CID - A Multi-Model AI Framework for High-Fidelity Cyber Intrusion Detection

Sarvagya Singh, Pritish Majumdar, and Daksh Kaushal
Lovely Professional University, Phagwara, Punjab, India
Submitted to: Dr. Geethika Sethi

***Abstract*—The exponential growth of internet-connected devices has led to a corresponding surge in sophisticated cyber threats, rendering traditional rule-based security measures insufficient. The primary goal of this project was to design, develop, and deploy a comprehensive, robust, and extensible AI-powered Cyber Intrusion Detection System (CIDS). This system is engineered to accurately identify, classify, and mitigate various types of network intrusions by leveraging a sophisticated suite of supervised machine learning models. A pivotal component of this objective was to bridge the gap between theoretical machine learning and practical application by deploying the system as an interactive, user-friendly web application. This paper details the complete development lifecycle, commencing with rigorous data preprocessing of the TII-SSRC-23 dataset, which involves complex feature engineering and dimensionality handling. The core research involved the training and hyperparameter tuning of 27 distinct models across nine algorithmic families, including Random Forest, XGBoost, and CatBoost. The final system, deployed on Streamlit Cloud, offers real-time prediction capabilities, effectively addressing real-world challenges such as extreme class imbalance, inference latency, and false positive reduction. The results demonstrate that ensemble methods significantly outperform singular algorithms in detecting complex attack vectors.**

***Index Terms*—Intrusion Detection, Supervised Learning, XGBoost, Streamlit, Cybersecurity, Machine Learning, Anomaly Detection, Network Forensics.**

I. Introduction

A. Background and Motivation

In the contemporary digital landscape, network security has become a paramount concern for nations, organizations, and individuals alike. As network infrastructures evolve to support high-speed data transfer and IoT ecosystems, the attack surface for malicious actors has expanded dramatically. Traditional intrusion detection systems (IDS), which rely heavily on signature-based detection, are increasingly failing to cope with the velocity and variety of modern cyberattacks, such as polymorphic malware and zero-day exploits. This necessitates a paradigm shift towards heuristic and behavioral analysis powered by Artificial Intelligence (AI).

B. Objectives of the Work Undertaken

To achieve the primary goal of creating a comprehensive CIDS, the project was systematically broken down into several specific, high-impact sub-objectives:

II. Data Preprocessing and Advanced Feature Engineering

A foundational objective was to prepare raw, noisy network traffic data for machine learning readiness. This involved more than simple cleaning; it required a rigorous process of handling inconsistencies, normalizing numerical distributions to facilitate convergence in gradient-based models, and encoding categorical variables. A crucial part of this stage was to engineer new, domain-specific features—such as flow inter-arrival times, byte ratios, and window size aggregates—to enhance the predictive power of the models regarding traffic behavior.

III. Model Training and Rigorous Comparative Evaluation

The core of the project was to train and evaluate a diverse set of machine learning models to identify the optimal architecture for network traffic classification. The objective was to train 27 distinct models spanning nine different algorithmic families: Random Forest, Decision Tree, Logistic Regression, Naive Bayes, XGBoost, LightGBM, K-Nearest Neighbors (KNN), AdaBoost, and CatBoost. A significant aim was to conduct a comparative analysis of these models by evaluating them on key performance metrics, including accuracy, precision, recall, F1-score, and ROC-AUC, specifically looking for trade-offs between sensitivity and specificity.

IV. System Integration and Full-Stack Deployment

To translate the trained models into a usable tool for security analysts, a major objective was to build an interactive and intuitive frontend using the Streamlit framework. This involved creating a seamless backend that connects the model predictions to the user interface, utilizing `joblib` for efficient model caching, serialization, and routing, ensuring that the system could run independently of a local Python environment.

V. Usability and Real-Time Inference

The project aimed to deliver a system with high usability, allowing users—regardless of their coding expertise—to simply upload a CSV file containing network traffic records and instantly receive classified predictions. This required optimizing the inference pipeline to handle data batches with minimal latency.

