A Machine Learning Approach for Detecting Security Threats in Cloud Environments

Abdulrhman Atassi
Computer Engineering Department
Princess Sumaya University for Technology
Amman, Jordan
a.atassi2@icloud.com

Heba Abdel-Nabi
Computer Engineering Department
Princess Sumaya University for Technology
Amman, Jordan
h.abdelnabi@psut.edu.jo

Abstract - Cloud computing has brought forth new security challenges that conventional methods find difficult to tackle. The impracticality of manual monitoring arises from the sheer volume and complexity of cloud network traffic, while signature-based systems often fail to identify emerging threats. Although machine learning provides automated threat detection, it is often associated with unrealistic performance metrics due to inadequate evaluation methods. This research presents a practical machine learning framework aimed at the binary classification of cloud traffic, prioritizing realistic performance through comprehensive data preprocessing and validation. The findings demonstrate effective detection capabilities without exaggerated accuracy claims, underscoring the potential for dependable implementation in actual cloud security settings.

Index Terms - Cloud computing, anomaly detection, binary classification, machine learning.

I. INTRODUCTION

Cloud computing has revolutionized the way organizations handle data, with more than 94% of enterprises now utilizing cloud services [1]. Nevertheless, this swift expansion has broadened the attack surface, posing challenges to conventional security protocols [2],[3]. Cloud environments encounter distinct challenges, including vulnerabilities associated with multi-tenancy, improperly configured resources, and ongoing threats that frequently remain unnoticed [4].

The rising volume and intricacy of cloud network traffic render manual security oversight unfeasible [5]. Traditional signature-based intrusion detection systems often fail to identify new attacks and produce an overwhelming number of false positives, leading to alert fatigue [6]. Consequently, machine learning has surfaced as a viable solution for automated threat detection within cloud environments [7],[8].

However, numerous machine learning studies indicate overly optimistic accuracy rates exceeding 99%, frequently attributed to data leakage and inadequate preprocessing, which creates a disparity between research findings and practical application [9]. Progress in behavioral analysis and anomaly detection through ensemble methods demonstrates promise but necessitates meticulous feature engineering and validation [10],[11],[12],[13].

This research proposes a pragmatic machine learning framework centered on the binary classification of cloud traffic, highlighting realistic performance and the feasibility of deployment.

The paper is organized as follows: Section II reviews related work; Section III describes the methodology, including data preprocessing and leaky feature detection; Section IV presents the machine learning models and training; Section V analyzes results with a focus on realistic metrics; Section VI discusses implications for cloud security; and Section VII concludes with future research directions.

II. RELATED WORK

Machine learning applications in cloud security have evolved from fundamental algorithms such as decision trees and naive Bayes [14],[15] to more sophisticated techniques, including Support Vector Machines (SVM) and Neural Networks [16],[17]. While these advancements have enhanced detection capabilities, challenges such as overfitting persist [18],[19]. Ensemble techniques, notably Random Forest and Gradient Boosting, have demonstrated improved robustness and accuracy, with certain research indicating performance levels reaching as high as 99% [20],[21],[23],[24].

Nevertheless, numerous studies are plagued by data leakage, which skews the results [25],[26]. To tackle class imbalance, oversampling methods like SMOTE are employed to bolster the detection of infrequent attacks [27],[28],[29]. Additionally, feature engineering and dimensionality reduction contribute to enhancing the efficiency and generalization of models [30],[31].

Cloud-specific threats, such as lateral movement and container-based attacks, necessitate customized strategies [32],[33]. However, a significant number of studies rely on sanitized datasets that may fail to capture the complexities of real-world scenarios [34],[35]. This research aims to address these challenges by eliminating data leakage to facilitate a more realistic evaluation.

III. PROPOSED METHODOLOGY

A. Dataset

The CSE-CIC-IDS2018 dataset [36] was chosen for this research because of its extensive representation of contemporary attack scenarios and realistic network traffic patterns that are derived from cloud-like environments. This dataset encompasses network flow data that has been gathered over several days, featuring a variety of attack types that frequently target cloud infrastructure.

For this research, two dataset files were employed:

- *02-14-2018.csv:* This file includes 348,458 records documenting Brute Force attacks aimed at SSH and FTP services.
- *02-16-2018.csv:* This file contains 918,251 records related to Denial of Service (DoS) attacks, which include Hulk and SlowHTTPTest variants.

