Optimizing Intrusion Detection Systems (IDs) Through Significant Feature Selection By ML Techniques

Name 1[a,1], **Name 2**[a,2], **Name 3**[a,3] **Name 4**[a,4]

[a]Department of Computer Systems Engineering, The Islamia University of Bahawalpur.

1. Article Info

Keywords: NIDS, Intrusion Detection Systems, UNSW-NB-15 dataset, Feature Selection(ANOVA-F-Static), Cyber-Security in Machine Learning, Guassian Naive Bayes, Logistic Regression, Gradient Boosting Algorithm

2. Abstract

Network Intrusion Detection System plays a vital role in the detection of malicious activity to catch a person who breaks a computer system before damage is done in the network. For almost every organization, IDS provides security built using many approaches with the aim of protection of network and resources from stealing, exposing, altering or destroying data, applications or other useful assets by using unauthorized access. Earlier, Most of the research has introduced difficulties related to over-fitting, different methods for feature selection, high-dimensionality of features with such a limited number of training samples and results on applying different machine learning models and different dataset has different features. In this study, ANOVA-B-LG, ANOVA-B-GN, ANOVA-B-GB, ANOVA-M-LG, ANOVA-M-GN and ANOVA-M-GB techniques are used to approach the problem of features selection with their specific model for Binary and Multi Classification respectively on a UNSW-NB-15 Benchmark dataset. In this paper, proposed an optimizing features selection and intrusion classification for IDS. This is a recently developed technique to model the data for the aim of optimizing classification with accurate measures. Comparing with other feature selection techniques, they offer approaches, which consider features interaction and dependencies,potentially leading to better feature subsets. The experiments outlined that proposed methods have better performance than most of the well-known technique used for intrusion detection. In future work, federating learning based models and proposed models will be discussed.

3. Introduction

As technologies like the Internet, Internet-of-Things (IoT), and communication systems progress rapidly, hackers are also advancing quickly, expanding their capabilities to compromise computer network security [1]. Consequently, Intrusion Detection Systems (IDSs) play a vital role as essential components within computer networks. An IDS is designed as either hardware or software to vigilantly monitor an organization's computer network, detecting and addressing potential threats and attacks[2] [3].

A distributive framework of Federated Learning proposed in 2016 by the research Team of Google, which can reach out of data growth and privacy protection at the same time, keeping the client data private [4]. To overcome the challenges faced in Network Intrusion Detection System, federated learning has promising approaches for developing effective and efficient (IDSs), But it will be the part of our future work [5].

In Intrusion Detection Systems (IDSs), they are typically categorized into three main types: signature-based, anomaly-based, and hybrid-based [6]. Signature-based IDSs rely on a repository of known attack patterns to identify intrusions [7]. Anomaly-based IDSs, on the other hand, focus on monitoring network behavior. They continuously analyze network activity, looking for deviations from normal behavior. Hybrid-based IDSs integrate elements of both signature-based and anomaly-based approaches [8]. There are multiple categories of attacks limited the network in integrity [9], security and confidentiality for any computer network[10]. Also, if know the steps of most harmfull attack is considered to be a Dental of Service attack (DOS) among others [11].

Detailed examinations of existing research are presented in Section 4. In Section 5, pre-processing is provided along with feature selection and target variables for binary and multi-classification. In Section 6, proposed methods used for model training and testing are defined. However, Section 8 discussed the experimental analysis of proposed methods also including the results of hyper-parameter tuning.In Section, 8-G, presenting the classification results for both binary and multi-classification.In final section 10, concluded the

Corresponding Authors:
Email Addresses: (Dr. Rehan Ali Shah),
This author contributed and participated equally in this work.

final results after applying classification techniques

4. Literature Review

The book titled ”Big Data Analytics in Cyber Security” [12] underscores the importance of leveraging data analytics in the realm of cyber security. It suggests that by analyzing large volumes of network data, Big Data offers new avenues for enhancing cyber security, such as developing models and configuring network infrastructure for organizations. Rajwat's research illustrates how the effectiveness of machine learning algorithms in the Internet of Things (IoT) [13] sector can bolster network security by ensuring the reliability, security, availability, and survivability of security assets. Deep learning is recommended for optimizing network construction in smart cities and IoT devices. The integration of machine learning and IoT reflects their increasing integration into modern life, with approximately 25 billion IoT-connected devices generating substantial amounts of data requiring analysis to enhance performance. Machine learning techniques like decision trees, neural networks, and Bayesian networks empower devices to recognize patterns in datasets and make decisions based on their analysis.

Additionally, a paper referenced as [14] discusses a novel intrusion detection system capable of identifying five distinct types of threats in a network, including Exploit, DOS, Probe, Generic, and Normal activities. Utilizing the UNSW-NB15 dataset, this system employs an integrated classification-based model to detect malicious activities, demonstrating a significantly higher accuracy rate of 83.8% on real-time data generated at NIT Patna CSE lab. Likewise, [15] asserts the efficacy of machine learning-based intrusion detection systems in identifying network attacks.

However, these systems face challenges when dealing with high-dimensional data and imbalanced datasets, which often lead to reduced accuracy and increased false positive rates. To address this, [15] presents a filter-based feature reduction method utilizing the XGBoost algorithm, alongside an analysis of the UNSW-NB15 intrusion detection dataset. Various machine learning approaches such as Support Vector Machine (SVM), k-Nearest-Neighbors (KNN), Logistic Regression (LR), Neural Network (NN), and Decision Tree (DT) were implemented using the reduced feature space. The findings indicate that the XGBoost-based feature selection approach improves the test accuracy of the binary classification scheme from 88.13% to 90.85% for techniques like DT.

In [16], the authors highlight the significance of Network Intrusion Detection Systems (NIDS) in addressing internet security issues within the IoT environment. They argue that the UNSW-NB15 dataset is more suitable for evaluating NIDS. Experimental results demonstrate that the SVM method outperforms other methods, achieving an accuracy of 85.99% for binary classification and 75.77% for multi-classification.

Classification algorithms are crucial for assisting IDS in identifying various types of attacks, as demonstrated by a comparative analysis of the UNSW-NB15 dataset conducted in [17]. The study suggests that the Random Forest classification model is particularly reliable, achieving an accuracy of 97.49%.

According to this study, the Random Forest model demonstrates higher accuracy compared to Decision Tree and Naïve Bayes models. [18] presents an experiment emphasizing the importance of network intrusion detection systems in identifying cyber threats and explores different approaches to developing such systems, including classical, hybrid, and ensemble methods. To improve the accuracy of IDS systems, this study proposes stacking machine learning models with Mutual Information Gain and Extra Tree Classifier feature selection techniques. Tested on the UNSW-NB15 dataset, this model achieves an accuracy of 96.24%, outperforming recent competing models.

TABLE I: Summary of Literature Review on Intrusion Detection Systems and Network Security

Reference	Methodology	Dataset Used	Key Findings
[12]	Data analytics in cyber security	Not specified	Big Data analytics offers new avenues for enhancing cyber security through analysis of large volumes of network data.
[13]	Machine learning algorithms in IoT for network security	Not specified	Machine learning algorithms in IoT bolster network security by ensuring reliability, security, availability, and survivability of security assets.
[14]	Intrusion detection system with integrated classification model	UNSW-NB15 dataset	Novel intrusion detection system capable of identifying five distinct types of threats with a significantly higher accuracy rate of 83.8% on real-time data.
[15]	Feature reduction with XGBoost for intrusion detection	UNSW-NB15 dataset	XGBoost-based feature reduction method improves test accuracy of binary classification scheme from 88.13% to 90.85%, addressing challenges of high-dimensional and imbalanced datasets.
[16]	Network Intrusion Detection Systems (NIDS) in IoT security	UNSW-NB15 dataset	SVM outperforms other methods with 85.99% accuracy for binary classification and 75.77% for multi-classification.
[17]	Comparative analysis of classification algorithms	UNSW-NB15 dataset	Random Forest classification model achieves highest accuracy of 97.49% compared to Decision Tree and Naïve Bayes models.
[18]	Development of network intrusion detection systems	UNSW-NB15 dataset	Stacking machine learning models with feature selection techniques achieves accuracy of 96.24%, outperforming recent competing models.

5. Methadology

A. Preprocessing

1) Removal Of Extra Feature: To train and test the model, extra features like id, index, etc. has to be removed. In the dataset uses, dropes feature 'Id' from the dataset.

2) Nature Of Features: The understanding of dataset's features have significant role in aspects of machine learning . They play an important role for building accurate and interpretable machine learning

model. They help in reducing dimensionality reduction, handling data-types, handling noise and outliers. The dataset nature's feature is given in 1.

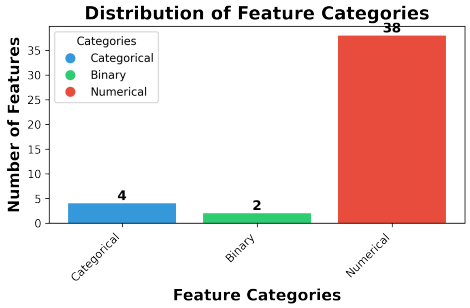

Fig. 1: Feature Categories

3) Null values Handling: Handling null values, are also known as missing values, is very crucial. They can disrupt data integrity, avoid data leakage and the most important improvement of model performance.