VI. Research Extension and Scalability

Looking toward the future, the project was designed with extensibility in mind. The architecture is modular, allowing for the seamless integration of future enhancements such as unsupervised anomaly detection for zero-day threats and Graph Neural Networks (GNNs) for topological network analysis.

VII. Scope of the Work

The scope of this project was to develop a comprehensive, end-to-end cyber intrusion detection system, covering the entire pipeline from data ingestion to deployment.

VIII. Current Capabilities

- **Supervised Learning Framework:** Utilizes a robust suite of 27 distinct models, allowing for ensemble voting strategies.
- **Data Input and Processing:** Capable of ingesting and preprocessing raw network traffic logs provided in standard CSV formats.
- **Multi-Class Classification:** Goes beyond binary detection to distinguish between benign traffic and specific attack categories (DoS, Brute Force, Mirai, Port Scanning, etc.).
- **Interactive Web Application:** A fully deployed interface on Streamlit Cloud facilitating real-time predictions without local installation requirements.
- **Comparative Analysis Dashboard:** A built-in platform for benchmarking different algorithms against one another visually.

A. Limitations

- **Detection of Unknown Attacks:** Relying primarily on supervised learning, the system is limited to detecting attack signatures present in the training data and cannot inherently detect zero-day vulnerabilities without retraining.
- **Real-Time Packet Capture:** The system does not currently support live packet sniffing (via tools like Wireshark/Scapy) and requires pre-captured CSV files.
- **Model Interpretability:** While highly accurate, complex ensemble models like XGBoost lack inherent interpretability compared to decision trees (SHAP analysis is planned for future iterations).
- **Advanced Network Analysis:** The current models treat connections independently, missing potential structural relationships that could be found by GNNs.

IX. Importance and Applicability

A. Importance

The project addresses the critical inadequacies of traditional rule-based systems against modern threats. By developing proactive defense mechanisms, it helps protect critical infrastructure from disruption. Furthermore, it provides a valuable comparative algorithmic analysis, helping researchers understand the trade-offs between training time, inference speed, and classification accuracy in a cybersecurity context.

B. Applicability

The system serves as a versatile tool for real-time network monitoring simulations, cybersecurity operations centers (SOCs), digital forensics investigations, and as a research/educational platform for students entering the field of AI security.

X. Machine Learning and Data Science

- Performed extensive feature engineering on the TII-SSRC-23 dataset, extracting statistical moments from byte streams and flag counts.
- Conducted hyperparameter tuning for 27 models across diverse families (Random Forest, XGBoost, LightGBM, CatBoost) to optimize for the F1-score.
- Assessed performance using a confusion matrix approach, focusing on minimizing False Negatives (attacks misclassified as benign).
- Addressed severe class imbalance using Synthetic Minority Over-sampling Technique (SMOTE) and algorithmic class weighting.

XI. Application Development and Deployment

- Architected a backend system capable of deserializing large models using `joblib` with caching strategies to ensure low-latency performance.
- Developed an intuitive, responsive dashboard using Streamlit to visualize data distributions and prediction results.
- Managed the CI/CD pipeline, handling version control via GitHub and deployment to Streamlit Community Cloud.

XII. Research and Analysis

The role involved deep-dive research into the efficacy of gradient boosting algorithms on tabular network data, authoring a detailed project report, and conducting a thorough literature review of existing IDS methodologies.

XIII. Activities and Equipment Handled

Activities spanned the full data science lifecycle: Data preprocessing (cleaning noise, null handling via imputation), categorical encoding (One-Hot vs Label), feature scaling (StandardScaler for distance-based algorithms), class balancing (SMOTE), and rigorous model evaluation. The technology stack included Python, Pandas for manipulation, Joblib for serialization, Matplotlib/Seaborn for visualization, Scikit-learn for baseline models, XGBoost/LightGBM/CatBoost for boosting, Streamlit for frontend, and GitHub for version control.