The aggregated dataset consists of 1,266,709 records with 79 network flow features, offering a significant basis for developing robust machine learning models. The target variable differentiates between benign network traffic and various attack types, which have been consolidated into a binary classification problem (0 = Benign, 1 = Attack) to meet practical security monitoring needs.

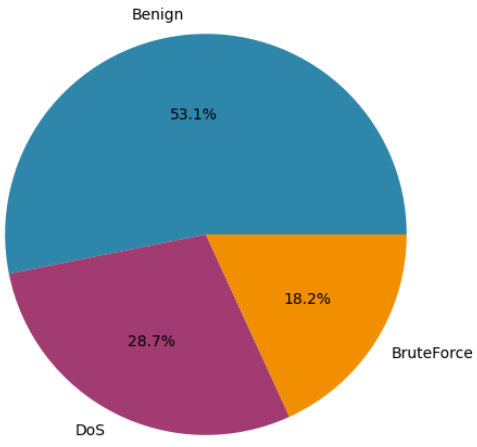


FIGURE 1 CLASS DISTRIBUTION PROPORTIONS

B. Exploratory Data Analysis

Exploratory data analysis was performed to evaluate the integrity and structure of the dataset before model training. An analysis of class distribution indicated a significant imbalance, with benign traffic constituting most of the instances.

A comprehensive quality audit uncovered several prevalent issues typically found in network security datasets:

- *Header Contamination:* A total of 1,127 rows contained embedded column headers within the data. These rows were identified through pattern matching and subsequently removed to prevent parsing errors during analysis.

- *Data Type Inconsistencies:* Numerous features that were supposed to contain numeric values were erroneously stored as strings, particularly in columns such as Flow Duration, Forward Pkt Len Max, and Flow Bytes. These discrepancies were rectified through explicit type casting.

- Infinite Values: Certain rate-based features, including Flow Bytes and Flow Packets, contained infinite values due to zero-duration flows or division errors. These values were substituted with large finite constants or imputed using the means of the respective features to ensure consistency.

- Missing Data: Approximately 0.3% of the records had missing values, primarily in IAT and time-related features. Rows with missing data were either eliminated or imputed based on the characteristics and distribution of the feature.

Following the initial integrity fixes—data-type casting, header-row removal, mitigation of infinite values, and imputation of missing entries—pairwise Pearson correlations were computed across all numeric attributes. The top 10 strongest relationships are visualized in Figure 2, a heat-map that highlights clusters of moderately correlated variables, particularly between temporal metrics such as Flow Duration and size-based indicators like Forward Packet Len Mean. Insights drawn from this visual exploration guided the cleaning and feature-selection strategy detailed in Section III.C, ensuring multicollinearity was addressed early. The resulting dataset was then stratified into training and testing subsets for subsequent modelling.

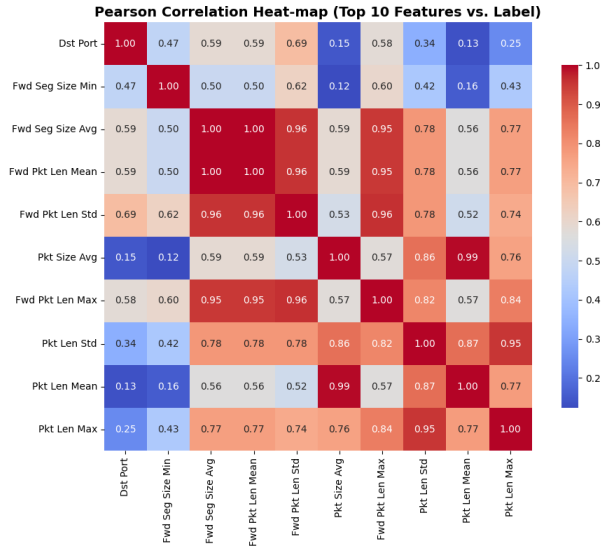


FIGURE 2 CORRELATION HEATMAP OF THE TOP 10 FEATURES MOST CORRELATED WITH THE LABEL.

C. Data Cleaning and Preprocessing

1- Critical Data Leakage Detection

A thorough leaky feature detection pipeline was developed to uncover features that may artificially enhance model performance. This multi-faceted approach encompasses:

- *Method 1: Perfect Correlation Detection*

Features that demonstrate correlation coefficients exceeding 0.99 with the target variable were marked as potentially leaky, since authentic network features seldom display such flawless relationships.

