

Performance And Efficiency Analysis Of Lightweight Intrusion Detection Models For Cloud Microservices

Abdullah Albalawi

Department of Computer Science, College of Computing and Information Technology,
Shaqra University, Shaqra, Saudi Arabia aalbalawi@su.edu.sa

Abstract: Cloud computing and microservices have revolutionized contemporary application development and deployment. Modern application architecture ensures scalability, reliability, and quick development. However, cloud computing expands cloud infrastructure vulnerability. Network-related threats, such as distributed denial-of-service, port scanning, brute force authentication, and hacking, remain common cloud threats in a microservices setup. The Intrusion Detection System (IDS) has a critical role to play in securing cloud infrastructure. Yet, most intrusion detection systems in use today rely on machine-learning and deep-learning techniques, which remain highly computation-intensive, contradictory to cloud-native application services. This paper examines whether lightweight machine learning models can meet intrusion detection requirements without imposing prohibitive overhead. Using the CICIDS2017 dataset, eight multi-day traffic files are merged and pre-processed to produce a large-scale, realistic, and class-imbalanced dataset. Lightweight linear classifiers are evaluated against a Random Forest baseline, with performance assessed through standard detection metrics accuracy, precision, recall, and F1-score together with training time and analysis using confusion matrices and performance-efficiency visualizations. The results show that Random Forest delivers near-perfect detection but at a computational cost that limits deployability in microservices environments. Lightweight linear models achieve comparable detection performance with substantially lower training time and resource consumption; Logistic Regression offers the most consistent balance between accuracy and F1-score, while SGDClassifier excels in efficiency and scalability. These findings establish lightweight IDS models as a practical and scalable alternative to heavyweight ensemble approaches for securing cloud microservices.

Keywords: intrusion detection system, cloud security, microservices architecture, lightweight machine learning, performance-efficiency trade-off

I. INTRODUCTION

Microservices architectures dominate current cloud deployments, replacing centralized application designs with collections of networked services. This distribution of functionality across independently deployable components substantially increases the exposed attack surface.

Security assumptions that once relied on clear network boundaries are often ineffective in this setting. Each microservice introduces an additional attack surface, while the predominance of east-west traffic obscures malicious behavior within internal service-to-service communication. Routine cloud operations intensify these challenges. Services scale automatically, workloads move between hosts, and traffic patterns shift rapidly, often without explicit operator intervention. As a result, the network context in which attacks occur is unstable and difficult to model with static assumptions.

Intrusion Detection Systems (IDS) continue to serve as a principal tool for observing network traffic and flagging malicious behavior. Signature-based systems remain effective for known attack patterns but provide little coverage when adversaries deviate from previously observed behavior. Machine learning-based IDS have therefore gained attention, as they infer attack patterns directly from traffic data rather than relying on predefined signatures.

Much of the recent IDS literature prioritizes detection accuracy, often through deep learning architectures or complex ensemble models. These methods often report strong results on benchmark datasets, but the performance gains rely on substantial computational overhead. Long training times and high memory consumption make them difficult to reconcile with microservices deployments, where latency budgets are tight, and resource efficiency is a practical constraint.

This paper adopts a different position. It argues that lightweight machine learning models offer a more practical foundation for intrusion detection in cloud microservices, provided their performance is evaluated alongside their computational footprint. The focus shifts from accuracy alone to the performance–efficiency trade-off, asking whether simpler models can deliver reliable detection while remaining deployable at scale.

The main contributions of this work are as follows:

- Construction of a large-scale and realistic IDS dataset through the merging and pre-processing of all CICIDS2017 traffic files.
- Systematic evaluation of multiple lightweight machine learning models under class-imbalanced conditions.
- Quantitative and visual analysis of detection performance and computational efficiency.
- Practical insights into the suitability of lightweight IDS models for deployment in cloud microservices environments.

II. Related Work

Intrusion detection has occupied a central place in network security studies for decades. Early IDS systems relied on signature- and rule-based mechanisms, exemplified by tools such as Snort and Suricata. These approaches perform reliably against known attack patterns but offer little protection against zero-day threats and depend on continual manual rule updates.

Machine learning was introduced to overcome these limitations. Initial research focused on classical algorithms, including Decision Trees, k-Nearest Neighbors, and Support Vector Machines. Mukkamala et al. showed that SVMs can achieve high detection accuracy while noting clear scalability constraints on large datasets [1], and Sabhnani and Serpen highlighted the tension between accuracy and computational cost in comparative evaluations of machine learning classifiers [2].

Ensemble methods later gained prominence, with Random Forest emerging as a dominant approach due to its robustness and strong classification performance. Numerous studies report consistently high detection accuracy for Random Forest-based IDS on benchmark datasets [3, 4]. At the same time, Buczak and Guven observed that ensemble techniques impose substantial computational overhead, limiting their suitability for real-time deployment [5], a problem that becomes more pronounced with gradient boosting and related methods despite modest performance gains [6].

Deep learning has also attracted significant attention. Recurrent neural networks and LSTM architectures have been used to model sequential traffic patterns [7], while convolutional networks have been applied to automated feature extraction [8]. Although these models often outperform classical techniques, their training complexity and resource demands restrict their practicality in cloud microservices and edge settings [9].