XIV. Challenges Faced and Solutions

1) **Class Imbalance:** The dataset was heavily skewed towards benign traffic. *Solution:* Addressed via SMOTE for linear models and `scale_pos_weight` parameters for tree-based models to penalize misclassification of the minority class.

2) **Noisy Data:** Real-world traffic contains corrupted headers. *Solution:* Rigorous cleaning pipelines were established to remove rows with NaN values or infinite flow durations.
3) **Latency:** Loading 27 models into memory is resource-intensive. *Solution:* Implemented lazy loading and model caching (@st.cache_resource) alongside `joblib` compression level 3.
4) **User Errors:** *Solution:* Robust error-handling was implemented in the UI to manage incorrect file formats or missing columns gracefully.
5) **Zero-Day Attacks:** *Solution:* While not solved in the current iteration, the architecture was documented to allow future integration of Isolation Forests for anomaly detection.

XV. Methodology

XVI. Data Acquisition and Understanding

The system utilizes the **TII-SSRC-23** dataset, a contemporary benchmark developed by the Technology Innovation Institute. Unlike older datasets (e.g., KDD99), TII-SSRC-23 reflects modern traffic patterns and encrypted flows. The dataset is rich in features, including:

- **Duration-related features:** Flow duration, idle time mean/std dev.
- **Traffic Volume Metrics:** Total bytes forward/backward, packet lengths.
- **Protocol/Port Information:** Source/Dest ports, Protocol (TCP/UDP/ICMP).
- **Behavioral Indicators:** TCP flag counts (SYN, ACK, FIN, RST).

The target variable includes multiple attack classes such as DoS, DDoS, Port Scanning, Botnet, Exfiltration, and Brute Force, providing a multi-class classification challenge.

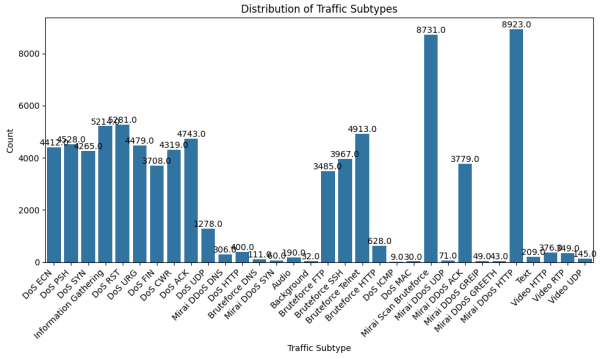

Figure 1. Distribution of traffic subtype

XVII. Data Preprocessing

The preprocessing workflow was designed to maximize signal and minimize noise:

1) **Data Cleaning:** Removal of timestamps and Flow IDs which cause overfitting, and dropping rows with missing values.

