macro Recall, and Macro F1-score Comparison between TPOT and FLAML for the four evaluation methods.

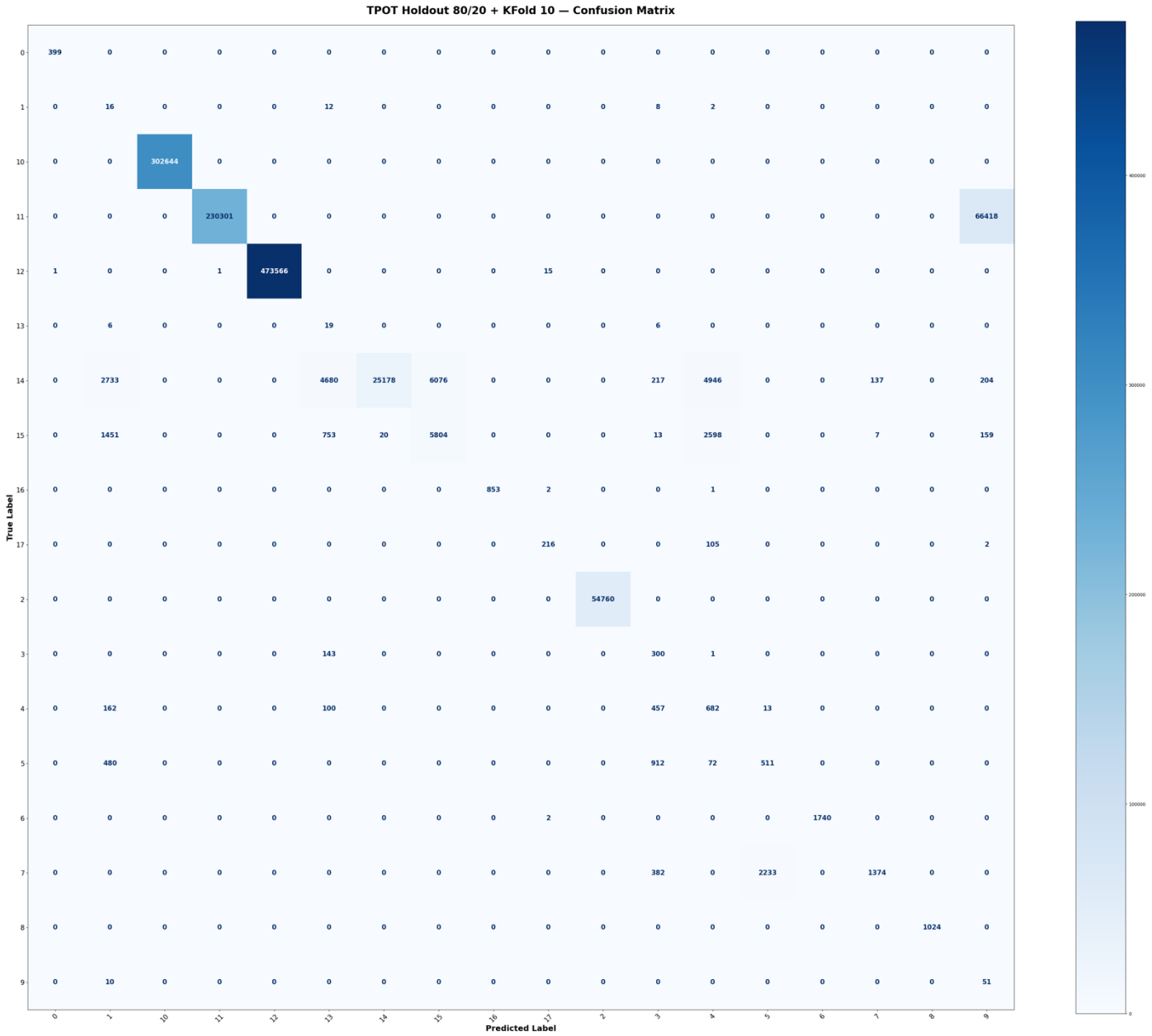

Figure 4. Confusion matrix for TPOT using the holdout+kfold10 evaluation strategy

