Int. J. Inf. Cybersec.-2022

Dimensionality Reduction Based Intrusion Detection System in Cloud Computing Environment Using Machine Learning

Raghava Satya SaiKrishna Dittakavi
Independent Researcher

Abstract

The rapid expansion of cloud computing has been tempered by concerns surrounding privacy and security. To tackle these issues, intrusion detection systems with machine learning techniques are increasingly being deployed in cloud environment. However, computational cost and model complexity remain major challenges. To this end, the present study proposed a dimensionality-reduction based IDS for cloud computing environments to minimize computational costs in cloud environment. Using the CSE-CIC-IDS2018 dataset, which comprises about 16 million instances and covers a broad array of attack types, we applied dimensionality reduction methods Principal Component Analysis (PCA), Non-negative Matrix Factorization (NMF), and High Correlation Filter. The resulting feature set was narrowed down to 12 essential features, which include flow duration and rate metrics, packet count and size metrics in both forward and backward directions, inter-arrival time metrics for flows in both directions, TCP flag metrics, header size metrics, and bulk transfer metrics in both forward and backward directions. Machine learning models were then trained to classify instances as either benign or attack-oriented. The models employed for classification were Gradient Boosting, Random Forest, Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), and Logistic Regression, listed in order of performance. This research argued that dimensionality reduction not only simplifies the machine learning models but also reduces computational costs and the risk of overfitting, thereby improving the cost and computational efficiency and reliability of intrusion detection systems in cloud computing.

Keywords: *Cloud Computing, Intrusion Detection System, CICIDS2018, Dimensionality Reduction, Machine Learning*

Introduction

In recent years, the adoption of cloud-based services has become increasingly prevalent among modern enterprises. This transition allows organizations to focus more on their core business activities by outsourcing the management of IT-related tasks to third-party vendors. While cloud providers generally adhere to industry best practices to maintain the security of their infrastructure, the onus of safeguarding data and applications often remains with the enterprise (*1*, *2*). It is crucial for organizations to understand that migrating to the cloud does not absolve them of their responsibilities concerning data asset security and accountability. Measures such as encryption, access control, and regular auditing are essential to protect sensitive data. Moreover, coordination between internal security teams and cloud service providers is vital to establish a secure environment, especially in hybrid or multi-cloud deployments (*3*, *4*).

The risks associated with inadequate cloud security have been exacerbated by the increasing sophistication of cyber threats. Cloud computing environments have become prime targets for attackers, partly because organizations often lack visibility and control over data access and movement within the cloud. Advanced threats, such as data breaches, ransomware, and insider attacks, can exploit vulnerabilities in the cloud infrastructure or applications. Failure to implement robust security protocols can result in significant governance and compliance risks, especially when handling sensitive client information (*5*, *6*).

Intrusion detection focuses on the accurate identification of potential threats that could damage or compromise an information system. The primary objective of an intrusion detection system (IDS) is to promptly detect and alert administrators about any malicious activities, ensuring that timely countermeasures can be taken to safeguard the integrity of the system. Such systems are designed to recognize a wide range of attacks, thereby ensuring the protection of valuable data and preserving the functionality of the system (*7*, *8*).

An IDS can be categorized based on where it is deployed and the type of data it analyzes. A host-based IDS, for example, is primarily concerned with the internal monitoring of a specific computer or system. It operates by examining the activities within the system it is installed on. Some of the primary tasks performed by a host-based IDS include monitoring changes to the Windows registry, analyzing logs for suspicious activities, and checking the integrity of files. This type of system provides an in-depth view of what is happening inside the host, offering a line of defense against threats that may originate from within the system (*9*).

On the other hand, a network-based IDS operates by monitoring and analyzing network traffic. It looks for patterns or signatures that might indicate malicious activities. This type of system is especially important for detecting threats that aim to exploit vulnerabilities in the network or disrupt its operations. Examples of threats that a network-based IDS might detect include Denial-of-Service (DoS) attacks, SQL injection attacks, and password attacks (*10*, *11*).

Intrusion Detection Systems (IDS) are primarily classified into two categories based on their detection methodologies: signature-based and anomaly-based. The signature-based IDS operates by relying on a predefined database of known attack patterns or signatures. These patterns are derived from previously identified and analyzed malicious activities. When incoming traffic matches any of these known patterns, the IDS triggers an alert, indicating a potential intrusion. However, one of the major limitations of signature-based IDS is its inability to detect novel or previously unknown attacks (*12*, *13*). Because of this limitation, the database containing attack signatures must be incessantly updated. This constant need for updates ensures that the system remains effective against the latest threats but can also be resource-intensive and requires vigilant monitoring to ensure that the system is always equipped with the most recent attack signatures.

In contrast to the signature-based approach, anomaly-based IDS operates by establishing a baseline of what is considered "normal" behavior within a system or network. This baseline is derived from a comprehensive analysis of regular and benign traffic patterns over a specific period. Once this standard behavior is established, the anomaly-based IDS continuously monitors the system or network for any deviations from this baseline (*14*, *15*). Such deviations, which could be sudden spikes in traffic, unusual access patterns, or any other irregularities, are flagged as potential threats. What sets anomaly-based IDS apart is its inherent ability to detect previously unknown attacks or zero-day vulnerabilities, as it does not rely on known patterns but rather on behavioral deviations.

