INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078
(Volume 11, Issue 5 - V11I5-1145)
Available online at: https://www.ijariit.com

Intrusion Detection in AWS Cloud Environments Using Machine Learning on Network Flow Data

Oduwunmi Odukoya
odukoyao@etsu.edu
East Tennessee State University, Tennessee

Mariam Adetoun Sanusi
Sanusiadetoun@gmail.com
University of Texas, Dallas, Texas

Samuel Adenekan
adenekanp@etsu.edu
East Tennessee State University, Tennessee

ABSTRACT

AWS Cloud Environments support core workloads and services, but are exposed to malicious actions and unauthorized activities in the transmission of network flow data. The threats subject cloud infrastructures to different types of attacks, thus Intrusion Detection in AWS Cloud Environments ensures privacy, reliability, and availability. This study explores the use of Machine Learning for intrusion detection by analyzing traffic patterns in cloud systems. The CSE-CIC-IDS2018 dataset, containing realistic benign and attack traffic, was employed for model training and evaluation. After comprehensive preprocessing and analysis, five Machine Learning algorithms were implemented: Random Forest, Decision Tree, Ridge Classifier, Logistic Regression, and Linear Support Vector Classifier. Their performance was measured using accuracy, precision, recall, F1 score, ROC-AUC, and detection time. Results showed that Random Forest and Decision Tree achieved the highest accuracy at 100%, with the Decision Tree demonstrating superior efficiency by classifying all instances in 0.056 seconds. Ridge Classifier followed with an accuracy of 99.2%, while Logistic Regression achieved 98.8%. The Linear Support Vector Classifier recorded the lowest performance with 96.2% accuracy. This research confirms the effectiveness of Machine Learning for cloud security. The Decision Tree Classifier, combining flawless accuracy with the fastest detection speed, emerges as the most practical model for real-time intrusion detection in AWS environments.

Keywords: *Cloud, IDS, FlowData, Machine Learning, AWS, Intrusion and Detection, Cyber-Attack.*

1. INTRODUCTION

AWS Cloud Environments have transformed computing in the contemporary era by offering scalability, flexibility, and affordability to companies around the globe. Organizations are migrating workloads to AWS for guaranteeing service reliability, minimizing infrastructure expenses, and permitting access on a global scale [27]. The strengths of AWS infrastructures notwithstanding, they add new attack surfaces that are now under attack by attackers. Attackers commonly attack misconfigured security groups, access keys, and network traffic in order to achieve unauthorized control or steal confidential information [34]. With AWS used for mission-critical missions in finance, healthcare, and government, having effective security controls is paramount. Network flow data plays a double role by facilitating seamless resource allocation while simultaneously providing a platform on which malware traffic patterns spread [18].

Intrusion Detection within AWS Cloud Environments is thus crucial to provide secure operation resilience. Network flow traffic is susceptible to various attacks including denial of service, brute force, botnet spreading, and abuse internally [12]. These attacks are especially hard to identify in AWS because of elasticity and dynamic scaling, which enable attacks to merge with normal workload spikes [30]. Multi-tenancy further introduces additional complexity to the setup, and intrusion detection systems capable of distinguishing between normal and abnormal traffic during high loads are needed. Anomalies go undetected if early detection does not occur, and perpetrators can be given extended access, leading to greater potential for service downtime and data breach [23]. As AWS is that crucial in today's digital spaces, the upgrading of the detection systems is not an option but a requirement.

Traditional techniques of intrusion detection have failed to meet these cloud-based challenges. Signature-based IDS solutions can detect known patterns of attacks but struggle when faced with zero-day attacks or new variants [16]. There needs to be continuous upgrading in order to remain up-to-date, an exercise unfeasible in the realm of AWS. Anomaly-based solutions, although able to detect new attacks, tend to overwhelm administrators with false positives [35]. Such inefficiencies undermine alert trust and exacerbate incident response. Furthermore, most legacy IDS lack computational scalability to examine realistically AWS traffic volumes. These shortcomings underline the need for adaptive approaches that can handle diverse, high-throughput traffic while minimizing errors. Machine Learning (ML) has emerged as a solution capable of overcoming these challenges through predictive and data-driven methods [8].

ML techniques leverage historical network data to learn distinguishing features of benign and malicious traffic.

Unlike static approaches, they generalize patterns and identify both known and previously unseen attacks [21]. For AWS traffic analysis, models such as Random Forest, Decision Tree, Ridge Regression, Logistic Regression, and Linear Support Vector Classifiers are widely applied. These algorithms were chosen because they represent complementary strengths across interpretability, scalability, and classification performance [37]. Tree-based models capture non-linear relationships effectively, while linear methods offer computational speed. Margin-based classifiers provide robust separation for high-dimensional data. Evaluating a diverse set of models enables a balanced understanding of trade-offs in performance and practicality when deployed in AWS environments [11].

Model evaluation requires comprehensive metrics beyond accuracy alone. Precision and recall measure how well models capture true attacks while minimizing false positives [28]. The F1 score provides a balance between these metrics, while ROC-AUC offers a threshold-independent view of classifier performance [36]. Confusion matrices reveal detailed misclassification patterns, identifying where models favor benign or malicious traffic incorrectly. Detection time is equally critical in AWS, where real-time response determines whether damage is contained. Together, these metrics provide a rigorous framework for assessing intrusion detection systems. The significance of this research lies in demonstrating how machine learning enables scalable, adaptive, and efficient security measures tailored for AWS cloud infrastructures [23].

2. LITERATURE REVIEW

Research in intrusion detection for cloud environments has increasingly shifted toward machine learning models capable of analyzing large volumes of network flow data. Traditional approaches relying on static signatures or manual rule sets have proven inadequate against polymorphic and unknown attacks, prompting a wave of experimental studies that explore modern architectures, benchmark datasets, and real-world deployment challenges [32].

Al-Ghuwairi et al. developed a time-series anomaly detection approach based on Long Short-Term Memory (LSTM) networks trained on cloud network flows. Their method achieved 96.5 % accuracy, 95.2 % recall, and 94.7 % precision, showing significant improvements over traditional anomaly detection techniques. The strength of this model lies in its ability to capture sequential dependencies in network flow data, which are especially important for cloud traffic where attack sequences evolve over time. However, the authors highlight limitations in handling encrypted traffic, as critical flow features were obscured, leading to reduced sensitivity against stealthy attacks. Furthermore, the requirement of large labeled datasets for training posed a scalability issue, as cloud traffic is often unlabeled and dynamically changing [2].

Manocchio et al. proposed FlowTransformer, a transformer-based intrusion detection framework that applies self-attention mechanisms to flow sequences. The model achieved an ROC-AUC of 98.7 % and a precision of 97.4 % on benchmark datasets, outperforming traditional recurrent neural networks. Its advantage lies in learning long-range dependencies across flow records, which enables it to detect sophisticated multi-stage intrusions. Yet, the study reported substantial computational overhead, with training times and memory usage significantly exceeding other baselines. Such complexity limits deployment in real-time AWS environments where efficiency is crucial, unless additional optimizations such as pruning or lightweight transformer variants are adopted [6].

Long et al. introduced another transformer-centric detection approach, emphasizing the representation of flow features as tokens to be processed by multi-head attention layers. Their system produced a detection rate of 94.3 %, F1 score of 93.8 %, and precision of 92.9 %, highlighting its ability to generalize across multiple attack types. A notable strength of their method is its capacity to capture both temporal and contextual relationships among flows, which is critical in cloud infrastructures with diverse traffic patterns. However, the authors noted that preprocessing flow data into the required tokenized format introduced added complexity and sensitivity to feature selection. Additionally, the performance of the model decreased noticeably on obfuscated or encrypted flows, limiting its applicability in secure cloud networks where traffic visibility is restricted [3].

Kao et al. also suggested a two-stage deep learning approach with a convolutional encoder as a feature compression phase and a dense neural classifier as an anomaly detection phase. The technique recorded 95.1 % precision, 94.8 % recall, and an F1 score of 95.0 %, which is similar to the performance of one-stage deep learning techniques but with lesser computations. The utilization of staged processing permitted the model to discover succinct yet descriptive representations of flow data, thus facilitating quick classification with little lost accuracy. Nonetheless, the authors reported challenges with class imbalance in the training data, where rare attack classes were misclassified as benign. They also highlighted that hyperparameter sensitivity introduced instability, requiring extensive tuning to maintain performance across different datasets, which hinders scalability to diverse AWS workloads [22].

Li et al. advanced the field with FlowGANAnomaly, a generative adversarial network (GAN)–based approach designed to improve anomaly detection by generating realistic benign flows. The model had 93.5 % accuracy and F1 score of 92.2 %, superior to conventional supervised classifiers in the detection of low-frequency or weak attacks. The main advantage of the approach is its applicability to the utilization of synthetic data for balancing training sets and consequently improving classifier robustness to rare attack classes. However, the GAN model exhibited stability issues, including mode collapse, which degraded performance when trained on highly heterogeneous traffic patterns. Moreover, the training process was computationally expensive, making real-time deployment in large AWS environments impractical without further architectural optimization [34].

Göcs et al. conducted a feature relevance study on the CSE-CIC-IDS2018 dataset, applying Random Forest models to assess the predictive value of individual flow features. They showed that limiting training to the top 20% most valuable features retained 96.8% accuracy, almost as high as when the full feature set is used. This paper is meaningful because it provides pragmatic suggestions for dimensionality reduction for large-scale deployments of IDS without sacrificing accuracy. But the authors warned that feature relevance can differ widely across workloads and tenants and that best practices in feature selection strategies in one setting might not apply to others. This finding underscores the importance of adaptive feature selection mechanisms in real-world AWS scenarios [16].

