ASEAN
Journal of Scientific and Technological Reports
Online ISSN:2773-8752

Research article

Comparative Analysis of DNN, GBT, and KNN Models for Network Intrusion Detection

Rattikan Viboonpanich[1], Amornvit Vatcharaphrueksadee[2*], and Wilairat Charoenmairungrueang[3]

[1] Faculty of Information Technology and Digital Innovation, North Bangkok University, Bangkok, 12130, Thailand
[2] Faculty of Information Technology and Digital Innovation, North Bangkok University, Bangkok, 12130, Thailand
[3] Faculty of Information Technology and Digital Innovation, North Bangkok University, Bangkok, 12130, Thailand
* Correspondence: amornvit.va@northbkk.ac.th

Citation:
Viboonpanich, R.; Vatcharaphrueksadee, A.; Charoenmairungrueang, W. Comparative analysis of DNN, GBT, and KNN models for network intrusion detection. *ASEAN J. Sci. Tech. Report.* **2024**, *27*(5), e252675. https://doi.org/10.55164/ajstr.v27i5.252675

Received: February 9, 2024
Revised: July 22, 2024
Accepted: August 8, 2024
Available online: August 27, 2024

Abstract: Network intrusion detection is critical to cybersecurity, aiming to identify and mitigate unauthorized access and attacks on computer systems and networks. This study evaluates the effectiveness of three machine learning techniques—deep neural networks (DNN), gradient boost trees (GBT), and k-nearest neighbors (KNN)—in detecting network intrusions. The performance of these models was assessed using a comprehensive dataset of 2,540,047 records encompassing 49 features across nine attack categories. The results indicate that GBT outperforms DNN and KNN in accuracy and robustness. These findings highlight the potential of GBT for enhancing intrusion detection systems and contribute valuable insights into the comparative performance of different machine learning algorithms in cybersecurity applications.

Keywords: Network Attacks Forecasting; UNSW-NB15 Dataset; Deep Neural Networks; Gradient Boost Trees; k-Nearest Neighbors

1. Introduction

In the current digital landscape, the security of data, computer systems, and networks is paramount due to the critical nature of the information they contain, ranging from healthcare and financial data to personal records [1]. The increasing reliance on digital storage has escalated the risks associated with data breaches and cyber intrusions, which can lead to significant economic damage and degrade the performance of information systems [1]. Therefore, developing advanced Intrusion Detection Systems (IDS) is essential for preserving data confidentiality and system integrity. This study makes significant contributions by evaluating the effectiveness of three advanced machine learning techniques—deep neural networks (DNN), gradient boost trees (GBT), and k-nearest neighbors (KNN)—in enhancing IDS. By comparing these techniques, the research provides valuable insights into their performance, aiding in developing more robust cybersecurity measures. Machine Learning (ML), with its adaptive learning capabilities from data, emerges as a potent tool for enhancing IDS by identifying and mitigating sophisticated cyber threats [2,3,4,5].

The application of ML in IDS development has attracted considerable attention due to its potential to improve intrusion detection efficacy significantly [6]. The UNSW-NB15 dataset, with its comprehensive coverage of various attack vectors such as Fuzzers, Analysis, Backdoors, DoS attacks, Exploits, and Viruses, provides an invaluable resource for IDS research and development, offering insights into potential vulnerabilities and the effectiveness of different detection strategies [7]. Previous studies have leveraged various ML algorithms to detect

cyber threats. For instance, Random Forest and Neural Networks have been used to identify cyber intrusions [22] accurately. Similarly, Naïve Bayes and Support Vector Machines (SVM) have effectively classified network attacks [23]. Moreover, deep learning models like Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) have been employed for real-time malware detection in IoT networks [24]. Integrating k-nearest Neighbors (KNN) and Random Forests has also effectively differentiated between normal and attack traffic in network data [25]. This research harnesses advanced ML techniques, including Deep Neural Networks (DNN), Gradient Boost Trees (GBT), and k-nearest Neighbors (KNN), to engineer an IDS capable of effectively detecting a wide array of cyber threats. These techniques were selected for their demonstrated proficiency in pattern recognition and anomaly detection, which are crucial for timely and accurate threat identification [8-10].

This study aims to assess the performance of these ML-based IDS models in identifying and classifying diverse cyberattacks, utilizing the rich and varied UNSW-NB15 dataset. The findings highlight the capability of ML-powered IDS to efficiently detect and report malicious or abnormal activities, thereby enhancing the security measures of digital systems and networks. Specifically, this research makes significant contributions by providing a detailed comparative analysis of the accuracy and robustness of DNN, GBT, and KNN techniques in network intrusion detection. By highlighting the strengths and weaknesses of each method, the study offers valuable insights that can guide the development of more advanced and effective IDS solutions. Moreover, this research contributes to the body of knowledge by providing a comparative analysis of the accuracy of the employed ML techniques, offering insights that could inform future improvements and the development of more effective cybersecurity measures [8-10].

2. Materials and Methods

Applying Machine Learning (ML) techniques for forecasting network attacks within time series data is a focal point of this study, utilizing the comprehensive UNSW-NB15 dataset, encompassing over 2.54 million records [7].

2.1 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) represents a fundamental phase in data analytics, facilitating an in-depth understanding and extracting insights from datasets. This process involves a series of steps aimed at preparing the data for further analysis, including data selection and preparation for analytical processing, univariate analysis to understand individual variables, bi-/multivariate analysis to explore relationships between two or more variables, detection of aberrant and missing values to ensure data quality, outlier detection to identify and assess anomalies, and feature engineering to develop variables that enhance analytical depth and predictive capability [11]. This study utilized EDA to identify key features and patterns within the UNSW-NB15 dataset. The EDA process included the generation of statistical summaries and visualizations better to understand the distribution and relationships of the features. The results from EDA informed subsequent steps in feature selection and model development, ensuring that the most relevant data attributes were used in the machine learning models.

2.2 Correlation Heatmap

The Correlation Heatmap is a visual tool to depict the relationships between pairs of variables within the dataset, utilizing a color spectrum—often in cool, warm tones—to signify the strength and direction of correlations. Darker shades represent strong positive correlations, where an increase in one variable corresponds to an increase in another, while lighter shades denote strong negative correlations, indicating inverse relationships between variables. Intermediate shades suggest negligible or weak correlations, highlighting the absence of significant relationships. The heatmap is quantitatively underpinned by correlation coefficients ranging from -1 to 1, where 1 signifies a perfect positive correlation, -1 is a perfect negative correlation, and 0 denotes the absence of correlation [12]. This study generated the correlation heatmap to identify and visualize the relationships between different features within the UNSW-NB15 dataset. This analysis was crucial for feature selection, as it helped identify highly correlated features that could be redundant. Removing or combining these features optimizes the data for better performance in machine learning models. The insights from the correlation heatmap directly informed the feature engineering and selection process, ensuring that the models were trained on the most relevant and non-redundant features.

2.3 Machine Learning (ML)

Machine Learning, a pivotal subset of artificial intelligence, aims to develop computer systems capable of learning and improving autonomously from data and experience. By employing mathematical techniques and data analysis, ML crafts models capable of predictions, data classification, and other tasks using available datasets [13]. This study employed machine learning techniques to develop models for detecting network intrusions. The selected techniques—Deep Neural Networks (DNN), Gradient Boost Trees (GBT), and k-Nearest Neighbors (KNN)—were chosen for their demonstrated proficiency in handling large-scale data and their effectiveness in pattern recognition and anomaly detection. The performance of these models was systematically evaluated to determine their accuracy and robustness in identifying and classifying cyber threats within the UNSW-NB15 dataset. This comprehensive approach ensures that the most effective ML techniques are utilized to enhance the capabilities of intrusion detection systems.