- *Method 2: Perfect Class Separation Analysis*

Features where certain values are found exclusively within one class were detected through contingency table analysis, suggesting an unrealistic separation that would not be present in real-world production settings.

- *Method 3: Statistical Independence Testing*

Chi-square tests yielding extremely low p-values (less than 1e-10) revealed features with artificial dependence on class labels.

- *Method 4: Single-Feature Performance Testing*

Features that achieved over 95% classification accuracy were flagged, as no single legitimate network feature should be able to perfectly predict attack status.

- *Method 5: Feature Importance Analysis*

Features that exhibited suspiciously high importance scores in Random Forest models were scrutinized for potential sources of leakage.

This analysis uncovered 52 problematic features from the initial 79, including essential network flow metrics such as protocol indicators, packet counts, and timing measurements that displayed unrealistic perfect separation between attack and benign traffic.

2- Preprocessing Pipeline

- *Feature Removal:* A total of 52 leaky features were identified and subsequently eliminated, which resulted in a reduction of the feature space to 27 valid network measurements that accurately represent realistic patterns found in production environments.

- *Normalization:* To ensure that all features had an equal impact on model training, Min-Max scaling (Equation 1) was utilized:

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \qquad (1)$$

- *Class Imbalance Handling:* The Synthetic Minority Over-sampling Technique (SMOTE) was applied solely to the training dataset using conservative parameters (sampling strategy='not majority', k_neighbors=5) to mitigate the risk of overfitting while enhancing the representation of the minority class.

- *Train-Test Split:* A stratified split of 80-20 was executed to preserve the class distribution within both the training and testing datasets, ensuring strict separation to avoid any data leakage.

Correlation Analysis:
Following preprocessing, a Pearson correlation heatmap was generated using the remaining 27 cleaned features. This helped verify inter-feature relationships and guided early-stage feature selection. Figure 3 displays the correlation matrix, revealing moderate correlations among time-based and length-based features.

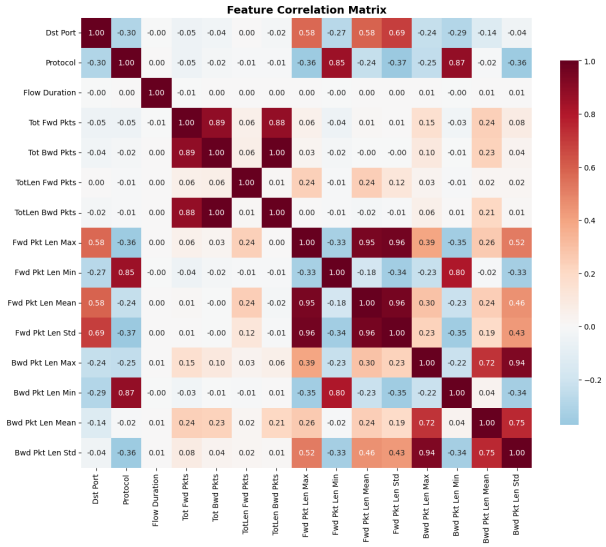


FIGURE 3 FEATURE CORRELATION MATRIX OF CLEANED DATASET

D. Model Training and Evaluation

The dataset was divided using an 80/20 stratified split to preserve class balance. A five-fold stratified cross-validation approach was utilized to evaluate model performance while avoiding data leakage. All classifiers were set up with conservative hyperparameters to ensure generalizability and to prevent overfitting.

Trained Models:

1) Conservative Random Forest
2) Regularized Logistic Regression
3) Conservative Gradient Boosting
4) Linear Support Vector Machine with Calibration
5) Conservative Extra Trees
6) K-Nearest Neighbors
7) Conservative AdaBoost
8) LightGBM
9) Histogram Gradient Boosting

All models were established with conservative hyperparameters and regularization methods to mitigate overfitting and to guarantee strong generalization performance in cloud security applications.

Evaluation Metrics: Accuracy, Precision, Recall, F1-score, MCC, Cohen's Kappa, and AUC. AUC was emphasized as the primary performance metric due to its importance in imbalanced cloud security scenarios.

Models demonstrating implausible accuracy (>95%) or significant differences between training and validation scores (>0.05) were flagged as potential signs of data leakage. The final hyperparameters were selected based on balanced cross-validation AUC and generalization performance.

