Arab American University – Palestine
Faculty of Graduate Studies

Combining Multiple Supervised Machine Learning Algorithms to Detect Network Intrusion Attempts

By

Helmi Naem Al haji

Supervisor

Dr. Mustafa Abusalah

This thesis was submitted in partial fulfillment of the requirements for the Master`s degree in Data Science and Business Analytics

August 2021

I

Combining Multiple Supervised Machine Learning Algorithms to Detect Network Intrusion Attempts

By

Helmi Naem Al haji

This thesis was defended successfully on 14/08/2021 and approved by:

Committee members	Signature
1. **Mustafa Abusalah, Supervisor**	
2. **Mohammed Maree, Internal Examiner**	
3. **Ayser Armiti, External Examiner**	

Declaration

I declare that this thesis has been composed solely by me and that it has not been submitted, in whole or in part, in any previous application for a degree. Except where states otherwise by reference or acknowledgment, the work presented is entirely my own.

Name: Helmi Al Haji

Signature: 

Date:11/1/2022

Abstract

The importance of using machine learning algorithms has emerged in many areas, including intrusion detection for networks and cyber systems. Conventional network monitoring systems approaches to detecting anomalies can be classified into two main categories: misuse detection and anomaly detection. Misuse detection (signature based) methods are intended to recognize known attack patterns in packets that matches their database of signatures representing known security threats. While Anomaly detection (behavior based) focuses on detecting unusual activity patterns. This is done by establishing a user behavior profile over a period, then certain parameters are stored, where any unfamiliar behavior appears from the user is considered anomaly.

However, these systems suffer from producing a high rate of false alarms. Therefore, this study presents an improved approach utilizing combined multiple supervised machine learning classification methods, Random Forest, Gradient Booting Trees, K-nearest neighbor, and Logistic Regression, to increase the likelihood of the accurate detection of anomaly events, decreasing the rate of false-positive alarms.

Contents

Table of Equations

Table of Figures

Table of Tables

1 Introduction

In the world of security, Intrusion Detection Systems (IDs) can be decomposed into two main categories, misuse detection and anomaly detection. Misuse detection methods are intended to recognize known attack patterns, therefore, misuse detection has a high false negative rate and a low false positive rate (Sen Jaydip, Mehtab Sidra, 2020). In cybersecurity systems that follow a signature-based approach, malicious events are identified by matching events against predefined signature (Kruegel C., Toth T, 2003). While anomaly detection (behavior based) focuses on detecting unusual activity patterns. This is done by establishing a user behavior profile over a period then certain parameters are stored, and any unfamiliar behavior appears from the user is considered as an attack (Sen Jaydip, Mehtab Sidra, 2020).

However, these systems suffer from a major drawback of producing a high rate of false alarms. False positive or false alarms refers to a situation where the system raises an alert of an attack while no attack happened (Sen Jaydip, Mehtab Sidra, 2020).

In this research, we will combine the results of multiple supervised machine learning algorithms to increase the accuracy of the intrusion traffic detection and therefor decreasing the rate of false-positive alarms.

The outcome will be evaluated against a ground-truth of attack and non-attack patterns or cases and also compared in term of performance with respect to other previously developed methods (Fares Ahmed H., Sharawy Mohamed I., 2011).

We will use five supervised machine learning classifiers for the anomaly detection and combine the results of each method, Logistic Regression, Random Forest classifier, Gradient

Booting Trees and, K-Nearest Neighbor, as combining the results assumed to increase the likelihood of the accurate detection of malicious events if compared with the results of a particular method (Belavagi Manjula C., Muniyal Balachandra, 2016).

In the literature review, we can find many kinds of literature comparing the results of supervised machine learning techniques in intrusion detection processing, while literature discussing combining results of multiple algorithms is limited. Therefore, this research will examine the results of the combining approach on the detection of the misuse attempts. (Belavagi Manjula C., Muniyal Balachandra, 2016).

1.1 Background

1.1.1 Intrusion Detection System

Computer systems are vulnerable to several possible attacks due to the errors in system programs and faulty design of the software. Malicious attackers can abuse these systems and cause significant damage to data and functionality. Cybersecurity is the practice of defending computer and information systems from malicious attacks (Solms Rossouw von, Niekerk Johan van, 2013).

There are several types of intrusion detection techniques: Host-based detection and Network-based detection. Host based intrusion detection is used to monitor individual host and employed to monitor the incoming and outgoing traffic by comparing the results with a pre-created image of the flow of the host's activity. Usually, we can do this by having a software agent that detects the intrusions by analyzing system activities and logs.

While network-based intrusion detection is done by examining network traffic by monitoring multiple hosts to detect any suspicious activity, this is done by analyzing the network, transport, application, and hardware layer protocols within the captured network traffic (Sen Jaydip, Mehtab Sidra, 2020).

1.1.2 Machine Learning Classifier

Supervised Machine Learning is a process of use of labeled datasets to train algorithms to classify data or predict outcomes accurately. The classifier is trained with a training dataset that is categorized to assist the model to classify the input to belong to one or more categories (Kotsiantis, S. B. Zaharakis I., Pintelas P., 2006).

Several machine-learning methods have been proposed to analyze network traffic for different anomalies. Most of these methods (classifiers) recognize the anomaly by looking for disparities from a basic normal traffic model. Usually, these models are trained with a set of clean traffic data that is collected over a long period. All machine-learning of anomaly detection methods are one of three broad categories: Supervised, Unsupervised, or Semi-supervised learning method. In this research, we will focus on the supervised learning classifiers.

Supervised learning is the type of model that takes both of input variable (X) and output variable (Y) to provide a learning basis to support future judgments by learning the mapping function $Y = f(X)$. Supervised learning uses training data that is a set of real cases with paired input records and their intended outputs. In this type of learning, the correct answer is known in advance. The learning algorithm iteratively makes predictions on the training data and

stops when it achieves an acceptable level of performance (Kotsiantis, S. B. Zaharakis I., Pintelas P., 2006). Therefore, this method is applicable when there is a specific target value. A supervised learning problem can be defined as either a classification problem or a regression problem. The output variable of the classification problem is a category (Discrete Value), like “white” or “black” and “Yes” or “No”. On the other hand, the output result of the regression problem is a real value (Continues Value), such as “cost in currency” or “the weight”. The most famous supervised learning algorithms are Decision Trees (DT), Support Vector Machines (SVM), Artificial Neural Network (ANN), k-nearest neighbors (k-NN), Logistic Regression (LR), Random Forest (RF) and Gradient Booting Trees (GB).

1.2 Performance Evaluation

Different performance metrics are used to assess the results of machine learning algorithms in classification problems. Commonly used metrics with Machine-Learning-based Intrusion Detection Systems:

1.2.1 Confusion Matrix

Metrics that determine the accuracy of machine-learning models called sensitivity and specificity measures. Sensitivity (Equation *1*) is referred to as the true positive rate (TPR), while specificity (Equation *2*) is referred to as the true negative rate (TNR) (Belavagi Manjula C., Muniyal Balachandra, 2016).

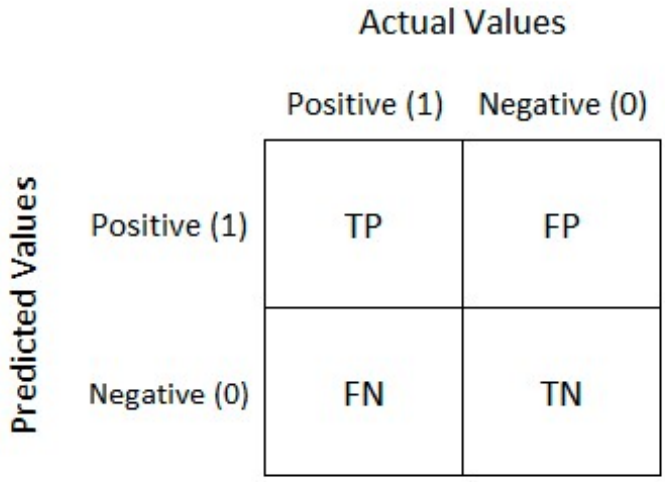

Figure 1: Confusion Matrix

- True positive (TP): The event is “attack”, and the model predicts as “attack”.
- True Negative (TN): The event is “benign”, and the model predicts as “benign”.
- False Positive (FP): The event is “benign”, and the model predicts as “attack”.
- False Negative (FN): The event is “attack”, and the model predicts as “benign”.

$$\text{Sensitivity} = \frac{\text{TP}}{(\text{TP} + \text{FN})}$$

Equation 1: Sensitivity

$$\text{Specificity} = \frac{\text{TN}}{(\text{TN} + \text{FP})}$$

Equation 2: Specificity

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

Equation 3: Accuracy

1.2.2 Recall, Precision and F1 Score

Precision and recall are evaluation metrics in the information retrieval area and classification. Precision (Equation *4*) refers to the fraction of the relevant instances among the retrieved instances. Recall (Equation *5*) refers to the fraction of relevant retrieved instances from the total number of the relevant instances. F-measure (Equation 6) is the precision and recall harmonic mean. Average precision score (Equation 7) summarizes a precision-recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight. (Abusalah Mustafa A., 2008).

$$\text{Precesion} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Equation 4: Precision

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Equation 5: Recall

$$\text{F1 Score} = \frac{2(\text{precesion} * \text{recall})}{(\text{recall} + \text{precesion})}$$

Equation 6: F1 Score

$$\text{AP} = \sum_{n} (R_n - R_{n-1})\, P_n$$

where R_n and P_n are the precision and recall at nth threshold

Equation 7: Average Precision Score

1.2.3 ROC Curve

Receiver Operating Characteristics (ROC) curve is an evaluation measure that visualizes the relation between True Positive (TP) and False Positive (FP) rates of IDs. It is also used to compare between two or more machine-learning classifiers in terms of their accuracy. (Kelly H. Zou, A. James O'Malley, Laura Mauri, 2006).