4) Data-types of Features: Understanding the data-types of features in a dataset is essential for effective data analysis. Different data-types require different pre-processing techniques and modelling approaches.It also facilitates better data preparation. The data-types of features are 2:

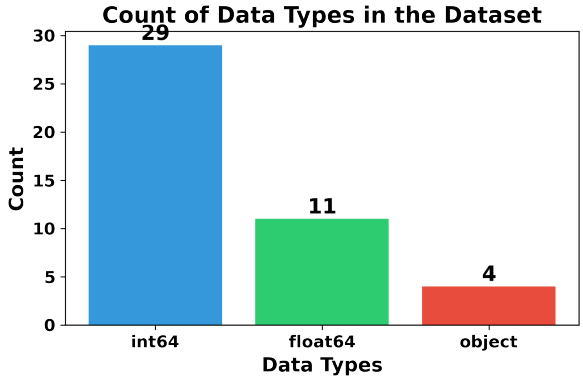

Fig. 2: Data-types of Features

5) Correction Of Features: For correction of any feature, domain knowledge plays an important role to understand the nature of feature. For example, If a feature is in binary form it must be in 0 and 1 form, if it has values other than these, it needs correction. The feature "is_ftp_login" indicates binary form. But it has other than these values, So, it needs correction.

B. Handling categorical features

Handling categorical features is crucial for effective analysis and modelling. This dataset contains various categorical features like proto, services, attack_cat and state 3. To handle categorical features, techniques like one-hot encoding or label encoding can be employed, depending on the nature of the features and the requirements of the machine learning algorithm being used. One-hot encoding converts categorical variables into binary vectors, representing each category as a binary column, while label encoding assigns a unique numerical label to each category. Due to dimensionality reduction, label encoder is used to make numeric the categorical features. Proper handling of categorical features ensures that the model can effectively learn from these variables, capturing the inherent information they contain and improving the accuracy of intrusion detection system (IDS) models trained on the UNSW-NB15 dataset. Additionally, it helps prevent biases and ensures that the model treats categorical variables appropriately during the learning process, ultimately leading to more reliable and robust intrusion detection capabilities.

Categorical Features

Attack	Numbers	States	Numbers	Services	Numbers	Protocols	Numbers
Normal	37000	FIN	39339	None	47153	TCP	43095
Generic	18871	INT	34163	DNS	21367	UDP	29418
Exploits	11132	CON	6982	HTTP	8287	UNAS	3515
Fuzzers	6062	REQ	1842	SMTP	1851	ARP	987
Dos	4089	ACC	4	FTP	1552	OSPT	676
Reconnaissance	3496	RST	1	FTP-DATA	1396		
Analysis	677	CLO	1	POP-3	423		
Backdoor	583			SSH	204		
Shellcode	378			SSL	30		
Worms	44			SNMP	29		
				DHCP	26		
				RADIUS	9		
				IRC	5		

Fig. 3: Categorical Features

C. Feature Selection

In this study, feature selection was implemented using scikit-learn's SelectKBest module with the ANOVA F-statistic. The objective was to identify the most informative features from the dataset while reducing dimensionality. The code snippet utilized SelectKBest to select the top 15 features based on their F-values, indicating their significance in relation to the target variable. By transforming the training data accordingly, the selected features were isolated for further analysis. This approach aims to enhance model performance and inter-pretability by prioritizing the most relevant features for prediction. The resulting selected features provide valuable insights into the underlying data patterns, potentially leading to more accurate and efficient machine learning models.

1. **Calculate Overall Mean**:

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$$

Where $\bar{y}$ is the mean of the target variable y and n is the total number of samples.

2. **Calculate Group Means**:

$$\bar{X}_{i,g} = \frac{1}{n_g} \sum_{j=1}^{n_g} X_{i,j}$$

Where $\bar{X}_{i,g}$ is the mean of feature X_i within group g, n_g is the number of samples in group g, and $X_{i,j}$ is the j-th value of feature X_i in group g.

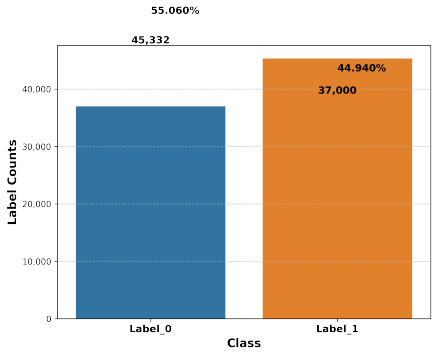

Fig. 4: label-counts

3. **Calculate Sum of Squares Total (SST)**:

$$SST_i = \sum_{j=1}^{n} (X_{i,j} - \bar{X}_i)^2$$

Where SST_i is the total sum of squares for feature X_i, $\bar{X}_i$ is the mean of feature X_i, and n is the total number of samples.

4. **Calculate Sum of Squares Between (SSB)**:

$$SSB_i = \sum_{g=1}^{k} n_g (\bar{X}_{i,g} - \bar{X}_i)^2$$

Where SSB_i is the sum of squares between groups for feature X_i, n_g is the number of samples in group g, $\bar{X}_{i,g}$ is the mean of feature X_i within group g, $\bar{X}_i$ is the overall mean of feature X_i, and k is the number of groups.

5. **Calculate Sum of Squares Within (SSW)**:

$$SSW_i = \sum_{g=1}^{k} \sum_{j=1}^{n_g} (X_{i,j} - \bar{X}_{i,g})^2$$

Where SSW_i is the sum of squares within groups for feature X_i, $X_{i,j}$ is the j-th value of feature X_i in group g, and $\bar{X}_{i,g}$ is the mean of feature X_i within group g.

6. **Calculate F-value**:

$$F_i = \frac{SSB_i/(k-1)}{SSW_i/(n-k)}$$

7. **Select Top 15 Features**: - Select the top 15 features with the highest F-values. These features are considered to have the most significant relationships with the target variable based on the ANOVA F-statistic.

D. Target Variables

1) Binary Classification:: The target variable ”label” in the dataset is used for binary classification, distinguishing between normal traffic and attack instances. Normal traffic instances are labeled as ”0,” indicating regular network behavior, while attack instances are labeled as ”1,” representing malicious activities or intrusion attempts 4.

The binary classification task aims to predict whether a network event is normal or indicative of an attack based on the input features.

Algorithm 1: Feature Selection Algorithm using ANOVA F-statistic

Data: Dataset with features X and target variable y, Number of desired features to select k
Result: Top k features with highest F-values
Step 1: Calculate Overall Mean;
$overall\_mean \leftarrow 0$;
for *each item* y *in dataset* **do**
  $overall\_mean \leftarrow overall\_mean + y$;
$overall\_mean \leftarrow overall\_mean$/total items in dataset;
Step 2: Calculate Group Means;
for *each group* g **do**
  $group\_mean \leftarrow 0$;
  for *each item* X_i *in group* g **do**
    $group\_mean \leftarrow group\_mean + X_i$;
  $group\_mean \leftarrow group\_mean$/total items in group g;
Step 3: Calculate Sum of Squares Total (SST);
for *each feature* X_i **do**
  $sst_i \leftarrow 0$;
  for *each item* X_i, j *in dataset* **do**
    $sst_i \leftarrow sst_i + (X_i, j - overall\_mean)^2$;
Step 4: Calculate Sum of Squares Between (SSB);
for *each feature* X_i **do**
  $ssb_i \leftarrow 0$;
  for *each group* g **do**
    $ssb_i \leftarrow ssb_i +$ total items in group $g \times$ $(group\_mean(X_i, g) - overall\_mean)^2$;
Step 5: Calculate Sum of Squares Within (SSW);
for *each feature* X_i **do**
  $ssw_i \leftarrow 0$;
  for *each group* g **do**
    $group\_mean(X_i, g) \leftarrow 0$;
    for *each item* X_i, j *in group* g **do**
      $group\_mean(X_i, g) \leftarrow$ $group\_mean(X_i, g) + X_i, j$;
    $group\_mean(X_i, g) \leftarrow$ $group\_mean(X_i, g)$/total items in group g;
    for *each item* X_i, j *in group* g **do**
      $ssw_i \leftarrow ssw_i + (X_i, j - group\_mean(X_i, g))^2$;
Step 6: Calculate F-value;
for *each feature* X_i **do**
  $f\_value_i \leftarrow (ssb_i/(num\_groups - 1))/(ssw_i/(total\_items - num\_groups))$;
Step 7: Select Top k Features;
Sort f_values in descending order;
Select top k features with highest f_values;

2) Multi-Class Classification:: The target variable ”attack_cat” in the dataset represents multi-class classification, categorizing network events into different types of attacks. Each category in the ”attack_cat” variable corresponds to a specific type of cyber attack, such as denial-of-service (DoS). The multi-class classification task involves predicting the specific type of attack based on the input features, enabling the detection and classification of various cyber-security threats.

6. Model Selection

In this section, the discussion will be on the selection of models for binary classification and multi-classification tasks.

A. Naive Bayes Model

The Naive Bayes model[19], a probabilistic classifier based on Bayes' theorem with the ”naive” assumption of independence among features, was chosen for its simplicity and computational efficiency. It assumes that the presence of a particular feature in a class is independent of the presence of other features, which allows for fast training and prediction times. Mathematically, the model calculates the probability of each class given the input features using Bayes' theorem and then selects the class with the highest probability

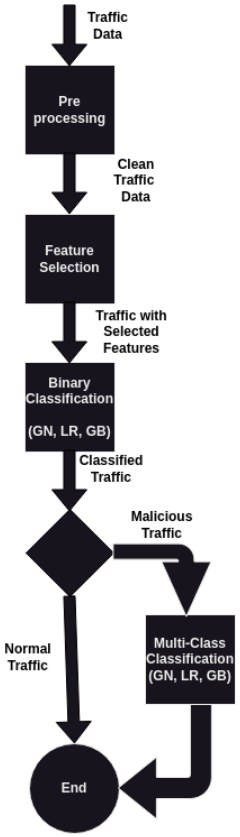

Fig. 5: Proposed Two-Stage Detection Process With Feature Selection

as the predicted class. The probability calculation is represented as:

$$P(y|x_1, x_2, ..., x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i|y)}{P(x_1, x_2, ..., x_n)}$$

where $P(y|x_1, x_2, ..., x_n)$ is the probability of class y given features $x_1, x_2, ..., x_n$, $P(y)$ is the prior probability of class y, $P(x_i|y)$ is the conditional probability of feature x_i given class y, and $P(x_1, x_2, ..., x_n)$ is the probability of observing features $x_1, x_2, ..., x_n$.