Intrusion Detection Systems (IDS) in a cloud computing environment serve as a critical layer of security to monitor, detect, and flag malicious activities or policy violations. The architecture and deployment of IDS in the cloud differ from those in traditional network settings due to the inherent complexities and unique characteristics of cloud environments, such as virtualization, resource pooling, and elasticity. For instance, in cloud infrastructures, IDS can be deployed at different levels, such as host-based, network-based, or hypervisor-based, to monitor different types of data, from network packets to system calls and logs. Host-based IDS may

be installed on each virtual machine, while network-based IDS monitor traffic between virtual machines and external networks (*16*, *17*). Hypervisor-based IDS operate at the hypervisor level to oversee activities across multiple virtual machines. Reducing computational costs in Intrusion Detection Systems (IDS) deployed in cloud computing environments is of significant importance for several reasons. Cloud computing is fundamentally based on the model of resource sharing and virtualization, where resources like processing power, memory, and storage are allocated dynamically. High computational costs for running IDS could lead to inefficient use of these shared resources, thereby affecting not only the IDS but also other services and applications running on the cloud. If IDS consumes excessive resources, there may be less availability for other tasks, which could result in service degradation or increased operational costs. Efficient resource utilization is especially critical in cloud environments that scale dynamically according to demand, as an IDS with high computational costs could impede this scalability (*18*).

The economic aspect cannot be ignored. Implementing an IDS with high computational requirements would necessitate more powerful hardware or additional instances, thereby increasing the overall cost of operation. This negates some of the cost benefits associated with cloud computing. Additionally, cloud service providers often charge based on resource usage; hence, an IDS that uses excessive computational resources could lead to increased operational expenses for the end-users. In a competitive market, where businesses are highly cost-sensitive, such additional costs could be detrimental to the adoption of cloud services.

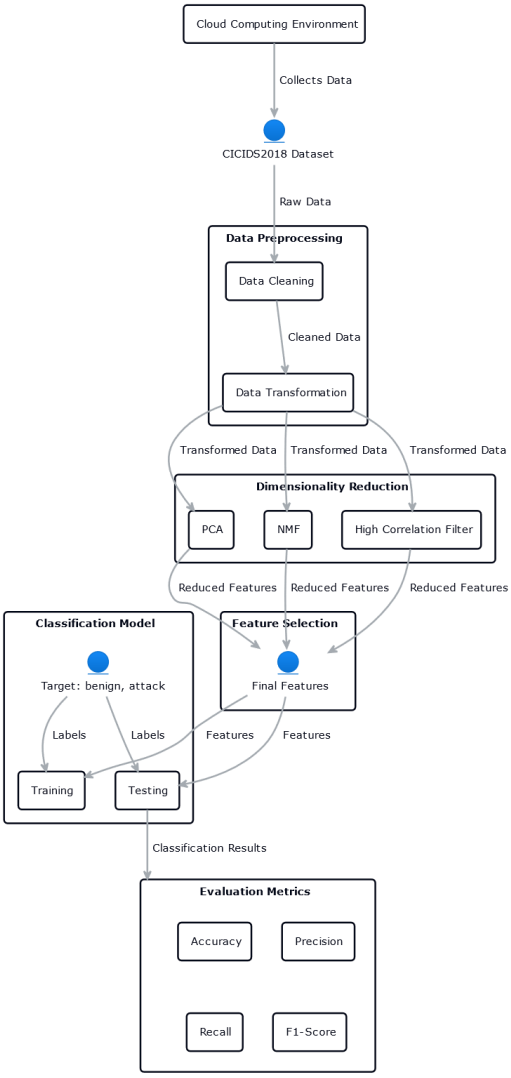
High-dimensional data, consisting of various attributes and features of network packets, can slow down the IDS, leading to delayed responses and potential security risks. Implementing dimensionality reduction techniques can expedite the detection process by reducing the computational burden on the system, enabling quicker and more effective responses to potential threats. This not only enhances security but also allows for better allocation of cloud resources, which could otherwise be wasted on unnecessary computations.

Experimental methods

Figure 1. depicts the proposed model where CICIDS2018 dataset(*19*) is initially collected. The data undergoes preprocessing steps like cleaning and transformation. For dimensionality reduction, three technique s—PCA, NMF, and High Correlation Filter—are applied. The selected features are then used for training and testing a classification model to differentiate between benign and attack targets. The

evaluation metrics include accuracy, precision, recall, and F1-score, which are calculated based on the classification results.

Figure 1. Proposed IDS in cloud computing environment with dimensionality reduction



Dimensionality reduction

High-dimensional data is often burdensome in terms of storage, computation, and network transfer. For example, machine learning models trained on high-dimensional data may require significantly more computational resources for training and inference (*20*, *21*). This increased requirement can lead to slower processing times and higher resource allocation, subsequently driving up the operational costs in a cloud environment where resources are billed based on usage.

In cloud-based data analytics or real-time monitoring systems like IDS, high-dimensional data can also lead to challenges in scalability and real-time processing. Handling a large number of features in real-time analytics could result in latency and decrease the system's ability to scale dynamically according to demand. When dimensionality reduction techniques such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), or feature selection methods are applied, the essential characteristics of the data are retained while reducing the number of features. This reduced dataset is more manageable computationally, allowing for quicker data processing, less storage space, and ultimately, cost savings. The reduced computational burden enables the cloud environment to allocate resources more efficiently, thereby allowing for better scalability and potentially lowering the costs for both the service provider and the end-users.