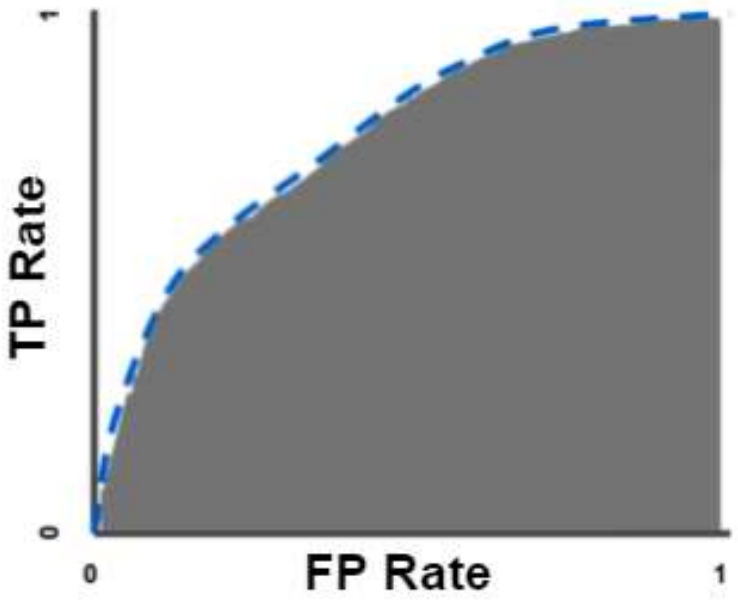

Figure 2: AUC (Area under the ROC Curve).[1]

[1] *Figure source: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

2 Dataset

Numerous contributions have been made for handling security-related data by building representative network-based datasets that provide a good basis for evaluating and comparing various network intrusion detection systems (NIDS); A labeled dataset in which each data point is assigned to the class normal or attack. unfortunately, there are no many representative datasets around.

The first intrusion detection dataset of DARPA was created in 1998 by the MIT Lincoln Laboratory under the DARPA-funded project. Later in 1999, the tcpdump files of DARPA were refined and processed by the researchers of the University of California to form a new dataset of the KDD Cup 1999. The new dataset consists of 4,898,430 training set records and 311,027 records as test set, with 41 features for each attack category, but this dataset suffers from many duplicate records (Tavallaee Mahbod, Bagheri Ebrahim, Lu Wei, Ghorbani Ali A, 2009).

One of the latest and realistic cyber datasets has been founded by the Canadian Institute for Cybersecurity (CIC) that contains benign and the most up-to-date common attacks; CIC-IDS-2017 and CSE-CIC-IDS-2018 (Ankit Thakkar, Ritika Lohiya, 2019).

CIC-IDS-2017 and CSE-CIC-IDS-2018 are datasets that consists of seven attack situations: Botnet, Heartbleed, Denial of Service, Distributed Denial of Service and Brute-force, inside network infiltration, and Web attacks. The Datasets are publicly freely available on the Canadian Institute for Cybersecurity (Canada's Cybersecurity Hub, n.d.) (Iman Sharafaldin, Arash Habibi Lashkari, Ali A. Ghorbani, 2018).

Later in 2018, during a collaborative project between the Communications Security Establishment (CSE) & the Canadian Institute for Cybersecurity (CIC), The CSE-CIC-IDS2018 was prepared from an extensive network of simulated client targets and attack machines. The new dataset contains 16,233,002 records gathered over ten days of network traffic with 80 features. Nearly 17% of the instances are attacking traffics. The attacking infrastructure includes 50 machines, and the victim organization has five departments and includes 420 machines and 30 servers. The dataset contains the captures network traffic and system logs of each machine, along with 80 features extracted from the captured traffic using CICFlowMeter [2] (Canada's Cybersecurity Hub, n.d.).

Table *1* shows the percentage of distribution of the seven types of network traffic represented by CSE-CIC-IDS2018.

Table 1: Distribution per Attack Type

Traffic type	Distribution (%)
Benign	83.07
DDoS	7.786
DoS	4.031
Brute force	2.347
Botnet	1.763
Infiltration	0.997
Web attack	0.006

[2] *CICFlowMeter is a network traffic flow generator and analyzer

The CSE-CIC-IDS-2018 dataset encountering some issues as it consists of some missing and redundant data records which are prone to a high-class imbalance that may result in low accuracy and high False Positive Rate (FPR) of the system. These issues can be handled by preprocessing the dataset, applying feature engineering, or eliminating the missing records. In addition, the high-class imbalance can be overcome by relabeling or resampling the data samples, in turn, this would increase the probability of occurrence of data samples of all classes (Ankit Thakkar, Ritika Lohiya, 2019).

3 Algorithms

The supervised algorithms that will be used in this study are the following.

3.1.1 Support Vector Machine (SVM)

SVM is one of the commonly used supervised ML algorithms. SVM can be used for both classification and regression by training with the labeled data. It can output the separation of data into classes by the hyperplane that maximizes the margin among all attack classes. SVM, as a binary classifier, will also perform multi-class classification by using a cascade manner. SVM mainly depends on the types of kernels used and parameters (Sen Jaydip, Mehtab Sidra, 2020). However, one of the main disadvantages of SVM is getting slow when the dataset size is of a large-scale.

3.1.2 Logistic Regression (LR)

LR is a supervised machine-learning classification algorithm used to observe the discrete set of classes. This type of statistical analysis is used for predictive analytics and modeling, whether the dependent variable is finite or categorical. It measures the association between the dependent variable and one or more independent variables by estimating probabilities using a Logistic Regression equation (Where $S(x)$ is an output between 0 and 1, x is an input to the function, and e is a base of natural log.

Equation 8) (Kalidindi Venkateswara Rao, G. Sharmila Sujatha2, 2019).

$$S(x) = \frac{1}{1 + e^{-x}}$$

Where $S(x)$ is an output between 0 and 1, x is an input to the function, and e is a base of natural log.

Equation 8: Sigmoid Function

3.1.3 Decision Tree (DT)

The key point behind the learning approach of the Decision Tree is a greedy algorithm, which uses the recursive top-down approach of the Decision Tree structure. Several Decision Tree algorithms have been developed with improved performance and the ability to handle several data types. Decision Tree are commonly used in the implementation of intrusion detection use cases. In Decision Tree, the major challenge is to identify the attribute of the root node in each level. This process is known as attribute selection and the Gini index measure (Equation *9*) is one of the methods used in Decision Tree algorithms to decide the optimal split from a root node and subsequent splits. (Brijain R Patel, Kushik K Rana, 2014).

3.1.4 Gradient Booting Trees (GB)

Gradient Booting Trees is a machine learning algorithm that produces a prediction model in an ensemble of weak prediction models, typically decision trees. The concept of boosting is to reconstruct a feature that is not strong in predicting outcome Y (weak learner) into a strong learner. After identifying weak learners in each regression tree, boosting will give equal weight to each observation while focusing on the errors between the fitted values predicted by the weak learner and the actual values. Y is assumed to be a real value (Fares Ahmed H., Sharawy Mohamed I., 2011).

3.1.5 Random Forest (RF)

RF is an ensemble supervised algorithm used for classification and regression. RF will construct many decision trees during the model's training, and the outcomes of predictions from all trees are pooled to make a result, therefore, it is usually referred to as ensemble learning. RF classifiers work as follows: the higher the number of trees in the model, the higher the accuracy; preventing overfitting of the model. (Sen Jaydip, Mehtab Sidra, 2020). RFs use Gini index (Equation 9) to decide how many nodes on a decision tree branch.

$$\text{Gini} = 1 - \sum_{i=1}^{c} (\text{pi}) .^2$$

where *pi is* the probability of an object being classified to a specific class.

Equation 9: Gini Index

3.1.6 k-nearest neighbors (k-NN)

The K-nearest-neighbor (k-NN) method is a classification method that uses a voting score amongst all the neighbors for a given data point in finding its class membership (Sen Jaydip, Mehtab Sidra, 2020). The k-NN classifier is based on the assumption that the classification of an instance is most similar to the category of other nearby instances in the vector space. Thus, k-NN does not rely on prior probabilities compared to other classification methods (Rao V. Vemuri, 2002).

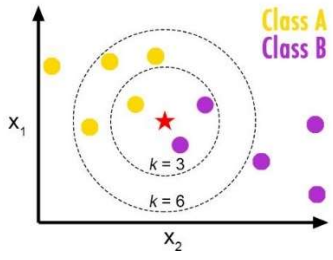

Figure 3: k-NN - How it works [3]

4 Related Works

This section discusses related work, three main baseline datasets in the field, KDD Cup 1999, CIC-IDS2017 and CSE-CIC-IDS2018.

The researchers in (Serpen, Gursel, Sabhnani, Maheshkumar, 2003) have tested nine distinct pattern recognition and ML algorithms using KDD Cup 1999 dataset. They aimed to represent a wide variety of fields: neural networks, probabilistic models, statistical models, fuzzy-neuro systems, and Decision Trees. They executed all algorithms to detect four different attack categories, Probe (attempt to gain access to a computer through a known weak point in the system), DoS (denial-of-service), U2R (User to root attack), R2L (Remote to local). Their results show that a single algorithm could not detect all attack categories with a high probability of detection, whereas specific algorithms demonstrate superior detection performance than others. For example, MLP, GAU, K-M, NEA, and RBF detected more than 85% of attack records for the probing category. MLP, K-M, NEA, LEA, and HYP scored a 97% detection rate for attack records in the DoS category. GAU and K-M, the two most successful classifiers for the U2R category, detected only around 22% of attack records.

In (Laskov Pavel, Düssel Patrick, Schäfer Christin, Rieck Konrad, 2005), the researchers applied supervised and unsupervised algorithms to detect known and unknown attacks. They used six different supervised algorithms for the classification. The supervised algorithms, in

[3] *Figure source: https://helloacm.com/a-short-introduction-to-k-nearest-neighbors-algorithm/

general, show excellent accuracy for the classification of the data of known attacks. The algorithm that achieves the best results is the C4.5, and the rate was 95% as true positive rate and only 1% false positive rate. The next best algorithms are Multi-Layer Perception neural network classifiers (MLP), Support Vector Machines (SVM) and k-nearest neighbors (k-NN)' algorithm; the difference between them is marginal. The remaining three algorithms produced the worst results.

(Ali Reza Zebarjad, Mohmmad Mehdi Lotfinejad, 2012) combined the results of three algorithms in the final decision of the classification. Methods used in the experiment are Multi-Layer Perception neural network classifiers (MLP), Support Vector Machines (SVM), and Decision Trees (DT). They compared the accuracy of detection of each method to the accuracy of detection of combining the results of the three methods. The accuracy of the combining approach increased for certain types of attacks as in Probe, while the performance decreased in others, as in U2R attacks.

