Astana IT University

UDC 81.93.29

Galymzhan Beketay
Adilet Torebek

Traffic analysis using machine learning algorithms

6B06301 — Cybersecurity

Diploma work

Supervisor
Tulepova G.
MSc / Masters in Technical science

Kazakhstan Republic
Astana, 2024

CONTENTS

Abstract

This study investigates machine learning models that aim to improve the security of IoT devices by introducing noise into labels to mitigate the effects of overfitting. The increasing complexity and dynamics of cyber threats require innovative approaches to ensure reliable security for the Internet of Things. We used probabilistic and linear models based on trees and examples to analyze network traffic and identify patterns and anomalies that indicate potential attacks.

Using advanced object engineering techniques, the study expands the ability to detect and classify various machine learning models. These models are trained and tested on the CICIoT2023 dataset, which represents different types of attacks such as DDoS attacks, DNS spoofing, and reconnaissance attacks. The results show that when properly configured and trained with noisy labels, probabilistic and tree-based models significantly increase the accuracy and reliability of threat detection systems. Furthermore, the use of classification models demonstrates the adaptability and effectiveness of these methods in real-time traffic analysis, also we offers a comprehensive evaluation framework for assessing the performance of various machine learning models in diverse IoT security scenarios, paving the way for enhanced threat mitigation strategies. This study highlights the potential of machine learning in transforming Internet of Things security practices, offering a robust solution to emerging cyber threats, and provides insights into further research to further optimize these models.

Keywords: Network Security, Internet of Things, Machine Learning, Decision Tree Classifier, Random Forest Classifier, k-Nearest Neighbors, Xgboots, Adaboost, Naive Bayes, Linear Regression, Feature Engineering, Noise Introduction, Dataset, Network traffic analysis.

Introduction

Network security is critical in an era characterized by digital dependency and interconnection. The risks presented by cyber threats are growing as firms depend more and more on network systems for communication, data sharing, and operational efficiency. Network traffic analysis (NTA), which provides insights into network traffic behavior and makes it possible to identify and mitigate anomalies and malicious activity, emerges as a critical tool in the toolbox against such threats. In the past, NTA used signature-based techniques and rule-based systems to recognize recognized threats. But given how constantly evolving cyber threats are, a more flexible and proactive strategy is required. The discipline of NTA experienced a revolution in recent years with the integration of Machine Learning (ML) techniques. This integration has allowed for automated, data-driven analysis that may identify minor trends that may indicate hostile behavior. This diploma project begins a thorough investigation of the intersection of network traffic analysis and machine learning. This study intends to clarify the effectiveness of ML algorithms in improving the precision, effectiveness, and flexibility of NTA processes by exploring the most recent research, techniques, and developments in the field. A thorough comparison of several machine learning models, such as Naive Bayes, Random Forest, Decision Tree, AdaBoost, XGBoost, K-Nearest Neighbors, and Logistic Regression, is included in the study. The effectiveness of each model in categorizing network traffic is examined through careful experimentation and evaluation, providing insight into their advantages, disadvantages, and applicability for various NTA scenarios. Furthermore, this diploma project creatively combines methods to deal with issues like dataset imbalance and overfitting. Through the use of ensemble approaches and data introducing using noisy labels, among other strategies, the study investigates ways to improve model robustness and generalization ability even in situations where data is limited. In addition, this diploma work breaks new ground by outlining possibilities for future study and new developments in the field of ML-driven NTA. The paper provides a roadmap for innovation and advancement in defending network infrastructures against changing cyber threats, from sophisticated deep learning models to real-time analysis and resistance against adversarial attacks.

1 Literature Review

Lately, researchers have addressed issues related to the application of ML techniques in analyzing network traffic. Conventional strategies for traffic can seldom meet the challenges of dynamically growing cyber threats and complexities. ML contains methods for conducting big data, recognizing patterns, and all that can bring innovation to the enhancement of the process of traffic analysis. This paper is precisely going to survey several research works in this domain and evaluate the key theories, methodologies, and outstanding findings stemming from very recent research results.

For instance, one work proposed an innovative, deep learning-based method for advanced cyber threat detection, recording a remarkable increase in the accuracy and efficiency of the method when compared to the traditional ones. Another research initiative emphasizes current pattern changes brought in by ML algorithms for constant monitoring of traffic and the potential function change of the remaining functions of network security.

Abbasi, Shahraki, and Taherkordi(2021) [1] have detailed how deep learning techniques can be applied to monitor and analyze network traffic. Various models have been looked at, such as Convolutional Neural Networks, Recurrent Neural Networks, and Autoencoders, for the reasons that these models can work for voluminous amounts of data and present great complexity in their patterns, respectively. The use of such models is, therefore, highly recommendable because they potentially offer greater strength in feature extraction and an enhancement in the accuracy and efficiency of Rasdaman in NTA functions. Convolutional Neural Networks used as a way of capturing the spatial features extracted from the traffic data of the network. Recurrent Neural Networks used to handle the sequence structure in a good manner, making it perfect for time-series extracted from network traffic. Autoencoder is a method that is used for extracting minimal normal representations of that traffic and detecting how they deviate.

Wei and Heidemann (2020) [2] have studied this common DNS spoofing attack by poisoning the DNS cache. The attack has been well studied from the perspectives of trend and detection for over six years. It investigates the trend of DNS spoofing and suggests methodologies for detection. It points out the requirements needed for real-time monitoring and the application of machine learning algorithms to find a spoofing pattern. Feature Engineering - ability to extract relevant features from DNS traffic for training ML models. Supervised Learning - ability to learn from labeled data to train classifiers that can discriminate between the real and spoofed DNS responses.

In addition, Priya et al. (2020) [3] investigated the identification of DDoS attacks with the use of machine learning algorithms. They applied a wide range of models, including support vector machines, decision trees, and neural networks,

in finding DDoS attacks from network traffic. This paper shows that the process accuracy is improved through selection attributes and a combination of models. Support Vector Machines (SVM) classifies high-dimensional data. Decision Trees: Works in such a way that the results are interpretable and it is strong while working with extensive datasets. Hybrid Models - combining more than one model taps into different strengths in a bid to deliver better results.

Gaber, El-Ghamry, and Hassanien (2022) [4] take on the inject attack against IoT applications via a machine learning-based detection approach. In the approach, the ML models are trained on IoT network traffic data in regard to their injection attack identification. In this respect, it is important to acknowledge that one of the characteristics of IoT systems is the diversity of devices involved in an IoT infrastructure, and it is essential to have scalable solutions for facing these challenges. Key ones include Random Forest, said to yield satisfactory results and to be relatively not susceptible to imbalanced datasets. Gradient Boosting has a high accuracy rate and works efficiently when it comes to classification.

Althnian et al. (2021) [5] aimed at reviewing how the size of the dataset impacted machine learning classifiers. While their work strived to apply in the health domain, the findings could be relevant for NTA as well. In NTA, a huge dataset could efficiently increase the accuracy of ML models. Authors argue that larger datasets tend to better model performance, but other important roles are also played by data quality and feature selection. Key takeaways: The larger the dataset, the better the model generally performs, although with diminishing returns beyond a certain point. Data quality has to be high and it should contain relevant features for efficient training of ML models.

These included clustering or anomaly detection algorithms, under unsupervised learning techniques. These are some of the most acceptable methods, as they include the ability to detect novel or unseen attacks without using labeled data. The most commonly used techniques for identifying network traffic anomalies include k-means clustering, DBSCAN, and isolation forests. k-Means Clustering organizes network traffic data into groups, in which anomalies are those that do not fit well into any group (Aditya et al., 2020) [6]. DBSCAN is a form of density-based clustering that identifies anything that lies outside the norm as suspicious (Zhang et al, 2021) [7]. Isolation Forests are nice because they partition data very effectively, allowing for the isolation of anomalies, a good approach for real-time anomaly detection (Kaiyoo, 2020) [8].

Supervised learning models are widely applied in traffic classification. It is defined as the classification of network traffic into corresponding classes against some a priori defined categories. According to learning styles, models need to get instances' labeled data. Some common algorithms of supervised learning models include logistic regression, naïve Bayes, a series of ensemble methods like random

Study	Focus Area	Models used	Key Findings
Abbasi, Shahraki, and Taherkordi (2021) [1]	Deep Learning in NTA	CNNs, RNNs, Autoencoders	Enhanced feature extraction and accuracy in threat detection
Wei and Heidemann (2020) [2]	DNS Spoofing Detection	Supervised Learning	Effective real-time detection of spoofed DNS responses using feature engineering.
Priya et al. (2020) [3]	DDoS Attack Detection	SVM, Decision Trees, Neural Networks	Improved detection accuracy through model combination.
Gaber, El-Ghamry, and Hassanien (2022) [4]	IoT Network Security	Random Forest, Gradient Boosting	Effective handling of diverse IoT datasets and improved attack detection.
Aditya et al. (2020); Zhang et al. (2021); Kaiyoo (2020) [6–8]	Anomaly Detection	k-means, DBSCAN, Isolation Forests	Effective detection of novel anomalies without labeled data.

Table 1.1 – Summary of Key Studies in Network Traffic Analysis

forests, and boosting. Logistic Regression is a simple and very useful approach, especially when dealing with NTA binary classification tasks (Hector, 2023) [9]. Naïve Bayes is Based on Bayes' theorem, especially useful in spam detection and other similar services (Thakkar et al., 2021) [10]. Ensemble Methods is an approach in which disparate models of calculation are used to reach the same purpose. It is used to enhance the precision and reliability of the classification simultaneously (Hu et al., 2023) [11].

Moreover, machine learning models focus not only on detection but also on strategizing methods to evade the attack's impact. For example, techniques in the learning mechanism of reinforcement try to adapt dynamically in accordance with the network conditions and nature of the threat for an automated response to detected anomalies. Reinforcement Techniques are techniques leading the system toward optimized policies of network security because of reward-based learning for automation of response strategies [12]. Adaptive Systems adjust their behavior according to the real-time analysis and predictions of the ML model for effective attack mitigation [13].

2 Methodology

2.1 Dataset for network classification

We used the CICIoT2023 dataset for network traffic analysis in this investigation that created by "Canadian Institute for Cybersecurity"and its IoT topology with more than 100 devices, which provides a comprehensive representation of network traffic in an IoT scenario. The dataset includes network traffic information from one benign type and thirty-three different attack types, for a total of thirty-four categories. These attacks includes:

- Distributed Denial of Service (DDoS)
- Denial of Service (DoS)
- Reconnaissance (Recon)
- Web-based attacks
- Brute Force attacks
- Spoofing
- Mirai botnet attacks

Because it includes a wide range of attack types also, this dataset is ideal for testing and training machine learning models for network intrusion detection. The dataset is composed of 169 distinct CSV files, totaling about 13 GB of data and 45,6 million rows data traffic. Every CSV file includes comprehensive records of network traffic, along with a range of attributes that characterize the kind and flow of the traffic. The protocols in the dataset is represented by a separate column containing boolean flags that indicate whether traffic for that protocol is present (True) or not (False). This offers an extensive perspective of the data flow across multiple protocols, facilitating in-depth examination. The following procedures are part of the dataset, http, https, dns, telnet, smtp, ssh, irc, tcp, udp, dhcp, arp, icmp, ipv, llc.

2.1.1 Data processing

To facilitate more manageable data processing and expedite the experimental phase, we decided to reduce the dataset to 0.001% of its original size. As authors from [14] made same to their dataset divided it into smaller portions of 10 MB so that the conversion could be completed in parallel. Retaining a representative sample while drastically reducing processing cost was the aim of this subset selection. Because the smaller dataset retained its original data distribution, the models trained on it would be able to generalize effectively to the whole dataset.

