Detection of Distributed Denial of Service (DDoS) Attacks Using Machine Learning Techniques

Azmina Sharaf [1], Dr. Kannan Ramakrishnan [2]

[1,2] Faculty of Computing and Informatics, Multimedia University, Cyberjaya, Selangor
Email: 1181102970@student.mmu.edu.my

Abstract. In the domain of network security, a distributed denial of service (DDoS) assault is one of the most harmful. DDoS attacks prevent vital services from operating normally for many online applications. DDoS-affected systems continue to be busy with erroneous requests rather than provide services to actual users. Attacks like these are becoming more frequent and sophisticated every day. As a result, it is now challenging to identify these assaults and protect online services from them. In this study, we applied a machine learning-based methodology to identify and categorize the traffic as either normal or DoS. The balanced and upgraded CICDDoS2019 dataset, which contains a mixture of various contemporary forms of assaults and regular traffic, is used to validate the suggested technique. The attacks are categorized using a Google Colab machine learning tool that is available online. The performance of the Logistic Regression Classifier will be compared with that of K-Nearest Neighbor and Random Forest in this study. Oversampling and undersampling will be used to address the class imbalance in the dataset. In addition, hyperparameter adjustment will be used to enhance the performance of the classifying models.

Keywords: Distributed Denial of Service, Machine Learning, Random Forest, K-Nearest Neighbor, Logistic Regression, Accuracy, Precision, Recall, F1-Score

1. Introduction

Distributed Denial of Service (DDoS) attacks are one of the most critical and dangerous threats facing the internet today because of the high volume and complexity involved in the traffic and service disruption to the legitimate users for a prolonged period. Early detection and prevention of DDoS attacks is essential for the efficient operations and availability of E-commerce, banking, communication, and educational facilities without disturbance to legitimate users. Even though several types of approaches have been proposed and implemented over the years for the detection of DDoS attacks, no single approach was able to satisfy all the essential goals of DDoS detection systems which includes high accuracy in early detection with low computational cost. Moreover, the existing techniques focus on detecting only specific types of DDoS attacks without considering the variations and complexities involved in newer type attacks. This project aims to do optimal feature selection for reducing computational cost, design and implement different machine learning models to detect DDoS attack traffic from the normal traffic and identify the optimal model by evaluating the performance over

multiple datasets with different DDoS attack types to improve scalability. In conclusion, the scope is limited to the application of three machine learning models. I aim to develop an ML model with LR and compare their performance with RF and KNN and identify the optimal model for classifying the network traffic into DDoS attack type or benign. We will be focusing on the available CICDDoS2019 dataset for the study.

2. Literature Review

2.1. Review on Journal Papers

2.1.1 DoS Attack Detection Using Machine Learning and Neural Network, 2018

The CIC IDS 2017 dataset was used in [1] article. They used two machine learning models (RF and MLP) to categorize the traffic into DoS attack and benign traffic. The process of feature selection is random. Accuracy is employed in the data analysis. With 50% training, the RF algorithm has accuracy that is 99.5% higher than MLP.

2.1.2 DDoS Attack Detection in Telecommunication Network Using Machine Learning, 2019

[2] manually collected data, observed traffic flow, and saved it. The machine learning models NN, SVM, and CNSVM were employed to categorize the obtained data into normal and abnormal categories. Domain knowledge was used to choose the features, such as traffic patterns, packet sizes, source IP addresses, and port numbers. Quadratic entropy was used to examine the collected data. Accuracy served as the benchmark for measuring performance. The suggested CNSVM provides a 40% improved identification of the distributed denial service when compared to SVM and NN for detection accuracy.

2.1.3 DDoS Attack Detection Using Machine Learning Techniques in Cloud Computing Environments, 2017

In an Oracle Virtual Box base environment, [3] created the attack traffic using Hping3 and regular network traffic using parameterized Python scripts. The data were divided into normal and abnormal categories using the ML models NB, C4.5, and K-Means. A branch was made for each potential value of the features chosen, with the biggest gain ratio serving as the splitting criterion. Using F-measure, which itself is based on Precision and Recall, the performance was examined. Running time and classification accuracy were better with C4.5 than with NB or K-Means.

2.1.4 The DDoS Attack Detection through Machine Learning and Statistical Methods in SDN, 2020

The datasets from UNB-ISCX, CTU-13, and ISOT were used in the study by [4]. The datasets were divided into attack and normal categories using the machine learning models BaysNet, J48, LR, RandomTree, and REPTree. Accuracy, Precision, and F-measure were all used in the analysis of the performance parameter. For the ISCX-SlowDDos-2016 and ISCX-IDS-2012 datasets, REPTree algorithm performed much better than the other models. The best model for the CTU-10 dataset was the RandomTree model, whereas the best model for the CTU-11 dataset was the logistic regression.

2.1.5 DDoS Attack Detection and Mitigation in SDN Using Machine Learning, 2019

[5] employed the hping3 tool in a Python script to generate both regular and DDoS packets (ICMP and TCP floods). Several ML algorithms, including J48, RF, SVM, and KNN, have been examined to identify and categorize traffic into DDoS and regular traffic, as well as to stop DDoS attacks in an SDN network. For feature extraction, the packet capture was converted to CSV format, and

unnecessary information like Address Resolution Protocol (ARP) was filtered out. Accuracy, specificity, sensitivity, Kappa, precision, recall, F-Score, and training and testing time were used to evaluate the performance matrix. J48 was shown to be the most accurate classifier.

2.1.6 DDoS Attack Detection with Feature Engineering and Machine Learning: the Framework and Performance Evaluation , 2019

[6] used the publicly accessible dataset provided by Alkasassbeh et al. for analysis. The dataset was divided into two categories, attack and normal, using the machine learning models KNN, NB, SVM, RF, and ANN. BE, the Chi-square test, and the IG Test are used for feature selection. Accuracy and Error are included in the performance metrics. With the exception of the RF model, all machine learning techniques display the highest accuracy scores on the "DS00 Full" dataset. For both SVM and ANN models, the highest AUC score of the "DS00 Full" dataset is 93.5319%. The KNN method and SVM algorithm both perform best overall, however the RF algorithm is superior at analyzing low-dimensional data.

2.1.7 Denial of Service Attack Detection through Machine Learning for the IOT, 2020

Two physical servers, one hosting the VerneMQ MQTT broker virtual machine and the other acting as the attacker, were recommended by [7] as the traffic production component for the dataset for the detection framework. Tshark was used to collect the PCAP files, which were later converted into a CSV file. In this study, the detection framework was merged with three fundamentally different machine learning algorithms, including AODE, C4.5, and decision trees (ANN). For each PCAP file, a specially developed tool extracted features like Source IP, Destination IP, Source Port, and Destination Port. The article uses the performance measures Detection Rate (DR), Accuracy, and False-Positive Rate (FPR). C4.5 outperforms all other models for features based on the 4-class classifier count, whereas AODE excels for features based on the whole dataset.

2.1.8 DDoS Attack Identification and Defense Using SDN Based on Machine Learning Method, 2018

For this paper, [8] used the KD99 dataset. The data were divided into DoS and normal categories using the SVM machine learning algorithm. Eight features were identified by combining the OPENFlow flow table's attributes. Accuracy is the performance metric used to assess the classifier's performance. The accuracy is 0.998, indicating a good recognition rate for the DDoS attack identification model.

