Ain Shams Engineering Journal xxx (xxxx) xxx

Contents lists available at ScienceDirect

Ain Shams Engineering Journal

journal homepage: www.sciencedirect.com

Full Length Article

EL-RFHC: Optimized ensemble learners using RFHC for intrusion attacks classification

P. Kuppusamy [a], Dev Kapadia [a], Edaboina Godha Manvitha [a], Sami Dhahbi [b], C. Iwendi [c], M. Ijaz Khan [d,e,*], Sachi Nandan Mohanty [a], Nidhal Ben Khedher [f,g]

[a] School of Computer Science & Engineering, VIT-AP University, Andhra Pradesh, India
[b] Department of Computer Science, College of Science and Art at Mahayil, King Khalid University, Muhayil Aseer 62529, Saudi Arabia
[c] School of Creative Technology, University of Bolton, United Kingdom
[d] Department of Mechanical Engineering, Lebanese American University, Beirut, Lebanon
[e] Department of Mathematics and Statistics, Riphah International University I-14, Islamabad 44000, Pakistan
[f] Department of Mechanical Engineering, College of Engineering, University of Ha'il, 81451 Ha'il City, Saudi Arabia
[g] Laboratory of Thermal and Energetic Systems Studies (LESTE) at the National School of Engineering of Monastir, University of Monastir, Tunisia

ARTICLE INFO

Keywords:
High correlation
IDS
SMOTE
Attention mechanism
Ensemble learning
LSTM
RFHC

ABSTRACT

The extensive growth of mobile technology leads to magnifying the usage of digital gadgets around the world. This requires a fast-interconnecting communication medium to transfer the data between the devices. Meanwhile, the intruders attempt to make huge traffic in the network that leads to loss of data. To identify the intrusion attacks, ensemble Machine Learning (ML) classifiers are applied using the various feature variables importance. However, most of the transmitting data contains high dimensions with numerous variables leads to more execution time to classify the attacks. This study initiated the novel approach fusion of the Random Forest classifier and High Correlation (RFHC) feature selection approach to diminish the quantity of the variables. Also, the count of intrusion attacks class is lower than the normal class leads to generating an imbalanced dataset. Hence, Synthetic Minority Over-Sampling Technique (SMOTE) is suggested to create a balanced dataset for multi-class classification, and Un-upsampled data for binary-class classification respectively. The pre-processed dataset fed into the ensemble machine learners, and attention mechanism-based LSTM to classify as various intrusion attacks and normal data. This research work focused on reducing the CICIDS2017 dataset's variable dimensions from 71 to 34 using RFHC. The performance results showed that RF classifier performed better with accuracy of 99.4 %, precision 99.4 %, average recall 99.2 % and average F1-score 99.6 % in binary-class classification, and Extreme Gradient Boosting (XGBoost) achieved better accuracy of 99.7 %, precision 98.7 %, average recall 99.5 % and average F1-score 99.2 % in multi-class classification.

1. Introduction

An Intrusion Detection System (IDS) is a software-based application that monitors network traffic for malicious activity and provides notifications immediately if it detects anything suspicious (commonly known as an attack) in the network. Over the years, the network has seen a sharp growth in the types of attacks, and the attackers' strategies have continued to evolve. Hence IDS must handle new, diverse forms of attacks and security threats for better security and functioning of a network. According to an Anti-Virus (AV) test report by a security software testing organization, around 140 million new malware samples were detected in the year 2020 [1]. The Corona Virus Disease of 2019 (COVID-19) pandemic has also created new opportunities for cybercriminals, significantly increasing phishing attacks as attackers sought to exploit uncertainties. On average, only five percent of companies' folders are adequately protected [2]. According to the Identity Theft Resource Center's 2021 Data Breach Report, there were 1,862 recorded data breaches in 2021, surpassing the 2017 record of 1,506 breaches [3].

Three methods exist for finding any intrusion (i) A Misuse-based or Signature-Based Approach, which uses the signatures of known attacks to identify them without raising plenty of false alarms [4,5], (ii) An analysis of traditional system and network performance using anomaly-based methodologies identifies inconsistencies as deviations from typical network behavior. This technique can identify new or recent

* Corresponding author at: Department of Mechanical Engineering, Lebanese American University, Beirut, Lebanon.
E-mail address: scientificresearchglobe@gmail.com (M. Ijaz Khan).

https://doi.org/10.1016/j.asej.2024.102807
Received 19 December 2023; Received in revised form 16 March 2024; Accepted 3 April 2024
2090-4479/© 2024 THE AUTHORS. Published by Elsevier BV on behalf of Faculty of Engineering, Ain Shams University. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

(zero-day) assaults. (iii) Hybrid methods combine signature-based and anomaly-based techniques [5].

The necessity to design a reliable and adaptable IDS arises from gradually upgrading intrusion types due to sophisticated attack approaches. The growth of effective IDS faces numerous difficulties like false alarms, low detection rates and unbalanced datasets. Various methods have been developed to improve IDS using data mining and ML techniques. All of these technologies still have some challenges, making it possible for an attacker to violate the system.

ML techniques process the dataset's variables to predict the possible outcomes. However, the dataset mostly contains high-dimensional variables in which many variables are irrelevant to the task. The feature selection technique is the strategy employed by researchers to address the data dimensionality issue. The feature selection technique decreases processing time, aids in data comprehension, eliminates the "curse of dimensionality" effects and enhances the performance of predictive algorithms [6,7].

In this paper, we propose a novel based network intrusion detection system for the task of intrusion detection in large-scale computer networks. The implementation includes data gathering, data preprocessing and attack classification. Attacks are detected based on the method of ensemble learning and Recurrent Neural Networks (RNNs). Ensemble methods and Long Short-Term Memory (LSTM) networks can effectively identify intrusions by learning complex patterns from the CICIDS2017 dataset's diverse, labeled network traffic. This approach helps these techniques overcome the challenges caused by network behavior variations and noise. Random Forest (RF) excels at handling high-dimensional data and detecting anomalies, while boosting methods like Adaptive Boosting (AdaBoost) and Gradient Boost sequentially improve model performance, focusing on challenging instances within the dataset. Moreover, XGBoost enhances predictive power through regularization and parallel computation. The CICIDS2017 dataset offers a realistic and diverse collection of labeled network traffic, including both normal activity and various types of attacks like Denial of Service (DoS), Distributed Denial-of-Service (DDoS), and brute-force attempts. This allows them to test and improve Intrusion Detection Systems (IDS) classification that ultimately lead to stronger defenses against cyber threats.

1.1. Motivation

The frequency of cyberattacks has increased dramatically and ingeniously. One of the defenses against such attacks is the use of IDS. Thus, IDS needs to improve its performance by decreasing its false rates and increasing accuracy. Despite the development of numerous approaches, there are still problems that include the enormous dimensionality of data [8], its effects on computational complexity [9,10], and computational time.

In the ML approach, integrating efficient feature selection techniques has shown a successful approach to intrusion detection. The classification of network traffic data with imbalanced class distribution has shown a significant drawback in the performance of classifier algorithms. The goal of this study is to develop an IDS that improves the classification performance of imbalanced data.

1.2. Contribution

Although there has been a lot of research on IDS to address issues with accuracy, recall, precision, and false negative rates, the work continues to be challenging. Also, research on imbalanced datasets, and multi-class classification has been relatively sparse. This study aims to offer new methodologies for multi-class, and binary-class classification. The key contributions include

- It reviews and analyses the Canadian Institute for Cybersecurity Intrusion Detection Scenario 2017 (CICIDS2017) dataset, recorded over 5 days.
- Reducing the dimensionality of the CICIDS2017 dataset through the initiated Random Forest with High Correlation (RFHC) attributes.
- It reduces the CICIDS2017 dataset variables from 81 to 34 while maintaining a high accuracy of 99.71 % in multi-class, and 99.40 % in binary classification.
- Investigating best ML ensemble models RF, AdaBoost, XGBoost and Gradient Boost, and deep learning model LSTM with hyperparameter optimization to ensure the best results.
- Conducting a comparative study between ensemble ML and deep learning models with Upsampled, and Un-Upsampled data to highlight the importance of imbalanced dataset problems and the necessity of using UpSampling techniques for balancing the dataset.
- Developing a knowledge base for cyber attackers

Section 2 of this study describes related works, Section 3 describes day-wise data analytics of the dataset, and Section 4 describes the outline of the initiated architecture's preprocessing, feature selection and classification algorithms. Section 5 introduces a brief description of the dataset, Section 6 introduces the experimental setup, and Section 7 introduces results and performance evaluation followed by a conclusion and future scope in Section 8.

2. Related works

Recently, researchers focused on developing ML-based IDS using two well-known datasets: NSL-KDD, and CICIDS2017. The CICIDS2017 dataset [11] is widely regarded as the ideal dataset for training IDS due to the realistic user behavior profiles it presents. This dataset encompasses various protocols such as HTTP, HTTPS, FTP, SSH, and email at the application layer, reflecting common internet usage scenarios. Furthermore, CICIDS2017 incorporates contemporary attack patterns, ensuring relevance and up-to-date training data for IDS evaluation purposes.

A Random Forest Regressor was employed by Sharafaldin et al. [12] to identify the ideal combination of attributes for identifying each attack family. The performance of these variables was then investigated using a variety of algorithms, including K-Nearest Neighbor (KNN), AdaBoost, Multi-Layer Perceptron (MLP), Naive Bayes, RF, Iterative Dichotomiser 3 (ID3), and Quadratic Discriminant Analysis (QDA) and obtained the maximum precision of 0.98 with RF and ID3. Vijayan et al. [13] initiated an IDS that used multiple Support Vector Machines (SVM) for classification, and the Genetic Algorithm (GA) for feature selection. Their solution was built on an ordered linear combination of various SVM classifiers, where the classifiers were ranked according to the intensity of the attacks. The GA picked a set of variables to train each classifier to recognize a certain assault category. A denial-of-service IDS was initiated by the authors in [14] that used the Fisher Score technique for feature selection and SVM, KNN, and Decision Tree (DT) as the classification algorithms. Models like SVM, KNN, and DT, their IDS had success rates of 99.7 %, 57.76 %, and 99 %, respectively.

An approach called Data Dimensionality Reduction (DDR) in which XGBoost, SVM, Conditional Inference Trees (CTree), and Neural Network (Nnet) classifiers were used to evaluate their initiated strategy [15]. In this dataset, 36 characteristics were chosen, and the maximum accuracy of 98.93 % was obtained using XGBoost. However, the authors excluded Monday traffic which only contains benign traffic. This study has considered all the files of the dataset that represent different classes of network traffic and was able to achieve 99.91 % accuracy with 10 classes with Upsampled data. An IDS based on feature selection and ensemble classifier was initiated by Zhou et al. [16]. For dimensionality reduction, the heuristic algorithms Correlation-based Feature Selection (CFS) and Bat Algorithm (BA) are also suggested. The ensemble technique combines the C4.5 and RF algorithms to classify the intrusions

Table 1
Summary of previous work related to IDS.

Feature Selection method	Classification method	Selected Variables Count	Accuracy	Precision	Recall	F1-Measure	Dataset
	KNN			0.96	0.96	0.96	
	RF			0.98	0.97	0.97	
	ID3			0.98	0.98	0.98	
	Adaboost			0.77	0.84	0.77	
Random Forest	MLP			0.77	0.83	0.76	
Regressor [12]	Naïve Bayes	54	–	0.88	0.04	0.04	
	QDA			0.97	0.88	0.92	
Fisher Scoring [14]	KNN	30	0.9997			0.9997	CICIDS2017 NSL-KDD, AWID, and CIC-IDS2017
DDR [15]	XGBoost	36	98.93	0.9985	0.9968	–	CIC-IDS2017, UNSW-NB15, and NSL-KDD
CFS_BA [16]	Voting contains (C4.5, RF, ForestPA)	13	99.89	–	–	–	CICIDS2017(Wed)
HPS-KODE [17]	Voting contains (K-means, One- class SVM, DBSCAN,and Maximization-Expectation, (KODE))	8	99.9	–	99.9	–	BoT-IoT
Deep neural network [22]	XG Boost algorithm	41	99	–	96.64	–	NSL-KDD, CIDDS-001, and CICIDS2017
		38	96	–	–	–	
		78	92	–	–	–	
Recursive feature elimination [40]	RF Classifier	17	99.62	98	100	99.91	Credit card fraud Dataset
Information Gain [40]		12	99.83	98	100	99.90	
Chi-Squared [40]		8	99.78	99	99	99.88	
Ensemble Features [40]		7	99.6	100	99.4	99.6	

using Network Security Laboratory Knowledge Discovery in Databases (NSL-KDD), Aegean Wi-Fi Intrusion Dataset (AWID), and CICIDS2017 datasets. Jaw and Wang [17] initiated a comprehensive IDS approach, a wrapper methodology based on GA which is a feature selection technique, and K-means, One-Class SVM, Density-Based Spatial Clustering of Applications with Noise (DBSCAN), and Expectation-Maximization (KODE).

The authors in [18] initiated an architectural concept for Risk Assessment (RA) of the information system using ML algorithms using the CICIDS2017 dataset. In this study, ML methods for RA evaluation included KNN, Naive Bayes (NB), GradientBoost tree, RF, and DT. A total of 15 ML models examined the risk matrix for RA. A novel Feed-Forward Neural Network (FNN) is suggested for binary and multi-classification, using the Botnet of Things Internet of Things (BoT-IoT) dataset [19]. To identify DDoS assaults, Deep Sparse Autoencoder-based Framework (EDSA) was initiated [20]. A novel deep neural network model was initiated by Ahmad et al. [21] for recognizing assaults from both legitimate and fraudulent sources.

Gupta et al. [22] recommended ensemble algorithms for class imbalance in the CICIDS2017 dataset. The ensemble learner is applied in separating and differentiating between normal and suspect traffic network attacks. These attacks are fed into XGBoost, and RF to identify big attacks. RF predicted the classification with 92 % accuracy. All earlier works (refer to Table 1) need to be more consistent in the measured values but lacking in robustness and efficiency. Hence the initiated system increases the robustness by using advanced feature selection techniques on ensemble learning algorithms and deep learning models. Previous studies have encountered challenges with discrepancies in their performance metrics such as the accuracy reached 96 % [16], the False Alarm Rate (FAR) was 0.15 % [17], the accuracy was 98.3 % but with a FAR of 0.14 %. To address this issue, the proposed system enhances robustness by implementing an advanced feature selection approach based on hybrid ensemble learning algorithms. The primary objective is to achieve both high accuracy and minimal FAR, ensuring improved performance across multiple evaluation metrics. The existing work does not include any HTTPS traffic in the generated dataset, which limits the applicability of the dataset for modern network security analysis [12].

