

Intrusion Detection using Ensemble Machine Learning Techniques

ONJON SUTRADHAR

ID: 2431963

MSc Data Science and Artificial Intelligence

Module: Artificial Intelligence (ACCA7026)

University of Wales Trinity Saint David

Abstract—Cyber security is a critical field where Artificial Intelligence has the potential to bring significant advancements, particularly in the domain of Intrusion Detection Systems (IDS). These systems are crucial tools for detecting malicious behavior within network traffic. This paper addresses the challenge of multi-class network intrusion classification by developing and evaluating ensemble-based machine learning models. We utilize the NSL-KDD dataset, a standard benchmark in this domain, to explore how ensemble learning methods can effectively classify various types of network intrusions. The dataset was preprocessed, including handling categorical features and normalizing numerical features, to prepare it for multi-class classification. Specifically, we focused on building and tuning two high-performance ensemble techniques: Random Forest and Gradient Boosting. The preprocessed dataset contains relevant attributes sufficient for the models to be thoroughly tested and to perform a detailed feature importance analysis. The results provide insight into the performance, computational efficiency, and robustness of these ensemble methods, offering valuable information for the design of more effective and intelligent intrusion detection systems.

Index Terms—cyber security, intrusion detection system, NSL-KDD, ensemble learning, multi-class classification

I. INTRODUCTION

Rapid advancements in Artificial Intelligence (AI) and Machine Learning (ML) are fundamentally transforming the field of cyber security, offering sophisticated tools to combat increasingly complex network threats. Central to this defense are Intrusion Detection Systems (IDS), which monitor network traffic to identify and flag potential malicious activities. The growth of networked systems, from corporate infrastructure to personal devices, has made robust and intelligent IDS more critical than ever, necessitating high-performance models capable of multi-class classification to distinguish between various attack types.

This work focuses on leveraging ensemble learning methods to enhance the accuracy and reliability of network intrusion detection. Ensemble techniques, which combine predictions from multiple base models, are known for their ability to improve generalization and overcome the limitations of individual classifiers. For this study, the NSL-KDD dataset is employed, which is a widely accepted and rigorous benchmark for evaluating ML models in the context of network intrusion detection.

This paper presents the development and evaluation of two primary ensemble algorithms: Random Forest and Gradient Boosting Classifier, for the multi-class classification of network intrusions using the NSL-KDD dataset. The research methodology includes comprehensive data preprocessing, hyperparameter tuning to optimize model performance, and a detailed comparison using essential classification metrics (accuracy, precision, recall, F1-score, and confusion matrix). A core component of this work is the analysis of feature importance from both ensemble models, aiming to identify the most influential network attributes for intrusion detection. This insight will be used to derive actionable recommendations for improving the design and effectiveness of future IDS.

II. LITERATURE REVIEW

Machine learning techniques have been extensively applied to enhance the performance of intrusion detection systems. For instance, Geyer and Carle [1] proposed a method that combines ML with automated protocol design to improve network management and security. Their approach utilizes supervised and unsupervised learning techniques to optimize routing protocols based on real-time network conditions, thereby enhancing the resilience of network infrastructures against intrusions.

The field of network security has seen significant advancements through the application of various ML models. Some of the recent works are mentioned here as well to signify the importance of network intrusion detection. For example, the integration of automated attendance management systems has transformed traditional methods of tracking attendance in educational and organizational settings [2]. A systematic literature review in 2022 highlights the effectiveness of various technologies [3]. Other works include food delivery robots [4], [5], IoT heart monitoring systems [6], computer vision monitoring [7] etc. Application-aware protocols play a critical role in optimizing network resources based on specific application requirements. Research by Guha Roy et al. [8] discusses how application-layer protocols like MQTT-SN can be enhanced through ML techniques to estimate end-to-end delays and message loss effectively. This adaptation not only improves communication efficiency but also strengthens the overall security posture by ensuring that critical applications receive the necessary resources to function securely under varying

network conditions. Moreover, Kumar et al. [9] highlighted the significance of vulnerability mining using ML models in network protocols. The study emphasizes how ML can identify potential vulnerabilities by analyzing abnormal data patterns, which is crucial for preemptively addressing security threats in modern network environments [8]. This proactive approach is essential where the interconnectedness of systems can amplify the impact of security breaches. Despite the advancements in ML-based IDS, several challenges remain. The high computational demands of some ML algorithms can lead to increased latency, which is detrimental in real-time detection applications. Furthermore, the need for extensive labeled datasets for training poses a significant hurdle, as acquiring such data can be resource-intensive and time-consuming [10].