We updated the attack column during the data processing stage in order to differentiate between attack and benign traffic. To be more precise, we classified traffic as "attack"(False) if it was any of the 33 sorts of attacks and as "benign"(True) if it wasn't. The dataset was successfully transformed into a binary classification issue consisting of two classes: attack traffic and benign

traffic. Working with a smaller subset allowed us to handle and analyze data more efficiently, which allowed us to concentrate on developing and evaluating the models within the limitations of the computational resources that were available. Because of the smaller dataset, testing and iterations of machine learning algorithms could occur more quickly, which expedited the entire research process.

2.1.2 Balancing reduced dataset

In the reduced version of the dataset, 1,093 cases are classified as True (indicating benign traffic) and 45,422 instances are labeled as False (showing bad traffic). This disparity represents a situation that occurs in real life, where malicious traffic predominates over benign traffic in a network that has been infiltrated. Machine learning model performance may suffer due to a large disparity between benign and attack occurrences. The majority class is typically favoured by most algorithms, which makes it difficult to identify the minority class. This imbalance in our dataset implies that models may find it difficult to correctly detect benign traffic because of its relative scarcity. In order to minimize the impact of this imbalance, we utilized an undersampling method. In order to produce a more balanced distribution, undersampling entails lowering the number of occurrences in the majority class (attack traffic). This facilitates the more efficient recognition of both classes by machine learning models. The undersampling is done using the 'RandomUnderSampler' from the 'imblearn' module. To match the number of occurrences in the minority class (benign traffic), the 'fit_resample' method randomly selects a subset of the majority class (attack traffic). As a result, there are an equal number of benign and attack cases in the new, balanced dataset. The results from [15] shows importance of balancing the dataset and it is crucial for several reasons:

Increased Generalization and Overall Performance: A balanced dataset makes sure that the model trains with equal attention to both classes, which improves generalization and overall performance.

Improved Evaluation Metrics: Since the model's performance on both classes is well represented, balanced datasets offer evaluation metrics that are reliable.

2.2 Model architecture

We explain our model's detailed architecture for network traffic analysis in this section. Data collection, feature engineering, model selection, training, testing, results evaluation, and model comparison are the main phases of our model. The figure 2.1 shows detailed view of model working process.

2.2.1 Feature Engineering

A crucial phase in getting the data ready for modeling is feature engineering. It involves choosing and modifying the dataset's most relevant attributes that have

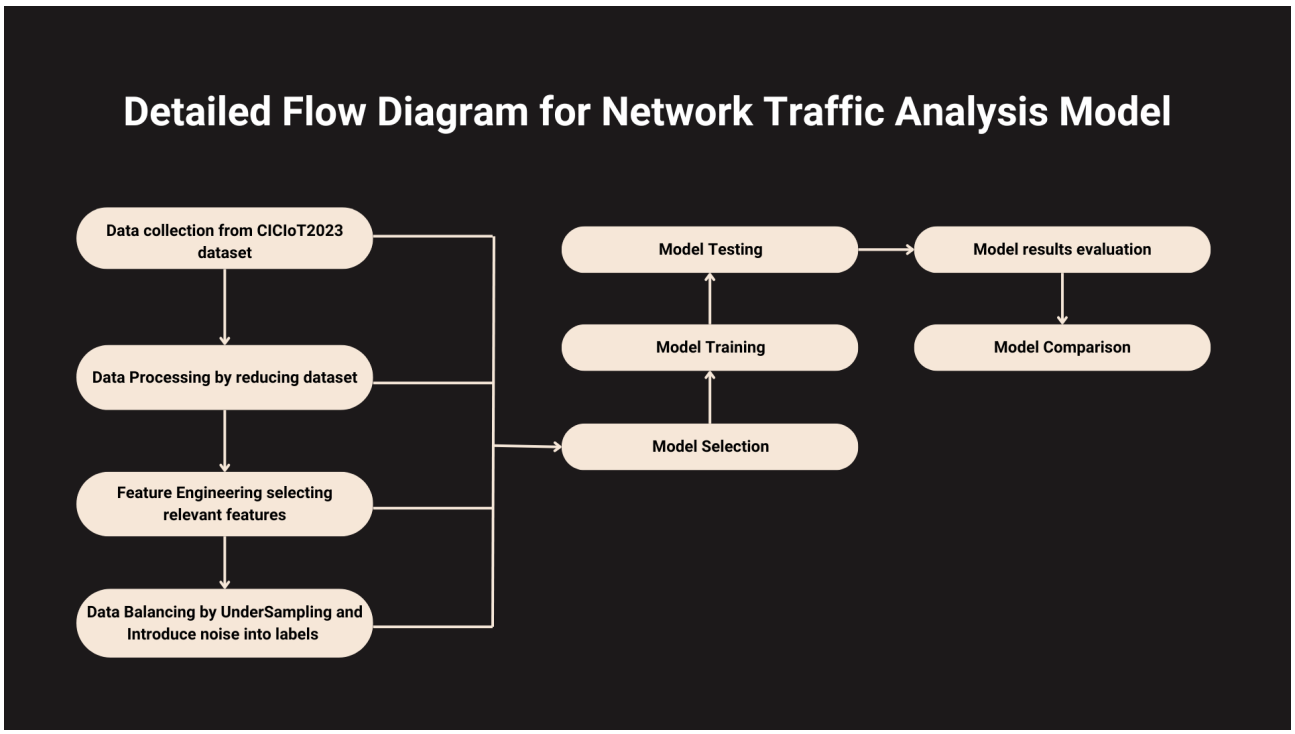

Figure 2.1 – Model architecture

a major impact on the model's performance. The CICIoT2023 dataset in our implementation has a number of protocol indicators, flags, and other features. In previous section 2.1 the protocols with boolean value was mentioned. To make categorical variables and boolean flags appropriate for machine learning methods, we encoded them. This stage made sure that every feature was numerical and that the models could use it to their maximum advantage.

In our work, we utilized Label Encoding for categorical variables, whereas the studies [10, 16] used One-Hot Encoding. In [16], categorical features like network protocols were transformed into binary vectors, resulting in 26 additional features. Similarly [10] encoded three categorical features (protocol type, type of service, and flag) into binary vectors and standardized them. Unlike One-Hot Encoding, which prevents ordinal relationships by creating multiple binary columns, Label Encoding assigns a unique integer to each category, which is computationally more efficient but can introduce unintended ordinal relationships.

Label Encoding is necessary for the dataset because it includes a categorical variable, protocol_type, which represents different network protocols, and machine learning algorithms require numerical input. To convert these categorical values into numbers, we use the LabelEncoder from the sklearn.preprocessing module, which assigns a unique integer to each category. For example, if the protocol_type column contains categories like HTTP, HTTPS, and DNS, the encoder might map them to 0, 1, and 2, respectively.

The dataset also contains several columns that represent flags and protocol

indicators as boolean values (True or False), such as fin_flag_number, syn_flag_number, rst_flag_number, and protocol indicators like http, https, dns, etc. Although boolean values are inherently binary, it's often good practice to explicitly convert them to integers (0 and 1) to ensure compatibility with machine learning algorithms. To achieve this, need to create boolean_columns, that includes all the columns with boolean values and then iterates over this list using a for loop. Inside the loop, each boolean column is converted to an integer type using the astype(int) method, transforming True values to 1 and False values to 0.

We make sure every feature in the dataset is numeric by going through these encoding processes, which is necessary for the majority of machine learning methods. This preprocessing stage guarantees that the data is in an appropriate format for additional analysis, which is essential for the effective training and assessment of our models.

2.2.2 Introducing Noise into labels

In the field of machine learning, adding noise to the training labels might be effective. This method can improve the model's robustness and help reproduce real-world situations where data may not always be accurately classified. Models can learn to avoid overfitting and improve their ability to generalize by being trained on slightly noisy data [17]. In machine learning, overfitting occurs when a model learns the training data—noise and outliers included—too well. As a result, the model performs exceptionally well on the training set but poorly when applied to new, untested data. When a model is overly complex, it is said to be overfitting and ends up capturing specifics of the training set instead of the underlying patterns.

We make the training environment more difficult for the model by intentionally flipping a tiny number of the labels. This method makes the model focus on the larger patterns in the data instead of learning individual instances by heart. In our method, we flipped 5% of the dataset's labels to introduce noise. This introduces an element of confusion to the data by indicating that some benign labels were converted to attack labels and vice versa.

The following steps are involved in the process:

Selecting the Noise Factor: We chose on a 5% noise factor [18] since it prevents overfitting while also adding just the right amount of noise to the dataset .

Choosing Loud Labels: To be flipped, a random selection of the labels—or 5% of the dataset—was chosen. The uniform distribution of noise throughout the dataset is provided through this randomness.

Label Flipping: The chosen labels were reversed, converting friendly traffic labels into dangerous ones and vice versa. There is noise introduced by this intentional mislabeling.

In addition, we normalized the data after adding noise to the labels. A critical stage in scaling the characteristics to have a mean of 0 and a standard deviation of 1 is standardization. This guarantees that every feature makes an equal contribution to the model's learning process and enhances the model's functionality.

2.2.3 Model Training

Fitting the chosen machine learning algorithms to the balanced and processed dataset was the process of model training. Our approach included a number of essential steps to guarantee the models' performance and endurance. An 80-20 split was used to divide the dataset into training and testing sets. This made it possible for us to train the models on an important portion of the data while keeping a separate set for performance evaluation. GridSearchCV was utilized to adjust the hyperparameters of models that include several parameters, including Random Forest and XGBoost. In order to maximize model performance, GridSearchCV thoroughly analyzes a given parameter grid to identify the ideal set of parameters. By ensuring that every model is set up with the best parameters possible, this method enhances generalizability and accuracy.

During the grid search process, we utilized cross-validation to further improve the durability of our models. In cross-validation 2.3.4, the training data is divided into numerous folds, and the model is trained on each fold while being validated on the remaining data.The optimal parameters found during hyperparameter tuning were used to train each algorithm on the training set. The strategy make sure that each algorithm had the highest chance of success.

By training multiple algorithms and performing thorough hyperparameter tuning, we can comprehensively evaluate and compare the performance of various models from table 3.2. The use of cross-validation and noise introduction in the labels enhances robustness, preventing overfitting and ensuring that the models generalize well to new data, which is crucial for real-world applications. Additionally, GridSearchCV and cross-validation optimize model performance by systematically exploring and validating hyperparameters, ensuring each model is fine-tuned for maximum accuracy and effectiveness.

2.2.4 Model evaluation metrics

The models were tested on a different test set to determine their performance after training. In order to make sure the models can generalize effectively to new, unseen data and are not overfitting to the training set, this phase is crucial. A range of performance metrics were employed to evaluate the models:

By finding out the percentage of correctly categorized examples among every instance, accuracy examines the model's overall correctness. It is a simple and easy-to-understand statistic that gives a brief summary of the model's

functionality. Understanding accuracy is key to understand the model's overall efficiency. Because it does not differentiate between the different kinds of errors committed, it might be confusing when there is a class difference. In unbalanced dataset, for example, a model that consistently predicts the majority class would have high accuracy but would miss attacks since benign traffic is considerably more common than attack traffic.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

Out of all the positive predictions the model generates, precision indicates the percentage of accurate positive predictions. The accuracy of the positive class predictions is the main focus, as this is important in situations where false positives (mislabeling benign data as attack traffic) can result in needless alarms and resource loss. A high precision means that the model is typically right when it expects an attack.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.2}$$

The amount of true positive cases that the model properly detected is measured by recall. It is especially important when there could be serious consequences if a true positive—an actual attack—is missed. A high recall rate means that the majority of actual attacks are effectively identified by the model. This is important to our application since network security may be affected if attacks escape undetected. On the other hand, a high recall could compromise precision and result in more false positives.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.3}$$

The harmonic mean of recall and precision is the F1 score. It gives a complete picture of the model's performance by presenting a single metric that achieves a balance between recall and precision. The F1 score combines the benefits of precision and recall, making it very helpful when working with imbalanced datasets. An high F1 score shows that the model effectively finds a balance between accurately detecting attacks and reducing false positives.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.4}$$

The ability of the model to identify between both positive and negative classes at various threshold values is measured using AUC-ROC. It is a reliable indicator of the model's overall effectiveness. The area under the ROC curve

(AUC) is the true positive rate (recall) plotted against the false positive rate at different threshold values. A high AUC-ROC value means that the model is able to differentiate between attack and benign traffic. This statistic is particularly useful for evaluating the performance of various models because it takes into account all potential categorization criteria. The plotted AUC-ROC results can be seen from figure 3.1

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP + TN} \tag{2.5}$$

All results of each metric for each model can be seen in the following table 3.1. This comprehensive evaluation helps us to understand the trade-offs and strengths of each algorithm in the context of network traffic analysis.