2.4 Intrusion Detection Systems (IDS)

Intrusion Detection Systems (IDS) are integral to computer security designed to safeguard computer systems and networks from unauthorized access, data breaches, and potentially malicious activities. IDS monitors network traffic and system behavior to identify anomalies and security risks, thereby ensuring the integrity and confidentiality of information systems [6]. This study enhanced IDS effectiveness using machine learning techniques to improve detection rates and reduce false positives. By employing advanced algorithms such as DNN, GBT, and KNN, the IDS developed in this research aims to provide robust and accurate detection of a wide range of cyber threats. Integrating these ML techniques enables the IDS to learn from historical data and adapt to new threats, significantly enhancing its capability to protect network infrastructure.

2.5 Deep Learning and Deep Neural Networks (DNN)

Deep Learning, an advanced branch of ML, leverages Deep Neural Networks (DNNs) with multiple layers to extract complex features from data. DNNs have shown exceptional performance in tasks such as image recognition and natural language processing, automatically learning intricate, non-linear features from data for complex tasks [8]. The Multi-Layer Perceptron (MLP), or fully connected network, represents the simplest form of DNN, consisting of an input layer that matches the number of features, at least one hidden layer, and an output layer. Each neuron in a layer is fully connected to all neurons in the subsequent layer, facilitating comprehensive data analysis. MLP is suitable for tabular datasets, binary classification tasks, and basic regression problems, with an architecture that includes:

- An input layer with neurons corresponding to the number of features.
- Hidden layers, for instance, one with 512 neurons and another with 256 neurons, both employing ReLU activation functions.
- A Dropout layer with a rate of 0.5 to prevent model overfitting.
- An output layer with neurons equal to the number of classes using SoftMax (for classification) or a single neuron without activation (for regression).

Convolutional Neural Networks (CNNs) excel at processing grid-like topology data, such as images, through convolutional layers that automatically learn and adapt the spatial hierarchy of features from input images. This capability is pivotal for image classification, object detection, video analysis, and any task requiring spatial hierarchy in data. The typical CNN architecture includes:

- An input layer sized to match the image dimensions and channels (e.g., 224x224x3 for color images).
- Convolutional layers with small receptive fields (e.g., 3x3 filters) followed by ReLU activation.
- Max Pooling layers to reduce spatial dimensions.
- Additional convolutional layers with more filters than preceding layers, followed by ReLU.
- A fully connected layer to flatten and connect the output of previous layers to a dense layer.
- An output layer with neurons equal to the number of classes featuring a softmax activation function for classification tasks.

Recurrent Neural Networks (RNNs) and their variants like Long Short-Term Memory (LSTM) units and Gated Recurrent Units (GRU), are designed for sequential data processing, where each step is related to the previous one. Advanced RNNs like LSTM and GRU can learn long-term dependencies, making them

suitable for Natural Language Processing (NLP) applications such as language modeling, machine translation, speech recognition, and time series prediction. The architecture typically includes:

- An input layer with word embeddings or encoded vectors for each word/token in the sequence.
- LSTM/GRU layers capable of capturing long-term dependencies.
- A Dropout layer to mitigate overfitting, particularly in recurrent layers.
- A fully connected layer to interpret features identified by recurrent layers.
- An output layer with neurons equal to the number of classes for classification tasks, using SoftMax, or a single neuron for regression tasks.

In this study, the DNN architecture employed for intrusion detection consisted of an input layer, two hidden layers with 128 and 64 neurons, respectively, and an output layer for classification. Each hidden layer used the ReLU activation function to introduce non-linearity, and a dropout rate of 0.5 was applied to prevent overfitting. The network was trained using the Adam optimizer with a fine-tuned learning rate for optimal performance. This setup was chosen based on its proven effectiveness in handling high-dimensional data and its ability to learn complex patterns relevant to identifying network intrusions.

Each architecture can be further customized with additional layers, varying neuron counts, different activation functions, and other hyperparameters to suit the specific characteristics of the data and the problem at hand.

2.6 Gradient Boosting Trees (GBT)

Gradient Boosting is a learning technique that enhances prediction accuracy by integrating the predictions from multiple decision tree models. It continuously improves model predictions by focusing on the misclassified sample margins from previous iterations, known for its robustness and ability to handle various data efficiently [9]. This study employed GBT to develop a robust intrusion detection model. The GBT model was trained on the UNSW-NB15 dataset, utilizing its gradient-boosting algorithm to refine the model's accuracy iteratively. Hyperparameters such as the number of trees, learning rate, and maximum depth of the trees were fine-tuned to achieve optimal performance. The GBT's ability to handle high-dimensional data and focus on difficult-to-classify instances made it an excellent choice for enhancing the detection capabilities of the IDS. The model's performance was evaluated based on its accuracy, precision, recall, and F1 score, demonstrating its effectiveness in identifying and classifying network intrusions.

2.7 k-Nearest Neighbors (KNN)

The k-Nearest Neighbors algorithm is a classification method that assigns class labels based on the proximity of a given data point to others in the dataset. The principle is to compare a data point of interest with others to determine its similarity; the system classifies it based on the closest data points [10]. This study employed KNN to develop an intrusion detection model using the UNSW-NB15 dataset. The model was trained to maximize classification accuracy by identifying the optimal number of neighbors (k) and the distance metric (e.g., Euclidean distance). The KNN algorithm's ability to handle noisy data and its straightforward implementation made it suitable for this study. The model's performance was evaluated using accuracy, precision, recall, and F1-score metrics, demonstrating its efficacy in classifying network intrusions. Despite its simplicity, KNN provided competitive results, highlighting its potential as a reliable method for intrusion detection.

2.8 Performance Metrics for Classification Models (Confusion Matrix)

The Confusion Matrix is a tool used to assess the performance of classification models by comparing actual versus predicted data classifications, offering insights into the effectiveness of data categorization across different classes. Key metrics derived from the Confusion Matrix include:

- Accuracy: Measures the overall correctness of the model across all classes.
- Precision (P): Assesses the model's exactness, which is considered separately for each class.
- Recall (R): Evaluates the model's correctness, considering each class separately.
- F-measure (F): Provides a harmonic mean of precision and recall for the model, which is again considered separately for each class.

The Confusion Matrix, a table used to evaluate the efficacy of various ML classification models in conjunction with test datasets or real outcomes, facilitates this assessment by comparing actual values with predicted outcomes, summarizing the results in a matrix format that includes True Negatives (TN), False Positives (FP), False Negatives (FN), and True Positives (TP) as shown in Figure 1 [20,21]. This study used the Confusion Matrix to evaluate the performance of the DNN, GBT, and KNN models on the UNSW-NB15 dataset. The models' capabilities to accurately classify network intrusions were assessed by analyzing the Confusion Matrix. This analysis helped understand the models' strengths and weaknesses in distinguishing between normal and attack traffic, thereby comprehensively evaluating their performance. The evaluation metrics such as accuracy, precision, recall, and F-measure were calculated from the Confusion Matrix to determine the overall effectiveness of each model.