3. Day-wise data analytics on the CICIDS2017 dataset

The dataset is recorded over 5 days, namely Monday, Tuesday, Wednesday, Thursday, and Friday. Moreover, Thursday splits into a morning session and an afternoon session accordingly as web attacks and infiltration. Friday splits into three sessions i.e., morning, afternoon PortScan and DDoS. We considered a few important variables as shown in Table 2, which gives insight into the minimum and maximum values of the variables.

Table 2
Descriptive statistics of selected features.

Features	Minimum Value (Bytes)	Maximum Value (Bytes)	Features	Minimum Value (Bytes)	Maximum Value (Bytes)
Average Forward Segment Size	0	5940.857	Destination Port	0	65,535
Average Backward Segment Size	0	5800.5	Total Length of Forward Packets	0	2,428,415
Average Packet Size	0	3893.33	Total Length of Backward Packets	0	655,453,030
Down/Up Ratio	0	156	Total Backward Packets	0	291,922
Total Forward Packets	1	219,759			

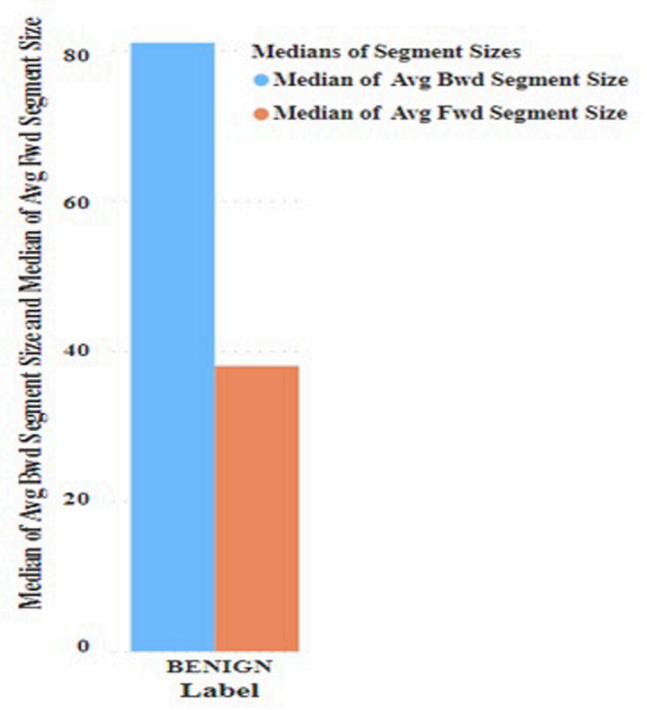

Fig. 1. Comparison between midpoints of *Total Backward* and *Forward Segment Size.*

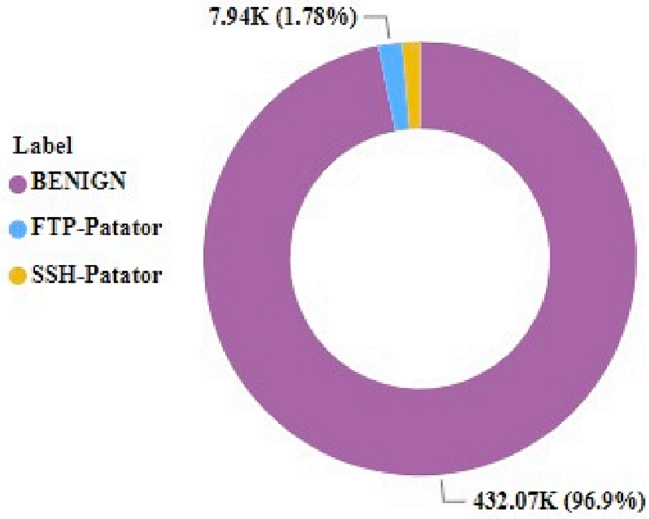

Fig. 2. The concentration of labels in Tuesday's data.

3.1. Monday's data

In Monday's data, we see only pure Benign flow data i.e., the non-harmful data which is a very usual type of network flow we all come across. While looking into medians of *Average Forward Segment Size* and *Average Backward Segment Size* as shown in Fig. 1, the difference between the medians infers that data flow is high from receiver to sender over sender to receiver.

3.2. Tuesday's data

In Tuesday's data, normal network flow i.e., Benign flow and attacks like File Transfer Protocol-Patator (FTP-Patator) and SecureShell-Patator (SSH-Patator) are considered. Patator is a brute-force password cracking tool that is designed to automatize the attempting of password combinations that can lead to intrusion of various systems. Here, FTP-Patator data is 1.78 %, SHH-Patator is 1.32 %, and the rest is Benign traffic as shown in Fig. 2.

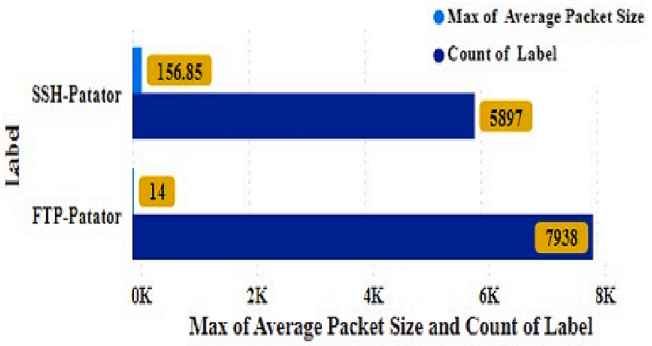

Fig. 3. Comparison between Patators in terms of *Average Packet Size.*

Fig. 3 shows the comparison between the *Maximum Average Packet Size* (maximum size of the data packet that was transmitted from sender to receiver) and the count of both attacks (the number of times the attack has occurred). Here, SSH-Patator has the *Maximum Average Packet Size* and lesser count of labels comparatively, and this shows SSH-Patator was carried in larger packets than FTP-Patator.

3.3. Wednesday's data

In Wednesday's data, a diversity of network traffic flow with the inclusion of Benign flow and attacks like DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS Slowloris, and Heartbleed. Fig. 4 emphasizes the enormity of the types of network flows that have occurred, in terms of percentages. Heartbleed attack has occurred only 11 times whereas Benign flow has occurred 440,031 times, which makes it difficult to show Heartbleed in the pie chart since it's negligible when compared.

Fig. 5 shows the standard deviation in *Total Forward Packets* of all types of attacks and Benign flow data. From Fig. 4, we can infer that Heartbleed is the least recorded attack, but it has the major deviation in *Total Forward Packets* after Benign flow. It suggests that the Heartbleed attack stands out in terms of affecting forward packets though it is less frequent and shows unique patterns when compared.

Fig. 6 shows the comparison between labels with the *Sum of Total Forward Packets* to see the movements of the types of traffic existing. It is observed that Benign, the usual traffic flow is major with a value of 5242.17 K as the sum of the *Total Forward Packets* i.e., the major movement in the network flow, and Heartbleed having the least about 28.2 K. It described that it is the least occurred movement in the network flow of Wednesday.

Fig. 7 shows the maximum *Down/Up ratio* of types of network traffic from the data and we can see it's mostly proportional to the number of times they have occurred. But in comparison with Fig. 6, DoS Slowhttptest has almost closer movement as Heartbleed has and, here we see that DoS Slowhttptest has more *Down/Up ratio* than DoS Hulk, which has occurred more than DoS Slowhttptest, we can infer that DoS Slowhttptest has a little different way of attacking that the others.

3.4. Thursday's data

The data recorded on Thursday comprises two sessions such as morning and afternoon containing web attacks, and Infiltration attacks respectively.

3.4.1. Thursday Morning session

The Web Attacks in this dataset are Brute Force, Cross-site Scripting (XSS) and Structured Query Language (SQL) Injection. However, a

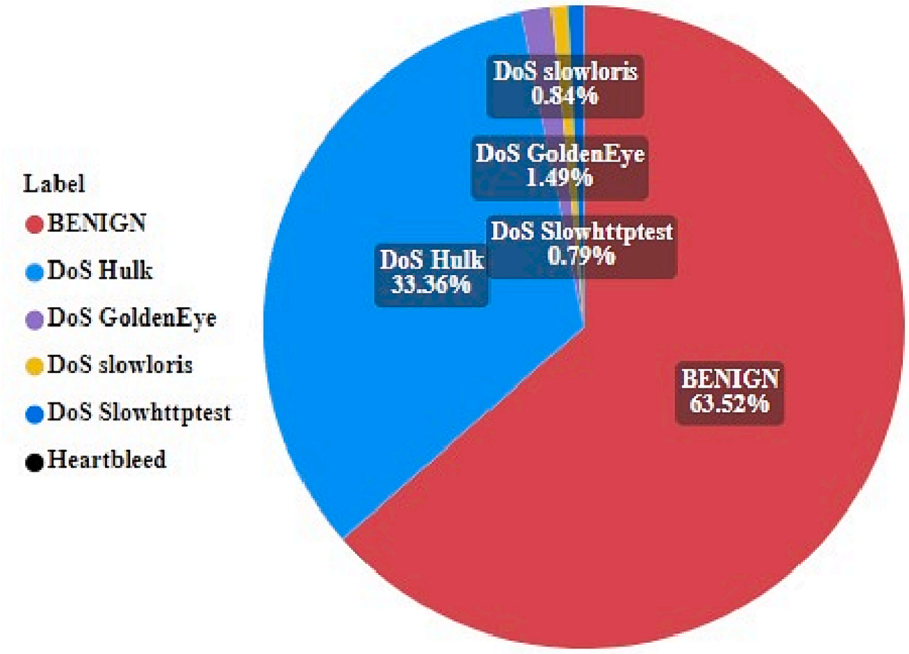

Fig. 4. Mixture of Network flows in Wednesday's data.

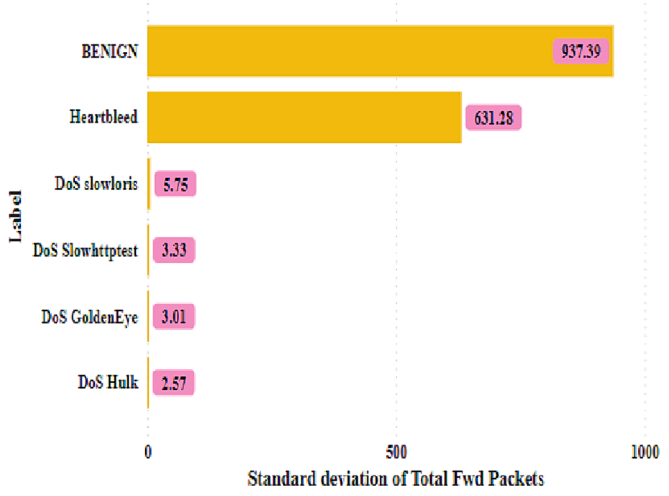

Fig. 5. Standard Deviation of *Total Forward Packets* in terms of Labels.

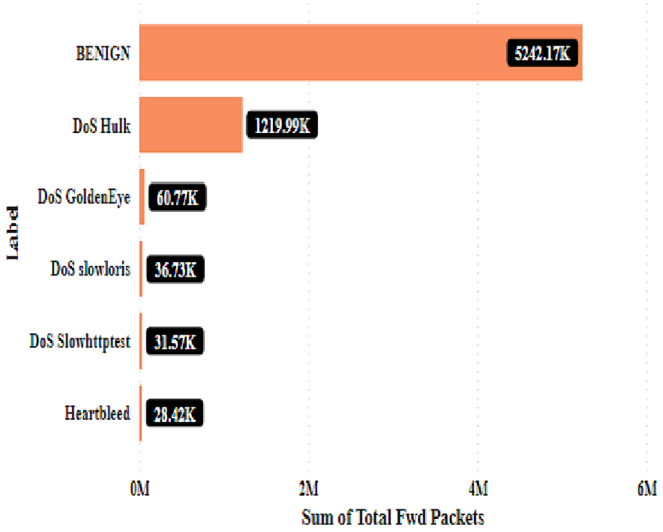

Fig. 6. Labels versus Sum of *Total Forward Packets*.

major part of the data is the Benign flow traffic, i.e., the regular type of network flow, and the least is the SQL Injection as shown in Fig. 8.

In Fig. 9, compare all the Web Attacks namely, SQL Injection, Brute Force, and XSS along with Benign flow in terms of the midpoint of the *Total Length of Backward Packets* along with *Forward Packets*. According to Fig. 8, SQL Injection had the least movement, but it has the highest midpoint of lengths of both the packets compared to either of the attacks or even Benign traffic data. This proves that it transmitted enormous amounts of data in a smaller number of packets.

3.4.2. *Thursday afternoon session*

Thursday's Afternoon data contains two categories Infiltration attack and Benign flow data. It is observed that variance in *forward packets* and *backward packets* is more in Infiltration than in Benign, and that could imply diversity of destination, variability in packet sizes, and irregularities as shown in Fig. 10. Since Infiltration is a cyberattack that accesses unauthorized entities to kill the system's integrity and perform malicious activities leads to an irregularity.

3.5. *Friday's data*

Morning and two-afternoon sessions with different intervals were recorded on Friday. Morning data consists of Benign flow traffic and Bot attacks. Two types of Afternoon data are considered which consist of a DDoS attack with Benign, and a PortScan attack with Benign traffic.

3.5.1. *Friday morning session*

Fig. 11 shows that the ratio of the Benign flow data of the midpoint of the *Total Length of Backward Packets* to the *Forward Packets* is 11:5, and for Bot attack data the ratio is 3:1 major part being the median of the *Total Length of Backward Packets*. It is inferred that higher backward packets tend to be longer compared to forward packets for Bot attacks.

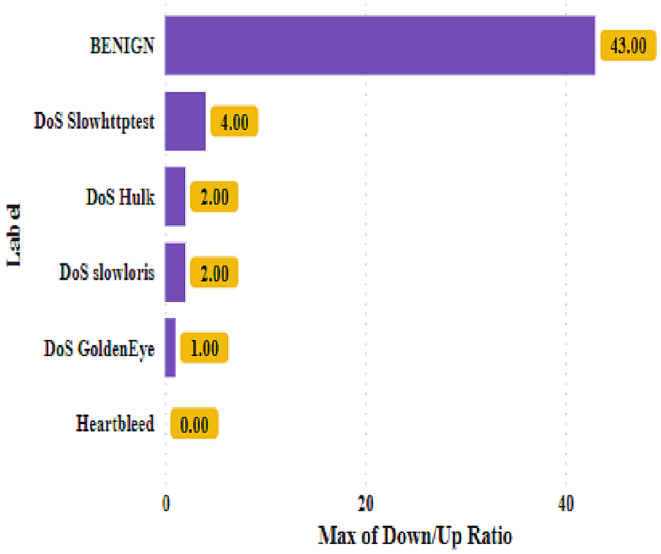

Fig. 7. Maximum *Down/Up ratio* of Wednesday Labels.