1) Prior Probability ($P(y)$): - The prior probability $P(y)$ represents the probability of encountering class y without considering any features. It provides the initial belief or likelihood of each class occurring. - In simpler terms, it reflects the distribution of classes in the dataset.

2) Conditional Probability of Features Given Class ($P(x_i|y)$): - $P(x_i|y)$ is the conditional probability of observing feature x_i given that the class is y. It represents the likelihood of a feature occurring within a specific class. - The "naive" assumption in Naive Bayes is that features are conditionally independent given the class. This means that the presence of one feature is assumed to be unrelated to the presence of any other feature, given the class label. - In practice, this assumption simplifies the model and makes computation tractable.

3) Posterior Probability ($P(y|x_1, x_2, ..., x_n)$): - The posterior probability $P(y|x_1, x_2, ..., x_n)$ is the probability of class y given the observed features $x_1, x_2, ..., x_n$. - It's computed using Bayes' theorem, which updates our belief about the probability of each class given the observed features. - Mathematically, it's calculated by multiplying the prior probability with the likelihood of observing the features given the class, and then dividing by the overall probability of observing the features.

4) Probability of Observing Features ($P(x_1, x_2, ..., x_n)$): - This term represents the overall probability of observing the set of features $x_1, x_2, ..., x_n$. - It acts as a normalization constant in Bayes' theorem, ensuring that the posterior probabilities sum up to 1 across all classes. - In many applications, this term can be ignored during classification since it's constant across all classes and doesn't affect the class selection.

B. Gradient Boosting Model

The Gradient Boosting model, a powerful ensemble method that combines multiple weak learners to create a strong learner, was selected for its ability to handle complex datasets and capture non-linear relationships between features and the target variable. It builds a sequence of decision trees, each focusing on the mistakes made by the previous trees, ultimately reducing the overall error. Mathematically, the model minimizes a loss function by iteratively fitting weak learners (typically decision trees) to the negative gradient of the loss function. The update rule for boosting can be represented as:

$$F_m(x) = F_{m-1}(x) + \lambda h_m(x)$$

where $F_m(x)$ is the prediction of the ensemble at iteration m, $F_{m-1}(x)$ is the prediction of the ensemble at iteration $m-1$, λ is the learning rate, and $h_m(x)$ is the weak learner at iteration m.

- $F_m(x)$ is the prediction of the ensemble at iteration m. It is the sum of the previous ensemble prediction $F_{m-1}(x)$ and the prediction of the current weak learner $h_m(x)$ multiplied by the learning rate λ.
- $F_{m-1}(x)$ represents the ensemble prediction at iteration $m-1$. This is the prediction made by the ensemble before incorporating the current weak learner.
- λ is the learning rate or shrinkage parameter. It controls the contribution of each weak learner to the overall ensemble. A smaller learning rate generally leads to better generalization.
- $h_m(x)$ is the weak learner at iteration m. In gradient boosting, decision trees are commonly used as weak learners. Each weak learner is trained to correct the errors of the previous ensemble.

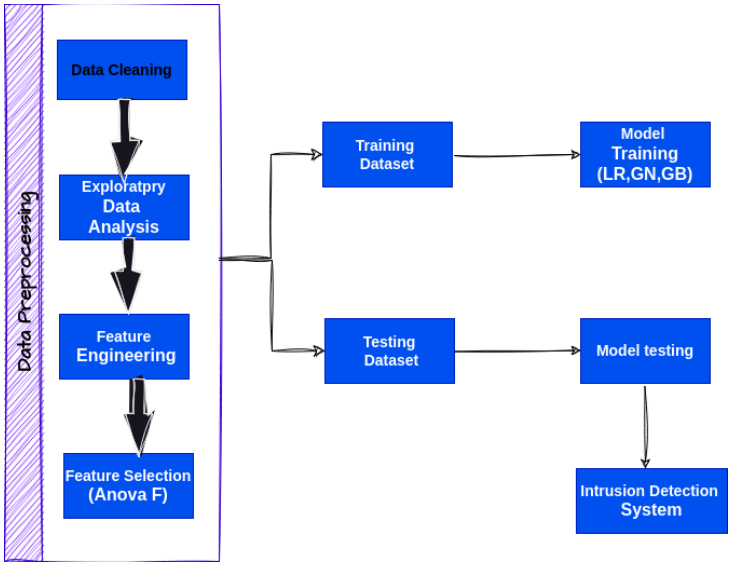

Fig. 6: Methadology

By iteratively adding weak learners to the ensemble and updating the predictions based on their contributions, Gradient Boosting effectively reduces the error and creates a strong predictive model.

C. Logistic Regression Model

The Logistic Regression model, a linear model used for binary classification tasks, was included for its inter-pretability and ease of implementation. Despite its name, Logistic Regression is a classification algorithm that models the probability of a binary outcome using a logistic function. Mathematically, it calculates the probability of an instance belonging to a particular class based on its input features using a linear combination of feature coefficients. The logistic function is represented as:

$$P(y = 1|x) = \frac{1}{1 + e^{-\beta^T x}}$$

where $P(y = 1|x)$ is the probability of the positive class given features x, β is the vector of coefficients, and x is the vector of input features.

The logistic function, also known as the sigmoid function, maps the linear combination of input features to the range [0, 1], representing probabilities. Here's a breakdown:

- $P(y = 1|x)$ is the probability of the positive class given features x. In binary classification, this represents the probability of the outcome being positive or belonging to class 1.
- β is the vector of coefficients. These coefficients are learned during the training process and represent the weights assigned to each feature. The larger the coefficient, the more influence the corresponding feature has on the prediction.
- x is the vector of input features. Each feature is multiplied by its corresponding coefficient in β and summed up to produce the linear combination $\beta^T x$.
- The logistic function $\frac{1}{1+e^{-\beta^T x}}$ transforms the linear combination into a probability value between 0 and 1. It smooths out the predictions and ensures they are interpretable as probabilities.

Logistic Regression predicts the class label based on whether the predicted probability exceeds a certain threshold (often 0.5). It's a simple yet effective algorithm for binary classification tasks, widely used in various domains.

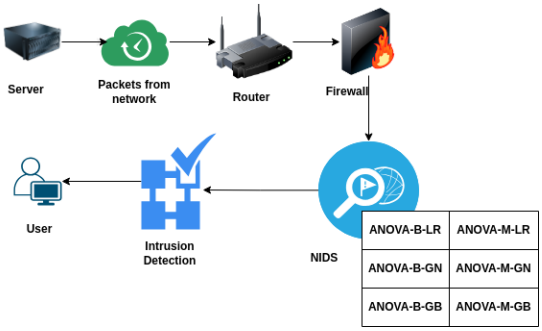

Fig. 7: Intrusion Detection System: ML Approaches

7. Big O Notation Of Models

A. Logistic Regression

1) Training Time:

1) **Forward Pass Through Data**:
   - Time Complexity: $O(nm)$
   - Description: In each iteration of gradient descent, predictions are computed for all training examples by performing a dot product of the feature vector with the learned weights.

TABLE II: Feature Selection and Classification Methods

Method	Method Title	Feature Selection Method	Classification Method	Description
ANOVA-B-LG	An Integrated Approach for Feature Selection and Binary Classification Using Logistic Regression	ANOVA (Binary)	Logistic Regression	This method combines ANOVA feature selection tailored for binary classification tasks with logistic regression as the classifier, offering a comprehensive approach.
ANOVA-B-GN	Feature Selection and Binary Classification with ANOVA and Gaussian Naive Bayes	ANOVA (Binary)	Gaussian Naive Bayes	Utilizing ANOVA feature selection for binary classification alongside Gaussian Naive Bayes as the classifier, it's a probabilistic approach suitable for various datasets.
ANOVA-B-GB	A Feature Selection and Binary Classification Framework Using ANOVA and Gradient Boosting	ANOVA (Binary)	Gradient Boosting	This framework integrates ANOVA feature selection with gradient boosting for binary classification tasks, leveraging boosting techniques for enhanced predictive performance.
ANOVA-M-LG	Multi-Class Classification via ANOVA Feature Selection and Logistic Regression	ANOVA (Multi)	Logistic Regression	Designed for multi-class classification, this method employs ANOVA feature selection in conjunction with logistic regression, suitable for handling multiple classes effectively.
ANOVA-M-GN	Multi-Class Classification via ANOVA Feature Selection and Gaussian Naive Bayes	ANOVA (Multi)	Gaussian Naive Bayes	Offering a probabilistic approach for multi-class classification, this method combines ANOVA feature selection with Gaussian Naive Bayes, accommodating diverse class distributions.
ANOVA-M-GB	Multi-Class Classification via ANOVA Feature Selection and Gradient Boosting	ANOVA (Multi)	Gradient Boosting	Tailored for multi-class classification tasks, this approach integrates ANOVA feature selection with gradient boosting, enhancing predictive accuracy across multiple classes.

- Reason: As there are m training examples, and each prediction operation takes $O(n)$ time (where n is the number of features), the total time complexity for the forward pass is $O(nm)$.