2.3 Machine learning algorithms

Several machine learning algorithms were used in our study to categorize network traffic data. Due to their unique characteristics, benefits, and limitations, each method can be used for any kind of categorization problems. The algorithms selected for this research are classification algorithms, which are intended to classify data into predetermined groups. An overview of each algorithm is given below, along with information on its training procedure, results in our model, and the reasons behind our selection of it.

2.3.1 Probabilistic and Linear

Naive Bayes is a probabilistic classifier that depends on feature independence and is based on the Bayes theorem. The Gaussian Naive Bayes variant that we employed is predicated on the idea that the features have a Gaussian distribution.

Large datasets may profit from the simplicity and computational efficiency of naive bayes. It works especially well for binary and multi-class classification issues and does well with high-dimensional data. The probability of the features given each class and the prior probabilities of the classes are computed using the algorithm. It uses the training data to estimate these probabilities during training. Naive Bayes provides an opportunity for comparison with complex structures in our model. Because it was so simple, its performance was unsatisfactory.

Logistic function is used in logistic regression, a linear model for binary classification, to represent the probabilities of the default class. We choose it because it produces probabilistic results and is easy to understand and apply to binary classification situations. Its benefits include being easy to understand and apply, offering probability estimates, and working well with big datasets. It can, however, have trouble with complex interactions in the data and assumes a linear relationship between the target's log chances and attributes. In the training process, gradient descent or other optimization techniques are frequently used to estimate parameters that maximize the likelihood of the observed data.

2.3.2 Tree-Based

Decision tree is a non-parametric supervised learning technique that divides the data into subsets according to the input feature values. It is used for regression and classification. We choose it because Decision Trees can handle both numerical and categorical data, require no data preprocessing, and are simple to analyze, visualize, and comprehend. Simpleness, little data preparation, and adaptability to various data kinds are among the benefits. If improperly removed, they can become overfitting and result in biased trees if certain classes take precedence. Recursively dividing the training data into subsets according to the feature that offers the most information gain or the lowest Gini impurity is the training process.

During training, the Random Forest ensemble learning approach builds numerous decision trees and outputs the mode of the classes of each individual tree. We choose it because Random Forest offers feature importance scores, handles both numerical and categorical data, and is resistant to overfitting. Because it averages over several trees, it has great accuracy, is resistant to overfitting, and can handle huge datasets and high-dimensional data. It is less interpretable than single decision trees and requires a lot of computation and time. Using various subsets of the training data and features, several decision trees are created during the training process. To improve generalization, feature randomness and bootstrapping are used.

The complex gradient boosting technique known as XGBoost (Extreme Gradient Boosting) is highly scalable and efficient. We selected it because of XGBoost's exceptional efficiency, scalability, and ability to manage big datasets with intricate interactions. High accuracy and efficiency, reliable management of outliers and missing data, and the ability to provide feature relevance rankings are some of its benefits. However, it can be extremely computationally complex, and requires careful parameter optimization. Building trees one after the other, optimizing for a differentiable loss function, and using regularization to reduce overfitting and enhance generalization are all part of the training process. XGBoost was one of the top-performing algorithms in our model, exhibiting excellent accuracy and resilience.

Adaptive Boosting, or AdaBoost, is an ensemble learning technique that builds a powerful classifier by combining several weak classifiers. We choose it because AdaBoost is adaptive, focused on more difficult-to-classify cases in later iterations, and can dramatically improve the performance of poor classifiers. Its benefits include the capacity to turn inexperienced students into proficient ones, flexibility in challenging situations, and a decrease in bias and variance. It can, however, be difficult to compute for large datasets and is susceptible to noisy data and outliers. Weak classifiers are iteratively trained, instances of wrong classifications are weighted appropriately, and the final model is built as a weighted sum of these

weak classifiers. AdaBoost performed well in our model, especially when paired with decision trees as the foundation

2.3.3 Instance-Based

A fundamental instance-based learning technique called KNN (K-Nearest Neighbors) classifies a sample according to the majority class of its k-nearest neighbors. We choose it because authors from [19] identified the powerfull use against noisy training data, and efficient—especially for smaller datasets with clearly defined class boundaries, because of this authors [19] called it lazy learning algorithm. Its benefits include being efficient with tiny datasets, easy to apply, and not requiring a training step. It may, however, perform poorly with high-dimensional data, be sensitive to the choice of k and feature scaling, and be computationally expensive at prediction time. KNN chooses the majority class among the k-nearest neighbors by calculating the distance between each training instance and the test instance. KNN offered competitive accuracy in our model, but in order to maximize performance, accurate feature scaling and k parameter adjustments were required.

2.3.4 Cross-Validation in ML algorithms

One effective method for regularly evaluating the performance of machine learning models is cross-validation. To make sure that our models perform effectively when applied to new data, we employed k-fold cross-validation, more precisely 5-fold cross-validation. Using this strategy, the dataset is divided into k folds, or subsets, and the model is trained on k-1 subsets before being validated on the remaining subset. Every subset serves as the validation set once during the k iterations of this process.

Reduces Overfitting Risk: Cross-validation lowers the chance that the model may overfit to a specific subset of the data by employing different subsets for training and validation.

Performance Stability: By averaging the findings over several folds, it offers a more accurate estimation of the model's performance and makes sure that it is independent of a specific train-test split.

Robust Evaluation: This is important for real-world applications since it helps determine how effectively the model generalizes to new, unseen data.

Additionally, it helps to achieve a balance between variance and bias, which improves generalization. The disadvantages include the need for careful implementation and maintenance of data splits and training procedures, as well as the fact that it is computationally complex and time-consuming, particularly when dealing with large datasets and advanced models. Our approach, which incorporates cross-validation, has enabled us to thoroughly assess and optimize our machine learning models, guaranteeing their accuracy and endurance.

3 Result and Findings

3.1 Histograms of metrics

Evaluating the effectiveness of machine learning algorithms through various metrics provides crucial insights into their performance. Our study focuses on Accuracy 3.2, Precision 3.3, Recall 3.4, F1 Score 3.5, and AUC-ROC 3.6 to comprehensively evaluate the models. Here, we will discuss the results of these metrics in the context of our chosen models and what they reveal about their performance.

Table 3.1 – Model Metrics

Model	Accuracy	Precision	Recall	F1 Score	AUC-ROC
Naive Bayes	0.86	0.92	0.77	0.84	0.85
Random Forest	0.93	0.91	0.95	0.93	0.93
Decision tree	0.92	0.91	0.92	0.92	0.92
AdaBoost	0.93	0.92	0.94	0.93	0.93
KNN	0.93	0.91	0.95	0.93	0.93
Logistic Regression	0.93	0.91	0.95	0.93	0.93
XGBoost	0.93	0.91	0.95	0.93	0.93

3.1.1 Naive Bayes

Out of all the models examined, the Naive Bayes method had the lowest accuracy of 0.86. The Naive Bayes classifier's high independence assumptions, which could not hold true for network traffic data, are the cause of its comparatively low accuracy. The model's accuracy of 0.92 suggests that it is likely correct when it predicts a traffic anomaly because it is quite good at distinguishing true positives among the positive predictions. The recall is 0.77, though, which means that a sizable percentage of true abnormalities are missed. The F1 Score of 0.84, while less than other models, shows a balance between recall and precision. Although the AUC-ROC of 0.85 is respectable, it indicates that the model's ability to differentiate between classes is not as strong as it may be. The primary benefits of Naive Bayes include its speed and simplicity; however, its performance in complicated datasets may be limited by the assumption of feature independence.

3.1.2 Random Forest

Random Forest performed exceptionally well, with an accuracy of 0.93. The precision and recall are 0.91 and 0.95, respectively, indicating a well-balanced model that is both precise and sensitive to actual anomalies. The F1 Score of 0.93 corroborates this balance. The AUC-ROC of 0.93 signifies excellent discriminative

ability. Random Forest's strength lies in its ability to handle large datasets with higher dimensionality and its robustness to overfitting due to ensemble learning. However, its complexity and longer training time can be considered disadvantages, particularly with very large datasets.

3.1.3 Decision Tree

The Decision Tree algorithm also performed well with an accuracy of 0.92. Both precision and recall are high at 0.91 and 0.92, respectively, resulting in an F1 Score of 0.92. The AUC-ROC of 0.92 indicates strong discriminative performance. Decision Trees are easy to interpret and visualize, which is advantageous for understanding model decisions. However, they can easily overfit to the training data, especially if not pruned properly, which can be a disadvantage.

3.1.4 AdaBoost

AdaBoost matched the top-performing models with an accuracy of 0.93. It showed a high precision of 0.92 and a recall of 0.94, leading to an F1 Score of 0.93. The AUC-ROC of 0.93 confirms its strong ability to differentiate between classes. AdaBoost's advantage is its ability to convert weak learners into strong ones by focusing on misclassified instances. However, it is sensitive to noisy data and outliers, which can degrade its performance.

3.1.5 K-Nearest Neighbors (KNN)

KNN also achieved an accuracy of 0.93. Its precision and recall are 0.91 and 0.95, respectively, resulting in an F1 Score of 0.93. The AUC-ROC is 0.93, indicating excellent performance. KNN's simplicity and effectiveness in small datasets are significant advantages. However, its computational cost becomes prohibitive as the dataset size grows, making it less suitable for large-scale applications.

3.1.6 Logistic Regression

Logistic Regression performed on par with an accuracy of 0.93. It had a precision of 0.91 and a recall of 0.95, leading to an F1 Score of 0.93. The AUC-ROC of 0.93 indicates robust performance. Logistic Regression is advantageous due to its simplicity and interpretability. However, it may struggle with complex, non-linear relationships in the data, which can limit its performance compared to more complex models.

3.1.7 XGBoost

XGBoost matched the performance of other top models with an accuracy of 0.93. It showed a precision of 0.91 and a recall of 0.95, leading to an F1 Score of 0.93. The AUC-ROC of 0.93 confirms its excellent discriminative ability. XGBoost's strength lies in its efficiency and performance in handling large-scale datasets with missing values. Its main disadvantage is the complexity and

computational intensity, which can be a challenge for real-time applications.

3.1.8 ROC and AUC Plots

The ROC and AUC plots 3.1 provide a visual representation of the trade-offs between true positive rates and false positive rates for different algorithms.

Naive Bayes: The ROC curve for Naive Bayes is less steep, with a more gradual slope, indicating a lower AUC and poorer overall performance in distinguishing between classes.

Random Forest, AdaBoost, KNN, Logistic Regression, and XGBoost: These algorithms have ROC curves that rise sharply towards the top left corner, indicating high true positive rates and low false positive rates, which corresponds to their high AUC values. This sharp rise demonstrates their superior ability to correctly classify both positive and negative instances across various thresholds.

