


Distributed Denial of Service Attacks Detection and Classification using Machine Learning in Cloud Environment

Hanan Hafiz

(Department of Computer Science, Taibah University, Madinah, Saudi Arabia
tu4359252@taibahu.edu.sa)

Maher Alharby

(Department of Cybersecurity, Taibah University, Madinah, Saudi Arabia
 <https://orcid.org/0000-0002-9278-8258>, mharby@taibahu.edu.sa)

Abstract: The rapid adoption of cloud computing has revolutionized how businesses and consumers access and utilize resources, offering scalability, flexibility, and cost effectiveness. However, this increased reliance on cloud services has also led to a rise in Distributed Denial of Service (DDoS) attacks, which can severely impact the availability and performance of these services. This study aims to address the critical need for effective detection and classification of DDoS attacks in cloud environments using machine learning techniques. We conducted binary and multiclass classification experiments using the CICDDoS2019 dataset, focusing on three specific types of attacks. Four machine learning models, namely Random Forest, K Nearest Neighbor, Naïve Bayes, and Logistic Regression, were implemented in a Kaggle notebook using Python. Feature selection techniques, including Chi square and Principal Component Analysis, were employed to identify the most relevant features, while the oversampling technique was used to handle imbalanced data. The experiments yielded impressive results, with Random Forest and K-Nearest Neighbor achieving the highest accuracy rates of 100% and 99.72% in binary classification, and 100% and 99.66% in multiclass classification, respectively. The study also measured training and testing times, along with other performance metrics. These findings highlight the effectiveness of machine learning approaches in tackling cloud based detection challenges while ensuring computational efficiency tailored for dynamic cloud environments.

Keywords: Machine Learning, DDoS Attacks, DoS attacks, Cloud Security

Categories: H.3.1, H.3.2, H.3.3, H.3.7, H.5.1

DOI: 10.3897/jucs.140733

1 Introduction

Cloud computing has revolutionized how computer resources are accessed and utilized, offering scalability, flexibility, and cost efficiency. It is widely adopted in various industries, including healthcare, banking, education, entertainment, and government. The proliferation of cloud-based platforms and applications has opened up global cooperation and innovation opportunities. However, the security of cloud computing is of the utmost importance, as it ensures the integrity, confidentiality, and availability of sensitive data [Furht et al. 2010]. The unique characteristics of cloud environments, including multi-tenancy, virtualization layers, and dynamic resource allocation, introduce distinct security challenges that differ fundamentally from traditional network infrastructures [Al-Jahdali 2017]. In cloud computing architectures, multiple virtual machines and containers

share physical infrastructure, creating complex network topologies where traditional perimeter-based security approaches are insufficient for comprehensive threat detection.

Distributed Denial of Service (DDoS) attacks pose a significant threat to the security and stability of cloud environments, causing excessive traffic and costly consequences. They can disrupt legitimate user access, leading to lost productivity and unavailability. Sensitive industries can suffer revenue losses and service disruptions, affecting financial transactions, regulatory compliance, and healthcare. DDoS attacks can also damage a company's reputation and require significant resources for recovery [Daffu and Kaur 2016]. In cloud environments, these attacks are particularly challenging to detect and mitigate due to the abstraction of network traffic through virtualization layers and the complexity of distinguishing malicious traffic from legitimate multi-tenant communications [AlJahdali 2017]. Furthermore, the elastic nature of cloud resources means that attackers can potentially leverage cloud infrastructure to launch attacks.

Numerous DDoS attacks have hit cloud environments worldwide, causing severe service disruptions for users and companies. According to the 2020 Annual DDoS Threat Report [NexusGuard 2020], DDoS attacks increased significantly from Q1 2019 to Q1 2020, with a record-breaking 809 million packets per second attack launched on a European bank in 2020, using a packet per second (PPS) method to overwhelm network hardware and applications [Olenick 2020]. Amazon Web Services (AWS) had the most significant recorded DDoS attack ever in February 2020 [Shield 2020]. An hour of downtime of IT services can affect companies and cost between \$300,000 and \$1,000,000, according to Information Technology Intelligence Consulting (ITIC) [DiDio 2019]. DDoS attacks continue to evolve, posing significant challenges to cyber network security, emphasizing the need for protection in digital security [Yoachimik and Pacheco 2024]. In Q1 2024, 4.5 million DDoS attacks were discovered and addressed, an increase of 50% from the previous year.

Various research studies have attempted to identify and detect DDoS attacks in cloud environments. However, a review of the recent literature using the CIC-DDoS 2019 dataset revealed several limitations in existing studies. Firstly, there is a lack of consistency in the approaches used by these studies, particularly regarding model selection, preprocessing techniques, and evaluation metrics. This makes it difficult to compare and evaluate the effectiveness of different methods. Secondly, many studies fail to address the issue of data imbalance, which can significantly impact the performance of the models. An imbalanced dataset can lead to biased results, with the model favoring the majority class and performing poorly on the minority class [Ramyachitra and Manikandan 2014]. Furthermore, most research studies, such as [Seifousadati et al. 2021, Akgun et al. 2022, Ramzan et al. 2023, Abbas and Almhanna 2021, Parfenov et al. 2020, Organization 2024, Becerra-Suarez et al. 2024], do not consider measuring training and testing times when evaluating machine learning models, which is crucial for evaluating their practicality and scalability in real-world scenarios. This timing consideration is particularly critical in cloud environments where detection systems must operate within strict service level agreements and support dynamic auto-scaling mechanisms.

To address these limitations, this study aims to develop machine learning models to identify and detect DDoS attacks in cloud environments using the CIC-DDoS 2019 dataset. It specifically addresses cloud-specific detection challenges, such as multi-tenant traffic patterns and the computational efficiency required in virtualized environments. We selected the CIC-DDoS2019 dataset because it covers recent DDoS attack types and has high-quality labeled traffic data. Unlike other datasets (e.g., CAIDA and DARPA [Najar and others 2024, Zy et al. 2024]) that focused on a limited set of attack vectors, the CIC-DDoS2019 dataset includes realistic DDoS attacks targeting application-layer

protocols, such as HTTP and HTTPS [Sharafaldin et al. 2019], making it suitable for evaluating detection algorithms in virtualized cloud environments.

The main contributions of this study are as follows:

- Conducting a binary classification experiment to detect DDoS attacks using four machine learning models: Random Forest, K-Nearest Neighbor, Naïve Bayes, and Logistic Regression. This experiment will provide insights into the effectiveness of these models in distinguishing between benign traffic and DDoS attacks when trained and tested on the CIC-DDoS 2019 dataset. The evaluation considers cloud-specific factors such as multi-tenant traffic discrimination and virtualized network monitoring challenges.
- Performing a multiclass classification experiment to classify the specific type of DDoS attack using the same four machine learning models and the CIC-DDoS 2019 dataset. This experiment will evaluate the models' ability to accurately identify different types of DDoS attacks, such as NTP, UDP-Lag, and SYN floods. This classification capability is essential for implementing targeted mitigation strategies across different virtual network segments and tenant configurations.
- Employing two distinct feature selection techniques, Chi-square and Principal Component Analysis (PCA), when preprocessing the CIC-DDoS 2019 dataset. By comparing the results obtained using these techniques, the study aims to identify the most effective method for selecting relevant features and improving model performance. The evaluation considers the consistency of these methods across varying virtualized environments and network topologies typical in cloud deployments.
- Measuring the training and testing times for the machine learning models, along with other relevant metrics such as accuracy, precision, recall, and F1-score, when trained and evaluated on the CIC-DDoS 2019 dataset. These timing measurements are analyzed in the context of cloud-specific requirements for real-time detection and auto-scaling defense mechanisms.
- Addressing the issue of data imbalance in the CIC-DDoS 2019 dataset by employing the oversampling method to ensure balanced datasets for both binary and multiclass classification experiments. This approach will help mitigate the potential bias introduced by imbalanced data and ensure that the models are trained and evaluated on representative samples of each class. This is particularly important in cloud environments where legitimate traffic volumes may vary dramatically between different tenants and services.

The main findings of this study, which investigated the performance of four machine learning algorithms (Random Forest, K-Nearest Neighbors, Naive Bayes, and Logistic Regression) in binary and multiclass experiments using Chi-square and PCA feature selection techniques, reveal several significant insights. Random Forest and K-Nearest Neighbors consistently achieved the highest accuracy scores, ranging from 99% to 100%. However, Naive Bayes and Logistic Regression initially demonstrated lower accuracy scores. However, their performance significantly improved after applying hyperparameter tuning using Grid Search, but at the cost of increased training times, especially for Logistic Regression in the multiclass experiment. Interestingly, the choice between Chi-square and PCA feature selection techniques did not substantially influence the accuracy or performance time of the machine learning algorithms. Furthermore, the multiclass experiment generally exhibited higher training and testing times than

the binary experiment, with Random Forest showing the highest training times and K-Nearest Neighbors showing the highest testing times. These findings demonstrate the effectiveness of machine learning approaches in addressing cloud-specific detection challenges while highlighting important trade-offs between accuracy and computational efficiency in virtualized environments.

This study is organized into six sections. Section 2 provides background information on cloud security and machine learning techniques. Section 3 reviews related work and highlights the main contributions of this study compared to previous research studies. Section 4 explains the methodology adopted in this research. Section 5 details the experimental results, including the development environment utilized. Section 6 concludes the article and describes future work.

2 Background

This section discusses the essential topics and general background on cloud computing security, distributed denial-of-service attacks, and machine learning algorithms used in this study.

2.1 Cloud Computing

Cloud computing is a large-scale distributed computing framework that integrates grid computing, virtualization, and service orientation, offering a method for managing IT resources. It allows users to access platforms, software, storage services, and processing power via the Internet [Alharby 2024]. Cloud computing is a model that allows on-demand network access to shared computing resources, offering five key features: on-demand service, fast elasticity, resource pooling, comprehensive network connectivity, and self-service. It allows users to provision resources automatically while providers aggregate resources on demand to serve other consumers [Mell et al. 2011]. Cloud computing consists of three service models: software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS) [Alharby 2024]. These models are enabled by the underlying layers of the cloud architecture, which are application, platform, infrastructure, and hardware/data center. Each model can be developed independently and utilized from the lower layers [Zhang et al. 2010]. Companies use various cloud computing models to reduce costs and improve business operations. These models include public, private, community, virtual private, and hybrid cloud. Third-party providers manage public clouds, while a company manages private clouds. Hybrid clouds combine public and private clouds, combining internal compliance, on-premises resources, and external capacity. Virtual private clouds use public cloud resources, and community clouds are semi-private clouds used by specific tenants [Ahmed and Alexandrov 2011].

2.1.1 Cloud computing security

The vulnerabilities and risks associated with cloud computing, as described in [Rachandran et al. 2017], should not be overlooked. They include vulnerabilities in shared technology, data breaches, account or service traffic hijacking, denial of service (DoS) attacks, malicious insider threats, injection vulnerabilities, malicious insiders at the cloud provider, malicious insiders at the cloud provider, availability issues, and external attacks via the network. These risks, if not addressed immediately, can have severe consequences.

The dangers and outcomes of DDoS attacks within cloud computing settings are discussed in [Potluri et al. 2020]. These dangers include resource downtime, financial loss, QoS deficiencies, damage to business reputation, and intense traffic assaults. DDoS attacks can make cloud resources inaccessible or fail, resulting in service disruptions for authorized users. The downtime from DDoS attacks can lead to substantial business and revenue losses for cloud service providers and their clients. Furthermore, DDoS attacks can degrade the quality of service (QoS) that the cloud provider provides by breaching service-level agreements. Successful DDoS attacks can degrade the reputation of the cloud service provider, affecting customer trust.