Zavrak et al. applied multivariate time-series anomaly detection methods to flow-based IDS in cloud and software-defined network environments, using recurrent neural networks to model temporal dependencies. Their model achieved 93.2 % recall and 92.5 % precision, demonstrating effectiveness in detecting attack sequences that unfold across multiple flows. The ability to capture time-dependent attack patterns gave the system an advantage over static classifiers. Nevertheless, scalability emerged as a key challenge, as the model struggled to handle high-throughput streaming data typical of large cloud deployments. The study also noted susceptibility to noisy telemetry data, which reduced performance when training on flows with incomplete or corrupted attributes [24].

In summary, collectively, these seven papers demonstrate a variety of GAN and transformer, feature selection, and recurrent network methods, performance generally between 92 % and 99 % on accuracy, precision, recall, F1 score, and ROC-AUC. With major potential for flow-based cloud intrusion detection despite some common challenges, i.e., heavy computational burden, low generalizability across workloads, instability of the generative model, and poor support for encrypted flows. These limitations provide strong justification for research focused on lightweight, adaptive, and high-performing intrusion detection models in AWS contexts [32].

3. METHODOLOGY

The study design for the present work was formulated to facilitate a methodical intervention of intrusion detection within AWS cloud computing environments based on the CSE-CIC-IDS2018 dataset. The process followed five key stages: data preprocessing, to clean and preprocess the dataset for reliability; exploratory data analysis (EDA), to discover patterns and relationships from network flow data; data transformation, to convert categorical labels to numerical binary values that can be fed to machine learning; model building, where five supervised learning algorithms were used—Random Forest, Decision Tree, Ridge Regression, Logistic Regression, and Linear Support Vector Classifier; and lastly, model evaluation, based on metrics like accuracy, precision, recall, F1-score, ROC-AUC, confusion matrix, and detection time. This systematized pipeline guarantees the results to be reproducible, accurate and can be employed in actual AWS intrusion detection systems.

3.1 About Dataset

This research utilizes the CSE-CIC-IDS2018 dataset, made available on Kaggle at IDS 2018 Intrusion CSV. The dataset was created by the Canadian Institute for Cybersecurity (CIC) and the Communications Security Establishment (CSE) to build a representative benchmark for cloud-like intrusion detection research. It was collected in a controlled network topology emulating AWS-style cloud offerings, both obtaining legitimate traffic and a range of malicious activity.

The original data set contained 80 columns and 1,048,575 rows. One row corresponds to an instance of a network flow, and one column corresponds to an instance of a specific flow-level feature. The features include protocols, byte counts, inter-arrival times, packet size statistics, and flow duration. Most notably, the dataset has a Label column that labels each example as either Benign or one of numerous types of attack categories, including brute force (FTP/SSH), denial-of-service (DoS), distributed denial-of-service (DDoS), infiltration, botnet, and web attacks, including SQL injection and cross-site scripting (XSS).

The dataset was boiled down to a binary classification problem for this research, with a focus on discriminating between Benign traffic and Bot traffic. In order to make it machine learning usable, the Label column was converted into a numeric Target variable where Benign instances were given 0 and Bot instances were given 1. This conversion allowed an even dataset structure that makes the development and testing of machine learning models possible for efficient intrusion detection within AWS cloud environments.

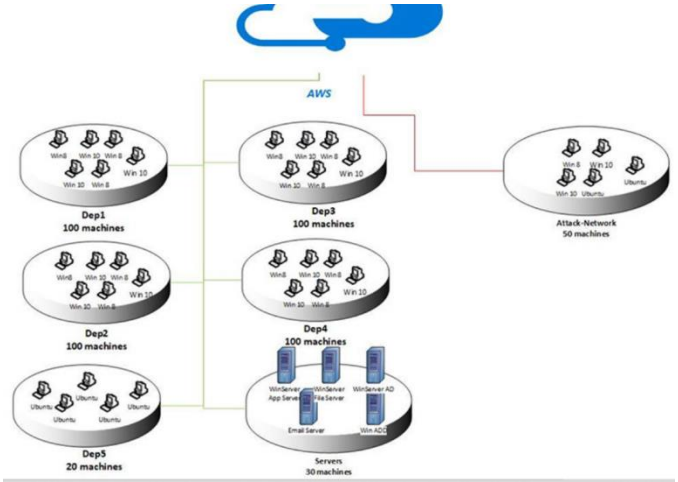


Figure 1: Network Topology (Iman et al., 2018)

3.2 Data Preprocessing

Data preprocessing was carried out to get the dataset ready for the rest of the data analysis and training of the machine learning models. The process made the dataset error-free, standardized, and in an appropriate format to train machine learning models.

Removing unused and redundant columns that did not contain any meaningful information for intrusion detection was the initial step in the data cleaning process. Removing these unnecessary features not only minimized noise in the data but also minimized dimensionality, thus minimizing the complexity of the feature space. Minimizing dimensionality minimized the size of the data set, improved computational speed during training, and enabled the machine learning algorithms to concentrate on the most relevant features, hence improving performance and interpretability.

The second was the management of missing values and infinity values, whose presence can diminish model performance greatly if eliminated. Upon examination, the dataset was seen to have a few such instances. Rows with NaN (Not a Number) and Infinity values were then dropped to avoid introducing bias and maintain data integrity.

Further, two of the columns in the data had negative values. As these columns had characteristics depending on count, negative values were semantically inaccurate. The values in these columns were thus converted to their absolute values so that the data still remained meaningful and within its intended interpretation.

Also, the data set was checked for single-valued columns, i.e., columns where all the records share the same value. They are variability-free columns and hence lack discriminatory capability for classification problems. Ten single-valued columns were identified and later removed from the data set in an attempt to boost efficiency, eliminate noise, and concentrate on informative features.

Lastly, the data set was also checked for duplicate records, which can potentially degrade data quality by artificially boosting some patterns and skewing the analysis with bias.

The screening revealed 235,979 duplicate records (excluding their first occurrence). All duplicate records were eliminated entirely, and only the first occurrence of each duplicate was saved to ensure that there were exclusive data points only.

These hybrid cleaning processes resulted in a high-quality, well-formatted dataset that reduced noise and redundancy to the fullest extent possible and increased the relevance and reliability of the dataset for AWS cloud environment intrusion detection.

3.3 Exploratory Data Analysis

This research utilized the application of the CSE-CIC-IDS2018 dataset, which has been openly shared on Kaggle through the following link: https://www.kaggle.com/datasets/solarmainframe/ids-intrusion-csv. The initial dataset had 1,048,575 rows and 80 columns and covered a broad spectrum of features derived from network traffic flows. Nonetheless, after utilizing the data cleaning processes, the dataset was sanitized to 808,546 rows and 43 columns. This sanitization process minimized the dataset to only the critical attributes required for efficient model training. Except for the Label column, where there were categorical values for the class of network traffic, the other features were numerical with integer or float values.

Table 1 presents a complete overview of the cleaned dataset with data type, number of unique values, and value range per column.

Table 1: Overview of Dataset

S/N	Variable	Data Type	Count of Unique Values	Range of Values
1.	Dst Port	Integer	1230	0 … 65534
2.	Protocol	Integer	5	0 … 17
3.	Tot Fwd Pkts	Integer	5691	1 … 43159
4.	Tot Bwd Pkts	Integer	109523	0 … 69241
5.	TotLen Fwd Pkts	Integer	2	0 … 1100627
6.	TotLen Bwd Pkts	Float	2	0 … 101000000
7.	Fwd Pkt Len Mean	Float	2	0 … 1460
8.	Bwd Pkt Len Mean	Float	29443	0 … 1459.62
9.	Flow Byts/s	Float	458103	0 … 480000000
10.	Flow Pkts/s	Float	455754	0.016669 … 3000000
11.	Flow IAT Mean	Float	400219	0.5 … 120000000
12.	Fwd IAT Tot	Float	280941	0 … 120000000
13.	Fwd IAT Mean	Float	351690	0 … 120000000
14.	Bwd IAT Tot	Float	251488	0 … 120000000
15.	Bwd IAT Mean	Float	316792	0 … 120000000
16.	Fwd PSH Flags	Integer	2	0 … 1
17.	Fwd Header Len	Integer	1419	0 … 2275036
18.	Bwd Header Len	Integer	1976	0 … 1384832
19.	Fwd Pkts/s	Float	453047	0.008567 … 3000000
20.	Bwd Pkts/s	Float	381426	0 … 2000000
21.	Pkt Len Mean	Float	43396	0 … 1454.275
22.	FIN Flag Cnt	Integer	2	0 … 1
23.	SYN Flag Cnt	Integer	2	0 … 1
24.	RST Flag Cnt	Integer	2	0 … 1
25.	PSH Flag Cnt	Integer	2	0 … 1
26.	ACK Flag Cnt	Integer	2	0 … 1
27.	URG Flag Cnt	Integer	2	0 … 1
28.	ECE Flag Cnt	Integer	2	0 … 1
29.	Down/Up Ratio	Integer	72	0 … 148
30.	Pkt Size Avg	Float	43072	0 … 1687.5
31.	Fwd Seg Size Avg	Float	20085	0 … 1460
32.	Bwd Seg Size Avg	Float	29443	0 … 1459.62
33.	Subflow Fwd Pkts	Integer	754	1 … 43159
34.	Subflow Fwd Byts	Integer	6266	0 … 1100627
35.	Subflow Bwd Pkts	Integer	1064	0 … 69241
36.	Subflow Bwd Byts	Integer	18068	0 … 100999700
37.	Init Fwd Win Byts	Integer	3799	0 … 65535
38.	Init Bwd Win Byts	Integer	4725	0 … 65535
39.	Fwd Act Data Pkts	Integer	278	0 … 9262
40.	Fwd Seg Size Min	Integer	9	0 … 44
41.	Active Mean	Float	72650	0 … 111000000
42.	Idle Mean	Float	32540	0 … 120000000
43	Label	Categorical	2	‘Benign’ & ‘Bot’

Figure 2 shows the distribution of the top three destination ports for Benign as well as Bot traffic.

