University of Hertfordshire UH

School of Physics, Engineering and Computer Science

MSc Computer Networks and Systems Security Project

7PAM2002-0901-2024

Department of Physics, Astronomy and Mathematics

FINAL PROJECT REPORT

Project Title:

Enhancing Network Security: Evaluating Anomaly Detection Systems Using Machine Learning on UNSW-NB15 Dataset

By

Waqar Ahmad Fayyaz

Student ID

23033690

Supervisor:	*Mr. Parham Sadeghi*
Date Submitted:	*2025*
Word Count:	
GitHub address:	*Click here to access GitHub*

DECLARATION STATEMENT

This report is submitted in partial fulfilment of the requirement for the degree of Master of Computer Networks and Systems Security at the University of Hertfordshire. I have read the guidance to students on academic integrity, misconduct and plagiarism information at Assessment Offences and Academic Misconduct and understand the University process of dealing with suspected cases of academic misconduct and the possible penalties, which could include failing the project module or course. I certify that the work submitted is my own and that any material derived or quoted from published or unpublished work of other persons has been duly acknowledged. (Ref. UPR AS/C/6.1, section 7 and UPR AS/C/5, section 3.6). I have not used chatGPT, or any other generative AI tool, to write the report or code (other than where declared or referenced). I did not use human participants or undertake a survey in my MSc Project.

I hereby give permission for the report to be made available on module websites provided the source is acknowledged.

Student SRN number:

Student Name printed:

Student signature:

DEDICATION

I dedicate this study to my friends, family, and coworkers, whose unwavering support and constant encouragement kept me going throughout my research journey, particularly when I was feeling challenged and uncertain.

ACKNOWLEDGMENT

I am incredibly grateful to Dr. Parham Sadeghi, my thesis advisor, for her constant encouragement and support during the writing of my thesis. Her passion and commitment were crucial in enabling me to make significant strides in my research. In addition to offering priceless academic advice, Dr. Parham Sadeghi also offered emotional support, which had a big impact. I also want to express my sincere gratitude to my family, without whose support and encouragement this achievement would not have been possible.

ABSTRACT

Growth in the number of devices and data has raised serious security concerns, that have increased the importance of the development of advanced intrusion detection systems (IDS). Deep learning can handle big data and in various fields has shown a great performance. Consequently, security specialists are aiming to adopt deep learning in an intrusion detection system. Numerous studies have been done on this topic which have led to many different approaches. Most of these approaches use predefined features extracted by an expert in order to classify network traffic. Machine learning models vary significantly in their performance, computational efficiency, and suitability for different applications. This study presents a comparative analysis of multiple machine learning models, including Logistic Regression, k-Nearest Neighbors (kNN), Decision Tree, Random Forest, Gradient Boosting, and Neural Network (MLP). The evaluation considers accuracy, recall, precision, F1-score, training time, and prediction time to assess their effectiveness in classification tasks.

Keywords: Machine Learning, Classification Models, Performance Analysis, Computational Efficiency, Training Time, Prediction Time, Model Selection, Random Forest, Neural Networks.

CONTENTS

1. CHAPTER I: INTRODUCTION

1.1. Introduction

Over the past decades, all aspects of our lives have been exposed out to the Internet.Experts predict that 50 billion connected devices will be usable by 2020 (Ouafiq et al., 2022). The difficulty of safeguarding networks and preventing security threats grow as infrastructure becomes more interconnected. Over the years, the vulnerabilities of banking systems, healthcare systems, and IoT tools have increased. These attacks annually lead to billions of dollars in losses in addition to system damage at critical periods. In cybersecurity, particularly in intrusion detection systems, has led to higher importance (Abiodun et al., 2021). One of the related problems with most new infrastructures is that security data specifications are often a backdrop. The result of any machine learning algorithm applied is expected to be affected, but an experiment to assess the discrepancies is still to be seen. The result of any machine learning algorithm applied toward a problem is believed to be affected, however, an analysis to analyse the variations needs to be seen (Berthier and Sanders, 2011). In the current era, computer security has become an indispensable necessity as digital technologies continue to permeate every aspect of daily life(Reveron, 2012). Anomaly detection plays a crucial role in various sectors, such as cybersecurity, manufacturing, and network management, by distinguishing irregularities from normal data patterns. The need for effective anomaly detection algorithms has increased as a result of increasing data complexity and evolving threats. These approaches have been widely applied in recent years to fields such as time series analysis, network traffic monitoring, and image processing, thus demonstrating their versatility and importance in modern applications. The increasing reliance on interconnected systems has amplified security concerns, introducing a plethora of sophisticated threats, including zero-day vulnerabilities, ransomware, advanced persistent threats (APTs), and mobile-specific exploits(Nassar and Kamal, 2021). Despite years of progress in cybersecurity research, many of these challenges remain unresolved, fueled further by the ongoing evolution of computer networks, cloud computing, and the proliferation of IoT devices(Redhu et al., 2024). As a result, protecting the integrity of digital infrastructures has become more critical than ever. Recent reports illustrate the escalating severity of cyber threats. For example, global cybercrime damages are projected to reach $10.5 trillion annually by 2025, with ransomware attacks occurring every 10 seconds. The availability of AI-driven hacking tools and automated malware has drastically lowered the technical barriers for cybercriminals, empowering both individual hackers and organized cybercriminal groups(Mphatheni and Maluleke, 2022).

1.2. Problem Statement

The increase of progressive technologies and connected schemes has led to progressively complex and frequent cyber threats, such as ransomware, zero-day attacks, and DDoS attacks. Out-of-date security events struggle to preserve pace, primarily with the growth of cloud computing and IoT devices. Machine learning-based anomaly detection agreements a real-time solution to these threats but expressions challenges like excessive datasets, high false positives, and scalability issues(Falowo et al., 2024). This research influences the UNSW-NB15 dataset to improve anomaly detection systems, focusing on accuracy, reduced false positives, and flexibility, causative to robust and scalable intrusion detection for contemporary cybersecurity needs.

1.3. Motivation of this Study

Organizations implement Intrusion Detection and Prevention Systems (IDPSs) not only to identify security vulnerabilities but also to record threats and enforce security policiesPatel et al., 2010. In the realm of intrusion detection, Big Data classification has become a key research area, highlighting the necessity of efficient anomaly detection in large-scale networksErhan et al., 2021. With the continuous expansion of computer networks, the significance of network security has grown, requiring innovative intrusion detection techniques and alarm correlation methods to counter evolving cyber threats. As cyberattacks become increasingly sophisticated, the adoption of intelligent security solutions, such as multi-layer hybrid machine-learning algorithms for anomaly detection, is essential in combating these advanced threatsChaabouni et al., 2019.

1.4. Reaserch Statment

This study evaluates machine learning models for detecting network anomalies using the UNSW-NB15 dataset, which includes nine types of cyberattacks(Moustafa and Slay, 2016). It follows a structured approach involving data preprocessing, feature selection, and training six models for comparative analysis. The models are assessed using key performance metrics to determine the most effective one for intrusion detection, contributing to enhanced cybersecurity solutions.

1.5. Research Question

This study seeks to answer the following primary question:

1. How can machine learning techniques be effectively applied to the UNSW-NB15 dataset to enhance the accuracy of network anomaly detection systems?

2. What strategies can be employed to reduce false positives in anomaly detection systems without compromising detection accuracy?
3. How can machine learning-based anomaly detection systems be designed to adapt to evolving network threats and ensure scalability in complex environments?

1.6. Aims and Objectives

This study's main objective is to classify

1. To evaluate the performance of various machine learning models on the UNSW-NB15 dataset, focusing on their accuracy in detecting network anomalies.
2. To develop and implement techniques for reducing false positive rates in anomaly detection systems to improve their reliability.
3. To design a robust and scalable anomaly detection framework capable of adapting to evolving cybersecurity threats in modern networks.

2. CHAPTER II: LITERATURE REVIEW

2.1. Literature Review

To proceed effectively, it is crucial for researchers to review prior studies related to their area of focus.This mechanism supports in generating a strong understanding foundation for the proposed work. This report that show crucial topics for additional research are included in this report literature review. In recent years, machine learning (ML) algorithms have gained significant traction as a problem-solving approach across various scientific domains, including computer vision, behavior analysisZ. Jiang et al., 2018, and cybersecurity, particularly in anomaly detection Abdulhammed et al., 2019.

The potential applications of ML continue to expand Portugal et al., 2018, offering promising solutions across multiple fields. While deep learning also presents effective alternatives, its reliance on large training datasets poses limitations.

Parampottupadam and Moldovann Parampottupadam and Moldovann, 2018 suggest that in certain applications, deep learning models may not necessarily outperform traditional ML models. ML algorithms utilize statistical models to enable systems to make predictions autonomously, learning from sample data—known as training data—without direct human intervention. These algorithms are particularly effective in analyzing large datasets and are categorized into four main types: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning.

In Gharaee and Hosseinvand, 2016, the authors introduced a hybrid Intrusion Detection System (IDS) model based on a Genetic Algorithm (GA) and Support Vector Machine (SVM) for analyzing attacks in the UNSW-NB15 dataset. They represented the features as chromosomes and selected the ones with the highest accuracy. As a detection method, they employed the Least Squares Support Vector Machine (LSSVM). The performance of the model was evaluated using accuracy, true positive rate, and false-positive rate.