More recent work has shifted toward lightweight intrusion detection models. Linear classifiers, including Logistic Regression and SGD-based methods, have

demonstrated competitive detection performance with substantially lower computational cost [10]. Shone et al. argued that simpler models frequently generalize more effectively in dynamic environments and are easier to deploy and maintain [11]. At the same time, Passive-Aggressive algorithms have been proposed for online detection due to their ability to adapt incrementally to streaming data [12].

A parallel development in IDS research is the recognition that accuracy alone provides an incomplete assessment, particularly for class-imbalanced datasets. Precision, recall, and F1-score are now widely recommended as core evaluation metrics [13], alongside explicit analyses of the performance–efficiency trade-off when designing IDS for cloud-native infrastructures [14, 15].

Despite this progress, systematic comparisons between lightweight IDS models and heavyweight ensemble baselines remain limited, especially on large-scale and realistic datasets that account for computational cost. This work addresses that gap. As summarized in Table 1, existing intrusion detection studies predominantly focus on improving detection accuracy using complex models, often overlooking computational efficiency and deployment feasibility in cloud microservices environments.

TABLE 1 Summary of Related Work on Intrusion Detection for Cloud and Network Environments

Ref .	Authors	Year	Dataset	Model(s) Used	Key Focus	Main Limitation
[1]	Mukkamala et al.	2005	KDD	SVM, Neural Networks	Early ML-based IDS	Limited scalability
[2]	Sabhnani & Serpen	2003	KDD	DT, SVM, kNN	Comparative ML study	High computational cost
[3]	Breiman	2001	—	Random Forest	Ensemble learning	Heavy training overhead
[4]	Zhang & Zulkernine	2006	KDD	Unsupervised outlier detection	Anomaly detection	Poor real-time performance
[5]	Buczak & Guven	2016	Multiple	RF, SVM, ANN	Survey of ML IDS	High model complexity
[6]	Chen & Guestrin	2016	—	XGBoost	Boosting methods	Computationally expensive
[7]	Yin et al.	2017	NSL-KDD	RNN	Deep learning IDS	Long training time
[8]	Kim et al.	2014	KDD	Hybrid IDS	Anomaly + misuse	Complex architecture
[9]	Tang et al.	2020	Custom	CNN	SDN-based IDS	Resource intensive
[10]	Ahmed et al.	2016	Multiple	LR, KNN	Lightweight ML IDS	Limited performance analysis
[11]	Shone et al.	2018	UNSW-NB15	Deep autoencoders	Feature learning	High complexity
[12]	Crammer	2000	—	Passive-	Online	Not evaluated

[13]	et al.	6		Aggressive	learning	on cloud data
[14]	Powers	2011	—	—	Evaluation metrics	No deployment analysis
[15]	Chawla et al.	2002	—	SMOTE	Class imbalance	Synthetic data bias
[16]	Conti et al.	2018	—	—	Cloud security survey	No IDS benchmarking

III. Dataset Description

A. CICIDS2017 Dataset

CICIDS2017 serves as a modern intrusion detection benchmark, replacing earlier datasets such as KDD99 and NSL-KDD that no longer reflect current attack behavior. CICIDS2017, instead, targets current protocols, realistic user behavior, and a broader range of attack strategies.

Traffic in CICIDS2017 was generated within a controlled testbed designed to approximate enterprise network operations. Both benign and malicious activities were produced using real network tools and attack frameworks, with traffic captured directly from live interactions. This approach yields data that bears a closer resemblance to operational network traffic than synthetic or heavily scripted alternatives. The dataset is provided as eight comma-separated value (CSV) files, each capturing traffic from a single working day. Individual files concentrate on different attack scenarios, enabling analysis of intrusion behavior across changing operational conditions.

The recorded traffic covers a broad range of attack types, alongside benign activity. The traffic includes brute-force authentication attempts, denial-of-service and distributed denial-of-service activity, port scanning, web-based exploits such as SQL injection and cross-site scripting, as well as infiltration scenarios intended to capture stealthy internal compromise.

Each network flow is represented by 78 numerical features derived from packet-level statistics generated with the CICFlowMeter tool. These features capture flow duration, packet and byte counts, inter-arrival timing, and protocol header information, providing sufficient resolution to distinguish subtle differences between benign and malicious behavior.

Each flow record is also assigned a class label identifying the traffic type. Although CICIDS2017 includes multiple attack labels, this study aggregates all attack categories into a single malicious class, reflecting a binary detection objective and aligning with deployment scenarios where the primary task is to separate benign traffic from potentially harmful activity.

CICIDS2017 exhibits pronounced class imbalance, with benign traffic accounting for the majority of flows. This class imbalance reflects conditions routinely encountered in operational networks and complicates evaluation, as strong aggregate accuracy can coincide with poor detection of low-frequency attack traffic. Precision, recall, and F1-score, therefore, provide a more meaningful basis for assessing IDS performance than accuracy alone.

On a holistic level, CICIDS2017 offers a large-scale benchmark grounded in realistic traffic behavior, with sufficient feature resolution to support rigorous intrusion detection studies. Its modern traffic characteristics, diverse attack coverage, and inherent class imbalance make it well-suited for evaluating lightweight machine learning models in cloud and microservices-based security settings.



Figure 1. Data pre-processing and integration pipeline for the CICIDS2017 dataset

B. Data Pre-processing and Integration

To ensure comprehensive coverage of all attack types, all eight CSV files were merged into a single dataset. As shown in Figure 1 the dataset underwent the following pre-processing steps:

- Removal of duplicate records
- Handling of missing and infinite values
- Standardization of feature names
- Feature normalization using standard scaling
- Binary label encoding (0 = benign, 1 = attack)

After pre-processing, the final dataset contained over 2.5 million records. The class distribution was approximately 83% benign and 17% attack traffic, reflecting a realistic class imbalance commonly observed in operational networks.