2) **Backward Pass to Compute Gradients**:
- Time Complexity: $O(nm)$
- Description: After computing predictions, gradients of the cost function with respect to each parameter (weight) are computed.
- Reason: Similar to the forward pass, the backward pass also involves m examples, each requiring $O(n)$ time for gradient computation.

3) **Iterations of Gradient Descent**:
- Time Complexity: $O(nm^2)$
- Description: Gradient descent typically converges in $O(m)$ iterations for logistic regression.
- Reason: As each iteration involves both forward and backward passes, and gradient descent converges in $O(m)$ iterations, the overall time complexity of training is $O(nm^2)$.

2) Prediction Time:
- Time Complexity: $O(n)$
- Description: Prediction involves a simple dot product between the feature vector and the learned weights.
- Reason: The time complexity for this operation is linear with the number of features.

B. Gradient Boosting

1) Training Time:

1) **Building Each Decision Tree**:
- Time Complexity: $O(nm \log m)$
- Description: Building each decision tree typically takes $O(nm \log m)$, where n is the number of features and m is the number of training examples.
- Reason: This time complexity arises from the construction of decision trees, which involves sorting data at each node.

2) Prediction Time:

- Time Complexity: $O(n \log m)$
- Description: Prediction involves traversing each tree in the ensemble, which typically takes $O(\log m)$ time per tree.
- Reason: As there are usually $O(n)$ trees in the ensemble, the overall prediction time is $O(n \log m)$.

C. Gaussian Naive Bayes

1) Training Time:

1) **Estimating Mean and Variance**:
   - Time Complexity: $O(nm)$
   - Description: The training process involves estimating the mean and variance of each feature for each class.
   - Reason: This operation takes $O(nm)$ time, where n is the number of features and m is the number of training examples.

2) Prediction Time:

- Time Complexity: $O(n)$
- Description: Prediction involves computing the probability of the input belonging to each class.
- Reason: This operation requires $O(n)$ operations for each class.

D. Time Complexity Analysis

TABLE III: Time Complexities of Machine Learning Algorithms

Algorithm	Training Time	Prediction Time
Logistic Regression	$O(nm^2)$	$O(n)$
Gradient Boosting	$O(nm \log m)$	$O(n \log m)$
Gaussian Naive Bayes	$O(nm)$	$O(n)$

8. Results And Discussions

A. Types of Cyber Attacks

According to [20], the UNSW-NB15 dataset contains information regarding nine instances of cyber attacks, each with its own characteristics 8:

B. Dataset Overview

In this work, the aim is to enhance Intrusion Detection Systems (IDS) by analyzing the UNSW-NB15 dataset [21]. This dataset was chosen due to its recentness and widespread use in the development of general-purpose IDS in the literature. Unlike similar datasets such as AWID [22],KDD99[23], which are mainly intended for intrusion detection in wireless data traffic, the UNSW-NB15 dataset provides a comprehensive collection of network traffic data.

C. Hyper-Parameter Tuning

In the process of hyper-parameter tuning, two machine learning models are employed: Logistic Regression and Gaussian Naive Bayes.

Name	Explanation	Size
Fuzzers	These attacks overwhelm and crash servers and network systems by flooding them with large amounts of randomized data	The training and testing instances for fuzzers are 18,184 and 6,062, respectively.
Backdoors	Exploits that utilize reputable system gateways to gain unauthorized access and install malicious software, allowing attackers remote access to a system	The training and testing instances for backdoors are 1,746 and 583, respectively.
Analysis	Also known as Active Reconnaissance, these attacks gather information about a network without exploiting it, using methods such as port scans and vulnerability scans.	The training and testing instances for analysis are 2,000 and 677, respectively.
Exploits	These attacks target vulnerabilities in operating systems to gain unauthorized access and control.	The training and testing instances for exploits are 33,393 and 11,132, respectively.
Denial of Service (DoS)	Involves overwhelming a network with unauthorized connections, temporarily or permanently blocking resources from authorized users.	The training and testing instances for DoS attacks are 12,264 and 4,089, respectively.
Generic	A type of cryptographic attack targeting the secret key used in encryption.	The training and testing instances for generic attacks are 40,000 and 18,871, respectively.
Reconnaissance	An exploit attack using a payload to gain unauthorized access to a target's computer.An exploit attack using a payload to gain unauthorized access to a target's computer.	The training and testing instances for reconnaissance are 10,491 and 3,496, respectively.
Shellcode	An exploit attack using a payload to gain unauthorized access to a target's computer.	The training and testing instances for shellcode attacks are 1,133 and 378, respectively.
Worms	Rapidly spreading cyber attacks infecting multiple systems and leveraging them as controlled devices.	The training and testing instances for worms are 130 and 44, respectively.

Fig. 8: Types of Cyber Attacks With Size Ratio

1) Logistic Regression:: For Logistic Regression, two different regularization penalties are explored: L1 and L2. Regularization is a technique used to prevent overfitting by penalizing large coefficient values. When utilizing L1 regularization, also known as Lasso regularization, the penalty term encourages sparsity by enforcing some coefficients to be exactly zero. On the other hand, L2 regularization, also known as Ridge regularization, penalizes the squared magnitudes of coefficients, thereby pushing them towards zero without enforcing sparsity.

$$\text{minimize } J(\theta) = \frac{1}{m} \sum_{i=1}^{m} [-y^{(i)} \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))] + \lambda ||\theta||_1$$

$$\text{minimize } J(\theta) = \frac{1}{m} \sum_{i=1}^{m} [-y^{(i)} \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))] + \lambda ||\theta||_2^2$$

Where $||\theta||_1$ represents the L1 norm of the parameter vector θ, and $||\theta||_2^2$ represents the squared L2 norm of the parameter vector θ.
To find the optimal hyperparameters for a machine learning model, a systematic search over a predefined parameter space is performed. The algorithm for hyperparameter tuning typically involves the following steps 2:

2) Gaussian Naive Bayes:: Gaussian Naive Bayes is a probabilistic classifier based on Bayes' theorem with the assumption of independence among features. In the context of hyper-parameter tuning, the primary parameter of interest is var_smoothing. This parameter is used to smooth the data by adding a small, positive value to the variance of all features, allowing the model to handle instances where the likelihood of a certain feature value given a class is zero in the training data. Tuning var_smoothing helps in controlling the model's sensitivity to noisy or irrelevant features, ultimately improving its generalization performance.

The algorithm for hyper-parameter tuning in Gaussian Naive Bayes can be outlined as follows 3:

TABLE IV: Description of Features

Feature	Abbreviation	Description
Record total duration	dur	Transaction duration in seconds
Transaction protocol	proto	Protocol used (http, ftp, etc.)
Service	service	Type of service (http, ftp, etc.)
State	state	Connection state (e.g., ACC, CLO)
Source to destination packet count	spkts	Number of packets from source to destination
Destination to source packet count	dpkts	Number of packets from destination to source
Source to destination transaction bytes	sbytes	Bytes transmitted from source to destination
Destination to source transaction bytes	dbytes	Bytes transmitted from destination to source
Rate	rate	Transmission rate
Source time to live value	sttl	Source time to live value
Destination time to live value	dttl	Destination time to live value
Source bits per second	sload	Source bits per second
Destination bits per second	dload	Destination bits per second
Source packets retransmitted or dropped	sloss	Source packets retransmitted or dropped
Destination packets retransmitted or dropped	dloss	Destination packets retransmitted or dropped
Source interpacket arrival time	sintpkt	Source interpacket arrival time (ms)
Destination interpacket arrival time	dintpkt	Destination interpacket arrival time (ms)
Source jitter	sjit	Source jitter (ms)
Destination jitter	djit	Destination jitter (ms)
Source TCP window advertisement value	swin	Source TCP window advertisement value
Source TCP base sequence number	stcpb	Source TCP base sequence number
Destination TCP base sequence number	dtcpb	Destination TCP base sequence number
Destination TCP window advertisement value	dwin	Destination TCP window advertisement value
TCP connection setup round-trip time	tcprtt	TCP connection setup round-trip time (ms)
TCP connection setup time (SYN to SYN-ACK)	synack	TCP connection setup time (SYN to SYN-ACK)
TCP connection setup time (SYN-ACK to ACK)	ackdat	TCP connection setup time (SYN-ACK to ACK)
Mean of flow packet size (source)	smeansz	Mean flow packet size (source)
Mean of flow packet size (destination)	dmeansz	Mean flow packet size (destination)
Pipelined depth of HTTP request/response transaction	trans_depth	Pipelined depth of HTTP transaction
Actual uncompressed content size	res_bdy_len	Actual uncompressed content size (HTTP)
No. of connections with same service and source address	ct_srv_src	No. of connections with same service and source address
No. of connections for each state	ct_state_ttl	No. of connections for each state
No. of connections with same destination address	ct_dst_ltm	No. of connections with same destination address
No. of connections with same source address and destination port	ct_src_dport_ltm	No. of connections with same source address and destination port
No. of connections with same destination address and source port	ct_dst_sport_ltm	No. of connections with same destination address and source port
No. of connections with same source and destination address	ct_dst_src_ltm	No. of connections with same source and destination address
FTP login status	is_ftp_login	FTP login status (1 for yes, 0 for no)
No. of flows with FTP command	ct_ftp_cmd	No. of flows with FTP command
No. of flows with HTTP methods	ct_flw_http_mthd	No. of flows with HTTP methods
No. of connections with same source address	ct_src_ltm	No. of connections with same source address
No. of connections with same service and destination address	ct_srv_dst	No. of connections with same service and destination address
Source and destination IP address equality	is_sm_ips_ports	Source and destination IP address equality
Attack category	attack_cat	Name of attack category
Label	label	0 for normal, 1 for attack records

D. Over-fitting Controll For Gradient Boosting

Gradient boosting overfitting is controlled through various techniques and adjustments. Initially, including XGBoost for gradient boosting, and tools for evaluation such as accuracy_score and cross_val_score from sklearn. Feature engineering is performed on the training data (X_{train}) to create new features and transformations aimed at improving model performance. This includes creating interaction features like 'rate_dload_interaction' and 'sttl_squared', as well as binning numeric features like 'rate' into categorical bins using pd.cut and performing one-hot encoding for categorical variables like 'state'.

