Machine Learning–Based Network Intrusion Detection Term Project Using the UNSW-NB15 Dataset

Patrick Kaggwa Pushya Damania Yewande Adegoroye
Muhammad Khubayeeb Kabir
Group 7

Abstract

Intrusion Detection Systems (IDS) are essential for protecting modern networks, yet traditional approaches struggle to keep up with the increasing sophistication of cyberattacks. Signature-based systems cannot detect new or modified attacks, while anomaly-based systems often generate excessive false alarms. Machine learning provides a more flexible and adaptive solution by learning patterns directly from network data. This analysis evaluates a range of classical machine learning models, tree-based ensemble algorithms, and deep learning approaches using the UNSW-NB15 dataset, a modern benchmark that contains realistic traffic and nine attack families. After applying systematic preprocessing, feature selection, and model comparison, the results show that ensemble boosting methods, particularly XGBoost and LightGBM, achieve the strongest performance. These findings highlight the potential of modern machine learning methods for developing effective IDS solutions.

1. Introduction and Background

The exponential growth of internet connectivity and digital transactions has considerably increased exposure to cyber threats, making effective intrusion detection a central requirement for network security [1]. Modern attackers employ advanced, constantly evolving techniques that can bypass or disguise themselves from conventional intrusion detection tools [3]. Signature-based IDS are efficient for known threats, but they are inherently limited when confronted with zero-day or complicated attacks [4]. Anomaly-based IDS attempt to identify deviations from normal behavior but often suffer from high false positive rates, which can overwhelm security analysts and reduce operational usefulness [2].

Machine learning has emerged as a promising alternative because it allows IDS to automatically learn discriminative patterns from data instead of relying solely on static rule sets [6]. However, much of the earlier research depended on legacy datasets such as KDD99 [7], which fail to reflect current traffic characteristics and contemporary attack strategies. The UNSW-NB15 dataset was designed to address these limitations by capturing realistic benign traffic alongside a diverse set of modern attacks using an instrumented testbed [5]. By leveraging this dataset, it becomes possible to develop and evaluate IDS models that are more representative of real-world network environments [8].

2. Main Objective

The main objective of this analysis is to evaluate and compare the performance of classical machine learning models, tree-based ensemble methods, and deep learning architectures in detecting malicious network traffic using the UNSW-NB15 dataset, with the aim of identifying the most effective approach for building a reliable intrusion detection system.

3. Problem Statement

Existing IDS solutions face significant challenges in keeping pace with the rapidly evolving threat landscape in modern networks. Signature-based methods are fundamentally limited to known attack patterns and therefore cannot detect novel or modified variants [4]. Anomaly-based systems aim to flag abnormal behavior but frequently misclassify legitimate traffic as suspicious, creating an unmanageable volume of alerts [2]. Furthermore, many machine learning–based IDS studies still rely on outdated datasets that are not representative of modern traffic and attack behaviors, which limits the external validity of their findings [6]. To develop more accurate and practical IDS models, it is necessary to systematically evaluate a variety of machine learning and deep learning techniques using a realistic and better dataset such as UNSW-NB15 [5].

4. Dataset Description

The UNSW-NB15 dataset was generated at the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) using the IXIA PerfectStorm traffic generator to simulate both normal and malicious network activity across a controlled testbed with multiple servers, clients, routers, and a firewall [5]. Packet captures were collected using tools such as Tcpdump and then processed by Argus and Bro-IDS to extract flow-level features. The dataset includes network flow characteristics, protocol information, service types, connection states, and various traffic statistics, resulting in 49 features in the original release, of which 44 were used after initial cleaning and 39 remained after further preprocessing in this study.

UNSW-NB15 contains traffic labeled as either normal or malicious and also provides attack category labels for nine families: Exploits, Denial of Service (DoS), Reconnaissance, Fuzzers, Generic, Shellcode, Backdoor, Worms, and Analysis. The subset used in this project consists of 82,332 samples with a binary target variable (0 for normal, 1 for malicious), where approximately 44.9% of observations are normal and 55.1% are malicious. This distribution exhibits a moderate imbalance that reflects realistic network traffic conditions and motivates careful handling of class proportions during model training and evaluation.