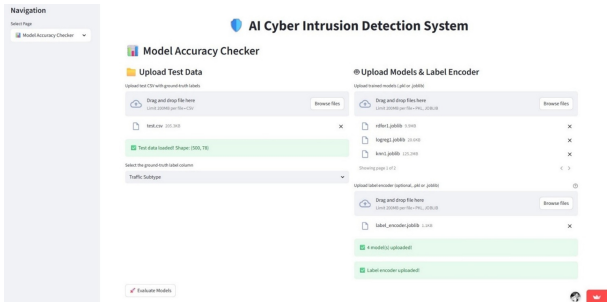

Figure 2. App Interface

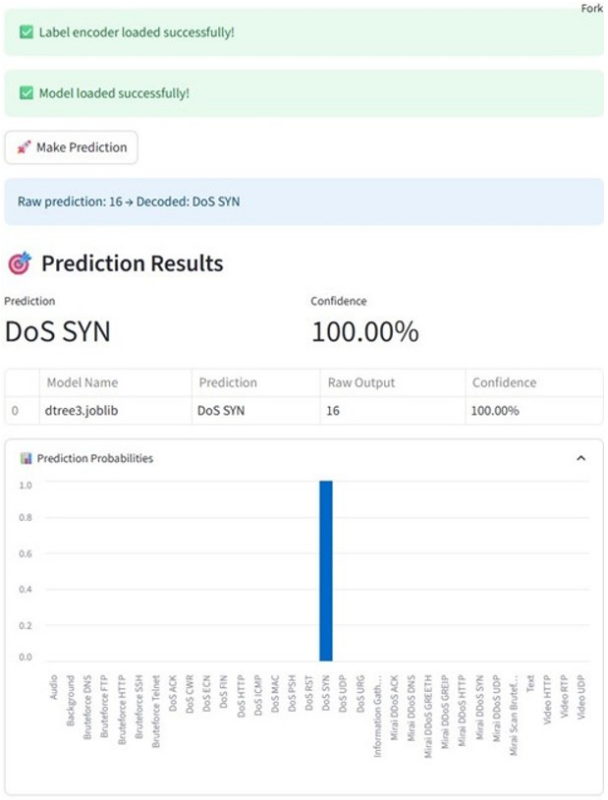

Figure 3. App interface (showing result)

2) **Categorical Encoding:** Transformation of non-numeric data (protocols, flags) using Label Encoding to create integer representations suitable for tree-based models.
3) **Feature Scaling:** Application of StandardScaler ($z = \frac{x-\mu}{\sigma}$) to normalize numerical features, essential for KNN and Logistic Regression convergence.
4) **Feature Engineering:** Derivation of high-level features such as Byte ratio (Forward/Backward), Flow symmetry score, and Packet density to capture attack dynamics.
5) **Class Balancing:** Implementation of SMOTE to synthesize minority class instances in the training set, ensuring the model learns to recognize rare attacks.
6) **Label Encoding:** Converting the textual multi-class target labels into a strictly integer format for compatibility with XGBoost/LightGBM.

XVIII. Model Architecture and Algorithmic Selection

The system incorporates a comprehensive array of 27 supervised models across 9 distinct families to ensure a robust comparison:

1) **Random Forest:** An ensemble of decision trees using bagging to reduce variance and prevent overfitting.
2) **Decision Tree:** Single tree models tuned for depth, offering high interpretability but prone to high variance.
3) **Logistic Regression:** A baseline linear model used to establish a performance floor.
4) **Gaussian Naive Bayes:** A probabilistic model assuming feature independence, useful for high-dimensional baselining.
5) **K-Nearest Neighbors (KNN):** A non-parametric distance-based classifier, effective but computationally expensive at inference time.
6) **XGBoost:** An optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable.
7) **LightGBM:** A gradient boosting framework that uses tree-based learning algorithms, designed for speed and efficiency with large datasets.
8) **AdaBoost:** An iterative boosting algorithm that focuses weight on misclassified instances to turn weak learners into strong ones.
9) **CatBoost:** A boosting algorithm specifically optimized for categorical data handling, reducing the need for extensive preprocessing.

Three variants of each family were trained with differing hyperparameters (e.g., varying estimators, max_depth, learning rates) to find the local optima for each algorithm.

XIX. Model Training and Evaluation Strategy

The dataset was split into an 80/20 training and testing set using stratified sampling to maintain class distribution. During the prototyping phase, 5-fold cross-validation was employed to ensure result stability. The evaluation focused on a holistic view of performance: Accuracy (overall correctness), Precision (trustworthiness of an alarm), Recall (ability to catch attacks), F1-Score (harmonic mean of Precision and Recall), and ROC-AUC (separability capability).

XX. Web Application Development

To make the intrusion detection system accessible beyond a code-centric research environment, an interactive web application was developed and deployed using the Streamlit framework. The live system is publicly accessible at:

https://cyber-intrusion-detection.streamlit.app/

The application is designed to operationalize the trained machine learning models, enabling real-time inference through an intuitive and platform-independent interface. Its architecture loosely follows the Model–View–Controller (MVC) paradigm, ensuring clean separation between model logic, user interaction, and data processing.