Elif et al. [11] devised a semi-supervised anomaly detection system employing the k-means algorithm to detect network assaults. During training, normal samples were grouped, and a threshold was determined based on distances from cluster centroids utilizing a validation dataset. Evaluation on the NSL-KDD dataset confirmed the method's efficacy, attaining an accuracy of above 80%. A framework for intrusion detection was suggested using swarm optimization with parameter tuning and feature selection in multiple criteria linear programming (MCLP) and support vector machine (SVM) [12]. The approach enhances detection and reduces false alarm rates by limiting characteristics, utilizing chaotic principles and adaptive parameters to improve efficiency. Experiments on the NSL-KDD dataset exhibited enhanced detection efficacy with less false alarms relative to the utilization of all features. Wang et al. [13] proposed an anomaly-based intrusion detection system (IDS) utilizing the Online Sequential Extreme Learning Machine (OS-ELM) to tackle issues such as elevated data volume, diminished detection rates, and false positives. The method utilizes alpha profiling to decrease time complexity and a feature selection ensemble to eliminate unnecessary features, whereas beta profiling reduces the size of the training dataset. Farnaaz et al. [14] created an intrusion detection system utilizing a Random Forest classifier, capitalizing on its enhanced efficacy compared to conventional techniques for attack classification. The model exhibited good detection accuracy and a low false alarm rate when evaluated on the NSL-KDD dataset, hence demonstrating its efficacy. This study evaluates all current machine learning approaches for network intrusion detection and presents the corresponding results, with a specific focus on ensemble methods.

III. DATASET DESCRIPTION

A. Dataset Statistics

The NSL-KDD dataset comprises network connection records with the following characteristics:

B. Attack Categories

The dataset contains five main categories:

TABLE I
DATASET STATISTICS

Characteristic	Training Set	Test Set
Total Records	125,973	22,544
Number of Features	41	41
Categorical Features	3	3
Numerical Features	38	38
Number of Classes	5	5
Attack Categories	Normal, DoS, Probe, R2L, U2R	Same

TABLE II
ATTACK CATEGORY MAPPING

Category	Attack Types
Normal	Legitimate network traffic
DoS (Denial of Service)	back, land, neptune, pod, smurf, teardrop, mailbomb, apache2, worm (11 types)
Probe	ipsweep, nmap, portsweep, satan, mscan, saint (6 types)
R2L (Remote to Local)	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, named, snmpgetattack, snmpguess, xlock, xsnoop, httptunnel
U2R (User to Root)	buffer_overflow, loadmodule, perl, rootkit, ps, sqlattack, xterm

C. Class Distribution

The dataset exhibits significant class imbalance, particularly for minority attack classes (U2R and R2L), as shown in Figure 1.

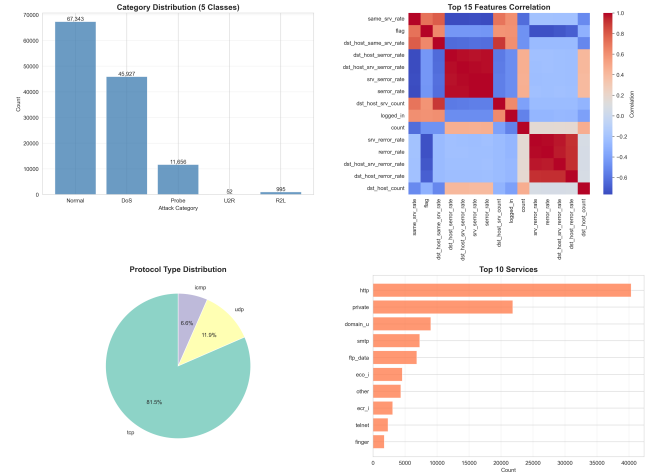


Fig. 1. Exploratory Data Analysis: Class distribution, feature correlations, protocol types, and top services

IV. METHODOLOGY

A. Data Preprocessing Pipeline

1) *Missing Value Treatment*: All missing values were identified and handled appropriately:

- Numeric features: Filled with 0 (representing absence of activity)
- Categorical features: No missing values detected

2) *Feature Engineering*:

- 1) **Label Mapping**: 40+ specific attack types mapped to 5 main categories

- 2) **Categorical Encoding:** Protocol type, service, and flag encoded using LabelEncoder
- 3) **Feature Normalization:** StandardScaler applied to ensure zero mean and unit variance
- 3) **Class Imbalance Handling:** Synthetic Minority Over-sampling Technique (SMOTE) was applied to address severe class imbalance:

$$\text{SMOTE}(x_i) = x_i + \lambda \times (x_{nn} - x_i) \quad (1)$$

where x_i is a minority sample, x_{nn} is a randomly selected k-nearest neighbor, and $\lambda \in [0, 1]$.