In Salman et al., 2017, the authors proposed using a random forest (RF) algorithm as a feature reduction method, focusing on eight specific attack types from the UNSW-NB15 dataset, excluding "Fuzzers" attacks. They developed a stepwise architecture to detect attacks, employing the random forest at each stage. The performance of their model was assessed using the false alarm rate (FAR) and the undetection rate (UND).Mixing multiple machine learning methods in studies is strongly recommended to exploit their strengths to improve the overall performance of IDS. For example, the study in Sharma et al., 2019 demonstrated that IDS can be achieved through a set of layers. The feature selection layer based on Extra Trees Classifiers for each threat was followed for detection by the extreme learning machine ensemble layer. Then, the outputs of the previous layer were collected with the softmax layer to make a soft decision for each attack. Results were limited to accuracy.

Researchers in Khammassi and Krichen, 2017 applied the Genetic Algorithm (GA) alongside a Logistic Regression (LR) wrapper-based feature selection approach to the UNSW-NB15 and KDDCup99 datasets. The study utilized the Weka simulation tool to evaluate performance. After multiple simulations, results indicated that the GA-LR method, when combined with a Decision Tree (DT) classifier, achieved a detection rate of 81.42% with a false alarm rate (FAR) of 6.39% using 20 out of the 42 available features in the UNSW-NB15 dataset. Similarly, for the KDDCup99 dataset, the GA-LR approach, integrated with the DT classifier, yielded a detection rate of 99.90% and a FAR of 0.105% using 18 selected features.

In Moukhafi et al., 2019, the authors aimed to achieve high detection rates by combining the Genetic Algorithm (GA) to eliminate irrelevant features with the Self-Organizing Map (SOM) classifier, which was optimized using the features selected by the GA.

In Ren et al., 2019, a similar approach was used, where GA was combined with a random forest (RF) to select optimal attributes, followed by the Isolation Forest (iForest) for data sampling. The RF classifier was then used to classify attack types for a different purpose. This approach resulted in high accuracy, high detection rates, and lower false alarm rates.

The performance of the IDS with feature reduction outperformed that of models using all features. In Manjunatha et al., 2019, the authors applied Mutual Information with Linear Correlation Coefficient (MI-LCC) to select optimal features, followed by a Support Vector Machine (SVM) classifier for multiclass detection.

In K. Jiang et al., 2020, to address class imbalance issues during the preprocessing stage, data resampling techniques were used, including One-Sided Selection (OSS) to reduce majority class samples and SMOTE to increase minority class samples. Spatial features were extracted using Convolutional Neural Networks (CNN), while temporal features were extracted using Bidirectional Long Short-Term Memory (BiLSTM). These features were then combined for the classification stage.

In Sethi et al., 2020, an IDS for cloud environments was introduced, using Chi-square for feature selection and deep reinforcement learning for classification. The results were evaluated using ROC curves, which presented the accuracy, false positive rate (FPR), and true positive rate (TPR) for each class.

2.2. Summary of Literature Review

Reference	**Method**	**Achievement**	**Limitation**
Portugal et al., 2018	Overview of ML applications	Expanding applications across various fields	Deep learning models rely on large datasets, limiting their effectiveness
Parampottupadam and Moldovann, 2018	Comparison of deep learning and traditional ML models	Highlighted that deep learning may not always outperform traditional ML in certain applications	Limitations in specific use cases where traditional ML might be preferable
Gharaee and Hosseinvand, 2016	Hybrid IDS model (GA + SVM)	High accuracy, evaluated with True Positive Rate (TPR) and False Positive Rate (FPR)	Limited to UNSW-NB15 dataset
Salman et al., 2017	Random Forest (RF) for feature reduction	Stepwise detection using RF, measured with False Alarm Rate (FAR) and Undetection Rate (UND)	Exclusion of 'Fuzzers' attacks
Sharma et al., 2019	Multi-layer IDS using Extra Trees & Extreme Learning Machine	Layered IDS model improved accuracy	Accuracy-focused, no real-time performance assessment
Khammassi and Krichen, 2017	GA + Logistic Regression (LR) feature selection with Decision Tree	Detection rate: 81.42% (UNSW-NB15), 99.90% (KDDCup99), FAR reduced	Limited feature selection, dataset dependency
Moukhafi et al., 2019	GA for feature selection with Self-Organizing Map (SOM) classifier	Feature selection improved IDS performance	Dependence on GA optimization
Ren et al., 2019	GA + RF for feature selection with Isolation Forest	High accuracy, lower false alarm rate	Feature selection complexity, applicability in real-time scenarios
Manjunatha et al., 2019	Mutual Information with Linear Correlation Coefficient (MI-LCC) & SVM	Feature selection improved performance, reduced unnecessary data	Dataset dependency, potential overfitting
K. Jiang et al., 2020	CNN for spatial features, BiLSTM for temporal features	Class imbalance handled using resampling, improved classification with CNN + BiLSTM	High computational complexity due to deep learning models

Sethi et al., 2020	Chi-square for feature selection, Deep Reinforcement Learning for classification	ROC-based evaluation, improved classification accuracy	Computation-intensive, applicability in cloud IDS needs further testing

3. CHAPTER III: METHODOLOGY

This study utilizes the UNSW-NB15 dataset, which includes diverse network traffic data and nine cyber attack types, to evaluate machine learning models for intrusion detection. The methodology involves data preprocessing, feature selection, and training six models for comparative analysis.Shown block diagram of proposed method Fig3.1 These models are assessed using accuracy, precision, recall, and F1 score to determine their effectiveness in detecting attacks. The study identifies the best-performing model, ensuring a robust evaluation of cybersecurity solutions.

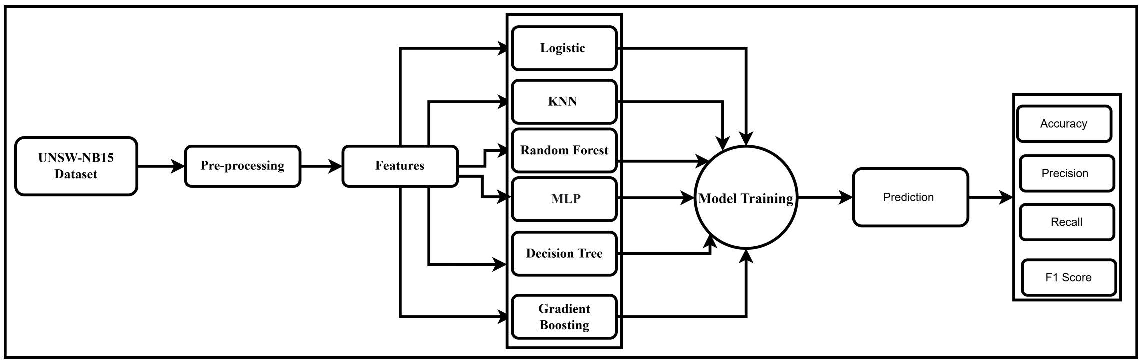

Fig. 3.1. Research Overview Block diagram

3.0.1. UNSW-NB15 Dataset

The UNSW-NB15 dataset is a benchmark for network intrusion detection, combining real-world activities and synthetic attack behaviors. Created at the ACCS using the IXIA PerfectStorm tool, it features nine attack types and 49 attributes derived from 100 GB of network traffic. The dataset, divided into 175,341 training and 82,332 testing records, is widely used in cybersecurity research and supports machine learning-based anomaly detection studies(Moustafa and Slay, 2015).

3.1. Resources

This project utilizes six models for feature extraction, with data preprocessing handled using Pandas and NumPy. Deep learning frameworks such as scikit-learn, TensorFlow, Keras, and PyTorch will be employed. Model performance will be evaluated using accuracy, precision, recall, and F1 score for a comprehensive analysis.

3.2. Logistic Regression

Logistic Regression is a statistical and machine learning algorithm used for binary classification problems. Despite its name, it is actually a classification algorithm, not regression. It models the probability that a given input belongs to a particular class using the logistic (sigmoid) function(Hotzy et al., 2018). Given an input feature vector, Logistic Regression computes a weighted sum of the inputs, applies the sigmoid function, and outputs a probability value between 0 and 1. The model is trained using Maximum Likelihood Estimation (MLE) by minimizing the logistic loss (binary cross-entropy). It is widely used for classification tasks such as spam detection, medical diagnosis, and credit risk prediction(Zaidi and Al Luhayb, 2023).

Mathematics of Logistic Regression

Given an input vector $X = [x_1, x_2, ..., x_n]$, the model computes a linear combination:

$$z = w_0 + \sum_{i=1}^{n} w_i x_i \tag{3.1}$$

where w_i are the model weights, and w_0 is the bias term. The probability of class 1 is obtained using the **sigmoid function**:

$$P(y = 1|X) = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.2}$$

For binary classification, the loss function is the **log-likelihood** (Binary Cross-Entropy):

$$L(w) = -\sum_{i=1}^{m} [y_i \log P(y_i) + (1 - y_i) \log(1 - P(y_i))] \tag{3.3}$$

To find the optimal weights, **gradient descent** is used, updating weights as follows:

$$w_j^{(t+1)} = w_j^{(t)} - \eta \frac{\partial L}{\partial w_j} \tag{3.4}$$

where η is the learning rate. This iterative process continues until the model converges to a minimum loss.