2.1.9 Machine-learning-assisted DDoS attack detection with P4 language, 2020

In this study, [9]concentrated on TCP flood assaults. Traffic data was transmitted to DDoS attack detection modules via P4 switches. RF, KNN and SVM are taken into account as classifiers for this paper. Average Length, TCP ratio, UDP ratio, TCP-UDP ratio, and flags were features that were retrieved from the time window. Algorithm Complexity and Accuracy were the performance measures. Performance is at its greatest when using RF and SVM.

2.1.10 IoT Ddos Attack Detection Using Machine Learning, 2020

[10] used WiFi-connected camera gadgets to collect traffic data. Data was gathered both before and after botnets infiltrated devices. Prior data collection was benign. A CSV file was used to store both sets of data. The feature extraction was carried out based on the Source IP, Source IP and MAC address, Channel, and Socket highlights from the information collected from clients. Utilizing the Pearson Coefficient approach, features were chosen. Many machine learning (ML) algorithms were used, including RF, Linear SVM, NN, and J-48. Precision, Recall, F-measure, and Error were the performance

indicators that were used. It was determined that the merger of decision tree and random forest produced a high accuracy of assault detection.

2.1.11 A DDoS Attack Detection Method Based on Machine Learning , 2019

TFN2K tool was used by [11] to produce DDoS attack traffic. The tool for packet capture was Tcpdump. RF and SVM were the machine learning models that were utilized to categorize the attacks as malicious or benign. False Positive Rate, Detection Rate, and Total Detection Rate were used to analyze the detection data. The results showed that the RF detection model outperformed the SVM model.

2.1.12 Design and Implementation of IoT DDoS Attacks Detection System based on Machine Learning, 2020

[12] took into account four distinct types of typical DDOS attacks, including ICMP, SYN, and UDP floods for network data and sensor data floods. They were independently gathered by port mirroring. Domain expertise is used to pick the features. Selected features included source IP, source MAC address, protocol, timestamp, packet length, and destination IP and MAC address. In order to determine if the packets are normal or aberrant, the Decision Tree method is used. Accuracy, Precision, Recall, and F1-score are the performance metrics that were utilized to evaluate the outcomes. The accuracy and F1-score of the DT algorithm are over 97%.

2.1.13 Machine-Learning-Enabled DDoS Attacks Detection in P4 Programmable Networks, 2021

Real-time traffic data was observed and retrieved by [13] to carry out attack detection. To do feature extraction from unprocessed traffic data, utilize the DAD module. In order to speed up the procedure, P4 switches are used because they can deduce traffic features at wire speed for rapid ML submission. The deployed ML models include RF, SVM, KNN, and ANN. The performance measures Accuracy, Precision, Recall, and F1-score were used to examine the results. All metrics for the findings are over 96.6% for the KNN method and above 98.6% for the RF, SVM, and ANN algorithms. All methods have precision values above 99%, demonstrating the high reliability of the classification of positive cases when using any of the ML algorithms.

2.1.14 Denial of service attack detection and mitigation for internet of things using looking-back-enabled machine learning techniques, 2022

For this article, [14] used data from the IoT network. Using some protocols, such sflow, flow-based properties are retrieved from network traffic. Six classifiers, including DT, RF, KNN, MLP, RNN, and LSTM, were put to the test. Accuracy, F1-measure, and Kappa performance indicators were used to assess the outcomes. Without using a Looking-Back step, the KNN classifier achieved the greatest accuracy of 99.93%. Although Looking-Back-enabled Random Forest yields an accuracy of 99.81%, evaluation findings have indicated encouraging results.

2.1.15 Detection of DDoS Attacks using Machine Learning Algorithms, 2020

The dataset compiled by Mouhammd Alkasassbeh, Ahmad B.A. Hassant, Ghazi Al-Naymat, and Mohammd Almseidin include four destructive forms of attacks: UDP flood, HTTP flood, Smurf, and SIDDoS. [15] uses this dataset. It uses the classifiers NB, J48, RF, and MLP. Accuracy, Precision, and Recall are the performance criteria used to evaluate the outcomes. The J48 classifier fared better than the

other two classifiers, according to the analysis of the classifiers. J48 provided accuracy of 98.64% whereas MLP, RF, and NB provided accuracy of 98.63%, 98.10%, and 96.93%, respectively.

2.1.16 Analysis and Detection of DDoS Attacks on Cloud Computing Environment using Machine Learning Techniques, 2019

[16] used the Tor Hammer tool to create the DDoS attack in a safe setting. The attacking device used the Kali 2018.2 operating system with Kernel 4.15.0 and GNOME 3.28.0. Use of domain knowledge was made when choosing the features. The attributes include the flow's duration, type of protocol, Internet Protocol addresses (source and destination), ports (source and destination), attack classification labels, and the number of sent bytes. For the classification of data, three ML algorithms—RF, NB, and SVM—were examined and put to the test. On DDoS packets and regular packets, comparison and analysis of Accuracy, Precision, Recall, Specificity, and F-measure were performed. SVM outperformed NB and RF in terms of accuracy and precision.

2.1.17 DDoS Intrusion Detection Through Machine Learning Ensemble, 2019

The dataset NSL-KDD was utilized in this study by [17]. Four distinct classifiers—MLP (NN), SMO (SVM), IBK (KNN), and J48—from different classifier families were used (DT-C4.5). There are three different types of feature sets: traffic features computed across a two-second time window, content features within a connection given by domain knowledge, and basic aspects of individual TCP connections. Accuracy, Precision, Recall, TPR, FPR, F-measure, and ROC Area are the performance measures used to assess the outcomes. The analysis finds that, in terms of Accuracy, TPR, FPR, Precision, Recall, F-Measure, and ROC Area, the suggested ensemble model beats each and every classifier employed.

2.1.18 DDoS Attack Detection and Mitigation in SDN using Machine Learning, 2021

In this research, [18] employed a simulated dataset. For this, a Mininet emulator-based SDN topology was created. Scapy scripts were used to create both legitimate and malicious traffic. SVM, LR, DT, KNN, NB, and RF are the ML algorithms that are used. Domain knowledge was used to choose features, including source and destination IP addresses, source and destination MAC addresses, TCP/UDP source and destination ports, and transport protocol identification. Accuracy, Precision, Recall, Specificity, Training Time, and Testing Time are the performance parameters used to assess the classifiers. The findings of the experiment led to the conclusion that RF is the ideal classifier for their network. The approach was effective at promptly and correctly identifying and stopping attacks without interfering with regular flow.

2.1.19 SDN-Based Architecture for Transport and Application Layer DDoS Attack Detection by Using Machine and Deep Learning, 2021

[19] used the CICDoS2017 and CICDDoS2019 datasets for this paper. The study focused on three attacks: the TCP-SYN flood assault, the UDP flood attack, and the DrDNS attack. Domain knowledge was used to choose 49 features for the CICDDoS2017 dataset and 50 features for the CICDDoS2019 dataset, excluding the flow ID (Src IP, Src Port, Dst IP, Dst Port, and Protocol) and Label variables. In this study, KNN, SVM, RF, MLP, CNN, GRU, and LSTM classifiers were used. Accuracy, Precision, FPR, and F1-score are performance indicators that were used to evaluate the outcomes. The GRU and LSTM models retained the highest detection rates in the inline model evaluation, according to the authors'

ranking of the top models tested on the testbed. These models demonstrated remarkable robustness by achieving high detection rates, up to 95% for slow-rate attacks and above 98% for high-volume attacks.