(Manjula C. Belavagi, Balachandra Muniyal, 2016) used four supervised machine-learning algorithms and compared their results: Logistic Regression, Gaussian Naive Bayes, Support Vector, and Random Forest classifier using NSL-KDD dataset. Results were represented by a reliability curve where estimated probabilities are plotted against the true empirical probabilities. The Random Forest classifier outperformed the other methods in identifying the network traffic where SVM identified the attacks with the lowest probability estimate. Support Vector Machine has the highest FPR (39%) and lowest TPR (75%) for intrusion detection.

Referring to the performance measurement of precision, recall, F1 Score, and accuracy of the supervised machine learning classifiers, the Random Forest classifier has the highest prediction results and outperforms the other methods with 0.99 for all measurements. SVM has the lowest prediction results of 0.76,0.79, 0.77 for precision, recall, and F1 Score, respectively. The Logistic Regression Algorithm has a good accuracy than Gaussian Naive Bayes and SVM.

In (Chibuzor John Ugochukwu, E. O Bennett, 2018) , the researchers used the KDD Cup 1999 datasets. They have used the following classification algorithms in their experiment: (Bayes Net, J48, Random Forest, and Random Tree). Researcher used Correlation-based approach for feature selection, and they have used the precision, recall, and F-measure as performance measurement metrics. The experimental results showed that the Random Forest and Random Tree algorithms achieved the best results in the classification process with precision and recall 97.7, 87.5 respectively.

The work proposed by (Kazi Abu Taher, Billal Jisan, Md. Rahman, 2019) is a composite of feature selection and learning algorithm. Researchers have used the filter method and wrapper method for feature selection. They have used two algorithms for the learning process, Support Vector Machine (SVM) and Artificial Neural Network (ANN).

With the features found in the feature selection part, they have built four models. They have used 20% of the KDD Cup 1999 dataset as training data that have 25,191 labeled data instances. To train the model, they have used SVM and ANN learning algorithms for each type of feature selection method (two models using SVM and another two using ANN). Two

models use 17 features, and another two use 35 features found in the feature selection part. The ANN model was experimented with different numbers of hidden layers, and they found that the detection success rate varies by the number of hidden layers. In comparison, the best detection rate was with three hidden layers. The analysis of the result of their four models showed that the ANN model with wrapper feature selection outperformed all other models in the network traffic success classification with a detection rate of 94.02%.

In (Laurens D'hooge, Tim Wauters, Bruno Volckaert, Filip De Turck, 2020), the authors experimented two different datasets, CIC-IDS2017 and CSE-CIC-IDS2018. They aimed to examine how efficiently the results of intrusion detection datasets can be generalized. They have used 12 supervised algorithms: Decision Tree (DT), Random Forest (RF), DT-based Bagging, Gradient Boosting Trees, Extratree, Adaboost, XGBoost, k-nearest neighbor (k-NN), Ncentroid, linearSVC, RBFSVC, and Logistic Regression.

Attributes normalized by feature scaling, the best-perform classifiers are the tree-based, and the XGBoost. The highest accuracy, precision, and recall scores for CSE-CIC-IDS2018 were 96%, 99%, and 79%, respectively. The authors concluded that the model trained on one dataset (CIC-IDS2017) could not generalize to another dataset (CSE-CIC-IDS2018).

In (V. Kanimozhi, Dr. T. Prem Jacob, 2019) the author using the CSE-CIC-IDS2018 dataset, incorporated the following classification methods K-nearest neighbor, Naïve Bayes, Adaboost with Decision Tree, Support Vector Machine classifier and Random Forest classifier to distinguish a portrayal of botnet attacks.

The accuracy of all classifiers is very high, and the supervised classifier of the highest accuracy was Adaboost with accuracy, precision, recall, and F1 Score of 0.9996, 0.9996, 0.9988, 0.9992, respectively.

The study did not mention any action for preprocessing nor feature selection process on the dataset. Also, the study did not note any action to solve the known issue of imbalanced distribution of attack, even though it aims to detect the attack type of botnet.

In (Marta Catillo, Massimiliano Rak, Umberto Villano, 2020), the authors present a deep learning anomaly detector for multiple attack classes. they were able to achieve 95.82% accuracy for DoS attacks on the CICIDS2017 dataset and showed the possibility to use the autoencoder model to detect DoS attacks not being in the training set, such as 0-day attacks.

The training performed using 83 features of 85 in the acquired datasets. The features Flow_ID and Timestamp were dropped during the preprocessing stage as they were considered non-relevant to the study. The splitting ratio for the training and testing dataset was 80-20. The implementation was done using Python, Keras, and TensorFlow.

For the CSE-CIC-IDS2018 dataset, the highest accuracy achieved is 99.2% for the botnet attack type, with 95.0% and 98.9% for precision and recall. The accuracy of detection for the same attack type on CIC-IDS2017 was 99.3%, with 94.8% for precision and 98.6% for recall.

The (Fitni QRS, Ramli K., 2020) used an ensemble model approach, the study compared seven single models to evaluate the top performers for the integration into a one classifier

unit using CSE-CIC-IDS2018 dataset. The seven models are Gradient Boosting Trees (GB), Random Forest (RF), Gaussian Naive Bayes (GNB), Decision Tree (DT), Logistic Regression (LR), and Quadratic Discriminant.

The implementation of models was with Python and Scikit-learn. Preprocessing actions were taken place by imputing the missing values and infinity by removing these instances. In addition, the duplicate headers rows were deleted.

The sample dataset was divided with 80-20 ration into training and testing sets. Feature selection techniques to select the strongest features for prediction were performed using Spearman's rank correlation coefficient and Chi-squared test. They were resulting in selecting only 23 features from 85 in the original dataset.

The models of Gradient Booting Trees, Logistic Regression, and Decision Tree achieved the highest performance among all models. Then these models emerged in the ensemble model using the majority voting approach in which every classifier produces its prediction. The voting model then chooses the highest number of prediction results that produce on each classifier. The results were as follows: accuracy, precision, and recall scores are 98.80%, 98.80%, and 97.10%, respectively, along with an Area Under the Receiver Operating Characteristic Curve (ROC) 0.94.

The researchers confirm that the implementation of the ensemble method and feature selection increases the performance and accuracy of machine learning based IDSs. As the accuracy was 98.8%, and the time of detection reduced from 34 min 2 sec to 10 min 54 sec.

In (Sen Jaydip, Mehtab Sidra, 2020) study, the researchers have discussed the categories of intrusion detection systems, the misused detection or signature-based and anomaly detection approach. They have discussed examples for machine-learning algorithms that used in both categories.

They proposed several algorithms for the misused-based IDS problems, Artificial Neural Networks (ANN), Support Vector Machines (SVM), Decision Tree and Naïve Bayes.

For anomaly-based IDS problem, researchers proposed several algorithms and approaches, some of them already used for misused-based. Algorithms are Artificial Neural Networks (ANN), Support Vector Machines (SVM), k-nearest neighbor (k-NN), Random Forest (RF). Researcher specific that network-wide anomaly detection using principal component analysis (PCA) has shown remarkable performance.

5 Methodology and Design

The proposed methodology includes the following stages: data collection, pre-processing, exploratory data analysis, modeling, and comparison of different classifiers for anomaly classification.

5.1 Data Collection

A publicly available dataset named CSE-CIC-IDS2018 will be used in the experiments. CSE-CIC-IDS2018 was prepared from an extensive network of simulated client targets and attack machines, resulting in a dataset that contains 16,233,002 records gathered over ten days of network traffic with 80 features. Dataset is available in 10 CVS files, where each file consists of a simulation a certain attack (Markus Ring, Sarah Wunderlich, Deniz Scheuring, Dieter Landes, Andreas Hotho, 2019).

5.2 Data Exploration and Pre-processing

As we mentioned before, the dataset is raw and will require preprocessing work to be carried out before starting the modeling stage. Following is the description of each step:

5.2.1 Modify/Delete Headers and Columns Names

The headers are duplicated, therefore the first step we have done is removing the headers from the loaded files.

To make column names more readable and easier in further processing, we have renamed some columns by removing whitespaces and non-word characters. Below in Table *2* list all renamed columns:

Table 2: List of Renamed Columns

Original Name	Description	New Name
bwd_byts_b_avg	Average number of bytes bulk rate in the backward direction	Bwd Pkts/b Avg
bwd_pkt_len_mean	Mean size of packet in backward direction	Flow IAT Std
flow_iat_mean	Average time between two flows	Fwd IAT Mean
fwd_byts_b_avg	Average number of bytes bulk rate in the forward direction	Fwd Pkts/b Avg
fwd_iat_min	Minimum size of packet in forward direction	Bwd IAT Min
fwd_iat_tot	Total time between two packets sent in the forward direction	Bwd IAT Tot
fwd_pkt_len_max	Maximum size of packet in forward direction	Bwd Pkt Len Max
fwd_pkt_len_std	Standard deviation size of packet in forward direction	Bwd Pkt Len Std
fwd_pkts_s	Number of forward packets per second	Bwd Pkts/s
fwd_urg_flags	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)	Bwd URG Flags
pkt_len_mean	Mean length of a flow	Pkt Len Std
pkt_size_avg	Average size of packet	Fwd Seg Size Avg
syn_flag_cnt	Number of packets with SYN	RST Flag Cnt
tot_fwd_pkts	Total packets in the forward direction	Tot Bwd Pkts
urg_flag_cnt	Number of packets with URG	CWE Flag Count

In addition, we found that one of the CSV files (Thuesday-20-02 2018_TrafficForML_CICFlowMeter.csv) consists of extra columns, and since their values in the other files will be null, we dropped these columns, as they will negatively affect the analyses.

5.2.2 Missing, NULL and Negative Values

Data Imputing involves representing the missing values in the acquired dataset. There are multiple methods for substituting missing values. One of these methods is mean imputation by replacing missing values of a particular variable with the mean of non-missing cases of that variable. It is suitable for numerical features in the dataset, and it's easy and fast to implement (Sonny Rosenthal, 2017).

Null values do not exist in the dataset; however, columns named "flow_byts_s" and "flow_pkts_s" contain "infinity" values in some rows. "Infinity" is replaced by the mean of each column. Eleven columns contained negative values that were replaced by the mean of the value under each column as well. However, infinity could be replaced with a maximum value while negative value could be replaced with the minimum value, but since the number of records which have such issues are small, for simplicity we choose to impute with the mean value.

5.2.3 Exploratory Data Analysis

Exploratory data analysis is the first thing to do with any dataset to become familiar with (Victoria Cox, 2017). In this section, we aim to answer the following questions:

- How many benign and malicious network flows exist in the dataset?
- How many network flows per attack type?
- Is there any strong correlation between certain features?
- Which features show a correlation with the target variable of a network flow?