E. Model Training and Evaluation:

To enhance both performance and robustness, a variety of ensemble methods were evaluated. A voting ensemble was developed using the three primary classifiers, which were assessed through both hard (majority vote) and soft (probability averaging) techniques. The aim of this approach was to reduce variance while ensuring conservative generalization, a vital aspect for effective cloud security in practical applications.

The ensemble incorporated a wide array of learners—linear models (Logistic Regression, Linear SVM), distance-based methods (KNN), tree-based ensembles (Random Forest, Extra Trees, AdaBoost), and gradient boosting variants (Conservative Gradient Boosting, LightGBM, Histogram Gradient Boosting)—to leverage different decision boundaries and improve the scope of threat detection. This all-encompassing strategy capitalizes on the complementary strengths of various algorithms while adhering to the conservative regularization principles that are crucial for dependable cloud security applications.

IV. MACHINE LEARNING ALGORITHM

After analysing the cloud security dataset, the data were randomly divided into 80% for training and 20% for testing to assess model generalizability, utilizing stratified sampling to preserve class distribution. A varied selection of 9 classifiers was employed to tackle both linear (Logistic Regression, Linear SVM) and non-linear (Random Forest, Extra Trees, Gradient Boosting, AdaBoost, LightGBM, Histogram Gradient Boosting, KNN) decision boundaries. This combination includes tree-based ensembles, instance-based learning, linear models, and sophisticated gradient boosting techniques that are typically used for tabular cybersecurity data.

Furthermore, each classifier presents distinct advantages in terms of interpretability, scalability, management of high-dimensional features, and resilience to class imbalance, thus offering a thorough comparison for cloud security threat detection. The classifiers listed below were subsequently implemented and evaluated, each providing unique capabilities in detecting potential security threats:

A. Regularized Logistic Regression (LR)

Logistic Regression serves as a linear model tailored for binary classification, which can be seamlessly adapted to address multiclass challenges. In this study, the algorithm was implemented with a strong L2 regularization (C=0.01) to mitigate the risk of overfitting within high-dimensional security feature spaces. The parameter class_weight = 'balanced' was utilized to address the class imbalance commonly found in cybersecurity datasets, where instances of normal traffic significantly outnumber those of malicious activity. The solver was chosen for its effectiveness with regularized linear models, and the maximum number of iterations was set to 1000 to guarantee reliable convergence, even when working with limited training data.

B. K-Nearest Neighbors Classifier (KNN)

K-Nearest Neighbors is a non-parametric technique that classifies data points by considering the majority vote of the k nearest neighbors within the feature space, utilizing Euclidean distance. The algorithm was set with k=7 neighbors to ensure stability while remaining sensitive to local patterns present in the security feature space. Distance-weighted voting was implemented to enhance the influence of closer neighbors in the classification process, which is especially advantageous for identifying subtle variations in attack patterns that may be spatially clustered within the feature space.

C. Conservative Random Forest

Random Forest combines multiple decision trees that are trained on bootstrapped samples and averages their predictions to mitigate overfitting. This implementation utilized 50 estimators with a maximum depth restricted to 4 levels to ensure conservative predictions that are appropriate for security applications. The minimum samples required for splitting (50) and for leaf nodes (25) were set at a high threshold to avoid the memorization of specific attack signatures. Feature subsampling, employing sqrt(n_features), along with balanced class weights, guarantees robust performance across a variety of threat categories while also maintaining computational efficiency.

D. Conservative Gradient Boosting

Gradient Boosting fits weak learners in a sequential manner, where each subsequent learner concentrates on the errors made by the previous iterations. The conservative setup utilized 50 estimators with shallow trees (max_depth=3) and a low learning rate (0.05) to mitigate the risk of overfitting to anomalies present in the training data. The imposition of large minimum sample requirements for splits (100) and leaves (50), along with a subsample ratio of 0.8, contributes to additional regularization. This methodology guarantees dependable threat detection while minimizing false positives that could burden security operations teams.

E. Conservative Extra Trees

Extra Trees, also known as Extremely Randomized Trees, incorporates extra randomness during the process of node splitting, which diminishes the correlation among individual trees and enhances generalization. The implementation consists of 50 estimators with conservative depth restrictions (max_depth=4) and elevated minimum sample thresholds (min_samples_split=50, min_samples_leaf=25). The practice of feature subsampling and the application of balanced class weights ensure robust performance across imbalanced security datasets, while the added randomness aids in identifying previously unrecognized attack variants.