2.1.20 DDOS Attack Identification using Machine Learning Techniques, 2021

For this paper, [20] utilized the CICIDS2017 dataset. Recursive Feature Elimination is the feature selection method employed in this paper (RFE). Machine learning classifiers include Adaboost, RF, and MLP. Accuracy, Precision, Recall, and F1-score are the performance metrics used to assess the outcomes. All other classifiers are outperformed by RF.

2.2. Discussions

One of the research gaps found was that most prior studies that used the DDoS datasets were later shown to be very uneven and less reflective of the range of real-world DDoS attacks. There are a few drawbacks that have been noticed while analyzing related work. I have chosen to use the CICDDoS2019 dataset for this work because of this. To guarantee the accuracy and quality of the data, a thorough cleaning and preparation procedure was applied to the dataset. It also offers a high level of anonymization, making it appropriate for usage in testing and research contexts.

The next research gap is that it is plausible to assume that the majority of earlier studies employed RF and KNN based on the literature analysis. Additionally, it should be emphasized that when compared to other models, the two ML algorithms offer greater accuracy and performance. As a result, the attack detection model for this project will be created using the two ML models, RF and KNN, as well as a third, less popular ML model, LR, to provide inclusivity to less popular algorithms in this domain. However, a number of the researchers did limit their feature selection for the attack detection model to just one method, namely Domain Knowledge. Because the Domain Knowledge approach is the most common, I will likewise use it. The models will be assessed using the four generally used assessment metrics, Accuray, Precision, Recall, and F1-score, based on earlier work.

3. Theoretical Framework

3.1 Pre-processing Tool

A well-liked cloud-based tool for creating and running deep learning and machine learning models is Google Colab. The platform is a great resource for preparing huge datasets like CICDDoS2019 because it offers free access to GPUs. The advantages of utilizing Google Colab to preprocess the CICDDoS2019 dataset include the following:

1. Free access to GPUs is available through Google Colab, which can greatly speed up the preprocessing of large datasets like CICDDoS2019. This is especially advantageous for computationally demanding activities like data normalization, feature extraction, and cleaning.
2. The preprocessed CICDDoS2019 dataset can be readily stored for later use by loading and saving Colab notebooks to and from Google Drive.
3. Programming environment with a wide range of support, including Python and R, makes it simple to use the tools and libraries that are best suited for the preprocessing task at hand.
4. Libraries are easily accessible through Google Colab, including TensorFlow, PyTorch, and scikit-learn, which can be utilized to carry out challenging preprocessing tasks.

In conclusion, Google Colab offers a productive, adaptable, and accessible platform for preparing the CICDDoS2019 dataset, and its features can substantially speed up and simplify the preprocessing procedure.

3.2 Feature Extraction/ Selection Techniques

3.2.1 Domain Knowledge

Data scientists can more precisely choose the most pertinent features for a given dataset thanks to domain knowledge, a type of skill. Domain expertise can assist data scientists in determining which variables are more predictive of a DDoS assault and which features may be redundant or less likely to aid in the identification of DDoS attacks in the context of a DDoS dataset. A data scientist with expertise in network traffic, for instance, would be able to determine which kinds of network traffic are most frequently linked to DDoS attacks. With this knowledge, the data scientist can exclude unnecessary features like user IP address and choose the features that are most useful for identifying a DDoS assault, including traffic volume and kind. Data scientists can improve the accuracy of their models and the quality of their outcomes by utilizing domain expertise.

3.3 Supervised Machine Learning Methods

3.3.1 Random Forest (RF)

The ensemble machine learning algorithm Random Forest is applied to classification and regression issues. To create a final forecast, the algorithm builds a lot of decision trees and then combines their predictions. The idea behind Random Forest is that a strong prediction model can be created by combining numerous weak decision trees. The Random Forest algorithm, which is a sophisticated algorithm that combines the predictions of numerous decision trees, cannot be fully described by a mathematical equation. However, a single decision tree's prediction in a Random Forest can be represented by the following formula:

$$y = f(x_1, x_2, \ldots, x_n) \qquad (3.3.1.1)$$

Where y is the prediction, $x_1, x_2, \ldots, x_n$ are the features, and f is the function that represents the decision tree. The prediction of the Random Forest is then obtained by combining the predictions of many decision trees.

3.3.2 Linear Regression (LR)

In order to predict continuous values (regression) based on a linear connection between the independent variables (features) and the dependent variable, linear regression is a widely used supervised machine learning approach (target). In order to translate the independent variables to the dependent variable, the algorithm seeks to find the optimum linear equation.

The following equation provides a mathematical representation of a linear regression model:

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + \ldots + b_n * x_n \qquad (3.3.2.1)$$

where y is the predicted target value, b_0 is the intercept, b_1, b_2, ..., b_n are the coefficients of the features x_1, x_2, ..., x_n, and x_1, x_2, ..., x_n are the independent variables.

The goal of linear regression is to find the values of b0, b1, b2, ..., bn that minimize the difference between the predicted target values and the actual target values in the training data. This is achieved by minimizing the sum of squared differences between the predicted and actual target values, also known as the mean squared error (MSE).

Different optimization procedures, like gradient descent or ordinary least squares, can be used to estimate the coefficients. By entering the values of the independent variables into the linear regression model after the coefficients have been estimated, predictions can be made on fresh, unforeseen data.

Generally speaking, linear regression is a straightforward and useful approach that may be used to solve a variety of issues, particularly when the connection between the independent variables and the dependent variable is roughly linear.

3.3.3 K-Nearest Neighbour (KNN)

A straightforward and popular supervised machine learning approach for classification and regression issues is K-Nearest Neighbors (KNN). Based on the premise that similar cases in the feature space are likely to belong to the same class, the algorithm was developed.

KNN assigns the class label that is most prevalent among its k nearest neighbors to a new, unlabeled instance in the classification setting, given a set of labeled instances. Different distance metrics, such as Euclidean distance or Manhattan distance, can be used to calculate the distance between instances.

Similar to how it operates in classification settings, KNN predicts a continuous value for the target variable when used in regression settings. This is accomplished by averaging the goal values of the k closest neighbors.

The mathematical representation of the KNN algorithm can be summarized as follows:

Given the set of labeled instances (x_1, y_1), (x_2, y_2), ..., (x_n, y_n), where x_i is a feature vector and y_i is the target variable.
For a new, unlabeled instance x *, find the k nearest neighbors using the distance metric.
For classification, assign the class label that is most frequent among the k nearest neighbors. For regression, predict the target value as the average of the target values of the k nearest neighbors.

The number of neighbors to take into account is determined by the value of the hyperparameter k, which can be modified by the practitioner. Although a bigger value of k makes the algorithm more noise-resistant, it could also cause overgeneralization. A smaller value of k, however, makes the algorithm more adaptable but may result in overfitting.

Overall, KNN is a straightforward and efficient technique that is simple to use and may be applied to classification and regression issues. Because the distance between instances must be calculated for each prediction, its biggest drawback is that it can be computationally expensive for large datasets.

3.4 Evaluation Metrics

3.4.1 Accuracy

A common performance indicator in supervised machine learning for assessing the effectiveness of a classification or regression model is accuracy. The percentage of examples that the model properly

classifies in a classification environment is known as accuracy. Accuracy in a regression context gauges how closely the values predicted and the values really occurred.