5.2.4 Data Analysis and Visualization

Records in the dataset were distributed into two types of records: benign (normal transactions) and attacks, whereas attacks consist of various kinds. 13,484,235 records are benign, representing around 83% of the whole dataset records, while attacks are only 17% with 2,748,235 records, which represents a regular network traffic distribution. Plotting the numbers of benign network flows versus malicious network flows shows that the dataset is heavily skewed toward benign network flows as shown in

Figure *4*

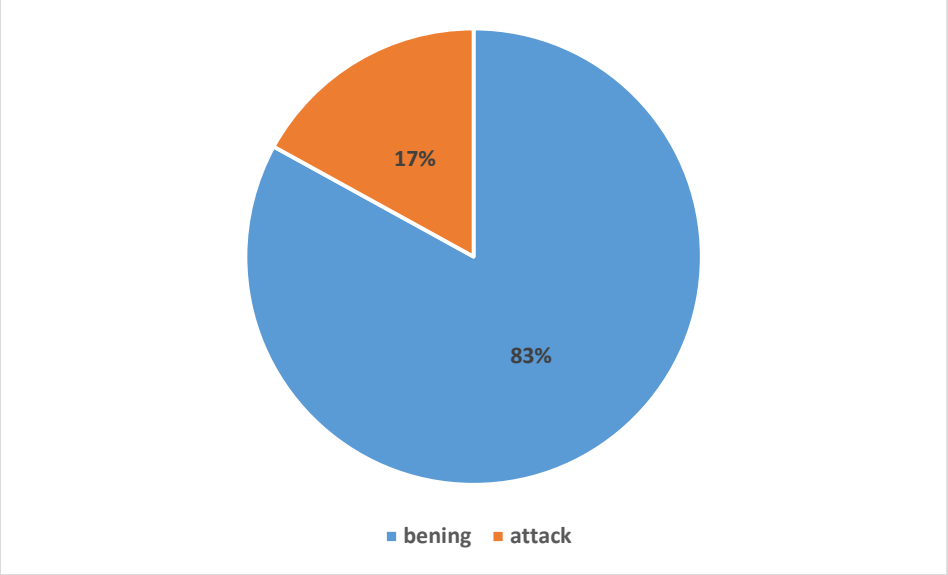

Figure 4: Distribution of Instances per Label

On the other hand, the distribution of attacks per type, as shown in Figure *5*, tells that the dataset consists of 14 different types of attacks. The graph reveals an under-representation of attack types in the below categories compared with the other attack types; massive variance in the number of records between attacks might challenge a multi-class classifier's training to find a proper prediction model for different attack types in the network traffic.

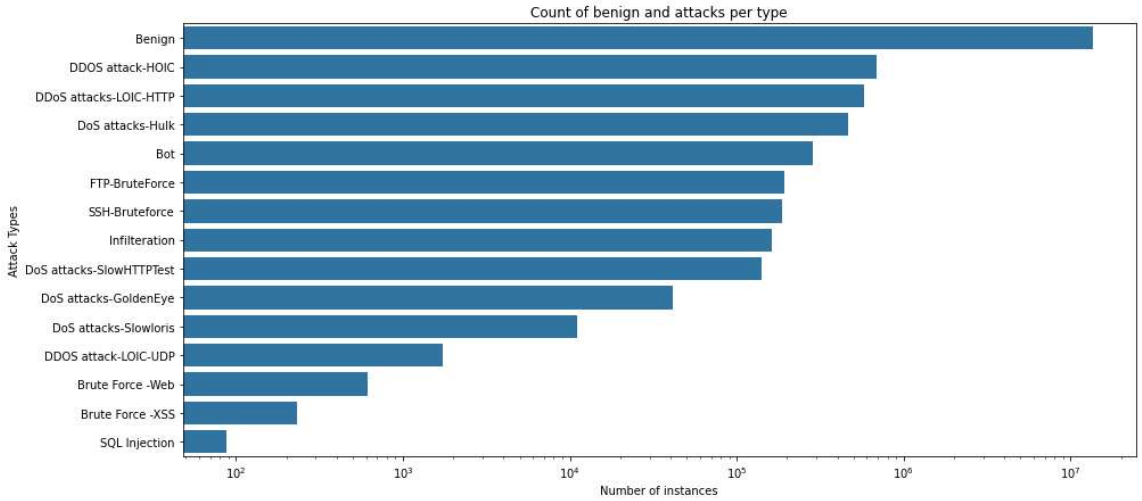

Figure 5: Distribution of Instances per Attack Type

5.2.5 Correlation between Features

It is necessary to discover and quantify the degree to which variables in the dataset are dependent upon each other. This knowledge can help preprocess the data to meet the expectations of machine learning algorithms, such as linear regression, whose performance will degrade with these interdependencies.

Correlation between sets of features is a measure of how well these features are related. The most common correlation measure in stats is the Pearson Correlation, which evaluates the linear relationship between two continuous variables.

A dataset of strongly correlated (positive or negative) attributes shows a high chance that the model performance will be impacted by a problem called “multi-collinearity,” which can lead to skewed or misleading results.

One of the good and quick ways to check the correlation among columns in any dataset is visualizing the correlation matrix as a heatmap.

The heatmap for the acquired dataset shows a strong correlation between many columns. However, when we test the correlation between the target variable (as the independent variable) and the remaining features (as the dependent variable), we found many features with a strong correlation. As appears in Table *3*, these are the highest five features that strongly correlate with the label feature (binary classification of the attack record).

Table 3: Top 5 Strongest Correlated Features

Feature name	R value
fwd_seg_size_min	0.429296
bwd_pkts_s	0.260203
ack_flag_cnt	0.220465
init_fwd_win_byts	0.218390
flow_pkts_s	0.148600

In the next section, we will discuss in more detailed the methods used to do feature engineering for the highly correlated features.

5.3 Modeling

As the pre-processing stage has been done and the dataset is cleaned up, we will start the stage where different machine learning classifiers will be examined to create the binary classifier/multi-classification to determine if the network traffic is benign or malicious.

5.3.1 Feature Engineering

Feature engineering is an important component in machine learning applications as learning performance is heavily dependent on the feature vector representation. It is not necessary

that all features of the dataset are relevant in building a good model for prediction. Using some of the features might even make the predictions worse (Jeff Heaton, 2016).

The first step in feature engineering for our acquired dataset is to remove all features with one unique value (zero variation features) because it does not influence the prediction of the target variable.

The exploratory data analysis for the features showed a high correlation between some of the features, therefore, one of the best approaches to remove the correlated features is clustering for them and selecting only one element from each cluster (Samina Khalid, Tehmina Khalil, Shamila Nasreen, 2014). To do that, we have calculated the correlation coefficient between features using the Spearman method. Spearman's correlation coefficient is a statistical measure of the strength of a monotonic relationship between paired data; it is denoted by r_s. the closer r_s is to ± 1, the stronger the monotonic relationship. Then the correlation matrix has been used as input for the clustering step by using the agglomerative hierarchical clustering method. This method is the most common type of hierarchical clustering and is used to group objects in clusters based on their similarity and use the first feature in the group to substitute all other features. However, there are many cluster agglomeration methods (i.e., linkage methods). The most common linkage method is Ward's Method. Ward method analyzes the variance of clusters, and it's the most suitable method for quantitative variables.

As a result of these steps, the number of features is reduced to 31 from 83 features in the original dataset as in Table *4*.

Table 4: List of Features that Used in Modeling

Feature Name	Description
ack_flag_cnt	Number of packets with ACK
active_mean	Mean time a flow was active before becoming idle
bwd_iat_min	Minimum time between two packets sent in the backward direction
bwd_iat_tot	Total time between two packets sent in the backward direction
bwd_pkt_len_mean	Mean size of packet in backward direction
bwd_pkts_s	Number of backward packets per second
down_up_ratio	Download and upload ratio
fin_flag_cnt	Number of packets with FIN
flow_byts_s	flow byte rate that is number of packets transferred per second
flow_duration	Flow duration
flow_iat_min	Minimum time between two flows
flow_iat_std	Standard deviation time two flows
flow_pkts_s	flow packets rate that is number of packets transferred per second
fwd_iat_min	Minimum size of packet in forward direction
fwd_iat_tot	Total time between two packets sent in the forward direction
fwd_pkt_len_mean	Average size of packet in forward direction
fwd_pkt_len_std	Standard deviation size of packet in forward direction
fwd_psh_flags	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
fwd_seg_size_min	Minimum segment size observed in the forward direction
fwd_urg_flags	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
idle_mean	Mean time a flow was idle before becoming active
init_bwd_win_byts	# of bytes sent in initial window in the backward direction
init_fwd_win_byts	Number of bytes sent in initial window in the forward direction
psh_flag_cnt	Number of packets with PUSH
rst_flag_cnt	Number of packets with RST
tot_bwd_pkts	Total packets in the backward direction
tot_fwd_pkts	Total packets in the forward direction
totlen_bwd_pkts	Total size of packet in backward direction
totlen_fwd_pkts	Total size of packet in forward direction
urg_flag_cnt	Number of packets with URG

5.3.2 Splitting Dataset for Training

When developing a predictive classifier from high dimensional data, the sample dataset is split into a training set and an independent test set, where the training is used to develop the classifier and the test to evaluate the model's performance. The proportion of the samples that should be devoted to the training set is about 2/3 of sample dataset (Kevin K Dobbin, Richard M Simon, 2011).

The hold-out method is splitting the sample dataset into different subsets and using one set for training the model and other sets for validating and testing. However, the hold-out method is ideal to use when you have an extensive dataset (Sebastian Raschka, 2018).

Training Dataset: the subset of sample data that used to fit the model.

Validation Dataset: The subset of sample data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters (Table 5).

Test Dataset: The subset of sample data used to provide an unbiased evaluation for the final model fit on the training dataset.

In our experiment, the acquired dataset was dissected into three subsets: training, evaluation, and testing with ratios of 0.8/0.1/0.1, and to ensure that all attacks have existed with the same ratio in all sets, the split will be stratified by the attack category (benign, attack).