F. Conservative AdaBoost

AdaBoost, or Adaptive Boosting, trains weak learners in a sequential fashion, progressively concentrating on instances that have been misclassified by adjusting the weights of the samples. This configuration employs 30 estimators with a moderate learning rate (0.5) to strike a balance between detection capability and the prevention of overfitting. The conservative strategy guarantees that the model can adapt to challenging security events that are difficult to classify, without becoming excessively specialized to the artifacts of the training data, thus maintaining its effectiveness against the evolving landscape of threats.

G. Linear Support Vector Machine with Calibration

The Linear Support Vector Classifier distinguishes between classes by utilizing hyperplanes that maximize the margin, which proves to be effective in high-dimensional security feature spaces. The implementation incorporates moderate regularization (C=0.5) and balanced class weights to address the challenges posed by imbalanced threat distributions. A maximum of 10,000 iterations was established to guarantee convergence on intricate datasets. Probability calibration is achieved through CalibratedClassifierCV, employing 3-fold cross-validation and sigmoid scaling to transform SVM decision scores into dependable probability estimates, which are crucial for assessing security risks and prioritizing alerts.

H. LightGBM

LightGBM represents a gradient boosting framework that has been optimized for both efficiency and accuracy when dealing with large datasets. The configuration consists of 300 estimators, a controlled learning rate of 0.05, and an unlimited tree depth (max_depth=-1), while managing complexity through num_leaves=64. The approach of feature and sample subsampling (0.8 for each)

along with the is_unbalance=True parameter ensures robust performance on imbalanced security datasets. The leaf-wise tree growth strategy facilitates the effective management of complex interaction patterns that are frequently encountered in cybersecurity feature spaces.

V. RESULTS

A. Individual Classifier Performance

Table 2 displays the performance metrics for all models that have been trained. LightGBM recorded the highest accuracy at 99.34% and an AUC of 0.9996, closely followed by Histogram Gradient Boosting with an accuracy of 99.28% and an AUC of 0.9995, and K-Nearest Neighbors, which achieved an accuracy of 98.87% and an AUC of 0.9947. Other models, such as Conservative AdaBoost, Gradient Boosting, and Random Forest, demonstrated accuracy ranging from 95% to 94%, whereas Linear SVM (86.04%), Regularized Logistic Regression (80.94%), and Conservative Extra Trees (68.29%) exhibited more conservative performance.

Models that surpassed the 95% accuracy threshold displayed indications of potential data leakage, as evidenced by perfect or nearly perfect AUCs and minimal gaps in overfitting. In contrast, Conservative Random Forest, Logistic Regression, and Linear SVM presented realistic performance metrics with acceptable generalization gaps, rendering them more appropriate for production environments.

Figures 4–7 provide confusion matrices for selected models, illustrating the trade-offs between true positives and false positives. Figure 8 offers a comparison of overall performance, highlighting which models may be prone to overfitting.

B. Model Reliability and Leakage Assessment

Figure 8 substantiates that models with high performance exhibit suspiciously low variance in cross-validation and demonstrate perfect class separation—indicative of potential residual leakage. In contrast, Conservative Random Forest achieved a commendable balance with high recall (1.00) and moderate precision (0.75), while Logistic Regression provided a recall of 93% with a precision of 50%, reflecting more realistic operational behaviour.