Moreover, substantial traffic floods generated by DDoS attacks consume excessive bandwidth and computing resources, disrupting normal operations. The article emphasizes that cloud computing is highly susceptible to DDoS attacks due to its dynamic resource provisioning, distributed paradigm, heterogeneity, and virtualized nature, which can have severe complications. It emphasizes the need for improved migration policies, collaborative frameworks, auto-scaling algorithms, and dedicated DDoS mitigation mechanisms to manage these risks and address attacks in cloud environments.

2.1.2 Distributed Denial of Service attack

DoS and DDoS are two types of cyberattacks that share the same objective of disrupting or degrading a service by exhausting the resources necessary to provide it. These attacks often target network bandwidth, server connection buffers, CPU cycles, and other resources. In a typical DoS attack, the attacker attempts to overwhelm the target's resources from a single computer. However, DDoS attacks are more powerful and sophisticated, involving many attacking hosts across the Internet, making them harder to prevent. In DDoS attacks, the attacker first compromises and controls numerous computers across the Internet, creating an attack network or botnet. The attacker then instructs all the compromised bots to send packets to the target, overwhelming it simultaneously. DDoS attacks can force the victim to significantly downgrade service performance or stop providing the service by flooding it with an enormous, aggregated attack volume that exceeds the victim's resources. Although individual DoS attacks can sometimes be addressed through better system security, DDoS attacks that coordinate distributed bots are much more complex and challenging to prevent and mitigate. In summary, a DoS attack originates from a single source, while a DDoS attack is a more powerful attack that comes from multiple sources across the Internet [Gu and Liu 2007].

A DoS attack aims to overwhelm a system with packets using one computer and an Internet connection. The objective is to consume CPU, memory, disk space, or bandwidth resources, rendering the system inaccessible to legitimate users [Rudman 2014]. Alternatively, a DDoS attack exploits compromised computers to assault a target system. The collection of compromised computers is known as a botnet. To establish a botnet, an attacker identifies a vulnerable computer on the Internet and installs a malicious program. This computer becomes the DDoS master and looks for other vulnerable computers on the network. Once identified, malware, called zombies, is installed on these computers. When the attacker wants to execute a DDoS attack, the DDoS master is instructed to signal the zombie computers to attack the victim. A DDoS attack has an advantage over a conventional DoS attack, since the botnet can generate more attack traffic than a single machine [Rudman 2014].

Below is an explanation of the three attack types used in this study: SYN, Network Time Protocol (NTP), and UDP flood.

TCP SYN flood: This attack sends numerous SYN packets, using either real or spoofed IP addresses, to establish connections with a target server. The server responds with a SYN/ACK message, but does not receive an ACK, leading to incomplete connections. This saturation of SYN packets makes it difficult for legitimate users to access the server, and it is difficult to defend due to the use of varied IP addresses. In cloud environments, SYN flood attacks are difficult as they can target virtual machines, load balancers, and API gateways [Rudman 2014, Eddy 2006].

Network Time Protocol (NTP): This protocol synchronizes time across computer networks using dedicated time servers, playing a crucial role in upholding national time standards. However, NTP is susceptible to exploitation through reflection and amplification attacks, in which adversaries misuse multiple NTP servers to significantly amplify malicious traffic. This vulnerability is exacerbated by the protocol's inherently distributed architecture, which poses significant challenges to implementing uniform traffic filtering mechanisms without risking the disruption of legitimate NTP operations [Mills 1991].

UDP-Lag: In this type of attack, the attacker floods a victim's machine with UDP packets directed at random ports, overwhelming the bandwidth and causing performance issues. UDP-Lag is commonly used in online gaming to disrupt client-server connections, causing slowdowns for other players and granting an unfair advantage by consuming bandwidth through mechanisms such as lag switches or specialized software. In cloud services, UDP-Lag attacks adversely affect user experience. Effective cloud-based detection requires monitoring latency patterns and UDP traffic distribution, which makes it difficult to distinguish between genuine congestion and intentional attacks [Sharafaldin et al. 2019].

Detecting DDoS attacks in cloud environments involves addressing various infrastructure challenges. Effective detection methods include traffic analysis in virtualized networks, which requires custom monitoring solutions for virtual switches and hypervisors to inspect traffic without causing performance bottlenecks [Watada et al. 2019]. Resource usage anomaly detection focuses on monitoring resource consumption patterns on virtual machines to identify unexpected spikes that may indicate potential DDoS attacks [Ntambu and Adeshina 2021]. Distributed detection frameworks use distributed sensors that share information to correlate events across cloud infrastructures, helping to identify coordinated attacks [Ficco 2013]. In addition, machine learning techniques for dynamic environments leverage advanced methods that adapt detection parameters as resources are scaled [Sundaramurthy et al. 2025].

2.2 Machine Learning Algorithms

This section presents an overview of the four machine learning algorithms utilized in this study: Random Forest, Naïve Bayes, K-Nearest Neighbors, and Logistic Regression.

Random Forest (RF): Combines multiple decision trees trained on different data subsets to improve the accuracy of the classification. This ensemble approach reduces overfitting while maintaining high predictive performance [Speiser et al. 2019].

K-Nearest Neighbors (KNN): Classifies data points based on the majority class of their k nearest neighbors. Despite its simplicity, KNN performs well when properly tuned. Its effectiveness depends primarily on the distance metric and the number of neighbors (k) considered [Zhang 2016].

Naïve Bayes (NB): Applies Bayes' theorem with an assumption of feature independence. Although this assumption rarely holds in practice, the algorithm remains effective for many classification tasks, particularly with limited training data [Rish and others 2001].

Logistic Regression (LR): Estimates the probability of binary outcomes by modeling the relationship between features and target variables using the logistic function. Its interpretability and efficiency make it widely applicable in all domains [Ramzan et al. 2023].

3 Related Work

This section reviews existing research studies for DDoS attack detection and classification, focusing on traditional machine learning, deep learning, and hybrid techniques.

3.1 Traditional Machine Learning Approaches for DDoS Detection

Ma et al. proposed a framework (FAMS) that automatically selects the best models and features with high prediction accuracy, short prediction time, and strong generalization capabilities for DDoS attack detection. FAMS consists of four phases: feature selection (FS), model selection (MS), random forest (RF) optimization, and data preprocessing. Using a natural selection technique based on fitness values, FAMS iteratively chooses the best features and models. The datasets used are CIC-DoS2016, CIC-IDS2017, CICIDS2018, and CICDDoS2019, achieving 99% accuracy with random forest [Ma et al. 2023].

Wankhede et al. developed a machine learning and neural network-based algorithm to detect DoS attacks. Their objective was to determine the optimal parameter values to maximize their model's accuracy compared to detection models in other similar studies. They achieved an accuracy of 99.95% using a Random Forest algorithm with 500 trees and a training dataset that accounted for 50% of the CIC IDS 2017 dataset [Wankhede and Kshirsagar 2018].

Wani et al. proposed the use of machine learning algorithms to detect and analyze DDoS attacks in a cloud environment. Using the Tor Hammer tool, they utilized a botnet and a computer running Kali Linux to initiate an attack against the ownCloud platform. The attack was detected in the cloud using SNORT, and a dataset was extracted from the traffic. The SNORT-generated dataset's class label consisted of normal and suspicious values. Three classification algorithms, namely the Support Vector Machine (SVM), Random Forest (RF), and Naïve Bayes (NB), were employed to detect attacks. The most remarkable detection rate was 99.8% for SVM, followed by 99.3% for Random Forest and 86.0% for NB. Consequently, it can be concluded that among the three methods, SVM had the optimal detection rate [Wani et al. 2019].

Hoon et al. examined and analyzed various supervised and unsupervised machine learning algorithms for DDoS detection. They determined the ideal hyperparameter values that could increase the algorithmic accuracy. The primary contribution of their study is the definition of a new parameter that serves as a threshold for more accurate decision-making during training. Using the NSL-KDD dataset, they found that supervised algorithms such as Random Forest, Gradient Boost, and Naïve Bayes perform better in terms of accuracy and training time [Hoon et al. 2018].

Sumathi and Karthikeyan conducted an extensive analysis of various hybrid and traditional machine learning techniques. By applying these methods to the KDDcup99

and DARPF datasets, they discovered that Fuzzy C-Means and Decision Trees yielded superior results compared to the other approaches. In particular, the fuzzy C-Mean algorithm exhibited exceptional performance, achieving a detection time of 0.15 seconds and an impressive accuracy rate of 98.7% in identifying DDoS activity [Sumathi and Karthikeyan 2018].

Wehbi et al. proposed three novel methods employing the Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Label Propagation Algorithm (LPA), and Quadratic Discriminant Analysis (QDA) algorithms. These methods were evaluated using the 1999 DARPA Intrusion Detection Dataset, CAIDA, and a simulated environment. In their study, they introduced a seven-layer sequential model for DDoS detection and developed a unique classifier for feature extraction. Additionally, they proposed two new criteria to prevent machine learning-based DDoS detection systems from incorrectly classifying legitimate traffic as DDoS traffic. Their results showed that the Random Forest algorithm achieved the highest accuracy of 99.99% among the three proposed techniques, all of which demonstrated impressive performance [Wehbi et al. 2019].

Polat et al. proposed a DDoS detection system that leverages machine learning and data mining techniques. Their study utilized the KDDCUP99 dataset to evaluate various machine learning algorithms, comparing their speed and accuracy. Through empirical research, they identified optimal values for selected hyperparameters, such as the Cross-Validation Ratio (10) and the training model size (66% of the dataset). Their findings suggest that the J48 algorithm exhibits the highest success rate in accurately detecting DDoS attacks [Polat et al. 2019].

Khedr et al. introduced FMDADM, a multi-layer system designed for detecting and preventing DDoS attacks in stateful SDN-based Internet of Things networks. The system was trained and tested using the Edge-IIoTset dataset, and categorical values were transformed into numerical values using the Label encoding library. Six machine learning models, including the Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Gaussian Naïve Bayes (GNB), Beacon-Less Routing (BLR), Decision Tree (DT), and Random Forest (RF), were employed. The Random Forest model achieved the highest accuracy at 99.79% [Khedr et al. 2023].

Using the CICDDoS2019 dataset, Ibrahim et al. proposed utilizing three machine learning algorithms, namely Naïve Bayes, Random Forest, and Support Vector Machine, to identify Port Scan attacks. In order to efficiently execute classification and identify the most relevant features, this study employs RStudio tools to implement the Boruta, Forward and Backward, and Variable Significance algorithms. Through the Variable Importance algorithm, the features are narrowed down to just 11 out of 79. At the same time, the Boruta method reduces the total number of features to 56, and the Forward and Backward method reduces it to 59. The paper assesses the performance of the classification algorithms, and the results indicate that the Random Forest with forward and backward feature selection yields the best results, achieving 100% accuracy [Ibrahim and Mohammed 2022]. This paper exclusively employed binary detection without implementing any balancing methods.

Also using the CICDDoS2019 dataset, Seifousadati et al. proposed a machine learning approach to identify DDoS attacks targeting Internet of Things (IoT) devices. They employed various algorithms and utilized correlation methods to eliminate irrelevant features. The method's effectiveness was measured using training time, F1 score, and accuracy. The findings revealed that AdaBoost and XGBoost attained an F1 score of 100% accuracy [Seifousadati et al. 2021]. The paper solely used binary detection and did not utilize any balancing techniques. There is no mention in the paper of how the hyperparameters were tuned.