Table 5: Hyperparameters Values

Classifier	Parameters	Values
Gradient Boosting Tree	loss_function	logless
	eval_metric	recall
	class_weights	[1,3.9] cnt of 0s/cnt of 1s
	verbosa	TRUE
	task_type	CPU
k-NN	n_neighbor	50,20,11
	n_jobs	4
Random Forest	verbosa	1
	n_jobs	32
	class_weigthts	balanced
Logistic Regression	solver	saga
	n_jobs	32
	verbosa	2

5.3.3 Metrics for Evaluation

Intrusion detection system is evaluated by the measure of accuracy, detection rate, and F1 Score.

Precision is the percentage of the total number of attacks that are correctly detected (Fares Ahmed H., Sharawy Mohamed I., 2011).

Detection rate or recall is described as the number of attacks detected by the proposed technique to the total number of attacks truly there.

Referring to the percentage of records of "benign" to the records of "attacks", data is highly imbalanced. Thus, the metric “accuracy” is not suitable to measure the performance of a classifier for this kind of dataset because the “accuracy” of a dummy classifier, is 0.83.

Therefore, we will use the following metric to evaluate the performance of the classifiers:

- **Recall** (weighted average) which is more important since the goal is to detect as many attacks as possible. Therefore, the classifier with higher recall will be better.
- **Precision** (weighted average) which should be large to reduce the number of false positives (when we said that a network flow is attack but it is not).
- **F-Score** or **F-measure** is a measure of a test's accuracy and calculated from the precision and recall of the test
- **The average Precision Score** is a way to interpret the precision-recall curve into a single value representing the average of all precisions Classifiers

5.3.4 Binary Classifiers

We will discuss the implementation of the binary classifiers used in the experiment:

5.3.4.1 Baseline Classifier

Dummy Classifier is choosing the class with the most frequent occurrences to make predictions. It is used only as a simple baseline for the other classifiers. In our case, all samples are classified as benign as this is the majority class in the dataset.

When a model has high recall but has low precision, the model classifies most positive samples correctly, but it has many false positives.

Table 6: Baseline Classifier Classification Report (Evaluation)

	Precision	recall	F1 Score	support
0	0.83	1.00	0.91	10787766
1	0.00	0.00	0.00	2198588
Accuracy			0.83	12986354
Macro avg	0.42	0.50	0.45	12986354
Weighted avg	0.69	0.83	0.75	12986354

Avg Precision Score: 0.16929986661383173

5.3.4.2 Logistic Regression

Logistic Regression is a classification algorithm used when the value of the output variable is categorical in nature. Logistic Regression is commonly used when the data in question has binary output, so when it belongs to one class or another or is either a zero or one.

Features standardization isn't required for Logistic Regression. However, standardizing the features makes the convergence faster. Therefore, as a pre-step for the prediction, we normalized value of the features for both training and testing subsets, because LR requires average or no multi-collinearity between independent variables.

Although the weighted average of precision and recall of the prediction is high, the average precision score is low compared to other classifiers.

Table 7: Logistic Regression Classification Report (Evaluation)

	Precision	recall	F1 Score	support
0	0.94	0.97	0.96	1348471
1	0.83	0.70	0.76	274823
Accuracy			0.93	1623294
Macro avg	0.89	0.84	0.86	1623294
Weighted avg	0.92	0.93	0.92	1623294

Avg Precision Score: 0.6329299079369365

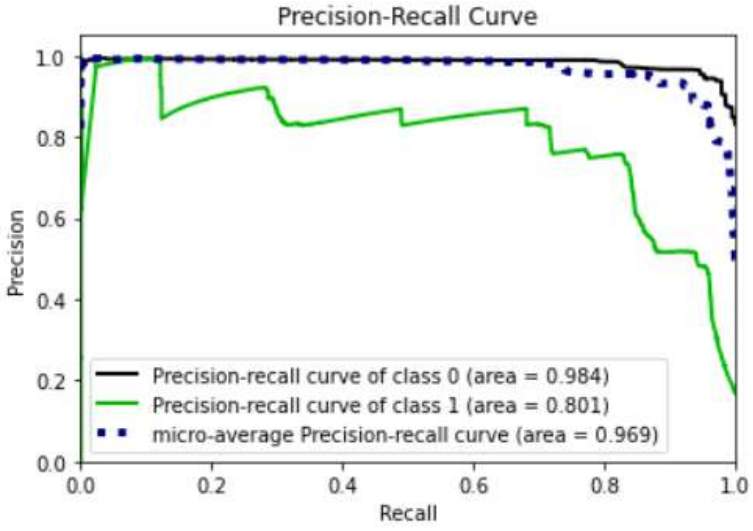

Figure 6: Recall Curve - Logistic Regression Classification Report (Evaluation)

5.3.4.3 Random Forest

Random Forest is a commonly used algorithm in classification because of its simplicity and diversity with great results without the need to tune the hyperparameter of the model.

Another great property of the Random Forest algorithm is that it can easily measure the relative importance of each feature on the prediction.

Scikit-learn provides a great tool for this as it calculates this score automatically for each feature after training and scales the results, so the sum of all importance is equal to one (Pedregosa et al., 2011) .

On the contrary of Logistic Regression, predictor variables did not standardize, although, results are much better than Logistic Regression as the average precision score is about 0.92 as in Table *8*

Table 8: Random Forest Classification Report (Evaluation)

	Precision	recall	F1 Score	support
0	0.99	0.99	0.99	1348471
1	0.96	0.95	0.96	274823
Accuracy			0.99	1623294
Macro avg	0.98	0.97	0.97	1623294
Weighted avg	0.99	0.99	0.99	1623294

Avg Precision Score: 0.925741778539307

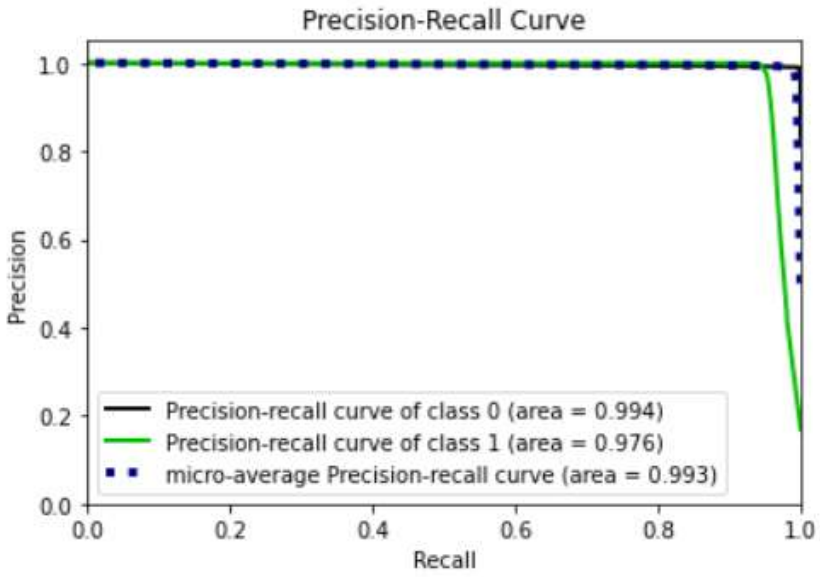

Figure 7: Recall Curve - Random Forest Classification Report (Evaluation)

5.3.4.4 Gradient Booting Trees

Gradient Boosting Tree uses an ensemble of decision trees to predict a target label. In gradient boots, hyperparameters (learning rate, number of estimators) are introduced to prevent overfitting. In the first iteration of model fit, we used default values. However, scikit-learn provides a method to measure the validation error and find the optimal number of estimators, but we did not elaborate more in this part as we have achieved good results using defaults values as shown in Table *9*.

Table 9: Gradient Booting Trees Classification Report (Evaluation)

	Precision	recall	F1 Score	support
0	0.99	0.99	0.99	1348471
1	0.97	0.96	0.96	274823
Accuracy			0.99	1623294
Macro avg	0.98	0.98	0.98	1623294
Weighted avg	0.99	0.99	0.99	1623294

Avg Precision Score: 0.9317103518127884

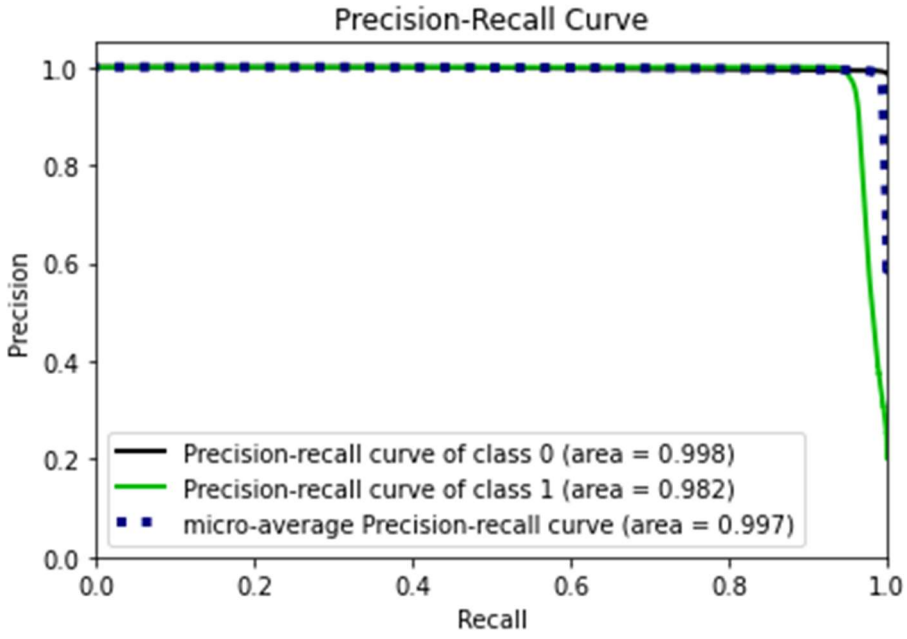

Figure 8: Recall Curve Gradient Booting Trees Classification Report (Evaluation)

5.3.4.5 k-nearest neighbor's algorithm (k-NN)

The k-nearest neighbors (k-NN) algorithm is one of the simplest supervised machine learning algorithms that can be used to solve classification problems. Like Logistic Regression classifier, the scale of the predictive variable can affect the performance of the classifier, so in as first step, we have standardized the predictive variable. High dimensionality dataset is another issue for the k-NN algorithm, so to overcome this issue, we have used PCA.

Principal Component Analysis, as the name states, help in determine which dimensions carry the information. Using the PCA method in python, we have reduced the number variable into only two (Kevin K Dobbin, Richard M Simon, 2011). Table *10* shows the model resuls.