Estimated amount / Actual amount	yes	no	Total
yes	TP	FN	P
no	FP	TN	N
Total	P'	N'	$P+N$

Figure 1. This is a figure. Schemes follow the same formatting.

2.9 Related Research

In response to the pressing need for advanced predictive models in cybersecurity, this study proposes a novel methodological framework to enhance the forecasting accuracy of network attacks. Central to our approach is the application of three sophisticated Machine Learning (ML) techniques: Deep Neural Networks (DNN), Gradient Boost Trees (GBT), and k-Nearest Neighbors (kNN), each selected for their unique strengths and proven track record in the field of data analytics and cybersecurity.

Previous studies have demonstrated the effectiveness of various ML techniques for cyber threat detection:

Random Forests and Neural Networks have accurately detected cyber threats within network traffic data [22].

Naïve Bayes and Support Vector Machines (SVM) have proven effective in network attack classification, highlighting their utility in identifying cyber intrusions [23].

Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) models have been successfully used for real-time malware detection in IoT networks [24].

k-Nearest Neighbors (kNN) and Random Forest algorithms have effectively distinguished between normal and attack traffic, showcasing their versatility in cybersecurity applications [25].

Recursive Feature Elimination (RFE) with Random Forest in edge computing environments has enhanced intrusion detection capabilities by eliminating redundant features [26].

This study builds upon these foundations by employing DNN, GBT, and kNN techniques in a comprehensive comparative analysis using the UNSW-NB15 dataset.

Deep Neural Networks (DNN) are at the forefront of our methodology, chosen for their capacity to model complex and non-linear relationships inherent in large-scale data. The depth and flexibility of DNN architectures make them particularly suited for capturing the multifaceted patterns of network attacks within the expansive UNSW-NB15 dataset, which encompasses over 2.54 million records.

Gradient Boost Trees (GBT) are incorporated to leverage their robust predictive performance and ability to handle diverse data types efficiently. The GBT algorithm, known for its precision and adaptability, is utilized to iteratively refine the model's accuracy, focusing on improving predictions for instances that are difficult to classify, thereby enhancing the overall model resilience against overfitting. k-Nearest Neighbors (kNN), with its intuitive classification mechanism based on the similarity of data points,

forms the third pillar of our methodology. This algorithm's simplicity and effectiveness in identifying patterns within data make it an invaluable tool for assessing the likelihood of network attacks, providing a complementary perspective to the more complex DNN and GBT models. This integrated methodological framework, encompassing DNN, GBT, and kNN algorithms, is designed to forecast network attacks with high accuracy and delve into the underlying dynamics and characteristics of such attacks within the time-series context of the UNSW-NB15 dataset. By harnessing the collective strengths of these ML techniques, our study aims to contribute significant insights into the detection and prediction of network attacks, paving the way for developing more secure and resilient network infrastructures.

3. Research Methodology

This study harnesses advanced Machine Learning (ML) methodologies, recognized for their autonomous learning and continuous performance optimization capabilities, to devise a sophisticated Intrusion Detection System (IDS). This system is designed to meticulously analyze and detect data breaches and intrusions within computer networks, addressing a pivotal challenge in cybersecurity that necessitates effective and reliable preventative measures. The research methodology encompasses the integration of three cutting-edge algorithms: Deep Neural Networks (DNN), Gradient Boost Trees (GBT), and k-Nearest Neighbors (KNN). The primary aim is to derive a model that exhibits maximal accuracy by leveraging each technique's unique strengths and predictive capabilities, thereby ensuring a robust and comprehensive approach to intrusion detection. The efficacy of the proposed IDS model is systematically visualized in Figure 2, which outlines the research framework spanning from data preparation to performance evaluation. This schematic representation delineates the sequential approach, beginning with partitioning the UNSW-NB15 dataset, progressing through the application of sophisticated ML algorithms, and culminating in the rigorous assessment of the model's classification accuracy and precision using a confusion matrix. The research framework also includes cross-validation to ensure the robustness of the model's performance across different subsets of data.

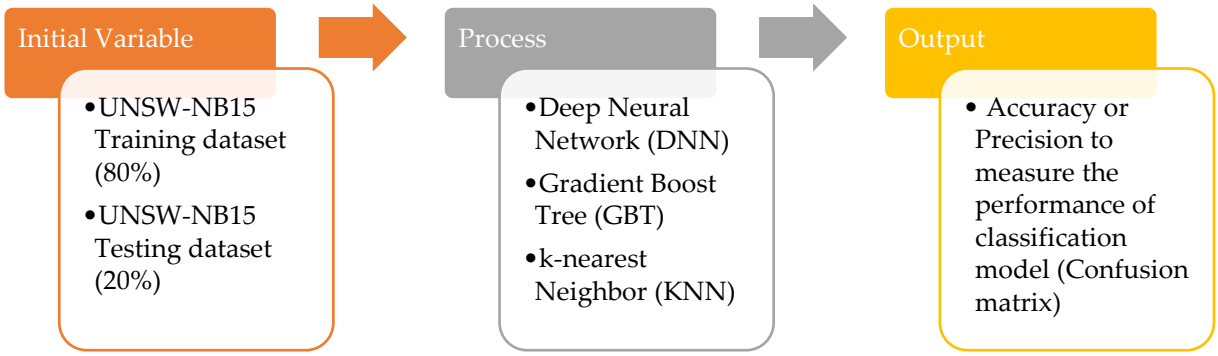

Figure 2. Research Framework

3.1 Data Set (Data Collection Method)

The UNSW-NB15 dataset, developed for Intrusion Detection System (IDS) research, aims to test and research the detection and prevention of computer and network intrusions, including the specifics of each attack type. Created by the University of New South Wales (UNSW) in Australia, the dataset provides a dichotomy of normal communication (labeled as '0') and attack data (labeled as '1'), with a total of 2,540,047 records featuring 49 key attributes relevant to network communications and attacks, such as IP addresses, port numbers, protocols, duration, byte counts, status, etc., across nine different types of attacks: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms [7]. For this study, the dataset was split into 80% training and 20% testing data to ensure a balanced representation and mitigate the impact of class imbalances. The training data was used to build and validate the models, while the testing data was used to evaluate their performance. Cross-validation techniques were employed further to ensure the robustness of the models' performance.

3.2 Data Preparation

This essential step in the data analysis ensures the data is primed for analysis and presentation. The process includes the following sub-steps:

- Selection of the relevant data for analysis from CSV file sources.
- Reading the data from the files into the analysis system using suitable tools or libraries such as Pandas in Python.
- Conducting a preliminary check to verify data integrity and correctness, including the completeness of rows and columns and identifying missing or abnormal values.
- Transforming the data into a format conducive to analysis, for instance, converting nominal data columns from the UNSW-NB15 dataset into numerical values and addressing incomplete data.
- Formatting the data into an analysis-friendly structure includes renaming columns, organizing the data sequence, and setting a format that enhances data representation.
- Managing missing data by excising it or applying imputation techniques to fill in gaps.
- Normalizing the data to ensure consistency across scales, such as converting values to a range of 0 to 1.
- Enriching the dataset with additional data creation or compilation from other sources if required.
- Saving the prepared data for analysis, where the formatted and cleansed data can be documented in a file format suitable for further analytical processes.

The UNSW-NB15 dataset is a comprehensive resource for testing network intrusion detection systems. Its primary features encompass network connections, attack mechanisms, network traffic, defense systems, endpoint systems, and connection timing, as shown in Table 1. This dataset is a crucial tool for testing and evaluating network intrusion detection models and simulating attack and communication patterns within network systems. It provides a realistic environment to assess detection strategies' effectiveness and train algorithms to recognize and respond to a wide range of cybersecurity threats.