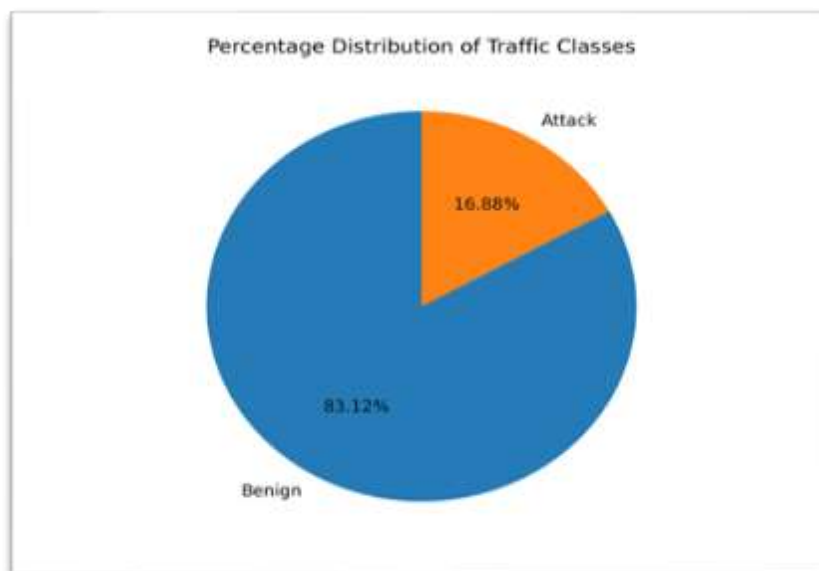


Figure 2 Percentage distribution of benign and malicious network flows in the CICIDS2017 dataset

IV. Methodology

This section presents an overview of the methodological framework adopted in this study, covering problem formulation, data preparation, model selection, mathematical foundations of the evaluated classifiers, and the experimental workflow. The methodology was designed to ensure reproducibility while emphasizing the suitability of lightweight machine learning models for intrusion detection in cloud microservices environments.

A. Problem Formulation

Intrusion detection in cloud environments is formulated as a binary supervised classification problem. Each network flow is represented by a feature vector, as follows:

$$\mathbf{x} = (x_1, x_2, \dots, x_d) \in \mathbb{R}^d$$

and is associated with a binary label, as follows:

$$y \in \{0,1\}$$

where ($y = 0$) denotes benign traffic and ($y = 1$) denotes malicious activity. The objective of IDS is to learn a decision function ($f(\mathbf{x})$) that accurately distinguishes between benign and attack traffic under realistic conditions characterized by large-scale data, high dimensionality, and class imbalance.

Unlike controlled laboratory datasets, cloud traffic is highly dynamic and continuously evolving. Therefore, the IDS must achieve not only high detection performance but also low computational overhead to support real-time detection and frequent retraining.

B. Data Preparation and Feature Processing

The CICIDS2017 dataset consists of multiple CSV files collected across different days, each representing distinct traffic patterns and attack scenarios. All files were merged into a unified dataset to ensure comprehensive coverage of benign and malicious behaviors.

B.1 Data Cleaning

Data pre-processing included the following steps:

- Removal of duplicate network flow records to prevent bias during training
- Elimination of samples containing missing or infinite values to ensure numerical stability
- Consolidation of multiple attack categories into a single malicious class to align with the binary classification objective

This pre-processing resulted in a clean and consistent dataset suitable for large-scale machine learning experiments.

B.2 Feature Scaling

Since the evaluated classifiers rely on gradient-based or margin-based optimization, feature scaling was applied using standardization, as follows:

$$x'_i = \frac{x_i - \mu_i}{\sigma_i}$$

where (μ_i) and (σ_i) denote the mean and standard deviation of the (i)-th feature, respectively. Feature standardization improves convergence speed and prevents features with large numeric ranges from dominating the learning process.

C. Model Selection Rationale

The models selected in this study were chosen based on their computational simplicity, scalability, and suitability for large-scale cloud environments. The following lightweight classifiers were evaluated in the current study:

- Stochastic Gradient Descent Classifier (SGDClassifier)
- Logistic Regression
- Passive-Aggressive Classifier
- Linear Support Vector Machine (Linear SVM)

A Random Forest model was included as a baseline to represent a high-performance but computationally expensive approach commonly reported in IDS literature. Deep learning and complex ensemble models were excluded from the primary analysis due to their high training cost, large memory footprint, and limited suitability for microservices deployment.

D. Mathematical Foundations of the Models

D.1 Logistic Regression

Logistic Regression estimates the posterior probability of a network flow being malicious using the sigmoid function, as follows:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

The probability model is defined as:

$$P(y = 1 | x) = \sigma(w^T x + b)$$

where (w) is the weight vector and (b) is the bias term. Model parameters are learned by minimizing the regularized log-loss function:

$$\mathcal{L} = - \sum_{i=1}^N [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)] + \lambda \|w\|_2^2$$

Logistic Regression offers a strong balance between detection performance, interpretability, and computational efficiency, making it well-suited for cloud-scale intrusion detection.