3.3. K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a simple and intuitive machine learning algorithm used for classification and regression tasks. It is a non-parametric, instance-based learning method that makes predictions based on the similarity between data points(S. Zhang et

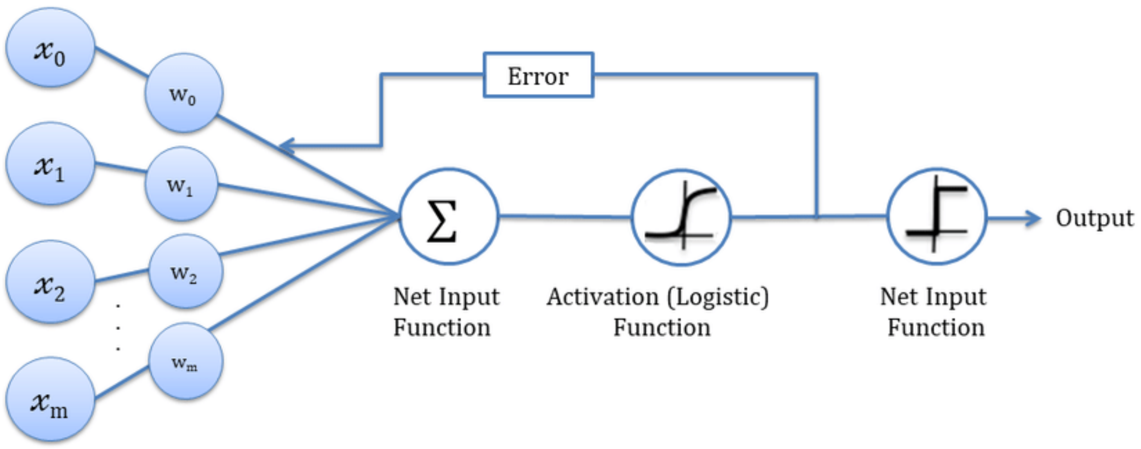

Fig. 3.2. Logistic regression model architecture Nobel et al., 2024

al., 2017). In classification, a new data point is assigned the most common class among its k nearest neighbors, while in regression, the output is the average (or weighted average) of the neighbors' values. KNN relies on a distance metric such as **Euclidean distance**, **Manhattan distance**, or **Minkowski distance** to determine proximity between points. Despite its simplicity, KNN can be computationally expensive for large datasets due to the need to calculate distances for every prediction(Cunningham and Delany, 2021).

Mathematics of KNN

For a given input X, the distance to each training point X_i is calculated using a distance metric, typically **Euclidean distance**:

$$d(X, X_i) = \sqrt{\sum_{j=1}^{n} (X_j - X_{i,j})^2}$$

where X_j and $X_{i,j}$ are the feature values of the new data point and training data point, respectively.

For classification, the predicted class $\hat{y}$ is determined by majority voting among the k nearest neighbors:

$$\hat{y} = \arg\max_{c} \sum_{i \in N_k} \mathbb{1}(y_i = c)$$

where N_k is the set of k nearest neighbors, y_i is the class label of neighbor i, and $\mathbb{1}(\cdot)$ is an indicator function.

For regression, the predicted value $\hat{y}$ is the mean (or weighted mean) of the k nearest neighbors:

$$\hat{y} = \frac{1}{k} \sum_{i \in N_k} y_i$$

A weighted version of KNN can assign weights based on the inverse distance:

$$\hat{y} = \frac{\sum_{i \in N_k} w_i y_i}{\sum_{i \in N_k} w_i}, \quad w_i = \frac{1}{d(X, X_i)}$$

KNN is easy to implement but requires careful selection of k and distance metrics to optimize performance.

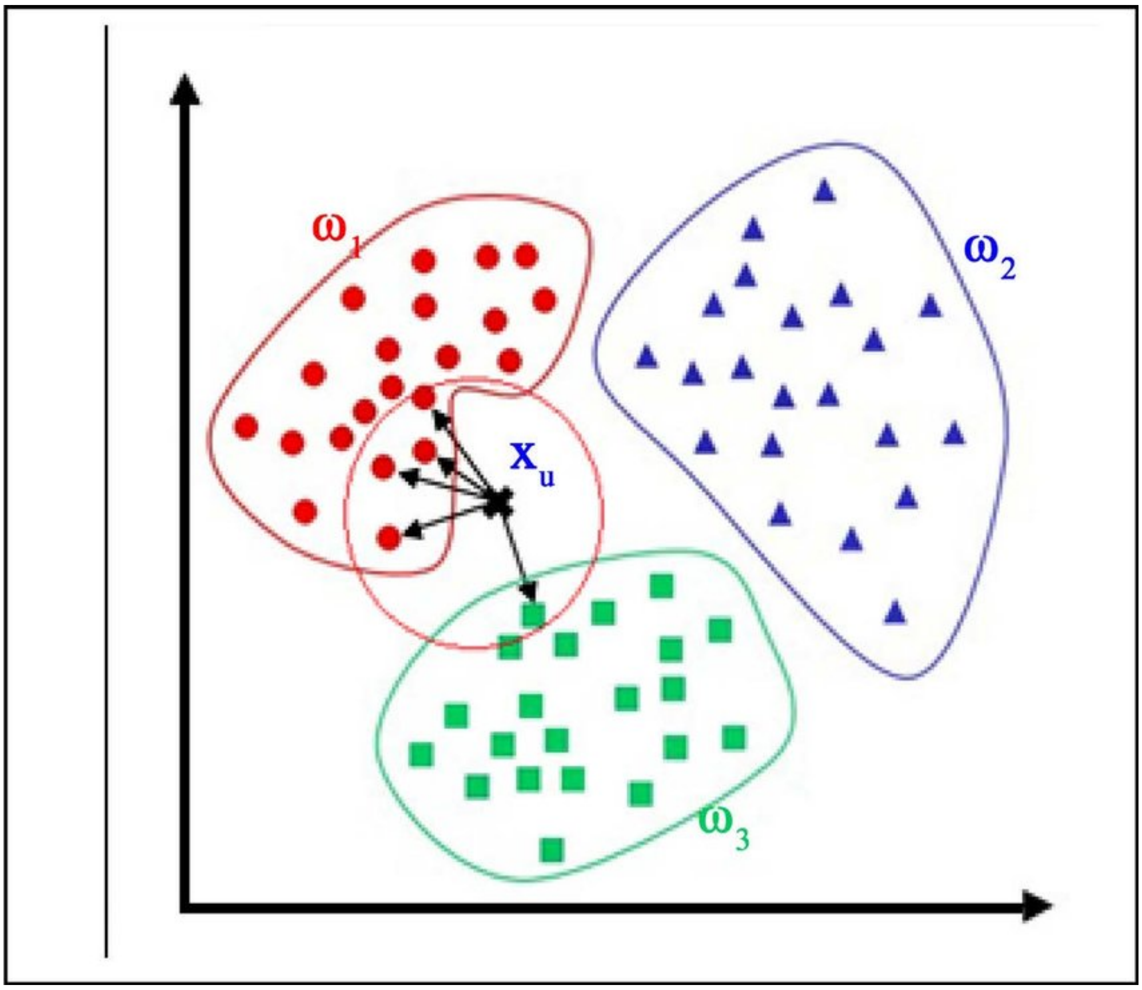

Fig. 3.3. illustration-of-K-nearest-neighbor-model W. Zhang et al., 2017

3.4. Random Forest

Random Forest is an ensemble learning method used for classification and regression tasks. It operates by constructing multiple decision trees during training and aggregating their outputs to improve accuracy and reduce overfitting. Each tree is built from a random subset of the training data using a process called bootstrapping, and at each split, a random subset of features is considered to determine the best split. The final prediction is obtained through majority voting (for classification) or averaging (for regression)(Jaiswal and Samikannu, 2017).

Mathematics of Random Forest

Given a dataset $D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, Random Forest constructs T decision trees $f_1, f_2, ..., f_T$ using different bootstrapped subsets D_t. Each tree predicts an output $\hat{y}_t$, and the final output is:

For classification:

$$\hat{y} = \text{mode}(\hat{y}_1, \hat{y}_2, ..., \hat{y}_T) \tag{3.5}$$

For regression:

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T}\hat{y}_t \tag{3.6}$$

Feature selection at each split is done by randomly selecting m features from the total p features, ensuring diversity among trees. The Gini Index or entropy (for classification) and Mean Squared Error (for regression) are used as impurity measures to decide the best splits.