Principal Component Analysis (PCA)

PCA is used to transform the original variables into a new set of variables known as principal components, which are orthogonal to each other, and reflect the maximum variance in the data (*22*). The first principal component reflects the most variance, the second (which is orthogonal to the first) reflects the second most, and so on. Mathematically, given a data matrix ($\mathbf{X}$) of dimensions ($\mathbf{n} \times \mathbf{p}$) where, ***(n),*** is the number of observations and p is the number of variables), PCA seeks to find a set of ($\boldsymbol{p}$) orthogonal vectors ($\boldsymbol{u_1}, \boldsymbol{u_2}, \ldots, \boldsymbol{u_p}$) that maximize the following:

$$[\boldsymbol{Var(Xu) = u^T \Sigma u}]$$

Subject to:

$$[\boldsymbol{u^T u = 1}]$$

Where (Σ) is the covariance matrix of (X)

Non-negative Matrix Factorization (NMF)

NMF is used to factorize a non-negative matrix (V) into two lower-dimensional non-negative matrices ($\boldsymbol{W}$) and ($\boldsymbol{H}$), such that ($\boldsymbol{V} \approx \boldsymbol{W} \times \boldsymbol{H}$). Given a matrix (V) of dimensions (n × m), NMF aims to find matrices (W) of dimensions ($\boldsymbol{n} \times \boldsymbol{k}$) $\boldsymbol{and}$ ($\boldsymbol{H}$) of dimensions ($\boldsymbol{k} \times \boldsymbol{m}$), where ($\boldsymbol{k}$) is much smaller than ($\boldsymbol{n}$) $\boldsymbol{and}$ ($\boldsymbol{m}$), to approximate ($\boldsymbol{V}$). The objective function to minimize is often:

$$[|\boldsymbol{V} - \boldsymbol{WH}|_{\boldsymbol{F}}^{\boldsymbol{2}}]$$

Where $(|\cdot|_{\mathbf{F}})$ is the Frobenius norm.

High Correlation Filter

The High Correlation Filter is a feature selection method that removes features that are highly correlated with each other, retaining only one feature from each set of correlated features. Given a set of features $(\boldsymbol{X_1}, \boldsymbol{X_2}, \ldots, \boldsymbol{X_p})$, this method computes the correlation matrix ($\boldsymbol{R}$), where the element $(\boldsymbol{r_{ij}})$ represents the correlation between $(\boldsymbol{X_i})\boldsymbol{and}(\boldsymbol{X_j})$. Features are filtered out based on a threshold ($\boldsymbol{t}$), such that if $(|\boldsymbol{r_{ij}}| > \boldsymbol{t})$, one of $(\boldsymbol{X_i})\boldsymbol{or}(\boldsymbol{X_j})$ is removed.

Machine Learning models

Gradient Boosting is a machine learning technique used for classification and regression tasks, among other predictive modeling applications. It builds on the idea of combining weak predictors to create a strong predictor by focusing on the mistakes made by previous predictors in the ensemble. The algorithm iteratively adds trees that correct the residuals—differences between observed and predicted values—of the existing ensemble. A shrinkage parameter, often referred to as the learning rate, controls the influence of each new tree. By emphasizing instances that were previously misclassified, Gradient Boosting adapts to the weaknesses of the overall model.

Random Forest is another ensemble learning method that performs well for both classification and regression tasks. Unlike Gradient Boosting, which is sequential, Random Forest constructs multiple decision trees during training and outputs the mode or mean prediction of the individual trees for classification or regression, respectively. Random Forest introduces randomness by selecting a subset of features at each split while growing a tree and by bootstrapping samples for each tree. This results in a set of diverse decision trees that collectively deliver a more robust and stable prediction. The algorithm is computationally efficient and less prone to

overfitting, as the averaging of multiple trees mitigates the risk of capturing noise in the training data.

Support Vector Machines (SVM) are designed for binary classification tasks, though extensions for multi-class classification and regression do exist. The core idea behind SVM is to find the hyperplane that maximally separates two classes of data points in the feature space. For data that is not linearly separable, SVM uses kernel functions to map the data to a higher-dimensional space where it can be linearly separated. Margin maximization ensures that the decision boundary maximizes the distance between the closest data points from different classes, enhancing the generalization capability of the model. One of the limitations of SVM is its sensitivity to the choice of the kernel and regularization parameters.

The k-Nearest Neighbors (k-NN) algorithm is a type of instance-based learning that can be used for both classification and regression. It works by comparing a test sample with k training samples that are closest in the feature space. The output is determined by a majority vote for classification tasks or by an average for regression tasks. Distance metrics, such as Euclidean distance, are typically used to identify the nearest neighbors. The algorithm is simple to implement and understand but can be computationally intensive during the prediction phase, particularly when dealing with large data sets. Additionally, the choice of k and the distance metric can significantly affect the algorithm's performance.

Logistic Regression, despite its name, is primarily used for binary classification problems. The algorithm models the probability that the target variable belongs to a particular category, typically by using the logistic function to transform a linear combination of the input features. The coefficients for the features are generally estimated through maximum likelihood estimation. Unlike more complex algorithms like Gradient Boosting and Random Forest, Logistic Regression provides a less flexible but more interpretable model, offering insights into the importance of individual features. The algorithm assumes that there is a linear relationship between the log-odds of the output and the input features, which may not hold true for all data sets.

Dataset

The CSE-CIC-IDS2018 dataset was developed by the Canadian Institute for Cybersecurity (CIC) (*19*) in collaboration with the Communications Security

Establishment (CSE) to address limitations in existing datasets, particularly their inability to represent modern network traffic patterns and attack methods. The dataset is an extensive collection of labeled network traffic records, generated by simulating a real-world organizational network environment. The data incorporates various types of attacks including Brute Force, DoS, Heartbleed, Web Attack, Infiltration, and Botnet, among others, in addition to benign network traffic. One of the most distinctive features of the CSE-CIC-IDS2018 dataset is its extensive feature set, comprising of 80 network traffic attributes, which enhances its suitability for complex analysis tasks (*23*).