D.2 Stochastic Gradient Descent (SGDClassifier)

The SGDClassifier optimizes a linear objective using incremental updates based on individual samples or small mini-batches. The weight update rule is given as follows:

$$w_{t+1} = w_t - \eta_t (\nabla l(w_t; x_i, y_i) + \lambda w_t)$$

where (η_t) is the learning rate and ($l(\cdot)$) represents the loss function. This incremental learning strategy allows SGDClassifier to scale efficiently to millions of samples and makes it particularly suitable for streaming and continuously evolving cloud traffic.

D.3 Passive-Aggressive Classifier

The Passive-Aggressive algorithm is an online learning method that updates its parameters in the event of a misclassification. The optimization problem is defined as follows:

$$w_{t+1} = \arg \min_w \frac{1}{2} |w - w_t|^2$$

It is subject to the following constraint:

$$y_t(w^\top x_t) \geq 1$$

This formulation allows the model to remain unchanged for correct predictions (passive) while aggressively updating its parameters when errors are detected. This makes it suitable for dynamic intrusion detection scenarios.

D.4 Linear Support Vector Machine

Linear SVM aims to maximize the margin between benign and malicious traffic by minimizing the hinge loss, as follows:

$$\mathcal{L} = \sum_{i=1}^N \max(0, 1 - y_i(w^\top x_i)) + \lambda |w|^2$$

Although effective, Linear SVM typically requires a higher computational cost than logistic Regression and SGDClassifier, especially when trained on large-scale datasets.

D.5 Random Forest (Baseline)

Random Forest constructs an ensemble of decision trees, and the final prediction is obtained by averaging individual tree outputs, as follows:

$$\hat{y} = \frac{1}{T} \sum_{t=1}^T h_t(x)$$

where (T) denotes the number of trees and ($h_t(\cdot)$) represents the prediction of the (t)-th tree. Despite its strong detection performance, Random Forest incurs significant computational overhead and is therefore used only as a performance baseline.

E. Training and Evaluation Protocol

An 80/20 stratified split was applied to preserve class proportions, and all models were trained and tested on identical partitions. Performance was evaluated using standard classification metrics and training time. Given the strong class imbalance, the F1-score was prioritized, since high accuracy can be achieved even when detection of attack traffic is poor.

F. Methodological Justification for Lightweight Models

The methodological design reflects the operational constraints of cloud microservices environments. Lightweight models enable frequent retraining, low-latency inference, and efficient resource utilization, which are critical for real-time intrusion detection in dynamic cloud infrastructures. By prioritizing computational efficiency alongside detection performance, this study aligns IDS design with practical deployment requirements rather than purely offline accuracy optimization.

V. Experimental Setup

This section describes the experimental framework used to evaluate the proposed lightweight intrusion detection models. It details the dataset partitioning strategy, feature pre-processing procedures, model training configuration, evaluation metrics, computational cost measurement, and reproducibility considerations. The experimental design aims to ensure fairness, transparency, and reproducibility while emphasizing practical deployment constraints in cloud microservices environments.

A. Dataset Partitioning

After pre-processing and integration, the final CICIDS2017 dataset was divided into training and testing subsets on an 80/20 ratio. Stratified sampling was employed to preserve the original class distribution of benign and malicious traffic in both subsets. This strategy ensures that the evaluation reflects realistic network conditions characterized by class imbalance.

Let the final pre-processed dataset be denoted as:

$$D = \{(x_i, y_i)\}_{i=1}^N$$

where

$(x_i \in \mathbb{R}^d)$ represents the feature vector of the (i)-th network flow and $(y_i \in \{0,1\})$ denotes the class label, with 0 indicating benign traffic and 1 indicating malicious traffic.

The dataset was divided into training and testing subsets using a stratified 80/20 split:

$$D_{\text{train}} = 0.8D, \quad D_{\text{test}} = 0.2D$$

Stratification preserves the original class distribution in both subsets, such that:

$$P(y = 1 \mid D_{\text{train}}) \approx P(y = 1 \mid D_{\text{test}})$$

This strategy ensures that the evaluation reflects realistic class imbalance commonly observed in operational network environments.

B. Feature Pre-processing

Prior to model training, all numerical features were standardized using z-score normalization. Feature standardization was performed after dataset splitting to avoid data leakage. Feature normalization parameters were computed on the training data and applied unchanged to the test set. This step is necessary for gradient- and margin-based classifiers, including Logistic Regression, SGDClassifier, Passive-Aggressive, and Linear SVM, whose optimization behavior depends on feature scale. Standardization stabilizes training and improves convergence.

All features were standardized using z-score normalization to ensure numerical stability and improve optimization efficiency for gradient-based and margin-based classifiers. Each feature was transformed as follows:

$$x' = \frac{x - \mu}{\sigma}$$

where (μ) and (σ) denote the mean and standard deviation computed exclusively from the training set. The same parameters were subsequently applied to the test set to prevent data leakage.

C. Model Training Configuration

All models were implemented using the scikit-learn machine learning library. Wherever possible, default hyperparameters were retained to reflect realistic deployment scenarios and avoid overfitting through excessive tuning. This design choice emphasizes model robustness and generalizability rather than optimal offline performance.