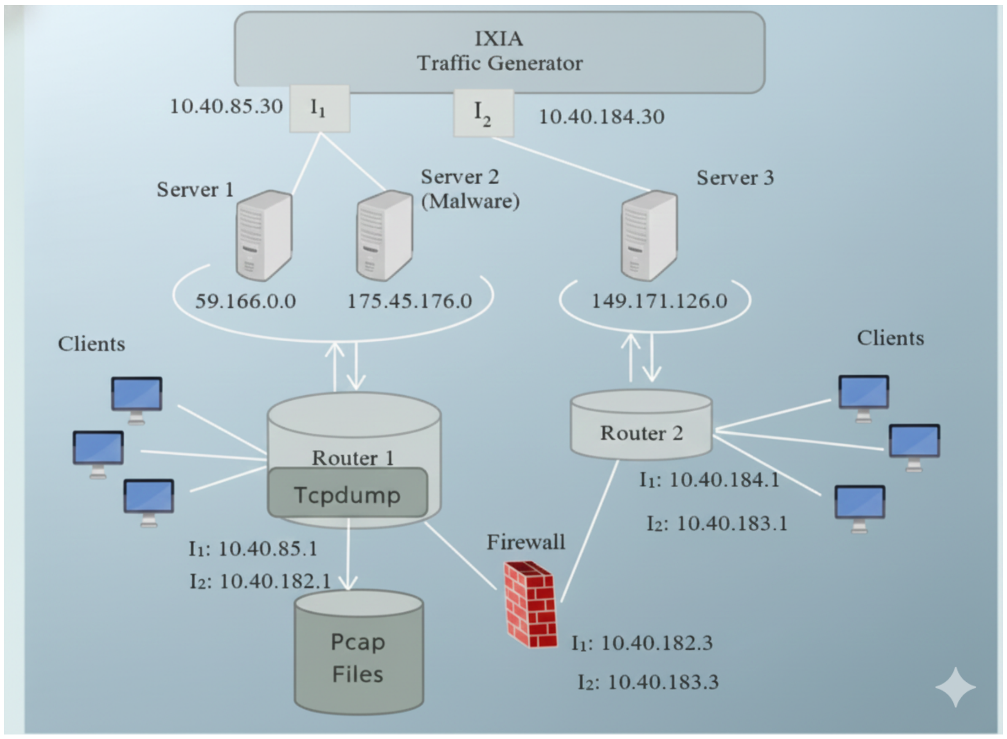

Figure 1: Schematic of the UNSW-NB15 testbed showing traffic generator, servers, clients, routers, and firewall used to capture benign and malicious network flows.

5. Data Preprocessing Techniques

A structured preprocessing pipeline was applied to prepare the dataset for modeling. Non-informative identifiers and some categorical protocol-related variables (such as protocol type, service, and state, depending on the modeling scenario) were removed to reduce redundancy and prevent potential data leakage. Infinite values and missing entries were handled by dropping severely corrupted records and then imputing remaining missing values using median imputation for numerical variables. Categorical features retained for certain models were encoded using one-hot encoding as appropriate.

To ensure consistent feature scales, StandardScaler normalization was applied to numerical features for algorithms sensitive to feature magnitude, such as logistic regression, KNN, SVM, and neural networks, while tree-based models were trained directly on the unscaled or minimally processed features. Outliers were examined using boxplots and Z-scores; where extreme values were likely to distort learning, they were capped at boundary values derived from the respective statistical limits, preserving overall data integrity while reducing distortion during training. Finally, an 80/20 train–test split with stratification on the binary class label was used to preserve class proportions across splits, and additional strategies such as class weighting or resampling (e.g., SMOTE or random oversampling) were considered to mitigate residual class imbalance.