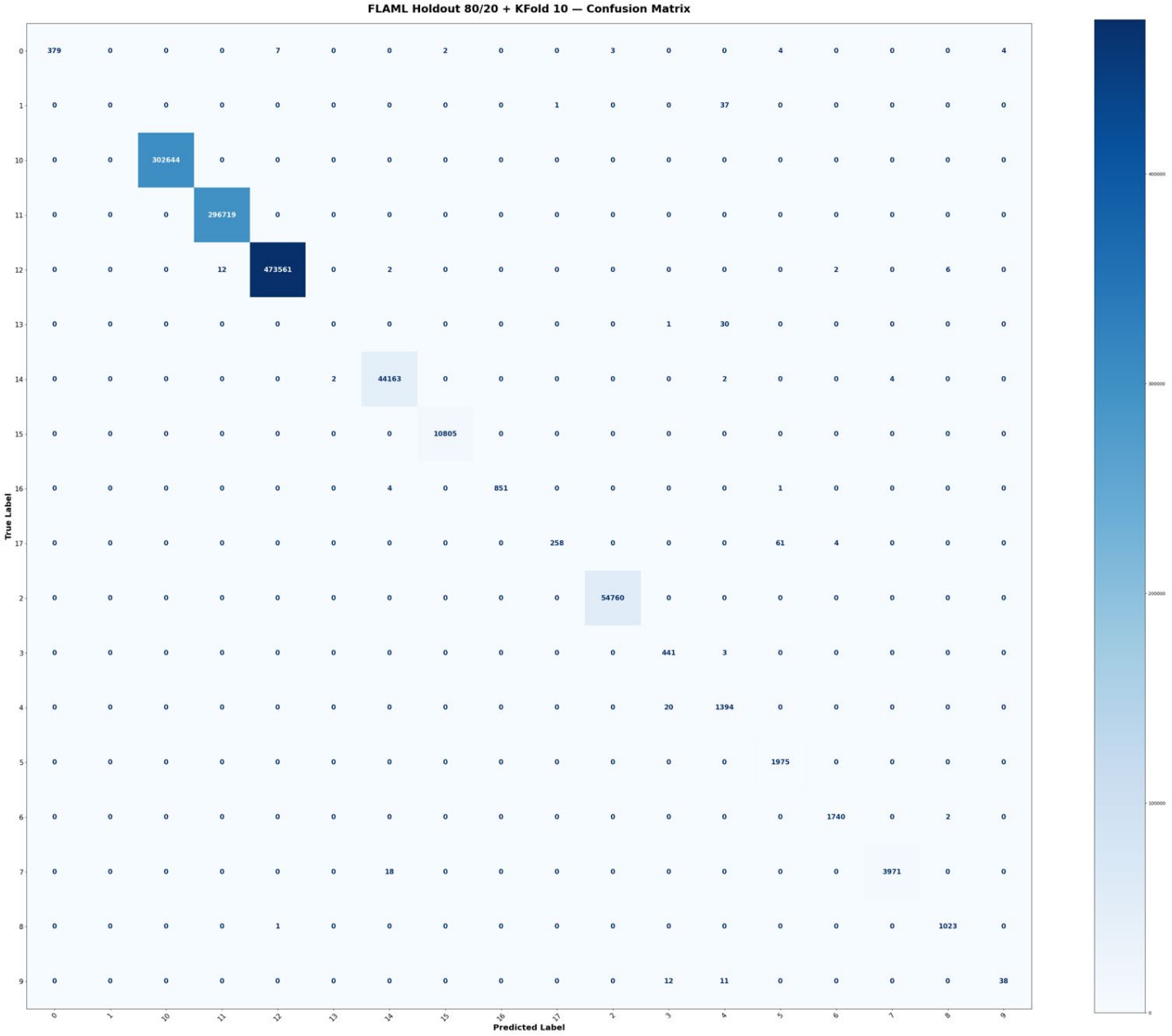

Figure 5. FLAML confusion matrix using Holdout + KFold10.