Most of the Benign traffic streams were directed to port 3389, or the Remote Desktop Protocol (RDP), which is widely utilized for remote control in cloud computing. There was a high volume also seen on port 53, the Domain Name System (DNS), and port 80, which is the Hypertext Transfer Protocol (HTTP) traffic. The flows are typical of a legitimate use in the cloud in which remote management, DNS resolution, and web access are the usual behaviors.

Conversely, Bot traffic utilized a different distribution pattern. Most malicious flows targeted port 8080, an average HTTP substitute, and one of the main targets of command-and-control communication from botnet attacks. Moreover, smaller volumes of Bot flows targeted non-standard ports like port 0 and port 50891.

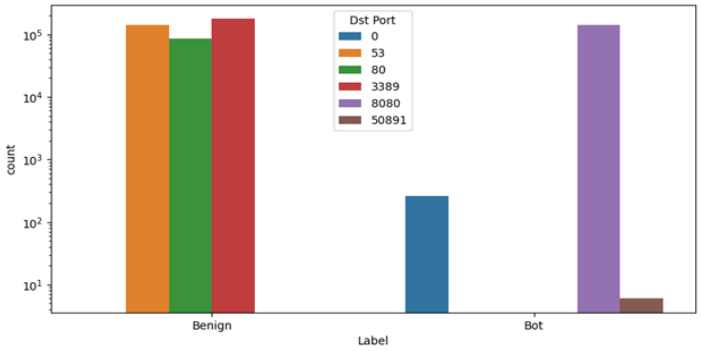


Figure 2: Distribution of Top Three Destination Ports for Benign and Bot Traffic

Figure 3 presents the distribution of protocol types for Benign and Bot traffic. Across both categories, Protocol 6 is the most frequently used protocol. In both categories, Protocol 6 (TCP) appeared most frequently, consistent with its widespread dominance in cloud traffic, where guaranteed, connection-oriented traffic is paramount. Reserved Protocol 0, typically an invalid protocol in normal networking, also appeared in both Benign and Bot flows, consistent with the presence of anomalous traffic.

Notably, Protocol 17 (UDP) was observed exclusively in Benign traffic and did not appear in Bot flows, providing a potential distinguishing feature for intrusion detection.

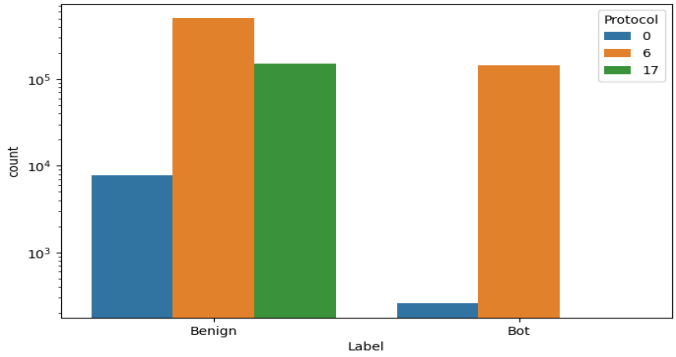


Figure 3: Distribution of Protocol Types for Benign and Bot Traffic

Figure 4 illustrates the mean number of forward and backward packets for Benign and Bot traffic. For both types, the mean number of forward packets is greater than the mean number of backward packets.

In addition, Benign traffic has considerably larger values for the forward and backward packets than Bot traffic and represents more normal, stable, and balanced communication patterns. These distinctions highlight the potential of packet-level features as indicative descriptors for machine learning solutions to distinguish benign traffic from possible intrusions.

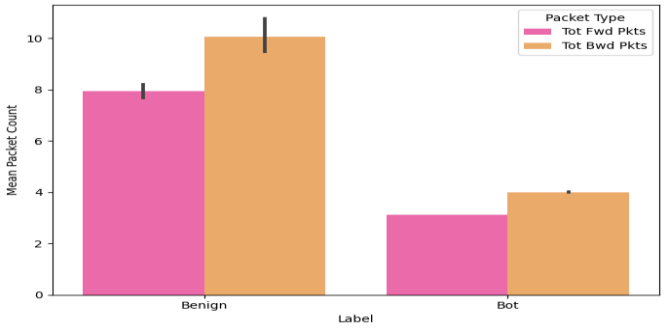


Figure 4: Average Forward and Backward Packets for Benign and Bot Traffic

Figure 5 demonstrates the Benign and Bot traffic mean packet size. Benign traffic has a significantly larger average packet size, close to double that of Bot traffic. This demonstrates benign cloud behavior as more packet data-centric and bot-originated traffic has smaller packets with a tendency of being produced by bot or scripted attack streams.

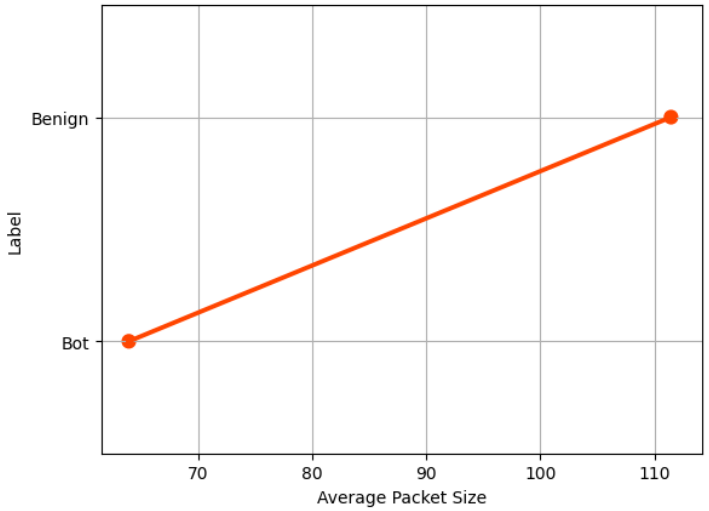


Figure 5: Average Packet Size for Benign and Bot Traffic

Figure 6 illustrates the Bot and Benign network traffic classes' flow bytes per second distribution. Bot traffic contains a much higher median Flow Bytes/s than Benign traffic. This indicates Bot-related flows tend to flow at higher speeds, perhaps because they capture automated attack activity like scanning, flooding, or large data exfiltration, while Benign traffic contains lower, more constant throughput characteristic of typical cloud operation.

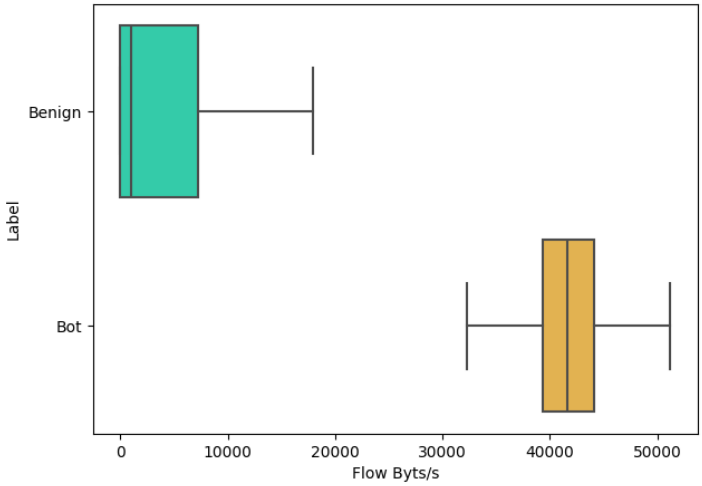


Figure 6: Distribution of Flow Bytes per Second for Benign and Bot Traffic.

Figure 7 displays the mean flow inter-arrival time (Flow IAT) between Benign and Bot traffic. The figure shows that Benign traffic has a significantly larger Flow IAT compared to Bot traffic. This indicates that normal cloud communications are more temporally dispersed, as would be expected in typical user-initiated traffic. Bot traffic, however, possesses much smaller inter-arrival times, which means quick, automated transmission of packets typical of malicious behavior such as scanning, flooding, or synchronized attacks.

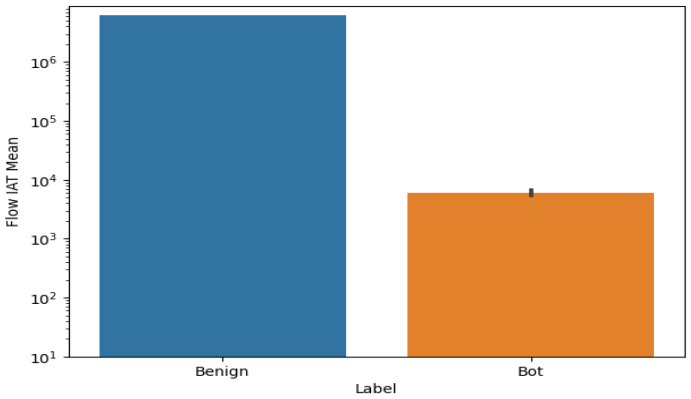


Figure 7: Average Flow Inter-Arrival Time for Benign and Bot Traffic.