Decision Tree: The ROC curve for Decision Tree also shows a strong performance, with a steep rise similar to the other top-performing models, indicating a high true positive rate and effective discrimination between classes.

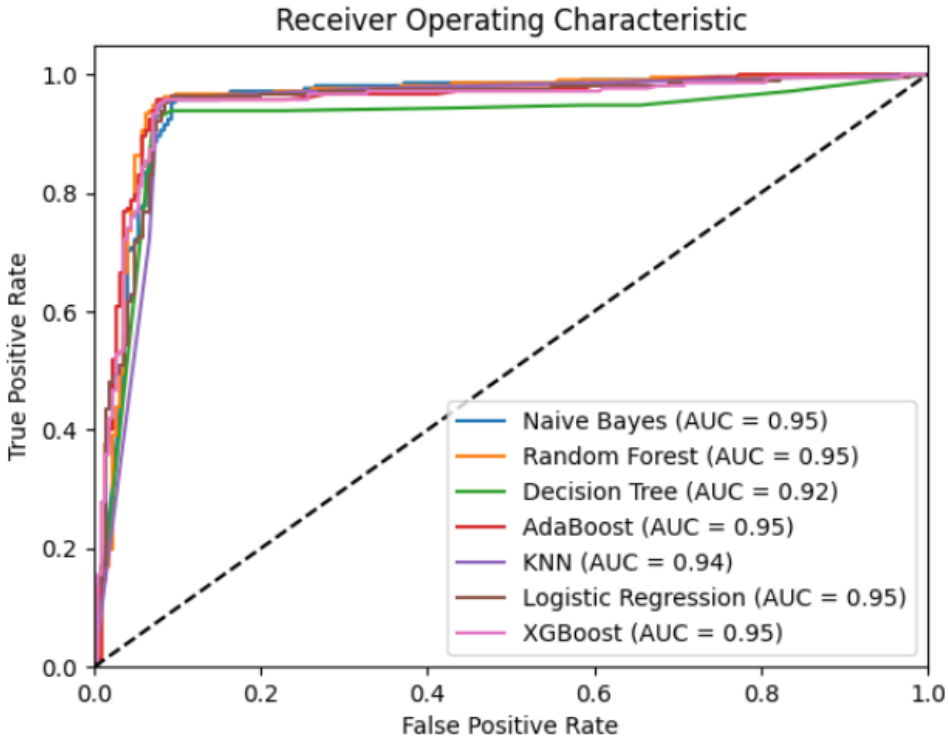

Figure 3.1 – AUC-ROC of each model plotted

These methods are particularly effective for network traffic analysis due to their ability to handle complex data patterns and maintain high precision and recall. Naive Bayes, despite its simplicity and high precision, falls short in recall and overall accuracy. Decision Trees provide a good balance but may require careful tuning to avoid overfitting.

The ROC and AUC plots further corroborate these findings, visually demonstrating the superior discriminative capabilities of the top-performing models. Overall, the use of ensemble methods and boosting techniques has proven to be highly effective for network traffic analysis, offering robust performance and reliability in detecting network anomalies. Ensemble methods was perfectly shown in the works of the authors [20].