Ali et al. studied the detection of DDoS attacks within software-defined networking (SDN) environments. Two datasets, namely CICDDoS2019 and CICIDS2017, were utilized in their research. The study involved the evaluation of various machine learning algorithms to determine their efficacy in identifying DDoS attacks. The tested models included the Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and Decision Tree (DT). The researchers applied preprocessing and feature selection techniques. Model accuracy was assessed, with SVM emerging as the most effective algorithm, achieving 97.81% training accuracy and 95.57% prediction accuracy with the CICDDoS2019 dataset [Ali et al. 2023]. The paper does not include feature selection or balancing methods.

Becerra-Suarez et al. conducted a study to enhance the detection of DDoS attacks by applying machine learning techniques and data processing methods. The research used the CICDDoS2019 dataset, and the authors utilized PCA feature selection techniques. They evaluated the performance of six machine learning algorithms using various accuracy metrics. The classification approach used was binary classification, with the RF classifier yielding the most impressive results, achieving an accuracy of 99.97% [Becerra-Suarez et al. 2024]. The paper focuses only on binary classification and uses no balancing methods.

Alqarni introduced an ensemble technique to identify DDoS attacks in a cloud environment using the CICDDoS2019 dataset. This method combines four machine learning models, namely NB, DT, SVM, and K-NN. Under-sampling and over-sampling techniques were employed to address imbalanced data. Additionally, the Chi-squared method was utilized to extract the top 15 features. The performance of the ensemble approach achieved an accuracy score of 98.02% [Alqarni 2022]. The dataset sample is small, with only 23,000 records, and needs more comparison with recent studies. Additionally, the paper only employs binary classification.

Abbas and Almhanna developed a machine learning approach to identify DDoS attacks using the CICDDoS2019 dataset. They employed two machine learning techniques, namely RF and NB, to classify four types of attacks. The PCA feature selection method was utilized for feature reduction, and five performance measures were considered. The best result was achieved using RF, with an accuracy of 99.97% [Abbas and Almhanna 2021]. This paper uses only the RF and NB classifiers. No details are provided about the training/test split used. Furthermore, it only involves multiclass classification.

3.2 Deep Learning Approaches for DDoS Detection

Cil et al. proposed a method for detecting and categorizing various DDoS attacks within network traffic using a deep neural network (DNN) model. The DNN structure included an input layer, three hidden layers with 50 units each using the sigmoid activation function, and an output layer activated by softmax. During training, the AdaMax optimization method and binary cross-entropy loss were applied. The authors utilized the CICDDoS2019 dataset and pruned 17 unnecessary features from the initial 86, leaving 69 features for the DNN model. The dataset was divided into testing and training sets (80% for training and 20% for testing). The results demonstrate exceptional performance, with an accuracy of 99.99% [Cil et al. 2021]. However, the paper lacks details on model implementation, hyperparameter tuning, and computational resources, making it difficult to reproduce and validate the findings. Additionally, it exclusively employed binary detection without implementing any balancing methods.

Akgun et al. proposed a new intrusion detection system (IDS) to detect DDoS attacks using deep learning models. The authors utilized the CIC-DDoS2019 dataset and applied

the Information Gain Attribute to reduce the feature set from 88 to 40 to extract essential characteristics. They employed random subset selection for data balancing, ensuring an equal number of records from benign and attack classes. The study utilized Long Short-Term Memory (LSTM), Convolutional Neural Networks (CNN), and Dense Neural Networks (DNN). The proposed model outperformed other models analyzed, achieving the highest multiclass accuracy (99.3%) and binary accuracy (99.99%) [Akgun et al. 2022].

Ramzan et al. employed advanced deep learning models to detect DDoS attacks using the updated CICDDoS2019 dataset. Their approach included utilizing recurrent neural network (RNN), gradient recurrent unit (GRU), and long short-term memory (LSTM) models. The findings were validated using the CICIDS2017 dataset. Additionally, the authors utilized the extra tree classifier method to extract the top 20 essential features and employed the label encoder to transform categorical data into a numerical format. The proposed RNN model achieved a remarkable accuracy rate of 99.99% [Ramzan et al. 2023]. However, there is ambiguity in the modified hyperparameter values and no attempt to balance the data.

3.3 Hybrid and Ensemble Approaches for DDoS Detection

Alghazzawi et al. introduced a hybrid approach that integrates Convolutional Neural Networks (CNN) and Bidirectional Long Short-Term Memory (BiLSTM) along with a Chi-square feature selection method for identifying various types of attacks using the CICDDoS2019 dataset. The authors evaluated the model's performance, achieving an accuracy of up to 94.52% [Alghazzawi et al. 2021]. This paper used only binary classification and did not employ any balancing methods.

Parfenov et al. sought to detect DDoS attacks on IoT networks using ensemble machine learning techniques. The researchers utilized the CICDDoS2019 dataset and employed three ensemble machine learning models (Gradient Boosting, AdaBoost, and Cat Boost) to classify the dataset. They also used the Extra Trees Classifier method to identify the 25 most important features. Moreover, they balanced the dataset using the ADASYN method to avoid skewed results. They employed precision, recall, and F1-score metrics to assess the models' performance, conducting both binary and multiclass classifications. The results revealed that the Cat Boost model was the most effective [Parfenov et al. 2020]. However, limited information is provided on the data preprocessing steps applied to the CICDDoS2019 dataset.

3.4 Research Gaps and Our Contributions

Previous studies exploring DDoS attack detection and classification have employed various methodologies, including traditional machine learning, deep learning, and hybrid approaches. Despite the diversity of techniques utilized, several limitations exist in the current literature. Firstly, there is a lack of consistency in the approaches used by these studies, particularly regarding model selection, preprocessing techniques, and evaluation metrics, making it difficult to compare and evaluate the effectiveness of different methods. Secondly, many studies fail to address the issue of data imbalance, which can significantly impact the performance of the models. An imbalanced dataset can lead to biased results, with the model favoring the majority class and performing poorly on the minority class. Furthermore, most research studies do not consider the measurement of training and testing times when evaluating machine learning models, which is crucial to assessing their practicality and scalability in real-world scenarios.

Paper	Year	Classification Algorithms	Features Selection Methods	Data Balancing Technique	Classification Type	Performance Measures
[Ibrahim and Mohammed 2022]	2022	SVM, NB, RF	Boruta, Forward and Backward, Variable Significance Algorithms	-	Binary only	Accuracy, Precision, Recall, F1-Score, Specificity, Classification Achievement
[Cil et al. 2021]	2021	DNN	Merged in the DNN model's structure	-	Binary only	Accuracy, Precision, Recall, F1-Score
[Alghazzawi et al. 2021]	2021	Hybrid (CNN and BiLSTM)	Chi-squared	-	Binary only	Accuracy, Precision, Recall, F1-Score
[Seifousadati et al. 2021]	2021	NB, SVM, AdaBoost, XGBoost, KNN, RF	Correlation (feature importance)	-	Binary only	Accuracy, F1-Score, ROC Curve, Training time
[Akgun et al. 2022]	2022	DNN, CNN, LSTM	Information Gain Attribute	Random subset selection	Binary and Multi	Accuracy only
[Ramzan et al. 2023]	2023	RNN, LSTM, GRU	Extra tree classifier	-	Binary and Multi	Accuracy, F1-Score, ROC Curve, Execution time
[Alqarni 2022]	2021	NB, DT, SVM, K-NN, Majority vote	Chi-squared	Under- and over-sampling	Binary only	Accuracy, Recall, Specificity, Execution time
[Abbas and Almhanna 2021]	2021	RF, NB	PCA	-	Multi	Accuracy, Precision, Recall, F1-Score, False alarm rate
[Parfenov et al. 2020]	2020	Gradient Boosting, AdaBoost, CatBoost	Extra Trees Classifier	Adaptive Synthetic Sampling	Binary and Multi	Precision, Recall, F1-Score
[Ali et al. 2023]	2023	SVM, KNN, DT, MLP, CNN	-	-	Binary and Multi	Accuracy, Precision, F1-Score, MCC, Testing time, Training time
[Becerra-Suarez et al. 2024]	2024	RF, DT, ADA, XGB, MTP, DNN	PCA	-	Binary only	Accuracy, Precision, Recall, F1-Score, AUC
This study	2024	RF, NB, K-NN, LR	Chi-squared, PCA	Oversampling	Binary and Multi	Accuracy, Precision, Recall, F1-Score, Testing time, Training time

Table 1: Comparison of the proposed machine learning models with state-of-the-art approaches for DDoS attack detection and classification.

To address these limitations, our work contributes several significant improvements over existing literature:

1. **Comprehensive Experimental Framework:** Unlike many previous studies that focus solely on binary classification [Ibrahim and Mohammed 2022, Cil et al. 2021, Alghazzawi et al. 2021, Seifousadati et al. 2021, Alqarni 2022, Becerra-Suarez et al. 2024] or utilize a limited set of performance metrics [Akgun et al. 2022], our research evaluates binary and multiclass classification techniques, using comprehensive performance metrics.

2. **Comparative Analysis of Feature Selection Methods:** We introduce the first systematic comparison of Chi-square and PCA feature selection methods, specifically within the context of DDoS detection in cloud environments. This analysis offers valuable information on the effectiveness of these methods in various classification algorithms and scenarios.

3. **Data Imbalance Mitigation:** Most existing studies ([Ibrahim and Mohammed 2022, Cil et al. 2021, Alghazzawi et al. 2021, Seifousadati et al. 2021, Ramzan et al. 2023, Abbas and Almhanna 2021, Ali et al. 2023, Becerra-Suarez et al. 2024]) overlook the critical issue of data imbalance, which significantly impacts model performance in real-world applications. Our work explicitly addresses this gap by implementing and evaluating an oversampling technique, thereby improving the robustness of our findings.

4. **Practical Performance Evaluation:** We extend beyond traditional accuracy-focused metrics by analyzing both training and testing times. Although these factors are crucial for detecting DDoS attacks in real-time within dynamic cloud environments, they are often neglected in the literature.

5. **Excellent Performance Across Multiple Classification Paradigms:** Our pro-

Paper	Best Method	Dataset	Accuracy(%)	Recall(%)	Precision(%)	F1-score(%)	Specificity(%)
[Ibrahim and Mohammed 2022]	RF (Forward and Backward)	CICDDoS2019	100	100	0.99	0.99	0.99
	NB (Boruta)		0.98	0.97	0.99	0.98	0.99
	SVM (Variable Importance)		0.99	0.99	0.99	0.99	0.99
[Cil et al. 2021]	DNN	CICDDoS2019	0.99	0.99	0.99	0.99	-
[Alghazzawi et al. 2021]	Hybrid (CNN and BiLSTM)	CICDDoS2019	94.52	92.04	94.74	93.44	-
[Seifousadati et al. 2021]	AdaBoost	CICDDoS2019	100	-	-	100	-
	XGBoost		100	-	-	100	-
[Akgun et al. 2022]	CNN (Binary)	CICDDoS2019	99.99	-	-	-	-
	CNN (Multiclass)		99.30	-	-	-	-
[Ramzan et al. 2023]	RNN (Binary)	CICIDS2017	99.99	99.99	99.99	99.99	99.99
	GRU (Multiclass)		99.54	99	98	98	-
[Alqarni 2022]	Ensemble	CICDDoS2019	98.02	97.45	-	-	98.65
[Abbas and Almhanna 2021]	RF	CICDDoS2019	99.97	100	-	99.9	-
[Parfenov et al. 2020]	CatBoost (Binary)	CICDDoS2019	-	99.3	99.1	96.9	-
	CatBoost (Multi)		-	96.8	96.7	97	-
[Becerra-Suarez et al. 2024]	RF	CICDDoS2019	99.97	99.80	99.98	99.98	-
This study	RF (Binary)	CICDDoS2019	100	100	100	100	-
	K-NN (Binary)		99.72	100	100	100	-
	RF (Multi)		100	100	100	100	-
	K-NN (Multi)		99.66	100	100	100	-

Table 2: Comparative analysis of the proposed machine learning models' performance against state-of-the-art results for DDoS attack detection and classification.

posed approach achieves exceptional performance metrics in binary and multiclass settings. This performance surpasses that of previous state-of-the-art methods while maintaining practical efficiency.