Figure 8 illustrates the RST (Reset) flag frequency in Benign and Bot traffic from the AWS cloud environment dataset.

The figure shows a stark contrast between the two categories. On Bot traffic, more than 98% of them carry the RST flag set, which means that these flows prefer to close connections abruptly. This is common with automated attacks or malicious traffic, as the bots tend to open and close connections rapidly, one after another, in an attempt to scan targets, avoid detection, or denial of service.

In contrast, less than 20% of Benign traffic carries the RST flag, which characterizes more stable, normal network sessions.

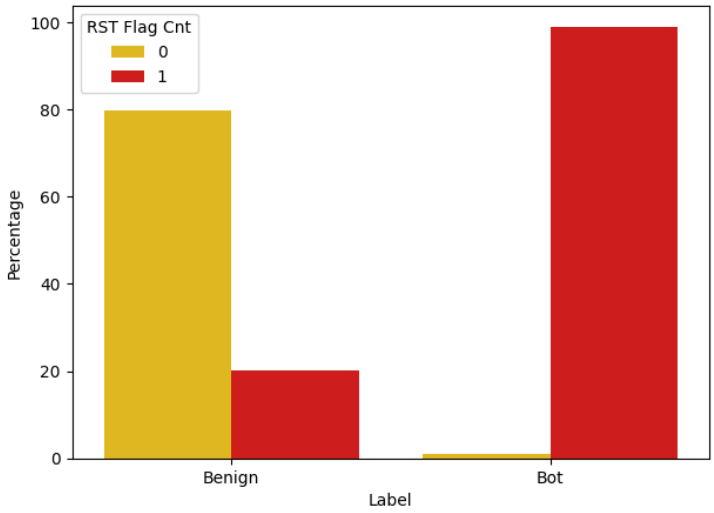


Figure 8: Distribution of RST Flag in Benign and Bot Traffic

3.4 Data Transformation

In this study, the target variable is the Label column, which classifies AWS cloud traffic into two distinct categories: Benign and Bot. The Benign class represents typical, legitimate network traffic within AWS Cloud systems, while the Bot class correspond to network traffic that pose security risks to AWS Cloud environments.

To make these categorical labels suitable for machine learning algorithms, a data transformation process was applied to convert these categorical values into numerical form. This was achieved by creating a new column, Target. In this column, Benign instances were assigned the value 0 and Bot instances were assigned the value 1. This binary encoding scheme focused on the core objective of this task: accurately identifying and isolating intrusions within large volumes of legitimate cloud traffic.

The distribution of instances across these two categories is presented in Figure 9.

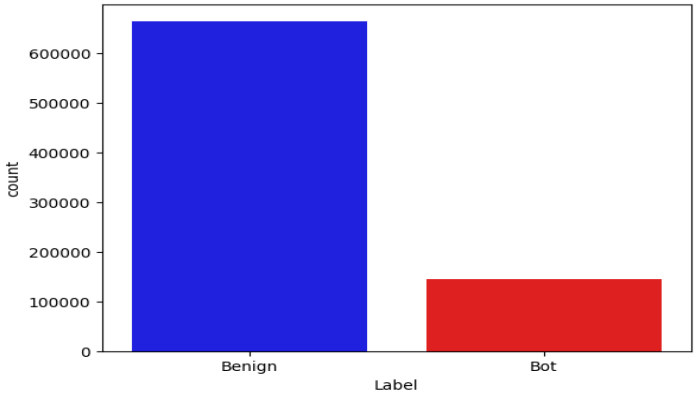


Figure 9: Distribution of Benign and Bot Instances in the Dataset

4. MODEL DEVELOPMENT

According to the usual convention of constructing a machine learning model, the target variable and predictor variables (features) were divided. The target variable is the variable on which the model is actually constructed to provide predictions, and the predictor variables are all the other dataset features with data regarding AWS cloud traffic behavior.

For uniformity and reproducibility across runs, a random seed of 123 was established before running the machine learning algorithms and the other random processes. This enables all of the random processes, like data splitting or model initialization, to consistently output the same results each time the code is run. The data was divided into a training set of 70% and a test set of 30%, employing stratified sampling to maintain the same class distribution on each set. The training data were employed to train the machine learning models, allowing them to learn relationships and patterns between the predictor variables and the response variable. The test data was employed to confirm the performance of the trained models on unseen data.

Five machine learning algorithms were utilized that utilized a novel approach towards intrusion detection: Random Forest, Decision Tree, Ridge Regression, Logistic Regression, and Linear Support Vector Machine.

4.1 Model Evaluation

Once the models had been trained against the training set, testing against the test set was carried out to determine whether or not they were able to classify unknown AWS cloud environment traffic effectively. A range of evaluation measures was used in order to obtain a complete picture of model performance, assessing both overall accuracy and class-specific performance. The outcome of these tests is outlined in the following table, presenting comparative strengths and weaknesses of each algorithm when detecting Benign and Bot traffic.

Table 2: Evaluation Outcomes of Machine Learning Models

Model	Accuracy	Precision	TPR	TNR	F1 Score	ROC AUC	Detection Time (s)
Random Forest Classifier	1.000	1.000	1.000	1.000	1.000	1.000	2.756
Decision Tree Classifier	1.000	1.000	1.000	1.000	1.000	1.000	0.056
Ridge Classifier	0.992	0.966	0.989	0.992	0.977	0.991	0.043
Logistic Regression	0.988	0.950	0.984	0.989	0.966	0.992	0.040
Linear SVC	0.962	0.838	0.979	0.959	0.903	0.962	0.074

On initial observation, the test table gives the impression that all models had good classification for AWS cloud traffic. On closer observation of the results, more detailed stories about the advantages and disadvantages of each machine learning algorithm are told.

Figure 10 plots the accuracy values of the five machine learning algorithms as the ratio of correctly classified instances in the test set. Among all the models, Random Forest Classifier and Decision Tree Classifier possessed the highest accuracy, which reflects on their greater capability to identify Benign versus Bot traffic on the AWS cloud platform. The reverse of this is that Linear SVC possessed the lowest accuracy among the five models but still a good performance.

In general, all the models were over 96% accurate and indicated their overall capacity to properly categorize Benign as well as Bot traffic and also established the appropriateness of these machine learning models for intrusion detection use in cloud environments.

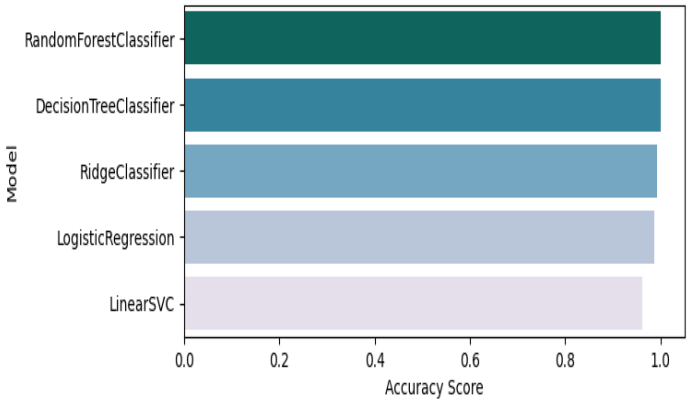


Figure 10: Accuracy Scores of Machine Learning Models

Figure 11 shows the precision of the models, which is the ratio of the instances of Bot traffic correctly predicted out of all the Bot-labeled instances. The Random Forest and Decision Tree classifiers were almost perfect in precision, meaning how dependable they were at accurately classifying Bot traffic. The worst precision was achieved by the Linear SVC (0.838) and it showed a larger number of false positives than the rest of the models.

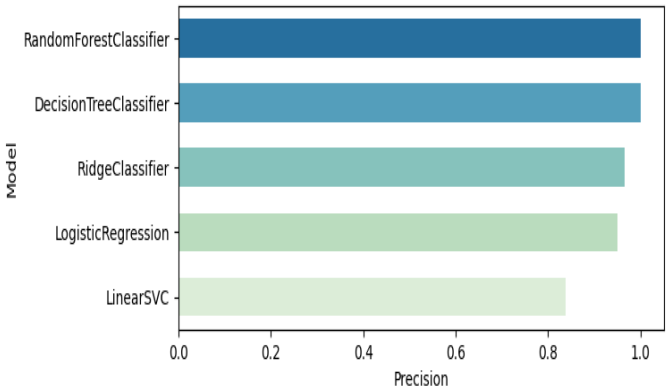


Figure 11: Precision Scores of Machine Learning Models

Figure 12 shows the F1 values of the models, the harmonic mean of the precision and recall, and present an equilibrium measure of the models' performance in intrusions detection. The Decision Tree and Random Forest classifiers obtained a perfect F1 value, which indicates that they can optimize recall and precision at the same time. The Linear SVC produced the lowest F1 value (0.903), indicating relatively higher misclassification.