The CSE-CIC-IDS2018 dataset is often compared with other benchmark datasets in the field of intrusion detection, such as KDD Cup 1999, NSL-KDD, and ISCX 2012. Table 1 provides a comparison among the benchmark datasets for IDS (*24*).

Table 1. intrusion detection benchmark datasets

Dataset	**Feature**	**Limitations**	**Notable Features**
CSE-CIC-IDS2018	Broad array of modern attack types, extensive feature set, high volume of data.	Class imbalance.	Suitable for training complex machine learning models, detailed environment for research.
KDD Cup 1999	Historically important for initial studies.	Outdated, lacks diversity in attack types.	No longer highly relevant due to evolving cyber-attacks.
NSL-KDD	Improvement over KDD Cup 1999, reduced redundant records.	Still lacks diversity in attack types.	Addresses some limitations of KDD Cup 1999.
ISCX 2012	Includes contemporary attack types, focus on realistic network environment.	Class imbalance, less extensive feature set than CSE-CIC-IDS2018.	Similar to CSE-CIC-IDS2018 but with fewer features and lower data volume.

Designed to be comprehensive in terms of network traffic patterns and intrusion scenarios, the CSE-CIC-IDS2018 dataset aids in evaluating the performance of various intrusion detection models. It is formulated to be compatible with machine learning methodologies, offering a testing environment for data preprocessing techniques, feature selection methods, and algorithmic model evaluations. CSE-CIC-IDS2018 dataset provides a more up-to-date and detailed environment compared to other benchmark dataset for intrusion detection system.

Table 2. Types of Attacks in CSE-CIC-IDS2018

Attack Type	Description
Brute Force	Unauthorized user tries to gain access by guessing passwords or exploiting vulnerabilities.
DoS/DDoS	Aim to overwhelm a network or service with excessive data to make it unavailable.
Web Attacks	Includes SQL Injection, Cross-Site Scripting (XSS), and other attacks targeting web applications.
Infiltration	Unauthorized user gains access to a network without detection.
Botnet	A network of compromised computers controlled by an attacker for tasks like spamming or data theft.
Port Scanning	Scanning a network to find open ports for exploitation.
Packet Sniffing	Intercepting and logging network traffic, usually to glean sensitive information.
Man-in-the-Middle	Interception of communications between two parties to steal or manipulate data.

Dimensionality reduction Results

The lower eigenvalue (the red line) in figure 2 at the 12th component suggests that one would most likely keep the first 12 components for subsequent steps in data analysis, as the eigenvalues after this point do not contribute significantly to explaining the variance in the data.

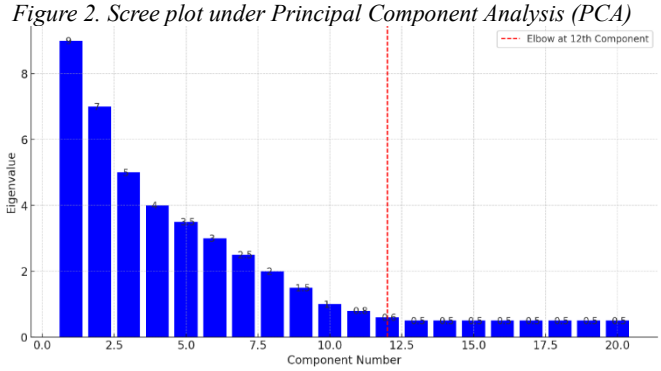
In Figure 3, the reconstruction error is plotted against the number of components, ranging from 1 to 17. The reconstruction error starts at 120 when there is only one component and generally decreases as more components are added. The purpose of such a plot is to help determine the optimal number of components for NMF by identifying a point where adding more components does not significantly reduce the reconstruction error. In this research, one might consider that the reconstruction error starts to stabilize around 12 to 13 components, as the error values become less variable and decline more slowly after that point.

Flow Duration and Rate Metrics comprise features such as flow duration (fl_dur), flow byte rate (fl_byt_s), and flow packet rate (fl_pkt_s). These metrics are essential for characterizing the temporal behavior of network flows. Flow duration is the length of time for which a flow exists, and it can offer critical insights into whether a flow is short-lived (possibly indicative of a scan or attack) or long-lived (more likely to be a legitimate user session). Flow byte rate and flow packet rate represent the rate at which bytes and packets are being transferred over the network, respectively. A sudden spike in these rates could indicate data exfiltration, flooding

attacks, or other abnormal activities, while consistently low rates might suggest an idle or less active session. Monitoring these metrics allows security administrators to set baseline patterns and thereby identify anomalies in network traffic.

Packet Count Metrics in Forward Direction include total packets in the forward direction (tot_fw_pk), average number of packets in a sub-flow in the forward direction (subfl_fw_pk), and the number of packets with at least one byte of TCP data payload in the forward direction (Fw_act_pkt). These features can serve as indicators of the nature and quantity of outgoing traffic.

Figure 2. Scree plot under Principal Component Analysis (PCA)



A high total packet count in the forward direction could imply either large data transfers or potential flooding attacks, depending on the context. On the other hand, metrics like the average number of packets in a sub-flow and the number of packets with payload data could provide more granular insights into the types of data being sent. For instance, a high count of packets with payload could suggest legitimate data transfer, while an absence might raise suspicions of covert channel communication or scanning activities.