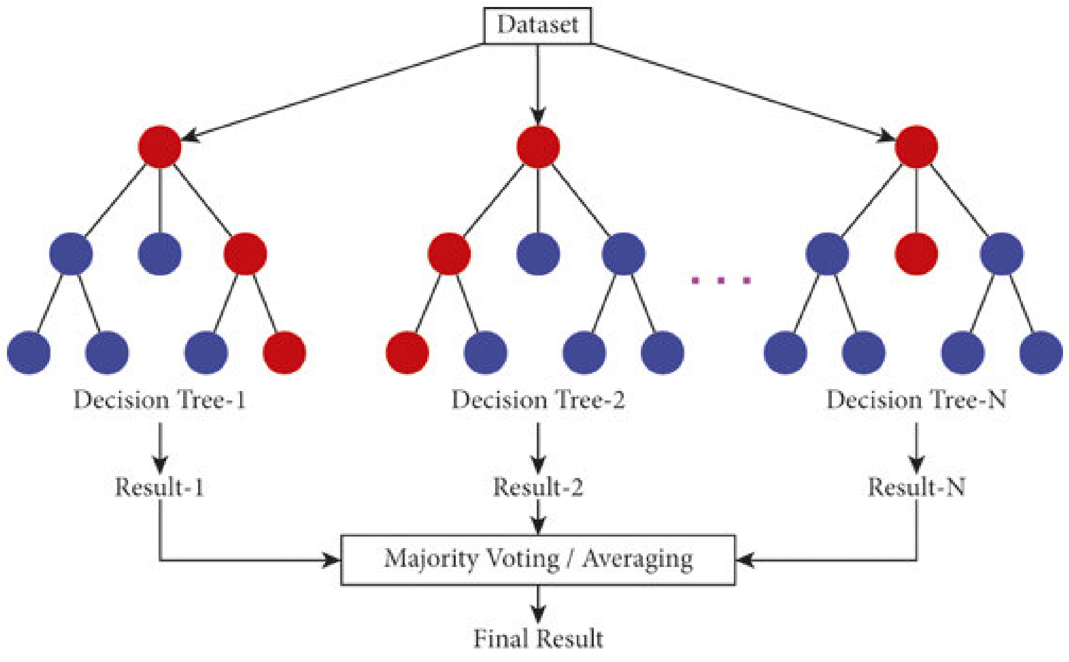

Fig. 3.4. Illustration-of-random-forest-trees Khan et al., 2021

3.5. Multi-Layer Perceptron (MLP)

A **Multi-Layer Perceptron (MLP)** is a type of artificial neural network (ANN) that consists of multiple layers of neurons, including an input layer, one or more hidden layers, and an output layer. Each neuron in a layer is connected to every neuron in the next layer through weighted connections. MLPs use activation functions (such as ReLU, sigmoid, or tanh) to introduce non-linearity, enabling them to learn complex patterns. The network is trained using backpropagation, where the error is propagated backward through the layers, and weights are updated using gradient descent or its variants (e.g., Adam, RMSprop). MLPs are widely used for classification and regression tasks due to their ability to model complex relationships(Riedmiller and Lernen, 2014).

Mathematics of MLP

Given an input vector $X = [x_1, x_2, ..., x_n]$, the output of a neuron in a hidden layer is computed as:

$$z_j = \sum_{i=1}^{n} w_{ij} x_i + b_j \tag{3.7}$$

where w_{ij} are the weights, and b_j is the bias term. The activation function $f(\cdot)$ is applied:

$$a_j = f(z_j) \tag{3.8}$$

For multiple layers, the process is repeated, with the output of one layer serving as the input for the next. The final output $\hat{y}$ is computed in the output layer using:

$$\hat{y} = f\left(\sum_{j=1}^{m} w_{jo} a_j + b_o\right) \tag{3.9}$$

where m is the number of neurons in the last hidden layer. The loss function $L(y, \hat{y})$, such as Mean Squared Error (MSE) for regression or Cross-Entropy for classification, is minimized using backpropagation:

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} - \eta \frac{\partial L}{\partial w_{ij}} \tag{3.10}$$

where η is the learning rate, ensuring weights are adjusted to minimize the error.

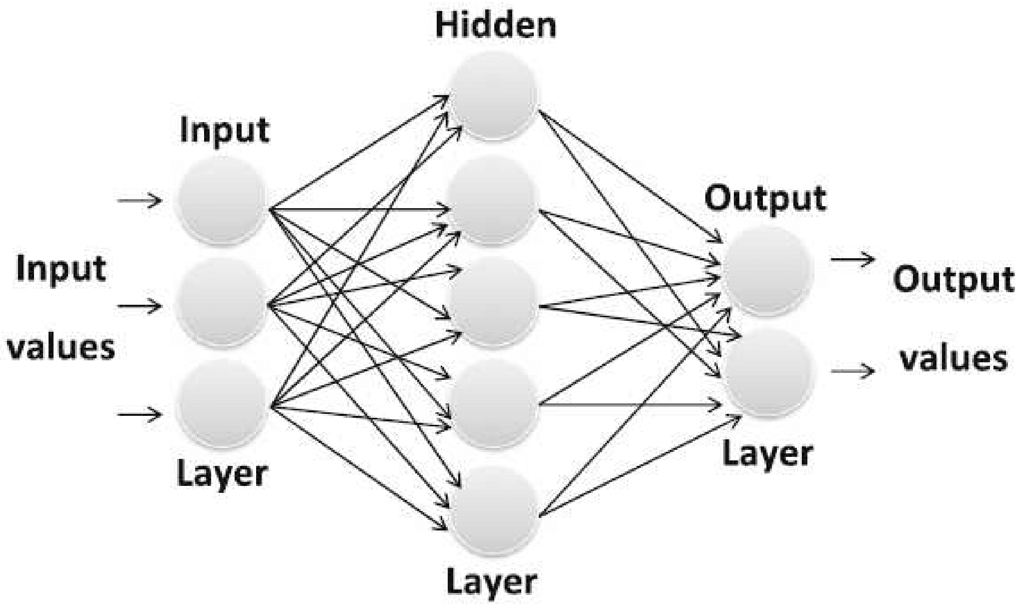

Fig. 3.5. MultiLayer-Perceptron-Classifier Al-Naymat et al., 2016

3.6. Decision Tree

A **Decision Tree** is a supervised machine learning algorithm used for both classification and regression tasks. It works by recursively splitting the dataset into subsets based on the feature that results in the best possible partitioning, typically measured using criteria like Gini Impurity, Information Gain (for classification), or Mean Squared Error (for regression). Each internal node of the tree represents a decision based on a feature, and each leaf node represents the output label or predicted value. The tree grows until a stopping condition is met, such as a maximum depth or minimum number of samples per leaf. Decision Trees are easy to interpret but can easily overfit if not properly pruned(Belgacem et al., 2018).

Mathematics of Decision Tree

The goal of a Decision Tree is to minimize impurity at each split. For classification, the Gini Impurity is used:

$$Gini(t) = 1 - \sum_{i=1}^{k} p_i^2 \tag{3.11}$$

where p_i is the probability of class i in the node t, and k is the number of classes. The algorithm selects the feature that maximizes the reduction in impurity, called the **Information Gain**:

$$IG(t, f) = Gini(t) - \sum_{v \in \{v_1, v_2, \ldots, v_m\}} \frac{|t_v|}{|t|} Gini(t_v) \tag{3.12}$$

where t_v represents the subsets created by splitting feature f at value v, and $|t_v|$ is the number of samples in t_v. For regression, the Mean Squared Error (MSE) is used as the impurity measure:

$$MSE(t) = \frac{1}{|t|} \sum_{i=1}^{|t|} (y_i - \hat{y})^2 \tag{3.13}$$

where $\hat{y}$ is the mean of the target values in the node. The tree is built by recursively splitting the data at the feature with the highest Information Gain or lowest MSE.

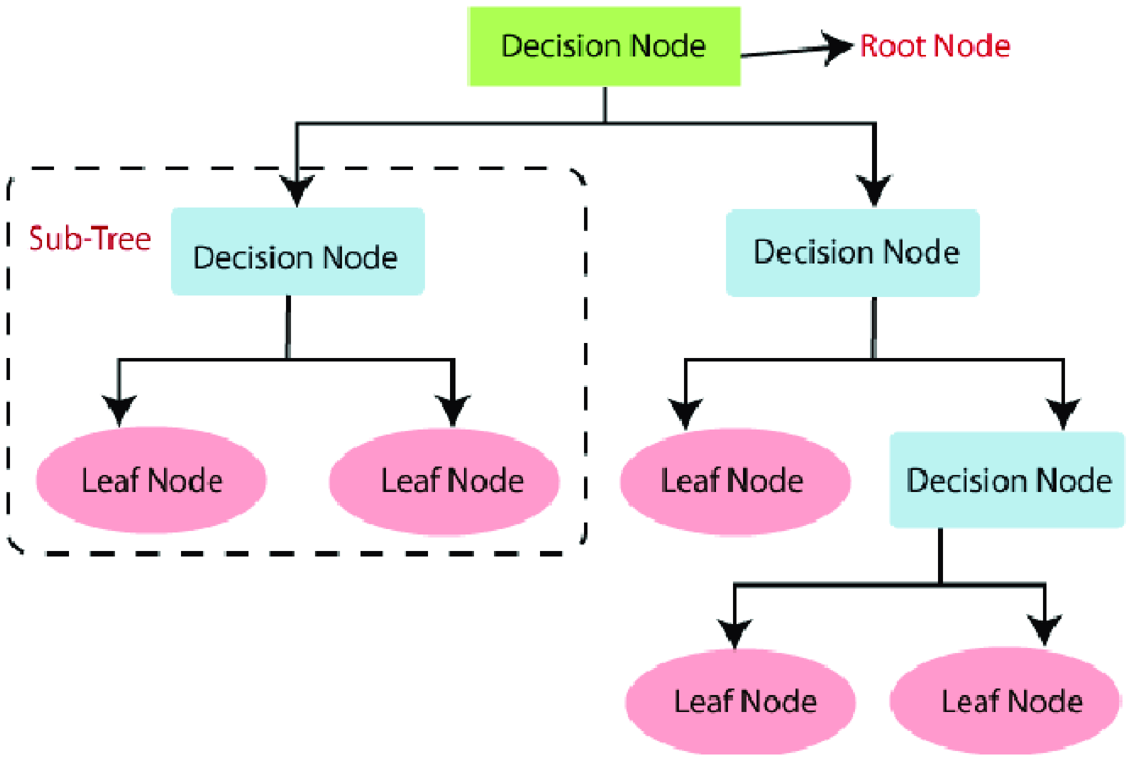

Fig. 3.6. Structure-of-the-decision-tree-DT Hafeez et al., 2021

3.7. Gradient Boosting

Gradient Boosting is a powerful ensemble learning technique used for both classification and regression tasks. It builds models sequentially, where each new model corrects the errors of the previous ones by minimizing the residual errors. Unlike bagging methods, Gradient Boosting combines weak learners (typically decision trees) in a stage-wise manner, optimizing the loss function using gradient descent. At each step, a new tree is trained on the negative gradient of the loss function with respect to the predictions, helping the model learn from mistakes and reduce bias. The final prediction is the sum of all weak learners' outputs, making Gradient Boosting highly effective but sensitive to hyperparameters like learning rate, tree depth, and number of iterations(Natekin and Knoll, 2013).