Let $(f(x; \theta))$ represent a classification model parameterized by (θ) . Model training aims to identify optimal parameters (θ^*) by minimizing a loss function over the training data:

$$\theta^* = \arg \min_{\theta} \sum_{i=1}^{N_{\text{train}}} l\big(f(x_i), y_i\big)$$

Five machine learning models were evaluated:

- Logistic Regression
- Stochastic Gradient Descent (SGDClassifier)
- Passive-Aggressive Classifier
- Linear Support Vector Machine (Linear SVM)
- Random Forest (baseline)

Default hyperparameters were retained to reflect realistic deployment scenarios and to avoid overfitting through excessive tuning. Random Forest was included solely as a computationally intensive baseline for comparison.

D. Evaluation Metrics

Because CICIDS2017 is strongly class-imbalanced, model performance was evaluated using multiple metrics rather than accuracy alone:

- Accuracy, measuring overall classification correctness
- Precision, indicating the proportion of correctly identified attacks among all predicted attacks
- Recall, measuring the ability to detect malicious traffic
- F1-score was treated as the primary performance metric, reflecting the balance between precision and recall under imbalanced conditions.

Training time was also recorded for each model to capture differences in computational efficiency.

E. Computational Cost Measurement

Training time was measured as the wall-clock time required to fit each model on the training dataset. Prediction time was measured as the time required to classify all samples in the test dataset. These measurements help distinguish models that are viable for real-time and large-scale deployment from those that are not. All timing results were obtained under the same hardware and software configuration to ensure comparability. Each model was trained once using the same data partition to avoid variability introduced by repeated random sampling.

To evaluate deployment feasibility, computational efficiency was quantified using training and inference time.

Training time was measured as the wall-clock duration required to fit each model:

$$T_{\text{train}} = t_{\text{end}} - t_{\text{start}}$$

Inference time measures the total time required to classify all test samples:

$$T_{\text{pred}} = \sum_{i=1}^{N_{\text{test}}} t_i$$

Low inference latency is critical for real-time intrusion detection in cloud microservices to enable timely attack detection, avoid service latency violations, and support scalable real-time traffic analysis.

F. Experimental Environment

All experiments were run on a multi-core CPU workstation with sufficient memory to process the full dataset. GPU acceleration was not used, reflecting deployment conditions typical of cloud microservices, where inference is commonly CPU-bound. The experimental pipeline was implemented in Python using standard scientific computing libraries, with model training and evaluation performed through scikit-learn.

G. Reproducibility and Fair Comparison

Random seeds were fixed for dataset partitioning and model initialization where applicable, ensuring consistent experimental behavior. All pre-processing steps and model configurations followed a deterministic pipeline.

A single integrated dataset and identical pre-processing workflow were used across all experiments, so observed performance differences reflect model behavior rather than variation in data handling or experimental setup:

seed = 42

All models were trained and evaluated on the same datasets and underwent identical pre-processing and evaluation procedures. Performance differences, therefore, reflect the behavior of the learning algorithms rather than differences in experimental setup. The experimental setup emphasizes practical relevance over aggressive optimization. By prioritizing lightweight models, minimal tuning, and computational efficiency, the design aligns with the operational requirements of cloud microservices-based intrusion detection systems.

VI. Experimental Results and Analysis

This section presents a comprehensive evaluation of the proposed lightweight intrusion detection models based on the CICIDS2017 dataset. The results are analyzed from the perspectives of both detection performance and computational efficiency. To ensure a fair comparison, all models were trained and evaluated under identical experimental conditions, and underwent the same data partitions and pre-processing pipeline.

A. Overall Classification Performance

Table 2 summarizes performance metrics for all models. Random Forest achieves the highest accuracy and F1-score, exceeding 99%, but requires markedly longer training time than all other approaches.

Among linear models, Logistic Regression performs best overall, while Linear SVM and Passive-Aggressive methods show similar detection behavior at higher computational cost. SGDClassifier records the shortest training time with only a modest reduction in F1-score.

SGDClassifier stands out for efficiency. It records the shortest training and inference times while maintaining competitive detection performance. Although its F1-score falls marginally below that of Logistic Regression and Linear SVM, the reduction in computational cost makes it particularly attractive for large-scale and real-time deployment scenarios.

B. Accuracy and F1-Score Comparison

Figure 3 compares accuracy and F1-score across all evaluated models. A clear divergence between the two metrics appears for several classifiers, underscoring the effect of class imbalance in CICIDS2017. Although accuracy remains high in most cases, the F1-score offers a more informative view of attack detection performance.

Random Forest attains near-perfect values for both measures, while the lightweight models show a modest drop in F1-score. The difference between Random Forest and the strongest lightweight model, Logistic Regression, is small when set against the much larger gap in computational cost. This contrast highlights the practical advantage of lightweight approaches when performance must be weighed against efficiency.

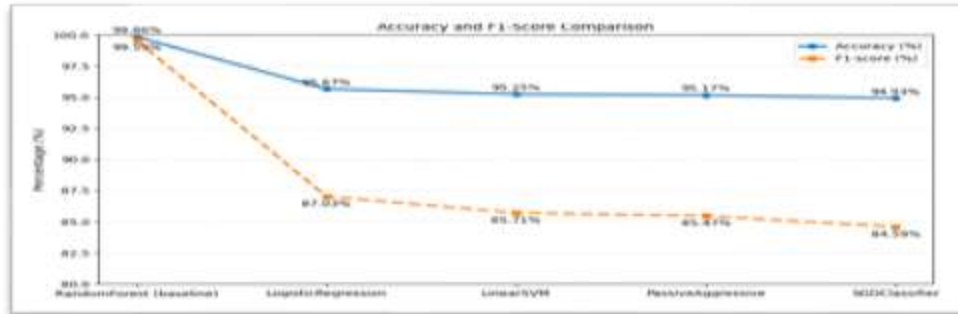


Figure 3. Comparison of accuracy and F1-score for all evaluated intrusion detection models

C. Confusion Matrix Analysis

Confusion matrices were examined for all classifiers to better characterize error behavior (Figure 4). The resulting patterns differ noticeably across models. Random Forest shows very low false positive and false negative counts in the confusion matrix, indicating consistent separation between benign and malicious traffic. Logistic Regression exhibits a similar balance, with comparatively few missed attack instances.