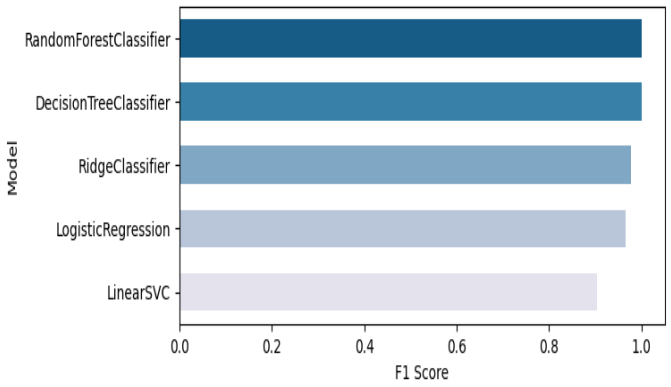


Figure 12: F1 Scores of Machine Learning Models

Figure 13 displays the true positive rates of the models, which measure the proportion of actual Bot traffic instances correctly identified. Random Forest Classifier and Decision Tree Classifier demonstrated superior ability in detecting intrusions by attaining the highest TPR. In contrast, Linear SVC exhibited the lowest TPR.

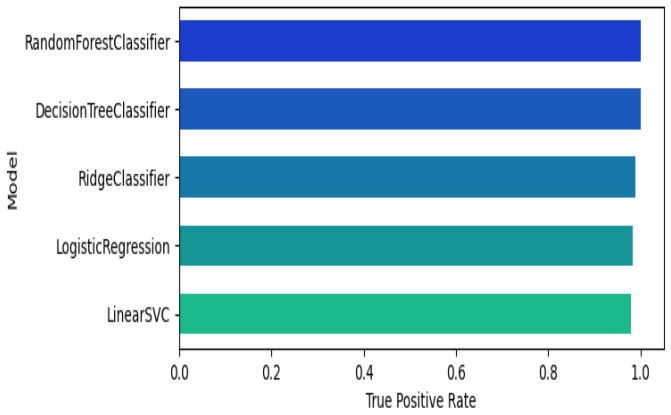


Figure 13: True Positive Rates of Machine Learning Models

Figure 14 shows the true negative rates of models, which measure the proportion of actual Benign traffic instances correctly identified. Both the Random Forest and Decision Tree classifiers achieved the highest TNR, meaning that it rarely misclassifies normal network traffic as intrusions. On the other hand, The Linear Support Vector Classifier recorded the Lowest TNR, suggesting a higher misclassification rate of Benign instances as Bot traffic.

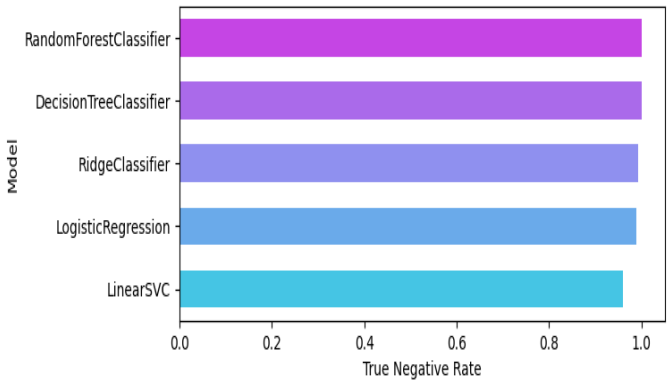


Figure 14: True Negative Rates of Machine Learning Models

Figure 15 shows the detection times in seconds for the various models. This is how long it takes a model to label each test set example as Benign or Bot.

Among the models, Logistic Regression possessed the shortest detection time, and thus it is best used where quick classification is imperative. The Random Forest Classifier, however, possessed the longest detection time, which corresponds to the added computational load inherent in its decision tree ensemble.

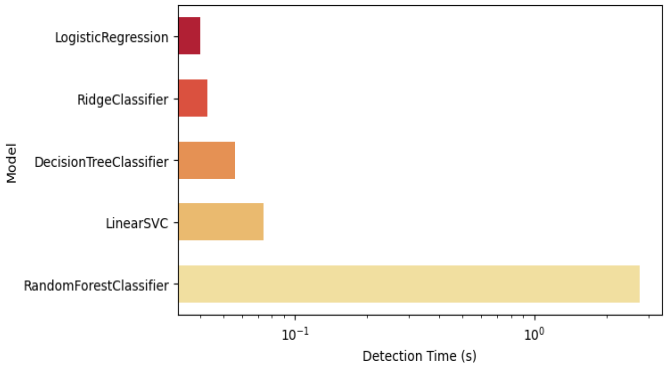
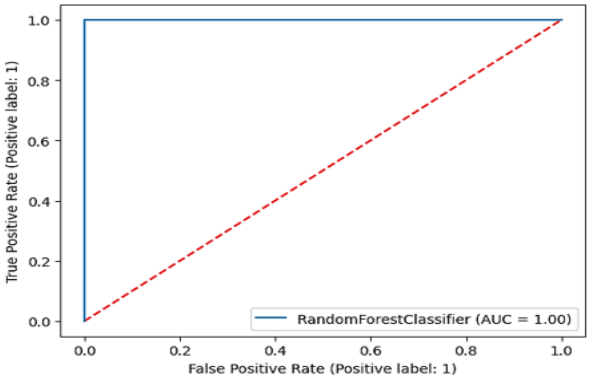
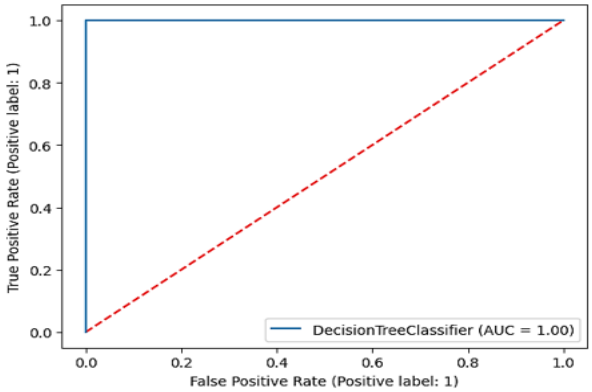


Figure 15: Detection Times of Machine Learning Models

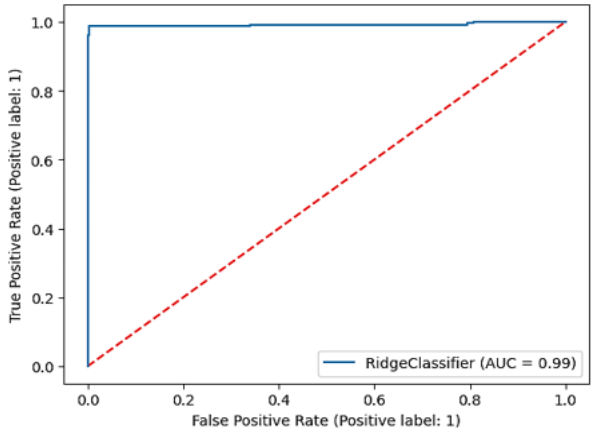
Fig. 16 (a-e) displays individual ROC AUC curves for each model recording models' discriminatory power in classifying normal traffic and intrusions on AWS cloud setups. Both Random Forest and Decision Tree classifiers obtained perfect ROC-AUC values (1.000), which indicate good two-class separability. The Ridge Classifier (0.991) and Logistic Regression (0.992) also performed extremely high discriminative power, whereas the Linear SVC had slightly lower ROC-AUC (0.962), indicating comparatively lower ability to classify between the classes.



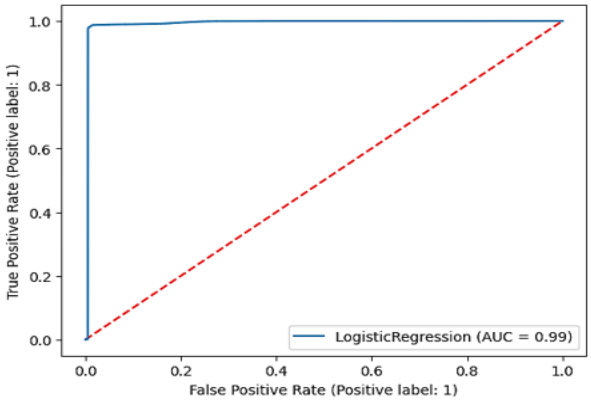
a) Random Forest Classifier



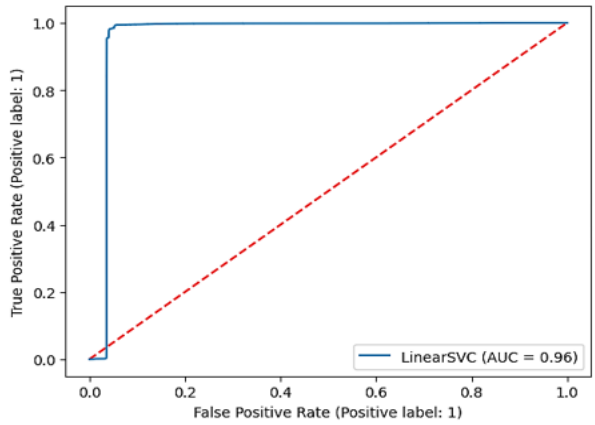
b) Decision Tree Classifier



c) Ridge Classifier



d) Logistic Regression



e) Linear SVC

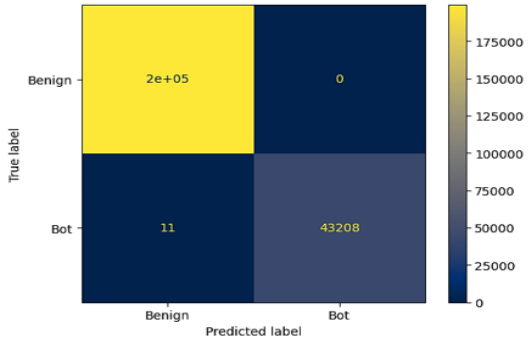
Figure 16: ROC AUC Curves of Machine Learning Models

4.3. Confusion Matrix

Figure 17 (a-e) illustrates confusion matrices that report the classification performance of the models, in terms of the number of accurate and incorrect classifications of Benign and Bot traffic.