Mathematics of Gradient Boosting

Given a dataset $D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, Gradient Boosting minimizes a loss function $L(y, F(x))$ iteratively. The model starts with an initial prediction $F_0(x)$, usually the mean of the target values for regression:

$$F_0(x) = \arg\min_c \sum_{i=1}^{n} L(y_i, c) \tag{3.14}$$

At each iteration m, we compute the negative gradient of the loss function:

$$r_i^{(m)} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \tag{3.15}$$

A weak learner (usually a decision tree) $h_m(x)$ is then trained to approximate these residuals:

$$h_m(x) = \arg\min_h \sum_{i=1}^{n} (r_i^{(m)} - h(x_i))^2 \tag{3.16}$$

The model is updated as:

$$F_m(x) = F_{m-1}(x) + \eta h_m(x) \tag{3.17}$$

where η is the learning rate controlling how much each weak learner contributes. The process repeats for M iterations until convergence. Gradient Boosting is highly flexible, supporting different loss functions such as squared error for regression and log loss for classification.

3.7.1. ReLU (Rectified Linear Unit)

ReLU is an activation function used in neural networks. It outputs the input directly if it is positive; otherwise, it outputs zero. This introduces non-linearity into the model and makes training more efficient(Kessler et al., 2017). The mathematical formula for ReLU is given by:

$$\text{ReLU}(x) = \max(0, x) \tag{3.18}$$

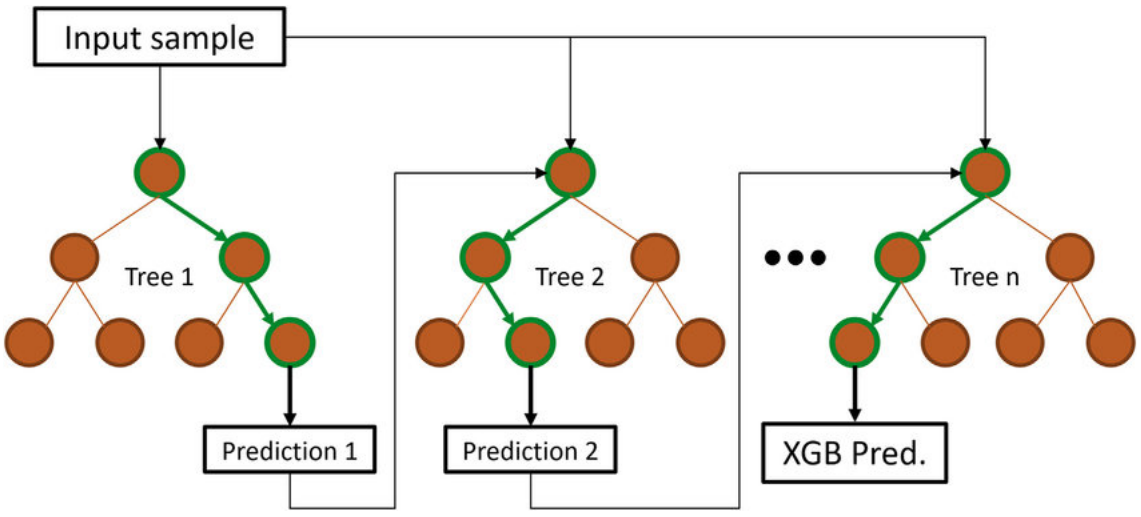

Fig. 3.7. Gradient-boosting-principleTausendschön et al., 2023

3.7.2. Adam (Adaptive Moment Estimation)

Adam is an optimization algorithm used in deep learning. It combines momentum (a moving average of past gradients) and adaptive learning rates to update model parameters(Yi et al., 2020) It maintains two main estimates:

- The first moment estimate (mean gradient)
- The second moment estimate (mean of squared gradients)

The mathematical update equations for Adam are:

$$\hat{m}_t = \beta_1 \hat{m}_{t-1} + (1 - \beta_1) g_t \tag{3.19}$$

$$\hat{v}_t = \beta_2 \hat{v}_{t-1} + (1 - \beta_2) g_t^2 \tag{3.20}$$

$$\theta_t = \theta_{t-1} - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{3.21}$$

where β_1, β_2 are decay rates, g_t is the gradient at time step t, and α is the learning rate.

3.8. Performance Metrics

3.8.1. Accuracy

Definition: Accuracy measures the overall correctness of a model. It is the ratio of correct predictions (both positives and negatives) to the total number of cases.

Formula:

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

3.8.2. Precision

Definition: Precision indicates how many of the predicted positive cases were actually positive. It is a measure of the model's exactness.

Formula:

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

3.8.3. Recall

Definition: Also known as sensitivity, recall measures how many of the actual positive cases were correctly identified by the model.

Formula:

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

3.8.4. F1-Score

Definition: The F1-score is the harmonic mean of precision and recall. It provides a balance between the two metrics, especially useful when you need to account for both false positives and false negatives.

Formula:

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

4. CHAPTER IV: EXPERIMENTAL RESULTS

This section presents the experimental results obtained from evaluating six machine learning models on the UNSW-NB15 dataset for intrusion detection. The assessment focuses on key performance metrics, including accuracy, precision, recall, and F1 score, to determine the effectiveness of each model in identifying cyber threats. The methodology incorporates data preprocessing, feature selection, and model training, ensuring a comprehensive comparative analysis. For optimization, the study follows best practices in cybersecurity research, utilizing appropriate loss functions and training configurations. To achieve reliable and efficient results, the models were trained and tested using a high-performance computing environment, ensuring a robust evaluation of intrusion detection capabilities.

4.1. Logistic Regression Model Performance

The Logistic Regression model achieved an accuracy of 92.56%, indicating a strong ability to classify network intrusions effectively. The recall (92.56%) suggests that the model successfully detects a high proportion of actual threats, while the precision (92.58%) shows that most of the predicted threats are indeed correct. The F1-score (92.56%), a balance between precision and recall, confirms the model's overall robustness. Additionally, the model demonstrates efficient training, completing in 1.21 seconds, with almost instantaneous prediction time (0.00s), making it a fast and reliable choice for intrusion detection.

4.2. Performance Table

Table 4.1. Performance metrics of the Logistic Regression model

Model	Accuracy	Recall	Precision	F1-Score	Training Time (s)	Prediction Time (s)	Total Time (s)
Logistic Regression	92.56%	92.56%	92.58%	92.56%	1.21	0.00	1.21

This analysis highlights the efficiency and high accuracy of the Logistic Regression model in detecting cyber threats.

4.3. K-Nearest Neighbors Model Performance

The K-Nearest Neighbors (KNeighborsClassifier) model achieved an impressive accuracy of 94.84%, indicating its strong ability to classify network intrusions effectively. The recall (94.84%) suggests that the model correctly identifies a high percentage of actual

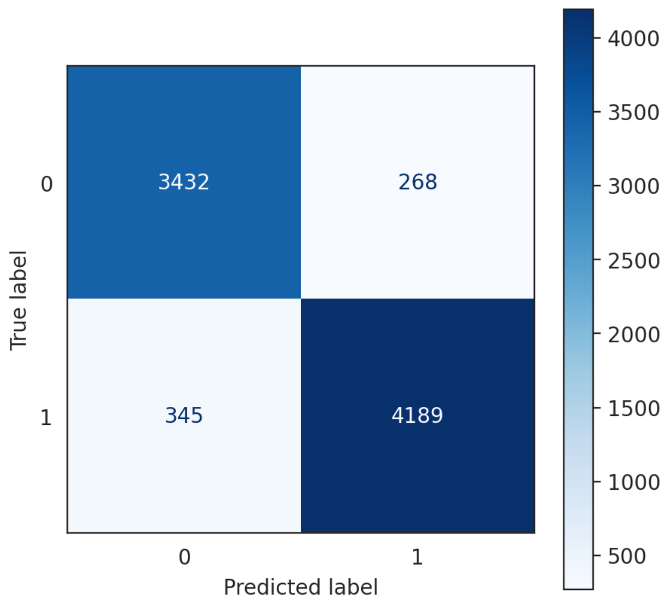

Fig. 4.1. Logistic Regression model confusion matrix

threats, while the precision (94.95%) highlights that most of its predictions are accurate. The F1-score (94.85%), balancing precision and recall, further confirms the model's reliability. The training time is exceptionally fast at 0.01 seconds, but the prediction time (4.03 seconds) is significantly higher, suggesting that the model is computationally expensive during inference.