Fig. 8. The concentration of web attacks in a Tree Map.

3.5.2. *Friday afternoon session*

It contains two different datasets with different attacks along with some benign data. One of the data consists of DDoS attacks and benign flow traffic. DDoS attempts to disturb the usual activities on the network by causing traffic to the attacker targets causing the unavailability of valid users. In Fig. 12, plots show the comparison between *Total Forward* and *Backward Packets* when it is a DDoS attack and Benign flow. It infers that the frequency of packets is higher in the Benign flow than DDoS attack because the Benign flow is the regular network flow. Usually, attacks send huge amounts of packets than Benign flow but in a smaller number of packets.

Fig. 13 shows the comparison of the midpoint of the *Down/Up ratio* (ratio of the movement of downloading and uploading) with the *Destination Port.*

The *Down/Up ratio* of the PortScan attack range is 0–5000 which is highlighted in green, and the Benign flow range is 50,000–55,000 highlighted in pink bars. It is inferred that at higher *destination ports*, there is more flow in Benign data, and less data flow in PortScan.

4. Methodology

The raw dataset contains a lot of variables in which many variables are not contributing to detect and classify the attacks. Hence, RFHC is initiated to identify the relevant variables that diminish the dataset dimensionality. This dimensionality reduction decreases the training time and storage space. The RFHC identifies the relevant and irrelevant variables using RF and correlation. The relevant variables of the dataset are fed into the ensemble learning algorithms to classify the input. Fig. 14 shows the initiated model's block diagram. The initiated model is implemented in IDS to classify the attacks.

4.1. *CICIDS-2017 dataset pre-processing*

The process starts with preparing the data cleaning and transforming it to improve the analysis. Then, the feature selection selects the most relevant variables or attributes from the data to be used in the model. The dataset contains eight Comma-Separated Value (CSV) files that are concatenated into one data frame. The data frame contains 79 columns and 23,62,181 instances. The columns *Bwd PSH Flags, Bwd URG Flags, Fwd Avg Bytes/Bulk, Fwd Avg Packets/Bulk, Fwd Avg Bulk Rate, Bwd Avg Bytes/Bulk, Bwd Avg Packets/Bulk,* and *Bwd Avg Bulk Rate* were removed that contain no values as indicated by descriptive statistics. Finally, the resulting data frame had 71 columns. A few class labels were renamed that had unprintable characters. 'Web Attack ◆ Brute Force' was renamed into 'Brute Force', 'Web Attack ◆ XSS' was renamed into 'XSS' and 'Web Attack ◆ SQL Injection' was renamed into 'SQL Injection'.

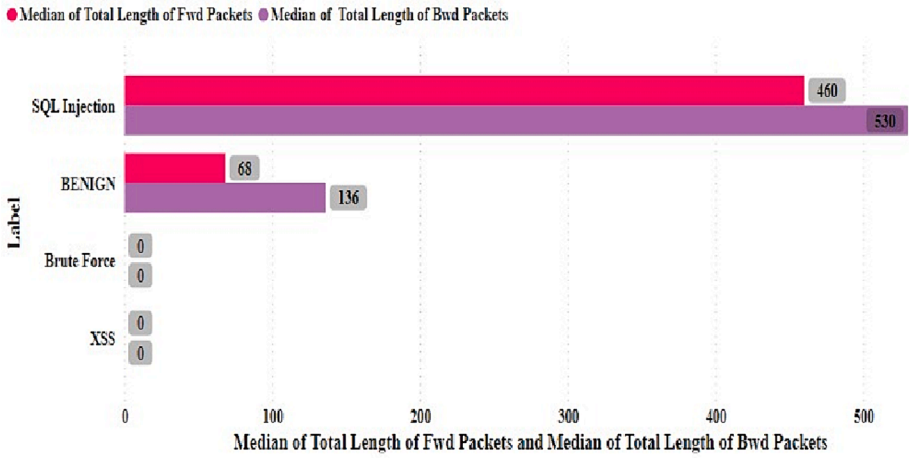

Fig. 9. Comparison between midpoints of *Total Length of Backward* and *Forward Packets.*

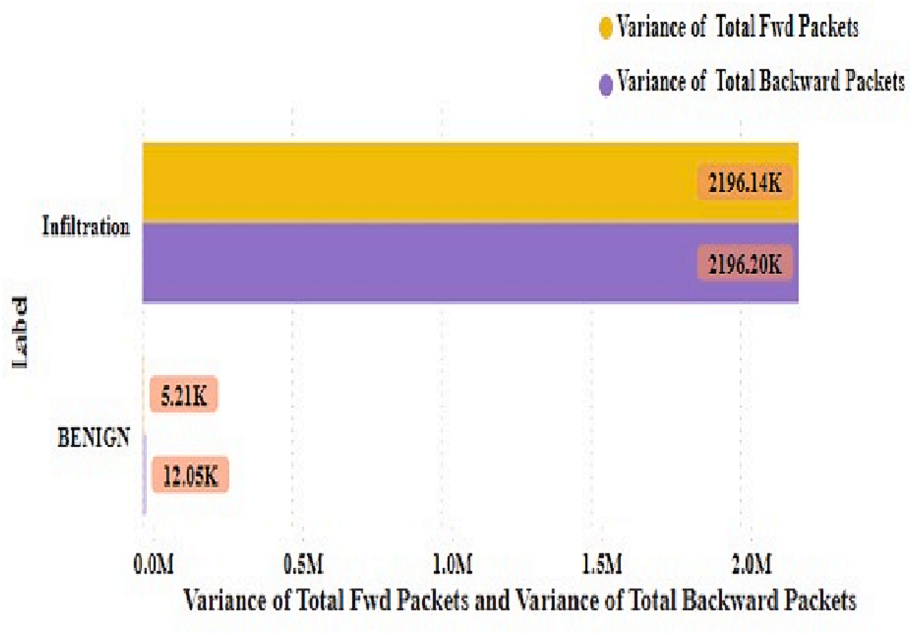

Fig. 10. The variance of *Total Forward Packets* in terms of *Labels.*

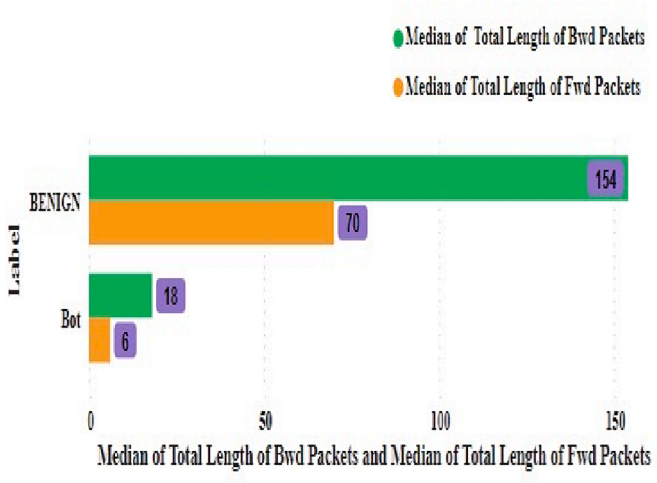

Fig. 11. Comparison between midpoints of *Total Backward* and *Forward Packets.*

4.2. *Feature selection using Random Forest and highly correlated pairs (RFHC)*

This RFHC method is used to reduce the dimensionality of data and focus on more important variables. It helps to reduce overfitting by selecting the most informative and relevant variables, which improves the model's predictive performance. The RF classifier and correlation function are utilized to identify the relevant and irrelevant variables of the dataset.

4.2.1. *Identify feature's importance using RF classifier*

The high-dimension dataset fed into the RF classifier with 70 variables. Fig. 15 depicts the process of feature importance computation by RF algorithm. The RF starts with initial hyperparameters to detect the importance of the feature. The gridSearchCV method is applied to select the optimal parameters such as estimators, depth, entropy, leaf nodes count, etc., The gridSearchCV is configured with various number of estimators (10, 25, 50), criterion [gini, entropy, log loss function], and maximum depth (4, 5, 6, 7). The heuristic search selects the subset of variables using the importance score. The feature importance is calculated using information gain and entropy. This process is repeated with

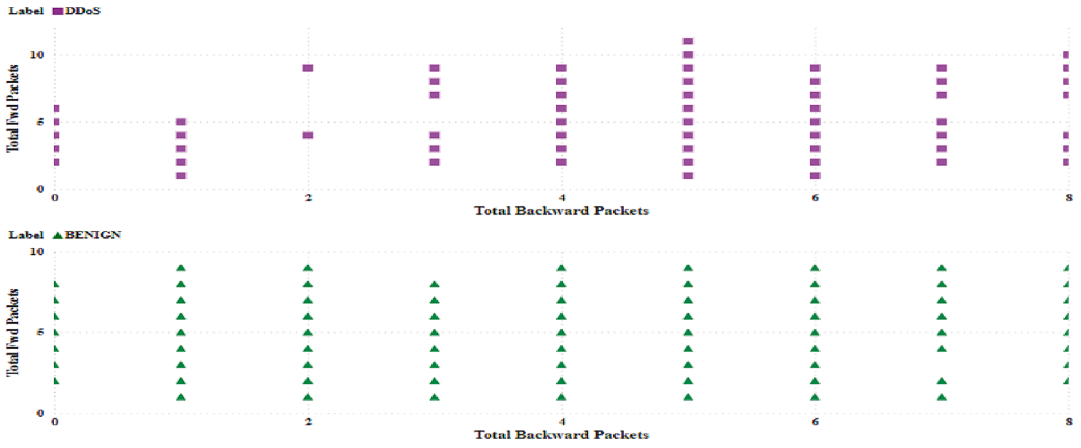

Fig. 12. DDoS with *Total Backward Packets* versus Benign with *Total Backward Packets.*

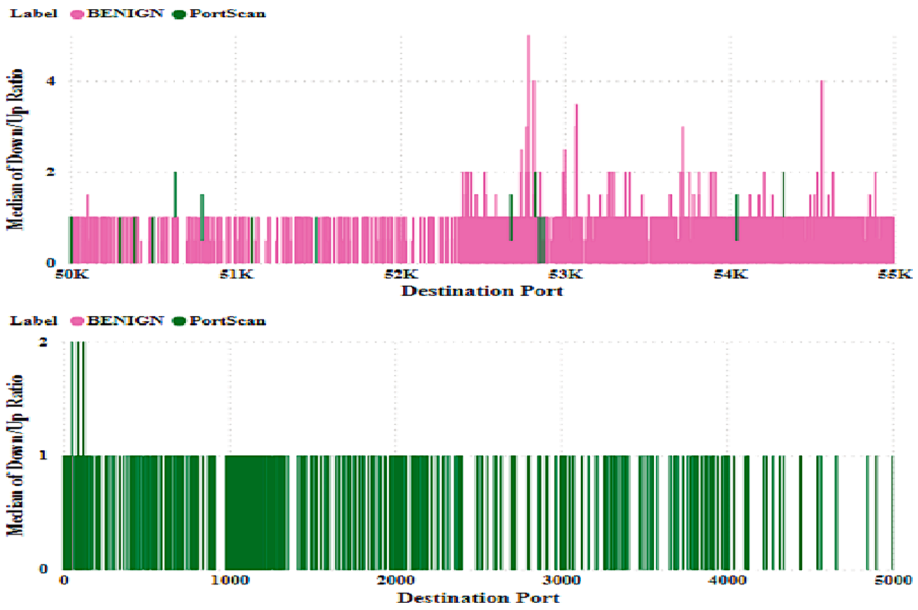

Fig. 13. Ranges of Benign and PortScan.

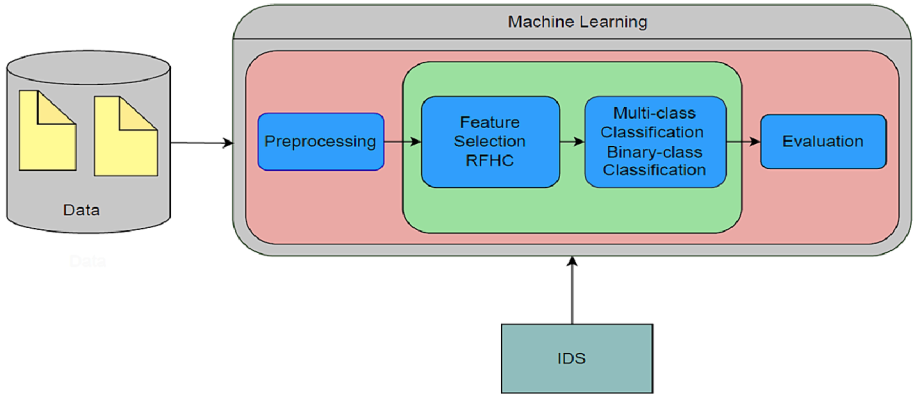

Fig. 14. Initiated Model Architecture.

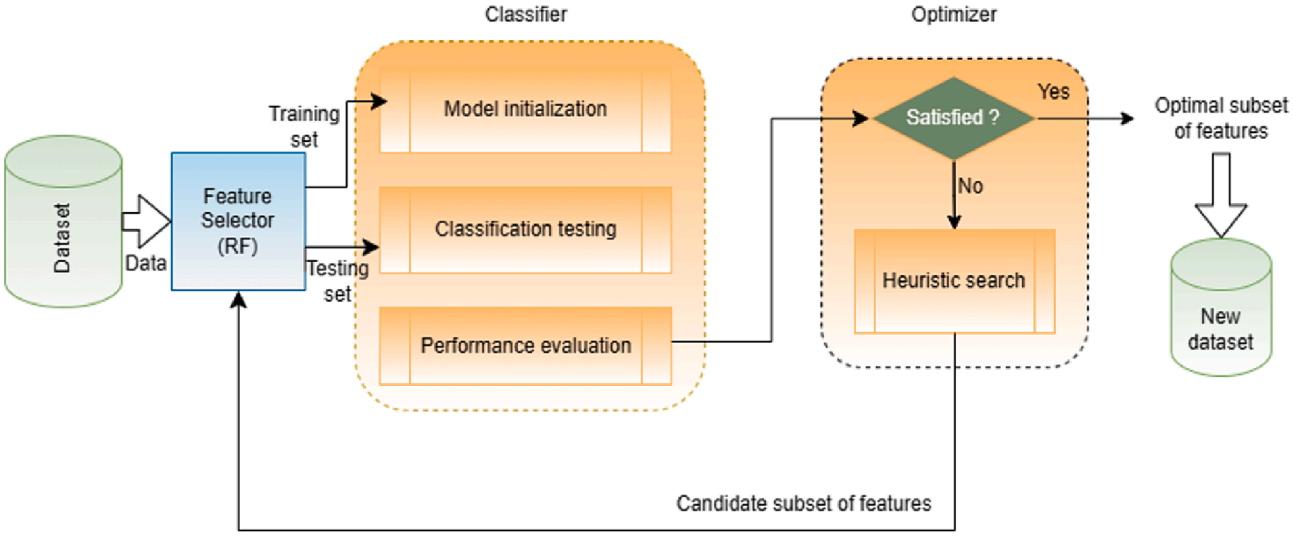

Fig 15. Identify feature importance value using RF classifier.