Table 1 presents a comparison of the proposed methodology with current state-of-the-art studies, detailing the classification algorithms employed, feature selection methods used, data balancing techniques implemented, classification types investigated, and performance measures reported. Table 2 highlights the best-performing methods from recent studies and compares their results with our findings, utilizing quantitative metrics such as accuracy, recall, precision, F1-score, and specificity.

4 Research Methodology

This study involves conducting two distinct and independent experiments for classifying network traffic in a cloud environment: binary classification and multiclass classification. The binary classification experiment aims to efficiently and quickly classify network traffic as either an attack or benign. The dataset used for this experiment, dataset 1, consists of the same number of samples from DDoS attacks and benign records. The binary classification model trained on this dataset focuses on distinguishing between malicious and non-malicious traffic, providing a rapid response to potential threats. The multiclass classification experiment focuses on identifying and distinguishing specific types of DDoS attacks within the network traffic. The dataset used for this experiment, dataset 2, contains sufficient samples for each of the three DDoS attack types (NTP, UDP-Lag, and SYN) and benign records. The multiclass classification model trained on this dataset classifies the traffic into one of four categories: benign or one of the three specific DDoS attacks. This approach offers a more granular analysis of network traffic, enabling the identification of the specific type of DDoS attack that occurs in the cloud environment.

This section presents the methodology for performing binary and multiclass classification experiments using different machine learning models to detect and classify DDoS attacks. It describes the steps from data collection and preprocessing to model building and evaluation. Figure 1 illustrates the methodology used to conduct the two



Figure 1: DDoS detection methodology workflow for cloud environments

experiments. Algorithm 1 presents a high-level overview of the complete experimental framework with cloud deployment considerations. The algorithm framework considers key aspects for cloud deployment, with fixed random seeds (`random_state=42`) ensuring reproducibility and stratified sampling maintaining class balance. The iterative structure facilitates systematic evaluation of feature selection and classification methods, allowing cloud administrators to optimize configurations based on deployment constraints. Performance evaluation includes timing metrics critical for real-time applications, where latency impacts service-level agreements and auto-scaling.

The following sections provide a detailed explanation of the methodology steps, which include data collection, data preprocessing, feature selection, dataset splitting for training and testing, model building, and evaluation.

4.1 Data Collection

In the field of detecting DDoS attacks, many datasets are used in similar research papers, including KDDCup99, NSL-KDD, UNSW-NB15, CICIDS2017, and CIC-DDoS2019. Each dataset has different types of attack and varying numbers of records and features. In this study, we adopt the CIC-DDoS2019 dataset because it is one of the most recent publicly available datasets that includes diverse DDoS attack types and a rich set of flow-based features.

Sharafaldin et al. [Sharafaldin et al. 2019] developed this dataset to address the limitations of previous datasets, such as outdated traffic patterns, lack of diversity in attack types, and insufficient labeling. Their work introduced a new classification and detection framework for DDoS attacks based on network flow features extracted from simulated traffic scenarios.

Unlike previous datasets that focused on limited attack vectors, the CIC-DDoS2019 dataset incorporates recent DDoS attacks targeting application-layer protocols such as HTTP, HTTPS, and DNS. Consequently, Sharafaldin et al. developed a new taxonomy and examined emerging attacks that can be launched against application-layer TCP/UDP-based protocols.

Algorithm 1 Cloud-Based DDoS Detection Framework**Input:** CIC-DDoS2019 dataset**Output:** Trained models with performance metrics**Parameters:** random_state=42, test_size=0.3, k_features=20

```

1: Load dataset and remove unnamed columns
2: Handle missing/infinite values
3: Apply LabelEncoder to categorical features
4: Remove 4,404 duplicate records
5: Create:
  - Dataset 1: Binary (Attack vs Benign)
  - Dataset 2: Multiclass (SYN, NTP, UDP-Lag, Benign)
6: Apply RandomOverSampler(random_state=42)
7: Apply StandardScaler() for normalization
8: for feature selection method  $\in$  {Chi-square, PCA} do
9:   Select top- $k$  features ( $k = 20$ ) using the method
10:  for dataset  $\in$  {Binary, Multiclass} do
11:    Split dataset into train/test (70%/30%, stratified)
12:    for classifier  $\in$  {Random Forest, K-NN, Naive Bayes, Logistic Regression}
13:      do
14:        Train model, record training time
15:        Predict on test set, record testing time
16:        Evaluate: Accuracy, Precision, Recall, F1-score
17:        if Accuracy < 85% then
18:          Apply GridSearchCV(cv=5) for hyperparameter tuning
19:        end if
20:        Store performance metrics and confusion matrix
21:      end for
22:    end for
23:  end for
24: Generate comparative analysis with cloud deployment considerations

```

The CIC-DDoS2019 dataset is publicly available on Kaggle [Organization 2024] and the Canadian Institute for Cybersecurity website [Cybersecurity 2019]. It contains both benign and malicious traffic, with detailed annotations and metadata. Additionally, the dataset provides findings from the CICFlowMeter-V3 network traffic analysis, categorizing flows based on time stamps, destination and source ports, source and destination IP addresses, protocols, and attack types.

4.2 Data Preprocessing

The preprocessing operation prepares the dataset for our proposed models. This process includes: (1) collecting only the desired records (Benign and Attack), (2) handling null and infinite values, (3) removing duplicate records, (4) transforming non-numeric values using Label Encoder, and (5) balancing classes with sampling techniques. Finally, we

scaled the data to convert the feature vector to a more suitable format for machine learning models.

Sample Collection: The CIC-DDoS2019 dataset comprises 88 features and contains over 50 million records, representing 13 types of DDoS attacks. In particular, benign traffic is significantly less prevalent than attack traffic. Therefore, we compiled all benign records from various CSV files within the dataset. This study employs two experimental setups, which require the creation of customized datasets. For this purpose, three specific attack types, NTP, UDP-Lag, and SYN flood, were selected based on their prevalence and impact. The samples were randomly extracted from the relevant CSV files corresponding to these attack types. Two custom datasets were developed: dataset 1 for binary classification and dataset 2 for multiclass classification. Dataset 1 focuses on distinguishing between attack and benign traffic. It consists of 34,000 samples from each attack type (NTP, UDP-Lag, and SYN), which are combined into a single "attack" class, together with an equal number of 34,000 benign samples. Dataset 2 is designed for multiclass classification and contains 100,000 records for each attack type, along with 100,000 benign records. This structure enables the predictive model to differentiate between individual attack types and benign traffic, allowing for a more granular analysis.

Data Cleaning: Standard data cleaning procedures were executed on both datasets before any machine learning techniques were applied. First, we eliminated an unnamed column (['Unnamed: 0']), which did not provide valuable information, reducing the number of features from 88 to 87. Next, We replaced null or infinite values with the mean of their columns to maintain the dataset's size and integrity. Finally, we identified and removed 4,404 duplicate benign records from both dataset 1 and dataset 2. These cleaning steps ensured that the datasets were consistent, complete, and ready for subsequent modeling.

Non-numeric Value Transformation: We converted all non-numeric values into numerical ones because many machine learning algorithms can only process numerical values. Six columns of type object were found, which are (Flow ID, Source IP, Destination IP, Timestamp, SimilarHTTP, and Label). As for the class column (Label), it was converted in the first dataset from Benign to 0, while all other types of attacks (SYN, NTP, and UDP-Lag) were converted to 1 for the binary classification experiment. In the second dataset, Benign was converted to 0, SYN to 1, UDP-Lag to 2, and NTP to 3 for the multiclass classification experiment.

Data Balancing: Since the number of benign samples is less than the number of attack samples, especially after the step of removing duplicates, it is important to balance the data so that the number of benign samples becomes equal to the number of attack samples to obtain the best results.

The oversampling technique depends on adding new samples to the minority class to equalize all types of classes. Oversampling can be random oversampling or synthetic oversampling. The technique of synthetic oversampling generates artificial samples for the minority class. In this study, random oversampling is used to improve the representation of the minority class by duplicating available minority records [Shelke et al. 2017]. The method utilized is the RandomOverSampler from the imbalanced-learn (imblearn) library, and a fixed random state is set to ensure reproducibility. After applying the oversampling technique to unbalanced datasets, all elements in the label class were equalized so that the number of records for each class value in the first dataset became 102,000, resulting in an approximate total of 204,000 records for the first dataset. In the

second dataset, the number of records for each class value became 100,000, resulting in approximately 400,000 records. Figure 2 shows the balanced class distribution for dataset 1 (binary classification) and dataset 2 (multiclass classification) achieved by applying the oversampling technique.

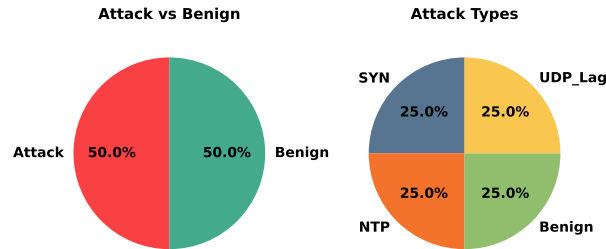


Figure 2: Balanced class distribution in dataset 1 (binary classification) and dataset 2 (multiclass classification) achieved through the application of the oversampling technique to address data imbalance and ensure equal representation of samples across all classes

Scaling Data: Feature scaling, or standardization, is an essential step in data preprocessing to normalize data in a specific range. It also helps to speed up the computations in the algorithm. Experiments depend on using Scikit-Learn, and machine learning usually requires feature scaling for the data. The CICDDoS 2019 dataset used in this study has variables with different scales. Therefore, feature scaling is performed in order to convert the feature vector to a more suitable format for machine learning models. StandardScaler() and MinMaxScaler() are considered the most widely used feature scaling techniques [Thara et al. 2019].

StandardScaler() [Organization 2024] is a technique that transforms the dataset such that the resulting distribution has a mean of zero and a standard deviation of one. The transformed value is calculated by subtracting the mean from the original value and then dividing the result by the standard deviation. The transformation is performed using Equation (1).

$$z = \frac{x - \mu}{\sigma} \quad (1)$$

In this equation, z represents the transformed value of the feature, μ is the mean of the feature, x is the original value of the feature, and σ is the standard deviation of the feature. MinMaxScaler() [Organization 2024] is another technique that scales the data so that all values in the dataset fall within the range of 0 to 1. The transformation is accomplished using Equation (2).

$$t = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \quad (2)$$

Here, t represents the transformed value of the feature, and X is the original value of the feature. X_{\min} and X_{\max} denote the minimum and maximum values of the feature x , respectively. In this study, StandardScaler was applied to ensure consistent feature distributions across all experiments.