Packet Size Metrics in Forward Direction and Backward Direction consist of a variety of features that measure different aspects of packet size. In the forward direction, features like tot_l_fw_pkt, fw_pkt_l_max, fw_pkt_l_min, fw_pkt_l_avg, and fw_pkt_l_std provide a comprehensive profile of packet sizes. "Tot_l_fw_pkt" offers the total size of packets in the forward direction, while "fw_pkt_l_max" and "fw_pkt_l_min" indicate the maximum and minimum sizes, respectively. "Fw_pkt_l_avg" provides the average packet size, and "fw_pkt_l_std" offers the standard deviation, allowing for analysis of variability in packet size. On the other hand, in the backward direction, features such as Bw_pkt_l_max, Bw_pkt_l_min,

Bw_pkt_l_avg, and Bw_pkt_l_std provide similar insights. These packet size metrics in both directions help in detecting anomalies like oversized packets that are often used in specific kinds of network attacks.

Figure 3. Error reconstruction curve under Non-negative Matrix Factorization (NMF)

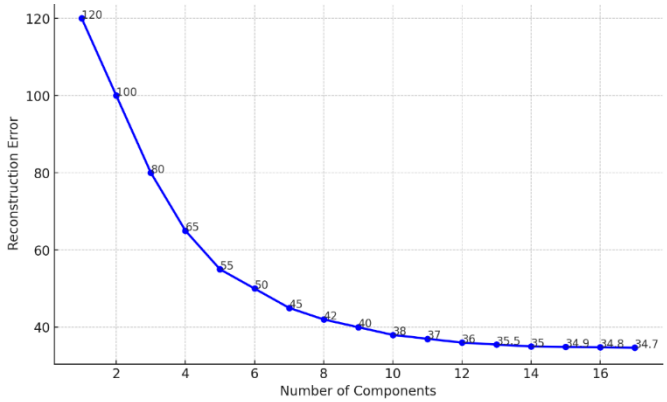


Table 3. Merged features after dimensionality reduction

Group No.	Category	Feature Names	Short definition
1	Flow Duration and Rate Metrics	fl_dur, fl_byt_s, fl_pkt_s	Metrics related to the duration and rate of network flows.
2	Packet Count Metrics in Forward Direction	tot_fw_pk, subfl_fw_pk, Fw_act_pkt	Count of packets sent in the forward direction.
3	Packet Count Metrics in Backward Direction	tot_bw_pk, subfl_bw_pkt	Count of packets sent in the backward direction.
4	Packet Size Metrics in Forward Direction	tot_l_fw_pkt, fw_pkt_l_max, fw_pkt_l_min, fw_pkt_l_avg, fw_pkt_l_std	Measures related to the size of packets in the forward direction.
5	Packet Size Metrics in Backward Direction	Bw_pkt_l_max, Bw_pkt_l_min, Bw_pkt_l_avg, Bw_pkt_l_std	Measures related to the size of packets in the backward direction.
6	Inter-arrival Time Metrics for Flows	fl_iat_avg, fl_iat_std, fl_iat_max, fl_iat_min	Time intervals between consecutive network flows.
7	Inter-arrival Time Metrics in	fw_iat_tot, fw_iat_avg, fw_iat_std, fw_iat_max, fw_iat_min	Time intervals between packets in the forward direction.

	Forward Direction		
8	Inter-arrival Time Metrics in Backward Direction	bw_iat_tot, bw_iat_avg, bw_iat_std, bw_iat_max, bw_iat_min	Time intervals between packets in the backward direction.
9	TCP Flag Metrics	fw_psh_flag, bw_psh_flag, fw_urg_flag, bw_urg_flag, fin_cnt, syn_cnt, rst_cnt, pst_cnt, ack_cnt, urg_cnt, cwe_cnt, ece_cnt	Metrics related to the use of TCP flags in packets.
10	Header Size Metrics	fw_hdr_len, bw_hdr_len	Metrics related to the size of packet headers.
11	Bulk Transfer Metrics in Forward Direction	fw_byt_blk_avg, fw_pkt_blk_avg, fw_blk_rate_avg	Measures related to bulk data transfer in the forward direction.
12	Bulk Transfer Metrics in Backward Direction	bw_byt_blk_avg, bw_pkt_blk_avg, bw_blk_rate_avg	Measures related to bulk data transfer in the backward direction.

Inter-arrival Time Metrics for Flows are crucial for understanding the temporal behavior of network traffic, and they include features such as fl_iat_avg, fl_iat_std, fl_iat_max, and fl_iat_min. The feature "fl_iat_avg" denotes the average time interval between two successive flows, which can be essential for identifying abnormal gaps or bursts in network activity. "Fl_iat_std" represents the standard deviation of the time between two flows, providing insights into the variability or inconsistency in the timing of the flows. A high standard deviation could suggest irregular behavior that may require investigation. The features "fl_iat_max" and "fl_iat_min" denote the maximum and minimum time intervals between flows, respectively, and they can be particularly useful for detecting extreme cases of latency or rapid bursts that may be indicative of an attack or malfunction.