6. Exploratory and Visual Analysis

Exploratory data analysis (EDA) was conducted to better understand the structure of the dataset and to guide feature selection and preprocessing choices. A correlation matrix heatmap of the numerical variables highlighted strongly correlated feature pairs, which helped identify redundant attributes and potential multicollinearity issues, particularly among traffic volume and timing features. Boxplots before and after outlier handling revealed substantial skewness and extreme values in some flow-level statistics, confirming the need for robust scaling and careful treatment of outliers.

Visual inspection of the class distribution confirmed that malicious traffic slightly outnumbered normal traffic, reinforcing the importance of stratified splitting and appropriate evaluation metrics such as F1-score and AUC in addition to accuracy. Feature-importance plots from preliminary tree-based models further suggested that certain traffic statistics, connection state features, and aggregation-based counts (e.g., connection counts per source/destination) contributed strongly to distinguishing normal from malicious flows, which informed subsequent feature selection and model refinement.

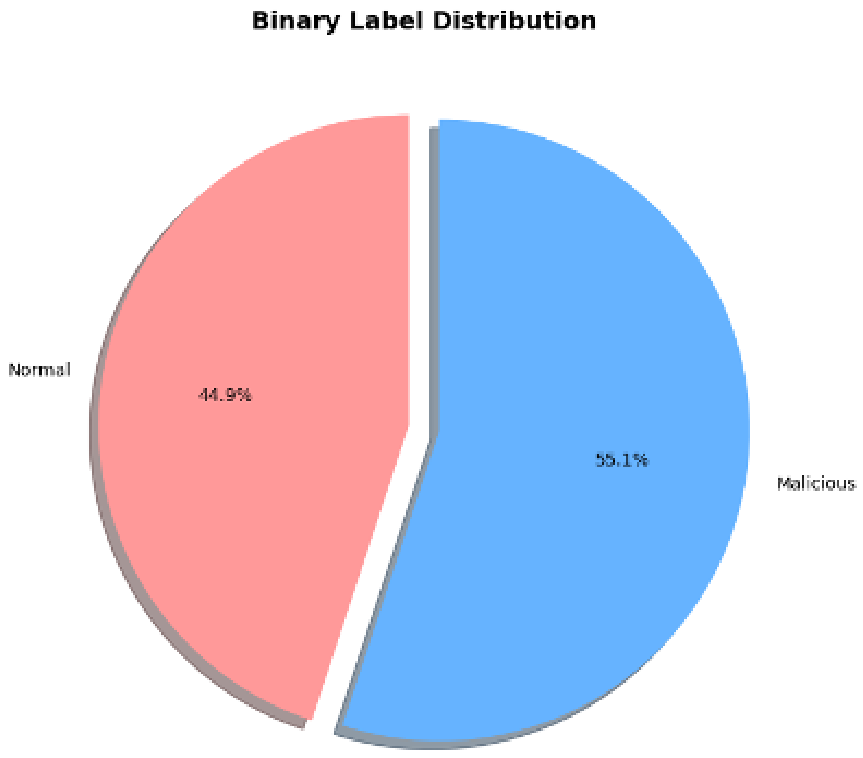

Figure 2: Binary class distribution of normal vs. malicious traffic in the UNSW-NB15 subset used in this study.