Table 3
Features dropped using RFC below the importance threshold 0.001.

Bwd IAT Std	Idle Mean	Active Std	ECE Flag Count
SYN Flag Count	Idle Min	Idle Std	RST Flag Count
Fwd PSH Flags	FIN Flag Count	CWE Flag Count	Idle Max
Active Max	Active Min	Fwd URG Flags	Active Mean

various combinations of parameters using a validation process. The optimal parameters selected by gridSearchCV are 10 estimators, and depth 5. The testing dataset is utilized to validate the RF classifier model (cv = 5) for optimizing the feature importance score calculation. Finally, the RF classifier results in the optimal subset of variables by dropping low importance variables of the dataset.

The RF classifier identifies the high and low-importance variables using the score. This study applied the various threshold values in the

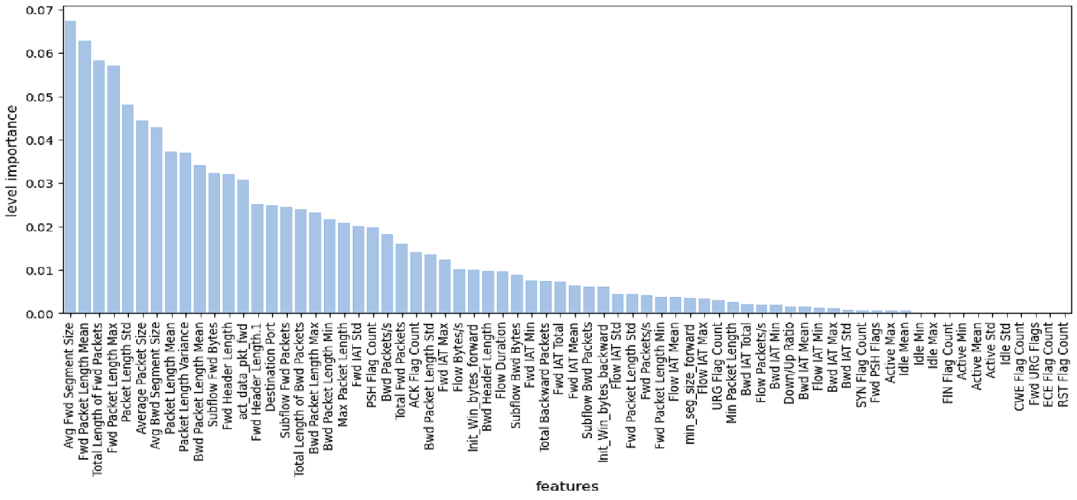

Fig. 16. Variables importance calculation using RF Classifier.

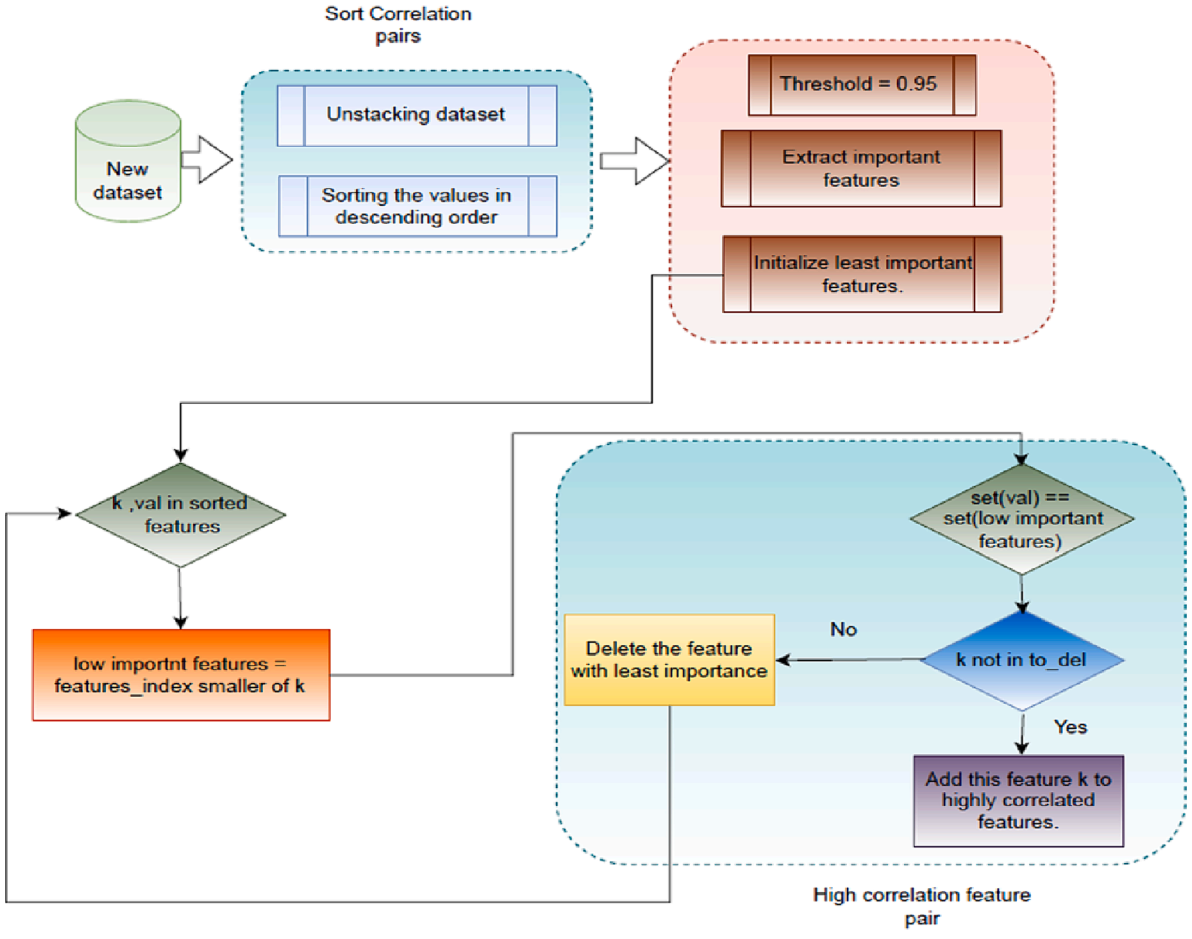

Fig 17. Feature selection by detecting highly correlated pairs.

range of 0.001–0.005 with step size 2. The optimal result is generated by threshold 0.001. Table 3 shows the result of the initial process identified 16 irrelevant variables that were dropped using the RF classifier as they were below the threshold of 0.001.

Fig. 16 depicts the variable's importance identified by the RF classifier. *Avg Fwd Segment Size* is the most important feature with a level importance of 0.068, followed by *Fwd Packet Length Mean* with a level importance of 0.063 and so on. *Avg Fwd Segment Size* is recognized as the most important feature by the RF classifier as it provides information about the average size of data segments being transmitted in the forward direction.

4.2.2. Highly correlated variables identification

The second process applied a threshold value greater than 0.95 to identify the highly correlated variables. Fig. 17 shows the process of variable selection using the correlation function. The result of the RF classifier is fed as input that is sorted in descending order. The correlation threshold value is set as 0.95 to identify the highly correlated variables. It identified the relevant variables that are less than the 0.95 threshold. Also, low-important variables are identified from the sorted variables using the smaller index value. Finally, highly correlated variables and low important variables are identified to drop from the dataset. It diminishes the dataset dimensionality.

High correlation pairs are calculated using Pearson's correlation.

Table 4
Variables dropped using highly correlated pairs threshold ≥ 0.95.

Bwd IAT Mean	Fwd IAT Total	Bwd Packets Length Std	Packet Length Variance
Flow Packets/s	Total Backward Packets	Total Fwd Packets	Packet Length Mean
Flow IAT Max	Subflow Bwd Bytes	Fwd Header Length.1	Average Packet Size
Fwd Packet Length Std	Fwd Packet Length Mean	Subflow Fwd Bytes	Max Packet Length
Subflow Bwd Packets	Bwd Packet Length Max	Bwd Packet Length Mean	Total Length of Bwd Packets

The correlation function results in the 20 irrelevant variables that were dropped using a high correlation of pairs as shown in Table 4. The optimal 34 variables are selected from the CICIDS-2017 dataset. This study applied two threshold values such as less than 0.001 and greater than 0.95. Initially, apply the below 0.001 threshold value to identify the irrelevant variables in the dataset.

4.2.3. Sampling using SMOTE and RandomUnderSampler

A new temporary column is introduced to distinguish between Normally and Attack types of traffic. Fig. 18 shows the distribution of Normal traffic, which outweighs the Attack traffic, and creates a class imbalance that affects the model of being biased toward the majority class. To avoid this bias, the downsampling technique

$$R = \frac{n(xy) - xy}{\sqrt{[nx - (x)][ny - (y)]}}$$

$$threshold\_correlation = 0.95$$

```
start_with_correlation_matrix
extract the important features and save in array
Initiate with first element as important feature
identify_highly_correlated_features(set_of_features):
    initialize least important feature = [ ]
    for k, val in sorted_features {
        low_important_features = features_index_smaller_of_k
        #Find the highly_correlation_features_pair
        if (set(val) : set(low_important_features) {
            if k not in to_del:
                Add this feature k to highly correlated features
            else:
                Delete the feature with lowest importance
    return Deleted features
```

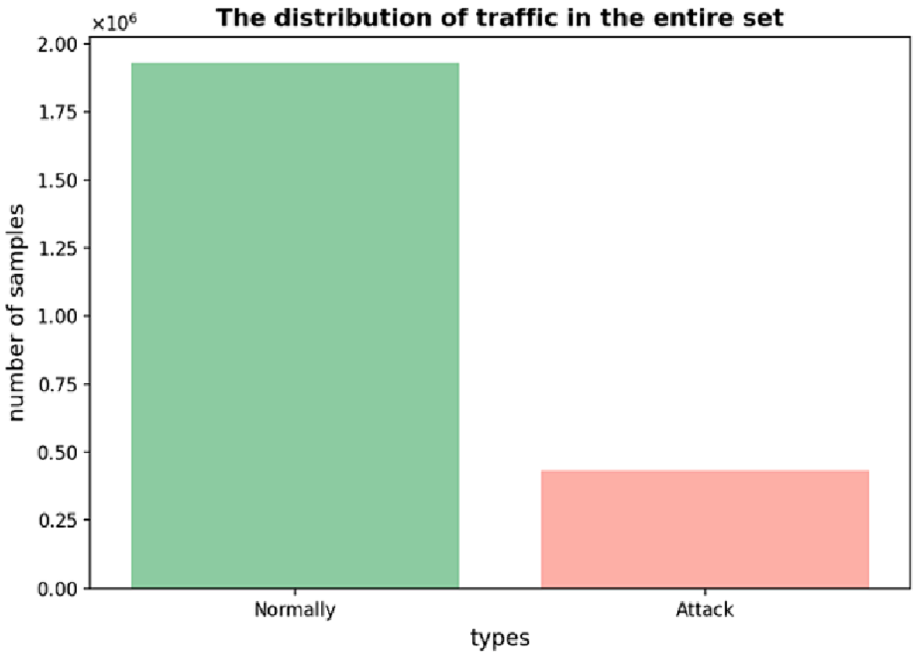

Fig. 18. Label count before downsampling.

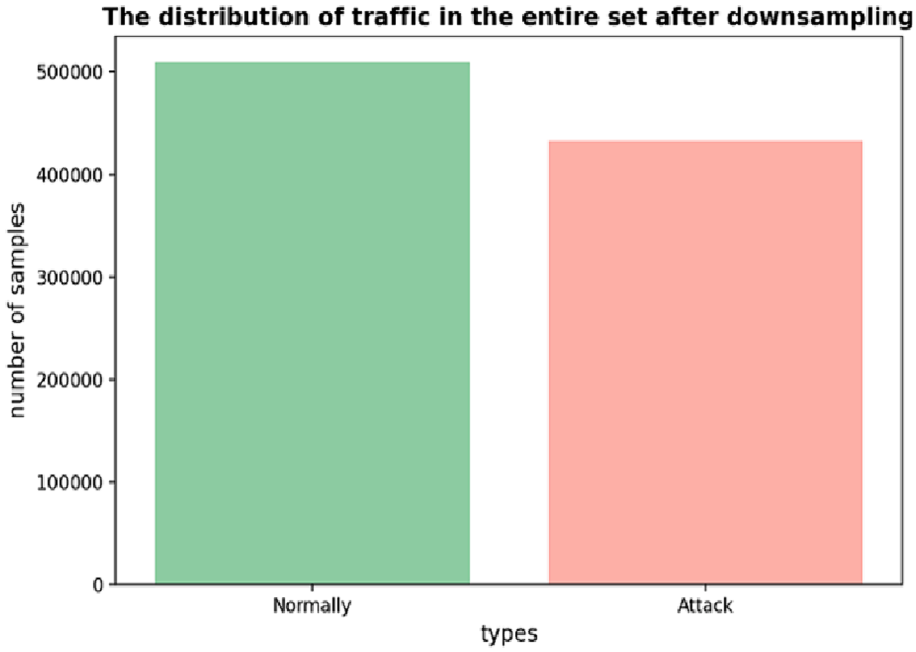

Fig. 19. Label count after downsampling.

RandomUnderSampler is introduced which helps to reduce the count of the majority label.

RandomUnderSampler randomly selects a subset of instances from the majority class to achieve the desired ratio (0.85) to equal the number of majority and minority instances. Fig. 19 depicts the label distribution after applying downsampling where the Normally label count is downsampled from 1.85×10^6 to 5×10^5.

ML algorithms converge faster during the backpropagation and perform better when variables are on a similar scale. Rescaling of the variables was performed using the maximum and minimum values of each feature. These variables are then normalized to scale them in the same range $[0, 1]$ using the following equation

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

Where X_{max} is the maximum value of the feature, X_{min} is the minimum value of the feature.

The train-test split was performed with a training size of 0.7, and an additional held-out validation set was created to evaluate the neural networks' predictions on the Upsampled training set with a training size of 0.8. The number of attack types in the target class is 10, and the distribution of each class is highly imbalanced as shown in Fig. 20.