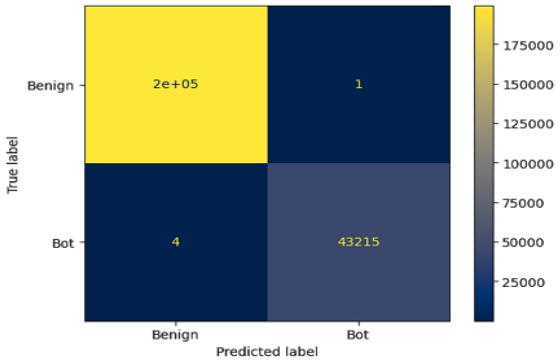
The confusion matrices display a step-by-step analysis of how each model classified Benign and Bot network traffic, including correctly and incorrectly classified instances. They are a useful tool in ascertaining the effectiveness of intrusion detection systems since they provide a means of monitoring false positives and false negatives. The division provides a better insight into how well each model can distinguish between malicious traffic and valid activity in AWS cloud infrastructure.

As shown in Figure 17a below, the Random Forest Classifier performed perfectly by marking all cases of Benign traffic with a pristine zero false positives. It did not, however, mark 11 cases of Bot traffic as Benign, causing it to have some false negatives. Apart from this minor blemish, the classifier did a perfect differentiation of normal traffic, indicating the mastery of its ability to prevent false alarms. This balance shows Random Forest's ability in high-security environments.



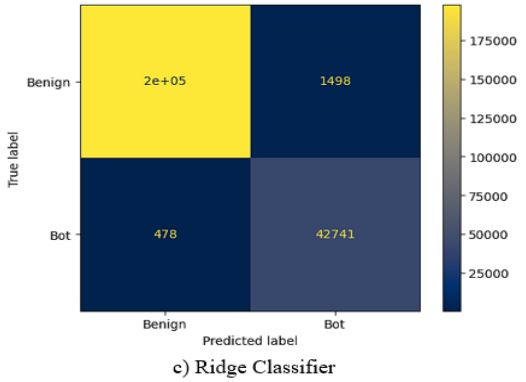
a) Random Forest Classifier

The Decision Tree Classifier performed excellently, classifying only 1 Benign traffic as Bot and 4 Bot instances as Benign. It's very low rate of misclassification makes it one of the best-performing models ever attempted. Furthermore, its accuracy and scalability to traffic streams enable the model to be applied with high success in real-time scenarios. The classifier is unique in that it minimizes both false positives and false negatives at the same time.



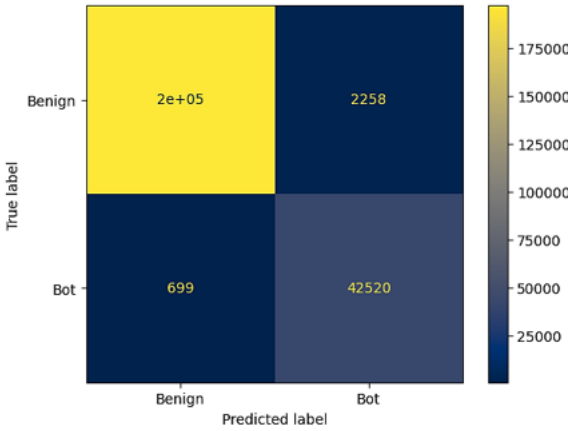
b) Decision Tree Classifier

The Ridge Classifier was robust but less accurate than the tree models. It incorrectly classified 1,498 Benign samples as Bot, yielding spurious false alarms, and 478 Bot samples as Benign, resulting in missed detections. Though still correct in the majority of instances, such misclassifications show their susceptibility to coping with wildly imbalanced traffic patterns. Its errors warrant caution in the use of Ridge Classifiers in cloud intrusion detection systems.



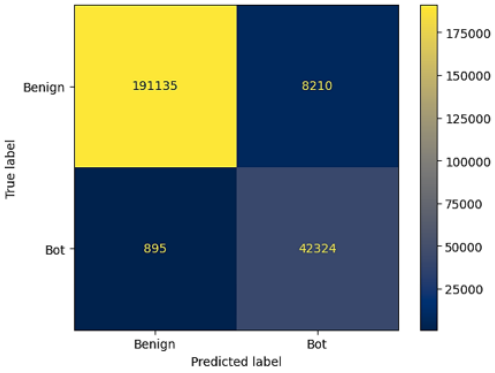
c) Ridge Classifier

The Logistic Regression model was less precise by misclassifying **2,258 Benign as Bot and 699 Bot as Benign.** It has a greater number of false positives and false negatives than the Ridge model. Although its computational behavior is superb, its misclassifications create weaknesses when used in high-demanding situations in terms of correct intrusion detection. This makes it less desirable for high-risk AWS installations.



d) Logistic Regression

The Linear SVC was the poorest performing among the models, with an astronomical number of misclassifications. It classified 8,210 Benign as Bot and created an outrageously enormous number of false positives, and 895 Bot as Benign and created false negatives. The high number makes the model unsafe for AWS intrusion detection. That it does not correctly identify normal and malicious traffic evidences linear methods' weakness in this aspect.



e) Linear SVC

This detailed analysis also brings forth the need for fine-grained measures, i.e., the confusion matrix, that give exact numbers of correct and incorrect classifications as opposed to just using close approximations like overall accuracy. Fine-grained granularity is needed in order to determine the strengths and weaknesses of single models so that high overall accuracy does not hide major misclassification patterns that would affect the effectiveness of intrusion detection on AWS cloud infrastructure.

4.4 Discussion of Results

Random Forest Classifier was outstanding on every criterion of measurement, achieving flawless results in accuracy (1.000), precision (1.000), true positive rate (1.000), true negative rate (1.000), F1 score (1.000), and ROC-AUC (1.000). This indicates that the model correctly labeled nearly all samples in the test set, effectively distinguishing Benign from Bot traffic in the AWS cloud environment.

As for detection time, the model, with great accuracy, identified all test instances in 2.756 seconds. Its moderate computational cost reflects the same. Reviewing its confusion matrix reveals that it correctly identified all Benign traffic instances and only 11 Bot instances as Benign. It's very low misclassification rate indicates that the model is highly robust and reliable since it correctly classifies nearly all the malicious traffic and has zero false positives on good traffic.

Decision Tree Classifier scored a flawless 1.000 on all the major performance metrics like accuracy, precision, TPR, TNR, F1 score, and ROC-AUC, which indicate its superb capability to classify AWS cloud traffic. Interestingly, the model only took 0.056 seconds for detection and was very efficient compared to other models. Its confusion matrix indicates that the classifier mislabeled only 1 Benign sample and 4 Bot samples, indicating its better reliability. These findings indicate that Decision Tree Classifier is not only very accurate but also very computationally efficient, and so it is the probable choice for real-time intrusion detection systems where accuracy as well as speed is the key.

The Ridge Classifier was great with 0.992 accuracy, 0.966 precision, 0.989 TPR, 0.992 TNR, 0.977 F1 score, and 0.991 ROC-AUC, and it plotted high overall performance in Benign vs. Bot traffic classification in AWS's cloud environment. In computational efficiency, it classified all instances in the test set in only 0.043 seconds, placing it among the fastest models to be tested. But unlike Decision Tree and Random Forest, more misclassifications were recorded by the Ridge Classifier, including 1,498 misclassifications of Benign samples to Bot incorrectly and 478 misclassifications of Bot samples to Benign incorrectly. This kind of misclassification shows that although Ridge Classifier is fine, it's not so accurate as compared to tree-based classifiers, particularly in the correct classification of Benign traffic. Nevertheless, it is quick and an acceptable choice for intrusion detection cases.

The Logistic Regression model was 0.988 accurate, 0.950 precise, 0.984 TPR, 0.989 TNR, 0.966 F1 score, and 0.992 ROC-AUC, with extremely good overall performance in classifying AWS cloud traffic. It was identified within 0.040 seconds, the quickest among models. Logistic Regression created more misclassifications, labeling 2,258 Benign samples as Bot's and 699 Bot samples as Benign. Higher error rates mean that, despite Logistic Regression being computationally economical, it is not as accurate in marking malicious traffic and avoiding false positives.

Linear Support Vector Classifier had an accuracy of 0.962, precision of 0.838, TPR of 0.979, TNR of 0.959, F1 score of 0.903, and ROC-AUC of 0.962, which has the worst total performance among the five models. It was identified in 0.074 seconds, slightly more than the others in the linear and ridge-based models. Its confusion matrix indicates that it mislabeled 8,210 Benign as Bot and 895 Bot as Benign, with the greatest number of mislabels. Such findings emphasize that although Linear SVC can identify traffic patterns, it does not fare so well for accurate intrusion detection in AWS cloud environments, particularly if reducing false positives and false negatives is essential.