4.3.1. Performance Table

Table 4.2. Performance metrics of the KNeighborsClassifier model

Model	Accuracy	Recall	Precision	F1-Score	Training Time (s)	Prediction Time (s)	Total Time (s)
KNeighborsClassifier	94.84%	94.84%	94.95%	94.85%	0.01	4.03	4.04

This analysis highlights the high accuracy and effectiveness of the KNeighborsClassifier, but also points to its relatively slow inference speed compared to other models.

4.4. Decision Tree Model Performance

The Decision Tree model achieved a high accuracy of 96.62%, demonstrating its strong ability to classify network intrusions effectively. The recall (96.62%) indicates that the model correctly identifies most actual threats, while the precision (96.63%) shows that a majority of its predictions are accurate. The F1-score (96.62%), which balances precision and recall, confirms the model's overall reliability. The training time (1.69 seconds) is moderate, and the prediction time (0.00 seconds) is exceptionally fast, making this model both effective and efficient for real-time intrusion detection.

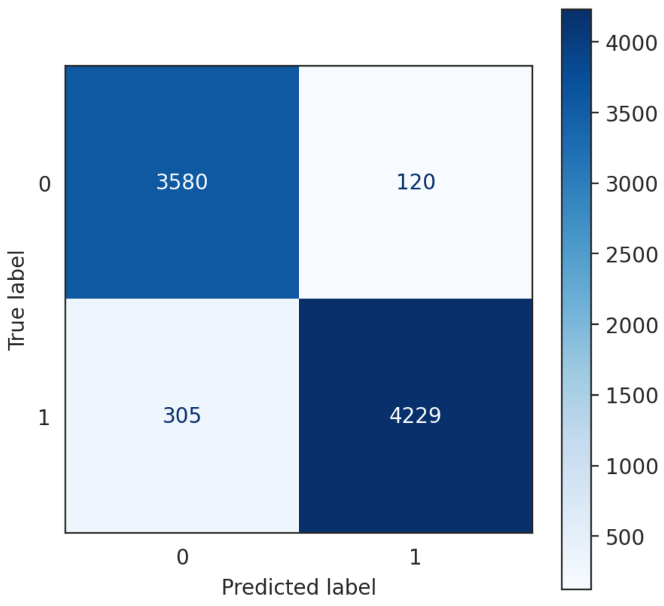

Fig. 4.2. KNeighbors Classifier model confusion matrix

4.4.1. Performance Table

Table 4.3. Performance metrics of the Decision Tree model

Model	**Accuracy**	**Recall**	**Precision**	**F1-Score**	**Training Time (s)**	**Prediction Time (s)**	**Total Time (s)**
Decision Tree	96.62%	96.62%	96.63%	96.62%	1.69	0.00	1.70

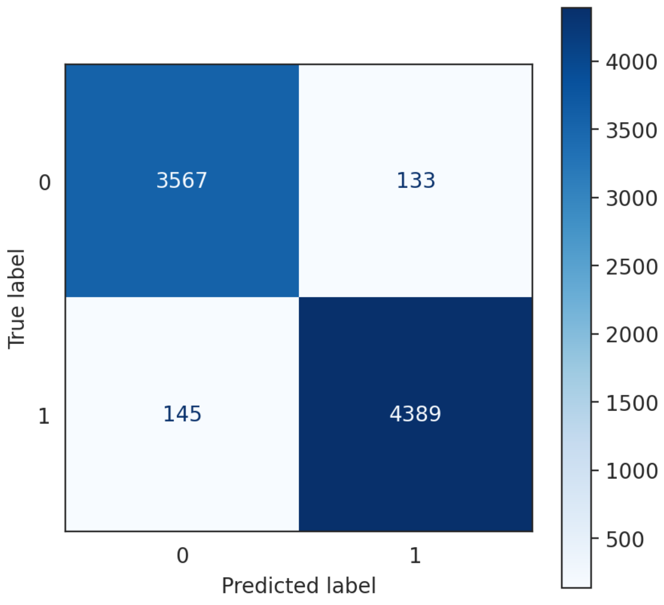

Fig. 4.3. Decision Tree model confusion matrix

This analysis highlights the high accuracy, efficiency, and real-time prediction speed of the Decision Tree model, making it a strong candidate for cybersecurity applications.

4.5. Random Forest Model Performance

The Random Forest model achieved an outstanding accuracy of 97.70%, demonstrating its superior capability in detecting network intrusions. The recall (97.70%) confirms that the model effectively identifies most actual threats, while the precision (97.71%) ensures that its predictions are highly accurate. The F1-score (97.71%) highlights the balance between precision and recall, validating the model's overall performance. However, the training time (56.75 seconds) is significantly higher compared to other models, indicating a computationally intensive process. The prediction time (0.71 seconds) is relatively fast, making it efficient for real-time detection while maintaining high accuracy.

4.5.1. Performance Table

Table 4.4. Performance metrics of the Random Forest model

Model	Accuracy	Recall	Precision	F1-Score	Training Time (s)	Prediction Time (s)	Total Time (s)
Random Forest	97.70%	97.70%	97.71%	97.71%	56.75	0.71	57.46

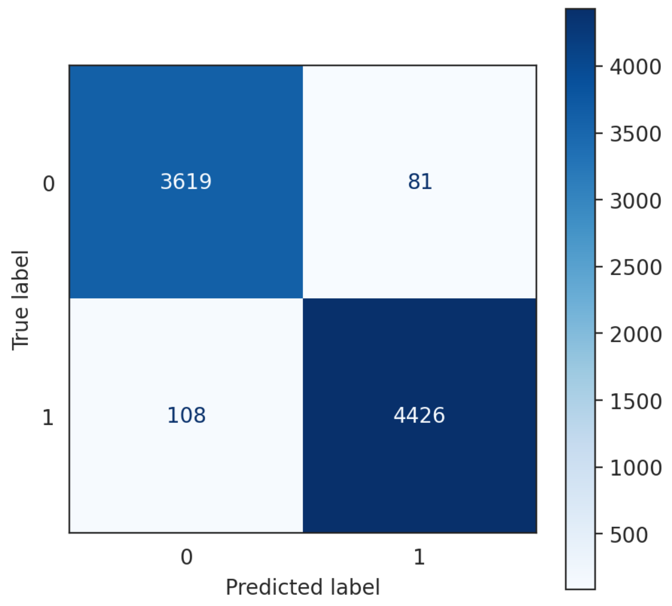

Fig. 4.4. Random Forest model confusion matrix

This analysis highlights the exceptional accuracy and effectiveness of the Random Forest model, making it a powerful choice for intrusion detection despite its higher training time.

4.6. Gradient Boosting Model Performance

The Gradient Boosting Classifier achieved a high accuracy of 96.60%, demonstrating its strong performance in detecting network intrusions. The recall (96.60%) suggests that the model effectively identifies most actual threats, while the precision (96.61%) confirms the accuracy of its predictions. The F1-score (96.60%) highlights the balance between precision and recall, making it a reliable model for intrusion detection. However, the training time (27.74 seconds) is relatively long, indicating higher computational complexity. The prediction time (0.00 seconds) is exceptionally fast, making it efficient for real-time threat detection despite its intensive training phase.

4.7. Performance Table

Table 4.5. Performance metrics of the Gradient Boosting Classifier model

Model	**Accuracy**	**Recall**	**Precision**	**F1-Score**	**Training Time (s)**	**Prediction Time (s)**	**Total Time (s)**
Gradient Boosting Classifier	96.60%	96.60%	96.61%	96.60%	27.74	0.00	27.74

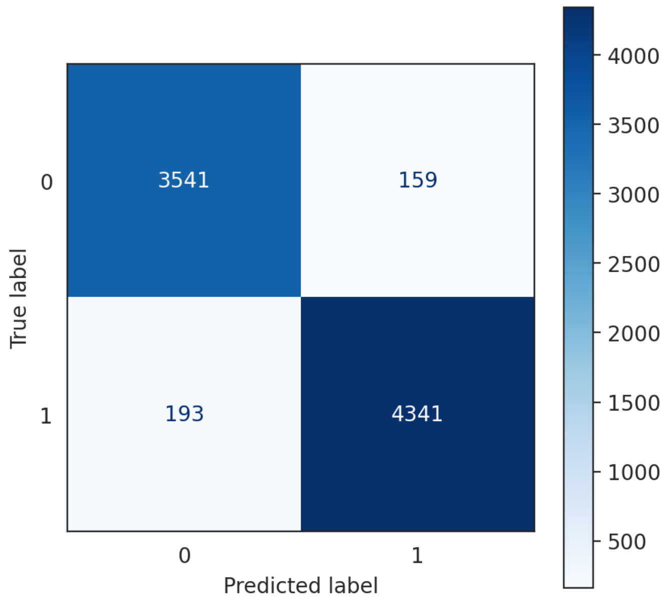

Fig. 4.5. Gradient Boosting model confusion matrix

This analysis highlights the high accuracy and efficiency of the Gradient Boosting Classifier, making it a strong contender for intrusion detection while requiring a relatively long training time.

4.8. Neural Network MLP Model Performance

The Neural Network MLP achieved an accuracy of 96.42%, indicating strong performance in detecting network intrusions. The recall (96.42%) shows that the model effectively identifies most actual threats, while the precision (96.45%) ensures that its predictions are highly accurate. The F1-score (96.42%) confirms the balance between precision and recall, making it a reliable model for intrusion detection. The training time (32.79 seconds) is relatively long, but the prediction time (0.01 seconds) is very efficient, making it suitable for real-time threat detection with minimal inference delay.