Table 10: k-nearest neighbors Algorithm Classification Report (Evaluation)

	Precision	recall	F1 Score	support
0	0.87	0.98	0.92	1348471
1	0.71	0.30	0.42	274823
Accuracy			0.86	1623294
Macro avg	0.79	0.64	0.67	1623294
Weighted avg	0.84	0.86	0.84	1623294

5.3.5 Binary Classifiers Performance on Test Dataset

As shown in the Table *11*, the performance measurement summary for all models, Gradient Booting Trees achieved the best results over all classifiers. Furthermore, Gradient Booting Trees has been tested against the test dataset to see if the model suffers from an overfitting problem or not. Again, results show very good performance on the new unseen dataset, whereas recall of 0.99 and precision of 0.99 and avg precision score is 0.930 Table *12*.

Table 11: Performance Measures Summary

Model	Recall	Precision	F1 Score	Avg PR	Recall Attack	Precision Attack
Baseline	0.83	0.69	0.75	0.169	0.00	0.00
k-NN	0.86	0.84	0.84	N/A	0.30	0.71
Logistic Regression	0.93	0.92	0.92	0.63	0.70	0.83
Random Forest	0.99	0.99	0.99	0.925	0.95	0.96
Gradient Booting Trees	0.99	0.99	0.99	0.932	0.96	0.96

Table 12: Gradient Booting Trees Classification Report (Test)

	Precision	recall	F1 Score	support
0	0.99	0.99	0.99	1348471
1	0.96	0.96	0.96	274824
Accuracy			0.99	1623295
Macro avg	0.98	0.98	0.98	1623295
Weighted avg	0.99	0.99	0.99	1623295

Avg Precision Score: 0.9303765866411948

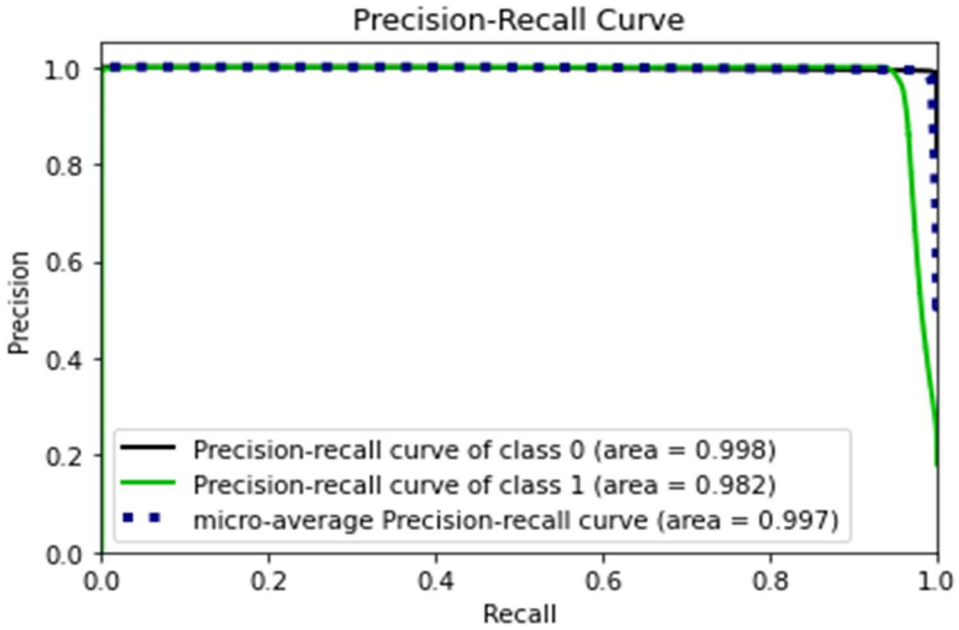

Figure 9: Recall Curve - Gradient Booting Trees Classification Report (Test)

As further analysis for the model performance, we have tested the misclassification (false-positive prediction) results per attack type on the test dataset. Results are promising as we can see in Table *13*.

Table 13: Performance Testing of Classification

Attack Type	Actual data	Misclassifications		Correct Classifications	
		# Of Records	Percentage	# Of Records	Percentage
Benign	1,348,471	9,341	0.006927	1,339,130	0.993073
Attack	274,824	11,777	0.042853	263,047	0.957147

5.3.6 Multi-Label Classifier - Predict Type of Attack

As the acquired dataset provides "type of attack" as a feature, we have developed a multi-label classification model to predict the type of attack. We will use only the best two models for that, Random Forest, and Gradient Booting Trees, as both achieved outstanding results in comparison to the others.

Referring to the bar plot in Figure *5*, the distribution of data per type of attack is imbalanced. Therefore, this issue would affect the performance of our models. However, oversampling may solve this problem, but as step one and for reference, we will run the Random Forest on the original distribution of the data in the train dataset, then we will rerun the model after doing oversampling and compare the results.

Table 14: Random Forest Multi-Label Classification Report (before oversampling)

Attack Type	precision	recall	F1 Score	support
Benign	0.99	0.93	0.96	1348471
Bot	0.98	1.00	0.99	28619
Brute Force -Web	0.44	0.72	0.54	61
Brute Force -XSS	0.79	0.65	0.71	23
DDOS attack-HOIC	1.00	1.00	1.00	68602
DDOS attack-LOIC-UDP	0.88	0.93	0.90	173
DDoS attacks-LOIC-HTTP	1.00	1.00	1.00	57619
DoS attacks-GoldenEye	1.00	1.00	1.00	4151
DoS attacks-Hulk	1.00	1.00	1.00	46191
DoS attacks-SlowHTTPTest	0.77	0.52	0.62	13989
DoS attacks-Slowloris	0.45	0.99	0.62	1099
FTP-BruteForce	0.72	0.89	0.79	19336
Infilteration	0.05	0.30	0.09	16194
SQL Injection	0.01	0.38	0.01	8
SSH-Bruteforce	1.00	1.00	1.00	18759
Accuracy			0.93	1623295
Macro avg	0.74	0.82	0.75	1623295
Weighted avg	0.98	0.93	0.95	1623295

5.3.7 Oversampling

We consider a dataset imbalanced if the classification categories are approximately not equally represented, as in Figure 4 (Chawla, Bowyer, Hall, & Kegelmeyer, 2002).

In machine learning, classifiers are assumed to minimize misclassification errors and maximize predictive accuracy. The underlying assumption in these classification algorithms is that the dataset under study has an approximately balanced number of instances per available class. Briefly, it assumes that the prior possibilities of the target classes are similar (Fadi Thabtah, Suhel Hammoud, Firuz Kamalov, Amanda Gonsalves, 2020).

Table 15: Number of Instances per Attack Type

Attack Type	Number of instances (Original dataset)	Number of instances after oversampling
Benign	10,787,766	10,787,766
DDOS attack-HOIC	548,809	548,809
DDoS attacks-LOIC-HTTP	460,953	460,953
DoS attacks-Hulk	369,530	369,530
Bot	228,953	228,953
FTP-BruteForce	154,688	154,688
SSH-Bruteforce	150,071	150,071
Infilteration	129,547	129,547
DoS attacks-SlowHTTPTest	111,912	111,912
DoS attacks-GoldenEye	33,206	100,000
DoS attacks-Slowloris	8,792	100,000
DDOS attack-LOIC-UDP	1,384	100,000
Brute Force -Web	489	100,000
Brute Force -XSS	184	100,000
SQL Injection	70	100,000

Class imbalance is one of the major obstacles in classification to get high accuracy for minority class data points. To solve this problem, SMOTE-NC oversampling method is used to synthesize a new instance and balance the dataset.

SMOTE-NC algorithm calculated a predetermined number of neighbors for each underrepresented instance, then randomly chosen some minority class instances for synthetic data point (Mimi Mukherjee, Matloob Khushi, 2021).

In our experiment, SMOTE-NC was applied on attack types were the number of instances less than 100,000 only to avoid any possibility of overfitting if we were oversampling the whole dataset. The new distribution of data per attack type is presented in Table *15*.

5.3.7.1 Multi-Label Classifier Performance after the Oversampling

After oversampling for the minority instances, we have rerun the multi-label classifier of Random Forest and measure the model performance. Table 16 *summarizes the new results. We can see a significant improvement in recall value for the oversampled attack types. The recall, which refers to the number of actual attacks detected correctly, is vital in the anomaly detection system. In contrast, we can see decreasing in the precision value in some cases, which means that we have more false-positive cases. Overall, the achieved enhancement in recall performance is considered a general improvement for the oversampling on the performance of this classifier as shown in*

Table *17*.

Table 16: Random Forest Multi-Label Classification Report (after oversampling)

Attack Type	precision	recall	F1 Score	support
Benign	0.99	0.93	0.96	1348471
Bot	0.99	1.00	0.99	28619
Brute Force -Web	0.17	0.92	0.28	61
Brute Force -XSS	0.58	0.83	0.68	23

DDOS attack-HOIC	1.00	1.00	1.00	68602
DDOS attack-LOIC-UDP	0.83	0.96	0.89	173
DDoS attacks-LOIC-HTTP	1.00	1.00	1.00	57619
DoS attacks-GoldenEye	1.00	1.00	1.00	4151
DoS attacks-Hulk	1.00	1.00	1.00	46191
DoS attacks-SlowHTTPTest	0.77	0.52	0.62	13989
DoS attacks-Slowloris	0.63	0.99	0.77	1099
FTP-BruteForce	0.72	0.89	0.79	19336
Infilteration	0.05	0.31	0.09	16194
SQL Injection	0.80	0.50	0.62	8
SSH-Bruteforce	1.00	1.00	1.00	18759
Accuracy			0.93	1623295
Macro avg	0.77	0.86	0.78	1623295
Weighted avg	0.98	0.93	0.95	1623295

Table 17: Oversampling of Minority Classes

Attack Type	Before Oversampling			After Oversampling		
	precision	recall	F1 Score	precision	recall	F1 Score
DoS attacks-GoldenEye	1	1	1	1	1	1

DoS attacks-Slowloris	0.45	0.99	0.62	0.63	0.99	0.77
DDOS attack-LOIC-UDP	0.88	0.93	0.9	0.83	0.96	0.89
Brute Force -Web	0.44	0.72	0.54	0.17	0.92	0.28
Brute Force -XSS	0.79	0.65	0.71	0.58	0.83	0.68
SQL Injection	0.01	0.38	0.01	0.8	0.5	0.62

5.3.7.2 Binary Classifier Performance After the Oversampling

We have rerun the binary classification for Gradient Booting Trees and Random Forest after doing the oversampling. For Gradient Booting Trees, a slight drop in the performance is noticed, however, the precision, recall and F1 Score remained the same in both experiments, but GB (Table *18*) was not improved due to the nature of the data and looks like the oversampling did not reduce the huge skew in the dataset. In contrast, the performance of the Random Forest classifier shows significant enhancement after oversampling and with better performance than Gradient Booting Trees (Table *19*) (Maxime Labonne, 2020).