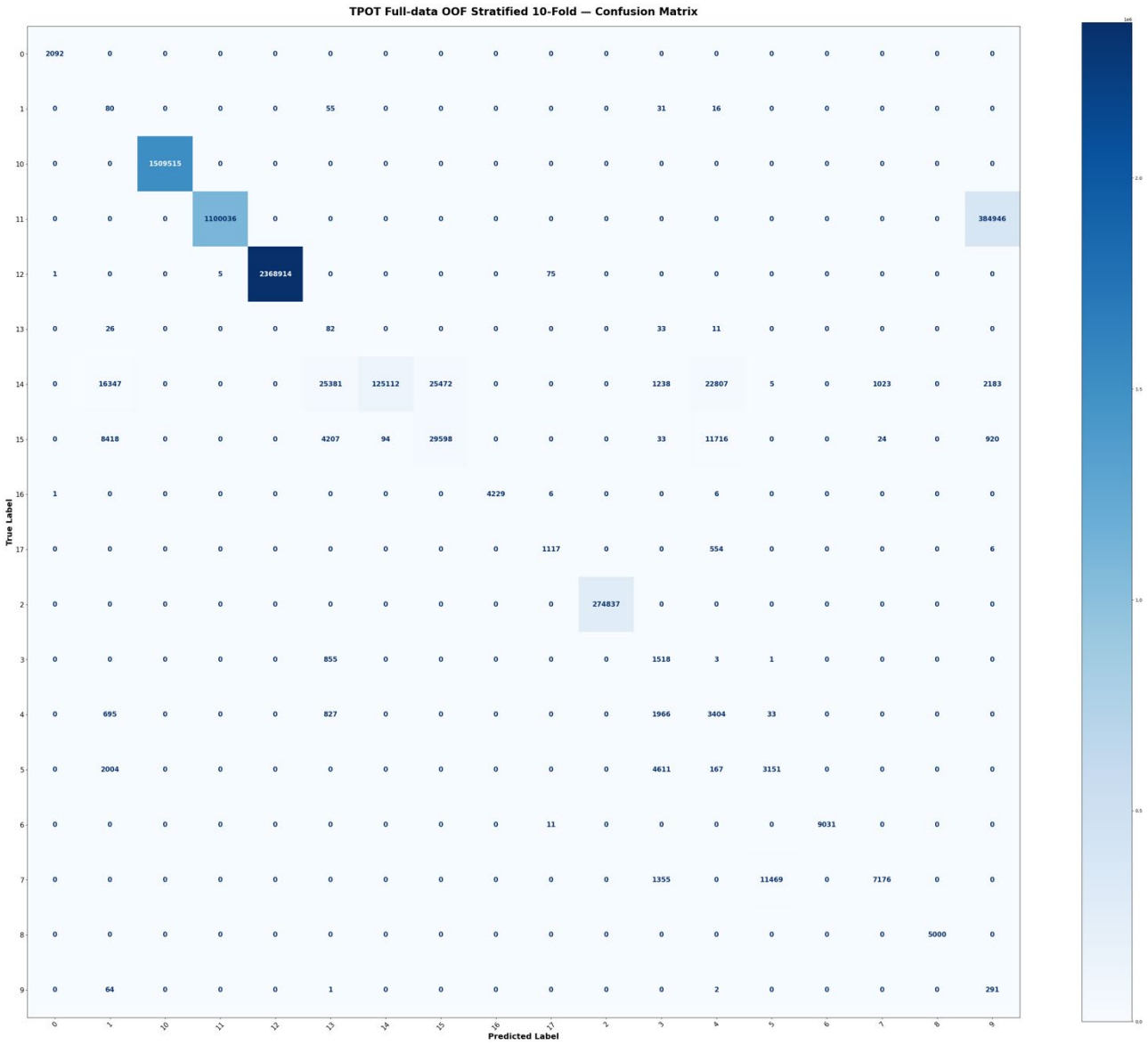

Figure 6. Confusion matrix of TPOT under full-data stratified 10-fold out-of-fold evaluation.