The accuracy metric is defined as follows:

$$Accuracy = (TP)/(TP + TN) * 100 \quad (3.4.1.1)$$

In conclusion, accuracy is a helpful performance indicator for assessing the effectiveness of machine learning models, particularly when the distribution of classes is balanced and accurate prediction is the aim. To fully comprehend a model's performance, it must be utilized in conjunction with other indicators, though.

3.4.2 Precision

Precision is a supervised machine learning performance statistic that assesses the proportion of true positive predictions among all positive predictions generated by the model. It is frequently applied to classification issues. In other words, accuracy assesses the proportion of positive events that the model actually identifies.

The precision metric is defined as follows:

$$Precision = (TP)/(TP + FP) \quad (3.4.2.1)$$

To acquire a complete perspective of a model's performance, precision should be utilized in conjunction with other performance metrics like recall. One performance statistic that balances precision and recall is the F1-score, which is the harmonic mean of the two metrics.

3.4.3 Recall

Recall, also known as sensitivity or true positive rate, is a performance indicator in supervised machine learning that assesses the percentage of positive examples that the model properly classifies. It is frequently used in classification issues. Recall, then, represents the proportion of genuine positive data points that the model accurately identified as such.

The Recall metric is defined as:

$$Recall = (TP)/(TP + FN) \quad (3.4.3.1)$$

3.4.4 F1-score

A performance statistic for classification tasks that strikes a balance between recall and precision is the F1-score. The single indicator of a model's performance that considers both precision and recall is the F1-score, which is the harmonic mean of precision and recall.

The F1-score is defined as follows:

$$F1 - score = 2 * (Precision * Recall)/(Precision + Recall) \quad (3.4.4.1)$$

3.5 Datasets

3.5.1 CICDDoS2019 Dataset

For testing DDoS (Distributed Denial of Service) attack detection techniques, the public can use the CICDoS2019 dataset. The dataset, which was produced by the Center for Information and Communication Technologies (CIC), contains information on different DDoS assaults, such as UDP Flood, ICMP Flood, and SYN Flood. To create a realistic simulation of DDoS attacks, data was gathered from a variety of sources, including network sensors and honeypots.

Utilizing the CICDDoS2019 dataset has a number of benefits, including:

1. Attack types: The dataset contains a variety of DDoS attacks, making it valuable for assessing how well machine learning algorithms perform under various attack conditions.
2. Realistic representation: Since the information was gathered from a variety of sources, DDoS attacks and their traits could be accurately depicted.
3. Huge and balanced: The dataset is appropriate for testing and training machine learning methods because it is large and contains more than 1 million occurrences. A same amount of normal and attack instances are present in the data, which is balanced. This lessens the likelihood that an ML model will become biased against one class.
4. Accessibility: The CICDDoS2019 is widely accessible and easy to access, making it a preferred option for scholars and professionals.

Researchers and practitioners can compare the effectiveness of their DDoS attack detection algorithms with other cutting-edge techniques by using the CICDDoS2019 dataset as a benchmark.

4. Research Methodology

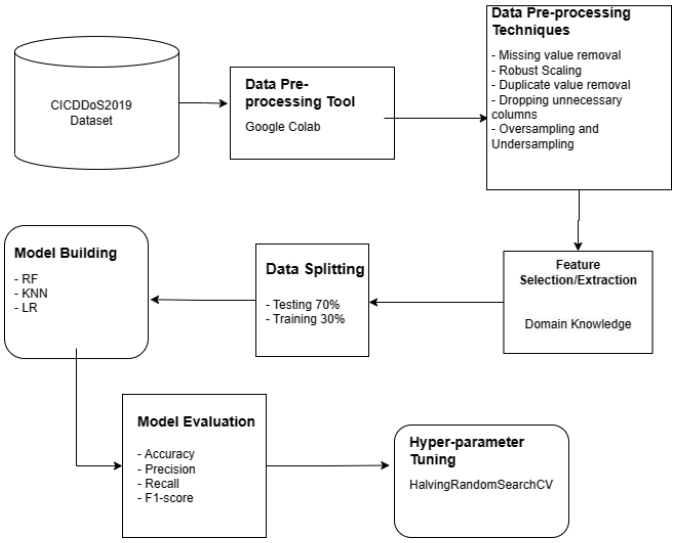

Figure 4.1 Proposed Methodology of the Project

- Step 1: Add the CICDDoS2019 dataset to Google Colab.
- Step 2: Missing and duplicate values are eliminated from the data to clean it up.. Because robust scaling is resistant to outliers, it is applied to the dataset. The

dataset's categorical variables have been encoded for usage by ML models. Any unnecessary columns of data are dropped. Oversampling and Undersampling is performed to even out unbalanced dataset.

- Step 3: Using domain knowledge, Feature Selection is carried out on the cleaned dataset. Source IP address, Destination IP address, Packet Size, Packet Rate, Protocol, Source Port, Destination Port, Flow Duration, etc. are the features that need to be chosen.
- Step 4: The dataset is divided into 30% for testing and 70% for training.
- Step 5: Use the machine learning algorithms LR, KNN, and RF.
- Step 6: Apply the three machine learning models to the train and test dataset and assess their effectiveness.
- Step 7: Analyze the findings using the accuracy, precision, recall, and F1-score performance indicators. Examine and contrast the outcomes with those from other ML models.
- Step 8: We apply hyperparameter tuning using HalvingRandomSearchCV to find the optimal values for hyperparameters.

5. Implementation

5.1 Exploring the dataset

Firstly, the dataset is imported to Google Colab, read and explored in many ways. I made use of the .info() method to print out information about the dataset. Next up, there is a usage of the .describe() method to generate the descriptive statistics like count, mean, standard deviation, minimum, 25th percentile etc. for the dataframe. Furthermore, I made use of the .nunique() method to count the number of unique elements for the data frame.

5.2 Assigning binary values

I map the value of the ‘Class’ column to a binary representation, where benign is mapped to ‘0’ and any value other than that maps to ‘1’. The results are stored in a new column ‘Binary_state’.

5.3 Data cleaning

I implemented the isnull().sum() method to check for any missing value. It was concluded that there were no missing values found for both data frames. The .duplicate() method is used to check for any duplicate values. In conclusion, no duplicate data was found. Aside from that, I will be dropping the ‘Label’ and ‘Class’ column since we don’t need the information to train and test the model.

5.4 Splitting the data into training and testing

The .drop() method is used to remove the column ‘Binary_state’ from the dataframe and assign them to X_train and X_test respectively. Also the column ‘Binary_state’ is singled out and assigned to Y_train and Y_test datasets. After that, we split the dataset into training and testing sets.

5.5 Scaling the dataset

Now I apply the RobustScaler module to scale the split datasets.

5.6 Defining the Evaluate function

I defined the function evaluate which predicts the output for the test set using Model_Abb. This function also calculates the Accuracy, Precision, F1-score and Recall.

5.7 Classification using ML algorithms

I made an instance of the RF model and made it fit on the 'X_train_scaled' and 'y_train' training data using the fit method. Then I call the function Evaluate to calculate the performance metric scores and display them. Similarly, I made an instance of the KNN model and made it fit on the 'X_train_scaled' and 'y_train' training data using the fit method. Then I call the function Evaluate to calculate the performance metric scores and display them. In the same way, I made an instance of the LR model and made it fit on the 'X_train_scaled' and 'y_train' training data using the fit method. Then I call the function Evaluate to calculate the performance metric scores and display them.