4.3 Feature Selection

Feature selection is a technique that improves the performance of machine learning models by eliminating irrelevant or redundant features. In this study, feature selection played a crucial role in enhancing the performance of the proposed machine learning models. To ensure that the most relevant attributes were maintained while reducing the dimensionality of the dataset, two widely used feature selection techniques were employed: Chi-square and Principal Component Analysis (PCA). Both methods were configured to select the top 20 features/components to maintain consistency across experiments and enable fair comparison between approaches.

4.3.1 Chi-Square Feature Selection

The Chi-square score [Liu and Setiono 1995, Li et al. 2017] utilizes the test of independence to show whether a feature is independent of the class label. For a given feature f_i with r distinct feature values, the Chi-square score for that attribute can be computed using Equation (3):

$$\chi^2(f_i) = \sum_{i=1}^r \sum_{j=1}^k \frac{(A_{ij} - E_{ij})^2}{E_{ij}} \quad (3)$$

In order to boost the performance of the proposed machine learning models, the Chi-square feature selection method was applied to identify the most significant features in our dataset. By calculating the Chi-square statistic for each feature concerning the target variable, the features with the strongest associations were determined. The implementation utilizes SelectKBest from scikit-learn with the chi2 scoring function, configured to select k=20 features based on the highest chi-square statistics.

4.3.2 PCA Feature Selection

PCA is a commonly used statistical technique that aims to simplify high-dimensional data by reducing it to fewer dimensions while preserving important information. It does this by converting the initial features into a new set of independent variables called principal components. These principal components are a linear combination of the original features, sorted by how much variance they capture from the data. The first principal component accounts for the largest variation, and each subsequent component captures the next highest variance while maintaining independence with the preceding ones. PCA is a popular technique in data analysis and machine learning for reducing dimensionality, noise reduction, and data representation. It simplifies complex datasets, increases computational performance, and reduces the risk of overfitting in predictive models [Jolliffe and Cadima 2016]. To improve the effectiveness of the proposed machine learning models, the PCA feature selection method was employed to discover the most important elements in our dataset. The PCA is implemented using the PCA class from Scikit-learn with `n_components` set to 20, selecting the first 20 principal components that account for the maximum variance in the data.

4.4 Dataset Splitting for Training and Testing

The dataset was divided using a 70/30 split, where 70% of the data was allocated for training the machine learning models and the remaining 30% was reserved for testing their

performance. This approach ensures that the models have sufficient data to learn from during the training phase while maintaining a separate dataset for unbiased evaluation. By reserving a substantial portion of the data for testing, we can assess how well the models generalize to unseen data, thereby providing a realistic measure of their performance. This split ratio helps in mitigating the risk of overfitting, where the model performs exceptionally well on training data but fails to generalize to new data. The chosen split is a standard practice in machine learning, providing a balance between training and testing data to ensure the robustness and reliability of the model evaluation. The performance metrics obtained from the test set offer a clear indication of how the models are likely to perform in real-world scenarios. The data splitting was performed using scikit-learn's `train_test_split` function with stratification to maintain class distribution proportions and a fixed `random_state=42` for reproducibility.

4.5 Model Building and Evaluation

This section provides a comprehensive explanation of the algorithms employed in the two experiments and explains the performance metrics utilized to assess the effectiveness of these models.

Machine Learning Models: In this study, four machine learning models were selected: Random Forest, K-Nearest Neighbor (K-NN), Naïve Bayes, and Logistic Regression. Each model was initially evaluated using its default hyperparameter settings to establish baseline performance. Based on these initial results, Random Forest and K-NN were identified as the most promising models and were subjected to comprehensive hyperparameter tuning to optimize their performance.

The hyperparameter optimization process employed Grid Search with 5-fold cross-validation to systematically identify the best parameter combinations. The optimal settings obtained from this process were used in the final models for all subsequent experiments. The selected hyperparameters and their corresponding optimal or default values are summarized in Table 3.

For Naïve Bayes and Logistic Regression, preliminary evaluations showed that hyperparameter tuning did not yield meaningful performance gains. Therefore, these models were retained with their default settings as provided by the scikit-learn library. All models were implemented using scikit-learn version 0.24.2 in Python 3.7 environment with consistent random states for reproducibility.

Performance Measures: The performance of the proposed models is assessed using several metrics [Powers 2020], including a confusion matrix, accuracy, precision, recall, F1 score, training time, and testing time. The confusion matrix comprises four parameters: True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). The accuracy of the trained models represents the frequency with which they correctly identify the desired attacks. Accuracy is calculated using Equation (4).

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN} \quad (4)$$

Precision is determined by dividing the total number of correct positive predictions by the total number of positive predictions made by the model. It is computed using Equation (5).

$$\text{Precision} = \frac{TP}{TP + FP} \quad (5)$$

Algorithm	Parameter	Setting
Random Forest	n_estimators	200
	max_depth	None
	min_samples_split	2
	min_samples_leaf	1
	max_features	'sqrt'
K-Nearest Neighbor	n_neighbors	5
	weights	'distance'
	algorithm	'auto'
	p	2
Naïve Bayes	var_smoothing	1×10^{-9} (default)
	priors	None (default)
Logistic Regression	C	1.0 (default)
	penalty	'l2' (default)
	solver	'lbfgs' (default)
	max_iter	100 (default)
	multi_class	'auto' (default)

Table 3: Hyperparameter configurations for all machine learning algorithms used in the experiments

The recall of the model is calculated using Equation (6).

$$\text{Recall} = \frac{TP}{TP + FN} \quad (6)$$

The F1 score can be regarded as a more comprehensive evaluation metric since it combines both precision and recall. The F1 score of the model can be determined by using Equation (7).

$$\text{F1 score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (7)$$

Training time and testing time are calculated using the time library in Python [Organization 2024] by measuring the duration taken to accomplish specific tasks. Training time refers to the time required to train the machine learning model on the given dataset, while testing time is the duration for the trained models to make predictions using the test dataset. A start timer is used before initiating the training or testing process. After executing the task, an end timer is used. The overall training or testing time is then calculated by subtracting the start time from the end time. All timing measurements were performed on the same hardware configuration (Intel Core i5, 8GB RAM) to ensure consistent and comparable results across all experiments.

5 Experimental Results and Discussion

This section presents the results of binary and multiclass classification experiments that detect and classify DDoS attacks in a cloud environment using PCA and Chi-square feature selection methods.

The main findings from this study include the results of binary and multiclass experiments using Chi-square and PCA feature selection techniques with four machine learning algorithms (Random Forest, k-Nearest Neighbors, Naive Bayes, and Logistic Regression), which are summarized as follows.

- Random Forest and K-Nearest Neighbors consistently achieved the highest accuracy scores (around 99-100%) across both binary and multiclass experiments, regardless of the feature selection technique used (Chi-square or PCA). This level of performance is crucial in cloud environments, where rapid decision-making is essential for ensuring service availability across multiple virtual instances.
- Naive Bayes and Logistic Regression initially had lower accuracy scores, but their performance improved significantly after hyperparameter tuning using Grid Search. However, this improvement came at the cost of increased training times, particularly for Logistic Regression in the multiclass experiment. Balancing accuracy with computational efficiency is crucial in cloud environments, where resources are allocated dynamically and cost is a key consideration.
- The choice of feature selection technique (Chi-square or PCA) did not significantly impact the accuracy or performance time of the machine learning algorithms. This insight is valuable for cloud deployments as it helps maintain consistent performance across different virtual environments and varying network structures.
- The multiclass experiment generally had higher training and testing times than the binary experiment, with Random Forest having the highest training times and K-nearest Neighbors having the highest testing times. These timing factors are especially relevant for cloud-based implementations, where auto-scaling mechanisms must adapt to attack patterns within time constraints.

5.1 Experiment environment

The experimental setup utilizes existing datasets to simulate a typical cloud computing environment, ensuring a realistic evaluation of DDoS detection capabilities in virtualized infrastructures. Both experiments were conducted on a MacBook Pro computer featuring an Intel Core i5 processor with three cores running at a base clock speed of 2.3 GHz. The system also had 8 GB of LPDDR3 RAM clocked at 2133 MHz and a 256 GB SSD for storage. The graphics were handled by Intel Iris Plus Graphics 640 with 1536 MB of shared memory. The operating system used was macOS Mojave, version 10.14.6.

For the experimental setup, Python 3.7 was utilized along with Kaggle Notebooks for code development and execution. Several libraries were utilized to conduct the experiments: Pandas 1.3.3 for data manipulation, Scikit-Learn 0.24.2 for machine learning models and algorithms, and Matplotlib 3.4.3, along with Seaborn 0.11.2 for data visualization. Feature selection was carried out using both Chi-Square and PCA. Preprocessing steps included scaling techniques such as Min-Max Scaler and Standard Scaler. The dataset was split into training and testing sets using the Train-Test Split technique. Hyperparameter tuning was performed using Grid Search to optimize the models, and label encoding was applied to convert categorical variables into numerical values. To address the class imbalance, the Imblearn library 0.8.0 was utilized for oversampling, while the Time library was used to measure training and testing times. Various metrics such as accuracy, precision, recall, f1-score, and confusion matrix were calculated to evaluate the models' performance. The experimental design considers cloud-specific factors,

including traffic aggregation patterns from multiple virtual machines, network virtualization overhead, and the temporal dynamics of resource allocation. All experiments were conducted with fixed random seeds (random_state=42) to ensure reproducibility.

5.2 Binary classification results

The binary classification experiment relies on dataset 1, which comprises 204,000 records categorized into two class values: attack and Benign. This binary classification approach addresses the critical challenge in cloud environments of distinguishing between legitimate multi-tenant traffic and malicious attack patterns. Traditional signature-based methods often struggle in this context due to the dynamic nature of virtualized services. Two new datasets were obtained using Chi-square and PCA techniques, and traffic types were detected using Random Forest, Logistic Regression, Naive Bayes, and K-nearest Neighbors.

5.2.1 Binary classification results using Chi-square technique

The Chi-square feature selection method is highly effective in cloud environments. It identifies the key features that differentiate legitimate cloud service requests from DDoS attack patterns across virtualized network interfaces. The random forest algorithm achieved 100% accuracy, while K-nearest Neighbors achieved 99.72% accuracy. The Logistic Regression algorithm achieved an accuracy of 96.08%. Naive Bayes had the lowest accuracy of 83.58%. Random Forest had the highest training time, 14.84 seconds, and K-nearest Neighbors had the highest testing time, 25.12 seconds. Other performance measures, including precision, recall, F1 score, training, and testing time, are shown in Table 4.

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Training time (sec)	Testing time (sec)
Random Forest	100	100	100	100	14.84	0.38
K-nearest Neighbor	99.72	100	100	100	0.3	25.12
Naïve Bayes	83.58	87	84	83	0.08	0.02
Logistic Regression	96.08	96	96	96	3.06	0.01

Table 4: Performance results for binary experiment using Chi-square technique

The initially lower performance of Naive Bayes is due to the complexity of cloud traffic patterns. The independence assumption between features often fails because of correlations from virtualization overhead and shared resource utilization. To improve model performance, a Grid Search method for hyperparameter tuning was performed. The smoothing parameter in Naive Bayes significantly enhanced the model's performance. After applying the new values for var_smoothing as suggested by Grid Search (a value of 1×10^{-7}), the performance of the model showed significant improvements: the accuracy of Naive Bayes increased from 83.58% to 89.02%, with a similar enhancement level in the rest of the performance measures. Figure 3 shows the confusion matrix results for the Naive Bayes algorithm with default settings and after optimizing the model using the Grid Search technique.