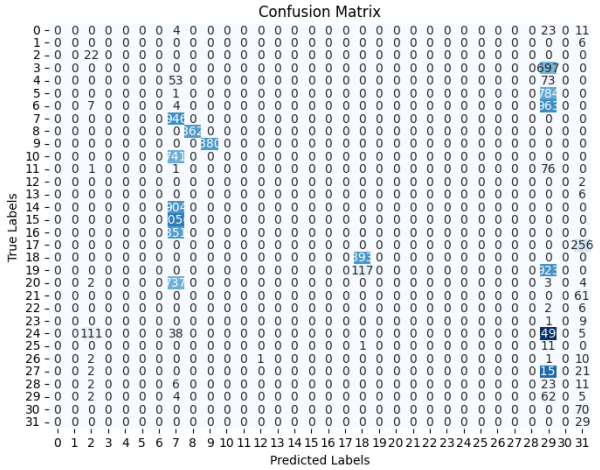

Figure 4. Confusion matrix)

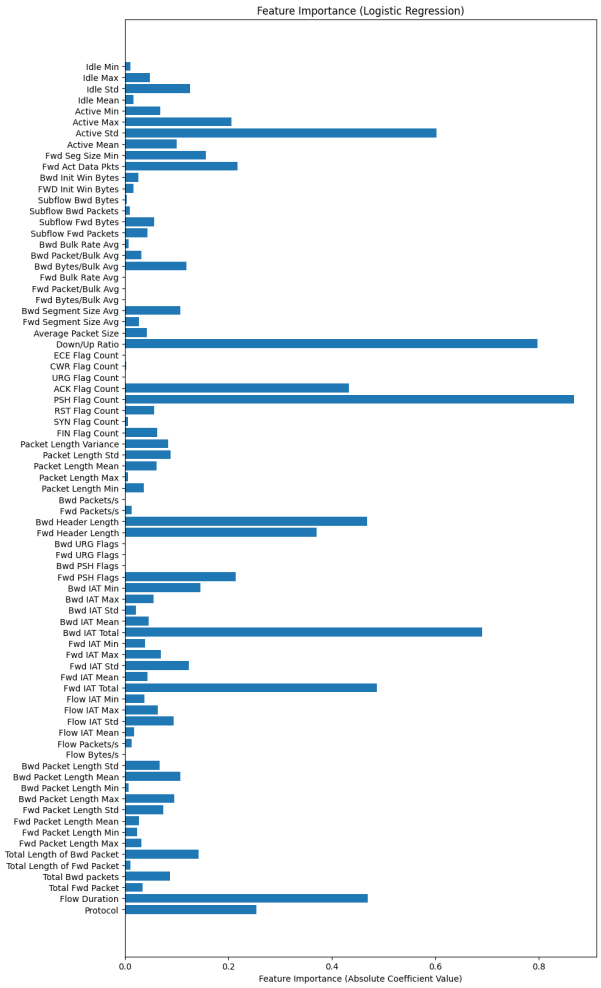

Figure 5. Example of Feature importance (for Logistic Regression)