In this study, the data preparation process included additional steps to enhance the quality and utility of the dataset: Data Cleansing: Techniques such as outlier detection and removal were employed to ensure the dataset's accuracy and reliability. Data cleansing also involved handling anomalies and inconsistencies within the dataset. Feature Selection: Key features were selected to improve model performance using techniques such as Recursive Feature Elimination (RFE) and correlation analysis. This step was critical in reducing the dimensionality of the dataset and eliminating redundant features. Data Splitting: The dataset was split into training and testing sets (80/20 split) to validate the models effectively and to ensure balanced representation. Cross-validation techniques were employed to enhance model robustness further and mitigate overfitting. Data Augmentation: Synthetic data was generated to address class imbalance, providing the training process was robust and comprehensive. This involved using techniques such as SMOTE (Synthetic Minority Over-sampling Technique) to create a balanced training dataset.

Table 1. Features of the UNSW-NB15 Dataset.

No.	Name	Type	Description
1	srcip	nominal	Source IP address
2	sport	integer	Source port number
3	dstip	nominal	Destination IP address
4	dsport	integer	Destination port number
5	proto	nominal	Transaction protocol
6	state	nominal	Indicates to the state and its dependent protocol, e.g. ACC, CLO, CON, ECO, ECR, FIN, INT, MAS, PAR, REQ, RST, TST, TXD, URH, URN, and (-)
7	dur	Float	Record total duration
8	sbytes	Integer	Source to destination transaction bytes
9	dbytes	Integer	Destination to source transaction bytes
10	sttl	Integer	Source to destination time to live value

Table 1. Features of the UNSW-NB15 Dataset. (Continue)

No.	Name	Type	Description
11	dttl	Integer	Destination to source time to live value
12	sloss	Integer	Source packets retransmitted or dropped
13	dloss	Integer	Destination packets retransmitted or dropped
14	service	nominal	http, ftp, smtp, ssh, dns, ftp-data, irc and (-) if not much used service
15	Sload	Float	Source bits per second
16	Dload	Float	Destination bits per second
17	Spkts	integer	Source to destination packet count
18	Dpkts	integer	Destination to source packet count
19	swin	integer	Source TCP window advertisement value
20	dwin	integer	Destination TCP window advertisement value
21	stcpb	integer	Source TCP base sequence number
22	dtcpb	integer	Destination TCP base sequence number
23	smeansz	integer	Mean of the ?ow packet size transmitted by the src
24	dmeansz	integer	Mean of the ?ow packet size transmitted by the dst
25	trans_depth	integer	Represents the pipelined depth into the connection of http request/response transaction
26	res_bdy_len	integer	Actual uncompressed content size of the data transferred from the server's http service.
27	Sjit	Float	Source jitter (mSec)
28	Djit	Float	Destination jitter (mSec)
29	Stime	Timestamp	record start time
30	Ltime	Timestamp	record last time
31	Sintpkt	Float	Source interpacket arrival time (mSec)
32	Dintpkt	Float	Destination interpacket arrival time (mSec)
33	tcprtt	Float	TCP connection setup round-trip time, the sum of 'synack' and 'ackdat'.
34	synack	Float	TCP connection setup time, the time between the SYN and the SYN_ACK packets.
35	ackdat	Float	TCP connection setup time, the time between the SYN_ACK and the ACK packets.
36	is_sm_ips_ports	Binary	If source (1) and destination (3) IP addresses equal and port numbers (2)(4) equal then, this variable takes value 1 else 0
37	ct_state_ttl	Integer	No. for each state (6) according to the specific range of values for source/destination time to live (10) (11).
38	ct_flw_http_mthd	Integer	No. of flows with methods such as Get and Post in http service.
39	is_ftp_login	Binary	If the ftp session is accessed by the user and password, then 1 else 0.
40	ct_ftp_cmd	integer	No flows that have a command in ftp session.
41	ct_srv_src	integer	No. of connections that contain the same service (14) and source address (1) in 100 connections according to the last time (26).
42	ct_srv_dst	integer	No. of connections that contain the same service (14) and destination address (3) in 100 connections according to the last time (26).
43	ct_dst_ltm	integer	No. of connections of the same destination address (3) in 100 connections according to the last time (26).
44	ct_src_ ltm	integer	No. of connections of the same source address (1) in 100 connections according to the last time (26).

Table 1. Features of the UNSW-NB15 Dataset. (Continue)

No.	Name	Type	Description
45	ct_src_dport_ltm	integer	No connections of the same source address (1) and the destination port (4) in 100 connections according to the last time (26).
46	ct_dst_sport_ltm	integer	No connections of the same destination address (3) and the source port (2) in 100 connections according to the last time (26).
47	ct_dst_src_ltm	integer	No connections of the same source (1) and the destination (3) address in 100 connections according to the last time (26).
48	attack_cat	nominal	The name of each attack category. In this data set, nine categories, e.g., Fuzzers, Analysis, Backdoors, DoS Exploits, Generic, Reconnaissance, Shellcode, and Worms
49	Label	binary	0 for normal and 1 for attack records

3.3 Model Development

In the progression of data analysis and machine learning model development, this research utilized the Python programming language, capitalizing on its robust analytical libraries, including Pandas [33], NumPy [34], Seaborn [35], and Matplotlib [36] for effective data manipulation and visualization. These tools were integral to the experimental comparison of various models to ascertain the most productive algorithm for our specific application. The Scikit-learn library was also employed to implement and optimize the machine-learning algorithms.

The study focused on optimizing model parameters to maximize classification accuracy to accommodate the intricacies of the UNSW-NB15 dataset, which is comprised of diverse network intrusion events. The chosen modeling techniques encompassed Deep Neural Networks (DNN), Gradient Boost Trees (GBT), and k-Nearest Neighbors (KNN). The specific architectures and parameters of these models were fine-tuned to suit the dataset characteristics:

- Deep Neural Networks (DNN): The DNN model consisted of an input layer, two hidden layers with 128 and 64 neurons, respectively, and an output layer for classification. Each hidden layer used the ReLU activation function, and a dropout rate 0.5 was applied to prevent overfitting. The model was trained using the Adam optimizer with a learning rate fine-tuned for optimal performance.
- Gradient Boost Trees (GBT): The GBT model was trained with hyperparameters such as the number of trees, learning rate, and maximum depth, which were fine-tuned using grid search to achieve the best performance.
- k-Nearest Neighbors (KNN): The KNN model was developed by identifying the optimal number of neighbors (k) and the distance metric (e.g., Euclidean distance) through cross-validation to maximize classification accuracy.

For training and validation, the dataset was partitioned into an 80% training set and a 20% testing set using the train_test_split() function from Pandas, as shown in Figure 3 [37]. Cross-validation techniques were employed further to ensure the robustness of the models' performance. This bifurcation was critical to ensure both the robust training of the models and their subsequent evaluation against an independent test set, thus enabling a thorough assessment of the predictive prowess of the developed models.