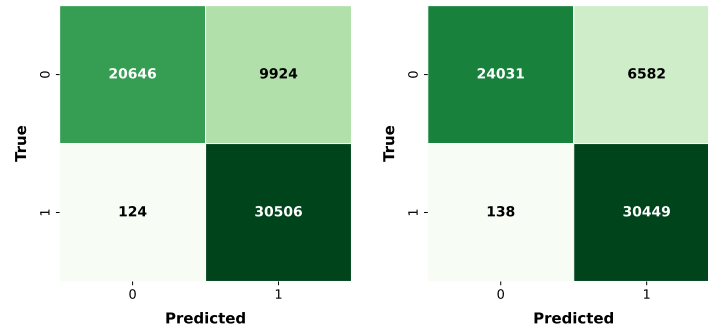


Figure 3: Confusion matrices for the Naive Bayes algorithm using the Chi-square technique, with default settings (left) and after hyperparameter tuning (right)

5.2.2 Binary classification results using PCA technique

The selection of PCA features helps to manage the complexities of the high-dimensional traffic of the cloud network. It reduces the number of features while keeping the variance needed to identify legitimate multi-tenant communications. The Random Forest algorithm achieved 100% accuracy, while K-nearest Neighbors achieved 99.54% accuracy. The Logistic Regression algorithm achieved an accuracy of 96.34%. Naive Bayes had the lowest accuracy of 70.67%. Random Forest had the highest training time with 12.26 seconds, and K-nearest Neighbors had the highest testing time with 25.83 seconds. Other performance measures, including precision, recall, F1 score, training, and testing time, are shown in Table 5.

After applying the new values for var_smoothing suggested by Grid Search (a value of 1×10^{-7}), the accuracy of Naïve Bayes increased from 70.67% to 81%, with a noticeable increase in the rest of the performance measures. This improvement highlights the need for optimizing algorithms in the PCA-transformed feature space. This is essential in cloud environments where network traffic characteristics are obscured by virtualization layers. Figure 4 shows the confusion matrix results for the Naive Bayes algorithm with default settings and after optimizing the model using the Grid Search technique.

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Training time (sec)	Testing time (sec)
Random Forest	100	100	100	100	12.26	0.39
K-nearest Neighbor	99.54	100	100	100	0.03	25.83
Naïve Bayes	70.67	81	70	67	0.08	0.03
Logistic Regression	96.34	96	96	96	2.72	0.01

Table 5: Performance results for binary experiment using PCA technique

5.2.3 Discussion

The binary classification results show that machine learning methods effectively differentiate between legitimate cloud service traffic and malicious attack patterns in virtualized environments. The final optimized results are presented in Table 6. Using Dataset 1

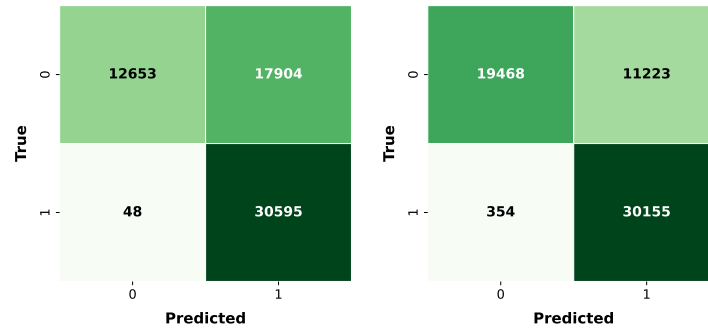


Figure 4: Confusion matrices for the Naive Bayes algorithm using the PCA technique, with default settings (left) and after hyperparameter tuning (right)

(204,000 records with 87 features), we applied Chi-square and PCA feature selection to create datasets with 20 features each, split 70%/30% for training/testing, and evaluated four algorithms across comprehensive performance metrics.

Random Forest achieved optimal performance with 100% accuracy for both feature selection methods and fast testing times (0.4 seconds), making it ideal for real-time cloud deployments requiring rapid threat identification. K-NN demonstrated comparable accuracy (99.72% Chi-square, 99.54% PCA) but significantly slower testing times (about 25 seconds), limiting its suitability for latency-sensitive cloud applications. This performance differential is crucial in cloud environments, where false positives can result in unnecessary resource allocation or service disruptions, while false negatives could allow attacks to affect multiple virtual instances simultaneously.

Naive Bayes initially showed lower performance (83.58% with Chi-square, 70.67% with PCA) but improved substantially through Grid Search hyperparameter optimization (89.02% and 81% respectively). Logistic Regression maintained consistent performance (about 96%) across both feature selection methods.

The consistent high performance across feature selection methods provides deployment flexibility, allowing cloud administrators to choose based on computational constraints rather than accuracy considerations. For multi-tenant environments, Random Forest offers immediate threat classification without detailed attack type identification, enabling rapid service isolation. Its reasonable training time (14.84 seconds) supports periodic model retraining in dynamic cloud environments where traffic patterns evolve.

Feature selection	Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Training time (sec)	Testing time (sec)
Chi-square	Random Forest	100	100	100	100	14.84	0.38
	K-nearest Neighbors	99.72	100	100	100	0.3	25.12
	Naive Bayes	89.02	91	89	89	0.08	0.03
	Logistic Regression	96.08	96	96	96	3.06	0.01
PCA	Random Forest	100	100	100	100	12.26	0.39
	K-nearest Neighbors	99.54	100	100	100	0.03	25.83
	Naive Bayes	81	85	81	80	0.12	0.03
	Logistic Regression	96.34	96	96	96	2.72	0.01

Table 6: Summary of binary experiment results

5.3 Multiclass classification results

This section presents the results of multiclass classification experiments using dataset 2, which comprises 400,000 records categorized into four class values: benign, NTP, UDP-Lag, and SYN. The multiclass approach meets the critical need in cloud environments to detect and classify attacks. This enables targeted mitigation strategies for various virtual network segments and tenant isolations. Two new datasets were obtained using Chi-square and PCA techniques, and four machine learning models were used to classify traffic types.

5.3.1 Multiclass classification results using Chi-square technique

The multiclass classification using Chi-square features effectively distinguishes between various attack vectors targeting cloud infrastructures. This includes protocol-specific attacks that exploit virtualization layers and multi-tenant network architectures. The Random Forest algorithm achieved 100% accuracy, while K-nearest Neighbors achieved 99.59% accuracy. The Naive Bayes algorithm achieved an accuracy of 92.34%. Logistic Regression had the lowest accuracy of 79.46%. Random Forest had the highest training time, 36.62 seconds, and K-nearest Neighbors had the highest testing time, 89.14 seconds. Other performance measures, including precision, recall, F1 score, training, and testing time, are shown in Table 7.

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Training time (sec)	Testing time (sec)
Random Forest	100	100	100	100	36.62	0.95
K-nearest Neighbor	99.59	100	100	100	0.05	89.14
Naïve Bayes	92.34	93	92	92	0.14	0.06
Logistic Regression	79.46	80	79	79	15.04	0.01

Table 7: Performance results for multiclass experiment using Chi-square technique

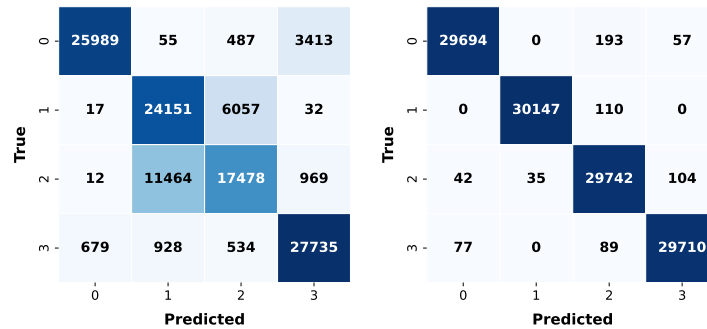


Figure 5: Confusion matrices for the Logistic Regression algorithm using the Chi-square technique, with default settings (left) and after hyperparameter tuning (right)

By applying the parameters recommended by the grid search ($C = 10$, $\text{penalty} = \text{l2}$, $\text{solver} = \text{lbfgs}$, $\text{max_iter} = 1000$), the accuracy of the logistic regression model improved

significantly, increasing from 79.46% to 99%. However, this enhancement resulted in considerably longer training times, which impacted overall performance efficiency. In cloud environments, it is crucial to balance accuracy and computational efficiency. Detection systems need to manage resource consumption while ensuring rapid threat identification across multiple virtual instances and tenant environments. Figure 5 shows the confusion matrix results for the Logistic Regression algorithm with default settings and after optimizing the model using the Grid Search technique.

5.3.2 Multiclass classification results using PCA technique

PCA-based multiclass classification effectively distinguishes between different attack types in cloud environments. It addresses the challenges posed by virtualization overhead and dynamic resource allocation, making it easier to analyze obscured traffic patterns. The Random Forest algorithm achieved 100% accuracy, while K-nearest Neighbors achieved 99.66% accuracy. The Naive Bayes algorithm achieved an accuracy of 90.63%. Logistic Regression had the lowest accuracy of 79.49%. Random Forest had the highest training time with 41.33 seconds, and K-nearest Neighbors had the highest testing time with 79.23 seconds. Other performance measures, including precision, recall, F1 score, training, and testing time, are shown in Table 8.

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Training time (sec)	Testing time (sec)
Random Forest	100	100	100	100	41.33	1.01
K-nearest Neighbor	99.66	100	100	100	0.05	79.23
Naïve Bayes	90.63	92	90	90	0.14	0.06
Logistic Regression	79.49	80	79	79	14.23	0.02

Table 8: Performance results for multiclass experiment using PCA technique

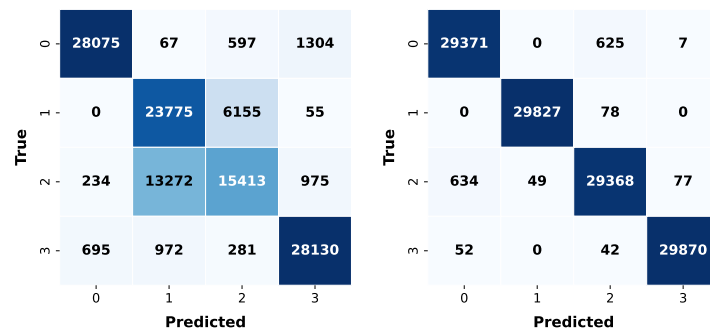


Figure 6: Confusion matrices for the Logistic Regression algorithm using the PCA technique, with default settings (left) and after hyperparameter tuning (right)

After applying the new values suggested by Grid Search ($C = 10$, $\text{penalty} = 'l2'$, $\text{solver} = 'lbfgs'$, $\text{max_iter} = 1000$), the accuracy of the Logistic Regression model improved significantly, rising from 79.49% to 99%. However, this enhancement resulted in

a significant increase in training time, rising from 14.23 seconds to 1739.45 seconds. This highlights the computational challenges of implementing complex detection algorithms in cloud environments, where resource management is critical among competing services and tenants. Figure 6 shows the confusion matrix results for the Logistic Regression algorithm with default settings and after optimizing the model using the Grid Search technique.

5.3.3 Discussion

The multiclass classification results demonstrate the ability to identify and categorize various DDoS attacks on cloud infrastructures. These attacks often target specific vulnerabilities in virtualization layers, API endpoints, and multi-tenant network architectures. The final optimized results are presented in Table 9. Using Dataset 2 (400,000 records across four classes: benign, SYN, NTP, UDP-Lag), we applied identical methodology with Chi-square and PCA feature selection, creating 20-feature datasets evaluated across comprehensive performance metrics.