4.8.1. Performance Table

Table 4.6. Performance metrics of the Neural Network MLP model

Model	Accuracy	Recall	Precision	F1-Score	Training Time (s)	Prediction Time (s)	Total Time (s)
Neural Network MLP	96.42%	96.42%	96.45%	96.42%	32.79	0.01	32.80

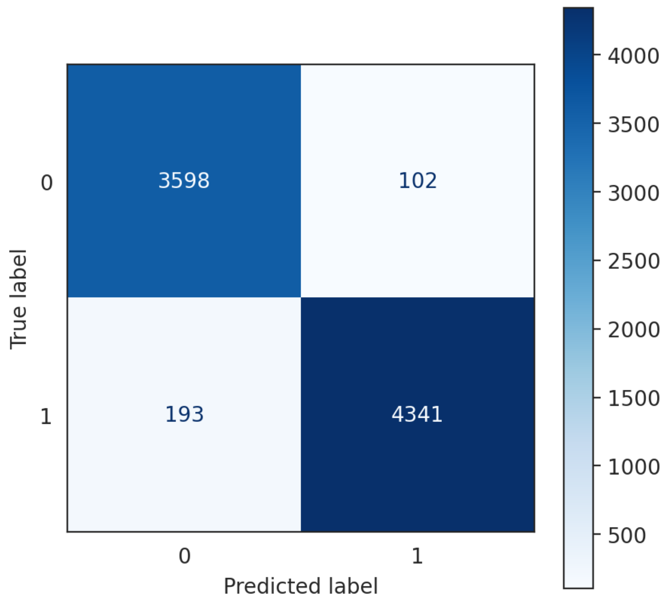

Fig. 4.6. Neural Network MLP model confusion matrix

This analysis highlights the high accuracy and efficiency of the Neural Network MLP model, making it a robust choice for intrusion detection with minimal prediction latency.

4.9. Comparative Analysis

The performance of different machine learning models was evaluated based on Accuracy, Recall, Precision, and F1-score, revealing notable variations. Among all models, Random Forest achieved the highest accuracy (97.70%), demonstrating its ability to handle complex patterns effectively. Decision Tree (96.59%) and Neural Network MLP (96.42%) followed closely, while kNN (94.84%) and Gradient Boosting Classifier (95.73%) also performed well but lagged slightly behind. Logistic Regression (92.56%) had the lowest accuracy. In terms of recall and precision, Random Forest led with 97.70% and 97.71%, respectively, followed by Decision Tree and MLP, both maintaining balanced scores around 96.59% and 96.42%. Gradient Boosting (95.73%) and kNN (94.84%)

were competitive, whereas Logistic Regression had the lowest values. The F1-score, being the harmonic mean of precision and recall, followed the same pattern, with Random Forest leading at 97.71%, Decision Tree and MLP scoring around 96.59% and 96.42%, and Logistic Regression at the bottom with 92.56%. Overall, Random Forest emerged as the best-performing model, excelling in accuracy and precision-recall balance. However, Decision Tree and MLP also provided strong alternatives with high accuracy and balanced classification. While Gradient Boosting performed well, it did not surpass tree-based models. kNN outperformed Logistic Regression, but its instance-based learning approach can make it computationally expensive for large datasets. Despite Random Forest's superior performance, its higher computational requirements make simpler models like Decision Tree or MLP more suitable for real-time or resource-constrained environments. Hence, if maximizing accuracy and robustness is the priority, Random Forest is the best choice, while Decision Tree and MLP provide strong alternatives with lower computational costs.

h

4.10. Model Training and Prediction Time Analysis

The table presents the training time, prediction time, and total execution time for various machine learning models. Logistic Regression and Decision Tree models demonstrate the fastest training times, requiring 2.47 seconds and 1.21 seconds, respectively, making them suitable for quick model deployment. However, k-Nearest Neighbors (kNN) has an extremely short training time (0.01 seconds) but suffers from a high prediction time of 6.36 seconds, making it inefficient for real-time applications. Random Forest and Gradient Boosting exhibit the longest training times, at 57.01 seconds and 40.53 seconds, respectively, due to their ensemble nature, but they provide high accuracy. Neural Network (MLP) requires 32.79 seconds for training, making it computationally expensive but still faster than ensemble models. The Random Forest model has a relatively high total execution time of 57.41 seconds, while the Decision Tree model remains the most efficient, with a total time of just 1.21 seconds. These results indicate a clear trade-off between model complexity, training efficiency, and inference speed, which should be considered when selecting an appropriate model for a given task.

Table 4.7. Training, Prediction, and Total Time for Models

Model	Training Time (s)	Prediction Time (s)	Total Time (s)
Logistic Regression	2.4677	0.0066	2.4743
kNN	0.0118	6.3597	6.3715
Decision Tree	1.2092	0.0021	1.2113
Random Forest	57.0069	0.3990	57.4059
Gradient Boosting	40.5316	0.0198	40.5515
MLP	32.7896	0.0057	32.7953

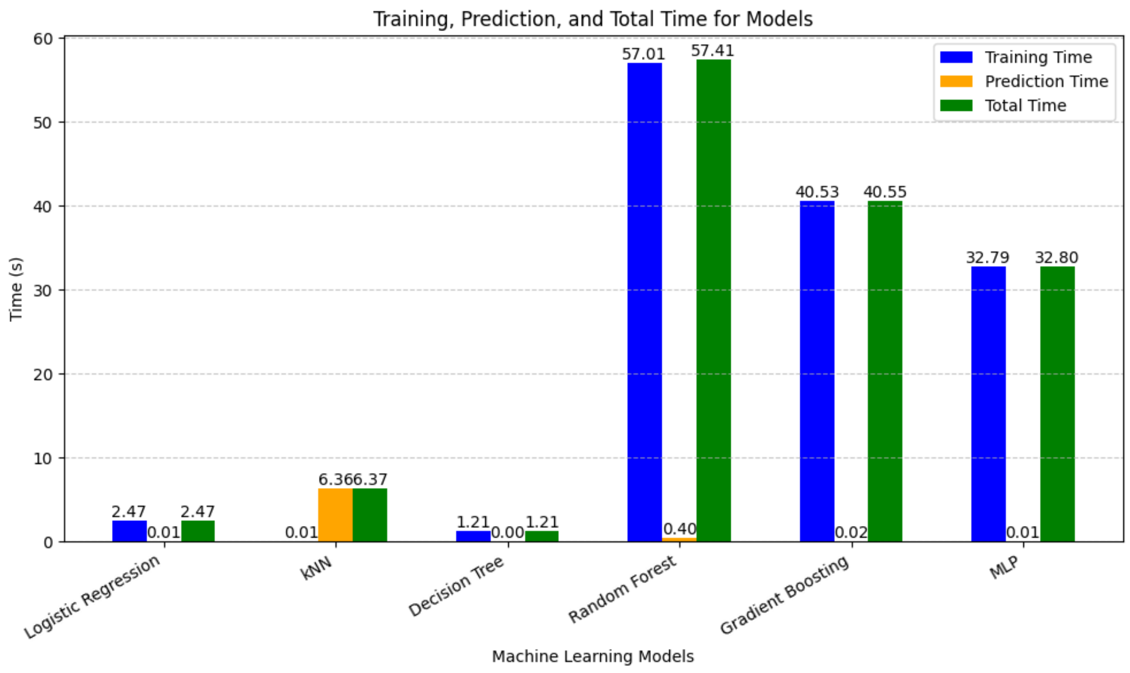

Fig. 4.7. Training, Prediction, and Total Time for Models graph

5. CONCLUSION AND FUTURE WORK

5.0.1. Conclusion

This study presents a comprehensive evaluation of six machine learning models for intrusion detection using the UNSW-NB15 dataset. The models were assessed based on key performance metrics, including accuracy, recall, precision, and F1-score, as well as their training and prediction times. The Random Forest model demonstrated the highest accuracy (97.70%), making it the most effective in detecting network intrusions, but it required significantly higher training time. The Decision Tree and Neural Network MLP models also showed competitive performance, achieving accuracy levels of 96.59% and 96.42%, respectively, with relatively lower computational costs. k-Nearest Neighbors (kNN) and Gradient Boosting Classifier performed well but exhibited trade-offs in terms of efficiency and inference speed. Logistic Regression, while the simplest model, had the lowest accuracy (92.56%) but provided fast training and prediction, making it a viable choice for lightweight applications.Furthermore, the analysis of training and prediction times revealed that ensemble models (Random Forest and Gradient Boosting) are computationally expensive, making them less suitable for real-time applications. Conversely, Decision Tree and Logistic Regression offered a balance between efficiency and performance, making them attractive for practical implementations where computational resources are limited. kNN had the slowest prediction time, indicating inefficiency in handling large-scale real-time intrusion detection. These findings emphasize the importance of selecting models based on the trade-off between accuracy, computational complexity, and real-time performance in cybersecurity applications.

5.0.2. Future Work

Although the models evaluated in this study achieved high detection rates, several areas for future research and improvements remain. First, deep learning approaches, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), can be explored to further improve accuracy and adaptability in detecting complex attack patterns. Second, feature engineering techniques such as autoencoders and principal component analysis (PCA) could be employed to enhance the quality of input data and reduce computational overhead. Third, incorporating hybrid models, combining machine learning and deep learning techniques, may lead to enhanced intrusion detection capabilities by leveraging the strengths of both approaches.