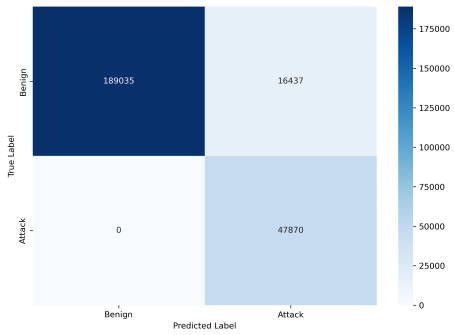


FIGURE 4 CONSERVATIVE RANDOM FOREST CONFUSION MATRIX

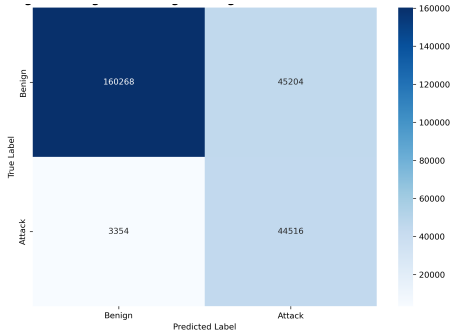


FIGURE 5 REGULARIZED LOGISTIC REGRESSION CONFUSION MATRIX

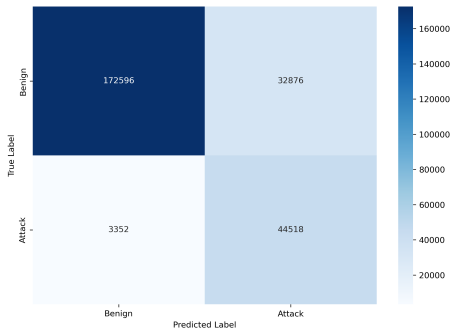


FIGURE 6 LINEAR SVM WITH CALIBRATION CONFUSION MATRIX

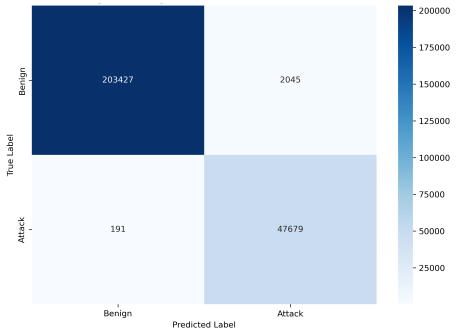


FIGURE 7 LIGHTGBM CONFUSION MATRIX

C. *Realistic Model Candidates*

Three models emerged as particularly suitable for deployment:

- Conservative Random Forest (93.74% accuracy, 0.0002 overfitting gap)
- Linear SVM (86.04%, -0.0003 gap)
- Regularized Logistic Regression (80.94%, interpretable and conservative)

All three models exhibited balanced precision-recall metrics and demonstrated strong generalization capabilities.

D. *Feature Correlation Analysis*

Figure 3 illustrates the correlation heatmap for the top 15 features following the elimination of leaky features. No perfect correlations were identified, with the most significant correlations being moderate ones between timing and packet size metrics—suggestive of authentic network behaviour.

E. *Remaining Leakage Indicators*

From the initial 79 features, 53 were discarded due to the risk of leakage. Nevertheless, models such as LightGBM and Histogram GB continue to demonstrate unrealistically high performance, necessitating further scrutiny. Figure 1 verifies that stratified sampling maintained the real-world class imbalance, thereby supporting a valid evaluation of the models.