Based on the experimental outcomes, the Decision Tree Classifier emerges as the most suitable choice for intrusion detection within AWS cloud environments due to its perfect classification performance coupled with improved computational efficiency. Similar to the Random Forest Classifier, it possessed accuracy, precision, TPR, TNR, F1 score, and ROC-AUC of 1.000, but misclassified fewer samples, marking only 1 Benign sample and 4 Bot samples compared to 11 misclassified Bot samples by Random Forest. Also, with a detection time of 0.056 seconds, its performance is significantly better than Random Forest's 2.756 seconds and therefore much more appropriate in real-time intrusion detection applications where speed is of the highest importance. With this best-of-both-worlds combination of impeccable accuracy and superior speed, the Decision Tree Classifier can precisely identify malicious traffic with minimal latency and therefore is the most efficient and realistic model to deploy in AWS cloud security systems.

5. RESEARCH GAP AND CONTRIBUTION TO CLOUD SECURITY

Despite the growing adoption of AWS cloud services, intrusion detection remains a pressing challenge, as traditional signature-based and anomaly detection systems are often unable to cope with the scale and complexity of modern cloud environments. Many existing approaches either fail to generalize to novel attack types or introduce significant false positives that compromise operational reliability [14]. The research gap lies in the lack of lightweight, high-performing machine learning models that can deliver accurate intrusion detection while maintaining computational efficiency suitable for AWS environments. Addressing this gap requires models that can analyze network flow data in real time and detect malicious activity with minimal latency [19].

This study contributes to filling that gap by implementing and comparing five machine learning algorithms on the CSE-CIC-IDS2018 dataset, which simulates AWS-style cloud network traffic. The Decision Tree Classifier demonstrated the best performance, achieving an accuracy of 100%, precision of 100%, recall of 100%, and F1 score of 100%, while maintaining a detection time of just 0.056 seconds. These results indicate that the model not only classifies traffic with near-perfect accuracy but also operates with exceptional efficiency, making it highly suitable for integration into AWS cloud monitoring workflows [27]. Unlike linear models such as Logistic Regression, which misclassified thousands of instances, the Decision Tree achieved almost flawless detection with minimal false positives, making it ideal for deployment in real-time environments where precision is critical [21].

Its impact on cloud security is considerable. By offering a balanced model of accuracy and response time, it presents a useful framework for AWS intrusion detection systems based on flow-level traffic analysis. In real-world applications, the model can be integrated with AWS-native offerings such as GuardDuty, VPC Flow Logs, and CloudWatch to extend an additional layer of visibility in the identification of malicious traffic before escalation into data breaches. Beyond AWS, the results apply to more general cloud security use cases, where cloud-scalable and affordable ML-driven intrusion detection can support organizations in protecting workloads without excessive resource utilization [33]. What this shows is that machine learning, if brought to bear on real data sets and tested through careful evaluation, can actually enhance the capacity of cloud-based systems to meet evolving cyber-attacks.

6. CONCLUSION

In this research, a machine learning application for intrusion detection in AWS cloud systems based on the CSE-CIC-IDS2018 dataset was explored.

The study aimed at determining effective models for classifying malicious and normal network traffic for resolving the growing issues of cloud security. Five machine learning models were tested and compared: Random Forest, Decision Tree, Ridge Regression, Logistic Regression, and Linear Support Vector Classifier. They were evaluated with various measures such as accuracy, precision, recall, F1-score, ROC-AUC, confusion matrix, and detection time.

The analysis indicated that the best-performing Decision Tree Classifier had the highest accuracy. It attained 100 percent accuracy, precision, recall, and F1 score with the best detection speed at 0.056 seconds. The outcome suggests that it is fitting for real-time intrusion detection in AWS networks, where both accuracy and speed are of utmost importance. In comparison to the other models, which had fluctuating rates of misclassification, the Decision Tree was remarkably efficient and reliable.

The conclusions of this research contribute to cloud security with evidence that comprehensible, light-weight machine learning models can be effectively deployed in AWS monitoring services. The deployment increases predictive threat detection, protects sensitive assets, and increases the reliability of cloud-hosted infrastructure.

7. FUTURE WORK

Although this research has proven that intrusion detection in cloud environments of AWS does leverage machine learning models, there are some avenues of future work. One of them is the use of more sophisticated deep learning algorithms, including Convolutional Neural Networks and Recurrent Neural Networks, that are capable of learning high-level temporal and spatial patterns from network traffic data. Such models have the capability to enhance detection even further, particularly against novel types of attacks.

Another promising direction is the adoption of hybrid and ensemble-based methods. A combination of classifiers might potentially bring together the advantages of various algorithms and eliminate the deficiencies that occur in single models. For instance, combining tree-based models with neural networks or the use of anomaly detection techniques could enhance robustness against a variety of attacks.

Lastly, production deployment of intrusion detection models in AWS needs scalability, flexibility, and integration with intrinsic security capabilities like AWS GuardDuty and CloudWatch. Online learning techniques that update models perpetually from new traffic patterns and keep them well-performing in changing cloud settings are equally important areas of future research. Solving these aspects will not only enhance AWS-specific security but also give a model to intrusion detection for more generic multi-cloud settings.

8. RECOMMENDATIONS

The results of this research form the basis of the potential that machine learning algorithms, particularly the Decision Tree Classifier, possess towards enhancing intrusion detection in AWS ecosystems. Several suggestions, in light of such outcomes, are made for both the researchers and the practitioners.

At first, organizations processing loads on AWS would be wise to focus on light but highly precise models like Decision Trees for real-time intrusion detection. The model, with 100% precision in keeping detection time low, is well suited for environments that demand rapid response to cyber-attacks [34]. These types of models being used in AWS security procedures can eliminate detection latency at low false positives.

Second, ML-based intrusion detection must be supplemented by AWS-native services like GuardDuty, CloudTrail, and CloudWatch. These vendors offer monitoring, anomaly detection, and logging capabilities that, when supplemented with ML classifiers, offer greater visibility and adaptive response to threats [27]. With such a hybrid configuration, organizations can establish more efficient security systems optimized specifically for dynamic cloud workloads.

Third, the models need to be retrained periodically to accommodate changing attack vectors, e.g., botnets and zero-day attacks. Static models become outdated over time, and therefore continuous learning mechanisms, e.g., online learning and federated learning, are critical to maintain them as robust [19]. Scalable retraining pipelines in future research should be able to update the models to new material without generating excessive downtime.

Lastly, intrusion detection systems should be incorporated into a multi-layered defense posture. ML models should operate in conjunction with encryption, proper access control, and optimal-practice vulnerability management to deliver an overall security posture [40]. In case detection measures fail, these extra measures deliver continuity and safeguarding of cloud-resident assets.

Overall, the use of Decision Tree-based IDS, AWS service integration, and a layered defense approach are prime advancements towards securing AWS environments from more sophisticated intrusion attempts.

REFERENCES

[1] B. R M and J. K. M K, “Intrusion Detection on AWS Cloud through Hybrid Deep Learning Algorithm,” Electronics, vol. 12, no. 6, p. 1423, Mar. 2023, doi: https://doi.org/10.3390/electronics12061423.

[2] Abdel-Rahman Al-Ghuwairi, Yousef Sharrab, Dimah Al-Fraihat, Majed AlElaimat, Ayoub Alsarhan, and Abdulmohsen Algarni, “Intrusion detection in cloud computing based on time series anomalies utilizing machine learning,” Journal of Cloud Computing, vol. 12, no. 1, Aug. 2023, doi: https://doi.org/10.1186/s13677-023-00491-x.

[3] Z. Long, H. Yan, G. Shen, X. Zhang, H. He, and L. Cheng, “A Transformer-based network intrusion detection approach for cloud security,” Journal of Cloud Computing, vol. 13, no. 1, Jan. 2024, doi: https://doi.org/10.1186/s13677-023-00574-9.

[4] M. Sajid et al., “Enhancing intrusion detection: a hybrid machine and deep learning approach,” Journal of Cloud Computing Advances Systems and Applications, vol. 13, no. 1, Jul. 2024, doi: https://doi.org/10.1186/s13677-024-00685-x.

[5] U. Ahmed et al., “Explainable AI-based innovative hybrid ensemble model for intrusion detection,” Journal of Cloud Computing Advances Systems and Applications, vol. 13, no. 1, Oct. 2024, doi: https://doi.org/10.1186/s13677-024-00712-x.

[6] L. D. Manocchio, S. Layeghy, W. W. Lo, G. K. Kulatilleke, M. Sarhan, and M. Portmann, “FlowTransformer: A transformer framework for flow-based network intrusion detection systems,” Expert Systems with Applications, vol. 241, p. 122564, May 2024, doi: https://doi.org/10.1016/j.eswa.2023.122564.