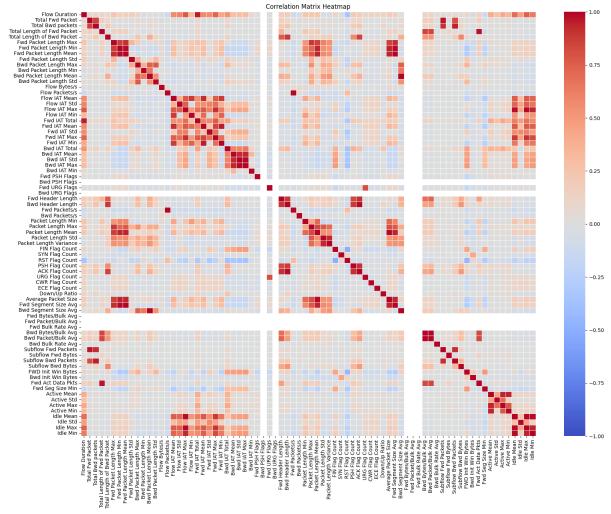

Figure 6. Correlation Heatmap

A. System Architecture

1) Model Layer (Backend): This layer encapsulates all trained classifiers—27 models spanning nine supervised learning families. The models are serialized using *joblib*, allowing compact storage and rapid deserialization. To prevent redundant loading operations—which are computationally expensive on cloud environments.

2) Controller Layer (Processing Pipeline): All uploaded CSV files pass through a structured preprocessing pipeline implemented with Pandas and Scikit-learn utilities. The controller is responsible for:

- validating input schema (column checks, type enforcement),
- performing label encoding and scaling consistent with training configurations,
- handling erroneous or malformed inputs using exception-safe routines,
- managing the routing logic that selects and applies the chosen model.

This layer ensures strict reproducibility of the training-time transformations, guaranteeing that inference remains aligned with the model's expectations.

3) View Layer (User Interface): The front-end is implemented entirely using Streamlit's high-level UI components. It features:

- a clean sidebar for model selection,
- a drag-and-drop CSV upload interface,
- dynamic preview of the uploaded dataset,
- real-time generation of classification metrics (accuracy, precision, recall, F1),
- visual indicators for predicted attack classes.

The lightweight UI design allows even non-technical users—such as security analysts, students, and system administrators—to interact with complex models without writing code.

B. Performance and User Experience Considerations

To support efficient inference at scale, several optimizations were integrated:

- **Model Caching:** Eliminates repeated disk I/O by persisting model objects in memory after the first load.
- **Pipeline Consistency:** Ensures that the preprocessing applied during inference exactly mirrors the training environment, preventing data drift.
- **Responsive Visualization:** Streamlit's asynchronous rendering enables near-instantaneous metric display even for large batches of traffic flows.
- **Cloud Deployment Optimizations:** Memory-efficient joblib compression and optional garbage collection maintain the application within Streamlit Cloud resource limits.

By bridging rigorous machine learning research with an accessible, production-ready interface, the application transforms the underlying models into a practical cyber-defense tool suitable for demonstrations, academic study, and prototype deployment within SOC environments.

XXI. Deployment and CI/CD

The application was deployed on the Streamlit Community Cloud, directly linked to a GitHub repository for Continuous Integration/Continuous Deployment (CI/CD). Deployment optimizations included memory management via garbage collection and joblib compression to fit within the cloud environment's resource constraints.

XXII. Result and Analysis

XXIII. Evaluation Metrics and Interpretation

The 27 models were rigorously evaluated using Accuracy, Precision, Recall, F1-Score (Macro & Weighted), and ROC-AUC. In intrusion detection, Recall is often prioritized to minimize missed attacks, while Precision ensures that administrators are not overwhelmed by false alarms.

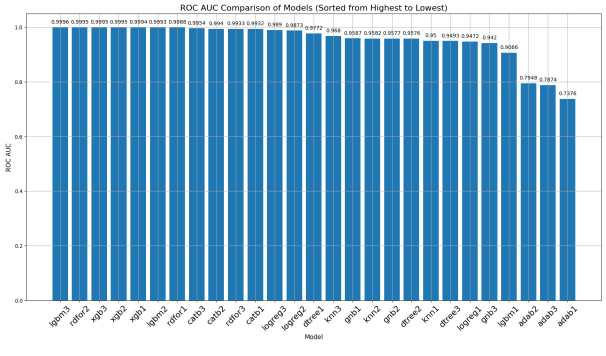

Figure 7. ROC-AUC Comparison of Models (Sorted).