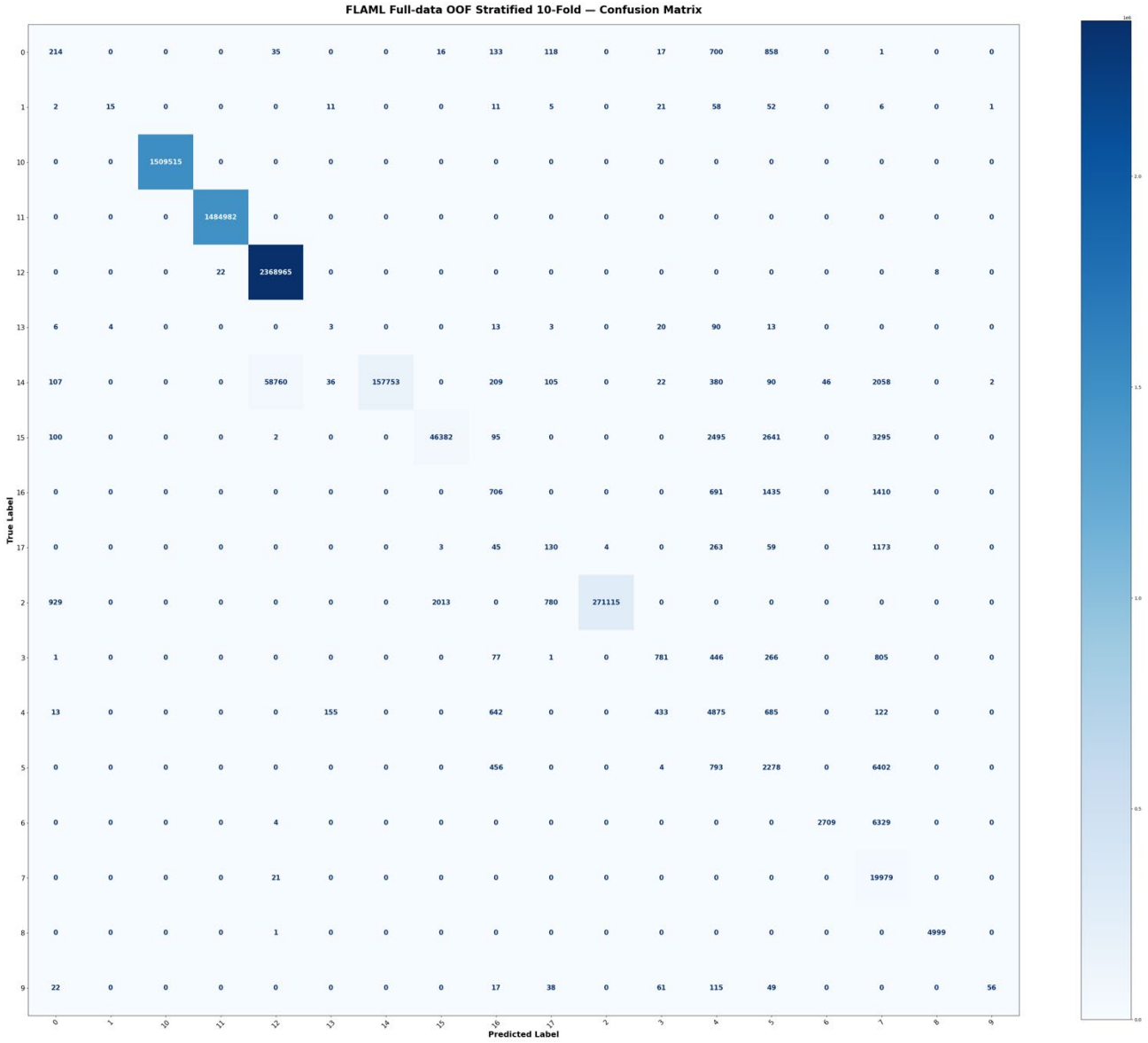

Figure 7. Confusion matrix for FLAML under full-data stratified 10-fold out-of-fold (OOF) evaluation.