Table 18: Gradient Booting Trees Classification Report (after oversampling)

	Precision	recall	F1 Score	support
0	0.99	0.99	0.99	1348471
1	0.96	0.96	0.96	274824
Accuracy			0.99	1623295
Macro avg	0.98	0.97	0.98	1623295
Weighted avg	0.99	0.99	0.99	1623295

Avg Precision Score: 0.9292485297007159

Table 19: Random Forest Classification Report (after oversampling)

	Precision	recall	F1 Score	support
0	1.00	0.99	0.99	1348471
1	0.97	0.98	0.97	274824
Accuracy			0.99	1623295
Macro avg	0.98	0.98	0.98	1623295
Weighted avg	0.99	0.99	0.99	1623295

Avg Precision Score: 0.9493976704744245

5.3.8 Combining between Models – Voting

In some cases, the best classifier may fail to classify the type of attack while others do correctly. Although, as said in an early stage, we would combine the results of multiple classifiers in detecting the attacks, whether it is benign or malicious. Therefore, a voting algorithm that weighted each classifier will be implemented to take this decision.

Two commonly voting schemes can be used in classifier voting. Hard voting, in which each classifier vote for a label and the majority win, for example, in our case, we have four different classifiers, the label of instance is voted based on the majority predicted label (Jingjing Cao a, Sam Kwong, Ran Wang, Xiaodong L, Ke Li, Xiangfei Kong, 2015).

While, soft voting assigns a probability value to each classifier, so the predictions are weighted by the classifier's importance and summed up. Then the target label with the

greatest sum of weighted probabilities wins the vote (Gustavo Henrique Paetzold, Lucia Specia, 2016).

The voting approach was used to combine the binary classification results of all the methods, firstly by using the original distribution of attacks categories, in another iteration by oversampling for the minority classes. We aimed from this approach to measure the impact of voting on the accuracy of detection compared with the individual's method results, then to measure the impact of oversampling on the voting approach as well.

In the experiment, the weight of each model was specified by considering the overall performance of each model before and after oversampling as in Table *20*.

Table 20: Soft Voting Classifiers Weights

Classifier	Weight without oversampling	Weight with oversampling
k-NN	0.05	0.05
Logistic Regression	0.15	0.15
Random Forest	0.30	0.5
Gradient Booting Trees	0.50	0.3

Table 21: Voting Results before and after Oversampling

Model	Recall	Precision	F1 Score	Avg PR	Recall Attack	Precision Attack
Before oversampling	0.99	0.99	0.99	0.941	0.96	0.98

After oversampling	0.99	0.99	0.99	0.951	0.98	0.98

6 Results

The study performed using a publicly available dataset of CSE-CIC-IDS2018 and divided into three main phases. In the first phase, the analyses performed using four binary classification models of (k-NN, Random Forest, Gradient Booting Trees, and Logistic Regression), where Gradient Booting Trees and Random Forest demonstrated outstanding results in comparison with k-NN and Logistic Regression with recall and precision ratio of 99% for both classifiers. The recall and precision ratio for "attack" class of RF is 96% and 95% respectively, and 96% for GB. The other two classifiers' performance of k-NN and LR were 86% and 92%, respectively, which is not considered poor but much less than the first two models.

In the second phase, we built a multi-label classifier to predict the type of attack; the prediction rate varies because of the imbalance issue in the sample data. To overcome this issue, we did synthetic oversampling for the attack types with a small distribution. We have led to a noticeable enhancement in the recall value of many types of attacks that have oversampled. As an example, the precision value of SQL Injection attack jumps from 0.01 to 0.8. In contrast, some of the attack type precision declined as in "Brute Force -Web" and "Brute Force -XSS". The reason could be that the training data is not sufficient to build a strong model for prediction.

In the third phase, we have used the voting approach by combining the results of all predictors in the binary classification using soft voting. This was done by assigning weight for each classifier based on the prediction performance. Assigned weights as per classifier

of k-NN, LR, RF, and DB were 0.05,0.15,0.30,0.50, respectively. When compared between the precision ratio of the "attack" of GB (96%) (since GB had the best performance among the prediction models) and soft voting prediction (98%), we can conclude that number of false alarms has been decreased, and this is the aiming of combining the results of classifiers in the prediction as shown in Table 21.

Lastly, previous procedure was repeated on the binary classification prediction results after doing oversampling for the minority classes, the achieved results were as following: prediction precision ratio of the "attack" is (98%) and average precision ration is 0.951, which is the best among all tests.

In an early stage, preprocessing methods were implemented; by imputing the default values and infinity with the mean of each variable. The duplicate header rows were deleted. Various feature engineering technique were applied to the acquired dataset, first by removing all features of one unique value (zero variation features) because they do not influence the prediction of the target variable, and by clustering the features then select only one feature from each cluster, so the study performed using 31 from 83 features in the original dataset.

We used the hold-out evaluation by dividing the dataset into three sub-datasets: training, evaluation, and testing, with the ratio of 80,10,10 consequently. The purpose of hold-out evaluation is to test a big data model (16 million records) on different data than it was trained on. This provides an unbiased estimate of learning performance. Recall, precision, F1 Score, and average precision score were used to evaluate the performance of the classifiers.

The experiment was done using Python 3.7 and scikit-learn on a huge VM server of 192G of memory and 128 cores. Even though the CPU time of the training process was a little bit long because we are using a large dataset. The Table *22* shows the CPU time of the training process for the classifiers; however, we decided to drop the SVM method from our experiment because we could not get any results after two days of continuous processing for the training process.

Table 22: CPU of the Training Process

Classifier	CPU (Second)
k-NN	6000
Logistic Regression	2400
Random Forest	960
Gradient Booting Trees	3000

7 Comparing with Previous Work

We compared our methodology and the results of our experiment with other related studies. In (Fitni QRS, Ramli K., 2020), the researchers used the ensemble method by emerging the best three models, which achieved the top results from seven used in the experiment using the majority voting method. In contrast, our study used the same approach, but we have combined the results of four algorithms by using the soft voting method as soft voting considers how confident each voter is, rather than just a binary input from the voter as the hard voting does.

When it comes to the step of comparing our results with the results of the referred study, we can see slightly better performance in all measurements as we have reached the percentage

of 99 for all of them compared to 0.988, 0.988, 0.979, 0.971 for accuracy, precision and F1 Score respectively when using 23 features, and 0.987, 0.980, 0.974, 0.977 when using all features in prediction.

In this study (Laurens D'hooge, Tim Wauters, Bruno Volckaert, Filip De Turck, 2020), the researchers used 12 supervised algorithms, including the algorithms we have used. The highest accuracy, precision, and recall scores for CSE-CIC-IDS2018 were 96%, 99%, and 79%, respectively; In comparison with our results, it is significantly lower.

Those were the most related studies to our approach, however, for full comparison between our results and previous work please refer to

Table 23.

Author	Used methods	Number of Features	Size of Dataset	Approach	Precision	Recall	F1 Score	Remark
Fitni QRS, Ramli K., 2020)	GB, RF, NB, DT, LR	Spearman's rank correlation (23features)	Full Dataset	ensemble approach - hard voting	0.988	0.971	0.979	
	GB, RF, NB, DT, LR	All features	Full Dataset		0,980	0,977	0,974	
Laurens D'hooge, Tim Wauters, Bruno Volckaert, Filip De Turck, 2020	DT, RF, DT-based Bagging, GB, Extratree, Adaboost, XGBoost, k-NN, Ncentroid, linearSVC, RBFSVC, LR	All features	Full Dataset	single classifier to predict attack types (brute force, DoS, DDoS and botnet)	0.79	0.99	0.88	Best results without specifying which attach type achieved these results
V. Kanimozhi, Dr. T. Prem Jacob, 2019	k-NN, NB, Adaboost with Decision Tree, SVM, RF, ANN	All features	1,048,575	detect specific attack type (botnet)	1.00	1.00	1.00	When compare it with our results, we achieved 0.99,1.0,0.99, however, it was not our goal to predict attack types
Marta Catillo, Massimiliano Rak, Umberto Villano, 2020)	auto encoder	All features	Full Dataset	detect specific attack type (botnet)	0.95	0.99	0.967	Again, our results are better
Our Results	k-NN, LR, GB, RF	Hierarchy clustering (31 features)	Full Dataset	Combined approach	0.99	0.99	0.99	Oversampling, Soft voting

As a conclusion, results achieved in our experiment are better than the results achieved in the previously referred papers. The outstanding performance of our models results from the

way we have handled the likelihood of overfitting. Our feature selection approach used the hierarchy clustering for the coefficient matrix results in selecting the most significant features for prediction. The way we were oversampling the low distribution attack types among all attacks degrade the impact of imbalanced data issue in the dataset on the modeling. Finally, the use of 3-sets Holdout evaluation positively affected the accuracy of the testing results. All these thoughtful actions responsible for this preference in the results over the referred papers.

Table 23: Compare our Results with Previous Work

Author	Used methods	Number of Features	Size of Dataset	Approach	Precision	Recall	F1 Score	Remark
Fitni QRS, Ramli K., 2020)	GB, RF, NB, DT, LR	Spearman's rank correlation (23features)	Full Dataset	ensemble approach - hard voting	0.988	0.971	0.979	
	GB, RF, NB, DT, LR	All features	Full Dataset		0,980	0,977	0,974	
Laurens D'hooge, Tim Wauters, Bruno Volckaert, Filip De Turck, 2020	DT, RF, DT-based Bagging, GB, Extratree, Adaboost, XGBoost, k-NN, Ncentroid, linearSVC, RBFSVC, LR	All features	Full Dataset	single classifier to predict attack types (brute force, DoS, DDoS and botnet)	0.79	0.99	0.88	Best results without specifying which attach type achieved these results
V. Kanimozhi, Dr. T. Prem Jacob, 2019	k-NN, NB, Adaboost with Decision Tree, SVM, RF, ANN	All features	1,048,575	detect specific attack type (botnet)	1.00	1.00	1.00	When compare it with our results, we achieved 0.99,1.0,0.99, however, it was not our goal to predict attack types
Marta Catillo, Massimiliano Rak, Umberto Villano, 2020)	auto encoder	All features	Full Dataset	detect specific attack type (botnet)	0.95	0.99	0.967	Again, our results are better
Our Results	k-NN, LR, GB, RF	Hierarchy clustering (31 features)	Full Dataset	Combined approach	0.99	0.99	0.99	Oversampling, Soft voting

8 Conclusions and Future Work

Our proposed approach of combining multiple classifiers to predict the Network access traffic, whether benign or attack, achieved promising results compared to single prediction methods. Oversampling also improved prediction accuracy for both binary and multi-label classification, as it handles imbalanced distribution in the dataset. The use of 3-sets Holdout evaluation affected positively the verification of the accuracy of the testing results.