5.8 Undersampling the dataset

I made use of the value_counts() method to count the number of attack samples and benign samples. As per the output, it's worth noting that the dataset is heavily unbalanced as attack samples outweigh the benign samples. In order to tackle this issue, I used the undersampling method, where it reduces the number of 'attack' samples to be of the same number as the 'benign' samples. We reuse the value_counts() method to verify that the under-sampling of the dataframe is successful. Next we split the undersampled dataframe into training and testing sets. Here we scale the undersampled dataset using Robust Scaler. I made an instance of RF, KNN and LR models each and made it fit on the 'X_train_scaled' and 'y_train' training data using the fit method. Then I called the function Evaluate to calculate the performance metric scores of the models and display them.

5.9 Oversampling the dataset

I used the oversampling method, where it increases the number of 'benign' samples through replication to be of the same number as the 'attack' samples. We reuse the value_counts() method to verify that the oversampling of the dataframe is successful.Next we split the oversampled data frame into training and testing sets. Here we scale the undersampled dataset using Robust Scaler.I made an instance of RF, KNN and LR models each and made it fit on the 'X_train_scaled' and 'y_train' training data using the fit method. Then I called the function Evaluate to calculate the performance metric scores of the models and display them.

5.10 Hyperparameter Tuning (Original Dataset)

Lastly, we conduct hyperparameter tuning on the classifying models from the original dataset using HalvingRandomSearchCV. This finds the best set of hyperparameters for the Random Forest model. It tries different combinations of hyperparameter values that are defined in 'paramGrid_rf', and evaluates their performance using cross validation. The search process will gradually reduce the number of evaluated combinations with a factor of 5 at each iteration, helping speed up the search. After that it displays the best parameters for RF and evaluates the performance metrics of the model. Next we find the best set of hyperparameters for the K-Nearest Neighbor model. It tests various combinations of the hyperparameter values listed in "paramGrid_knn" and uses cross validation to assess how well they perform. In order to quicken the search, the number of evaluated combinations will be gradually reduced by a factor of 2 at each iteration. The best KNN parameters are then shown, and the model's performance metrics are assessed. Moving on, we find the best set of hyperparameters for the Logistic Regression model. It tests various combinations of the hyperparameter values listed in "paramGrid_lr" and uses cross validation to assess how well they perform. In order to quicken the search, the number of evaluated

combinations will be gradually reduced by a factor of 2 at each iteration. It then shows the ideal LR settings and assesses the model's performance indicators.

5.11 Hyperparameter Tuning (Undersampled Dataset)

We conduct hyperparameter tuning on the classifying models from the undersampled dataset using HalvingRandomSearchCV. We reuse the ‘paramGrid_rf’, ‘paramGrid_knn’, ‘paramGrid_lr’ from earlier. Here, the hyperparameter tuning works the same way as described earlier.

5.12 Hyperparameter Tuning (Oversampled Dataset)

We use HalvingRandomSearchCV to do hyperparameter tweaking on the classifying models from the oversampled dataset. We recycle the previously used "paramGrid_rf," "paramGrid_knn," and "paramGrid_lr." In this case, the hyperparameter adjustment functions as previously explained.

6. Evaluation of Findings

6.1 Performance result comparison for Accuracy

Dataset	RF	KNN	LR
Original	0.999	0.993	0.925
Undersampled	0.999	0.993	0.871
Oversampled	0.999	0.996	0.862

Table 6.1.1 Accuracy

Here, the results indicate that RF consistently achieved the highest accuracy across all three datasets. KNN also performed well, having high accuracy in both the original and oversampled datasets. LR had the lowest accuracy compared to the other two algorithms, with the lowest scores in both undersampled and oversampled datasets. All in all, RF seems to be the most reliable algorithm in terms of accuracy, regardless of what sampling technique was implemented. KNN also exhibits a very good performance, while LR seems to be less effective in capturing the underlying patterns in the data.

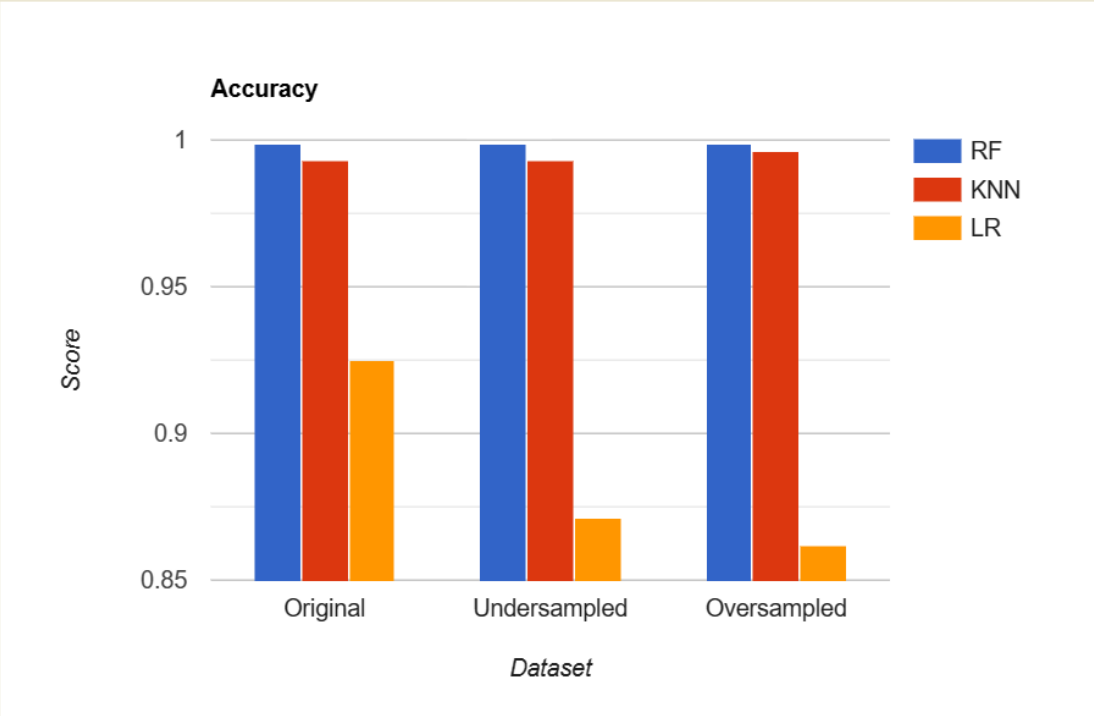

Figure 6.1.1 Accuracy Bar Graph

6.2 Performance result comparison for Precision

Dataset	RF	KNN	LR
Original	1.0	0.996	0.962
Undersampled	1.0	0.993	0.884
Oversampled	1.0	0.996	0.86

Table 6.2.1 Precision

From these results, we observe that RF consistently achieved the highest precision across all three datasets. KNN also has good performance, with high precision scores in both original and oversampled datasets. LR had the lowest precision within the three algorithms, with the lowest scores in both undersampled and oversampled datasets. Overall, RF shows the highest precision, which indicates that it has a lower rate of false positives. KNN also shows good precision, while LR relatively has lower precision scores, indicating a higher rate of false positives compared to other algorithms.