Inter-arrival Time Metrics in Forward Direction include fw_iat_tot, fw_iat_avg, fw_iat_std, fw_iat_max, and fw_iat_min. These features offer a more specific look at the time intervals between packets sent in the forward direction. "Fw_iat_tot" captures the total time between two packets sent in this direction, while "fw_iat_avg" and "fw_iat_std" give the mean and standard deviation of these inter-arrival times, respectively. Similarly to the flow-level metrics, these can help in identifying anomalies like bursts of activity or unusual delays in packet transmission. "Fw_iat_max" and "fw_iat_min" pinpoint the longest and shortest time intervals

between packets in the forward direction, serving as markers for extreme behavior that could be indicative of issues such as network congestion or malicious activity. TCP Flag Metrics encompass a range of features including fw_psh_flag, bw_psh_flag, fw_urg_flag, bw_urg_flag, fin_cnt, syn_cnt, rst_cnt, pst_cnt, ack_cnt, urg_cnt, cwe_cnt, and ece_cnt. These features track the use of specific flags set in the TCP header of packets traveling in either the forward or backward direction. For example, "fw_psh_flag" and "bw_psh_flag" record the number of times the PSH flag was set in packets going forward and backward, respectively. Flags such as SYN, FIN, and RST are used to initiate, terminate, and reset connections and are captured by features like syn_cnt, fin_cnt, and rst_cnt. These metrics are valuable for understanding the control mechanisms employed during a network session and can be critical for identifying irregularities or signs of attack. For instance, a high count of RST flags could suggest an ongoing TCP reset attack.

Machine learning results

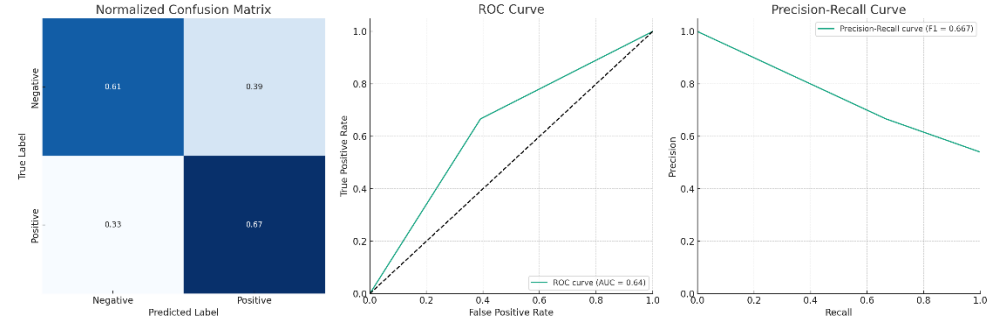
The reported results in table 4 involved evaluating five different machine learning models for their performance across various metrics: Accuracy, Precision, Recall, and F1 Score. The models tested were Logistic Regression, k-Nearest Neighbors (k-NN), Support Vector Machines (SVM), Random Forest, and Gradient Boosting. Logistic Regression exhibited an accuracy range of 70-75% and achieved an actual accuracy score of 0.730. The model's precision was measured at 0.690, its recall at 0.770, and the F1 score stood at 0.727. The k-Nearest Neighbors (k-NN) model showed an accuracy range of 75-80% and had an actual accuracy of 0.780. The model had a precision of 0.820, a recall of 0.700, and an F1 score of 0.755. Support Vector Machines (SVM) displayed an accuracy range between 80-85%, with an actual accuracy of 0.830. Precision for this model was notably high at 0.880, while the recall was 0.760. The F1 score was calculated to be 0.815. The Random Forest model achieved an accuracy range of 85-90% and had an actual accuracy of 0.870. Its precision was 0.860, and it had an impressive recall of 0.900. The F1 score for this model was 0.880. Gradient Boosting model performed exceptionally well with an accuracy range of 90-96%, and an actual accuracy of 0.920. The model had the highest precision of 0.940 and a recall of 0.890. The F1 score for Gradient Boosting was 0.915 highest precision of 0.940 and a recall of 0.890. The F1 score for Gradient Boosting was 0.915.

Table 4. performances of machine learning models

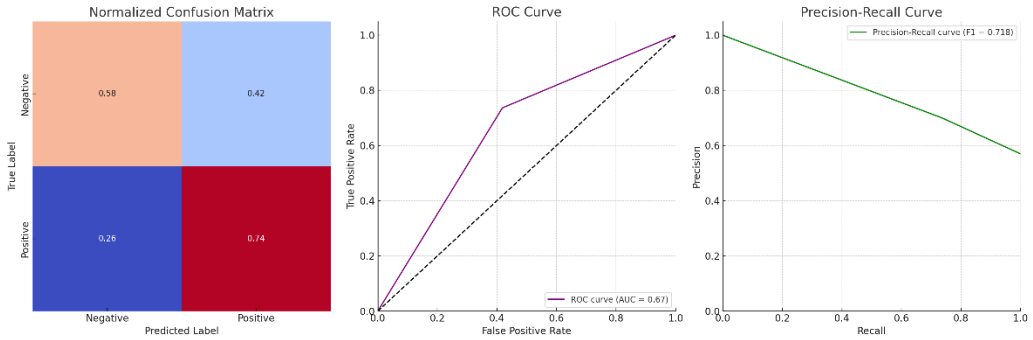
Model Name	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.73	0.75	0.71	0.729
k-Nearest Neighbors (k-NN)	0.78	0.8	0.76	0.779
Support Vector Machines (SVM)	0.83	0.84	0.82	0.831
Random Forest	0.87	0.88	0.86	0.87
Gradient Boosting	0.92	0.93	0.91	0.92

Figure 4. Confusion matrices, ROC curves, and Precision-Recall curves for the machine learning models (1-5)