The count of Benign Flow data is 356,589 while XSS, Infiltration and SQL Injection are 464, 25 and 15 respectively. The class imbalance can

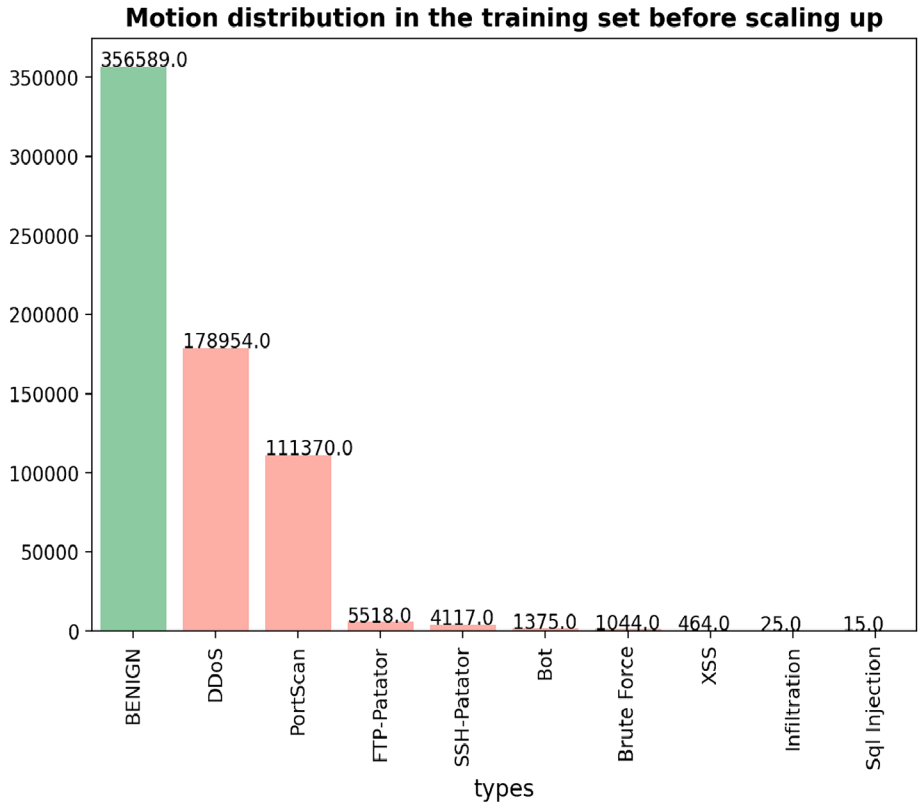

Fig. 20. Label count of traffic types before oversampling.

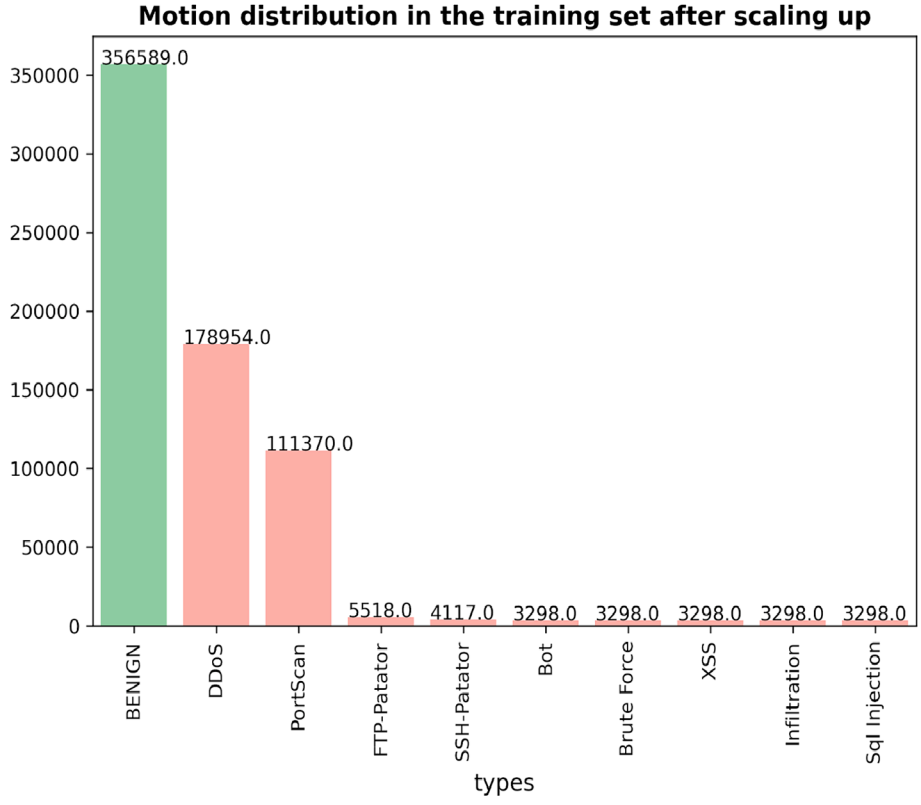

Fig. 21. Label count of traffic types after oversampling.

Table 5
Summary of preprocessed dataset.

Description	Initial Dataset	After feature reduction by RFHC	After over sampling by SMOTE	After Under Sampling	Class	
					Binary	Multi-class
Number of samples	2,362,181	2,362,181	673,038	936,073	936,073	673,038
Number of features	79	34	34	34	34	34

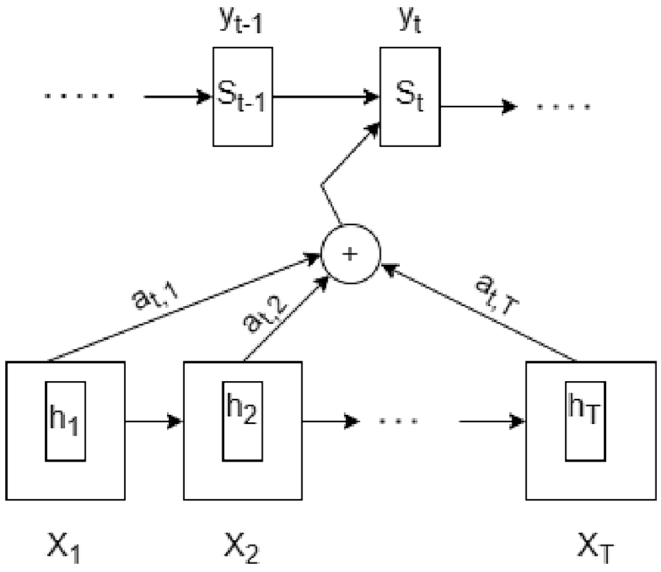

Fig. 22. Attention Mechanism.

lead to biased models that perform poorly in predicting the minority class. The Imbalance Ratio (IR) can be defined as

$$ImbalanceRatio = \frac{Majorityclassinstances}{Minorityclassinstances} \quad (2)$$

High IR values in the data will result in classifiers being less accurate and reliable.

An imbalanced class distribution problem arises due to the quantity of traffic and some sorts of anomalies as shown in Fig. 20. This provides a brief description of SMOTE for handling imbalanced datasets.

SMOTE evens the distribution of the classes [23]. SMOTE generates synthetic samples from the minority class by interpolating new instances between existing instances of the class as illustrated in Fig. 21. SMOTE is only performed on train data to prevent data leakage on the test set. The minority class with respect to label count is increased to 3298, 3298, 3298 for XSS, Infiltration and SQL Injection respectively. One-hot encoding is applied to transform categorical variables into numerical representations that can be easily understood for multi-class classification problems. Xu et al. [25], Xia et al. [26] and Li and Sun [27] used AI technique to elaborate the importance of heterogeneous traffic-agents using knowledge correction data-driven model, lightweight resemblance detection for efficient post-deduplication delta compression and RBF neural network optimal segmentation algorithm in credit rating. General framework with quantifiable privacy preservation for destination prediction in LBSs, semi-supervised probabilistic collaborative learning model for online review spammers detection and thin-film artificial intelligence transistors is examined by Jiang et al. [28], Wu et al. [29] and Xu and Shin [30]. Label encoding was performed to transform categorical variables into numerical representation for binary-class classification problems. Label encoding assigns one label as 0 and the other label as 1, effectively converting the categorical variable into a binary numeric representation. Table 5 shows the description of dataset before and after applying proposed feature selection and sampling approaches.

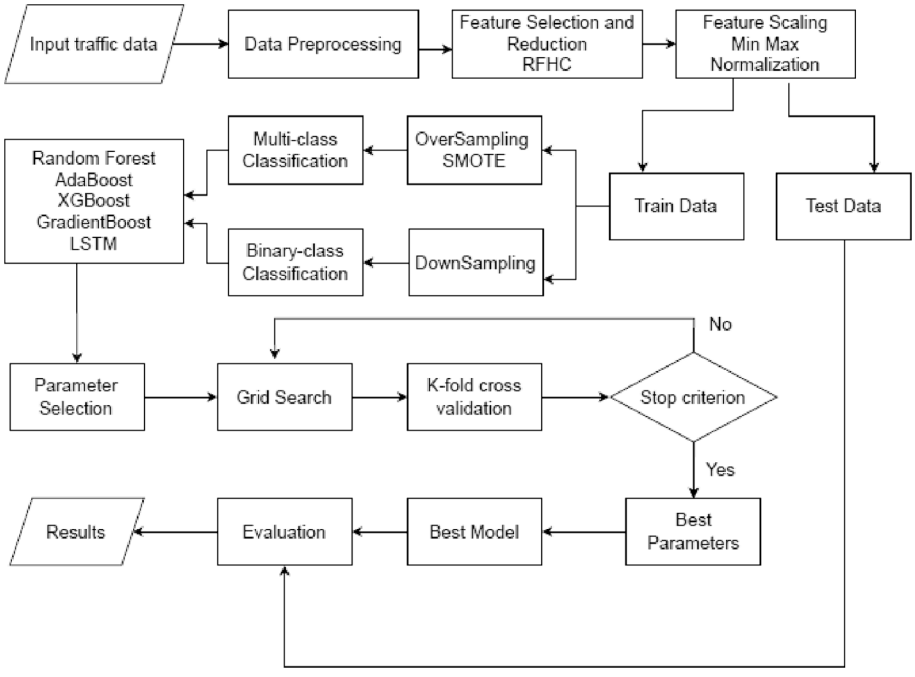

Fig. 23. Architectural diagram of attacks classifier.

4.3. Ensemble machine learning models

4.3.1. Random Forest

RF is constructed using a multitude of DTs [24]. Instead of depending on a single tree, it leverages predictions from all these trees to predict the ultimate outcome by considering which predictions assembled the highest number of votes. The RF is a bagging method that lowers the overall variance of these predictions by combining numerous learners (such as DTs) that are each fitted on distinct bootstrapped data and averaging their predictions.

$$\widehat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B}\widehat{f}^{b}(x) \tag{3}$$

Where B has bootstrapped samples of training data, and $\widehat{f}^{b}$ represents model estimator. RF uses a feature bagging technique which offers the benefit of substantially reducing the correlation between each DT, enhancing its average prediction accuracy. Time complexity is $O(nlogn)$. Train-time complexity takes $O(t \times u \times nlogn)$, Test-time complexity takes $O(t \times logn)$, Where t represents the number of trees, and u denotes the number of variables after splitting.

Hyperparameters and tuning parameters for RF are *n_estimators(50)*, *min_sample_leaf(20)*, *n_jobs(−1)*. n_estimators depict the number of trees in the forest. The higher number of trees, higher the precision of the outcome. However, this parameter makes the model slower. For hyperparameter tuning the model has been tested with range of *n_estimators* ranging from 10 to 50. *min_sample_leaf* depicts the minimum number of samples required to be at a leaf node. Smaller the value of *min_sample_leaf*, model is more vulnerable to detecting noise. *n_jobs* depict number of jobs the model can run parallelly, hence number of processors it can use. Hyperparameter range for *min_sample_leaf* ranges from 1 to 20 and for *n_jobs* 1 and −1.

4.3.2. AdaBoost

AdaBoost follows a stage-wise approach, employing multiple weak learners to create strong learners. In the beginning, the model assigns equal weights to all samples, but subsequently, it assigns larger weights to points that were initially misclassified. In the subsequent models, points with higher weights are given greater importance, and the process continues until a reduction in error is achieved. The algorithm for AdaBoost follows

1. Initialize the weight of each data point with N samples $w_i = \frac{1}{N}$
2. For $m = 1$ to M:
   a. Sample the dataset using w_i^m to obtain x_i
   b. Fit the classifier f_m using x_i
   c. Compute $\in = \frac{\sum_{y_i \neq f_{m(x_i)}} W_i^m}{\sum_{y_i} W_i^m}$, y_i is ground truth value
   d. Compute $\alpha_m = \frac{1}{2}\ln\frac{1-\in}{\in}$
   e. Update $w_i^{m+1} = w_i^m e^{-\alpha_m y f_m(x)}$
3. Predictions $f(x) = sign[\sum_{m=1}^{M}\alpha_m f_m(x)]$

Train-time complexity is $O(n \times p \times n_{trees})$, and Test-time complexity is $O(p \times n_{trees})$, Where n denotes the data points, p represents number of variables, and n_{trees} denotes number of estimators. Hyperparameters and tuning parameters for AdaBoost are *n_estimators(50)*, *learning_rate(0.97)*. *n_estimators* is the same parameter used in RF and it ranges from 10 to 50 for hyperparameter tuning. *learning_rate* controls the loss function used for calculating the weights of the model. It highly depends on *n_estimators*. The range of *learning_rate* ranges from 0.1 to 1 with step of 0.1.

4.3.3. Gradient Boost

Gradient Boosting is a powerful boosting technique that transforms multiple weak learners into strong ones. In each subsequent step, a model is trained using gradient descent to minimize the loss function of the previous model, which could be metrics like mean squared error or cross-entropy. The algorithm computes the gradient of the loss function with respect to the current ensemble's predictions during each iteration. It then trains a new weak model to minimize this gradient. The ensemble is updated with the predictions from the new model, and this process repeats until a stopping condition is met.

$$L(f) = \sum_{i=1}^{N} L(y_i, f(x_i)) \tag{4}$$

Where $f(x_i)$ is the function f(x) that maps the input variables X to the target variables y.

$$g_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x_i)=f_{m-1}(x_i)} \tag{5}$$

Where g_{im} is the gradient of loss function $L(f)$, Train-time complexity takes $O(M \times n \times d \times logn)$, and Test-time complexity takes $O(Mlogn)$, M, *nandd* denotes the number of trees, samples, and height of the tree respectively. Hyperparameters and tuning parameters for Gradient Boost are *learning_rate*(0.69), *n_estimators*(40), *min_sample_split*(0.2), *min_sample_leaf*(0.1) and *max_features*(34). *min_sample_split* defines minimum number of observations which are required for splitting and ranges from 0.1 to 1. These parameters generally used to control overfitting. *min_sample_leaf* defines the minimum samples required in a leaf. This parameter should have a low value for imbalanced class problem as minority class tree would have shorter length trees. It ranges from 0.1 to 0.5. *max_features* define number of features to consider while searching for the split. It ranges from 5 to 34.