Random Forest achieved perfect accuracy (100%) for both feature selection methods with reasonable training times (36.62-41.33 seconds), making it optimal for cloud deployments requiring periodic retraining. K-NN demonstrated comparable accuracy (99.59% Chi-square, 99.66% PCA) but significantly higher testing times (79-89 seconds), creating computational challenges for real-time cloud applications. This performance highlights their effectiveness in managing complex feature interactions in cloud network traffic, effectively differentiating between legitimate multi-tenant communications and attack patterns that resemble normal service requests.

Logistic Regression initially underperformed (about 79% accuracy) but achieved 99% accuracy after Grid Search optimization. However, this improvement came with substantial training time penalties (from 15.04 to 791.77 seconds for Chi-square; 14.23 to 1739.45 seconds for PCA). This computational intensity raises key deployment considerations for cloud environments, where longer training times may require offline training or distributed computing to ensure real-time detection capabilities.

Multiclass classification enables targeted defense strategies crucial for cloud environments, allowing protocol-specific mitigation for SYN, NTP, and UDP-Lag attacks. The ability to distinguish attack types facilitates advanced defense mechanisms such as protocol-specific filtering and tenant-aware traffic shaping, essential for maintaining service quality in multi-tenant infrastructures. This granular classification also enables detailed security reporting and compliance documentation for cloud service providers, supporting comprehensive analysis across different virtual network segments and tenant configurations.

Feature selection	Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Training time (sec)	Testing time (sec)
Chi-square	Random Forest	100	100	100	100	36.62	0.95
	K-nearest Neighbors	99.59	100	100	100	0.05	89.14
	Naive Bayes	92.34	93	92	92	0.14	0.06
	Logistic Regression	99	99	99	99	791.77	0.23
PCA	Random Forest	100	100	100	100	41.33	1.01
	K-nearest Neighbors	99.66	100	100	100	0.05	79.23
	Naive Bayes	90.63	92	90	90	0.14	0.06
	Logistic Regression	99	99	99	99	1739.45	0.15

Table 9: Summary of multiclass experiment results

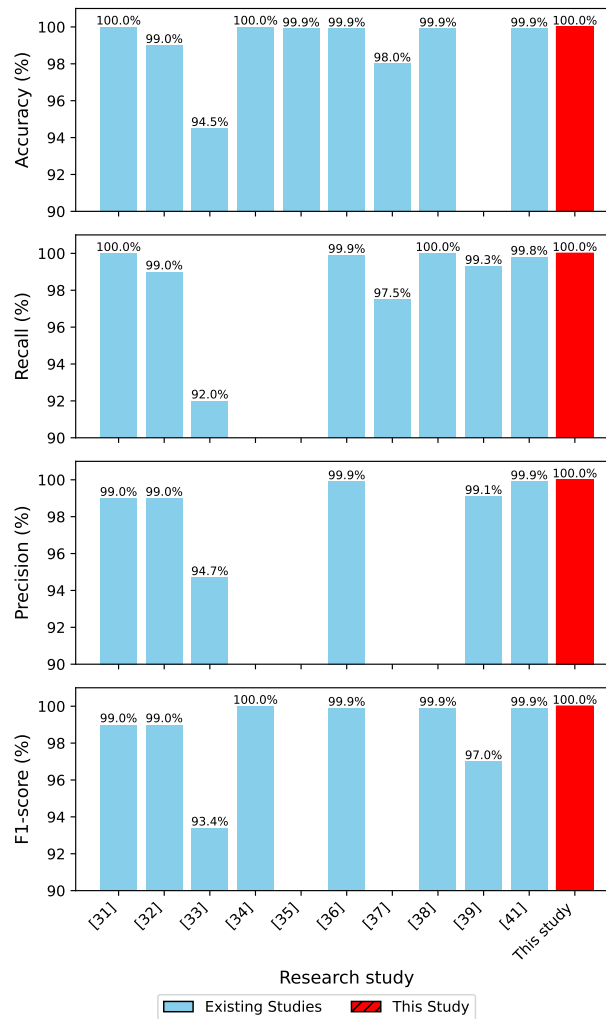


Figure 7: Comparative analysis of DDoS detection methods across different existing studies and this study

5.4 Comparison with Existing Studies and Performance Analysis

Figure 7 compares DDoS detection methods proposed in existing studies with that of this study in terms of four key performance metrics: accuracy, recall, precision, and F1-score. The exceptional performance highlighted in this study (indicated by red bars) is particularly significant for cloud environments. In these settings, detection systems need to maintain high accuracy while functioning across virtualized infrastructures. They must also handle varying network topologies and multi-tenant traffic patterns. It is evident that this study consistently achieves top-tier performance across all four metrics, achieving scores of 100% in accuracy, recall, precision, and F1-score. This exceptional performance

underscores the robustness and effectiveness of the methods employed in the current study, particularly in achieving a balance between correctly identifying DDoS attacks (high recall) and minimizing false positives (high precision). In cloud environments, this balance is crucial since false positives can trigger unnecessary resource scaling and associated costs. False negatives, on the other hand, can enable attacks to spread across multiple virtual instances, threatening the availability of services for multiple tenants simultaneously.

Our study demonstrates several advantages over existing research, particularly in addressing cloud-specific challenges. Unlike many existing studies that lack consistency in their approaches, this study addresses multiple crucial aspects: data processing, feature selection (using Chi-squared and PCA methods), data balancing (through oversampling), and employs a variety of algorithms (RF, NB, K-NN, LR) for both binary and multiclass classification. This comprehensive approach is essential for cloud environments where attack patterns may vary significantly across different virtualization layers, tenant configurations, and dynamic resource allocation scenarios. While studies like [Ibrahim and Mohammed 2022], [Seifousadati et al. 2021], [Ramzan et al. 2023], and [Becerra-Suarez et al. 2024] showed high accuracy, they neglected to address data imbalance or failed to report on all crucial metrics. In contrast, our study explicitly tackles the data imbalance issue through oversampling, ensuring more reliable and unbiased results across all classes. This is particularly important in cloud environments where legitimate traffic volumes may vary dramatically between different tenants and services, creating natural imbalances that can skew detection results. Furthermore, unlike most previous research, our study considers both training and testing times, which are crucial for assessing the practicality and scalability of the models in real-world scenarios. In cloud deployments, these timing considerations significantly impact the implementation of auto-scaling defense mechanisms and the maintenance of service-level agreements during active attack scenarios.

6 Conclusion

In cloud services, organizations are increasingly facing DDoS attacks that threaten operational continuity. This study developed practical approaches for detecting and classifying these attacks using machine learning techniques. Our comprehensive evaluation of four algorithms with two feature selection methods demonstrates an effective detection framework that achieves remarkable accuracy in both classification experiments.

The empirical results demonstrated the superior performance of Random Forest (100% accuracy) with both Chi-square and PCA features. K-Nearest Neighbors also performed exceptionally well (>99.5%) though with different computational characteristics. These high-performance algorithms are ideal for cloud settings where quick decisions are crucial to maintaining service availability in multiple virtual instances. Furthermore, our results showed that the proper feature selection and data balancing techniques significantly enhance detection performance, although the differences between specific feature selection methods were minimal. This uniformity in performance across feature selection methods is essential in cloud deployments, where consistent detection is critical for maintaining service level agreements.

Although this study has yielded promising results, it has several limitations. These include reliance on a single dataset with only three attack types, potential scalability issues with the K-NN algorithm in high-volume environments, and static feature selection that may not adapt to evolving attack patterns.

Future work includes implementing deep learning techniques (particularly RNNs), using alternative datasets for cross-validation, exploring more complex attack types at the application layer, and developing a real-time detection framework with adversarial machine learning techniques. Future research should also focus on designing adaptive detection mechanisms for auto-scaling environments, exploring federated learning for multi-tenant security with privacy considerations, and developing cost-effective detection strategies that align security effectiveness with cloud resource usage.

Acknowledgements

The author expresses appreciation to Taibah University for its supervisory assistance.

References

- [Abbas and Almhanna 2021] Abbas, S., Almhanna, M.: “Distributed denial of service attacks detection system by machine learning based on dimensionality reduction”; *Proc. Journal of Physics: Conference Series*, pp. 012136. IOP Publishing (2021).
- [Aceto et al. 2013] Aceto, G., Botta, A., De Donato, W., Pescapè, A.: “Cloud monitoring: A survey”; *Computer Networks* 57, 9, 2093–2115 (2013), Elsevier.
- [Ahmed and Alexandrov 2011] Ahmed, K., Alexandrov, V.: “Identity and Access Management in Cloud Computing”; In: *Cloud Computing for Enterprise Architectures*, pp. 115–133. Springer (2011).
- [Akgun et al. 2022] Akgun, D., Hizal, S., Cavusoglu, U.: “A new DDoS attacks intrusion detection model based on deep learning for cybersecurity”; *Computers & Security* 118, 102748 (2022), Elsevier.
- [Alghazzawi et al. 2021] Alghazzawi, D., Bamasag, O., Ullah, H., Asghar, M.: “Efficient detection of DDoS attacks using a hybrid deep learning model with improved feature selection”; *Applied Sciences* 11, 24, 11634 (2021), MDPI.
- [Alharby 2024] Alharby, M.: “Preserving Data Secrecy and Integrity for Cloud Storage Using Smart Contracts and Cryptographic Primitives.”; *Computers, Materials & Continua* 79, 2 (2024).
- [Ali et al. 2023] Ali, T., Chong, Y., Manickam, S.: “Comparison of ML/DL approaches for detecting DDoS attacks in SDN”; *Applied Sciences* 13, 5, 3033 (2023), MDPI.
- [AlJahdali 2017] AlJahdali, H.: “Improving multi-tenancy security by controlling resource allocation in IaaS public clouds”; Ph.D. thesis, University of Leeds (2017).
- [AlJahdali et al. 2014] AlJahdali, H., Albatli, A., Garraghan, P., Townend, P., Lau, L., Xu, J.: “Multi-tenancy in cloud computing”; *Proc. 2014 IEEE 8th international symposium on service oriented system engineering*, pp. 344–351. IEEE (2014).
- [Alqarni 2022] Alqarni, A.: “Majority vote-based ensemble approach for distributed denial of service attack detection in cloud computing”; *Journal of Cyber Security and Mobility*, 265–278 (2022).
- [Becerra-Suarez et al. 2024] Becerra-Suarez, F., Fernández-Roman, I., Forero, M.: “Improvement of Distributed Denial of Service Attack Detection through Machine Learning and Data Processing”; *Mathematics* 12, 9, 1294 (2024), MDPI.
- [Cil et al. 2021] Cil, A., Yildiz, K., Buldu, A.: “Detection of DDoS attacks with feed forward based deep neural network model”; *Expert Systems with Applications* 169, 114520 (2021), Elsevier.
- [Cybersecurity 2019] Canadian Institute for Cybersecurity: “DDoS evaluation dataset (CIC-DDoS2019)”; <https://www.unb.ca/cic/datasets/ddos-2019.html>. Accessed: January 4, 2024.