[7] A. Zohourian, S. Dadkhah, H. Molyneaux, E. C. P. Neto, and A. A. Ghorbani, "IoT-PRIDS: Leveraging packet representations for intrusion detection in IoT networks," Computers & Security, vol. 146, p. 104034, Aug. 2024, doi: https://doi.org/10.1016/j.cose.2024.104034.
[8] M. Awad, S. Fraihat, K. Salameh, and A. Al Redhaei, "Examining the Suitability of NetFlow Features in Detecting IoT Network Intrusions," Sensors, vol. 22, no. 16, p. 6164, Aug. 2022, doi: https://doi.org/10.3390/s22166164.
[9] W. H. Aljuaid and S. S. Alshamrani, "A Deep Learning Approach for Intrusion Detection Systems in Cloud Computing Environments," Applied Sciences, vol. 14, no. 13, p. 5381, Jan. 2024, doi: https://doi.org/10.3390/app14135381.
[10] Q. Zhou and D. Pezaros, "Evaluation of Machine Learning Classifiers for Zero-Day Intrusion Detection -- An Analysis on CIC-AWS-2018 dataset," arXiv.org, 2018. https://arxiv.org/abs/1905.03685?utm_source=chatgpt.com (accessed Aug. 30, 2025).
[11] S. S. Nasim, P. Pranav, and S. Dutta, "A systematic literature review on intrusion detection techniques in cloud computing," Discover Computing, vol. 28, no. 1, Jun. 2025, doi: https://doi.org/10.1007/s10791-025-09641-y.
[12] H. Attou, A. Guezzaz, S. Benkirane, M. Azrour, and Y. Farhaoui, "Cloud-Based Intrusion Detection Approach Using Machine Learning Techniques," Big Data Mining and Analytics, vol. 6, no. 3, pp. 311–320, Sep. 2023, doi: https://doi.org/10.26599/bdma.2022.9020038.
[13] W. Almuseelem, "Perspective Chapter: Intrusion Detection Systems in Cloud Environment," Mastering Intrusion Detection for Cybersecurity [Working Title], Jan. 2025, doi: https://doi.org/10.5772/intechopen.1008756.
[14] A. Alshammari and A. Aldribi, "Apply machine learning techniques to detect malicious network traffic in cloud computing," Journal of Big Data, vol. 8, no. 1, Jun. 2021, doi: https://doi.org/10.1186/s40537-021-00475-1.
[15] Kumar Saurabh, V. Sharma, U. Singh, Rahamatullah Khondoker, R. Vyas, and O. P. Vyas, "HMS-IDS: Threat Intelligence Integration for Zero-Day Exploits and Advanced Persistent Threats in IIoT," Arabian Journal for Science and Engineering, Jul. 2024, doi: https://doi.org/10.1007/s13369-024-08935-5.
[16] László Göcs and Zsolt Csaba Johanyák, "Identifying relevant features of CSE-CIC-IDS2018 dataset for the development of an intrusion detection system," Intelligent Data Analysis, vol. 28, no. 6, pp. 1527–1553, Mar. 2024, doi: https://doi.org/10.3233/ida-230264.
[17] G. T V, D. A J, and M. L. M, "Deep learning method for efficient cloud IDS utilizing combined behavior and flow-based features," Applied Intelligence, vol. 54, no. 8, pp. 6738–6759, Apr. 2024, doi: https://doi.org/10.1007/s10489-024-05505-y.
[18] R. S. S. Kumar, A. Wicker, and M. Swann, "Practical Machine Learning for Cloud Intrusion Detection," Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, Nov. 2017, doi: https://doi.org/10.1145/3128572.3140445.
[19] A. Umar et al., "Energy-efficient deep learning-based intrusion detection system for edge computing: a novel DNN-KDQ model," Journal of Cloud Computing Advances Systems and Applications, vol. 14, no. 1, Jul. 2025, doi: https://doi.org/10.1186/s13677-025-00762-9.
[20] Z. kadhim Alitbi, S. A. Hosseini Seno, A. Ghaemi Bafghi, and D. Zabihzadeh, "A Generalized and Real-Time Network Intrusion Detection System Through Incremental Feature Encoding and Similarity Embedding Learning," Sensors, vol. 25, no. 16, p. 4961, Aug. 2025, doi: https://doi.org/10.3390/s25164961.
[21] D.-J. Munoz, M. Pinto, and L. Fuentes, "Detecting feature influences to quality attributes in large and partially measured spaces using smart sampling and dynamic learning," Knowledge Based Systems, vol. 270, pp. 110558–110558, Jun. 2023, doi: https://doi.org/10.1016/j.knosys.2023.110558.
[22] M.-T. Kao, D.-Y. Sung, S.-J. Kao, and F.-M. Chang, "A Novel Two-Stage Deep Learning Structure for Network Flow Anomaly Detection," Electronics, vol. 11, no. 10, p. 1531, May 2022, doi: https://doi.org/10.3390/electronics11101531.
[23] M. Saqib, D. Mehta, F. Yashu, and S. Malhotra, "Adaptive Security Policy Management in Cloud Environments Using Reinforcement Learning," arXiv.org, 2025.https://arxiv.org/abs/2505.08837?utm_source=chatgpt.com (accessed Aug. 31, 2025).
[24] S. Zavrak and M. Iskefiyeli, "Flow-based intrusion detection on software-defined networks: a multivariate time series anomaly detection approach," Neural Computing and Applications, Mar. 2023, doi: https://doi.org/10.1007/s00521-023-08376-5.
[25] T. B. Ogunseyi and G. Thiyagarajan, "An Explainable LSTM-Based Intrusion Detection System Optimized by Firefly Algorithm for IoT Networks," Sensors, vol. 25, no. 7, p. 2288, Apr. 2025, doi: https://doi.org/10.3390/s25072288.
[26] C. Du, Y. Guo, and Y. Zhang, "A Deep Learning-Based Intrusion Detection Model Integrating Convolutional Neural Network and Vision Transformer for Network Traffic Attack in the Internet of Things," Electronics, vol. 13, no. 14, pp. 2685–2685, Jul. 2024, doi: https://doi.org/10.3390/electronics13142685.
[27] H. Kamal and M. Mashaly, "Enhanced Hybrid Deep Learning Models-Based Anomaly Detection Method for Two-Stage Binary and Multi-Class Classification of Attacks in Intrusion Detection Systems," Algorithms, vol. 18, no. 2, p. 69, Jan. 2025, doi: https://doi.org/10.3390/a18020069.
[28] Yutaro Iizawa, Norihiro Okui, Y. Akimoto, S. Fukushima, A. Kubota, and T. Yoshida, "A Study of Anomalous Communication Detection for IoT Devices Using Flow Logs in a Cloud Environment," pp. 410–415, Jan. 2025, doi: https://doi.org/10.5220/0013461200003944.
[29] C. Xu, F. Zhang, Z. Yang, Z. Zhou, and Y. Zheng, "A few-shot network intrusion detection method based on mutual centralized learning," Scientific Reports, vol. 15, no. 1, Mar. 2025, doi: https://doi.org/10.1038/s41598-025-93185-0.
[30] F. Ullah, A. Turab, S. Ullah, Diletta Cacciagrano, and Y. Zhao, "Enhanced Network Intrusion Detection System for Internet of Things Security Using Multimodal Big Data Representation with Transfer Learning and Game Theory," Sensors, vol. 24, no. 13, pp. 4152–4152, Jun. 2024, doi: https://doi.org/10.3390/s24134152.
[31] J. Toldinas, A. Venčkauskas, A. Liutkevičius, and N. Morkevičius, "Framing Network Flow for Anomaly Detection Using Image Recognition and Federated Learning," Electronics, vol. 11, no. 19, p. 3138, Sep. 2022, doi: https://doi.org/10.3390/electronics11193138.

[32] A. Pinto, L.-C. Herrera, Y. Donoso, and J. A. Gutierrez, "Survey on Intrusion Detection Systems Based on Machine Learning Techniques for the Protection of Critical Infrastructure," Sensors, vol. 23, no. 5, p. 2415, Feb. 2023, doi: https://doi.org/10.3390/s23052415.
[33] M. Rodríguez, Á. Alesanco, L. Mehavilla, and J. García, "Evaluation of Machine Learning Techniques for Traffic Flow-Based Intrusion Detection," Sensors, vol. 22, no. 23, p. 9326, Jan. 2022, doi: https://doi.org/10.3390/s22239326.
[34] Z. Li, P. Wang, and Z. Wang, "FlowGANAnomaly: Flow-Based Anomaly Network Intrusion Detection with Adversarial Learning," Chinese Journal of Electronics, vol. 33, no. 1, pp. 58–71, Jan. 2024, doi: https://doi.org/10.23919/cje.2022.00.173.
[35] J. V, M. P. K, G. G, and D. B. D, "DLMHS: Flow-based intrusion detection system using deep learning neural network and meta-heuristic scale," International Journal of Communication Systems, vol. 35, no. 10, Apr. 2022, doi: https://doi.org/10.1002/dac.5159.
[36] Y. Xiao, Y. Feng, and K. Sakurai, "An Efficient Detection Mechanism of Network Intrusions in IoT Environments Using Autoencoder and Data Partitioning," Computers, vol. 13, no. 10, pp. 269–269, Oct. 2024, doi: https://doi.org/10.3390/computers13100269.
[37] A. Alabbadi and F. Bajaber, "An Intrusion Detection System over the IoT Data Streams Using eXplainable Artificial Intelligence (XAI)," Sensors, vol. 25, no. 3, p. 847, Jan. 2025, doi: https://doi.org/10.3390/s25030847.
[38] CSE-CIC-IDS2018, "AWS Marketplace: A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018)," Amazon.com, 2018. https://aws.amazon.com/marketplace/pp/prodview-qkyroawpr2aw6?utm_source=chatgpt.com (accessed Aug. 31, 2025).
[39] Dmytrii Tykholaz, R. Banakh, Lesya Mychuda, Andrian Piskozub, and R. Kyrychok, "Incident response with AWS detective controls," Cybersecurity Providing in Information and Telecommunication Systems II 2024, Oct. 2024, Available: https://www.researchgate.net/publication/385858920_Incident_response_with_AWS_detective_controls?utm_source=chatgpt.com.
[40] None Kenda Alamer, "A Machine Learning-Driven Cloud Intrusion Detection: Impact of Feature Engineering and Dimensionality Reduction on Classification and Optimization," Journal of Information Systems Engineering & Management, vol. 10, no. 43s, pp. 551–570, May 2025, doi: https://doi.org/10.52783/jisem.v10i43s.8444.

DATASET REFERENCE

Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization", 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018