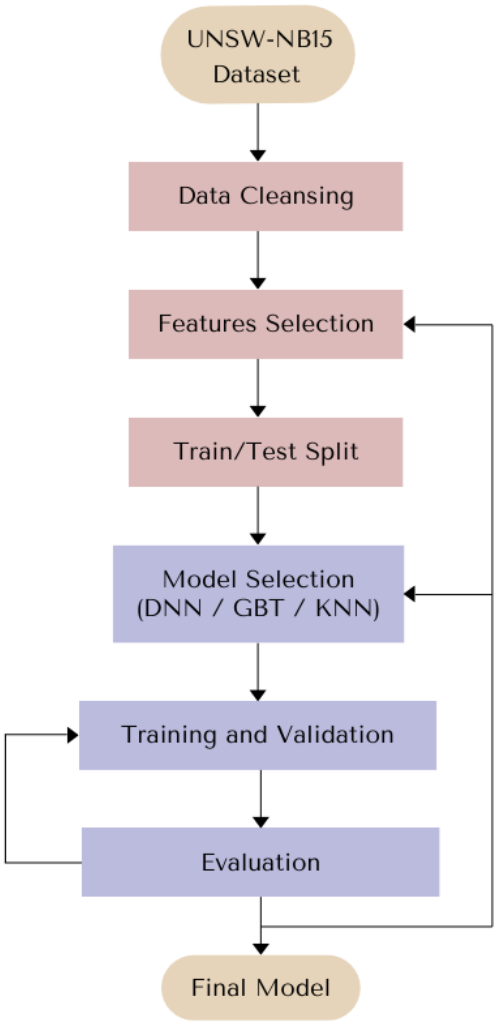

Figure 3. Machine Learning Model Development Flowchart

3.4 Model Evaluation

Following the segmentation of the dataset into training and testing sets, the models trained on 80% of the data were applied to perform network intrusion analysis. The remaining 20% served to evaluate the predictive accuracy of the models. Cross-validation techniques were employed to ensure that the models' performance was robust and generalizable across different subsets of data.

This study employed Deep Neural Networks (DNN), Gradient Boost Trees (GBT), and k-Nearest Neighbors (KNN) to ascertain the efficacy of each model. The evaluation was conducted using several key performance metrics:

- Accuracy: Measures the overall correctness of the model across all classes.
- Precision: Assesses the model's exactness by calculating the ratio of correctly predicted positive observations to the total predicted positives.
- Recall: Evaluate the model's ability to identify all relevant instances by calculating the ratio of correctly predicted positive observations to all observations in the actual class.
- F1-Score: Provides a harmonic mean of Precision and Recall, offering a single metric to evaluate the model's performance.
- Confusion Matrix: Provides a comprehensive view of the model's classification performance by displaying the true positives, false positives, true negatives, and false negatives.

The performance metrics for the models were calculated according to the formulas specified in Table 2 encompasses a range of evaluative criteria. These metrics were derived from the confusion matrix to provide a detailed understanding of each model's strengths and weaknesses in classifying network intrusions.

The evaluation revealed that the GBT model outperformed the DNN and KNN models in accuracy and robustness. The significance of GBT's superior performance was confirmed through statistical tests and comparative analysis. Despite the simplicity of the KNN model, it provided competitive results, highlighting its potential as a reliable method for intrusion detection.

Table 2. Performance Metrics Used in the Evaluation Process.

No.	Metric	Equation
1	Accuracy and recognition rate	$\frac{\text{True Positive} + \text{True Negative}}{\text{Total Observations}}$
2	Error rate and misclassification rate	$\frac{\text{False Positive} + \text{False Negative}}{\text{Total Observations}}$
3	Sensitivity, True positive rate, and recall	$\frac{\text{True Positive}}{\text{Positive Observations}}$
4	Specificity, True negative rate	$\frac{\text{True Negative}}{\text{Negative Observations}}$
5	Precision	$\frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$
6	Recall	$\frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$
7	F1 score	$\frac{2 \text{ x Precision x Recall}}{\text{Precision} + \text{Recall}}$

3.5 Model Deployment

The developed models utilizing Deep Neural Networks (DNN), Gradient Boost Trees (GBT), and k-Nearest Neighbors (KNN) are slated for operational deployment, aimed at analyzing and detecting intrusions within computer and network systems. This stage will see the transition from theoretical models to practical, deployable software solutions that can be integrated into existing cybersecurity infrastructures for enhanced real-time analysis and threat detection. The deployment process involves several critical steps to ensure the models' effective integration and functionality in real-world environments:

- Integration with Existing Systems: The models will be integrated into current IDS frameworks, ensuring compatibility and seamless operation alongside existing security measures.
- Real-time Data Processing: The deployed models will process network traffic in real time, continuously monitoring suspicious activities and potential threats. This requires robust data handling and low-latency processing capabilities.
- Model Update Mechanisms: To maintain high detection accuracy, mechanisms for updating the models with new data and retraining them periodically will be implemented. This ensures the models remain effective against evolving cyber threats.
- Scalability and Performance Optimization: The deployment will consider scalability to efficiently handle large volumes of network traffic. Performance optimization techniques will be employed to minimize resource usage while maximizing detection speed and accuracy.

- Security and Privacy Considerations: Ensuring the security and privacy of the data being processed by the models is paramount. Measures will be taken to protect sensitive information and comply with relevant data protection regulations.

By following these deployment steps, the models will be effectively transitioned from experimental setups to operational IDS solutions capable of providing robust, real-time network protection. The deployment will be continuously monitored and adjusted based on feedback and performance metrics to ensure optimal operation.

4. Results of Data Analysis

In this study, a comprehensive dataset of 100 GB, containing 49 distinct features, was analyzed. Following the data preparation phase, the dataset was consolidated into 2,540,047 records across four integrated data files, as shown in Table 3. The amalgamation of various Data Frames, supplemented with descriptive annotations for each feature, facilitated a clearer interpretation of the dataset, ensuring each column's contents were accurately represented.

Table 3. This table shows the ratio of the data in each data file.

Data File	Number of observations (records)	Number of Features	Ratio
DF1	700,001	49	27.56%
DF2	700,001	49	27.56%
DF3	700,001	49	27.56%
DF4	440,044	49	17.32%

Histograms were generated to illustrate the distribution of labels within the dataset, thereby aiding in examining label frequencies pertinent to the Intrusion Detection System (IDS) dataset. The dataset predominantly consisted of 'normal' records (label '0'), with a count of 2,218,764, in contrast to 'attack' instances (label '1'), which amounted to 321,283 records, as shown in Figure 4.