SGDClassifier and Passive-Aggressive classifiers produce a higher number of false negatives. This behavior coincides with their reduced computational footprint. False positive rates remain low for both models. Overall, the confusion matrices show that lightweight models introduce a modest increase in detection error relative to Random Forest, while maintaining substantially lower computational cost.

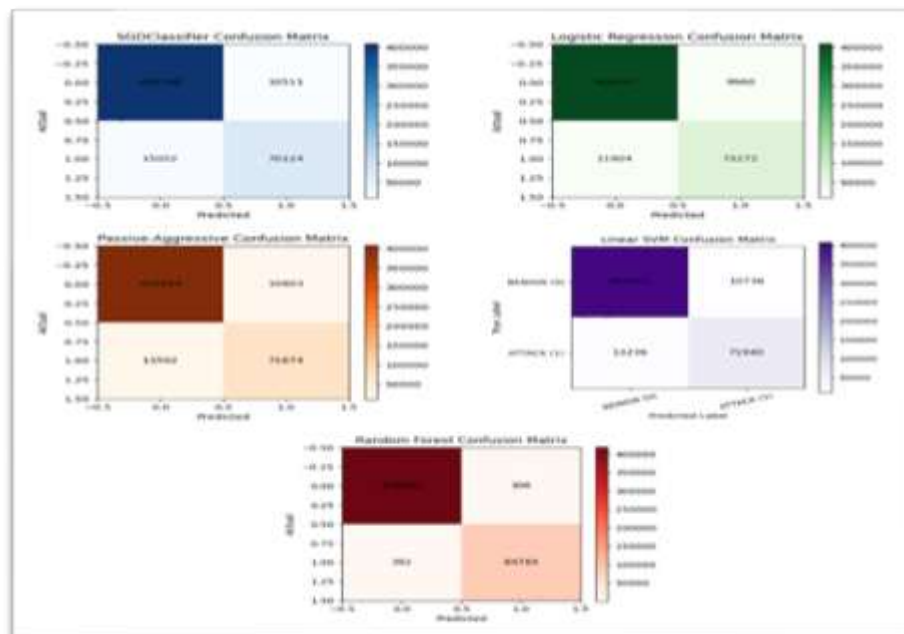


Figure 4. Confusion matrices for all evaluated intrusion detection models on the CICIDS2017 dataset, showing true positives, true negatives, false positives, and false negatives for benign (0) and attack (1) traffic under class-imbalanced conditions.

D. Computational Efficiency Analysis

Figure 5 compares training time across all evaluated models. A clear separation emerges between lightweight classifiers and the heavier ensemble approach. Random Forest records the longest training and inference times, consistent with the overhead associated with ensemble construction.

The linear models train substantially faster. SGDClassifier is the most efficient, with training times an order of magnitude lower than those of Logistic Regression and Linear SVM. Passive-Aggressive also shows low training cost, aligning with its design for incremental updates.

This difference matters in cloud microservices settings, where models may need to be retrained frequently as traffic patterns change and where computational resources are shared across services.

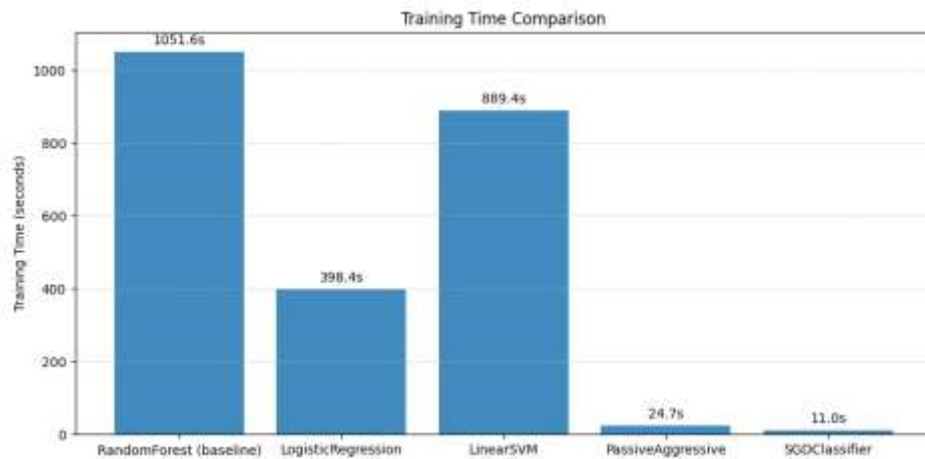


Figure 5. Training time comparison across all evaluated models.

E. Performance-Efficiency Trade-off

Figure 6 plots F1-score against training time. Random Forest appears in the upper-right region, with the highest F1-score and the longest training time.

The lightweight models cluster at substantially lower training times while maintaining moderately high F1-scores. Logistic Regression lies nearest the upper-left portion of this region, reflecting comparatively strong performance at lower computational cost. SGDClassifier lies at the extreme low end of training time, achieving acceptable detection performance with minimal computational overhead.