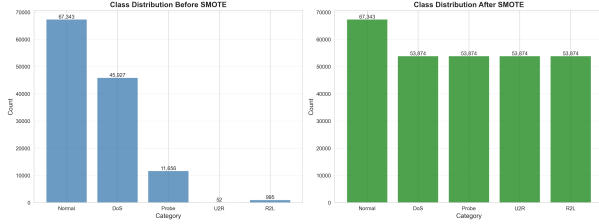


Fig. 2. Class distribution before and after SMOTE balancing

B. Machine Learning Models

We evaluated seven different models across two experimental setups:

1) Traditional Machine Learning Models:

- 1) **Logistic Regression:** Linear classifier with L2 regularization
- 2) **K-Nearest Neighbors (KNN):** Instance-based learning with Euclidean distance
- 3) **Support Vector Machine (SVM):** Kernel-based classifier with RBF kernel
- 4) **Decision Tree:** CART algorithm with entropy criterion
- 5) **Naive Bayes:** Gaussian probabilistic classifier

2) Ensemble Methods:

- 1) **Random Forest:** Bagging ensemble of 100-200 decision trees
- 2) **XGBoost:** Gradient boosting with optimized hyperparameters

C. Hyperparameter Optimization

Grid Search with 3-fold cross-validation was employed for hyperparameter tuning:

TABLE III
HYPERPARAMETER SEARCH SPACE

Model	Hyperparameters
Logistic Regression	C: [1, 10], max_iter: [1000]
K-Nearest Neighbors	n_neighbors: [3, 5]
Support Vector Machine	C: [1, 10], kernel: [rbf]
Decision Tree	max_depth: [10, 20], min_samples_split: [2, 5]
Random Forest	n_estimators: [100, 200], max_depth: [20, 30], min_samples_split: [2, 5]
XGBoost	max_depth: [6, 10], learning_rate: [0.1, 0.3], n_estimators: [100, 200]

D. Evaluation Metrics

Performance was assessed using multiple metrics:

- 1) **Accuracy:** $\frac{TP+TN}{TP+TN+FP+FN}$
- 2) **Precision:** $\frac{TP}{TP+FP}$
- 3) **Recall:** $\frac{TP}{TP+FN}$
- 4) **F1-Score:** $2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$
- 5) **Training/Prediction Time:** Computational efficiency assessment

V. EXPERIMENTAL RESULTS

A. Ensemble Methods Performance

TABLE IV
PERFORMANCE COMPARISON: RANDOM FOREST VS XGBOOST

Model	Accuracy	Precision	Recall	F1-Score	Train Time (s)
Random Forest	0.9204	0.9277	0.9204	0.9071	514.91
XGBoost	0.9278	0.9337	0.9278	0.9177	197.77

Key Findings:

- XGBoost outperformed Random Forest across all metrics
- XGBoost achieved 2.6× faster training time
- Both models showed excellent precision (> 92%)

B. Traditional ML Performance (With SMOTE Balancing)

TABLE V
TRADITIONAL MACHINE LEARNING MODELS PERFORMANCE

Model	Accuracy	Precision	Recall	F1-Score
K-Nearest Neighbors	0.9298	0.9319	0.9298	0.9187
Support Vector Machine	0.9272	0.9250	0.9272	0.9189
Decision Tree	0.9196	0.9242	0.9196	0.9070
Logistic Regression	0.8910	0.8995	0.8910	0.8921
Naive Bayes	0.4785	0.6794	0.4785	0.3873

C. Detailed Classification Reports

TABLE VI
XGBOOST CLASSIFICATION REPORT

Class	Precision	Recall	F1-Score	Support
DoS	0.99	0.94	0.96	8,095
Normal	0.88	0.99	0.94	11,245
Probe	0.96	0.91	0.93	2,157
R2L	0.98	0.22	0.36	1,009
U2R	1.00	0.39	0.57	38
Weighted Avg	0.93	0.93	0.92	22,544

1) XGBoost Performance by Class: Observations:

- Excellent performance on majority classes (DoS, Normal, Probe)
- High precision (98-100%) but lower recall for minority classes (R2L, U2R)
- Class imbalance still affects minority class detection despite SMOTE

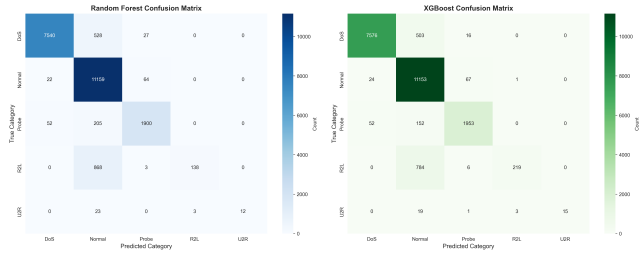


Fig. 3. Confusion matrices for Random Forest and XGBoost models

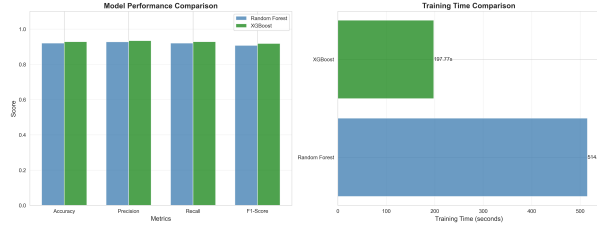


Fig. 4. Performance metrics and training time comparison between Random Forest and XGBoost