There is a large discrepancy between prediction accuracy for the type of attack in the multi-label classification. One of the main reasons is the imbalance distribution of instances of attack types in the dataset. Oversampling positively influenced the prediction of some attack types but also negatively on the other types.

Critical characteristics should exist in any dataset to be used in developing and evaluating the ID models efficiently. Characteristics as the diversity of attacks, anonymity, available protocols, feature set, labeled data samples, heterogeneity, and metadata. All of them are satisfied by the CSE-CIC-IDS-2018 dataset.

However, preprocessing is necessary to achieve better performance for the classifiers and feature engineering to reduce the processing time.

Deployment of the voting approach in a real intrusion detection system is possible. One of the drawbacks of having multiple classifiers in the prediction process is the high training time (up to 100 minutes for a single classifier).

As future work, the voting approach could also be used in the multi-label classification, and it is expected to significantly improve the accuracy of prediction of attack types as happens in binary classification.

9 Bibliography

Abusalah Mustafa A. (2008). Cross language information retrieval using ontologies. Sunderland: ISNI: 0000 0004 2680 6723, University of Sunderland.

Ali Reza Zebarjad, Mohmmad Mehdi Lotfinejad. (2012, 12). Combination of Three Machine Learning Algorithms for Intrusion Detection Systems in Computer Networks. p. 10.

Ankit Thakkar, Ritika Lohiya. (2019). A Review of the Advancement in Intrusion Detection Datasets. p. 10.

Belavagi Manjula C., Muniyal Balachandra. (2016, 12). Performance Evaluation of Supervised Machine Learning. p. 7.

Brijain R Patel, Kushik K Rana. (2014). A Survey on Decision Tree Algorithm For Classification. IJEDR, p. 5.

Canada's Cybersecurity Hub. (n.d.). Canadian Institute for Cybersecurity. Retrieved from Canadian Institute for Cybersecurity: http://www.unb.ca/cic/datasets/ids-2018.html

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002, 01 09). SMOTE: Synthetic Minority Over-sampling Technique. p. 37.

Chibuzor John Ugochukwu, E. O Bennett. (2018). An Intrusion Detection System Using Machine Learning Algorithm. p. 9.

Fadi Thabtah, Suhel Hammoud, Firuz Kamalov, Amanda Gonsalves. (2020, 03). Data imbalance in classification: Experimental evaluation. pp. 429-441.

Fares Ahmed H., Sharawy Mohamed I. (2011, 12). Intrusion Detection: Supervised Machine Learning. p. 9.

Fitni QRS, Ramli K. (2020). Implementation of ensemble learning and feature selection for performance improvements in anomaly-based intrusion detection systems. IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), pp. 118–124.

Gustavo Henrique Paetzold, Lucia Specia. (2016). A voting approach of modulation classification for wireless network. SV000gg.

Iman Sharafaldin, Arash Habibi Lashkari, Ali A. Ghorbani. (2018, 01). “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. 4th International Conference on Information Systems Security and Privacy (ICISSP).

Jeff Heaton. (2016, 04). An empirical analysis of feature engineering for predictive modeling. SoutheastCon 2016, pp. 1-6.

Jingjing Cao a, Sam Kwong, Ran Wang, Xiaodong L, Ke Li, Xiangfei Kong. (2015). Class-specific soft voting based multiple extreme learning. Neurocomputing, pp. 275-284.

Kalidindi Venkateswara Rao, G. Sharmila Sujatha2. (2019, 08). Supervised Learning Techniques for Intrusion Detection System. IJIRSET, p. 6.

Kazi Abu Taher, Billal Jisan, Md. Rahman. (2019). Network Intrusion Detection using Supervised Machine Learning Technique with Feature Selection. p. 4.

Kelly H. Zou, A. James O’Malley, Laura Mauri. (2006, 02 6). Receiver-Operating Characteristic Analysis for Evaluating Diagnostic Tests and Predictive Models. pp. 654-657.

Kevin K Dobbin, Richard M Simon. (2011, 04). Optimally splitting cases for training and testing high dimensional classifiers.

Kotsiantis, S. B. Zaharakis I., Pintelas P. (2006, 11). Supervised machine learning: A review of classification techniques. p. 32.

Kruegel C., Toth T. (2003, 09). Using Decision Tree s to improve signature-based intrusion detection. In International Workshop on Recent Advances in Intrusion Detection. pp. 173-191.

Laskov Pavel, Düssel Patrick, Schäfer Christin, Rieck Konrad. (2005). Learning Intrusion Detection: Supervised or Unsupervised? p. 8.

Laurens D'hooge, Tim Wauters, Bruno Volckaert, Filip De Turck. (2020). Inter-dataset generalization strength of supervised machine learning methods for intrusion detection. Journal of Information Security and Applications.

Manjula C. Belavagi, Balachandra Muniyal. (2016). Performance Evaluation of Supervised Machine Learning. p. 7.

Markus Ring, Sarah Wunderlich, Deniz Scheuring, Dieter Landes, Andreas Hotho. (2019, 07 6). A Survey of Network-based Intrusion Detection Data Sets. p. 17.

Marta Catillo, Massimiliano Rak, Umberto Villano. (2020). 2L-ZED-IDS: a Two-Level Anomaly Detector. Workshops of the International Conference on Advanced Information Networking and Applications, pp. 687–696.

Maxime Labonne. (2020). Anomaly-based network intrusion detection using machine learning. Cryptography and Security [cs.CR]. Institut Polytechnique de Paris, p. 122.

Mimi Mukherjee, Matloob Khushi. (2021, 03 2). SMOTE-ENC: A Novel SMOTE-Based Method to Generate Synthetic Data for Nominal and Continuous Features.

Pedregosa et al. (2011). Scikit-learn: Machine Learning in Python. JMLR 12, pp. 2825-2830.

Rao V. Vemuri. (2002, 10). Use of K-Nearest Neighbor classifier for intrusion detection. Computers & Security, p. 10.

Samina Khalid, Tehmina Khalil, Shamila Nasreen. (2014). A survey of feature selection and feature extraction techniques in machine learning. Science and Information Conference, pp. 372-378.

Sebastian Raschka. (2018, 11). Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. University of Wisconsin–Madison Department of Statistics, p. 49.

Sen, Jaydip; Mehtab, Sidra. (2020, 06 19). Machine Learning Applications in Misuse and Anomaly Detection. p. 21.

Serpen, Gursel, Sabhnani, Maheshkumar. (2003, 01). Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context. p. 7.

Solms Rossouw von, Niekerk Johan van. (2013). From information security to cyber security. p. 8.

Sonny Rosenthal. (2017, 11). Data Imputation. The International Encyclopedia of Communication Research Methods, p. 27.

Tavallaee Mahbod, Bagheri Ebrahim, Lu Wei, Ghorbani Ali A. (2009). A detailed analysis of the KDD CUP 99 data set. p. 7.

V. Kanimozhi, Dr. T. Prem Jacob. (2019, 10). Calibration of various optimized machine learning classifiers in network intrusion detection system on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing. International Journal of Engineering Applied Sciences and Technology, 2019, pp. Pages 209-213.

Victoria Cox. (2017, 03 11). Exploratory Data Analysis. Translating Statistics to Make Decisions, pp. 47-74.

10 Appendix

10.1 Experimental Environment

The Environment used for thesis experiments composed of the following components:

- Server:
  - Server: Processor Intel(R) Xeon(R) Gold 6262V CPU @ 1.90GHz with 96 Core.
  - Memory: 128 GB RAM.
  - OS: Windows Server 2019 - Version 1809
- Toolkit:
  - Anaconda: 2020.11
  - Jupiter-notebook: 6.1.4
  - python: 3.8.5
  - Sklearn: 0.24.1
  - pandas: 1.1.3
  - NumPy: 1.19.2
  - seaborn: 0.11.0
  - matplotlib: 3.3.2

10.2 Source Code

I have uploaded the source code into GitHub; please use the following link to access the source code repository: https://github.com/helmipal/supervided-miuse-Detection

10.3 Dataset

Publicly available dataset of Canadian Establishment for Cybersecurity CSE-CIC-IDS2018

https://registry.opendata.aws/cse-cic-ids2018/

الملخص

ظهرت أهمية استخدام خوارزميات تعلم الآلة في كثير من المجالات ومنها أنظمة الحماية من الاختراق للشبكات والأنظمة السيبرانية (Intrusion Detection System). تعتمد أنظمة الحماية من التسلل والاختراق التقليدية على طريقتين في اكتشاف حالات الاختراق وهي عن طريق مقارنة أنماط الحركات بأنماط معروفة ومصنفة مسبقاً (Misuse Detection) أو من خلال تحليل أنماط الأحداث وتصنيفها بين حركات طبيعية وغير طبيعية (Anomaly Detection).

يمكن للأنظمة التي تعتمد على تحليل أنماط الأحداث من اكتشاف أنواع غير معروفة من الهجمات ومحاولات التسلل، لكن في المقابل كثيراً ما تعاني هذه الأنظمة من مشكلة التصنيفات الإيجابية الخاطئة (false-positive)، لذلك سنستعرض في دراستنا هذه آليات استخدام خوارزميات تعلم الآلة بالأشراف (Supervised Machine Learning) في اكتشاف الهجمات الغير معروفة مسبقا حيث سيتم دمج نتائج عدة نماذج مجتمعة للتقليل من نسبة التصنيفات الايجابية الخاطئة لهذه الحركات.

في هذه الدراسة تم استخدام أربع خوارزميات مختلفة من خوارزميات تعلم الآلة بالأشراف وهي: Random Forest, Gradient Booting Trees, Logistic Regression, Decision Tree, K-nearest neighbor.