4.3.4. XGBoost

XGBoost is a distributed boosting library designed for rapid and scalable ML model development. It utilizes ensemble learning to combine predictions from multiple weak models, creating a more powerful prediction. XGBoost places considerable emphasis on the importance of weights. Before being input into the DT for outcome prediction, each independent variable is assigned a weight. Variables that were incorrectly predicted by the tree are given greater weight before being used in the subsequent DT. These individual classifiers or predictors are then aggregated to generate a strong and precise model. Mathematically, write the model in the form:

$$\widehat{y_i} = \sum_{k=1}^{K} f_k(x_i), f_k \in F \tag{6}$$

Where K is the number of trees, f is the functional space of F, and F is the set of possible decision trees. Train-time complexity for XGBoost takes $O(t \times d \times x \times logn)$, and test-time complexity takes $O(td)$. t represents the number of trees, d denotes the height of the tree, x indicates the number of non-missing entries. Hyperparameters and tuning parameters for XGBoost are *eta* (0.05), *max_depth*(5), *subsample*(0.6), *min_child_weight* (8). *min_child_weight* defines minimum sum of weights required by a leaf node. It used to control over-fitting hence it should be tuned using cross-validation and it ranges from 1 to 13. eta is analogues to *learning_rate* meaning, it shrinks the feature weight to make boosting process more conservative and it ranges from 0 to 1 for hyperparameter tuning. *subsample* defines the fraction of observations to be randomly sampled for each tree. Subsampling will only occur in every iteration. It ranges from 0 to 1.

4.4. LSTM with attention layer

The looping feedback connections and feedforward connections in LSTM architecture help the model retain knowledge over time. In addition to learning from both long and short dependencies without a loose or excessive buildup of data. LSTM is also capable of remembering past events and making predictions about future events. In order to regulate the information flow in each cell of the architecture, LSTM employs a number of gates, including forget gate, input gate, and output gate [31,32].

The forget gate's output is designated as F_t, while its weights and bias parameters are W_F, U_F and b_F. Likewise, W_I, U_I and b_I are the input gate weight and bias, and I_t is the input gate's output. These weight and bias parameters are optimized during training. At time t, the input and hidden vectors are respectively X_t and h_t.

$$F_t = \sigma(W_F X_t + U_F h_{t-1} + b_F) \tag{7}$$

$$I_t = \sigma(W_I X_t + U_I h_{t-1} + b_{I)} \tag{8}$$

C_t stores the value that was determined by the output of an input and forget gate along with the current value of the input. These values are used to calculate the output and hidden states.

$$O_t = \sigma(W_O X_t + U_O h_{t-1} + b_O) \tag{9}$$

$$C_t = F_t \odot C_{t-1} + I_t \odot tanh(W_C X_t + U_C h_{t-1} + b_C) \tag{10}$$

$$h_t = O_t \odot tanh(C_t) \tag{11}$$

$$O_t = f(W_o h_t + b_o) \tag{12}$$

The intuition behind backpropagation is we compute the gradients of the final loss with respect to the weights of the network and during optimization, moving along this direction, and updating the weights, minimizing the loss.

$$\frac{dE}{dW_{xo}} = E_{delta} * \tanh(c_t) * sigmoid(z_0) * (1 - sigmoid(z_0)) * x_t \tag{13}$$

$$\frac{dE}{dW_{ho}} = E_{delta} * \tanh(c_t) * sigmoid(z_0) * (1 - sigmoid(z_0)) * h_{t-1} \tag{14}$$

$$\frac{dE}{db_o} = E_{delta} * \tanh(c_t) * sigmoid(z_0) * (1 - sigmoid(z_0)) \tag{15}$$

$$\frac{dE}{dW_{xf}} = E_{delta} * o * \left(1 - tanh^2(c_t)\right) * c_{t-1} * sigmoid\left(z_f\right) * \left(1 - sigmoid\left(z_f\right)\right) * x_t \tag{16}$$

$$\frac{dE}{dW_{hf}} = E_{delta} * o * \left(1 - tanh^2(c_t)\right) * c_{t-1} * sigmoid\left(z_f\right) * \left(1 - sigmoid\left(z_f\right)\right) * h_{t-1} \tag{17}$$

$$\frac{dE}{db_0} = E_{delta} * o * \left(1 - tanh^2(c_t)\right) * c_{t-1} * sigmoid\left(z_f\right) * \left(1 - sigmoid\left(z_f\right)\right) \tag{18}$$

$$\frac{dE}{dW_{xi}} = E_{delta} * o * \left(1 - tanh^2(c_t)\right) * g * sigmoid(z_i) * (1 - sigmoid(z_i)) * x_t \tag{19}$$

$$\frac{dE}{dW_{hi}} = E_{delta} * o * \left(1 - tanh^2(c_t)\right) * g * sigmoid(z_i) * (1 - sigmoid(z_i)) * h_{t-1} \tag{20}$$

$$\frac{dE}{db_i} = E_{delta} * o * \left(1 - tanh^2(c_t)\right) * g * sigmoid(z_i) * (1 - sigmoid(z_i)) \tag{21}$$

$$\frac{dE}{dW_{xg}} = E_{delta} * o * \left(1 - tanh^2(c_t)\right) * i * \left(1 - tanh^2\left(z_g\right)\right) * x_t \tag{22}$$

$$\frac{dE}{dW_{hg}} = E_{delta} * o * \left(1 - tanh^2(c_t)\right) * i * \left(1 - tanh^2\left(z_g\right)\right) * h_{t-1} \tag{23}$$

$$\frac{dE}{db_g} = E_{delta} * o * \left(1 - tanh^2(c_t)\right) * i * \left(1 - tanh^2\left(z_g\right)\right) \tag{24}$$

Where W_{xi}, W_{xg}, b_i, W_{hj}, W_g, b_g are the input gates, W_{xf}, W_{hf}, b_f are forget gates and W_{xo}, W_{ho}, b_o are output gates, E_{delta} is $\frac{dE}{dh_t}$. The idea of the attention mechanism is an imitation of the human ability to concentrate on different aspects of information when processing enormous amounts of information. The foundation of the attention mechanism was brought into existence due to the possible malfunction that can occur when models suffer from information overload caused by large datasets. When large amounts of data are taken as input, this mechanism works on differentiating the data based on the importance given i.e., attention score, and even traces out the credible data in order to improve the efficiency and accuracy of the model. The attention mechanism is also used for the generation process of the hidden state matrix H in the LSTM framework.

The attention mechanism sorts the variables of the CICIDS 2017 dataset through the LSTM model as shown in Fig. 22. For deriving the

Table 6
CICIDS-2017 dataset description.

File name	Traffic types	Number of records
Monday-WorkingHours.pcap_ISCX.csv	Benign	529,918
Tuesday-WorkingHours.pcap_ISCX.csv	Benign	432,074
	SSH-Patator	5,897
	FTP-Patator	7,938
Wednesday-WorkingHours.pcap_ISCX.csv	Benign	440,031
	DoS Hulk	231,073
	DoS GoldenEye	10,293
	DoS Slowloris	5,796
	DoS Slowhttptest	5,499
	Heartbleed	11
Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv	Benign	168,186
	Web Attack-Brute Force	1,507
	Web Attack-SQL Injection	21
	Web Attack-XSS	652
Thursday-WorkingHours-Afternoon-Infiltration.pcap_ISCX.csv	Benign	288,566
	Infiltration	36
Friday-WorkingHours-Morning.pcap_ISCX.csv	Benign	189,067
	Bot	1,966
Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv	Benign	127,537
	PortScan	158,930
Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv	Benign	97,718
	DDoS	128,027
Total samples		**2,830,743**

important variables in LSTM, context vector is used for selecting important variables within each network flow. To do this, calculate attention weights for each feature of the network flow data. These weights decide the importance of each feature in the context of attack detection. The below-mentioned formula gives us the context vector S_i, which is a creation of a set of the weighted sum of the annotations, where each annotation is weight-based on its significance to deriving important variables, and where T represents the number of variables of the data.

$$C_i = \sum_{i=1}^{T} a_{(t,T)}.h_t \tag{25}$$

The weights $a_{(t,T)}$ are calculated using SoftMax function as given below:

$$e_{ij} = \alpha(h_j) \tag{26}$$

$$a_{(t,T)} = softmax(e_{ij}) \tag{27}$$

Where e_{ij} represents the output score which will define the importance of the variables depending on the function α that attempts to capture the arrangement between i and j. $a_{(t,T)}$ represents the attention weights assigned to different elements in the input variables while calculating the context vector. The three hidden layers of the implemented LSTM model include 64, 64 and 128 neurons respectively with an attention layer. The activation function for the hidden layers is Rectified Linear Unit (ReLU), and the activation function for the output layer is SoftMax. The categorical cross-entropy function and the binary cross-entropy function were used to fit model two's loss function for categorical data, and binary data respectively. LSTM method likely stems from its ability to handle long-term dependencies to mitigate the vanishing gradient problem, suitable for sequential data processing, simplicity, interpretability, and empirical success in similar tasks. These factors collectively make LSTM a practical and reliable choice for modelling intrusion detection scenarios.

Hyperparameters and tuning parameters for LSTM are *Optimizer*, *loss function*, *batch_size*(32), *epochs*(10), and *dropout*(0.2). *Adam optimizer* with *learning_rate* of 0.001 is chosen as it better handle the complex training dynamics of RNNs. *Categorical_crossentropy* is chosen as the loss function for multi-class classification and *binary_crossentropy* for binary-class classification. *batch_size* defines the number of samples to work before the internal parameters are updated. For hyperparameter tuning different *batch_size* values were experimented like 64, 128 and 256. *epochs* defines the number of complete iterations of dataset should run. This has to be carefully validated with validation accuracy and training accuracy. To find the right trade-off between over-fitting and under-fitting. *dropout* helps in tackling over-fitting problems by bypassing randomly selected neurons, thereby reducing the sensitivity to specific weight of a neuron.

4.4.1. Classification of attacks using ensemble learners and LSTM

After preprocessing of CICIDS2017 dataset, handle the class imbalance for multi-class classification and binary-class classification. Then, initialize all the models (RF, AdaBoost, XGBoost, GradientBoost and LSTM) with initial parameters. Multiple parameter combinations are selected and train the models to get the best-optimized result. The best parameters are picked using Grid Search, and K-fold cross-validation (cv). The Grid Search approach tries different combinations of hyper-parameters for training the model, and selects the best combination based on a scoring metric. It creates a grid of all possible values for each hyperparameter and evaluates the model for each combination using cross-validation. The combination that gives the highest score is chosen as the best one. K-fold cv is a method that splits the data into K equal parts (folds). It uses one-fold as the test set, and the rest as the training set. This process is repeated K times, each time using a different fold as

Table 7
Performance in multi-class classification.

Classifier	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
	Before applying UpSampling				After applying UpSampling			
RF	1.000	0.892	0.871	0.888	0.978	0.887	0.904	0.903
AdaBoost	0.860	0.784	0.801	0.827	0.727	0.593	0.615	0.635
XGBoost	0.982	0.927	0.882	0.896	0.997	0.987	0.995	0.992
GradientBoost	0.940	0.951	0.927	0.952	0.959	0.813	0.778	0.805
LSTM	1.000	0.971	0.952	0.959	0.995	0.983	0.989	0.990

The color only represents the difference between the data before upsampling and after upsampling.

Table 8
Performance of XGBoost classifier.

Attacks	Precision	Recall	F1	Support	Precision	Recall	F1	Support
Before applying SMOTE UpSampling					After applying SMOTE UpSampling			
Benign	1.000	1.000	1.000	152655	1.000	1.000	1.000	152655
Bot	0.971	0.982	0.987	581	0.921	0.856	0.843	581
Brute Force	0.946	0.973	0.961	463	0.858	0.829	0.850	463
DDoS	1.000	1.000	1.000	77096	1.000	1.000	1.000	77096
FTP-Patator	1.000	1.000	1.000	2417	1.000	0.962	0.951	2417
Infiltration	1.000	1.000	0.987	11	0.832	0.793	0.824	11
PortScan	1.000	1.000	1.000	47434	0.979	0.914	0.902	47434
SSH-Patator	1.000	1.000	1.000	1780	1.000	0.899	0.920	1780
SQL Injection	0.982	1.000	1.000	6	0.797	0.763	0.776	6
XSS	0.977	0.991	0.993	188	0.893	0.838	0.862	188
Weighted Average	**0.987**	**0.995**	**0.992**	**282631**	**0.928**	**0.883**	**0.895**	**282631**

The color only represents the difference between the data before upsampling and after upsampling.

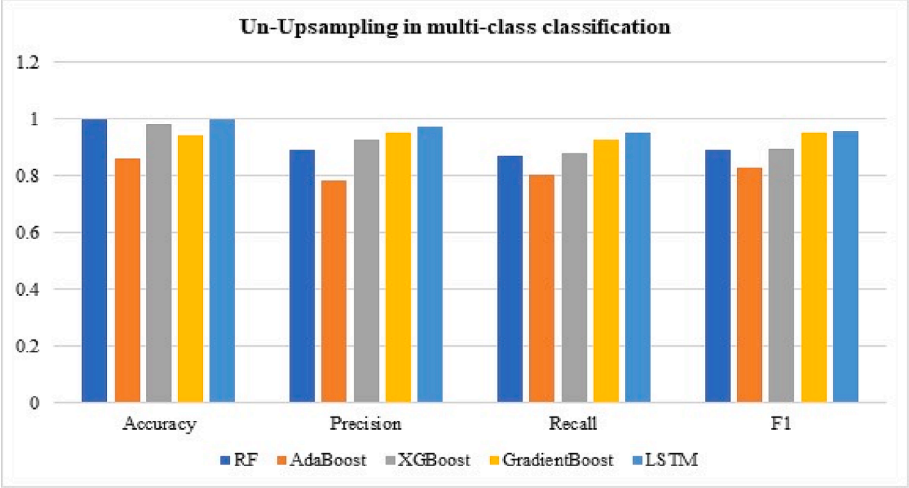

Fig. 24. Comparison of models with un-upsampled data for multi-class.

the test set as shown in Fig. 23. The average score of the K iterations is used as the final score of the model. K-fold cv is useful for reducing the variance of the model that avoids overfitting or underfitting. It also allows the model to use all the data for both training and testing, unlike a simple train-test split. The best parameters are used for training the model and followed by evaluation with the test set.

4.5. Data balancing using sampling

The preprocessed training data does not consist of balanced samples with respect to attacks categories. Hence, this study initiated the SMOTE method to UpSample the training data. Consequently, upsampled data fed into the ensemble classifiers, and LSTM with attention mechanism to detect the attacks with multi-class categories. Similarly, downsampling is applied to the training dataset that results the subset. This downsampled dataset fed into the classifiers that classified the data with binary class categories either attack or normal.