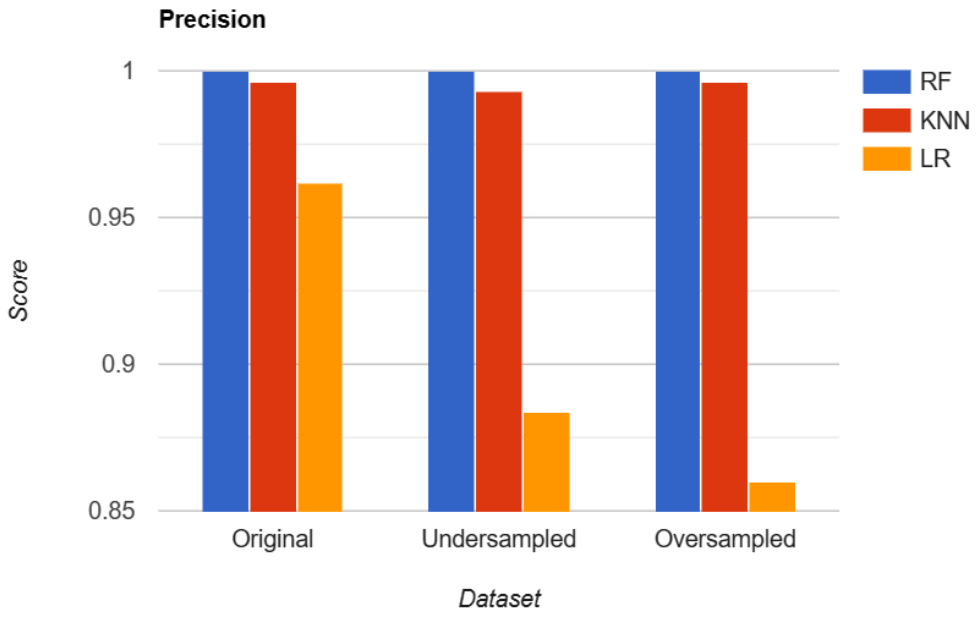

Figure 6.2.1 Precision Bar Graph

6.3 Performance result comparison for Recall

Dataset	RF	KNN	LR
Original	0.999	0.995	0.941
Undersampled	0.998	0.992	0.856
Oversampled	0.999	0.997	0.865

Table 6.3.1 Recall

These results indicate the ability of the models to correctly identify instances of DoS attacks. RF stays consistent in achieving the highest recall across all three datasets, indicating that it has a lower rate of false negatives. KNN also performs well, having high recall in the original and oversampled datasets. LR had the lowest recall compared to the other two algorithms, which indicates a higher rate of false negatives compared to the other algorithms. Overall, RF demonstrates the highest recall, which shows that it has the higher ability to correctly detect positive instances.

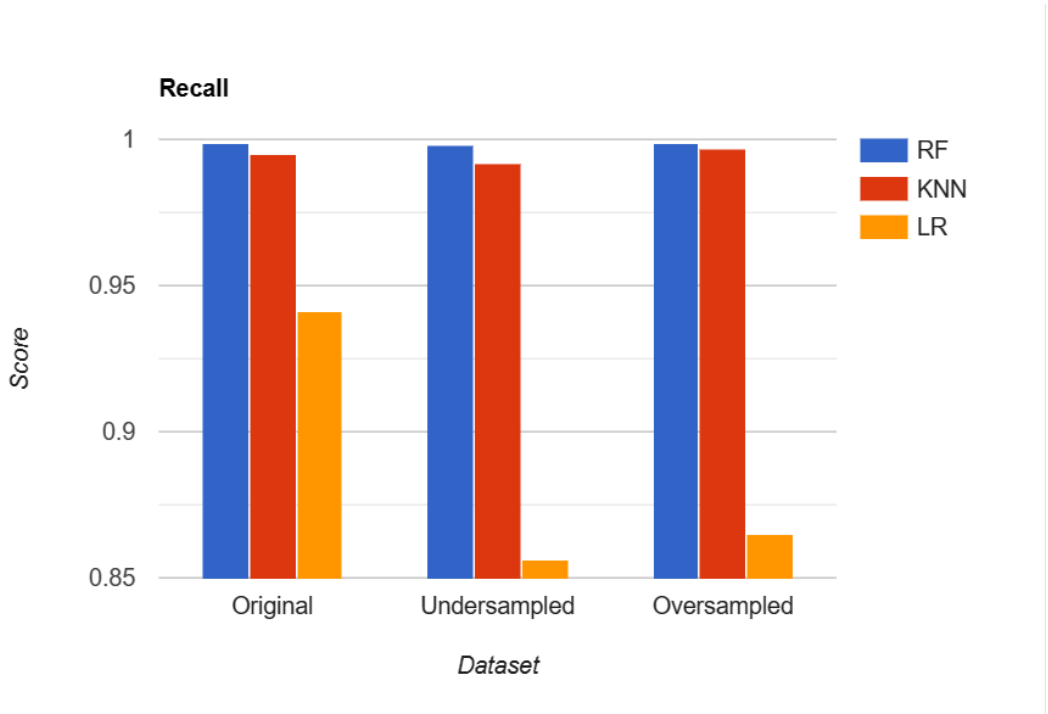

Figure 6.3.1 Recall Bar Chart

6.4 Performance result comparison for F1-score

Dataset	RF	KNN	LR
Original	1.0	0.995	0.951
Undersampled	0.999	0.993	0.87
Oversampled	0.999	0.996	0.862

Table 6.4.1 F1-Score

From observing the comparison, we gather that RF consistently achieves the highest F1-score across all the datasets, which indicates a good balance between precision and recall. KNN also has good performance, with high F1-scores in both the original and oversampled datasets. LR exhibits the lowest F1-score, which shows a trade- off between precision and recall.

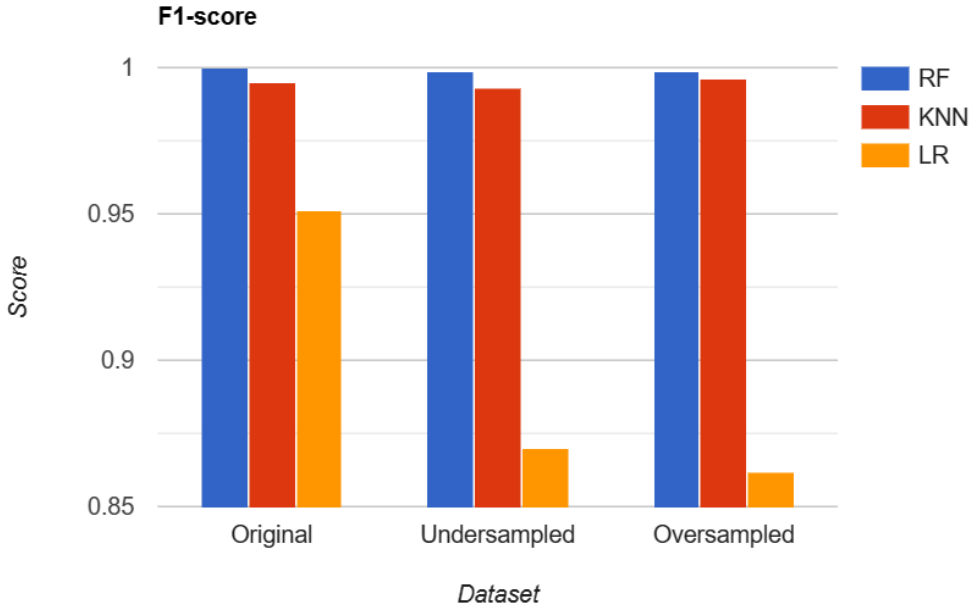

Figure 6.4.1 F1-Score Bar Graph

6.4 Performance result comparison for Hyperparameter Tuning

Algorithm	Hyperparameters	Datasets	Best Parameters

RF	{
	
	'n_estimators':[100,200,300],
	'max_depth': [None,5,10],
	'min_samples_split': [2,5,10],
	'min_samples_leaf' : [1,2,4]
	}
		Original	{'n_estimators': 100, 'min_samples_split': 5, 'min_samples_leaf': 1, 'max_depth': 5}