Viewed together, the distribution of points shows that improvements beyond the lightweight models yield diminishing returns. Gains in detection performance from heavier models are accompanied by a disproportionate increase in computational cost, underscoring the practical motivation for lightweight intrusion detection in cloud environments.

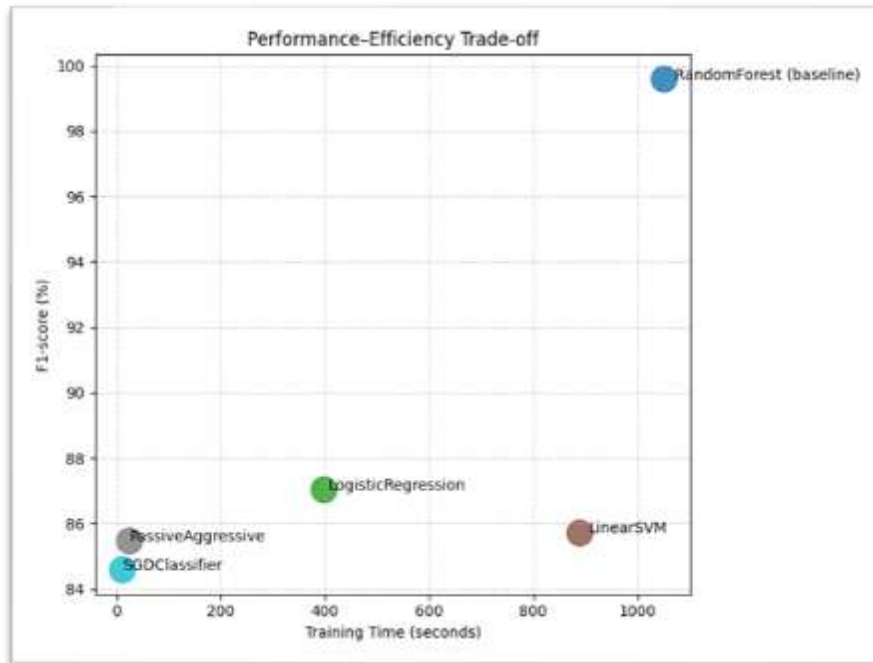


Figure 6. Trade-off between detection performance (F1-score) and training time for all evaluated models

F. Implications for Cloud Microservices Deployment

Accuracy on its own is a poor proxy for suitability in cloud microservices deployments. Ensemble models score highly, but the associated computational cost limits where they can be used.

Lightweight approaches such as Logistic Regression and SGDClassifier avoid this constraint. Their performance is lower, but the reduction in training and inference cost makes them easier to integrate into distributed and containerized environments.

G. Summary of Experimental Findings

The key findings from the experimental evaluation can be summarized as follows:

- Random Forest achieves the highest detection performance but incurs a prohibitive computational cost.
- Logistic Regression offers the best balance between accuracy and F1-score among lightweight models.
- SGDClassifier provides exceptional computational efficiency with competitive detection performance.
- Lightweight models significantly reduce training and inference time while maintaining acceptable detection effectiveness.

These results support the hypothesis that lightweight machine learning models represent a practical and scalable solution for intrusion detection in cloud microservices environments.

VII. Discussion

Detection accuracy alone provides a limited basis for evaluating intrusion detection systems in cloud microservices. Random Forest attains the highest accuracy (99.86%) and F1-score, but this performance is coupled with substantially higher training and

inference costs. These costs constrain use in cloud-native settings where latency budgets are fixed, and computational resources are shared.

The lightweight linear models show a different set of trade-offs. Logistic Regression, Linear SVM, Passive-Aggressive, and SGDClassifier all achieve competitive detection performance with significantly lower computational cost. Logistic Regression performs best within this group, reaching approximately 95.67% accuracy with a balanced precision-recall profile. Compared with Random Forest, the lightweight models exhibit higher false negative rates while maintaining consistently low false positive rates. This pattern is apparent in the confusion matrices and is characteristic of class-imbalanced datasets such as CICIDS2017, where aggregate accuracy provides limited information about minority-class detection.

Inference latency further distinguishes the models. Microservices environments generate large volumes of inter-service traffic and operate under tight service-level constraints, leaving little tolerance for additional processing overhead. The lightweight models achieve substantially lower classification latency, enabling near real-time operation without introducing bottlenecks. SGDClassifier and Passive-Aggressive models, in particular, support incremental updates, allowing adaptation to evolving traffic patterns that are difficult to sustain with ensemble-based methods.

Scalability also emerges as a practical concern. Random Forest shows limited scalability as data volume increases, making horizontal scaling expensive. In contrast, the linear models scale more predictably with both sample size and feature dimensionality, supporting deployment across distributed cloud nodes, containers, and edge components. Their low resource footprint simplifies integration into cloud security pipelines that rely on decentralized monitoring rather than centralized analysis.

Taken together, the results indicate that the most suitable intrusion detection model for cloud microservices is not the one that maximizes accuracy in isolation, but the one that balances detection capability with efficiency and scalability. Random Forest provides a useful reference point for achievable performance. Still, lightweight models such as Logistic Regression and SGDClassifier offer a more practical basis for deployment in real-world cloud environments. Table 2 summarizes these trade-offs, highlighting the relationship between detection performance, computational cost, and deployment feasibility.