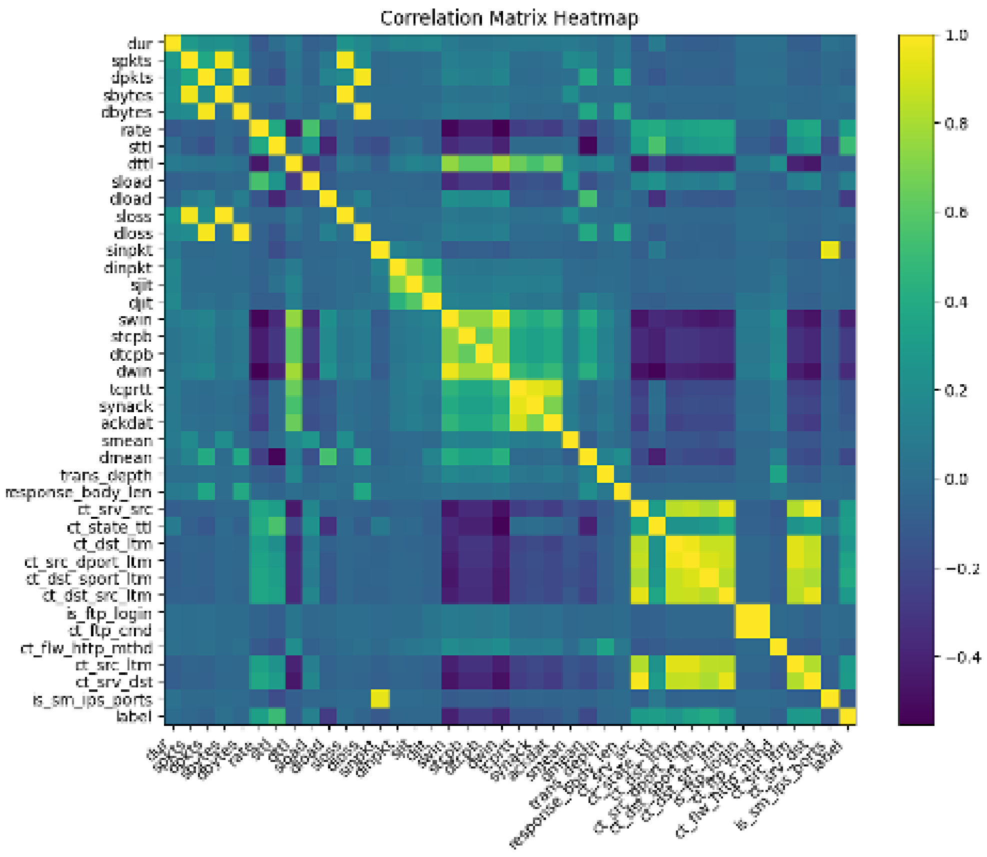
Figure 3: Correlation heatmap of selected numerical features highlighting strongly correlated traffic statistics.

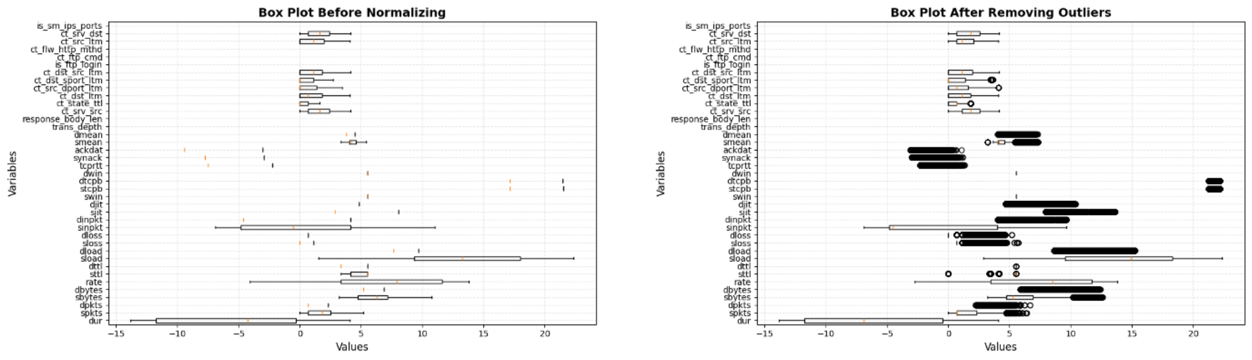
Figure 4: Feature value distributions before (left) and after (right) outlier removal and normalization for selected traffic features.