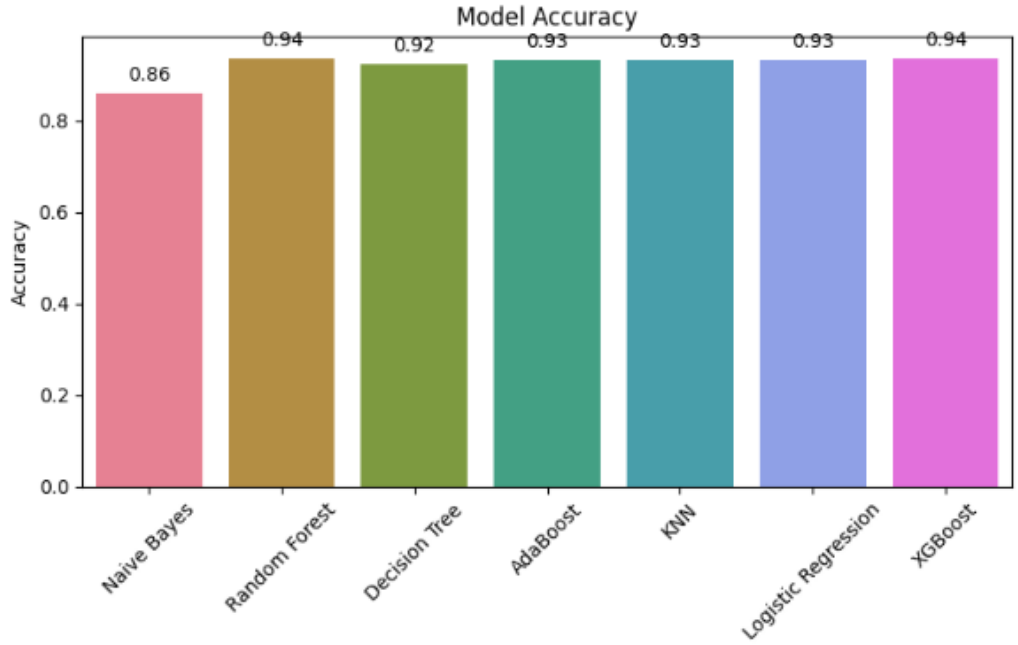

Figure 3.2 – Accuracy barplots

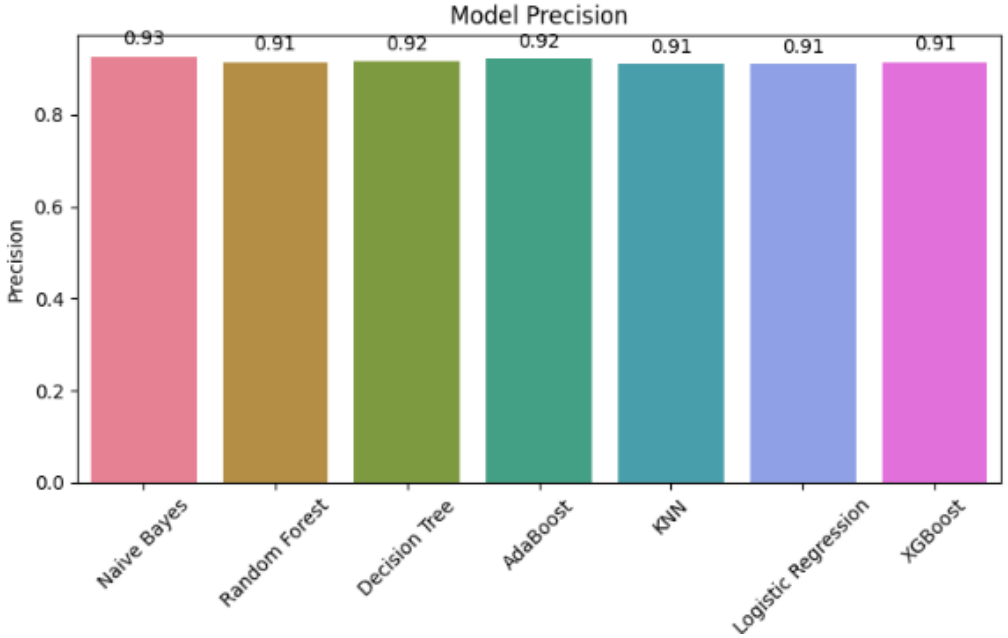

Figure 3.3 – Precission barplots

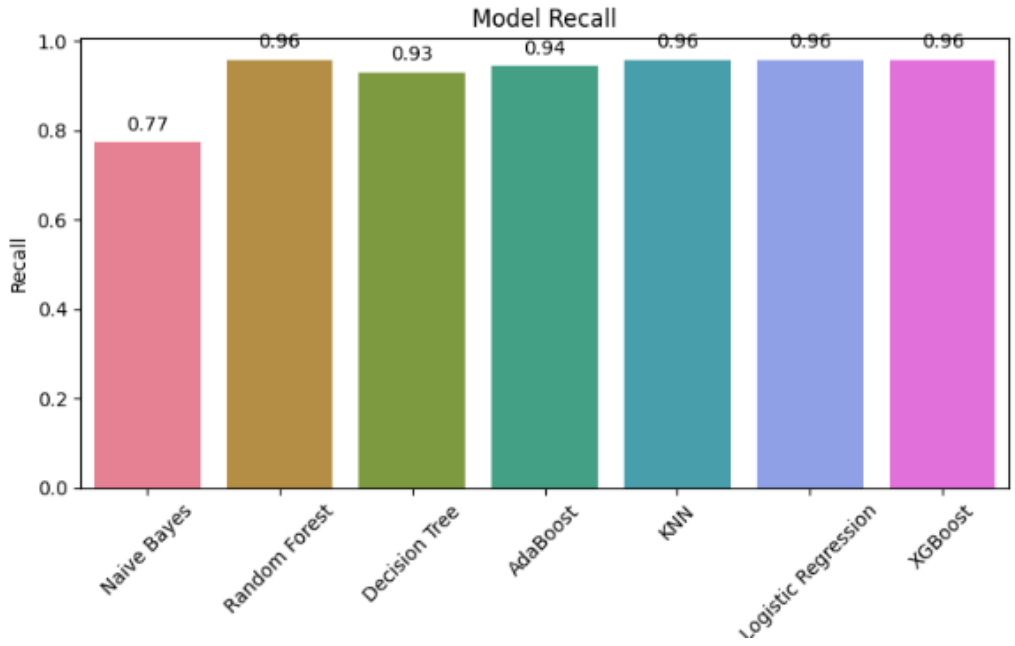

Figure 3.4 – Recall barplots

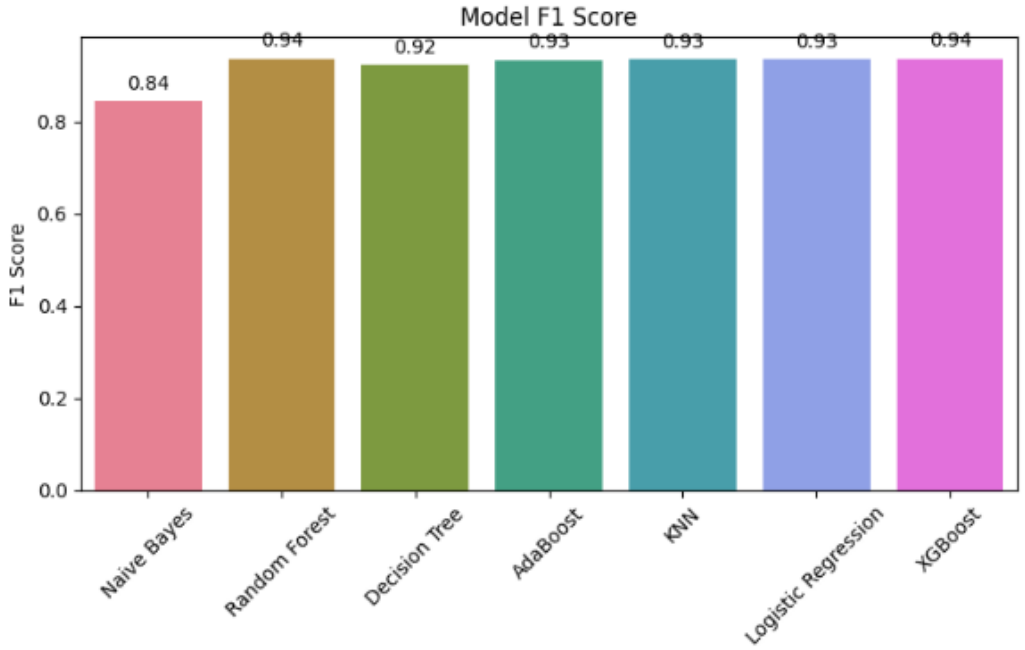

Figure 3.5 – F1 Score barplots

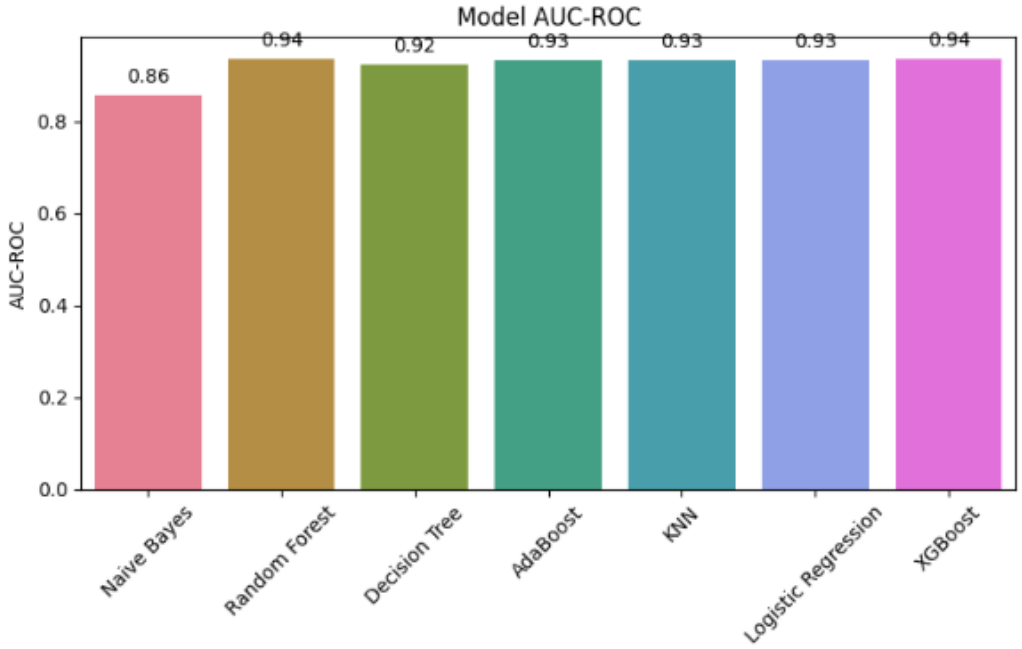

Figure 3.6 – AUC-ROC barplots

3.2 Model training and testing results

To determine how well a machine learning model generalizes to new data, it is essential to compare test and train accuracy. We note the following train and test accuracy for several models in our study, table 3.2.

It is generally expected that the train accuracy might be slightly higher than the test accuracy because the model is evaluated on the data it was trained on. However, significant differences between train and test accuracy can indicate overfitting or underfitting:

Overfitting: When the model performs exceptionally well on the training data but poorly on the test data. This means the model has learned the noise and details in the training data to the extent that it negatively impacts its performance on new data.

Underfitting: When the model performs poorly on both the training and test data, indicating that it is too simple to capture the underlying patterns in the data.

Naive Bayes: Interestingly, the test accuracy (0.86) is slightly higher than the train accuracy (0.84). This anomaly can occur due to several reasons, such as the model benefiting from the specific characteristics of the test data or due to regularization effects that prevent overfitting. It suggests that Naive Bayes, despite its assumptions, manages to generalize well in this context.

Random Forest, AdaBoost, XGBoost, KNN, and Logistic Regression: These models have very close train and test accuracies, with the train accuracy being marginally higher. This is indicative of a good balance between bias and variance, showing that these models generalize well and are not overfitted. The

Table 3.2 – Train and test accuracy

Model	Train accuracy	Test accuracy
Naive Bayes	0.84	0.86
Random Forest	0.95	0.93
Decision tree	0.94	0.92
AdaBoost	0.94	0.93
XGBoost	0.94	0.93
KNN	0.94	0.93
Logistic Regression	0.93	0.93

high accuracy of these models on both training and test data (around 0.93-0.95) indicates their robustness and ability to capture the underlying patterns effectively.

Decision Tree: The slight drop from train accuracy (0.94) to test accuracy (0.92) is typical for decision trees, which are prone to overfitting. However, the decrease is not severe, suggesting that the tree is relatively well-pruned and balanced.

Balancing train and test accuracy 3.7 is crucial in ensuring that the model performs well on unseen data, which is the primary goal of machine learning. High train accuracy with low test accuracy is undesirable as it indicates overfitting, meaning the model may not perform well in real-world scenarios.

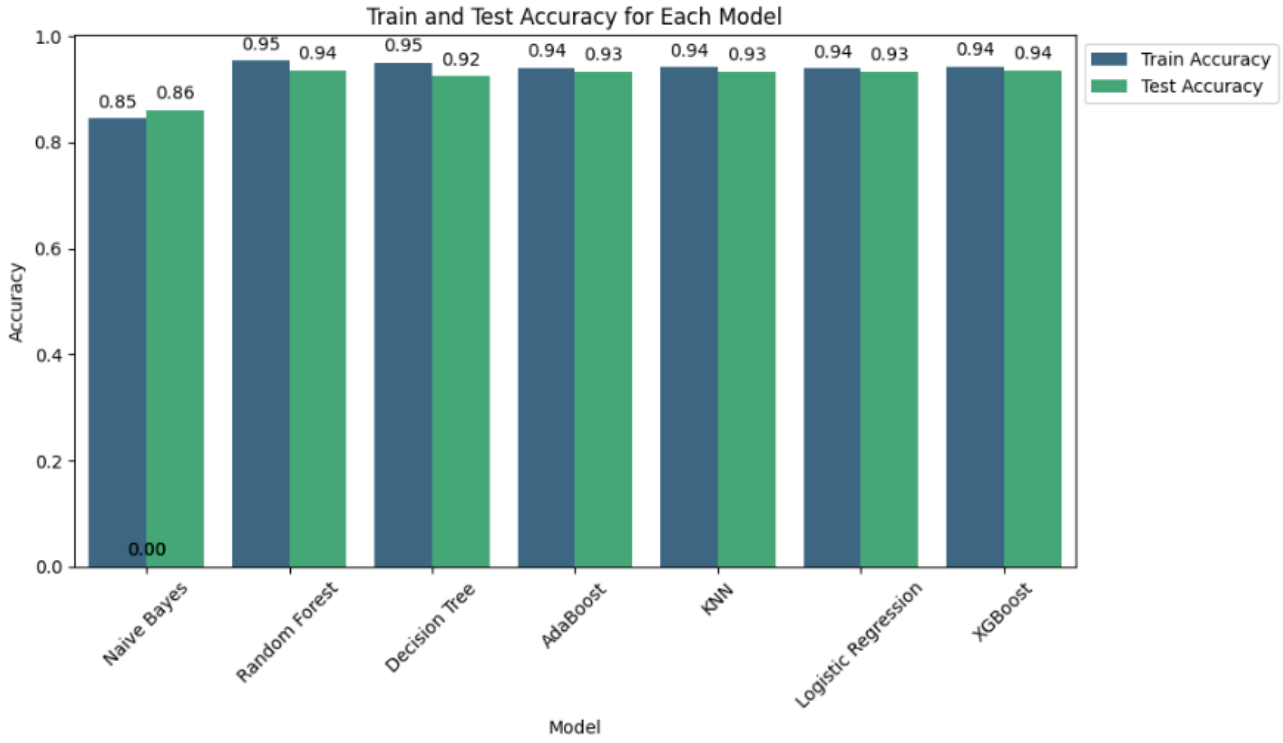

Figure 3.7 – Train and Test accuracy comparison result

Models with high test accuracy, close to their train accuracy, are preferable as they demonstrate that the model generalizes well to new, unseen data. In our results, Random Forest, AdaBoost, XGBoost, KNN, and Logistic Regression showcase this desirable trait. The choice of models like Random Forest, XGBoost, and AdaBoost is validated by their high and balanced accuracy scores, which are crucial for tasks such as network traffic analysis where generalization to new data is critical.

The slight variation between train and test accuracy in models like Random Forest and XGBoost indicates their robustness and effective handling of both the training and test datasets. These models are known for their ability to manage overfitting through ensemble methods and regularization techniques. The preprocessing steps, including balancing the dataset through undersampling, have likely contributed to the high and balanced performance of the models. This preprocessing ensures that the models are trained on representative samples, thereby improving their generalization capabilities. The close alignment of train and test accuracies across most models highlights their robustness and suitability for the network traffic analysis task. The slight deviations observed provide insights into the models' behaviors and affirm the effectiveness of our chosen preprocessing and modeling approaches.

3.3 Confusion matrix

A confusion matrix is a crucial tool for evaluating the performance of classification algorithms. It provides a detailed breakdown of the true positive (TP), false positive (FP), true negative (TN), and false negative (FN) predictions. Here's what each term represents:

- True Positives (TP): The number of correctly predicted positive instances.
- False Positives (FP): The number of instances incorrectly predicted as positive.
- True Negatives (TN): The number of correctly predicted negative instances.
- False Negatives (FN): The number of instances incorrectly predicted as negative.

The confusion matrix helps to identify class imbalance by showing how well the model is performing across different classes, revealing any potential imbalance. It also aids in evaluating model performance as metrics like precision, recall, F1 score, and accuracy are derived from it. Additionally, it helps understand errors by highlighting specific areas where the model is making mistakes, which can be crucial for improving model performance.

3.3.1 Analysis of each algorithms confusion matrix

Let's delve into the confusion matrix results for each algorithm and analyze why false positives and false negatives are almost equal across most algorithms,

and why Naive Bayes has significantly different false negatives.

Naive Bayes 3.8 has a relatively high number of false negatives (48), which suggests that it is often predicting negative when the actual class is positive. This could be due to the strong assumptions Naive Bayes makes about feature independence, which might not hold true for our dataset. This model tends to perform well when the features are truly independent, but any correlation between features can degrade its performance, leading to higher false negatives.

Random Forest 3.9 shows a balanced confusion matrix with low false positives and false negatives. This indicates that the model is well-tuned and performs effectively on both classes. The ensemble nature of Random Forest, which combines multiple decision trees, helps in reducing variance and overfitting, leading to more accurate predictions.

The Decision Tree 3.10 model also shows a balanced performance but slightly higher false negatives compared to Random Forest. Decision Trees can be prone to overfitting if not properly pruned, but in this case, the model seems to handle the data well, though not as robustly as Random Forest.

AdaBoost 3.11 also performs well, maintaining a good balance between false positives and false negatives. AdaBoost works by combining weak learners to form a strong classifier, focusing on errors of previous learners, which helps in improving accuracy over iterations. This iterative correction of errors helps in maintaining a low error rate.

XGBoost 3.12, like Random Forest, shows excellent performance with very low false negatives and false positives. XGBoost's strength lies in its ability to handle a wide range of data types and its robustness to overfitting through regularization. This makes it particularly effective in producing balanced and accurate predictions.

KNN 3.13 shows low false negatives and slightly higher false positives. KNN is a simple yet effective instance-based learning algorithm that performs well with sufficient representative data. However, its performance can degrade with noisy or irrelevant features, which might explain the slight increase in false positives.

Logistic Regression 3.14 demonstrates balanced performance with low false positives and false negatives. This model benefits from being a linear model, which works well with large datasets and is less prone to overfitting due to regularization techniques.

3.3.2 Explanation of Equal False Positives and False Negatives

The near equality of false positives and false negatives across most models suggests that the dataset is well-balanced and the models are trained effectively. This balance indicates that the models are not biased towards one class over the other, which is critical in applications like network traffic analysis where both false positives and false negatives can have significant implications.