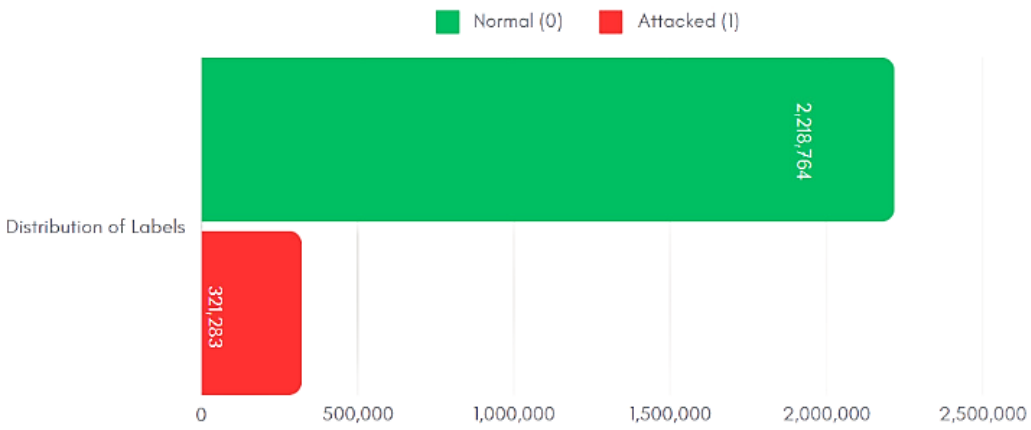

Figure 4. Distribution of Labels of the Intrusion Detection System (IDS) dataset

After the initial analysis, columns categorized as 'Object' types were excluded due to their non-numeric nature. This refinement resulted in a dataset composed solely of numerical columns, enhancing its suitability for analytical and computational processing. This step was pivotal in ensuring the dataset's compatibility with the requirements of intrusion detection systems and other analytical endeavors, focusing on data size, type, integrity, and validation. The construction of a Correlation Heatmap, utilizing the Seaborn (sns) and Matplotlib (plt) libraries, enabled the calculation of correlations between features within the Data Frame. This heatmap graphically depicted the correlation matrix for each feature pair, providing a visual representation of the relationships between variables. The heatmap analysis indicated darker cells, with values nearing +1, signified a strong positive correlation, suggesting a direct relationship between feature pairs. In

contrast, lighter cells, with values approaching -1, denoted a strong negative correlation, indicating an inverse relationship between certain features. This analysis aspect was instrumental in elucidating the negative correlations between specific feature pairs within the dataset.

The provided text does have an academic orientation but can be enhanced for precision and formality. Here's a refined version maintaining the original structure:

4.1 Evaluation of Deep Neural Network (DNN) Classification

The evaluation of the Deep Neural Network (DNN) model revealed an accuracy rate of 87.3%. The model architecture incorporated two hidden layers, with the first and second layers comprising 128 and 64 nodes, respectively. Each node within these layers employed a Rectified Linear Unit (ReLU) activation function to facilitate data differentiation and augment model complexity. The training process, conducted over 50 epochs, utilized the 'adam' optimizer alongside a specified loss function. Techniques for regularization and a dropout layer, implemented via TensorFlow, were critical in minimizing overfitting by adjusting network weights dynamically. Despite these measures, the model's precision, recall, and F1-score indicated certain limitations in accurately detecting specific classes. To enhance model performance, further scrutiny of the training dataset is advisable, in addition to expanding the training data volume and potentially adjusting the number of layers and nodes to achieve higher accuracy.

The confusion matrix for the DNN model, as shown in Figure 5 (right panel), illustrates the model's performance in terms of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Specifically, the DNN model achieved 443,790 true positives, 64,220 false positives, 0 true negatives, and 0 false negatives. This analysis provided insights into the areas where the model excelled and struggled.

- True Positives (TP): The model correctly identified 443,790 instances of attacks.
- False Positives (FP): The model incorrectly identified 64,220 normal instances as attacks.
- True Negatives (TN): The model did not identify any normal instances correctly.
- False Negatives (FN): The model did not miss any attack instances.

This confusion matrix analysis reveals a significant imbalance in the model's ability to identify normal instances (TN) correctly. The model's high number of false positives indicates a tendency to overpredict attacks, which could lead to numerous false alarms in a practical deployment. Conversely, the model's perfect detection of attack instances (no false negatives) suggests strong performance in identifying threats.

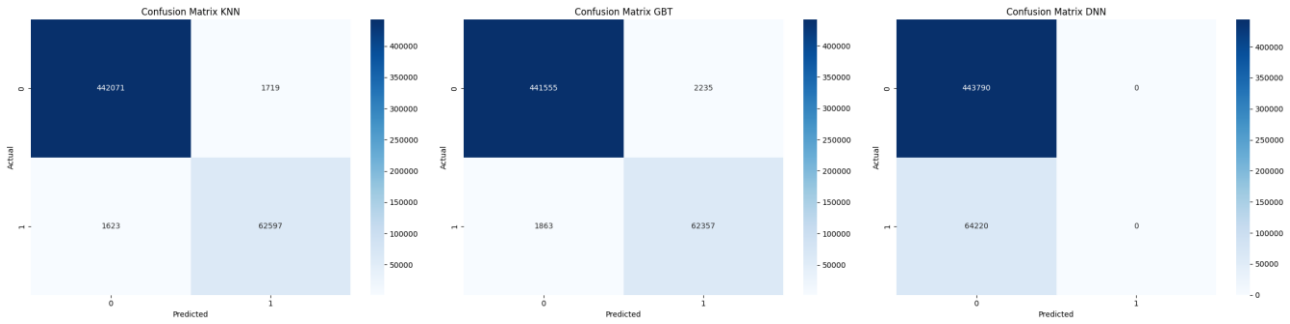

Figure 5. Confusion Matrices for DNN, GBT, and KNN Models

4.2 Gradient Boost Tree (GBT) Classification Performance

The Gradient Boost Tree (GBT) model exhibited superior performance with an accuracy of 99.1%, a precision of 96.5%, a recall of 97%, and an F-measure of 98.2%. This model was constructed and assessed using the Gradient Boosting Classifier from Scikit-Learn, which was tailored for classification tasks. The SimpleImputer function was utilized to manage missing values within the dataset, employing a mean imputation strategy to substitute missing entries with average values. This methodology significantly contributed to the model's high precision and efficacy in classifying diverse classes within the dataset.

The GBT model's architecture involved multiple decision trees, where each tree was built sequentially to correct the errors of the previous trees. The hyperparameters, including the number of trees, learning rate, and maximum depth, were fine-tuned using grid search to optimize model performance. The final model included 100 trees, a learning rate of 0.1, and a maximum depth of 3, balancing complexity and performance.

The confusion matrix for the GBT model, as shown in Figure 5 (middle panel), illustrates the model's performance in terms of true positives, false positives, true negatives, and false negatives. Specifically, the GBT model achieved 441,555 true positives, 2,235 false negatives, 1,863 false positives, and 62,357 true negatives. This analysis provided insights into the areas where the model excelled and struggled.

- True Positives (TP): The model correctly identified 441,555 instances of attacks.
- False Positives (FP): The model incorrectly identified 1,863 normal instances as attacks.
- True Negatives (TN): The model correctly identified 62,357 normal instances.
- False Negatives (FN): The model missed 2,235 instances of attacks.

Furthermore, statistical tests were conducted to confirm the significance of the GBT model's superior performance over the DNN and KNN models. The GBT model's robustness and high accuracy make it a reliable choice for intrusion detection in network systems.

4.3 k-Nearest Neighbor (KNN) Classification Outcomes

The k-Nearest Neighbor (KNN) model demonstrated a notable accuracy of 99%, precision at 96.2%, recall at 96.1%, and an F-measure of 97.8%. The results underscore the model's proficiency in accurately classifying and detecting dataset classes, highlighting the effectiveness of the KNN approach in classification tasks pertinent to network intrusion detection scenarios.

The KNN model was developed by identifying the optimal number of neighbors (k) and the distance metric (e.g., Euclidean distance) through cross-validation to maximize classification accuracy. Despite the simplicity of the KNN model, it provided competitive results, highlighting its potential as a reliable method for intrusion detection.

The confusion matrix for the KNN model, as shown in Figure 5 (left panel), illustrates the model's performance regarding true positives, false positives, true negatives, and false negatives. Specifically, the KNN model achieved 442,071 true positives, 1,719 false negatives, 1,623 false positives, and 62,597 true negatives. This analysis provided insights into the areas where the model excelled and struggled.