5. Dataset description

The real-world network traffic CICIDS-2017 dataset from the Information Security Center for Excellence (ISCX) Consortium is used in this research work. Eight traffic monitoring sessions have been considered as a CSV file, constitute MachineLearningCSV. This file contains both abnormal traffic known as “Attacks” traffic, and typical traffic is “Benign” traffic. In addition to regular and benign traffic, this dataset

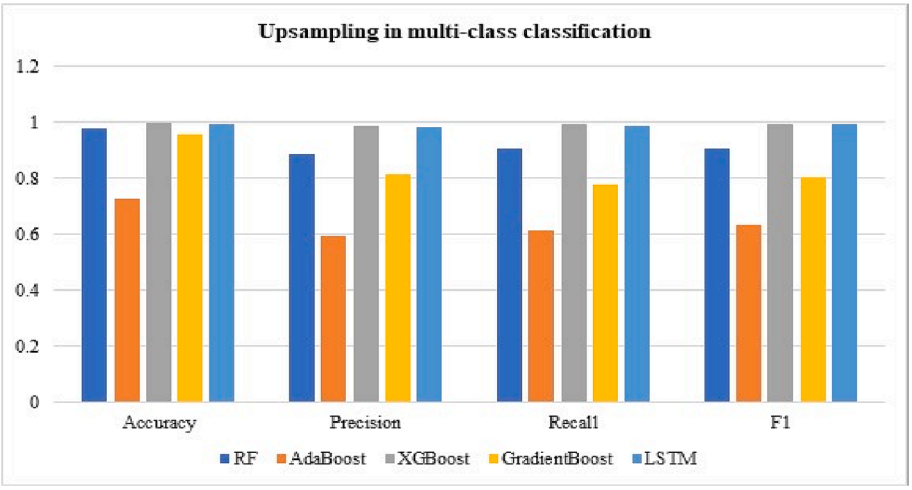

Fig. 25. Comparison of models with UpSampled data for multi-class.

Fig. 26. Time to construct and evaluate models for multi-class classification.

contains 14 different forms of attack traffic as shown in Table 6. The DDoS, DoS Hulk, Denial of Effect (DoE) GoldenEye, and Heartbleed attack types must be detected using the *Bwd Packet Length Std* feature. Regular traffic must be recognized using the *Min Bwd Package Length* feature, and *Fwd Average Package Length* variables [33].

6. Experimental tools

This study intends to develop an IDS classifier with higher accuracy, higher dependability, fewer false alarms, and fewer false negatives. The initiated method implies the RFHC technique to select only those variables that affect the results. The initiated system is implemented using

Table 9
Performance in binary class classification.

Classifier	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
Before applying SMOTE UpSampling					**After applying SMOTE UpSampling**			
RF	0.981	0.990	0.989	0.986	0.994	0.994	0.992	0.996
AdaBoost	0.960	0.970	0.970	0.968	0.962	0.969	0.971	0.979
XGBoost	0.885	0.887	0.892	0.917	0.987	0.991	0.989	0.985
GradientBoost	0.992	0.993	0.994	0.991	0.990	0.994	0.996	0.996
LSTM	0.989	0.986	0.970	0.990	0.992	0.971	0.985	0.989

The color only represents the difference between the data before upsampling and after upsampling.

Table 10
Performance of RF classifier.

Data	Precision	Recall	F1	Precision	Recall	F1
Before applying UpSampling				After applying UpSampling		
Normal	0.990	0.986	0.989	0.996	0.994	0.996
Traffic	0.990	0.991	0.983	0.990	0.900	0.992
Weighted Average	**0.990**	**0.989**	**0.986**	**0.994**	**0.992**	**0.994**

The color only represents the difference between the data before upsampling and after upsampling.

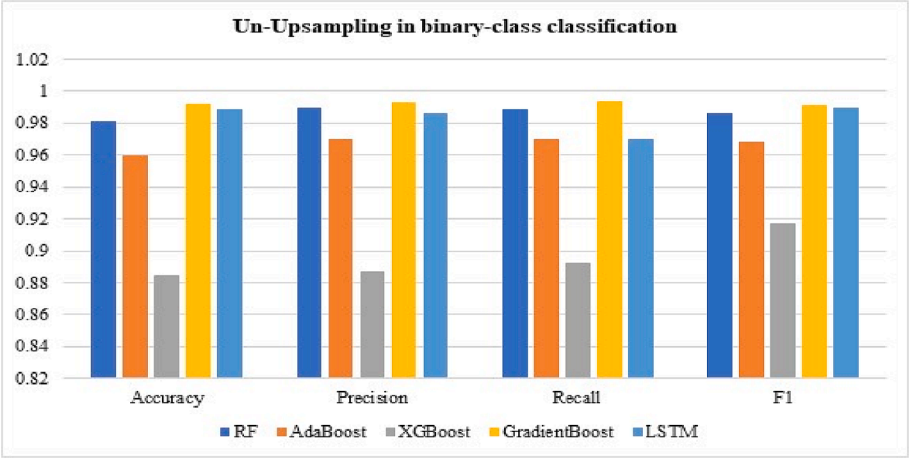

Fig. 27. Comparison of models with Un-UpSampled data for class binary.

the CICIDS-2017 dataset of Upsampled, and Un-Upsampled data to test multi-class and binary-class classification. The experimentation is carried out on a laptop with a 10th generation Core i5 processor and 12 GB of RAM. This model makes use of a number of Sklearn packages from Python 3.11, including RFHC selection and Classifier from the Ensemble package. Refs. [34–37] highlights some applications of fake news detection and neural network based simulations.

6.1. Metrics for evaluating models

Intrusion detection performance can be evaluated through well-known metrics like False Positive Rate (FPR), True Positive Rate (TPR,) F1-measure and accuracy [38,39]. True Positive (TP) defines the model correctly predicts positive class input data as positive class. False Positive (FP) represents the negative data is predicted wrongly as

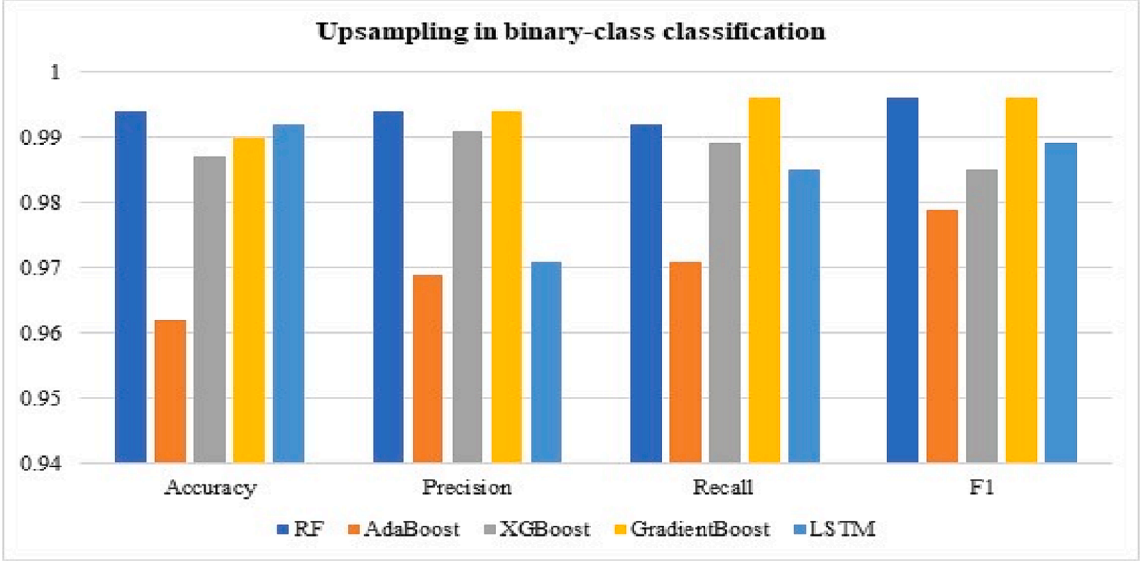

Fig. 28. Comparison of models with UpSampled data for binary class.

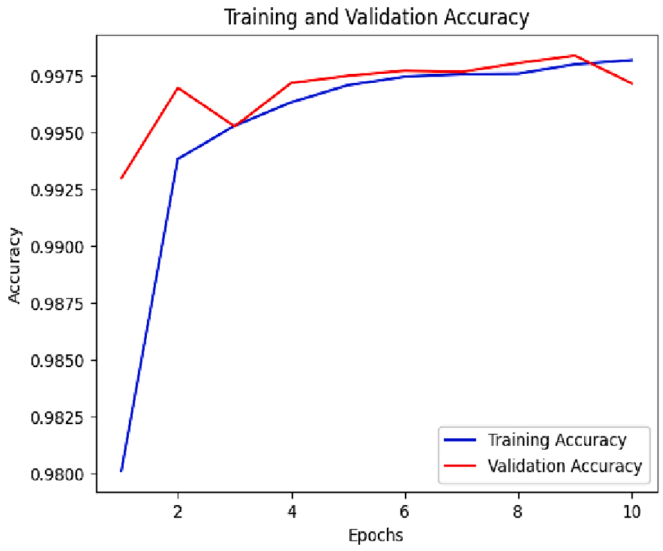

Fig. 29. Performance of upsampled data for binary class.

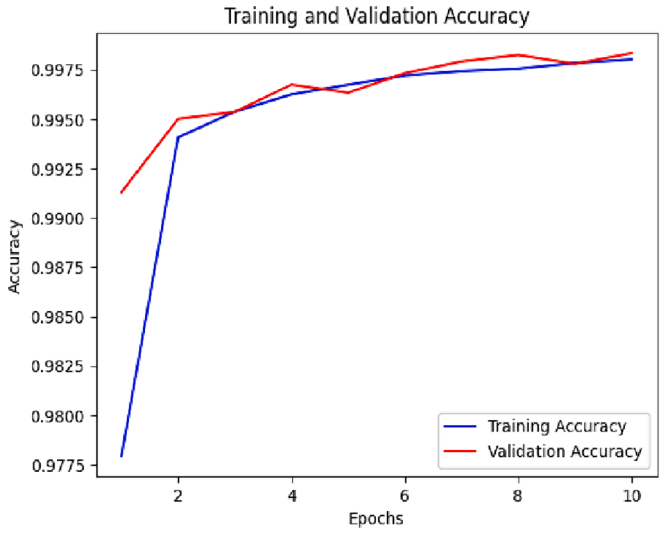

Fig. 30. Performance of un-upsampled data for binary class.

positive output. True Negative (TN) represents the negative class is correctly predicted as negative output. False Negative (FN) represents the positive class is predicted as negative class correctly. Precision and recall are used together to give a complete representation of the model's ability to make correct predictions.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{28}$$

$$\text{F1 score} = \frac{(\text{Precision}*\text{Recall})*2}{\text{Precision} + \text{Recall}} \tag{29}$$

In the field of IDS, where defending cyber threats is paramount, the choice of evaluation metrics is critical. While accuracy measures overall correctness, it weakens the presence of class imbalances common in IDS datasets. However the F1 score plays a pivotal role in the metrics by balancing precision and recall. Unlike accuracy, the F1 score offers nuanced insight into the system's efficacy in identifying attacks while capturing their full spectrum. By considering false positives and false negatives, it comprehensively assesses IDS performance, ensuring accurate threat detection while minimizing the risk of overlooking genuine attacks. Thus, relying on accuracy and F1 score underscores the need for a balanced approach, essential for robust cybersecurity defenses.

7. Results and discussion

7.1. Multi-class classification

This study contributes to benchmarking five classification algorithms. It focuses on comparing them with UpSampled and Un-UpSampled data. Table 7 shows the classifiers with UpSampled data outperform the classifiers with Un-UpSampled data. XGBoost shows a significant and efficient growth in all the metrics after applying SMOTE. Table 7 displays the outcomes obtained with 34 variables prior to the application of UpSampling, and 34 variables after UpSampling has been applied. UpSampling is employed to create a balanced dataset.

Comparing Infiltration, SQL Injection and XSS class from Table 8, it is inferred that UpSampling technique has helped XGBoost with unbalanced dataset problem, and increasing the F1-score, Recall and Precision. AdaBoost and GradientBoost models perform poorly with UpSampling data as shown in Table 8. Focusing on minority instances, AdaBoost already takes care of it by giving them larger weights. Additionally, including the minority class in UpSampling, can place an undue focus on those cases, which could result in an uneven boost and even lower overall performance. In GradientBoost, the underlying patterns in the data cannot be detected by weak learners therefore increasing their prominence through UpSampling may not result in meaningful improvements. The complexity and depth of the trees can influence their capacity to successfully model the data. LSTM outperforms other models in Un-UpSampled multi-class classification as shown in Fig. 24. LSTM networks are inherently robust to imbalanced datasets and can effectively assign appropriate weights to different classes during training, mitigating the need for UpSampling or downsampling techniques. Furthermore, their hierarchical representation learning enables them to capture both low-level and high-level features in the input sequences, allowing for the effective capture of complex patterns and relationships in network traffic data compared to traditional ML algorithms that typically rely on shallow representations of data.

Fig. 25 illustrates the performance of classifiers using SMOTE based upsampled dataset. XGBoost emerges as the best algorithm in multi-class regarding accuracy, precision and recall. The SMOTE based UpSampling process generated the subset of samples with balanced category of each class in the dataset. XGBoost constructs decision trees sequentially, honing in on errors from preceding trees to refine subsequent ones. This iterative process of refinement often yields superior performance, particularly in scenarios with imbalanced datasets. By integrating SMOTE, XGBoost adeptly addresses class imbalance concerns by synthesizing new minority class samples, thereby enhancing its ability to learn from both minority and majority class instances. Furthermore, XGBoost's ensemble of decision trees, coupled with sophisticated regularization techniques and novel split-finding algorithms, contributes to its superior model performance. Regularization techniques such as shrinkage and column subsampling mitigate overfitting while advanced optimization techniques facilitate the efficient search for optimal tree structures, enhancing the model's generalization capabilities. Additionally, XGBoost's proficiency in capturing complex, non-linear relationships between features and target variables are attributed to its capacity to fit decision trees with multiple splits.

The evaluation of model performance, concerning the time required for both model construction and testing, is visually presented in Fig. 26. The LSTM with attention mechanism consumes more time than other algorithms due to more parameters involving in the classification and due to the number of epochs. In ensemble learning classifiers, XGBoost takes more training time than other algorithms. However, XGBoost classifier provides better performance than other classifiers.

Fig. 31. Time to build and test models for binary-class classification.