Subsequently, a subset of features ('selected_features_updated') is chosen for training the XGBoost model. Hyperparameters are set for XGBoost with L1 regularization ('params_l1') to help control overfitting. Notably, the depth of the trees is reduced (max_depth: 3) and the number of

Algorithm 2: Hyperparameter Tuning for Logistic Regression

1: Define the parameter grid specifying possible values for hyperparameters
2: Select a scoring metric to evaluate the performance of models with different hyper-parameters
3: Split the dataset into training and validation sets **for** *each combination of hyperparameters in the parameter grid* **do**
4: Train the model on the training set
5: Evaluate the model's performance on the validation set using the chosen scoring metric
6:
7: Identify the hyperparameters that yield the best performance according to the scoring metric
8: Train the final model using the entire training dataset with the selected hyperparameters

Algorithm 3: Hyperparameter Tuning for Gaussian Naive Bayes

1: **Input:** Data X, labels y, parameter grid for var_smoothing
2: **Output:** Best performing var_smoothing value
3: Split the dataset into training and validation sets **for** *each var_smoothing value in the parameter grid* **do**
4: Train a Gaussian Naive Bayes classifier with the given var_smoothing value on the training set
5: Evaluate the classifier's performance on the validation set using a chosen metric
6:
7: Select the var_smoothing value that maximizes performance on the validation set

boosting rounds is decreased ($n\_estimators$: 10), both strategies to prevent the model from learning the training data too well and thereby overfitting.
The model is trained using the XGBoost train function with the specified parameters. To evaluate the model's performance on unseen data, predictions are made on the test set (X_{test}) and accuracy is calculated. Additionally, the threshold for binary predictions is adjusted to experiment with different values, which can affect the trade-off between precision and recall.

E. Confusion Matrix

A confusion matrix 9 [24] is typically used in the field of machine learning to describe the performance of a classification model. It presents a tabular representation of predicted classes versus actual classes. The rows of the matrix represent the actual classes, while the columns represent the predicted classes.

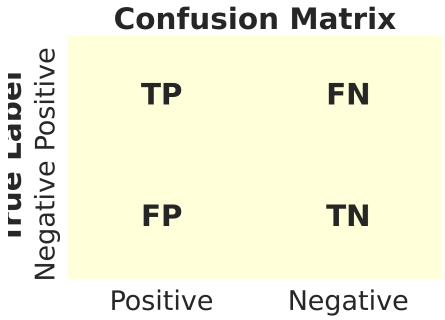

Fig. 9: Confusion-Matrix

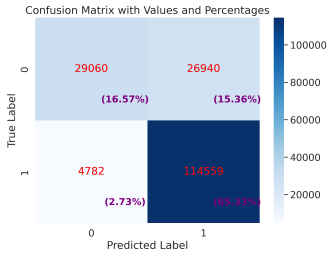

Fig. 10: Confusion-Matrix Guassian Naive Bayes

- **True Positives (TP)**: Cases where the model correctly predicts the positive class.
- **True Negatives (TN)**: Cases where the model correctly predicts the negative class.
- **False Positives (FP)**: Cases where the model incorrectly predicts the positive class.
- **False Negatives (FN)**: Cases where the model incorrectly predicts the negative class.

1) Confusion Matrix For Binary Classification: The confusion matrix of Binary Classification for Guassian Naive Bayes 10, Gradient Boosting 11 and Logistic Regression 12 is given.

2) Confusion Matrix For Multi Classification: The confusion matrix of Multi Classification for Guassian Naive Bayes 13, Gradient Boosting 11 and Logistic Regression 12 is given.

F. Receiver Operating Characteristic

ROC (Receiver Operating Characteristic)[25] is a graphical plot that illustrates the performance of a

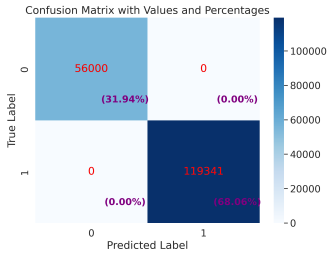

Fig. 11: Confusion-Matrix Gradient Boosting Algorithm

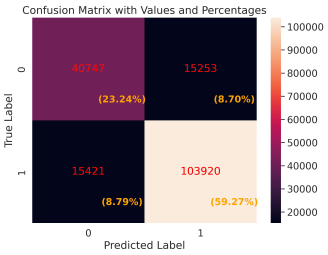

Fig. 12: Confusion-Matrix Linear Regression

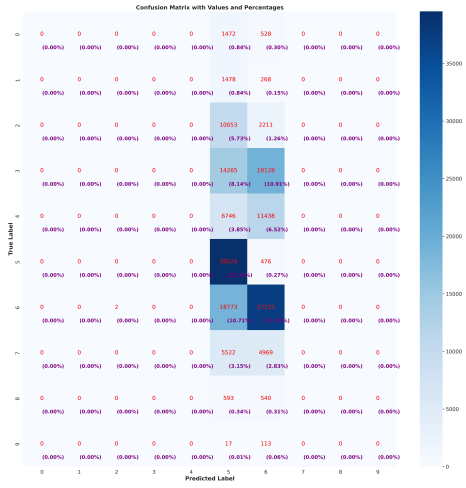

Fig. 13: Confusion-Matrix Guassian Naive Bayes

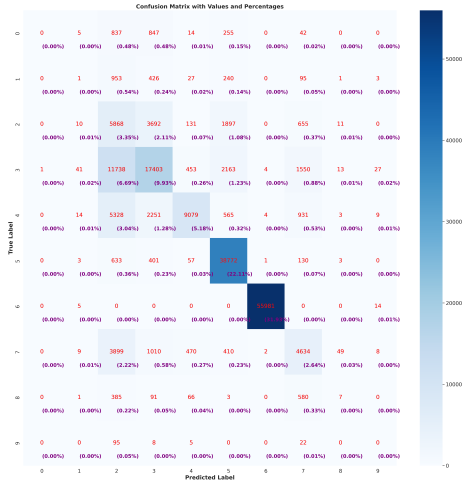

Fig. 14: Confusion-Matrix Gradient Boosting

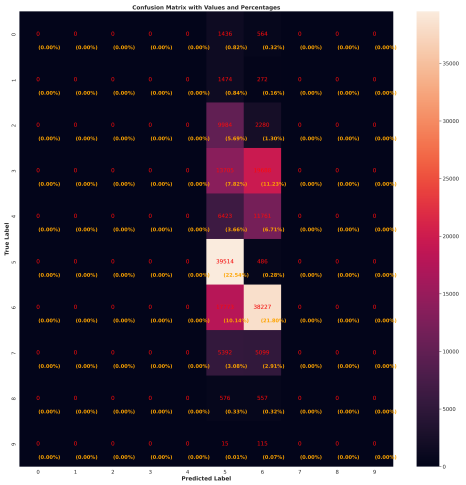

Fig. 15: Confusion-Matrix Logistic Regression

binary classification model across different thresholds. It's a widely used tool for diagnostic test evaluation and machine learning model assessment.

In a binary classification problem, there are two possible outcomes, typically labeled as positive and negative. The ROC curve is created by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings. Here are the key components:

- True Positive Rate (TPR), also known as sensitivity or recall, is the ratio of correctly predicted positive observations to the total actual positive observations in the data.
- False Positive Rate (FPR) is the ratio of incorrectly predicted positive observations to the total actual negative observations in the data. It's calculated as 1 - Specificity.

The ROC curve visually represents the trade-off between TPR and FPR. A classifier with perfect discrimination will have an ROC curve that passes through the upper left corner (0, 1) of the plot, indicating high TPR and low FPR across all thresholds. On the other hand, a random classifier will have an ROC curve that is close to the diagonal line (the line of no-discrimination), which represents random guessing.

The area under the ROC curve (AUC) is a summary statistic that quantifies the overall performance of the classifier. AUC ranges from 0 to 1, where a higher AUC value indicates better discrimination ability of the model. An AUC of 0.5 suggests that the model performs no better than random guessing, while an AUC of 1 indicates perfect classification.

In summary, the ROC curve and AUC provide valuable insights into the predictive performance of a binary classifier, helping users to choose the most appropriate threshold for making classification decisions

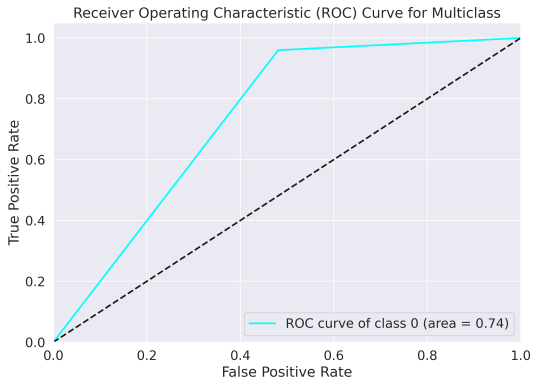

Fig. 16: **Guassian Naive Bayes**

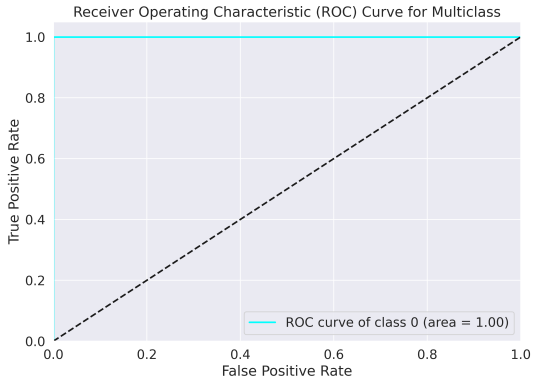

Fig. 17: **Gradient Boosting**

and comparing different models.