7. Modeling Techniques

To obtain a comprehensive comparison, three categories of models were implemented:

Classical Machine Learning Models:

- Logistic Regression

- Naïve Bayes
- K-Nearest Neighbors (KNN, with $k = 7$)
- Support Vector Machine (SVM) with an RBF kernel

Tree-Based Ensemble Methods:

- Decision Tree
- Random Forest
- Extra Trees
- Gradient Boosting (GBM)
- XGBoost
- LightGBM
- CatBoost

Neural Network and Deep Learning Models:

- Multi-Layer Perceptron (MLP)
- Deep Neural Network (DNN)
- Convolutional Neural Network (CNN)
- Transformer-based architecture

The ensemble methods were configured with 200–300 estimators to ensure sufficient model capacity, while the neural architectures were trained for up to 50 epochs with appropriate regularization and early stopping to prevent overfitting. This model suite covers linear, probabilistic, ensemble, and deep learning paradigms, enabling a robust and fair comparison across diverse algorithmic families.

7.1 Deep Learning Architectures

Several deep learning models were implemented to evaluate how different neural architectures handle the UNSW-NB15 intrusion detection task. All models were trained on the same standardized feature vectors, each consisting of 39 numerical attributes after preprocessing.

Multilayer Perceptron (MLP) The first neural network model was a Multilayer Perceptron (MLP) implemented using the `MLPClassifier` in scikit-learn. The input to the network was the 39-dimensional feature vector, followed by three fully connected hidden layers with 128, 64, and 32 neurons, respectively. Each hidden layer used the ReLU activation function, and the output layer consisted of a single neuron with a logistic activation for binary classification. The model was trained using the Adam optimizer for up to 300 iterations. In total, the MLP contained approximately 15,500 trainable parameters.

Deep Neural Network (DNN) A deeper feed-forward neural network (DNN) was implemented in PyTorch. The network consisted of four fully connected layers: an input layer mapping the 39 input features to 256 hidden units, followed by layers of size 256–128 and 128–64, each with ReLU activation. The final layer projected the 64-unit representation to a single output neuron with sigmoid activation. The DNN was trained using the Adam optimizer, binary cross-entropy loss, a batch size of 64, and 50 training epochs. This architecture contained approximately 51,500 trainable parameters, enabling it to learn richer nonlinear patterns than the shallow MLP.

Convolutional Neural Network (CNN) A one-dimensional Convolutional Neural Network (CNN) was designed to capture local spatial patterns in the feature space. The 39-dimensional input vector was reshaped to a single-channel sequence of length 39 and passed through two convolutional layers. The first convolution had 1 input channel, 32 output channels, a kernel size of 3, and padding of 1; the second convolution expanded the representation from 32 to 64 channels, also using a 3-element kernel with padding 1. Each convolutional layer used ReLU activation. An AdaptiveMaxPool1d layer compressed the temporal dimension to length 1, producing a 64-dimensional vector that was fed into a final fully connected layer with sigmoid activation for binary classification. This CNN architecture contained roughly 6,400 trainable parameters.

Transformer-Based Classifier A transformer-inspired attention model was implemented to allow the network to learn relationships between features through self-attention mechanisms. The 39-dimensional input vector was first projected into a 128-dimensional embedding space. This embedding was then passed through a TransformerEncoder composed of two stacked encoder layers, each with a hidden dimension of 128, four attention heads, and the default 2048-unit position-wise feed-forward network. Residual connections and layer normalization were applied within each encoder layer. After the encoding stage, the representation was average-pooled along the sequence dimension and passed to a fully connected output layer with sigmoid activation. This transformer-based model contained approximately 1.19 million trainable parameters, making it the most expressive and computationally intensive network evaluated in this study.

8. Rationale for Model Choices

Classical models were included as interpretable and computationally efficient baselines that provide insight into the linear separability and probabilistic structure of the data. Logistic regression and Naïve Bayes offer simple decision boundaries and probabilistic outputs, while KNN and SVM can capture moderately nonlinear relationships and serve as strong non-ensemble benchmarks.