Table 2. Summary of Key Insights from Experimental Evaluation

Aspect	Key Insight	Evidence from Results	Practical Implication
Detection Accuracy	Highest accuracy does not guarantee deployability	Random Forest achieved 99.86% accuracy, but required the highest training and inference time	Accuracy alone is insufficient for cloud microservices IDS
Lightweight Model Performance	Lightweight models achieved competitive accuracy	Logistic Regression achieved 95.67% accuracy; all lightweight models exceeded 94%	Lightweight models provide strong detection at a lower cost
Computational Efficiency	Heavy models incur prohibitive	Random Forest training time was orders of magnitude higher than	High computational cost limits real-

	computational overhead	linear models	time deployment
Class Imbalance Handling	F1-score provides a more reliable evaluation than accuracy	Lightweight models maintained stable F1-scores under imbalance	F1-score is essential for IDS evaluation
False Positive Control	Lightweight models maintain low false positive rates	Confusion matrices show limited benign traffic misclassification	Reduces alert fatigue in operational environments
Inference Latency	Low inference time is critical in microservices	SGDClassifier and Passive-Aggressive showed the fastest inference	Enables real-time intrusion detection
Scalability	Linear models scale better with large datasets	Training time grows linearly for lightweight models	Suitable for large-scale cloud environments
Adaptability	Incremental learning improves responsiveness	SGD and Passive-Aggressive support online updates	Enables rapid adaptation to new attacks
Deployment Feasibility	Simpler models are easier to deploy	Lightweight models require fewer resources	Ideal for containerized and edge deployments
Overall Trade-off	Optimal IDS balances performance and efficiency	Logistic Regression offers the best performance-efficiency trade-off	Lightweight models are preferable for real-world cloud IDS

VIII. Conclusion and Future Work

Lightweight machine learning models were evaluated for intrusion detection in cloud microservices using the CICIDS2017 dataset. Four linear classifiers—Logistic Regression, SGDClassifier, Passive-Aggressive, and Linear SVM—were compared with a Random Forest baseline. Random Forest achieved the highest detection accuracy, but this result was accompanied by substantially higher computational cost, limiting its use in real-time and large-scale cloud deployments.

The lightweight models followed a different performance profile. All achieved competitive detection results with markedly lower training and inference costs. Logistic Regression performed best within this group, reaching approximately 95.67% accuracy with a balanced precision-recall profile. The remaining linear models also exceeded 94% accuracy, indicating that effective intrusion detection does not require high-complexity ensemble methods when deployment constraints are taken into account.

The analysis also shows that accuracy alone is insufficient for evaluating intrusion detection systems on class-imbalanced data. Metrics such as F1-score and confusion matrix analysis provide clearer insight into detection behavior. The

lightweight models maintained low false positive rates with manageable false negative levels, supporting their use in continuous monitoring scenarios where alert volume must be controlled.

Future work will extend this analysis in three directions. Online and incremental learning methods will be examined to support continuous adaptation to evolving attack patterns. Hybrid approaches that combine lightweight models with selective deep learning components will be explored to improve detection capability without incurring excessive overhead. Finally, the proposed methods will be evaluated on additional cloud-focused datasets and tested in containerized environments under live traffic conditions.

Funding Statement: "This research received no external funding"

Conflicts of Interest: "The authors declare no conflict of interest."

References

- [1]. Mukkamala, S., Sung, A. H., & Abraham, A., Intrusion detection using ensemble of soft computing paradigms. *Journal of Network and Computer Applications*, 28(2), 167–182, 2005.
- [2]. Sabhnani, M. & Serpen, G., Application of machine learning algorithms to KDD intrusion detection dataset. *Proc. IEEE Int. Conf. Machine Learning and Applications*, Los Angeles, CA, USA, 209–215, 2003.
- [3]. Breiman, L., Random forests. *Machine Learning*, 45(1), 5–32, 2001.
- [4]. Zhang, J. & Zulkernine, M., Anomaly based network intrusion detection with unsupervised outlier detection. *Proc. IEEE Int. Conf. Communications*, Istanbul, Turkey, 2388–2393, 2006.
- [5]. Buczak, A. L. & Guven, E., A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2), 1153–1176, 2016.
- [6]. Chen, T. & Guestrin, C., XGBoost: A scalable tree boosting system. *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 785–794, 2016.
- [7]. Yin, C., Zhu, Y., Fei, J., & He, X., A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access*, 5, 21954–21961, 2017.
- [8]. Kim, G., Lee, S., & Kim, S., A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Systems with Applications*, 41(4), 1690–1700, 2014.
- [9]. Tang, T., Mhamdi, Z., McKeown, D., & Krishnamachari, B., Deep learning approach for network intrusion detection in software defined networking. *IEEE Wireless Communications*, 27(4), 90–97, 2020.
- [10]. Ahmed, M., Mahmood, A. N., & Hu, J., A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60, 19–31, 2016.
- [11]. Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q., A deep learning approach to network intrusion detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(1), 41–50, 2018.
- [12]. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y., Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7, 551–585, 2006.
- [13]. Powers, D. M. W., Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, 2(1), 37–63, 2011.

- [14]. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P., SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321–357, 2002.
- [15]. Conti, M., De Cristofaro, E., Prasad, R. R., & Crispo, B., A survey on security and privacy issues of cloud computing. *IEEE Communications Surveys & Tutorials*, 20(1), 107–131, 2018.