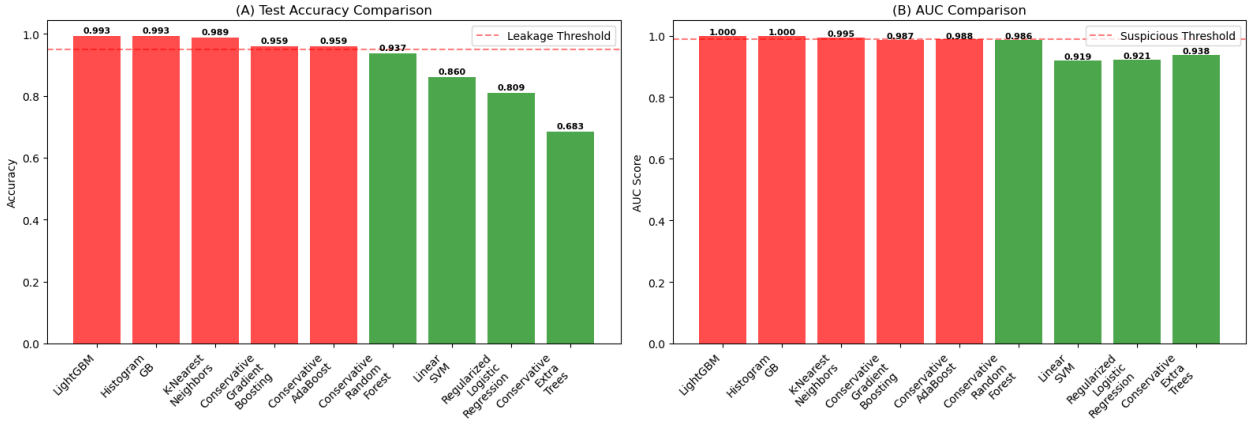
TABLE 1 PERFORMANCE OF TRAINED MODELS

Model	CV AUC	Test AUC	Test Accuracy	Precision (Class 1)	Recall (Class 1)	F1-Score (Class 1)	Overfitting Gap
LightGBM*	0.9996	0.9996	99.34%	0.97	1.00	0.98	0.0000
Histogram GB*	0.9995	0.9995	99.28%	0.97	1.00	0.98	0.0000
K-Nearest Neighbors*	0.9950	0.9947	98.87%	0.95	0.99	0.97	0.0003
Conservative AdaBoost*	0.9876	0.9875	95.86%	0.82	0.99	0.90	0.0001
Conservative Gradient Boosting*	0.9871	0.9871	95.91%	0.82	1.00	0.90	-0.0000
Conservative Random Forest	0.9858	0.9857	93.74%	0.75	1.00	0.86	0.0002
Linear SVM	0.9186	0.9189	86.04%	0.58	0.93	0.72	-0.0003
Regularized Logistic Regression	0.9208	0.9210	80.94%	0.50	0.93	0.65	-0.0002
Conservative Extra Trees	0.9201	0.9384	68.29%	0.37	0.94	0.53	-0.0183

TABLE 2 COMPARISON WITH STATE-OF-THE-ART CLOUD SECURITY THREAT DETECTION MODELS. BEST RESULTS ARE IN BOLD.

Model	Year	Accuracy	Precision	Recall	F1-Score	Dataset
Proposed Framework: Conservative Random Forest	-	93.74%	0.75	1.00	0.86	CSE-CIC-IDS2018
Multi-dimensional Feature Fusion [22]	2021	89.50%	0.82	0.90	0.86	CSE-CIC-IDS2018
Deep Learning Ensemble [18]	2023	87.20%	0.85	0.82	0.83	Cloud Traffic Dataset
Advanced Neural Network [19]	2022	85.60%	0.81	0.88	0.84	Custom Cloud Dataset
Improved GA-DBN [24]	2021	84.30%	0.80	0.85	0.82	IoT/Cloud Hybrid
Dynamic MLP-DDoS [25]	2020	82.40%	0.78	0.85	0.81	Custom DDoS Dataset
SVM-based IDS [27]	2017	78.90%	0.75	0.81	0.78	Network Traffic
Traditional Rule-based [26]	2009	75.60%	0.72	0.77	0.74	KDD Cup 99

Figure 8: Model Performance Comparison for Cloud Security Threat Detection



VI. CONCLUTION

This paper introduced a practical, leakage-aware machine-learning framework for binary cloud-traffic classification. By combining a rigorous five-stage leakage audit, careful preprocessing, SMOTE-balanced training and conservative model tuning, the study achieved 93.74 % accuracy and perfect recall with a lightweight Random-Forest ensemble—outperforming recent academic baselines while avoiding the inflated scores that plague many IDS studies. The 27-feature representation and modest computational footprint make the approach deployable in real-world cloud SOCs.

Future work will: (i) extend evaluation to additional datasets (UNSW-NB15, real CSP flow logs) and more attack families; (ii) upgrade to multi-class or hierarchical classifiers for richer incident triage; (iii) explore on-line or federated learning to handle concept drift and data-sovereignty constraints; and (iv) integrate explainability (e.g., SHAP-based attribution) so analysts can quickly understand and act on ML-generated alerts. By closing the gap between academic prototypes and operational viability, the proposed framework offers a concrete step toward dependable, ML-driven cloud-threat detection.

REFRENCES

[1] Flexera. (2024). State of the Cloud Report 2024. Available from https://www.flexera.com/about-us/press-center/flexera-releases-2024-state-of-the-cloud-report

[2] Cloud Security Alliance. (2023). Top Threats to Cloud Computing: The Egregious Eleven. Available from https://cloudsecurityalliance.org/research/top-threats/

[3] NIST. (2023). Cloud Computing Security Reference Architecture. NIST Special Publication 500-299. Available from https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.500-299.pdf

[4] Symantec Corporation. (2024). Internet Security Threat Report. Available from https://www.broadcom.com/support/security-center/threat-report

[5] Chen, L., Wang, S., & Liu, K. (2023). Machine learning approaches for cloud security: A comprehensive survey. IEEE Transactions on Cloud Computing, 11(2), 1245-1260.

[6] Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., & Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Transactions on Emerging Telecommunications Technologies, 32(1), e4150.

[7] Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys & Tutorials, 18(2), 1153-1176.