		Undersampled	{'n_estimators': 200, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_depth': None}
		Oversampled	{'n_estimators': 100, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_depth': None}
KNN	{ 'n_neighbors':[3,5,7], 'weights':['uniform','distance'], 'metric'	Original	{'weights': 'distance', 'n_neighbors': 5, 'metric': 'manhattan'}
		Undersampled	{'weights': 'distance', 'n_neighbors': 5, 'metric': 'manhattan'}
		Oversampled	{'weights': 'distance', 'n_neighbors': 3,

	:['euclidiean','manhattan'] }		'metric': 'manhattan'}

LR	{
	'C':[0.01,0.1,1],
	'penalty': ['l1','l2'],
	
	'solver':['liblinear','saga']
	}
		Original	{'solver': 'liblinear', 'penalty': 'l2', 'C': 0.1}

		Undersampled	{'solver': 'saga', 'penalty': 'l2', 'C': 0.1}
		Oversampled	{'solver': 'liblinear', 'penalty': 'l1', 'C': 1}

Table 6.4.1 Best parameters

The hyperparameter tuning process revealed different optimal parameter settings for each algorithm across different datasets. The best parameters were identified to achieve the highest performance in terms of accuracy, precision, recall and f1-score. These findings give us valuable insights into the impact of hyperparameter settings on model performance and help in selecting the appropriate parameters for future applications of the algorithms on similar datasets.

Datasets	Algorithm	Accuracy	Precision	Recall	F1-Score
Original	RF	0.996	0.998	0.997	0.996
	KNN	0.994	0.996	0.996	0.996
	LR	0.936	0.955	0.959	0.963
Undersampled	RF	0.413	1.0	0.389	0.241
	KNN	0.996	0.996	0.996	0.996
	LR	0.842	0.835	0.844	0.853
Oversampled	RF	1.0	1.0	1.0	0.999
	KNN	0.998	0.998	0.998	0.998
	LR	0.996	0.997	0.996	0.996

Table 6.4.2 Performance metrics (Hyperparameter tuned)

In the original dataset, RF achieved high accuracy (0.996) with a balanced precision, recall and f1-score (0.998, 0.997, 0,996). KNN also performed well with high accuracy (0.994) and balanced precision, recall and f1-score (0.996). LR achieved lower accuracy (0.936) compared to RF and KNN but still had reasonably balanced precision, recall and F1-score (0.955, 0.959, 0.963)

Looking into the undersampled dataset, the accuracy for RF dropped significantly (0.413) compared to the original dataset, the precision remained high (1.0), but the recall and F1-score were lower (0.389,0.241). KNN demonstrated consistent performance with high accuracy (0.996) and balanced precision, recall and f1-score (0.996). LR showed moderate accuracy (0.842) with reasonably balanced precision, recall and f1-score (0.835, 0.844, 0.853).

For an oversampled dataset, RF achieved perfect accuracy (1.0) with balanced precision, recall and f1-score (1.0, 1.0, 0.999). KNN maintained high accuracy (0.998) with reasonably balanced precision, recall and f1-score (0.998). LR demonstrated high accuracy (0.996) with reasonably balanced precision, recall and f1-score (0.997, 0,996, 0.996).

Overall, we can observe that RF generally performs well with hyperparameter tuning across all the datasets, with high accuracy and balanced precision, recall and f1-score. KNN too showed consistent performance. LR had lower accuracy in the original dataset, but its performance improved with the sampling techniques. These findings provide insights into the impact of different algorithms and sampling techniques on the classification task.

6.5 Overall Findings

Hyperparameter Tuned?	Datasets	Algorithm	Accuracy	Precision	Recall	F1-score
No	Original	RF	0.999	1.0	0.999	0.999
		KNN	0.993	0.996	0.995	0.995
		LR	0.925	0.962	0.941	0.951
	Undersampled	RF	0.999	1.0	0.998	0.999
		KNN	0.993	0.993	0.994	0.993
		LR	0.864	0.89	0.832	0.86
	Oversampled	RF	1.0	1.0	0.999	1.0
		KNN	0.996	0.996	0.996	0.996
		LR	0.855	0.856	0.854	0.855
Yes	Original	RF	0.996	0.998	0.997	0.996
		KNN	0.994	0.996	0.996	0.996
		LR	0.936	0.955	0.959	0.963
	Undersampled	RF	0.413	1.0	0.389	0.241
		KNN	0.996	0.996	0.996	0.996
		LR	0.842	0.835	0.844	0.853
	Oversampled	RF	1.0	1.0	1.0	0.999
		KNN	0.998	0.998	0.998	0.998

		LR	0.996	0.997	0.996	0.996

Table 6.5.1 Overall Findings

From the table, we witness the impact of hyperparameter tuning on the performance of the algorithms across different datasets. In most cases, hyperparameter tuning resulted in more improved performance, with much higher accuracy and balanced precision, recall and f1-scores. However, in some cases (e.g. undersampled dataset with RF), the tuning process hasn't led to improved results. Notably, in the non-hyperparameter tuned datasets, the RF algorithm outperformed the other two algorithms in terms of accuracy, precision, recall and f1- score

6.6 Comparison with previous works

Authors	Dataset	Accuracy of the classifier
(Dev et al., 2020)[24]	CIC-DDoS2019	0.99
(Yildiz et al., 2021)[25]	CIC-DDoS2019	0.9457
(Kumar et al., 2023)[26]	CIC-DDoS2019	0.98
Proposed Model	CIC-DDoS2019	1.00

Table 6.6.1 Previous work comparison

The table above provides an overview of the accuracy achieved by different authors. Overall, the results of the proposed work demonstrates the highest accuracy when compared to the previous works done by different authors using the CIC-DDoS2019 dataset.

7. Conclusion

One of the issues with my analysis is that the LR model doesn't perform as well as the other two algorithms when correctly classifying every event. In addition, more care should be used to choose the right tuning procedures in future studies. As for future work, the proposed model should be applied in real time. Overall, the analysis reveals that RF is the DDoS classification model that performs the best. One of the issues with my analysis is that the LR model doesn't perform as well as the other two algorithms when correctly classifying every event. In addition, more care should be used to choose the right tuning procedures in future studies. Overall, the analysis reveals that RF is the DDoS classification model that performs the best.

Acknowledgements

Big thanks to my project supervisor, Dr. Kannan Ramakrishnan, a lecturer at the Faculty of Computing and Informatics at Multimedia University, for his time, advice, and patience in working with me throughout the entire project's final year. Finally, I want to thank my family and close friends for their unfailing support.