1) For Binary Classification:

2) For Multi Classification:

G. Classifier Results

1) Binary Classification : The performance of three distinct classifiers 22 - Gaussian Naive Bayes (ANOVA-B-GN), Gradient Boosting Classifier

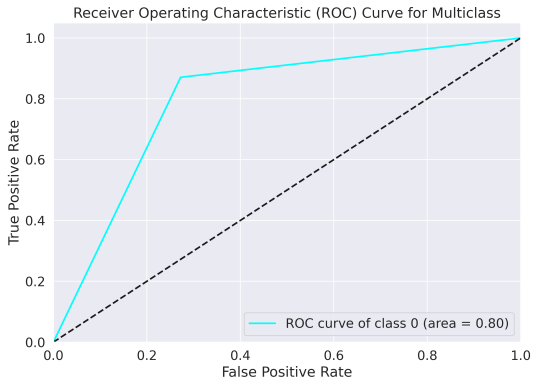

Fig. 18: **Logistic Regression**

Fig. 19: **Guassian Naive Bayes**

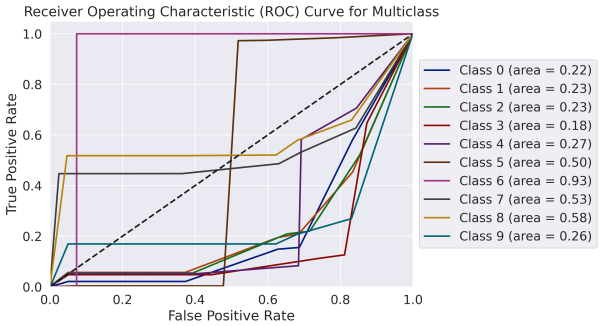

Fig. 20: **Gradient Boosting**

(ANOVA-B-GB), and Logistic Regression (ANOVA-B-LG) - through various performance measures. Gaussian Naive Bayes achieves an accuracy of 82%, accompanied by a precision of 86% for Class 0 and 81% for Class 1, and corresponding recalls of 52% and 96%, respectively. However, it also exhibits a notably high False Alarm Rate of 48.11%. In contrast, the Gradient Boosting Classifier achieves perfect accuracy, precision, and recall, indicating potential overfitting to the training data. Logistic Regression strikes a balance with an accuracy of 83%, featuring a precision of 73% for Class 0 and 87% for Class 1, alongside recalls of 73% and 87%, respectively. Furthermore, Logistic Regression demonstrates a lower False Alarm Rate of 27.24% compared to Gaussian Naive Bayes. These measures provide a comprehensive understanding of each classifier's performance, emphasizing the need

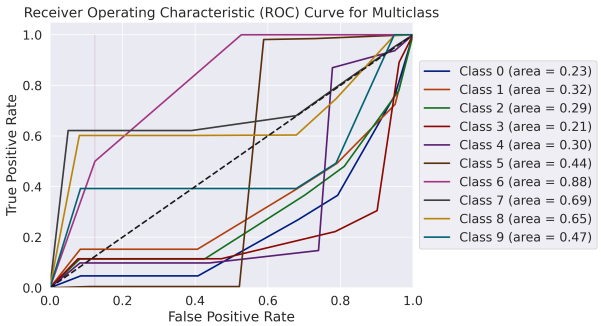

Fig. 21: **Logistic Regression**

for meticulous model selection and fine-tuning to optimize classification outcomes.

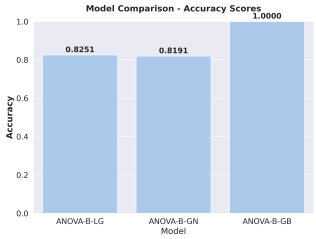

Fig. 22: **Before HyperParameter Tuning**

Logistic Regression, with both L1 and L2 regularization, demonstrates strong precision and recall for both classes. For L1 regularization, precision values are 80% for Class 0 and 100% for Class 1, with recalls of 99% and 88%, respectively, resulting in an accuracy of approximately 91.69%. Similarly, L2 regularization yields precision values of 80% for Class 0 and 100% for Class 1, along with recalls of 99% and 88%, respectively, achieving an accuracy of around 91.67%.

Gradient Boosting, following modification of L1 regularization, exhibits slightly improved accuracy compared to regular Logistic Regression. Precision scores for this model are 89% for Class 0 and 94% for Class 1, with corresponding recall values of 86% and 95%, respectively, resulting in an accuracy of approximately 92.31%.

Gaussian Naive Bayes emerges as the most accurate model, achieving an exceptional accuracy of approximately 98.67% after hyperparameter tuning. Precision scores for both classes are near-perfect, with values of 100% for Class 0 and 98% for Class 1, along with recalls of 96% and 100%, respectively.

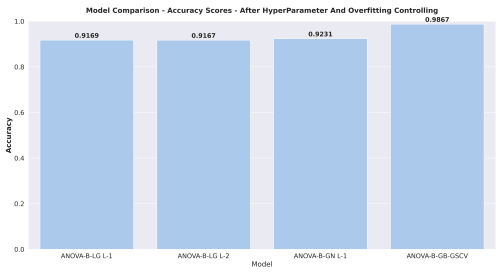

Fig. 23: **After HyperParameter Tuning**

2) Multi-Classification: Starting with Gaussian Naive Bayes (ANOVA-M-GN), 24 the model achieved an accuracy of 44%. However, its precision, recall, and F1-score metrics varied widely across different classes. For example, classes 5 and 6 displayed relatively higher precision and recall values, with class 5 achieving a precision of 40% and recall of 99%, resulting in an F1-score of 0.57, while class 6 achieved a precision of 48% and recall of 66%, resulting in an F1-score of 0.56. Conversely, many other classes exhibited poor performance, with precision, recall, and F1-scores close to zero.

Gradient Boosting (ANOVA-M-GB) achieved an accuracy of 75%, indicating an improvement over Gaussian Naive Bayes. Similar to Gaussian Naive Bayes, precision, recall, and F1-score metrics varied across different classes. Classes 6, 5, and 4 showed relatively higher precision and recall values, with class 6 achieving a precision of 100% and recall of 100%, resulting in an F1-score of 1.00, while class 5 achieved a precision of 88% and recall of 97%, resulting in an F1-score of 0.92. However, some classes, such as 0, 1, and 8, displayed very low precision and recall values.

Logistic Regression (ANOVA-M-LG) also achieved an accuracy of 44%, similar to Gaussian Naive Bayes. Precision, recall, and F1-score metrics varied across different classes, with classes 5 and 6 showing relatively higher precision and recall values. For instance, class 5 achieved a precision of 41% and recall of 99%, resulting in an F1-score of 0.58, while class 6 achieved a precision of 48% and recall of 68%, resulting in an F1-score of 0.57. However, like Gaussian Naive Bayes and Gradient Boosting, many classes exhibited poor performance, with precision, recall, and F1-scores close to zero.

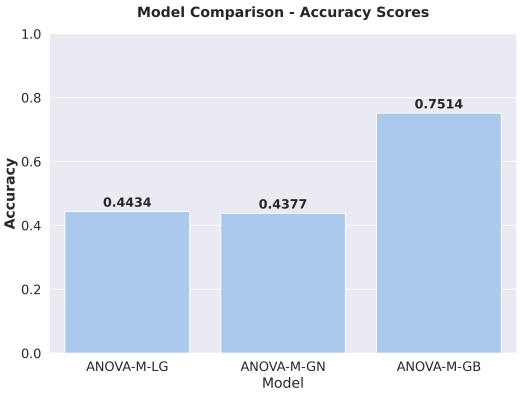

Fig. 24: **Before HyperParameter Tuning**

After Hyper-Parameter Tuning: For L1 regularization (ANOVA-M-LG L-1), 25 the precision values range from 0% to 100%. Class 6 achieved perfect precision, indicating that all instances classified as class 6 were correct. However, precision for other classes varied widely, with classes 4 and 5 exhibiting relatively high precision values of 91% and 68%, respectively, while classes 0, 1, 8, and 9 had precision values of 0%, indicating that no instances were correctly classified for these classes. The recall values also varied, with class 6 achieving a perfect recall of 100%, and class 5 achieving a high recall of 98%. However, many classes had low recall values, indicating that the model failed to correctly identify a significant portion of instances for those classes. Similarly, F1-scores ranged from 0% to 100%, with classes 6 and 5 achieving high F1-scores of 100% and 80%, respectively, while other classes had

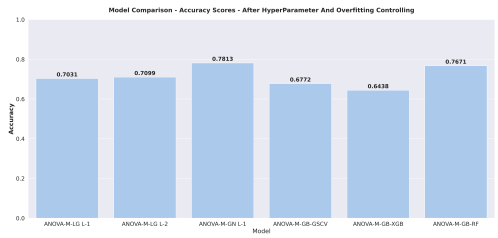

Fig. 25: **After HyperParameter Tuning**

lower F1-scores, indicating challenges in achieving both high precision and recall simultaneously.