Tree-based ensemble methods were selected because they are well suited to structured tabular data, can naturally handle mixed feature types, and tend to perform strongly on imbalanced classification tasks. Random Forest and Extra Trees use bagging to reduce variance, while Gradient Boosting, XGBoost, LightGBM, and CatBoost employ boosting strategies to iteratively correct errors and capture complex patterns. XGBoost and LightGBM, in particular, have become standards in many applied machine learning tasks due to their high accuracy, built-in regularization, and efficiency.

Deep learning models were incorporated to evaluate their ability to learn rich feature representations and capture higher-order interactions in network flows. The MLP and DNN architectures provide flexible nonlinear decision boundaries, CNNs can exploit local patterns in feature space. Although more computationally demanding, these architectures help assess whether deep feature learning provides added value over strong tree-based ensembles in this IDS context.

9. Feature Selection

Feature selection in this analysis focused on identifying the most meaningful and non-redundant features before training the models. The initial step involved examining the correlation structure of the dataset to identify pairs of features that were strongly correlated with each other. Highly redundant features particularly those related to packet counts, byte volume, and timing patterns were removed so that the models would not learn the same information multiple times.

Mutual information between each feature and the target label was then assessed to understand which variables carried the strongest predictive signal for distinguishing between benign and malicious traffic. Features with very low contribution were noted but not aggressively removed, since eliminating too many variables can risk losing subtle attack-related patterns.

Although Recursive Feature Elimination and dimensionality-reduction methods such as PCA or autoencoders are commonly used in IDS research, these techniques were not implemented in the current modeling pipeline. Instead, the analysis preserved most original features after correlation filtering to allow the machine learning and deep learning models to learn from the full richness of the UNSW-NB15 dataset.

Stepwise feature selection

The following table presents the feature importance scores derived from stepwise feature selection. These scores indicate the relative contribution of each feature to the model's prediction accuracy, with higher values representing greater importance.

Table 1: Feature Importance Scores with Descriptions

Feature Name	**Full Variable Name**	**Importance**
dur	Duration of the flow	-0.0
sttl	Source TTL (Time to Live)	1.0
swin	Source window size	0.1997
label	Class label (normal/malicious)	0.1525
ct_state_ttl	Count of state TTL	0.2224
dwin	Destination window size	0.0875
ct_dst_src_ltm	Count destination source long term	0.1424
ct_srv_dst	Count service destination	0.0346
is_sm_ips_ports	Is source or destination IPs and ports same	0.0203
dttl	Destination TTL	0.0165
ct_srv_src	Count service source	0.0149
dmean	Destination mean packet size	0.0142
smean	Source mean packet size	0.0061
tcprtt	TCP round trip time	0.0040
ct_flw_http_mthd	Count flow HTTP methods	0.0030
rate	Flow rate (packets per second)	0.0027
dload	Destination bits per second	0.0019
ct_dst_ltm	Count destination long term	0.0094
ct_src_dport_ltm	Count source destination port long term	0.0019
djit	Destination jitter	0.0009
trans_depth	HTTP transaction depth	0.0009
ct_dst_sport_ltm	Count destination source port long term	0.0008
dbytes	Destination bytes	0.0040
dinpkt	Destination input packets	0.0008
sjit	Source jitter	0.0010
ct_src_ltm	Count source long term	0.0005
sload	Source bits per second	0.0003
ackdat	TCP ACK data	0.0002
sinpkt	Source input packets	0.0001
ct_ftp_cmd	Count FTP commands	0.0001
is_ftp_login	Is FTP login	0.00006
response_body_len	Response body length (HTTP)	0.00002
spkts	Source packets	0.00002
sbytes	Source bytes	0.0013
dpkts	Destination packets	0.0016

10. Results

All models were evaluated on the held-out test set using accuracy, F1-score, and Area Under the ROC Curve (AUC). The ensemble methods, especially XGBoost and LightGBM, achieved the highest performance across all metrics, with Random Forest, Extra Trees, and CatBoost also performing strongly. Deep learning models obtained competitive results but did not surpass the best gradient-boosting methods, while classical models trailed behind but still provided useful baselines.