XXIV. Performance Overview and Comparative Analysis

The analysis yielded several key insights into algorithmic performance for cybersecurity data:

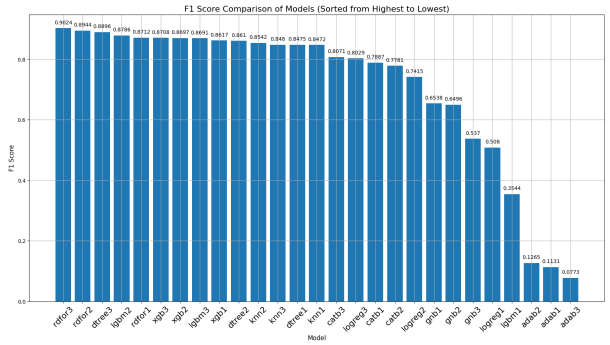

Figure 8. F1 Score Comparison of Models (Sorted).

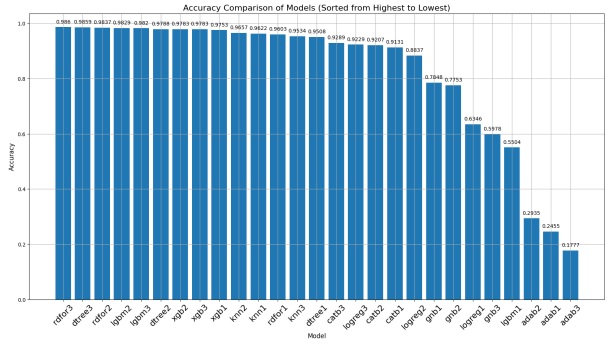

Figure 9. Accuracy Comparison of Models (Sorted).

- **Top Performing (Gradient Boosting):** XGBoost (specifically variant 'xgb2') achieved the highest F1-score ($\sim$0.985). Its ability to handle non-linear relationships and built-in regularization prevented overfitting on the training data. LightGBM followed closely, offering comparable accuracy with faster training times.
- **Strong Contenders (Bagging):** Random Forest performed exceptionally well, proving robust against noise and requiring less hyperparameter tuning than boosting methods.
- **Moderate Performance:** Logistic Regression and Decision Trees provided a solid baseline but struggled with the complex, non-linear boundaries of sophisticated attack patterns.
- **Lower Performing:** Certain variants of AdaBoost and CatBoost struggled with the specific feature distribution of this dataset, likely due to suboptimal hyperparameter configurations for this specific task.

XXV. Inference Latency and Model Size Analysis

For a CIDS, accuracy is moot if detection is too slow. An analysis of prediction time and memory usage is presented in Table I. While XGBoost is the most accurate, LightGBM offers a compelling trade-off for high-throughput environments.

XXVI. Caveats

While the system demonstrates strong predictive capability across multiple attack classes, several caveats emerge from the experimental design and operational assumptions. First, the dataset (TII-SSRC-23) is a static snapshot of network behavior and may not fully represent the evolving threat landscape, especially adversarial traffic crafted to evade detection. Furthermore, the evaluation environment is controlled; real-world deployments often encounter packet loss, encrypted payloads, incomplete flows, and noisy metadata that can degrade model reliability. Finally, the reported performance metrics assume consistent feature availability. Any drift in upstream logging infrastructure—such as missing TCP flag counts or truncated flow durations—can significantly weaken model validity.