For L2 regularization (ANOVA-M-LG L-2), the precision, recall, and F1-score metrics followed a similar pattern. Class 6 achieved perfect precision, recall, and F1-score values, indicating perfect classification for this class. Other classes exhibited varying performance, with classes 4 and 5 showing relatively high precision, recall, and F1-scores, while classes 0, 1, 8, and 9 had lower performance metrics, indicating challenges in correctly classifying instances for these classes.

For XGBoost (ANOVA-M-GB-XGB), the model with modified L1 regularization achieved an accuracy of approximately 78.13%. The precision, recall, and F1-score metrics varied across different classes. Classes 6 and 5 exhibited relatively high precision, recall, and F1-score values, while other classes showed more mixed results, indicating some challenges in classification.

The GridSearchCV (ANOVA-M-GB-GSCV) optimization technique for XGBoost yielded the best parameters, resulting in an accuracy of 78.13%. However, further insights into precision, recall, and F1-score metrics for individual classes are not provided.

For Gaussian Naive Bayes (ANOVA-M-GN L-1), the model with optimized parameters achieved an accuracy of approximately 67.72% on the test set. Precision, recall, and F1-score metrics varied across classes, with classes 6 and 5 displaying relatively higher values compared to other classes.

Random Forest (ANOVA-M-GB-RF) achieved a test set accuracy of approximately 76.71%. Precision, recall, and F1-score metrics also varied across classes, with classes 6 and 5 exhibiting relatively higher values compared to others.

9. Model Comparision

In this section, comparing machine learning models, it's essential to assess their performance comprehensively across various metrics to understand their strengths and weaknesses. Model comparison is a critical step in data analysis and machine learning, involving the evaluation of different statistical or machine learning models to determine which one performs best for a particular dataset and problem.

This process typically involves assessing various metrics such as accuracy, precision, recall, F1-score, or area under the ROC curve, depending on the nature of the problem. Researchers and practitioners often employ techniques like cross-validation or train-test splits to ensure unbiased evaluation and to guard against overfitting 8. Additionally, visualizations such as confusion matrices or ROC curves are commonly used to gain deeper insights into model performance across different classes or thresholds8.

Algorithm	Accuracy	Precision	F-measure
LogisticRegression	70%	90%	54%
MultinomialNB	75%	68%	76%
GaussianNB	71%	75%	2%
BernoulliNB	75%	67%	75%
KNN	78%	87%	71%
DecisionTree	86%	93%	83%
AdaBoost	84%	95%	80%
RandomForest	87%	98%	84%
MLPClassifier	13%	71%	73%
GradientBoosting	86%	98%	82%

TABLE V: [26] Performance Comparision with other classifier models

ML Methods	**Feature Selection Method**	**AC. (%)**
RF, XG Boost, DT[27]	Feature Importance (RF)	75.65, 71.74, 74.49
SVM [16]	All features	85.99
DT, MLP [[15]	XGBoost	90.85, 84.39
Proposed DT	Backward Elimination	92.76
Proposed RF	(Chi Square)	86.16
Proposed GBT	-	88.03
Proposed MLP	-	84.91

TABLE VI: [28] Model Performance with Feature Selection Methods

Classifier	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**MSE**	**TPR**	**FPR**
LR	98.42	0.98	0.99	0.99	0.015	0.994	0.048
NB	76.59	1.00	0.69	0.82	0.234	0.693	0.007
RE	99.57	1.00	1.00	1.00	0.004	0.997	0.009
SGD	98.16	0.98	1.00	0.99	0.018	0.983	0.043
KNN	98.28	0.99	0.99	0.99	0.017	0.989	0.039

TABLE VII: [29] Classifier Performance Metrics

10. Conclusion

The study conducted a comprehensive analysis of the efficiency and reliability of the UNSW-NB15 features through the application of ANOVA F-statistic for feature selection in conjunction with various machine learning (ML) techniques. Figures 22, 23, 24, and 25 depicted the comparative evaluation of these techniques.

In case of binary classification, the results highlighted that Gradient Boosting, coupled with GridSearch Cross Validation (ANOVA-B-GB-GSCV), exhibited the highest accuracy, achieving an impressive score of 0.9867. This underscores the robustness and efficiency of Gradient Boosting in discerning between normal and malicious network activities based on the UNSW-NB15 features. The utilization of GridSearch Cross Validation further optimized the model parameters, enhancing its predictive performance and generalization capabilities.

On the other hand, in the domain of multiclassification, Gaussian Naive Bayes in conjunction

TABLE VIII: Comparison of models with additional metrics for the Best Parameters

Metric	ANOVA-B-GN	ANOVA-B-GB	ANOVA-B-LR	ANOVA-B-LG L-1	ANOVA-B-LG L-2	ANOVA-B-GN L-1	ANOVA-B-GB-GSCV
Precision	0.86	1.00	0.54	0.80	0.80	0.89	1.00
	0.81	1.00	0.92	1.00	1.00	0.94	0.98
Recall	0.52	1.00	0.88	0.99	0.99	0.86	0.96
	0.96	1.00	0.65	0.88	0.88	0.95	1.00
F1-score	0.65	1.00	0.67	0.88	0.88	0.88	0.98
	0.88	1.00	0.76	0.93	0.94	0.94	0.99
Support	56,000	56,000	56,000	56,000	56,000	56,000	56,000
	119,341	119,341	119,341	119,341	119,341	119,341	119,341
Accuracy	0.82	1.00	0.72	0.9166	0.9168	0.9231	0.9653
Macro Avg	0.83	1.00	0.73	0.90	0.90	0.91	0.99
	0.74	1.00	0.77				
Weighted Avg	0.83	1.00	0.80	0.92	0.92	0.92	0.99
	0.82	1.00	0.72				
Best Parameters							{ 'var_smoothing': 1e-20 }
Best Accuracy							0.9867

TABLE IX: Proposed Models and their methods

Class	Precision	Recall	F1-Score
0	0.00	0.00	0.00
1	0.00	0.00	0.00
2	0.00	0.00	0.00
3	0.00	0.00	0.00
4	0.00	0.00	0.00
5	0.40	0.99	0.57
6	0.48	0.66	0.56
7	0.00	0.00	0.00
8	0.00	0.00	0.00
9	0.00	0.00	0.00
Accuracy			0.4434
Macro Avg	0.09	0.17	0.11
Weighted Avg	0.25	0.44	0.31

TABLE X: ANOVA-M-GN

Class	Precision	Recall	F1-Score / Support	
0	0.00	0.00	0.00	2000
1	0.01	0.00	0.00	1746
2	0.20	0.48	0.28	12264
3	0.67	0.52	0.58	33393
4	0.88	0.50	0.64	18184
5	0.88	0.97	0.92	40000
6	1.00	1.00	1.00	56000
7	0.54	0.44	0.48	10491
8	0.08	0.01	0.01	1133
9	0.00	0.00	0.00	130
Accuracy			0.75	
Macro Avg	0.42	0.39	0.39	
Weighted Avg	0.78	0.75	0.76	

TABLE XI: ANOVA-M-GB

with L1-Regularization (ANOVA-M-GN L-1) emerged as the most proficient technique, attaining a commendable accuracy of 0.7813. This signifies the suitability of Gaussian Naive Bayes for discerning among multiple network intrusion categories with the aid of the UNSW-NB15 features set. The incorporation of L1-Regularization facilitated feature selection and regular-

Class	Precision	Recall	F1-Score / Support	
0	0.00	0.00	0.00	2000
1	0.00	0.00	0.00	1746
2	0.00	0.00	0.00	12264
3	0.00	0.00	0.00	33393
4	0.00	0.00	0.00	18184
5	0.41	0.99	0.58	40000
6	0.48	0.68	0.57	56000
7	0.00	0.00	0.00	10491
8	0.00	0.00	0.00	1133
9	0.00	0.00	0.00	130
Accuracy			0.44	175341
Macro Avg	0.09	0.17	0.11	175341
Weighted Avg	0.25	0.44	0.31	175341

TABLE XII: ANOVA-M-LR

Class	Precision	Recall	F1-Score / Support	
0	0.00	0.00	0.00	2000
1	0.00	0.00	0.00	1746
2	0.29	0.22	0.25	12264
3	0.63	0.44	0.52	33393
4	0.91	0.16	0.28	18184
5	0.68	0.98	0.80	40000
6	1.00	1.00	1.00	56000
7	0.30	0.74	0.43	10491
8	0.00	0.00	0.00	1133
9	0.00	0.00	0.00	130
Accuracy			0.70	175341
Macro Avg	0.38	0.35	0.33	175341
Weighted Avg	0.73	0.70	0.67	175341

TABLE XIII: ANOVA-M-LG L-1

Class	Precision	Recall	F1-Score / Support	
0	0.00	0.00	0.00	2000
1	0.00	0.00	0.00	1746
2	0.30	0.37	0.33	12264
3	0.54	0.49	0.51	33393
4	0.92	0.13	0.23	18184
5	0.73	0.98	0.84	40000
6	1.00	1.00	1.00	56000
7	0.34	0.61	0.44	10491
8	0.00	0.00	0.00	1133
9	0.00	0.00	0.00	130
Accuracy			0.71	175341
Macro Avg	0.38	0.36	0.34	175341
Weighted Avg	0.73	0.71	0.68	175341

TABLE XIV: ANOVA-M-LG L-2

Class	Precision	Recall	F1-Score / Support	
0	0.00	0.00	0.00	2000
1	0.00	0.00	0.00	1746
2	0.30	0.37	0.33	12264
3	0.54	0.49	0.51	33393
4	0.92	0.13	0.23	18184
5	0.73	0.98	0.84	40000
6	1.00	1.00	1.00	56000
7	0.34	0.61	0.44	10491
8	0.00	0.00	0.00	1133
9	0.00	0.00	0.00	130
Accuracy			0.71	175341
Macro Avg	0.38	0.36	0.34	175341
Weighted Avg	0.73	0.71	0.68	175341