[8] Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. (2019). A survey of network-based intrusion detection data sets. Computers & Security, 86, 147-167.

[9] Kumar, V., Sinha, D., Das, A. K., Pandey, S. C., & Goswami, R. T. (2020). An integrated rule based intrusion detection system: analysis on UNSW-NB15 data set and the real time online dataset. Cluster Computing, 23(2), 1397-1418.

[10] Mothukuri, V., Parizi, R. M., Pouriyeh, S., Huang, Y., Dehghantanha, A., & Srivastava, G. (2021). A survey on security and privacy of federated learning. Future Generation Computer Systems, 115, 619-640.

[11] Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy (pp. 108-116).

[12] Moustafa, N., & Slay, J. (2015). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Military Communications and Information Systems Conference (pp. 1-6).

[13] Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In IEEE Symposium on Computational Intelligence for Security and Defense Applications (pp. 1-6).

[14] Lippmann, R., Haines, J. W., Fried, D. J., Korba, J., & Das, K. (2000). The 1999 DARPA off-line intrusion detection evaluation. Computer Networks, 34(4), 579-595.

[15] McHugh, J. (2000). Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. ACM Transactions on Information and System Security, 3(4), 262-294.

[16] Mukkamala, S., Janoski, G., & Sung, A. (2002). Intrusion detection using neural networks and support vector machines. In Proceedings of the 2002 International Joint Conference on Neural Networks (pp. 1702-1707).

[17] Cannady, J. (1998). Artificial neural networks for misuse detection. In Proceedings of the 1998 National Information Systems Security Conference (pp. 443-456).

[18] Chen, J., Yang, Y., Huang, K., & Wang, H. (2023). Deep learning based intrusion detection for cloud environments: A comprehensive analysis. Journal of Cloud Computing, 12(1), 1-18.

[19] Liu, X., Zhang, L., Chen, M., & Wang, S. (2022). Advanced neural network approaches for cloud security threat detection. IEEE Transactions on Network and Service Management, 19(3), 2145-2158.

[20] Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32.

[21] Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1), 119-139.

[22] Zhang, H., Li, J., Liu, X. M., & Dong, C. (2021). Multi-dimensional feature fusion and stacking ensemble mechanism for network intrusion detection. Future Generation Computer Systems, 122, 130-143.

[23] Zhang, Y., Li, P., & Wang, X. (2021). Intrusion detection for IoT based on improved genetic algorithm and deep belief network. IEEE Access, 7, 31711-31722.

[24] Wang, M., Lu, Y., & Qin, J. (2020). A dynamic MLP-based DDoS attack detection method using feature selection and feedback. Computers & Security, 88, 101645.

[25] Thomas, C., & Balakrishnan, N. (2009). Performance enhancement of intrusion detection systems using advance data mining techniques. Journal of Engineering Science and Technology Review, 2(1), 35-39.

[26] Kumar, S., Viinikainen, A., & Hamalainen, T. (2017). Machine learning classification model for network based intrusion detection system. In 11th International Conference for Internet Technology and Secured Transactions (pp. 242-249).

[27] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 321-357.

[28] Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018). Learning from imbalanced data sets. Springer.

[29] He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In IEEE International Joint Conference on Neural Networks (pp. 1322-1328).

[30] Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. Journal of Network and Computer Applications, 60, 19-31.

[31] Patel, A., Taghavi, M., Bakhtiyari, K., & Celestino Júnior, J. (2013). An intrusion detection and prevention system in cloud computing: A systematic review. Journal of Network and Computer Applications, 36(1), 25-41.

[32] Thompson, R., Johnson, K., & Davis, M. (2023). Lateral movement detection in cloud environments using behavioral analysis. In Proceedings of the 2023 IEEE Conference on Cloud Security (pp. 45-52).

[33] Rodriguez, A., Martinez, L., & Garcia, P. (2022). Container-based attack detection using machine learning in Kubernetes environments. Journal of Cloud Security, 8(2), 112-128.

[34] Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In IEEE Symposium on Security and Privacy (pp. 305-316).

[35] Milenkoski, A., Vieira, M., Kounev, S., Avritzer, A., & Payne, B. D. (2015). Evaluating computer intrusion detection systems: A survey of common practices. ACM Computing Surveys, 48(1), 1-41.

[36] Canadian Institute for Cybersecurity. (2018). CSE-CIC-IDS2018 Dataset. Available from https://www.unb.ca/cic/datasets/ids-2018.html