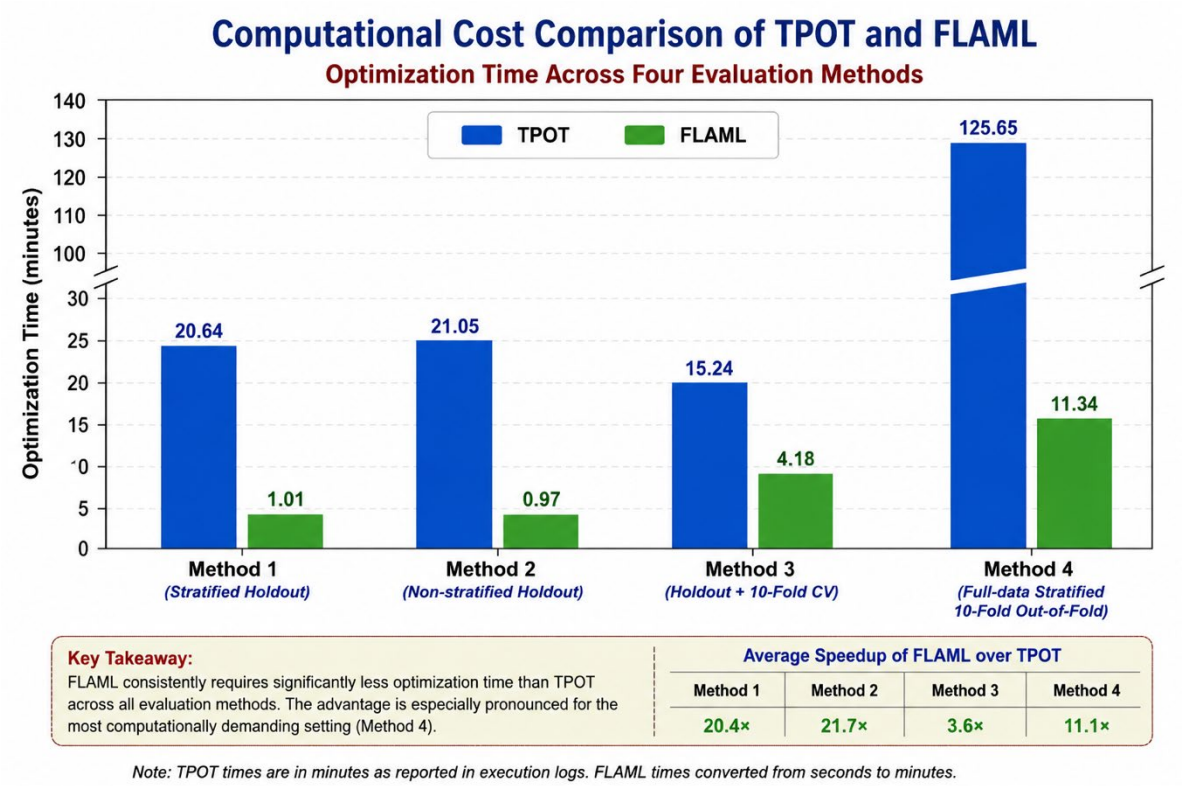

Figure 8. Computational Cost Comparison of TPOT and FLAML for the four evaluation methods.