TABLE XV: ANOVA-M-LG L-2

Class	Precision	Recall	F1-Score / Support	
0	0.00	0.00	0.00	2000
1	0.02	0.03	0.02	1746
2	0.28	0.09	0.14	12264
3	0.85	0.40	0.54	33393
4	0.28	0.20	0.24	18184
5	0.98	0.91	0.95	40000
6	1.00	1.00	1.00	56000
7	0.05	0.03	0.04	10491
8	0.03	0.98	0.06	1133
9	0.00	0.05	0.01	130
Accuracy			0.64	175341
Macro Avg	0.35	0.37	0.30	175341
Weighted Avg	0.76	0.64	0.68	175341

TABLE XVI: ANOVA-M-GB-GSCV

Class	Precision	Recall	F1-Score	Support
0	0.00	0.00	0.00	2000
1	0.02	0.03	0.02	1746
2	0.26	0.11	0.16	12264
3	0.84	0.40	0.54	33393
4	0.29	0.23	0.26	18184
5	0.98	0.91	0.94	40000
6	1.00	1.00	1.00	56000
7	0.05	0.04	0.04	10491
8	0.03	0.91	0.06	1133
9	0.01	0.09	0.02	130
Accuracy			0.64	175341
Macro Avg	0.35	0.37	0.30	175341
Weighted Avg	0.75	0.64	0.68	175341

TABLE XVII: ANOVA-M-GB-XGB

Class	Precision	Recall	F1-Score	Support
0	0.00	0.00	0.00	2000
1	0.22	0.00	0.00	1746
2	0.22	0.24	0.23	12264
3	0.64	0.60	0.62	33393
4	0.77	0.63	0.69	18184
5	0.81	0.97	0.88	40000
6	1.00	1.00	1.00	56000
7	0.47	0.48	0.47	10491
8	0.04	0.02	0.02	1133
9	0.00	0.00	0.00	130
Accuracy			0.77	175341
Macro Avg	0.42	0.39	0.39	175341
Weighted Avg	0.75	0.77	0.75	175341

TABLE XVIII: ANOVA-M-GB-RF

ization, thereby enhancing the model's interpretability and mitigating the risk of overfitting.

In conclusion, the findings of this study underscore the importance of feature selection and the judicious selection of ML techniques in the context of network intrusion detection using the UNSW-NB15 dataset. Gradient Boosting with Grid-Search CV excelled in binary classification tasks, while Gaussian Naive Bayes with L1-Regularization demonstrated prowess in multi-classification scenarios. These insights could inform the development of more effective and reliable intrusion detection systems, thereby bolstering cybersecurity measures in network environments. Further research could goes into exploring ensemble methods, deep learning architectures and importance of federateed learning for enhanced performance and resilience in the face of evolving cyber threats. Additionally, investigating the impact of feature engineering and data augmentation techniques on model performance could offer valuable insights for future endeavors in this domain.

REFERENCES

[1] J. M. Kizza, "Computer network security protocols," in *Guide to Computer Network Security*. Springer, 2024, pp. 409–441.

[2] H. Satilmiş, S. Akleylek, and Z. Y. Tok, "A systematic literature review on host-based intrusion detection systems," *IEEE Access*, vol. 12, pp. 27 237–27 266, 2024.

[3] O. H. Abdulganiyu, T. Ait Tchakoucht, and Y. K. Saheed, "A systematic literature review for network intrusion detection system (ids)," *International Journal of Information Security*, vol. 22, no. 5, pp. 1125–1162, 2023.

[4] J. Li, X. Tong, J. Liu, and L. Cheng, "An efficient federated learning system for network intrusion detection," *IEEE Systems Journal*, 2023.

[5] A. Alazab, A. Khraisat, S. Singh, and T. Jan, "Enhancing privacy-preserving intrusion detection through federated learning," *Electronics*, vol. 12, no. 16, p. 3382, 2023.

[6] S. Yaras and M. Dener, "Iot-based intrusion detection system using new hybrid deep learning algorithm," *Electronics*, vol. 13, no. 6, p. 1053, 2024.

[7] A. Singh, J. Prakash, G. Kumar, P. K. Jain, and L. S. Ambati, "Intrusion detection system: A comparative study of machine learning-based ids," *Journal of Database Management (JDM)*, vol. 35, no. 1, pp. 1–25, 2024.

[8] F. Louati, F. Barika Ktata, and I. Amous, "An intelligent security system using enhanced anomaly-based detection scheme," *The Computer Journal*, p. bxae008, 2024.

[9] K. Alsubhi, "A secured intrusion detection system for mobile edge computing," *Applied Sciences*, vol. 14, no. 4, p. 1432, 2024.

[10] N. Lewandowska, "Intrusion detection systems: Categories, attack detection and response." 2024.

[11] C. M. Santander, S. Galms, and Y. d. l. N. C. Gavilánez, "Iomt data server risks and vulnerabilities," *Migration Letters*, vol. 21, no. S2, pp. 711–728, 2024.

[12] O. Savas and J. Deng, *Big data analytics in cybersecurity*. CRC Press, 2017.

[13] S. Mishra and A. K. Tyagi, "The role of machine learning techniques in internet of things-based cloud applications," *Artificial intelligence-based internet of things systems*, pp. 105–135, 2022.

[14] V. Kumar, D. Sinha, A. K. Das, S. C. Pandey, and R. T. Goswami, "An integrated rule based intrusion detection system: analysis on unsw-nb15 data set and the real time online dataset," *Cluster Computing*, vol. 23, pp. 1397–1418, 2020.

[15] S. M. Kasongo and Y. Sun, "Performance analysis of intrusion detection systems using a feature selection method on the unsw-nb15 dataset," *Journal of Big Data*, vol. 7, no. 1, p. 105, 2020.

[16] D. Jing and H.-B. Chen, "Svm based network intrusion detection for the unsw-nb15 dataset," in *2019 IEEE 13th international conference on ASIC (ASICON)*. IEEE, 2019, pp. 1–4.

[17] R. Tahri, A. Jarrar, A. Lasbahani, and Y. Balouki, "A comparative study of machine learning algorithms on the unsw-nb 15 dataset," in *ITM Web of Conferences*, vol. 48. EDP Sciences, 2022, p. 03002.

[18] M. H. Kabir, M. S. Rajib, A. S. M. T. Rahman, M. M. Rahman, and S. K. Dey, "Network intrusion detection using unsw-nb15 dataset: stacking machine learning based approach," in *2022 International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE)*. IEEE, 2022, pp. 1–6.

[19] K. P. Murphy *et al.*, "Naive bayes classifiers," *University of British Columbia*, vol. 18, no. 60, pp. 1–8, 2006.

[20] S. More, M. Idrissi, H. Mahmoud, and A. T. Asyhari, "Enhanced intrusion detection systems performance with unsw-nb15 data analysis," *Algorithms*, vol. 17, no. 2, p. 64, 2024.

[21] A. Dickson and C. Thomas, "Analysis of unsw-nb15 dataset using machine learning classifiers," in *Machine Learning and Metaheuristics Algorithms, and Applications: Second Symposium, SoMMA 2020, Chennai, India, October 14–17, 2020, Revised Selected Papers 2*. Springer, 2021, pp. 198–207. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-16-0419-5_16

[22] A. A. Bhutta, M. u. Nisa, and A. N. Mian, "Lightweight real-time wifi-based intrusion detection system using lightgbm," *Wireless Networks*, vol. 30, no. 2, pp. 749–761, 2024.

[23] F. Chen, J. Bai, and W. Gao, "Anomaly detection: research on model selection based on multiple models," in *Ninth International Symposium on Sensors, Mechatronics, and Automation System (ISSMAS 2023)*, vol. 12981. SPIE, 2024, pp. 395–402. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/12981/129811P/Anomaly-detection--research-on-model-selection-based-on-multiple/10.1117/12.3015099.full?tab=ArticleLinkCited

[24] E. Helmud, F. Fitriyani, P. Romadiana *et al.*, "Classification comparison performance of supervised machine learning random forest and decision tree algorithms using confusion matrix," *Jurnal Sisfokom (Sistem Informasi dan Komputer)*, vol. 13, no. 1, pp. 92–97, 2024.

[25] P. McCarthy, "Predicting trips to health care facilities: A binary logit and receiver operating characteristics (roc) approach," *Research in Transportation Economics*, vol. 103, p. 101411, 2024.

[26] O. Almomani, M. A. Almaiah, A. Alsaaidah, S. Smadi, A. H. Mohammad, and A. Althunibat, "Machine learning classifiers

for network intrusion detection system: Comparative study," in *2021 International Conference on Information Technology (ICIT)*, 2021, pp. 440–445.

[27] N. M. Khan, N. Madhav C, A. Negi, and I. S. Thaseen, "Analysis on improving the performance of machine learning models using feature selection technique," in *Intelligent Systems Design and Applications: 18th International Conference on Intelligent Systems Design and Applications (ISDA 2018) held in Vellore, India, December 6-8, 2018, Volume 2*. Springer, 2020, pp. 69–77.

[28] R. A. Disha and S. Waheed, "A comparative study of machine learning models for network intrusion detection system using unsw-nb 15 dataset," in *2021 International Conference on Electronics, Communications and Information Technology (ICECIT)*. IEEE, 2021, pp. 1–5.

[29] G. Kocher and G. Kumar, "Analysis of machine learning algorithms with feature selection for intrusion detection using unsw-nb15 dataset," *Available at SSRN 3784406*, 2021.