- [Daffu and Kaur 2016] Daffu, P., Kaur, A.: “Mitigation of DDoS attacks in cloud computing”; Proc. 2016 5th International Conference on Wireless Networks and Embedded Systems (WECON), pp. 1–5. IEEE (2016).
- [DiDio 2019] Laura DiDio: “Hourly Downtime Costs Rise: 86% of Firms Say One Hour of Downtime Costs \$300,000+; 34% of Companies Say One Hour of Downtime Tops \$1 Million”; <https://shorturl.at/wCFI>. Accessed: May 10, 2024.
- [Eddy 2006] Eddy, W.: “Defenses against TCP SYN flooding attacks”; The Internet Protocol Journal 9, 4, 2–16 (2006), Cisco.
- [Ficco 2013] Ficco, M.: “Security event correlation approach for cloud computing”; International Journal of High Performance Computing and Networking 17, 3, 173–185 (2013), Inderscience Publishers Ltd.
- [Furht et al. 2010] Furht, B., Escalante, A., others: “Handbook of cloud computing”; vol. 3, Springer (2010).
- [Gu and Liu 2007] Gu, Q., Liu, P.: “Denial of service attacks”; Handbook of Computer Networks: Distributed Networks, Network Planning, Control, Management, and New Trends and Applications 3, 454–468 (2007), John Wiley & Sons Hoboken, NJ, USA.
- [Hoon et al. 2018] Hoon, K., Yeo, K., Azam, S., Shunmugam, B., De Boer, F.: “Critical review of machine learning approaches to apply big data analytics in DDoS forensics”; Proc. 2018 International Conference on Computer Communication and Informatics (ICCCI), pp. 1–5. IEEE (2018).
- [Ibrahim and Mohammed 2022] Ibrahim, Z., Mohammed, I.: “Analysis of Features Selection Effects on Different Classification Algorithms with Performance Metrics Improvement based on PortScan-attack of CICDDoS2019 Dataset”; Journal of Algebraic Statistics 13, 3, 1712–1723 (2022).
- [Jolliffe and Cadima 2016] Jolliffe, I., Cadima, J.: “Principal component analysis: a review and recent developments”; Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences 374, 20150202 (2016), the Royal Society publishing.
- [Khan et al. 2024] Khan, M., Siam, R., Adnan, M.: “A framework for checking and mitigating the security vulnerabilities of cloud service RESTful APIs”; Service Oriented Computing and Applications, 1–22 (2024), Springer.
- [Khedr et al. 2023] Khedr, W., Gouda, A., Mohamed, E.: “FMDADM: A multi-layer DDoS attack detection and mitigation framework using machine learning for stateful SDN-based IoT networks”; IEEE Access 11, 28934–28954 (2023), IEEE.
- [Alharby 2024] Alharby, M.: “Preserving Data Secrecy and Integrity for Cloud Storage Using Smart Contracts and Cryptographic Primitives”; Computers, Materials & Continua 79, 2 (2024).
- [Li et al. 2017] Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R., Tang, J., Liu, H.: “Feature selection: A data perspective”; ACM computing surveys (CSUR) 50, 6, 1–45 (2017), ACM New York, NY, USA.
- [Liu and Setiono 1995] Liu, H., Setiono, R.: “Chi2: Feature selection and discretization of numeric attributes”; Proc. Proceedings of 7th IEEE international conference on tools with artificial intelligence, pp. 388–391. Ieee (1995).
- [Ma et al. 2023] Ma, R., Chen, X., Zhai, R.: “A DDoS Attack Detection Method Based on Natural Selection of Features and Models”; Electronics 12, 4, 1059 (2023), MDPI.
- [Mell et al. 2011] Mell, P., Grance, T., others: “The NIST definition of cloud computing” (2011), Computer Security Division, Information Technology Laboratory, National ...
- [Mills 1991] Mills, D.: “Internet time synchronization: the network time protocol”; IEEE Transactions on communications 39, 10, 1482–1493 (1991), Ieee.

- [Najar and others 2024] Najar, A., others: “A robust ddos intrusion detection system using convolutional neural network”; *Computers and Electrical Engineering* 117, 109277 (2024), Elsevier.
- [NexusGuard 2020] NexusGuard: “DDoS Annual Threat Report 2020”; <https://www.nexusguard.com/file/nexusguard-ddos-threat-report-2020-annual>. Accessed: May 02, 2024.
- [Ntambu and Adeshina 2021] Ntambu, P., Adeshina, S.: “Machine learning-based anomalies detection in cloud virtual machine resource usage”; *Proc. 2021 1st International Conference on Multidisciplinary Engineering and Applied Science (ICMEAS)*, pp. 1–6. IEEE (2021).
- [Olenick 2020] Doug Olenick: “European Bank Targeted in Massive Packet-Based DDoS Attack”; <https://www.bankinfosecurity.com/european-bank-targeted-in-massive-packet-based-ddos-attack-a-14505>. Accessed: May 02, 2024.
- [Organization 2024] Kaggle Organization: “Kaggle: Your Machine Learning and Data Science Community”; <https://www.kaggle.com/>. Accessed: February 16, 2024.
- [Organization 2024] Scikit-learn Organization: “StandardScaler”; <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>. Accessed: Mays 9, 2024.
- [Organization 2024] Python Organization: “time — Time access and conversions”; <https://docs.python.org/3/library/time.html>. Accessed: Mays 9, 2024.
- [Organization 2024] Scikit-learn Organization: “MinMaxScaler”; <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html#sklearn.preprocessing.MinMaxScaler>. Accessed: Mays 9, 2024.
- [Parfenov et al. 2020] Parfenov, D., Kuznetsova, L., Yanishevskaya, N., Bolodurina, I., Zhigalov, A., Legashev, L.: “Research application of ensemble machine learning methods to the problem of multiclass classification of DDoS attacks identification”; *Proc. 2020 International Conference Engineering and Telecommunication (En&T)*, pp. 1–7. IEEE (2020).
- [Polat et al. 2019] Polat, H., POLAT, O., SÖÜT, E., ERDEM, O.: “Performance analysis of between software defined wireless network and mobile ad hoc network under dos attack”; *Proc. 2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pp. 1–5. IEEE (2019).
- [Potluri et al. 2020] Potluri, S., Mangla, M., Satpathy, S., Mohanty, S.: “Detection and prevention mechanisms for DDoS attack in cloud computing environment”; *Proc. 2020 11th international conference on computing, communication and networking technologies (ICCCNT)*, pp. 1–6. IEEE (2020).
- [Powers 2020] Powers, D.: “Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation”; *arXiv preprint arXiv:2010.16061* (2020).
- [Ramachandra et al. 2017] Ramachandra, G., Iftikhar, M., Khan, F.: “A comprehensive survey on security in cloud computing”; *Procedia Computer Science* 110, 465–472 (2017), Elsevier.
- [Ramyachitra and Manikandan 2014] Ramyachitra, D., Manikandan, P.: “Imbalanced dataset classification and solutions: a review”; *International Journal of Computing and Business Research (IJCBR)* 5, 4, 1–29 (2014).
- [Ramzan et al. 2023] Ramzan, M., Shoaib, M., Altaf, A., Arshad, S., Iqbal, F., Castilla, , Ashraf, I.: “Distributed denial of service attack detection in network traffic using deep learning algorithm”; *Sensors* 23, 20, 8642 (2023), MDPI.
- [Rish and others 2001] Rish, I., others: “An empirical study of the naive Bayes classifier”; *Proc. IJCAI 2001 workshop on empirical methods in artificial intelligence*, pp. 41–46. Citeseer (2001).
- [Rudman 2014] Lauren Rudman: “Analysis of ntp based amplification ddos attacks”; <https://www.cs.ru.ac.za/research/g11r0252/Thesis.pdf>. Accessed: May 14, 2024.
- [Seifousadati et al. 2021] Seifousadati, A., Ghasemshirazi, S., Fathian, M.: “A Machine Learning approach for DDoS detection on IoT devices”; *arXiv preprint arXiv:2110.14911* (2021).

- [Sharafaldin et al. 2019] Sharafaldin, I., Lashkari, A., Hakak, S., Ghorbani, A.: “Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy”; Proc. 2019 international carnahan conference on security technology (ICCST), pp. 1–8. IEEE (2019).
- [Shelke et al. 2017] Shelke, M., Deshmukh, P., Shandilya, V.: “A review on imbalanced data handling using undersampling and oversampling technique”; Int. J. Recent Trends Eng. Res 3, 4, 444–449 (2017).
- [Shield 2020] AWS Shield: “Threat Landscape Report – Q1 2020”; https://aws-shield-tlr.s3.amazonaws.com/2020-Q1_AWS_Shield_TLR.pdf. Accessed: May 10, 2024.
- [Speelman 2014] Speelman, D.: “Logistic regression”; Corpus methods for semantics: Quantitative studies in polysemy and synonymy 43, 487–533 (2014).
- [Speiser et al. 2019] Speiser, J., Miller, M., Tooze, J., Ip, E.: “A comparison of random forest variable selection methods for classification prediction modeling”; Expert systems with applications 134, 93–101 (2019), Elsevier.
- [Sumathi and Karthikeyan 2018] Sumathi, S., Karthikeyan, N.: “Search for effective data mining algorithm for network based intrusion detection (NIDS)-DDoS attacks”; Proc. 2018 International Conference on Intelligent Computing and Communication for Smart World (I2C2SW), pp. 41–45. IEEE (2018).
- [Sundaramurthy et al. 2025] Sundaramurthy, S., Ravichandran, N., Inaganti, A., Muppalaneni, R.: “AI-Driven Threat Detection: Leveraging Machine Learning for Real-Time Cybersecurity in Cloud Environments”; Artificial Intelligence and Machine Learning Review 6, 1, 23–43 (2025).
- [Thara et al. 2019] Thara, D., PremaSudha, B., Xiong, F.: “Auto-detection of epileptic seizure events using deep neural network with different feature scaling techniques”; Pattern Recognition Letters 128, 544–550 (2019), Elsevier.
- [Wani et al. 2019] Wani, A., Rana, Q., Saxena, U., Pandey, N.: “Analysis and detection of DDoS attacks on cloud computing environment using machine learning techniques”; Proc. 2019 Amity International conference on artificial intelligence (AICAI), pp. 870–875. IEEE (2019).
- [Wankhede and Kshirsagar 2018] Wankhede, S., Kshirsagar, D.: “DoS attack detection using machine learning and neural network”; Proc. 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), pp. 1–5. IEEE (2018).
- [Watada et al. 2019] Watada, J., Roy, A., Kadikar, R., Pham, H., Xu, B.: “Emerging trends, techniques and open issues of containerization: A review”; IEEE Access 7, 152443–152472 (2019), IEEE.
- [Wehbi et al. 2019] Wehbi, K., Hong, L., Al-salah, T., Bhutta, A.: “A survey on machine learning based detection on DDoS attacks for IoT systems”; Proc. 2019 SoutheastCon, pp. 1–6. IEEE (2019).
- [Yoachimik and Pacheco 2024] Omer Yoachimik, Jorge Pacheco: “DDoS threat report for 2024 Q1”; <https://blog.cloudflare.com/ddos-threat-report-for-2024-q1>. Accessed: May 11, 2024.
- [Zhang 2016] Zhang, Z.: “Introduction to machine learning: k-nearest neighbors”; Annals of translational medicine 4, 11 (2016), AME Publications.
- [Zhang et al. 2010] Zhang, Q., Cheng, L., Boutaba, R.: “Cloud computing: state-of-the-art and research challenges”; Journal of internet services and applications 1, 7–18 (2010), Springer.
- [Zy et al. 2024] Zy, A., Rifa’i, A., Kamalia, A., Sulaeman, A., others: “Detecting DDoS attacks through decision tree analysis: an EDA approach with the CIC DDoS 2019 dataset”; Proc. 2024 8th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE), pp. 202–207. IEEE (2024).