References

[1] S. Wankhede and D. Kshirsagar, "DoS Attack Detection Using Machine Learning and Neural Network," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 2018, pp. 1-5, doi: 10.1109/ICCUBEA.2018.8697702.

[2] A, Pasumponpandian & S., Smys. (2019). DDOS ATTACK DETECTION IN TELECOMMUNICATION NETWORK USING MACHINE LEARNING. Journal of Ubiquitous Computing and Communication Technologies. 01. 33-44. 10.36548/jucct.2019.1.004.

[3] M. Zekri, S. E. Kadhali, N. Aboutabit and Y. Saadi, "DDoS attack detection using machine learning techniques in cloud computing environments," *2017 3rd International Conference of Cloud Computing Technologies and Applications (CloudTech)*, Rabat, Morocco, 2017, pp. 1-7, doi: 10.1109/CloudTech.2017.8284731.

[4] Banitalebi Dehkordi, A., Soltanaghaei, M. & Boroujeni, F.Z. The DDoS attacks detection through machine learning and statistical methods in SDN. *J Supercomput* 77, 2383–2415 (2021). https://doi.org/10.1007/s11227-020-03323-w

[5] O. Rahman, M. A. G. Quraishi and C. -H. Lung, "DDoS Attacks Detection and Mitigation in SDN Using Machine Learning," 2019 IEEE World Congress on Services (SERVICES), Milan, Italy, 2019, pp. 184-189, doi: 10.1109/SERVICES.2019.00051.

[6] Aamir, M., Zaidi, S.M.A. DDoS attack detection with feature engineering and machine learning: the framework and performance evaluation. *Int. J. Inf. Secur.* 18, 761–785 (2019). https://doi.org/10.1007/s10207-019-00434-1

[7] Alkasassbeh, M., Al-Naymat, G., Hassanat, A.B., Almseidin, M.: Detecting distributed denial of service attacks using data mining techniques. Int. J. Adv. Comput. Sci. Appl. 7(1), 436–445 (2016)

[8] Naeem Firdous Syed, Zubair Baig, Ahmed Ibrahim & Craig Valli (2020) Denial of service attack detection through machine learning for the IoT, Journal of Information and Telecommunication, 4:4, 482-503, DOI: 10.1080/24751839.2020.1767484

[9] L. Yang and H. Zhao, "DDoS Attack Identification and Defense Using SDN Based on Machine Learning Method," 2018 15th International Symposium on Pervasive Systems, Algorithms and Networks (I-SPAN), Yichang, China, 2018, pp. 174-178, doi: 10.1109/I-SPAN.2018.00036.

[10] F. Musumeci, V. Ionata, F. Paolucci, F. Cugini and M. Tornatore, "Machine-learning-assisted DDoS attack detection with P4 language," ICC 2020 - 2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 2020, pp. 1-6, doi: 10.1109/ICC40277.2020.9149043.

[11] M. H. Aysa, A. A. Ibrahim and A. H. Mohammed, "IoT Ddos Attack Detection Using Machine Learning," 2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Istanbul, Turkey, 2020, pp. 1-7, doi: 10.1109/ISMSIT50672.2020.9254703.

[12] Pei, J., Chen, Y., & Ji, W. (2019). A DDoS Attack Detection Method Based on Machine Learning. *Journal of Physics: Conference Series*, *1237*(3), 032040. doi:10.1088/1742-6596/1237/3/032040

[13] Y. -W. Chen, J. -P. Sheu, Y. -C. Kuo and N. Van Cuong, "Design and Implementation of IoT DDoS Attacks Detection System based on Machine Learning," 2020 European Conference on Networks and Communications (EuCNC), Dubrovnik, Croatia, 2020, pp. 122-127, doi: 10.1109/EuCNC48522.2020.9200909.

[14] Musumeci, F., Fidanci, A.C., Paolucci, F. *et al.* Machine-Learning-Enabled DDoS Attacks Detection in P4 Programmable Networks. *J Netw Syst Manage* 30, 21 (2022). https://doi.org/10.1007/s10922-021-09633-5

[15] Alaeddine Mihoub, Ouissem Ben Fredj, Omar Cheikhrouhou, Abdelouahid Derhab, Moez Krichen, Denial of service attack detection and mitigation for internet of things using looking-back-enabled machine learning techniques, Computers & Electrical Engineering, Volume 98, 2022, 107716, ISSN 0045-7906, https://doi.org/10.1016/j.compeleceng.2022.107716.

[16] P. S. Saini, S. Behal and S. Bhatia, "Detection of DDoS Attacks using Machine Learning Algorithms," 2020 7th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2020, pp. 16-21, doi: 10.23919/INDIACom49435.2020.9083716.

[17] A. R. Wani, Q. P. Rana, U. Saxena and N. Pandey, "Analysis and Detection of DDoS Attacks on Cloud Computing Environment using Machine Learning Techniques," 2019 Amity International Conference on Artificial Intelligence (AICAI), Dubai, United Arab Emirates, 2019, pp. 870-875, doi: 10.1109/AICAI.2019.8701238.

[18] S. Das, A. M. Mahfouz, D. Venugopal and S. Shiva, "DDoS Intrusion Detection Through Machine Learning Ensemble," 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), Sofia, Bulgaria, 2019, pp. 471-477, doi: 10.1109/QRS-C.2019.00090.

[19] F. Khashab, J. Moubarak, A. Feghali and C. Bassil, "DDoS Attack Detection and Mitigation in SDN using Machine Learning," 2021 IEEE 7th International Conference on Network Softwarization (NetSoft), Tokyo, Japan, 2021, pp. 395-401, doi: 10.1109/NetSoft51509.2021.9492558.

[20] N. M. Yungaicela-Naula, C. Vargas-Rosales and J. A. Perez-Diaz, "SDN-Based Architecture for Transport and Application Layer DDoS Attack Detection by Using Machine and Deep Learning," in IEEE Access, vol. 9, pp. 108495-108512, 2021, doi: 10.1109/ACCESS.2021.3101650.

[21] S. Peneti and H. E, "DDOS Attack Identification using Machine Learning Techniques," 2021 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2021, pp. 1-5, doi: 10.1109/ICCCI50826.2021.9402441.

[22] D. Hesham (2022) Network Security Data Preprocessing + Classification [Source code]. https://www.kaggle.com/code/daliahesham/network-security-attack-classification/notebook

[23] Talukder, Md Alamin; Uddin, Md Ashraf (2023), “CIC-DDoS2019 Dataset”, Mendeley Data, V1, doi: 10.17632/ssnc74xm6r.1

[24] M. S. Elsayed, N. -A. Le-Khac, S. Dev and A. D. Jurcut, "DDoSNet: A Deep-Learning Model for Detecting Network Attacks," 2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), Cork, Ireland, 2020, pp. 391-396, doi: 10.1109/WoWMoM49955.2020.00072.

[25] Cil, A. E., Yildiz, K., & Buldu, A. (2021). Detection of DDoS attacks with feed forward based deep neural network model. *Expert Systems with Applications*, *169*, 114520. https://doi.org/10.1016/j.eswa.2020.114520

[26] Kumar, D., Pateriya, R. K., Gupta, R. K., Dehalwar, V., & Sharma, A. (2023). DDoS Detection using Deep Learning. *Procedia Computer Science*, *218*, 2420–2429. https://doi.org/10.1016/j.procs.2023.01.217