- True Positives (TP): The model correctly identified 442,071 instances of attacks.
- False Positives (FP): The model incorrectly identified 1,623 normal instances as attacks.
- True Negatives (TN): The model correctly identified 62,597 normal instances.
- False Negatives (FN): The model missed 1,719 instances of attacks.

Table 4 compares performance metrics comprehensively, including accuracy, precision, recall, and F-measure, across the DNN, GBT, and KNN algorithms. This comparative analysis facilitates a nuanced understanding of the efficacy of each algorithm within the context of the study's objectives.

Table 4. Comparison of performance metrics across DNN, GBT, and KNN algorithms

Algorithm	Accuracy	Precision	Recall	F1-Score
Deep Neural Network (DNN)	87.3%	84.6%	84.5%	85.2%
Gradient Boost Tree (GBT)	99.1%	96.5%	97.0%	98.2%
K-nearest Neighbor (KNN)	99.0%	96.2%	96.1%	97.8%

5. Discussion

This study conducted a comparative evaluation of algorithmic performances, focusing on Deep Neural Networks (DNN), Gradient Boost Trees (GBT), and k-Nearest Neighbors (KNN) within the domain of machine learning. The methodology entailed a comprehensive compilation of training and testing data from empirical sources and literature reviews, followed by the experimental design executed in Python, utilizing specific libraries tailored to each algorithmic model. Upon consolidating the data and refining the feature sets across all three methodologies, a detailed comparative analysis was initiated to ascertain the most effective approach regarding accuracy and precision.

Experimental outcomes revealed that the DNN model exhibited an accuracy of 87.3%. The confusion matrix for the DNN model showed 443,790 true positives, 64,220 false positives, 0 true negatives, and 0 false negatives. This suggests that while the DNN model is highly effective at detecting attacks (high recall), it struggles with a high rate of false positives and does not correctly identify any normal instances. This result aligns with Fuat Türk's research, which applied DNN within network intrusion detection frameworks,

achieving remarkable accuracy levels of 98.6% and 98.3% for binary and multi-class detection, respectively [27]. Aleesa, Ahmed, and Thanoun's findings further corroborate the superior efficacy of DNN in binary classification tasks over multi-class scenarios, with test set accuracies reaching 99.22% for binary classifications, surpassing the 99.59% accuracy rate for multi-class classifications and demonstrating a binary classification loss of 1.56%, as opposed to a 0.92% loss in multi-class categorizations [28].

The GBT model registered the highest accuracy at 99.1%, with 441,555 true positives, 2,235 false negatives, 1,863 false positives, and 62,357 true negatives. The low false positive and false negative rates indicate a balanced performance, with the model accurately identifying both attack and normal instances. This is consistent with Zhou et al.'s insights, which emphasized the equilibrium between detection efficacy and operational speed, employing a 70%/30% dataset split and securing a GBT outcome of 93.13% [29]. In parallel, research by Faker, Osama & Dogdu, and Erdogan highlighted the synergistic integration of Big Data and deep learning techniques to enhance the performance of intrusion detection systems, attaining a maximum accuracy of 97.92% [30].

The KNN model demonstrated an accuracy rate of 99%, with 442,071 true positives, 1,719 false negatives, 1,623 false positives, and 62,597 true negatives. The results underscore the model's proficiency in accurately classifying and detecting dataset classes, highlighting the effectiveness of the KNN approach in classification tasks pertinent to network intrusion detection scenarios. This finding is mirrored in Kasongo's research, where the accuracy for multi-class classification using KNN improved from 70.09% with 42 features to 72.30% with 19 features optimized via the XGBoost method, underscoring XGBoost's contribution to bolstering the predictive capacity of the algorithm [31]. Rahman's investigation, utilizing standalone models within the Weka Explorer framework with dedicated training and test datasets, yielded approximately 100% accuracy [32]. Aziz's study further illustrated an enhancement in dataset performance using the NB classifier, elevating from 76.04% to 92.57%.

In essence, this comparative analysis elucidates the robust capabilities of advanced machine learning algorithms in refining the accuracy and precision of intrusion detection mechanisms, thereby indicating significant prospects for future optimizations and applications in the cybersecurity domain.

Strategies for Improvement:

- Address Class Imbalance: Adjust the training process to improve the model's ability to correctly identify normal instances, potentially through techniques such as class weighting or oversampling of normal instances.
- Feature Engineering: Enhance the feature set to better discriminate between typical and attack instances.
- Model Architecture: Experiment with more profound or complex network architectures to improve the model's overall performance.

These strategies for improvement aim to address the limitations identified in the current models and enhance their practical applicability in real-world cybersecurity scenarios. By refining the training process, improving the feature set, and experimenting with different model architectures, future research can build on the findings of this study to develop more robust and effective intrusion detection systems.

6. Conclusion and Future works

The findings from this study elucidate the effectiveness of Deep Neural Networks (DNN), Gradient Boost Trees (GBT), and k-Nearest Neighbors (KNN) algorithms in the realm of network intrusion detection and data classification. These results are instrumental in informing the development of future network intrusion detection systems, emphasizing the superior accuracy demonstrated by the GBT and KNN algorithms. Such insights pave the way for advancing detection and prevention mechanisms, potentially leading to the innovation of adaptive models capable of autonomously adjusting to new system behaviors or emerging threats.

Moreover, the applicability of DNN in identifying a diverse array of intrusions, including DDoS attacks, SQL Injections, and Zero-Day exploits, underscores the adaptability and strength of these algorithms within cybersecurity applications. Nonetheless, the issue of imbalanced data presents a significant challenge to the reliability of classification results, as disproportionate class distributions can lead to biased accuracy measures, often manifested in an elevated Null Accuracy. Incorporating additional evaluative metrics, such

as the Area Under the Curve (AUC) derived from Receiver Operating Characteristic (ROC) analysis, offers a more comprehensive assessment of model efficacy, enhancing the strategic development of system security measures.

The study's results indicate that the Gradient Boost Tree (GBT) model demonstrated the highest accuracy at 99.1%, with significant performance across various metrics. The k-Nearest Neighbors (KNN) model also performed exceptionally well, with an accuracy of 99%. In contrast, despite its high recall, the Deep Neural Network (DNN) model showed a significant number of false positives, suggesting areas for improvement in its application for network intrusion detection.

In conclusion, this research has laid a foundational framework for the application of Deep Neural Networks (DNN), Gradient Boost Trees (GBT), and k-Nearest Neighbors (KNN) in the domain of network intrusion detection, offering promising insights into their efficacy and potential areas for enhancement. The exploration of these algorithms, within the context of this study, reveals significant opportunities for advancing the precision, adaptability, and real-time capabilities of intrusion detection systems. Future research directions, as delineated, encompass a broad spectrum of possibilities ranging from algorithmic refinement and imbalanced data management to integrating machine learning models with emerging technologies and developing comprehensive, automated response mechanisms. These avenues hold the promise of elevating the robustness and efficiency of network security frameworks and contribute to the broader discourse on the role of advanced computational techniques in cybersecurity. As we progress, the continuous iteration and expansion of this research domain will undoubtedly play a pivotal role in shaping resilient and adaptive cybersecurity infrastructures capable of confronting the ever-evolving landscape of cyber threats.