However, the significantly higher false negatives in Naive Bayes point to its limitations with this particular dataset, likely due to its assumption of feature independence which might not hold true, causing it to misclassify positive instances as negative more often.

Advantages: The balanced false positive and false negative rates across most models indicate robustness and reliability, making these models suitable for real-world applications where both types of errors need to be minimized.

Disadvantages: The high false negatives in Naive Bayes highlight its limitations and suggest that it may not be the best choice for this dataset. This reinforces the importance of model selection based on the specific characteristics of the dataset.

Upon introducing noise into the training labels, notable variations in the confusion matrix results were observed across different classification algorithms. While some models exhibited a robust response to noisy data, maintaining relatively stable performance metrics, others displayed heightened sensitivity, resulting in discernible shifts in classification accuracy.

For instance, the Naive Bayes algorithm demonstrated a considerable increase in false negatives following the introduction of noisy labels. This suggests that the model's inherent assumption of feature independence may lead to misclassifications, particularly in the presence of correlated features within the dataset. Conversely, models such as Random Forest and XGBoost exhibited a more resilient response to noisy labels, with minimal fluctuations in false positives and false negatives. This resilience can be attributed to the ensemble nature of these algorithms, which leverage multiple decision trees to mitigate the impact of noise and improve overall predictive accuracy.

Notably, models such as Logistic Regression and KNN displayed a higher susceptibility to noise, evidenced by pronounced fluctuations in classification accuracy metrics. In contrast, Decision Tree and AdaBoost exhibited a more robust response, maintaining relatively stable performance even in the presence of noisy data.

Analysis of specific metrics further elucidated the sensitivity of models to noise. For instance, Naive Bayes experienced a notable decrease in recall, indicating its tendency to miss positive instances in the presence of noisy labels. Conversely, Random Forest and XGBoost demonstrated consistent performance across all metrics, underscoring their resilience to noise and suitability for real-world applications where data quality may vary.

Models that exhibit consistent performance despite the presence of noise are deemed more reliable and better equipped to handle real-world scenarios. However, the observed variations in model sensitivity highlight the importance of robust evaluation frameworks and ongoing monitoring to ensure the reliability of predictions in dynamic environments.

In particular, the reliability of predictions varied across different algorithms, with ensemble methods such as Random Forest and AdaBoost demonstrating enhanced stability in the face of noise. These models leverage the collective wisdom of multiple weak learners to mitigate the impact of individual errors, resulting in more reliable predictions overall. Conversely, models with inherent assumptions or simplistic architectures may be more susceptible to noise, necessitating careful consideration in their deployment.

The confusion matrix provides valuable insights into the performance of our models, helping us understand their strengths and weaknesses in classifying network traffic. The balanced results across most models affirm the effectiveness of our preprocessing and modeling approach, while the disparities highlight areas for potential improvement.