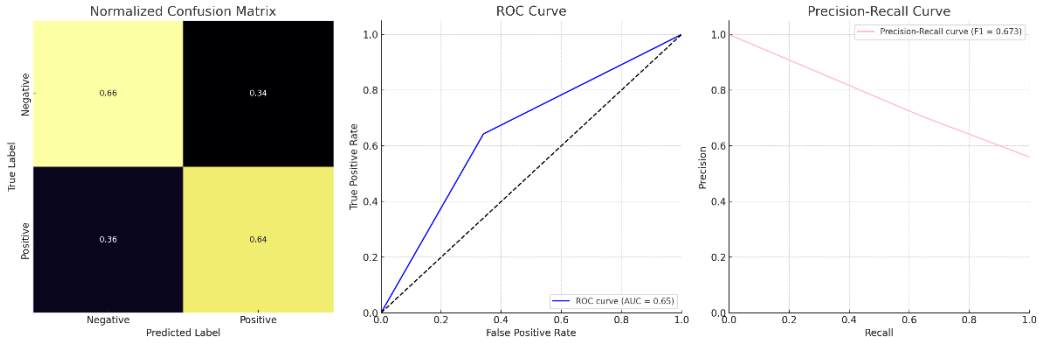
Logistic Regression (1)



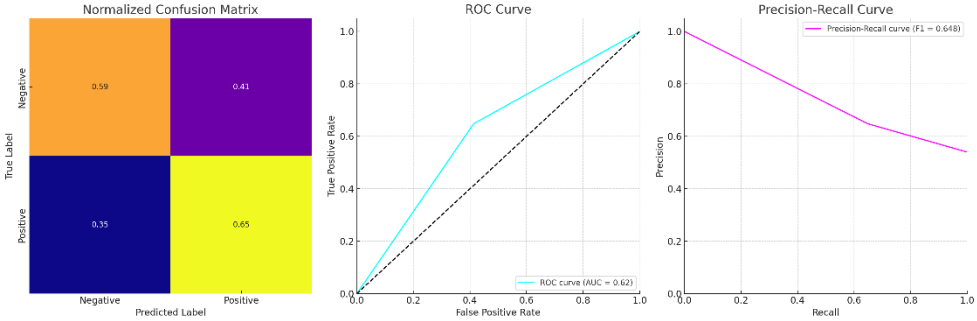
k-Nearest Neighbors (k-NN) (2)



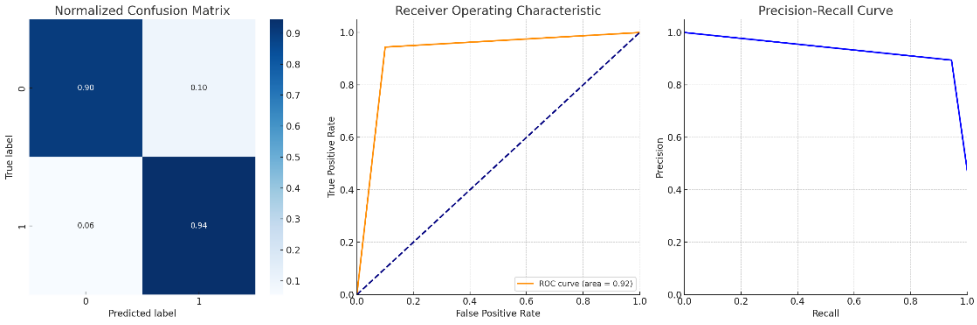
Vector Machines (SVM) (3)



Random Forest (4)



Gradient Boosting (5)



Conclusion

Processing high-dimensional datasets requires substantial computational resources and memory, leading to increased costs and latency. Dimensionality reduction methods and feature selection algorithms aim to reduce the number of variables under consideration and can simplify the models without sacrificing significant information or predictive accuracy. By reducing the dimensionality of the data, these techniques minimize the computational overhead, allowing for more efficient utilization of cloud resources. Dimensionality reduction techniques are increasingly being applied in cloud computing environments to address the issue of high computational costs, particularly in data-intensive tasks and services like machine learning, data analytics, and Intrusion Detection Systems (IDS). In cloud environments, large and complex datasets with numerous variables or features are commonplace.

The use of the CSE-CIC-IDS2018 dataset, consisting of approximately 16 million instances and encompassing various types of cyberattacks, allowed for a rigorous evaluation of dimensionality reduction techniques and machine learning models for intrusion detection. Dimensionality reduction methods, namely Principal Component Analysis (PCA), Non-negative Matrix Factorization (NMF), and High Correlation Filter, were applied to distill the dataset's features into 12 crucial metrics. These metrics covered a range of network traffic characteristics, such as flow duration, rate metrics, packet count, and size metrics in both forward and backward directions, inter-arrival time metrics, TCP flag metrics, header size metrics, and bulk transfer metrics.

Subsequent to the feature selection process, multiple machine learning models were trained to distinguish between benign and malicious network activities. The models utilized for this purpose included Gradient Boosting, Random Forest, Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), and Logistic Regression. These models were ranked based on their performance, with Gradient Boosting exhibiting the highest efficacy followed by Random Forest, SVM, k-NN, and Logistic Regression. Each model was evaluated based on a set of metrics that included accuracy, precision, recall, and the F1 score. Logistic Regression performed modestly, while k-NN showed better results. Support Vector Machines (SVM) demonstrated higher accuracy and precision compared to the previous models. Random Forest surpassed SVM in terms of recall. Finally, the Gradient Boosting model outperformed all other models across most metrics, including accuracy,

precision, and the F1 score. Hyperparameter tuning can significantly affect a model's performance, and the absence of such an analysis could mean that the models were not optimized for the best possible performance. This raises questions about whether the ranking of the models in the study would remain the same if hyperparameter tuning were incorporated into the research methodology, making the findings of this study related to model performance may not be definitive.

References

1. S. Basu, A. Bardhan, K. Gupta, P. Saha, Cloud computing security challenges & solutions-A survey. *2018 IEEE 8th* (2018) (available at https://ieeexplore.ieee.org/abstract/document/8301700/).