Table I
Inference Latency and Model Size Metrics

Model Family	Avg. Inference Time	Model Size	Efficiency Score
XGBoost	$\sim$70ms	$\sim$5.2 MB	High
LightGBM	$\sim$40ms	$\sim$3.8 MB	Very High
Random Forest	$\sim$85ms	$\sim$6.1 MB	Moderate
KNN	$\sim$200ms	$\sim$1.2 MB	Low
GNB	$\sim$12ms	$\sim$0.3 MB	Very High
Decision Tree	$\sim$15ms	$\sim$0.8 MB	High

XXVII. Limitations

Although the system achieves high accuracy, several structural limitations constrain its practical deployment:

- **Dependence on Supervised Labels:** The system is restricted to attacks represented in the training data. It lacks intrinsic capacity to detect zero-day or novel attack vectors.
- **Absence of Real-Time Packet Capture:** The pipeline relies on preformatted CSV logs, preventing direct ingestion of live packet streams or PCAP files. This limits operational use in fast-moving SOC environments.
- **Model Interpretability Challenges:** While gradient-boosting models deliver superior performance, they operate as black boxes without integrated explainability tools. This complicates incident analysis and forensic investigation.
- **Resource Constraints:** Loading 27 models simultaneously, even with caching, is computationally heavy. On lower-end hardware or cloud-restricted environments, latency may rise under high request volume.
- **Flow Independence Assumption:** All models treat network flows as independent observations, ignoring temporal correlations and multi-hop attack sequences often present in coordinated intrusions.

XXVIII. Future Work

Several promising directions can enhance both the scientific depth and operational utility of the system:

- **Integration of Unsupervised and Hybrid Methods:** Incorporating Isolation Forests, Autoencoders, or Deep SVDD can enable detection of previously unseen attacks and reduce reliance on labeled datasets.
- **Graph Neural Networks (GNNs):** Modeling network traffic as a dynamic graph could reveal structural patterns, lateral movement, and coordinated scans that traditional tabular models overlook.

- **Time-Series and Sequential Modeling:** LSTM, GRU, or Transformer-based approaches can capture temporal dependencies and identify multi-stage intrusion campaigns.
- **Explainability Frameworks:** Implementing SHAP or LIME would offer actionable insights to analysts, bridging the gap between high-performance models and operational trust.
- **Real-Time Data Pipeline:** Adding live packet capture modules (Scapy, PyShark, or Zeek integration) would transform the system from batch inference to a functioning NIDS.
- **Adversarial Robustness Testing:** Evaluating the system against evasion strategies such as feature manipulation, traffic morphing, and adversarial perturbations can harden it for real-world deployment.
- **Model Compression and Optimization:** Techniques such as quantization, pruning, and ONNX export can significantly reduce inference latency for edge deployments.

XXIX. Conclusion

This project successfully culminated in the development, testing, and deployment of a robust and scalable Cyber Intrusion Detection System (CIDS) based entirely on supervised machine learning techniques. By training and evaluating a massive suite of 27 models, the research demonstrates that ensemble-based models, particularly XGBoost and LightGBM, significantly outperform classical statistical models in identifying malicious network traffic. The work addresses the critical need for automated defense systems. The deployed Streamlit application bridges the gap between theoretical research and real-world applicability, providing a tool that is both powerful and accessible. Future directions for this work include the integration of time-series analysis for sequential attack detection and the implementation of unsupervised learning modules to handle the challenge of zero-day threats.

Acknowledgment

The authors gratefully acknowledge the invaluable guidance and supervision of Dr. Geethika Sethi and for providing necessary technical mentorship, and domain expertise that made this project possible.

References

[1] N. Moustafa, J. Slay, “The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set,” in *Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS)*, IEEE, 2016.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” *Journal of Artificial Intelligence Research*, vol. 16, pp. 321-357, 2002.

[3] TII-SSRC-23 Dataset. Technology Innovation Institute. Available: https://ieeexplore.ieee.org/document/10262330

[4] Scikit-learn Developers. ”Scikit-learn: Machine Learning in Python.” Available: https://scikit-learn.org/stable/

[5] Streamlit Inc. ”Streamlit Documentation.” Available: https://streamlit.io/

[6] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[7] G. Ke et al., “LightGBM: A Highly Efficient Gradient Boosting Decision Tree,” in *Advances in Neural Information Processing Systems*, 2017.