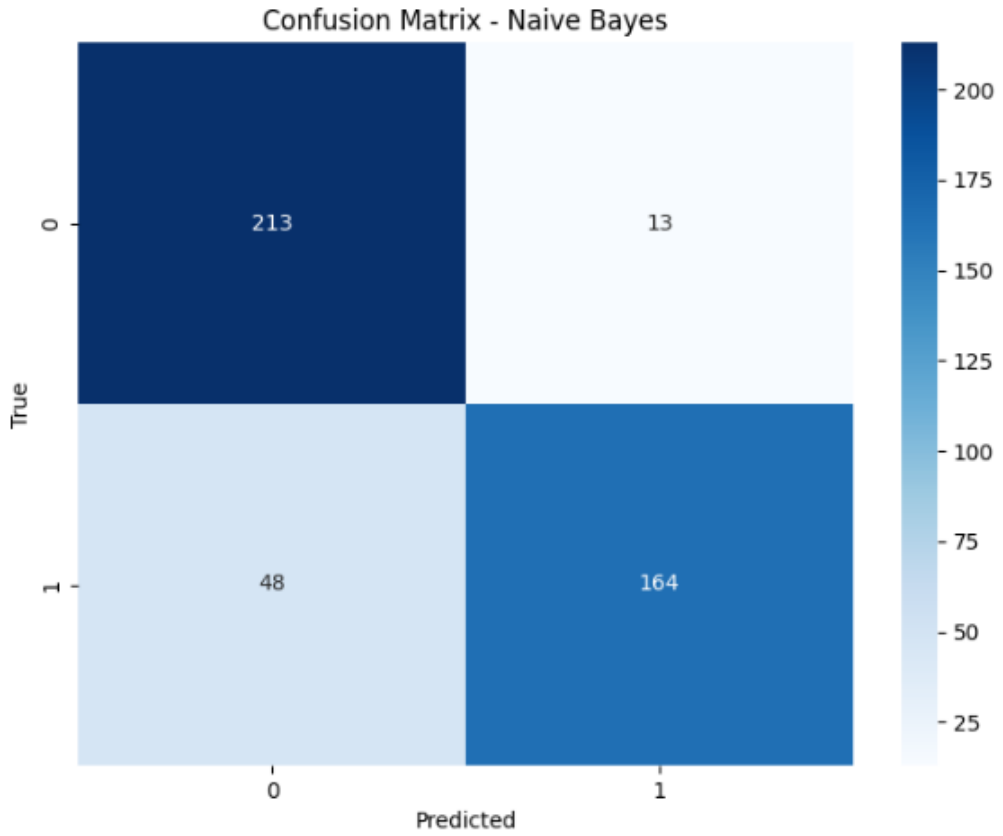

Figure 3.8 – Confusion matrix for Naive bayes

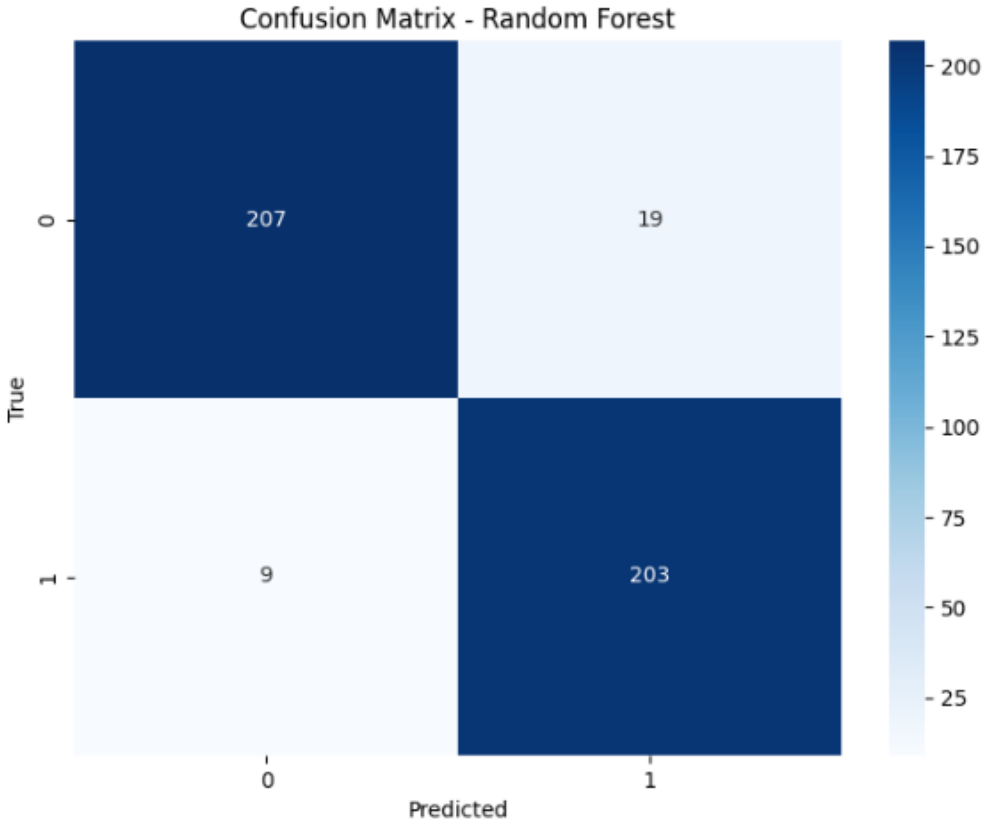

Figure 3.9 – Confusion matrix for RF

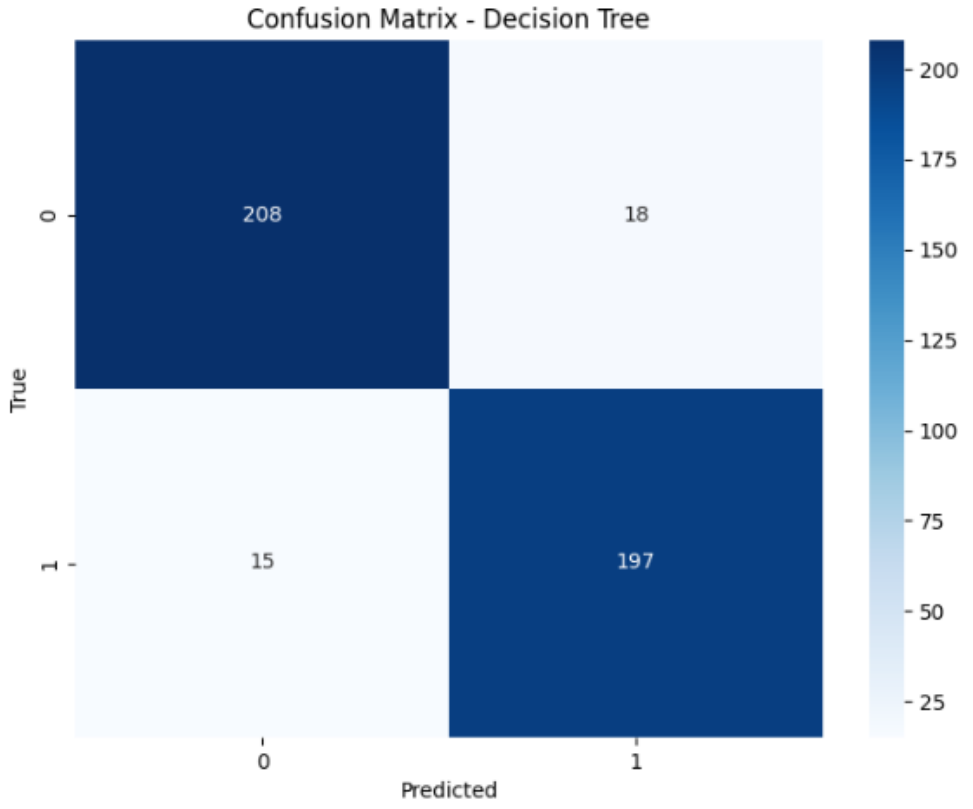

Figure 3.10 – Confusion matrix for DT

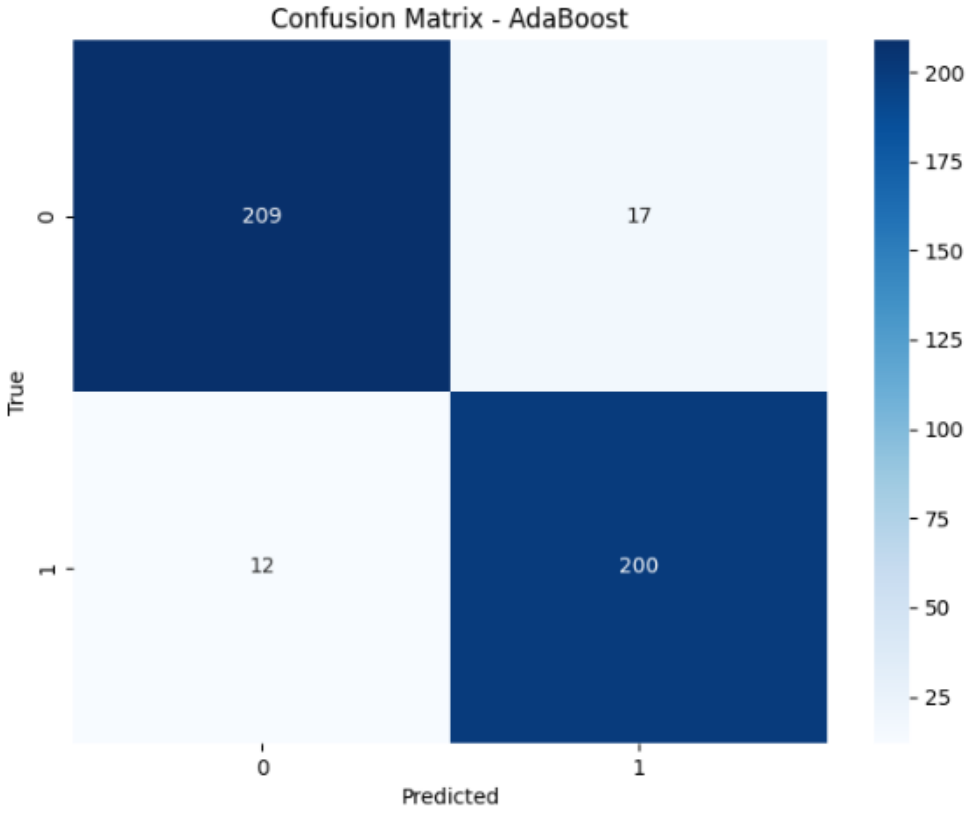

Figure 3.11 – Confusion matrix for AdaBoost

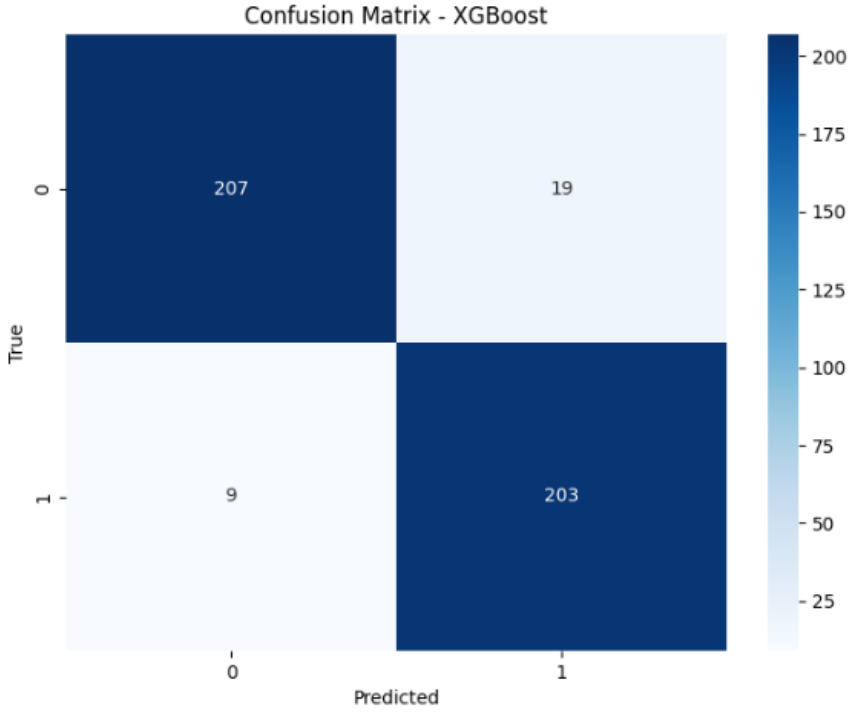

Figure 3.12 – Confusion matrix for XGB

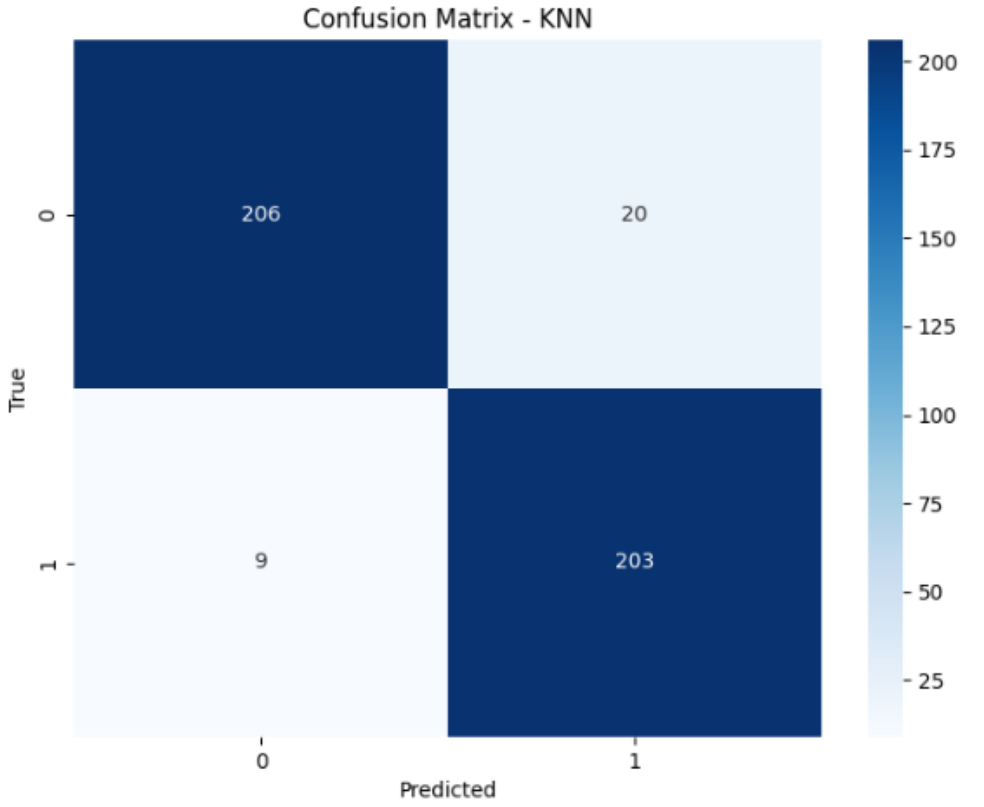

Figure 3.13 – Confusion matrix for KNN

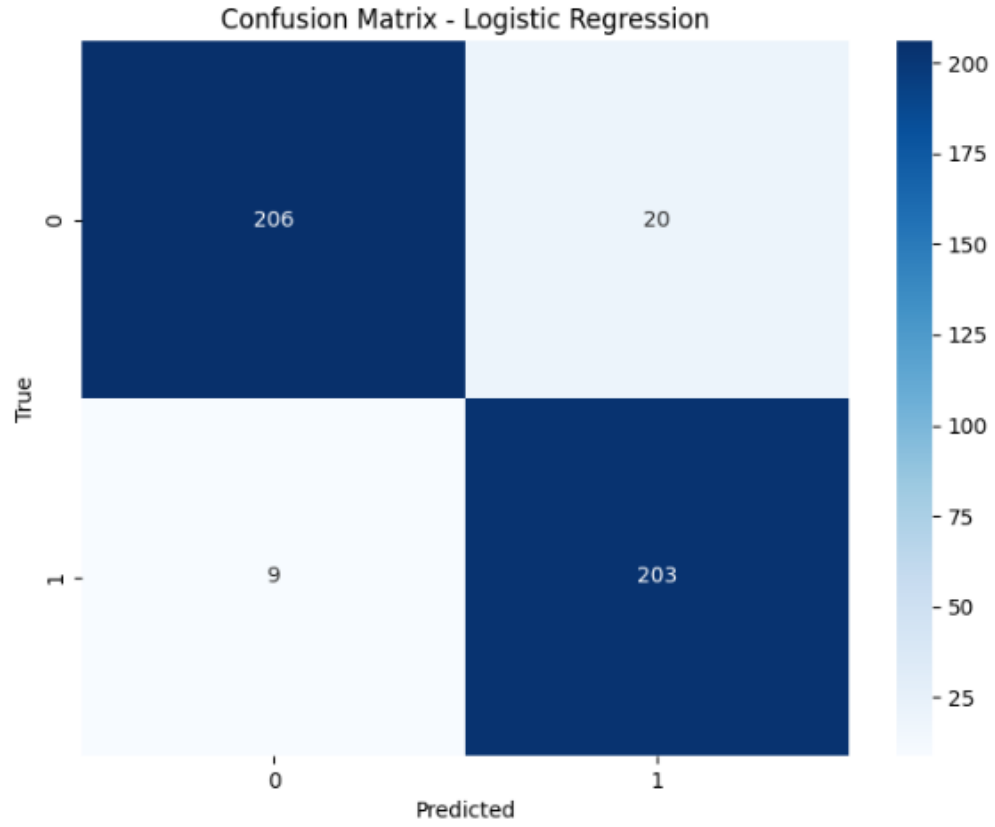

Figure 3.14 – Confusion matrix for LR

4 Discussion

The use of machine learning (ML) algorithms in network traffic analysis (NTA) has proved to be very useful in improving how anomalies and threats are detected and characterized by the network. This section highlights the results of our research, compares the performance of the individual ML algorithms in the analysis of network traffic, and expounds the implication of the results arrived at.

4.1 Comparative Performance of Machine Learning Models

Our research experimented with a variety of ML models, including Naive Bayes, Random Forest, Decision Tree, AdaBoost, XGBoost, K-Nearest Neighbors (KNN), and Logistic Regression, to classify network traffic. The findings revealed that the choice of the model plays an important role in the classification accuracy and robustness of traffic flows.

Naive Bayes: It provided a train and test accuracy of 86%, which is comparatively lower than the other models. Although Naive Bayes is computationally efficient and works well for large datasets, its assumption of feature independence is often not true for network traffic data, leading to poor performance.

Random Forest performed exceptionally well, achieving a train accuracy of 95% and a test accuracy of 93%. This model is known for its robustness in handling large datasets with high dimensionality and for its ability to reduce overfitting due to its ensemble nature. The slight decrease in accuracy from training to testing data suggests that the model generalizes well to unseen data.

The Decision Tree algorithm also demonstrated strong performance, with a train accuracy of 94% and a test accuracy of 92%. This slight drop is typical of decision trees, which are prone to overfitting but can be mitigated through proper pruning. Decision Trees are advantageous for their interpretability and ease of visualization, making it easier to understand model decisions. However, the relatively small decrease in accuracy indicates that the model was well-pruned and balanced, making it suitable for analyzing complex network traffic data that involve non-linear relationships and feature interactions .

AdaBoost and XGBoost: These ensemble methods also provided good performance, with test accuracies around 93-94%. The ability to combine multiple weak classifiers into a strong one helps in capturing more subtle patterns in the data, such as in network traffic data.

KNN: This model provided a test accuracy of 93%, which is a little lower compared to the performance of the tree-based methods and boosting algorithms. Its performance is sensitive to the choice of the number of neighbors and the distance metric, and it may not scale well for very large datasets.

Logistic Regression: Surprisingly, Logistic Regression gave the best test

accuracy (93%) compared to all other models used above, which suggests that it can be very powerful for binary classification problems as in network traffic analysis, especially when the feature-target relationship is approximately linear.

4.2 Implications of Results

Model Selection and Application. Tree-based models and ensemble methods, such as Random Forest, AdaBoost, and XGBoost, are highly recommended for network traffic analysis due to their robustness and high accuracy. These models excel in capturing intricate patterns within the data, making them particularly suitable for identifying subtle anomalies in network traffic. On the other hand, Logistic Regression can be a strong candidate in scenarios where computational efficiency is crucial. This model is effective for problems where the relationships in the data are predominantly linear, providing a good balance between simplicity and performance, which is especially valuable in resource-constrained environments.