Table 11
Comparison of CICIDS-related studies and initiated framework.

Feature Selection method	Classification method	Selected Variables Count	Accuracy	Precision	Recall	F1-Measure	Dataset
Random Forest Regressor [12]	KNN	54	–	0.96	0.96	0.96	CICIDS2017
	RF			0.98	0.97	0.97	
	ID3			0.98	0.98	0.98	
	Adaboost			0.77	0.84	0.77	
	MLP			0.77	0.83	0.76	
	Naïve Bayes			0.88	0.04	0.04	
	QDA			0.97	0.88	0.92	
Fisher Scoring [14]	KNN	30	0.9997	0.9985	0.9968	0.9997	NSL-KDD, AWID, and CIC-IDS2017
DDR [15]	XGBoost	36	98.93	–	–	–	CIC-IDS2017, UNSW-NB15, and NSL-KDD
CFS_BA [16]	Voting contains (C4.5, RF, ForestPA)	13	99.89	–	99.9	–	CICIDS2017(Wed)
HPS-KODE [17]	Voting contains (K-means, One- class SVM, DBSCAN, and Maximization-Expectation, (KODE))	8	99.9	–	96.64	–	BoT-IoT
Deep neural network [22]	XG Boost algorithm	41	99	–	–	–	NSL-KDD, CIDDS-001, and
		38	96	–	–	–	CICIDS2017
		78	92	–	–	–	
Proposed RFHC	**XGBoost (multi-class)**	**34**	**99.7**	**98.7**	**99.5**	**99.2**	**CICIDS2017**
	RF (binary class)	**34**	**99.4**	**99.4**	**99.2**	**99.6**	

7.2. Binary classification

This study also considers binary classification to categorize the traffic data as either normal or attack. In multi-classification, different attacks categorized from the abnormal traffic.

However, binary classification is applied to classify the traffic based on its behavior either attack or normal. It is not focusing on the various categories of attacks. Models trained on UpSampled data outperform the ones with Un-UpSampled data as shown in Table 9. Table 10 depicts the results with 34 variables (before applying UpSampling), and after applying UpSampling.

Comparing Table 10 showed the UpSampling technique makes RF model to better classify on Traffic data.

GradientBoost outperforms other models in Un-UpSampled binary-class classification as shown in Fig. 27. GradientBoost is consistent with accuracy, precision, and recall. GradientBoost corrects the errors of the previously trained models. GradientBoost achieves higher precision by effectively classifying positive instances and minimizing false positives and achieves higher recall by correctly classifying most of the positive instances, thus minimizing false negatives.

RF emerges as the better algorithm in binary class regarding accuracy, precision and recall as shown in Fig. 28. The boosting ensemble learning models also perform slightly lower than the bagging ensemble learner. The boosting ensemble learners used more parameters and decision trees to classify the dataset.

Figs. 29, 30 shows the training epoch curve of the LSTM model with Upsampled and Un-UpSampled data respectively. In Fig. 29, the line graph represents the training and validation accuracy of a LSTM over 10 epochs. The blue line, representing the training accuracy, starts at around 0.98 and increases to around 0.9975 over 10 epochs.

On 2nd epoch, the training accuracy reaches at 0.9932. The red line, representing the validation accuracy, starts at around 0.9925 and increases to around 0.995 over the same period. At epoch 9, training and validation accuracy are same (0.997) This suggests that the model is learning effectively from the training data and generalizing well to the validation data. In Fig. 30, validation accuracy is 0.9983, whereas training accuracy is 0.9973, hence validation accuracy is higher than training accuracy. At epoch 9, training and validation accuracies are same (0.9975). This indicates the problem of data imbalance. The training accuracy reflects the true performance of the model, where

validation accuracy is boosted by performing on minority classes.

The performance evaluation of models in terms of time to build and test the model is presented for binary-class classification as shown in Fig. 31. The LSTM with attention mechanism consumes more time than other algorithms due to more parameters involving in the classification and due to the number of epochs.

Table 11 provides a detailed comparison between our initiated framework and prior studies. Given the data imbalance in the CICIDS2017 dataset, we employed the F1-Measure for a thorough assessment and comparison. As depicted in Table 11, our methodology surpasses previous research in both accuracy and F1-Measure performance.

8. Conclusion and future scope

Existing IDSs are still ineffective by some standards despite prior attempts to improve their effectiveness by utilizing different ML techniques. This study initiated a unique IDS approach using hybrid techniques based on the intended FS for coping with unbalanced and high-dimensional traffic. To obtain the best feature subset, the RFHC strategy is applied that selected.

34 variables. The suggested model's final experimental findings use the CICIDS2017 dataset which demonstrated an accuracy of 99.40 %, a recall of 99.20 %, a precision of 99.41 % and an F1-measure of 99.62 % for binary-class classification. The model demonstrated an accuracy of 99.71 %, a recall of 99.50 %, a precision of 98.70 % and an F1-measure of 99.20 % for multi-class classification.

Despite the fact that deep learning models are more accurate, our future plans include creating a hybrid deep learning model for IoT intrusion detection that is more accurate at predicting attacks on real-time data. Deep learning models, particularly Convolutional Neural Networks (CNNs) and RNNs such as LSTMs, have demonstrated remarkable performance in various domains, including intrusion detection. These models are capable of learning intricate patterns and representations directly from raw data, making them well-suited for capturing complex relationships in IoT network traffic. However, deep learning models may not always generalize well to new, unseen data, especially in scenarios with limited labeled training data or in the presence of imbalanced classes. By integrating traditional ML techniques such as ensemble learning, feature selection, or dimensionality reduction with deep learning architectures, it's possible to enhance the model's robustness and generalization ability.

More sophisticated feature selection algorithms are required as high-dimensional datasets can be difficult to work with and can result in overfitting or increased processing complexity.

CRediT authorship contribution statement

P. Kuppusamy: Methodology, Coding, Conceptualization, Analysis of data. **Dev Kapadia:** Methodology, Coding, Conceptualization, Analysis of data. **Edaboina Godha Manvitha:** Methodology, Coding, Conceptualization, Analysis of data. **Sami Dhahbi:** Methodology, Coding, Conceptualization, Analysis of data. **C. Iwendi:** Methodology, Coding, Conceptualization, Analysis of data. **M. Ijaz Khan:** Methodology, Coding, Conceptualization, Analysis of data. **Sachi Nandan Mohanty:** Formal analysis, Writing–review & editing. **Nidhal Ben Khedher:** Formal analysis, Writing–review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors extend their appreciation to the Deanship of Scientific Research at King Khalid University for funding this work through large group Research Project under grant number RGP2/337/44.

References

[1] Test: antivirus & security software & antimalware reviews. AV. (n.d.). Retrieved April 28, 2023.
[2] We protect data. Varonis. (n.d.). Retrieved April 28, 2023.
[3] Data breaches break record in 2021. CNET. (n.d.). Retrieved April 28, 2023.
[4] Modi C, Patel D, Borisaniya B, Patel H, Patel A, Rajarajan M. A survey of intrusion detection techniques in cloud. J Netw Comput Appl 2013;36(1):42–57.
[5] Khraisat A, Gondal I, Vamplew P, Kamruzzaman J. Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity 2019;2(1):1–22.
[6] Chandrashekar G, Sahin F. A survey on feature selection methods. Comput Electr Eng 2014;40(1):16–28.
[7] Eesa AS, Orman Z, Mohsin A, Brifcani A. A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems. Expert Syst. Appl. 2014;(November):1–10.
[8] Zhang J, Li H, Gao Q, Wang H, Luo Y. Detecting anomalies from big network traffic data using an adaptive detection approach. Inf Sci (NY) 2015;318(August):91–110.
[9] Jyothsna V, Prasad VR. FCAAIS: Anomaly based network intrusion detection through feature correlation analysis and association impact scale. ICT Express 2016;2(3):103–16.
[10] Satoh A, Nakamura Y, Ikenaga T. A flow-based detection method for stealthy dictionary attacks against secure shell. J Inf Secur Appl 2015;21:31–41.
[11] Kurniabudi, Stiawan D, Darmawijoyo, Idris MYB, Bamhdi AM, Budiarto R. CICIDS-2017 dataset feature analysis with information gain for anomaly detection. IEEE Access 2020;8:132911–21.
[12] Sharafaldin I, Lashkari AH, Ghorbani AA. Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp 2018;1:108–16.
[13] Vijayan R, Devaraj D, Kannapiran B. Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection. Comput Secur 2018;77:304–14.
[14] Aksu D, Üstebay S, Aydin MA, Atmaca T. Intrusion detection with comparative analysis of supervised learning techniques and fisher score feature selection algorithm. In: Computer and information sciences: 32nd international symposium, ISCIS 2018, held at the 24th IFIP world computer congress, WCC 2018, Poznan, Poland, September 20–21, 2018, Proceedings 32. Springer International Publishing; 2018. p. 141–9.
[15] Bansal A. DDR scheme and LSTM RNN algorithm for building an efficient IDS. PhD thesis; 2018.
[16] Zhou Y, Cheng G, Jiang S, Dai M. Building an efficient intrusion detection system based on feature selection and ensemble classifier. Comput Netw 2020;174:107247.
[17] Jaw E, Wang X. Feature selection and ensemble-based intrusion detection system: an efficient and comprehensive approach. Symmetry 2021;13:1764.
[18] Abbas A, Khan MA, Latif S, Ajaz M, Shah AA, Ahmad J. A new ensemble-based intrusion detection system for internet of things. Arab J Sci Eng 2022;47:1805–19.
[19] Ge M, Fu X, Syed N, Baig Z, Teo G, Robles-Kelly A. Deep learning-based intrusion detection for IoT networks. In: 2019 IEEE 24th Pacific rim international symposium on dependable computing (PRDC); Dec. 2019. p. 256–25609. doi: 10.1109/PRDC47002.2019.00056.
[20] Sindian S. An enhanced deep autoencoder-based approach for DDoS attack detection. WSEAS Trans Syst Control 2020;15:716–24. https://doi.org/10.37394/23203.2020.15.72.
[21] Ahmad S, Arif F, Zabeehullah Z, Iltaf N. Novel approach using deep learning for intrusion detection and classification of the network traffic. In: 2020 IEEE international conference on computational intelligence and virtual environments for measurement systems and applications (CIVEMSA); Jun. 2020. p. 1–6. doi: 10.1109/CIVEMSA48639.2020.9132744.
[22] Gupta N, Jindal V, Bedi P. CSE-IDS: using cost-sensitive deep learning and ensemble algorithms to handle class imbalance in network-based intrusion detection systems. Comput Secur 2022;112:102499.
[23] Fernández A, Garcia S, Herrera F, Chawla NV. SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. J Artif Intell Res 2018;61:863–905.
[24] Sekulíc A, Kilibarda M, Heuvelink GBM, Nikolíc M, Bajat B. Random forest spatial interpolation. Remote Sens 2020;12:1687.
[25] Xu X, Liu W, Yu L. Trajectory prediction for heterogeneous traffic-agents using knowledge correction data-driven model. Inf Sci 2022;608:375–91.
[26] Xia W, Pu L, Zou X, Shilane P, Li S, Zhang H, et al. The design of fast and lightweight resemblance detection for efficient post-deduplication delta compression. ACM Trans Storage 2023;19(3):1–30.
[27] Li X, Sun Y. Application of RBF neural network optimal segmentation algorithm in credit rating. Neural Comput Appl 2021;33(14):8227–35.
[28] Jiang H, Wang M, Zhao P, Xiao Z, Dustdar S. A utility-aware general framework with quantifiable privacy preservation for destination prediction in LBSs. IEEE/ACM Trans Netw 2021;29(5):2228–41.
[29] Wu Z, Liu G, Wu J, Tan Y. Are neighbors alike? A semi-supervised probabilistic collaborative learning model for online review spammers detection. Inf Syst Res 2023. https://doi.org/10.1287/isre.2022.0047.

[30] Xu P, Shin I. Preparation and performance analysis of thin-film artificial intelligence transistors based on integration of storage and computing. IEEE Access 2024;12:30593–603. https://doi.org/10.1109/ACCESS.2024.3369171.
[31] Dhillon H, Haque A. Towards network traffic monitoring using deep transfer learning. In: 2020 IEEE 19th international conference on trust, security and privacy in computing and communications (TrustCom); 2020. p. 1089–96. https://doi.org/10.1109/TrustCom50675.2020.00144.
[32] Van Houdt G, Mosquera C, Nápoles G. A review on the long short-term memory model. Artif Intell Rev 2020;53(8):5929–55. https://doi.org/10.1007/s10462-020-09838-1.
[33] Sharafaldin I, Lashkari A, Ghorbani A. A new intrusion detection dataset and intrusion traffic characterization. In: 4th International conference on information systems security and privacy (ICISSP); 2018. p. 108–16.
[34] Ding Y, Zhang W, Zhou X, Liao Q, Luo Q, Ni LM. FraudTrip: taxi fraudulent trip detection from corresponding trajectories. IEEE Internet Things J 2021;8(16):12505–17.
[35] Liao Q, Chai H, Han H, Zhang X, Wang X, Xia W, et al. An integrated multi-task model for fake news detection. IEEE Trans Knowl Data Eng 2022;34(11):5154–65.
[36] Xu Y, Wang E, Yang Y, Chang Y. A unified collaborative representation Learning for neural-network based recommender systems. IEEE Trans Knowl Data Eng 2022;34(11):5126–39.
[37] Zheng W, Lu S, Cai Z, Wang R, Wang L, Yin L. PAL-BERT: an improved question answering model. Comput Model Eng Sci 2023. https://doi.org/10.32604/cmes.2023.046692.
[38] Guillet F, Hamilton HJ, editors. Quality measures in data mining, Vol. 43. Springer; 2007.
[39] Abdulhammed R, Musafer H, Alessa A, Faezipour M, Abuzneid A. Variables dimensionality reduction approaches for machine learning based network intrusion detection. Electronics 2019;8(3):322.
[40] Akazue MI, Debekeme IA, Edje AE, Asuai C, Osame UJ. UNMASKING FRAUDSTERS: ensemble features selection to enhance random forest fraud detection. J Comput Theories Appl 2023;1(2):201–11.

Dr. M. Ijaz Khan: Received MS and PhD from Quaid-I-Azam University, Islamabad, Pakistan in the year 2016 and 2019 in Applied Mathematics for work in the field of CFD analysis, Flow Behavior and Numerical techniques (AI). Currently working as Assistant Professor in the Department of Mechanical Engineering, Prince Mohammad Bin Fahd University, P. O. Box, 1664, Al-Khobar 31952, Kingdom of Saudi Arabia. Contributed more than 50 research-level papers to many International journals. Research interests include Sensor Networks, Machine Learning, and Cloud computing, Flow Behavior, CFD analysis.