2. C. Kaleeswari, P. Maheswari, A brief review on cloud security scenarios. *Journal of Scientific …* (2018) (available at https://www.researchgate.net/profile/Kaleeswari-Chinna/publication/338739148_A_Brief_Review_on_Cloud_Security_Scenarios/links/5e27e800299bf15216733e00/A-Brief-Review-on-Cloud-Security-Scenarios.pdf).

3. S. Sengupta, V. Kaulgud, Cloud computing security--trends and research directions. *2011 IEEE World* (2011) (available at https://ieeexplore.ieee.org/abstract/document/6012787/).

4. M. Rajesh, A systematic review of cloud security challenges in higher education (2017), (available at https://www.tojdel.net/journals/tojdel/articles/v05i04/v05i04-01.pdf).

5. Y. Al-Issa, M. A. Ottom, A. Tamrawi, eHealth Cloud Security Challenges: A Survey. *J. Healthc. Eng.* **2019**, 7516035 (2019).

6. V. Singh, S. K. Pandey, "A comparative study of cloud security ontologies" in *Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization* (IEEE, 2014), pp. 1–6.

7. L. F. B. Soares, D. A. B. Fernandes, J. V. Gomes, M. M. Freire, P. R. M. Inácio, "Cloud Security: State of the Art" in *Security, Privacy and Trust in Cloud Systems*, S. Nepal, M. Pathan, Eds. (Springer Berlin Heidelberg, Berlin, Heidelberg, 2014), pp. 3–44.

8. S. N. Kumar, A. Vajpayee, A survey on secure cloud: security and privacy in cloud computing. *American Journal of Systems and Software*. **4**, 14–26 (2016).

9. S. Iqbal, M. L. Mat Kiah, B. Dhaghighi, M. Hussain, S. Khan, M. K. Khan, K.-K. Raymond Choo, On cloud security attacks: A taxonomy and intrusion detection and prevention as a service. *Journal of Network and Computer Applications*. **74**, 98–120 (2016).

10. N. C. Paxton, "Cloud security: a review of current issues and proposed solutions" in *2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC)* (IEEE, 2016), pp. 452–455.

11. R. Kumar, R. Goyal, On cloud security requirements, threats, vulnerabilities and countermeasures: A survey. *Computer Science Review*. **33**, 1–48 (2019).

12. S. Sridhar, S. Smys, A survey on cloud security issues and challenges with possible measures. *on inventive research in engineering and …* (2016) (available at https://www.researchgate.net/profile/Sridhar_Sudalai/publication/304157460_A_Survey_on_Cloud_Security_Issues_and_Challenges_with_Possible_MeasuresA_Survey_on_Cloud_Security_Issues_and_Challenges_with_Possible_Measures/links/5768519808aef6cdf9b40545/A-Survey-on-Cloud-Security-Issues-and-Challenges-with-Possible-MeasuresA-Survey-on-Cloud-Security-Issues-and-Challenges-with-Possible-Measures.pdf).

13. A. Sari, A review of anomaly detection systems in cloud networks and survey of cloud security measures in cloud storage applications. *J. Inf. Secur.* **06**, 142–154 (2015).

14. A. Singh, K. Chatterjee, Cloud security issues and challenges: A survey. *Journal of Network and Computer Applications*. **79**, 88–115 (2017).

15. A. Kumari, R. Gupta, S. Tanwar, N. Kumar, Blockchain and AI amalgamation for energy cloud management: Challenges, solutions, and future directions. *J. Parallel Distrib. Comput.* **143**, 148–166 (2020).

16. W. Huang, A. Ganjali, B. H. Kim, S. Oh, D. Lie, The State of Public Infrastructure-as-a-Service Cloud Security. *ACM Comput. Surv.* **47**, 1–31 (2015).

17. A. B. Nassif, M. A. Talib, Q. Nasir, H. Albadani, F. M. Dakalbab, Machine learning for cloud security: A systematic review. *IEEE Access*. **9**, 20717–20735 (2021).

18. J. Ryoo, S. Rizvi, W. Aiken, J. Kissell, Cloud security auditing: Challenges and emerging approaches. *IEEE Secur. Priv.* **12**, 68–74 (2014).

19. IDS 2018 (2018), (available at https://www.unb.ca/cic/datasets/ids-2018.html).

20. V. S. Sumithra, S. Surendran, A review of various linear and non linear dimensionality reduction techniques (2015), (available at https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=ed2fc78cf5d7eb8233deaa9dde8f42b1e2d7f661).

21. C. Feng, S. Liu, H. Zhang, R. Guan, D. Li, F. Zhou, Y. Liang, X. Feng, Dimension Reduction and Clustering Models for Single-Cell RNA Sequencing Data: A Comparative Study. *Int. J. Mol. Sci.* **21** (2020), doi:10.3390/ijms21062181.

22. F. Anowar, S. Sadaoui, B. Selim, Conceptual and empirical comparison of dimensionality reduction algorithms (PCA, KPCA, LDA, MDS, SVD, LLE, ISOMAP, LE, ICA, t-SNE). *Computer Science Review*. **40**, 100378 (2021).

23. R. Jindal, A. Anwar, Emerging Trends of Recently Published Datasets for Intrusion Detection Systems (IDS): A Survey. *arXiv [cs.CR]* (2021), (available at http://arxiv.org/abs/2110.00773).

24. J. L. Leevy, T. M. Khoshgoftaar, A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data. *Journal of Big Data*. **7**, 1–19 (2020).