Dataset Characteristics. The quality and size of the dataset play a significant role in the performance of ML models. Larger datasets with high-quality, relevant features tend to enhance model performance by providing more information for the learning process. However, the marginal benefits of increasing dataset size diminish beyond a certain point, highlighting the importance of quality over sheer volume. Proper preprocessing steps, such as feature engineering, balancing the dataset to address class imbalances, and introducing noise to mitigate overfitting, are critical for achieving optimal model performance. Feature engineering, in particular, can uncover latent structures in the data, thereby improving the predictive power of the models.

Adaptability in the face of new threats. Cyber threats do not remain the same; they constantly change. The ML models should also be adapted by regular changes in the threat landscapes. This is through continuous updating and retraining to ensure they can detect new types of attacks. Cyber threats are, by design, dynamic; hence, static models become outdated and accept only a short period of attacks. Therefore, mechanisms for automated retraining with the latest data should be implemented. This might make use of online learning techniques themselves, in that the models are updated with new data to keep them current and accurate. The feedback loops from cybersecurity experts help better retrain the model, aiding it in recognizing and reacting to growing threats. Continuous adaptation makes the network resilient not only to known threats but also to deal with attack vectors not previously seen.

Scalability and Real-time Analysis. Models like Random Forest and boosting algorithms, while accurate, can be computationally intensive and may require significant resources for real-time traffic analysis. These models often necessitate optimization techniques, such as parallel processing or model pruning, to be

feasible for real-time applications. Conversely, simpler models like Logistic Regression and Naive Bayes, despite their relatively lower accuracies, offer advantages in terms of computational efficiency and faster processing times. This makes them suitable for real-time classification tasks where quick decision-making is essential, such as in intrusion detection systems that need to promptly respond to potential threats.

Noise Labels in Network Traffic Data. Introducing noise labels, or deliberately adding incorrect labels, can help models generalize better by preventing overfitting to the training data. This technique enhances the model's robustness to slight variations in the data, which is particularly useful in dynamic network environments where patterns can change frequently. Noise labels can also simulate real-world scenarios where data might be mislabeled due to human error or system inconsistencies, helping the model become more resilient to such issues. However, excessive noise can degrade model performance by confusing the learning process. If the proportion of noise labels is too high, the model might struggle to distinguish between actual patterns and noise, leading to poor accuracy. Furthermore, noise labels can lead to increased training times and computational costs, as the model requires more iterations to converge on a reliable solution. This is especially problematic for real-time applications where efficiency is critical.

4.3 Future Research Directions

The field of machine learning and network traffic analysis is rapidly variable, and here are some potential directions that can open more features and applications for machine learning in network traffic analysis:

Advanced deep learning models, which are more sophisticated for network traffic analysis, can be created. One of the reasons for such an investigation could be an advancement of state-of-the-art data sequencing in network traffic.

Integrating multimodal data across network traffic logs, system logs, and user behavior could provide a more holistic view of network activities. This can be pursued through research to identify effective ways of fusing and analyzing multi-modal data to enhance anomaly detection and threat identification.

Real-Time Analysis and Response. The real-time capabilities of network traffic analysis systems should be improved. Future works should consider optimizing algorithms for speed and making decisions in real-time. That is, developing lightweight models that can run efficiently on edge devices to provide faster responses to threats.

Explainability and Interpretability of Models. The more significant the sophistication of machine learning models, the less it becomes possible to understand the decisions being made. It can be the focal point of research toward improving interpretability and explainability of models to an extent where the network administrator will be able to respect and use insights gained from the

machine learning system.

Robustness Against Adversarial Attacks. Research must also be carried out to ensure that the machine learning models are robust concerning adversarial attacks. Specifically, this study could focus on techniques that aim to give better accuracy in detecting and mitigating adversarial inputs that seek to deceive network traffic analysis systems.

Enhanced Privacy-Preserving Techniques. There are considerable privacy considerations in analyzing network traffic data. Research could be done on advanced privacy-preserving techniques, such as federated learning, in which models can be trained over many decentralized devices without the raw data being shared, thus keeping privacy.

The addition of *noise labels* in the training data can make the models much more robust by testing out several cases for imperfect data, which is a real challenge for the real world and is, however not always perfectly labeled. From a general perspective, there can be further research to look for optimal strategies to add these noise labels, which must have a good trade-off between robustness and accuracy.

Scalability and Handling of Big Data. As the volume of data in network traffic increases, scalability remains a giant challenge. Research may well be directed towards how to build scalable machine-learning frameworks that can very quickly digest and analyze massive volumes of network traffic data.

Cross-Domain and Transfer Learning. Using models trained on a source domain to apply in a target domain (transfer learning) would be an approach to drastically minimize data quantity and the training time of new models. There will be much exploration, in the future, of the possibility of cross-domain learning paths in network traffic analysis by the sharing of knowledge from associated domains.

Integration with Emerging Technologies. Integrating machine learning-based network traffic analysis with two emerging technologies—like blockchain and the Internet of Things (IoT)—opens up thrilling opportunities, and areas on this can be researched. This research can focus on ways by which technologies can complement each other regarding security and efficiencies to be realized in network management.

In other words, the future of network traffic analysis with machine learning could be auspicious, and innumerable research pathways could be followed to make these network management systems more secure, efficient, and intelligent. In summary, research along these directions will assist in the development of more advanced and reliable tools for network traffic analysis.

CONCLUSION

We explored the application of various machine learning (ML) algorithms for network traffic analysis (NTA) to improve the detection and characterization of network anomalies and threats. Our study aimed to identify the most effective models for classifying network traffic, focusing on models such as Naive Bayes, Random Forest, Decision Tree, AdaBoost, XGBoost, K-Nearest Neighbors (KNN), and Logistic Regression.

The results of our research indicate that ensemble methods and tree-based models, particularly Random Forest, XGBoost, and AdaBoost, offer superior performance in terms of accuracy, precision, recall, and F1 score. These models effectively handle large datasets with high dimensionality and exhibit robust generalization capabilities, making them suitable for real-world NTA applications. Logistic Regression also demonstrated strong performance, particularly in scenarios where computational efficiency is critical, due to its simplicity and interpretability.

One of the key findings is the importance of data preprocessing, including feature engineering, balancing datasets, and introducing noise into labels to enhance model robustness. Our experiments showed that these preprocessing steps are crucial for achieving optimal model performance and ensuring that the models can generalize well to new, unseen data.

Moreover, the study underscores the significance of model selection based on specific dataset characteristics and application requirements. While complex models like Random Forest and XGBoost offer high accuracy, they are computationally intensive and may require optimization for real-time applications. Simpler models like Logistic Regression and Naive Bayes, despite their relatively lower accuracy, provide advantages in terms of faster processing times and computational efficiency.

The introduction of noise labels proved to be a valuable technique for improving model generalization by preventing overfitting. However, the proportion of noise must be carefully balanced to avoid degrading model performance.

In conclusion, our research has demonstrated the effectiveness of ML algorithms in NTA and highlighted key considerations for their application. By addressing the identified challenges and pursuing the proposed future research directions, we can further enhance the capabilities of ML models in securing network environments against evolving cyber threats.

BIBLIOGRAPHY

1 *Abbasi, Mahmoud.* Deep learning for network traffic monitoring and analysis (NTMA): A survey / Mahmoud Abbasi, Amin Shahraki, Amir Taherkordi // *Computer Communications.* — 2021. — Vol. 170. — Pp. 19–41.

2 *Wei, Lan.* Whac-A-Mole: Six Years of DNS Spoofing / Lan Wei, John Heidemann // *arXiv preprint arXiv:2011.12978.* — 2020.

3 Machine learning based DDoS detection / S Shanmuga Priya, M Sivaram, D Yuvaraj, A Jayanthiladevi // 2020 international conference on emerging smart computing and informatics (ESCI) / IEEE. — 2020. — Pp. 234–237.

4 *Gaber, Tarek.* Injection attack detection using machine learning for smart IoT applications / Tarek Gaber, Amir El-Ghamry, Aboul Ella Hassanien // *Physical Communication.* — 2022. — Vol. 52. — P. 101685.

5 Impact of Dataset Size on Classification Performance: An Empirical Evaluation in the Medical Domain / Alhanoof Althnian, Duaa AlSaeed, Heyam Al-Baity et al. // *Applied Sciences.* — 2021. — Vol. 11, no. 2.

6 *Vikram, Aditya.* Anomaly detection in Network Traffic Using Unsupervised Machine learning Approach / Aditya Vikram, Mohana // 2020 5th International Conference on Communication and Electronics Systems (ICCES). — 2020. — Pp. 476–479.

7 Unsupervised anomaly detection based on deep autoencoding and clustering / Chuanlei Zhang, Jiangtao Liu, Wei Chen et al. // *Security and Communication Networks.* — 2021. — Vol. 2021. — Pp. 1–8.

8 *Kaiyoo, Hongsang.* Detection of network traffic anomalies using unsupervised machine learning / Hongsang Kaiyoo // *Github.* — 2020.

9 *W, Hector.* Network Traffic Classification Methods Using Machine Learning / Hector W. — 2023.

10 *Thakkar, Ankit.* Attack classification using feature selection techniques: a comparative study / Ankit Thakkar, Ritika Lohiya // *Journal of Ambient Intelligence and Humanized Computing.* — 2021. — Vol. 12, no. 1. — Pp. 1249–1266.

11 Network traffic classification model based on attention mechanism and spatiotemporal features / Feifei Hu, Situo Zhang, Xubin Lin et al. // *EURASIP Journal on Information Security.* — 2023. — Vol. 2023, no. 1. — P. 6.

12 Exploration in deep reinforcement learning: A survey / Pawel Ladosz, Lilian Weng, Minwoo Kim, Hyondong Oh // *Information Fusion.* — 2022. — Vol. 85. — Pp. 1–22.

13 Adaptive machine learning based distributed denial-of-services attacks detection and mitigation system for SDN-enabled IoT / Muhammad Aslam, Dengpan Ye, Aqil Tariq et al. // *Sensors.* — 2022. — Vol. 22, no. 7. — P. 2697.

14 CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment / Euclides Carlos Pinto Neto, Sajjad Dadkhah, Raphael Ferreira et al. // *Sensors*. — 2023. — Vol. 23, no. 13. — P. 5941.

15 Botnet attack detection in iot using machine learning / Khalid Alissa, Tahir Alyas, Kashif Zafar et al. // *Computational Intelligence and Neuroscience*. — 2022. — Vol. 2022.

16 On designing machine learning models for malicious network traffic classification / Talha Ongun, Timothy Sakharaov, Simona Boboila et al. // *arXiv preprint arXiv:1907.04846*. — 2019.

17 How benign is benign overfitting? / Amartya Sanyal, Puneet K Dokania, Varun Kanade, Philip HS Torr // *arXiv preprint arXiv:2007.04028*. — 2020.

18 *Li, Junnan*. Dividemix: Learning with noisy labels as semi-supervised learning / Junnan Li, Richard Socher, Steven CH Hoi // *arXiv preprint arXiv:2002.07394*. — 2020.

19 *Sen, Pratap Chandra*. Supervised classification algorithms in machine learning: A survey and review / Pratap Chandra Sen, Mahimarnab Hajra, Mitadru Ghosh // Emerging Technology in Modelling and Graphics: Proceedings of IEM Graph 2018 / Springer. — 2020. — Pp. 99–111.

20 *Jeffrey, Nicholas*. Using Ensemble Learning for Anomaly Detection in Cyber–Physical Systems / Nicholas Jeffrey, Qing Tan, José R Villar // *Electronics*. — 2024. — Vol. 13, no. 7. — P. 1391.