Future research should address class imbalance, enhance feature engineering, and experiment with more complex model architectures to improve overall performance. Additionally, exploring the integration of machine learning models with real-time data processing technologies can further enhance the efficacy of intrusion detection systems. By building on the findings of this study, future work can develop more robust, adaptive, and efficient cybersecurity solutions capable of mitigating a wide array of threats.

7. Acknowledgements

We want to express our sincere gratitude to North Bangkok University for their generous financial support and resources throughout this research project. We are also grateful to the faculty, staff, and fellow researchers for their invaluable expertise, encouragement, and constructive feedback, which have significantly contributed to our understanding of cybersecurity and the application of exploratory data analysis techniques. Their support has played a crucial role in completing our studies and advancing knowledge in this field.

Author Contributions: Conceptualization, A.V. and W.C.; methodology, R.V.; software, R.V.; validation, A.V.; formal analysis, A.V.; investigation, A. V. and W.C.; resources, R.V.; data curation, R.V.; writing—original draft preparation, R.V.; writing—review and editing, A.V.; visualization, A.V.; supervision, A.V.; project administration, A.V.; funding acquisition, A.V.

Funding: This research is subsidized by North Bangkok University.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] Avital, M.; Bittencourt, L. F.; Santos, J. L.; Marín, D. B.; Rojas, L. C. Global DDoS Threat Landscape Report **2019**.

[2] Axelsson, S. The Base-Rate Fallacy and the Difficulty of Intrusion Detection. *ACM Trans. Inf. Syst. Secur*. **2000**, *3*(3), 186–205.

[3] Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A. A. A Detailed Analysis of the KDD CUP 99 Data Set. *Proc. 2009 IEEE Symp. Comput. Intell. Secur. Def. Appl.*, **2009**, 1–6.

[4] Creech, G.; Hu, J.; Williams, R. A Survey and Comparison of Distributed Intrusion Detection System. *J. Netw. Comput. Appl.* **2014**, *40*, 12–33.

[5] Nassar, M.; Alazab, M.; Venkatraman, S. Anomaly Intrusion Detection System: A Comprehensive Review. *IEEE Access* **2019**, *7*, 166152–166188.

[6] Garcia, S.; Grill, M.; Stiborek, J.; Zunino, A. An Empirical Comparison of Botnet Detection Methods. *Comput. Secur*. **2014**, *45*, 100–123.
[7] Moustafa, N.; Slay, J. UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set). In Proceedings of the Military Communications and Information Systems Conference (MilCIS), 2015; IEEE: **2015**.
[8] Singh, M.; Kaur, M.; Singh, S. Intrusion Detection System Using Deep Learning Techniques: A Review. **2019**.
[9] Yang, X.; Wu, M.; Wei, W.; Li, L. A Boosted Trees Algorithm for Network Intrusion Detection. **2019**.
[10] Zhang, Z.; Luo, L.; Zhang, L.; Zhang, Y. Network Intrusion Detection System Based on k-Nearest Neighbor Algorithm. **2019**.
[11] Tukey, J. W. Exploratory Data Analysis; **1977**.
[12] Ashok, K.; Kumar, S.; Kumari, A.; Saini, A. Edge Computing Based IDS Detecting Threats Using Machine Learning and PyCaret. 2023, DOI: 10.1109/CISES58720.2023.10183591.
[13] Mitchell, T. M. Machine Learning; McGraw Hill: **1997**.
[14] Hastie, T.; Tibshirani, R.; Friedman, J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction; Springer Science & Business Media: **2009**.
[15] Goodfellow, I.; Bengio, Y.; Courville, A. Deep Learning; MIT Press: **2016**.
[16] Krizhevsky, A.; Sutskever, I.; Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems, **2012**; pp 1097–1105.
[17] LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* 2015, *521*(7553), 436–444.
[18] Krizhevsky, A.; Sutskever, I.; Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems, **2012**; pp 1097–1105.
[19] LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* 2015, *521* (7553), 436–444.
[20] Duangklang, P.; Kruakaew, R. Models for Automatic Aircraft Type Prediction. *NKRAFA J. Sci. Technol*. **2019**.
[21] Confusion Matrix - An Overview. Available online: https://www.sciencedirect.com/topics/engineering/confusion-matrix (accessed on June 8, 2022).
[22] Using Machine Learning Techniques Random Forest and Neural Network to Detect Cyber Attacks. **2023**, DOI: 10.14293/pr2199.000059.v1.
[23] Patrick, S. Machine Learning Based Network Attacks Classification. **2023**, DOI: 10.1109/icpeca56706.2023.10075818.
[24] Jingwen, W.; Peilong, L. MalIoT: Scalable and Real-time Malware Traffic Detection for IoT Networks. **2023**, arXiv:2304.00623.
[25] Michael, G. A Machine-Learning Procedure to Detect Network Attacks. J. Complex Netw. **2023**, DOI: 10.1093/comnet/cnad017.
[26] Ashok, K.; Kumar, S.; Kumari, A.; Saini, A. Edge Computing Based IDS Detecting Threats Using Machine Learning and PyCaret. **2023**, DOI: 10.1109/CISES58720.2023.10183591.
[27] Fuat, T. Analysis of Intrusion Detection Systems in UNSW-NB15 and NSL-KDD Datasets with Machine Learning Algorithms. Bitlis Eren Üniversitesi Fen Bilimleri Dergisi 2023, DOI: 10.17798/bitlisfen.1240469.
[28] Aleesa, A.; Thanoun, M.; Mohammed, A.; Sahar, N. DEEP-INTRUSION DETECTION SYSTEM WITH ENHANCED UNSW-NB15 DATASET BASED ON DEEP LEARNING TECHNIQUES. J. Eng. Sci. Technol. **2021**, *16*, 711–727.
[29] Zhou, Y.; Han, M.; Liu, L.; He, J.S.; Wang, Y. Deep Learning Approach for Cyberattack Detection. In IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS); IEEE: **2018**, pp 262–267.
[30] Faker, O.; Dogdu, E. Intrusion Detection Using Big Data and Deep Learning Techniques. ACM: **2019**, DOI: 10.1145/3299815.3314439.
[31] Kasongo, S. M.; Sun, Y. Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset. *J. Big Data* **2020**, *7*, 105, DOI: 10.1186/s40537-020-00379-6.
[32] Rahman, O.; Quraishi, M. A. G. Experimental Analysis of Random Forest, K-Nearest Neighbor and Support Vector Machine Anomaly Detection. **2019**, DOI: 10.13140/RG.2.2.19998.18245.
[33] McKinney, W. Pandas, Python Data Analysis Library. **2015**.
[34] Oliphant, T. E. Guide to Numpy; Trelgol Publishing: **2006**, 1, 85.

[35] Khandare, A.; Agarwal, N.; Bodhankar, A.; Kulkarni, A.; Mane, I. Analysis of Python Libraries for Artificial Intelligence. In Intelligent Computing and Networking: Proceedings of IC-ICN 2022; Springer Nature Singapore: Singapore, **2023**, pp 157–177.
[36] Ari, N.; Ustazhanov, M. Matplotlib in Python. In 2014 11th International Conference on Electronics, Computer and Computation (ICECCO); IEEE: **2014**, pp 1–6.
[37] Sundaram, J.; Gowri, K.; Devaraju, S.; Gokuldev, S.; Jayaprakash, S.; Anandaram, H.; Thenmozhi, M. An Exploration of Python Libraries in Machine Learning Models for Data Science. In Advanced Interdisciplinary Applications of Machine Learning Python Libraries for Data Science; IGI Global: **2023**, pp 1–31.