Table 2: Performance of Models on the UNSW-NB15 Dataset

Model	**Accuracy**	**F1-Score**	**AUC**
XGBoost	0.979	0.981	0.998
LightGBM	0.977	0.979	0.998
Random Forest	0.976	0.978	0.997
CatBoost	0.976	0.978	0.997
Extra Trees	0.975	0.977	0.997
Decision Tree	0.965	0.968	0.964
Gradient Boosting	0.964	0.967	0.994
MLP	0.962	0.966	0.994
Transformer	0.960	0.963	0.994
DNN	0.955	0.959	0.994
CNN	0.953	0.958	0.992
SVM	0.934	0.940	0.981
KNN	0.927	0.932	0.980
Logistic Regression	0.884	0.896	0.961
Naïve Bayes	0.771	0.780	0.855

11. Interpretation of Results

The results clearly indicate that ensemble boosting models, particularly XGBoost and LightGBM, are the most effective for detecting malicious traffic in the UNSW-NB15 dataset. XGBoost achieved the highest accuracy (approximately 97.9%) and an AUC close to 0.998, demonstrating strong discrimination between normal and malicious flows. LightGBM achieved comparable accuracy and AUC while offering faster training times, which is valuable for environments with limited computational resources or where rapid retraining is required.

Random Forest, Extra Trees, and CatBoost also delivered excellent performance, reinforcing the advantage of tree-based ensembles for this kind of structured tabular data. Deep learning models such as MLP, DNN, CNN, and Transformer architectures performed competitively, with accuracies generally above 95% and AUC values around 0.99, but they required more extensive tuning and longer training times to reach their best performance. Classical models like logistic regression, Naïve Bayes, KNN, and SVM were useful as baselines but could not match the predictive power of the ensemble methods, especially in capturing complex nonlinear interactions present in the network traffic features.

12. Conclusions and Recommendations

This analysis successfully compared 15 machine learning models across three major categories for network intrusion detection using the UNSW-NB15 dataset. The findings demonstrate that modern tree-based ensemble methods, particularly XGBoost and LightGBM, provide the best overall performance, achieving high accuracy, strong F1-scores, and near-perfect AUC values on realistic network traffic data. Comprehensive preprocessing, careful feature selection, and robust evaluation were essential in obtaining these results.

For practical deployment, XGBoost is recommended for high-stakes production environments where maximum detection performance is critical, while LightGBM is well suited for scenarios requiring a balance between accuracy and training speed. Deep learning models remain promising, particularly if combined with advanced representation learning and hyperparameter tuning, but they may be more appropriate for settings with ample computational resources. Future work should extend this analysis to multiclass attack classification, temporal modeling of network flows (e.g., using sequence models on time-ordered connections), adversarial robustness against evasion attacks, and hybrid IDS architectures that combine the strengths of ensemble methods and deep neural networks.

References

[1] Stefan Axelsson. Intrusion detection systems: A survey and taxonomy. *Technical report, Chalmers University of Technology*, 2000.

[2] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[3] Pedro Garcia-Teodoro, Pablo Diaz-Verdejo, Gonzalo Macia-Fernandez, and Enrique Vazquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1-2):18–28, 2009.

[4] Richard P Lippmann, Jeffrey W Haines, David J Fried, Joseph Korba, and Kumar Das. Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, 2:12–26, 2000.

[5] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). *Military Communications and Information Systems Conference (MilCIS), 2015*, pages 1–6, 2015.

[6] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 305–316, 2010.

[7] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6, 2009.

[8] Mahbod Tavallaee, Wei Lu, and Ali A Ghorbani. Detailed analysis of the unsw-nb15 dataset. *Australian Centre for Cyber Security*, 2010.