Additionally, real-world deployment and evaluation in high-traffic network environments will be crucial in validating the effectiveness of these models. Further studies could also investigate the adversarial robustness of these models against evasion attacks, ensuring that intrusion detection systems remain resilient against adversaries. Moreover, optimizing the models for low-latency environments using edge computing techniques can improve real-time detection performance. Lastly, semi-supervised and unsupervised learning approaches could be explored to handle zero-day attacks and novel threat patterns, which are challenging to detect using traditional supervised models.By integrating these advancements, future research can develop more efficient, scalable, and robust intrusion detection systems, ensuring greater cybersecurity in real-world applications.

BIBLIOGRAPHY

Abdulhammed, R., Musafer, H., Alessa, A., Faezipour, M., & Abuzneid, A. (2019). Features dimensionality reduction approaches for machine learning based network intrusion detection. *Electronics*, *8*(3), 322.

Abiodun, O. I., Abiodun, E. O., Alawida, M., Alkhawaldeh, R. S., & Arshad, H. (2021). A review on the security of the internet of things: Challenges and solutions. *Wireless Personal Communications*, *119*, 2603–2637.

Al-Naymat, G., Al-Kasassbeh, M., Abu-Samhadanh, N., & Sakr, S. (2016). Classification of voip and non-voip traffic using machine learning approaches. *Journal of Theoretical & Applied Information Technology*.

Belgacem, R., Malek, I. T., Trabelsi, H., & Jabri, I. (2018). A supervised machine learning algorithm skvms used for both classification and screening of glaucoma disease. *New Front. Ophthalmol*, *4*(4), 1–27.

Berthier, R., & Sanders, W. H. (2011). Specification-based intrusion detection for advanced metering infrastructures. *2011 IEEE 17th Pacific rim international symposium on dependable computing*, 184–193.

Chaabouni, N., Mosbah, M., Zemmari, A., Sauvignac, C., & Faruki, P. (2019). Network intrusion detection for iot security based on learning techniques. *IEEE Communications Surveys & Tutorials*, *21*(3), 2671–2701.

Cunningham, P., & Delany, S. J. (2021). K-nearest neighbour classifiers-a tutorial. *ACM computing surveys (CSUR)*, *54*(6), 1–25.

Erhan, L., Ndubuaku, M., Di Mauro, M., Song, W., Chen, M., Fortino, G., Bagdasar, O., & Liotta, A. (2021). Smart anomaly detection in sensor systems: A multi-perspective review. *Information Fusion*, *67*, 64–79.

Falowo, O. I., Ozer, M., Li, C., & Abdo, J. B. (2024). Evolving malware & ddos attacks: Decadal longitudinal study. *IEEE Access*.

Gharaee, H., & Hosseinvand, H. (2016). A new feature selection ids based on genetic algorithm and svm. *2016 8th International Symposium on Telecommunications (IST)*, 139–144.

Hafeez, M. A., Rashid, M., Tariq, H., Abideen, Z. U., Alotaibi, S. S., & Sinky, M. H. (2021). Performance improvement of decision tree: A robust classifier using tabu search algorithm. *Applied Sciences*, *11*(15), 6728.

Hotzy, F., Theodoridou, A., Hoff, P., Schneeberger, A. R., Seifritz, E., Olbrich, S., & Jäger, M. (2018). Machine learning: An approach in identifying risk factors for coercion compared to binary logistic regression. *Frontiers in psychiatry*, *9*, 258.

Jaiswal, J. K., & Samikannu, R. (2017). Application of random forest algorithm on feature subset selection and classification and regression. *2017 world congress on computing and communication technologies (WCCCT)*, 65–68.

Jiang, K., Wang, W., Wang, A., & Wu, H. (2020). Network intrusion detection combined hybrid sampling with deep hierarchical network. *IEEE access*, *8*, 32464–32476.

Jiang, Z., Crookes, D., Green, B. D., Zhao, Y., Ma, H., Li, L., Zhang, S., Tao, D., & Zhou, H. (2018). Context-aware mouse behavior recognition using hidden markov models. *IEEE Transactions on Image Processing*, *28*(3), 1133–1148.

Kessler, T., Dorian, G., & Mack, J. H. (2017). Application of a rectified linear unit (relu) based artificial neural network to cetane number predictions. *Internal combustion engine division fall technical conference*, *58318*, V001T02A006.

Khammassi, C., & Krichen, S. (2017). A ga-lr wrapper approach for feature selection in network intrusion detection. *computers & security*, *70*, 255–277.

Khan, M. Y., Qayoom, A., Nizami, M. S., Siddiqui, M. S., Wasi, S., & Raazi, S. M. K.-R. (2021). Automated prediction of good dictionary examples (gdex): A comprehensive experiment with distant supervision, machine learning, and word embedding-based deep learning techniques. *Complexity*, *2021*(1), 2553199.

Manjunatha, B., Gogoi, P., & Akkalappa, M. (2019). Data mining based framework for effective intrusion detection using hybrid feature selection approach. *International Journal of Computer Network and Information Security*, *11*(8), 1.

Moukhafi, M., Yassini, K. E., Bri, S., & Oufaska, K. (2019). Artificial neural network optimized by genetic algorithm for intrusion detection system. *Advanced Intelligent Systems for Sustainable Development (AI2SD'2018) Volume 5: Advanced Intelligent Systems for Computing Sciences*, 393–404.

Moustafa, N., & Slay, J. (2015). Unsw-nb15: A comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). *2015 military communications and information systems conference (MilCIS)*, 1–6.

Moustafa, N., & Slay, J. (2016). The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set. *Information Security Journal: A Global Perspective*, *25*(1-3), 18–31.

Mphatheni, M. R., & Maluleke, W. (2022). Cybersecurity as a response to combating cybercrime: Demystifying the prevailing threats and offering recommendations to the african regions. *International Journal of Research in Business and Social Science (2147-4478)*, *11*(4), 384–396.

Nassar, A., & Kamal, M. (2021). Machine learning and big data analytics for cybersecurity threat detection: A holistic review of techniques and case studies. *Journal of Artificial Intelligence and Machine Learning in Management*, *5*(1), 51–63.

Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, *7*, 21.

Nobel, S. N., Sultana, S., Singha, S. P., Chaki, S., Mahi, M. J. N., Jan, T., Barros, A., & Whaiduzzaman, M. (2024). Unmasking banking fraud: Unleashing the power of machine learning and explainable ai (xai) on imbalanced data. *Information*, *15*(6), 298.

Ouafiq, E. M., Saadane, R., & Chehri, A. (2022). Data management and integration of low power consumption embedded devices iot for transforming smart agriculture into actionable knowledge. *Agriculture*, *12*(3), 329.

Parampottupadam, S., & Moldovann, A.-N. (2018). Cloud-based real-time network intrusion detection using deep learning. *2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, 1–8.

Patel, A., Qassim, Q., & Wills, C. (2010). A survey of intrusion detection and prevention systems. *Information Management & Computer Security*, *18*(4), 277–290.

Portugal, I., Alencar, P., & Cowan, D. (2018). The use of machine learning algorithms in recommender systems: A systematic review. *Expert Systems with Applications*, *97*, 205–227.

Redhu, A., Choudhary, P., Srinivasan, K., & Das, T. K. (2024). Deep learning-powered malware detection in cyberspace: A contemporary review. *Frontiers in Physics*, *12*, 1349463.

Ren, J., Guo, J., Qian, W., Yuan, H., Hao, X., & Jingjing, H. (2019). Building an effective intrusion detection system by using hybrid data optimization based on machine learning algorithms. *Security and communication networks*, *2019*(1), 7130868.

Reveron, D. S. (2012). *Cyberspace and national security: Threats, opportunities, and power in a virtual world*. Georgetown University Press.

Riedmiller, M., & Lernen, A. (2014). Multi layer perceptron. *Machine learning lab special lecture, University of Freiburg*, *24*.

Salman, T., Bhamare, D., Erbad, A., Jain, R., & Samaka, M. (2017). Machine learning for anomaly detection and categorization in multi-cloud environments. *2017 IEEE 4th international conference on cyber security and cloud computing (CSCloud)*, 97–103.

Sethi, K., Kumar, R., Prajapati, N., & Bera, P. (2020). Deep reinforcement learning based intrusion detection system for cloud infrastructure. *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, 1–6.

Sharma, J., Giri, C., Granmo, O.-C., & Goodwin, M. (2019). Multi-layer intrusion detection system with extratrees feature selection, extreme learning machine ensemble, and softmax aggregation. *EURASIP Journal on Information Security*, *2019*(1), 1–16.

Tausendschön, J., Stoeckl, G., & Radl, S. (2023). Machine learning for heat radiation modeling of bi-and polydisperse particle systems including walls. *Particuology*, *74*, 119–140.

Yi, D., Ahn, J., & Ji, S. (2020). An effective optimization method for machine learning based on adam. *Applied Sciences*, *10*(3), 1073.

Zaidi, A., & Al Luhayb, A. S. M. (2023). Two statistical approaches to justify the use of the logistic function in binary logistic regression. *Mathematical Problems in Engineering*, *2023*(1), 5525675.

Zhang, S., Li, X., Zong, M., Zhu, X., & Cheng, D. (2017). Learning k for knn classification. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *8*(3), 1–19.

Zhang, W., et al. (2017). Machine learning approaches to predicting company bankruptcy. *Journal of Financial Risk Management*, *6*(04), 364.

6. APPENDICES