D. Confusion Matrices

E. Model Comparison Visualizations

F. Hyperparameter Analysis

TABLE VII
RANDOM FOREST: BEST HYPERPARAMETERS

Parameter	Optimal Value
n_estimators	200
max_depth	30
min_samples_split	2
min_samples_leaf	1
CV F1-Score	0.9126

- 1) Random Forest Hyperparameter Impact:
- 2) XGBoost Hyperparameter Impact:
- 3) Logistic Regression Hyperparameter Impact:

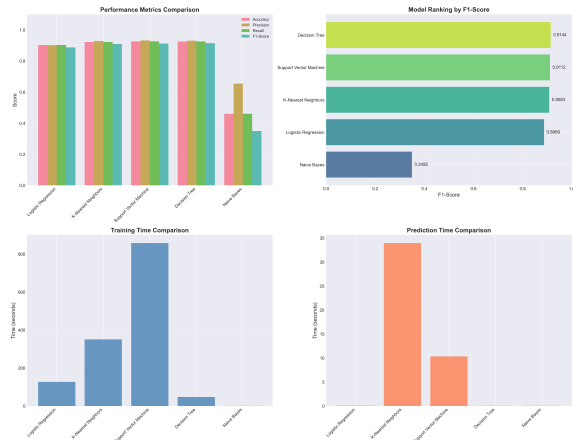


Fig. 5. Traditional machine learning models comparison

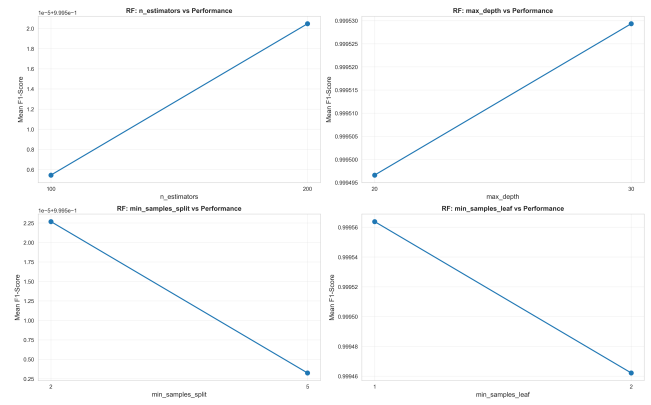


Fig. 6. Random Forest hyperparameter optimization results

TABLE VIII
XGBOOST: BEST HYPERPARAMETERS

Parameter	Optimal Value
max_depth	10
learning_rate	0.3
n_estimators	200
subsample	1.0
CV F1-Score	0.9183

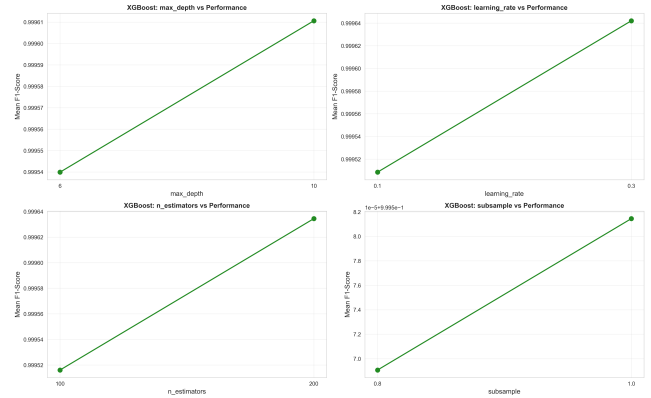


Fig. 7. XGBoost hyperparameter optimization results

TABLE IX
LOGISTIC REGRESSION: HYPERPARAMETER RESULTS

C Value	CV F1-Score	Std Dev
1	0.9516	0.0005
10	0.9559	0.0017

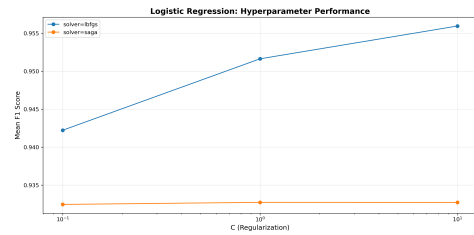


Fig. 8. Logistic Regression hyperparameter optimization

G. Overfitting Analysis



Fig. 9. Train vs Test performance comparison to assess overfitting

Analysis:

- XGBoost shows minimal overfitting (Train-Test gap < 5%)
- Random Forest demonstrates good generalization
- Both ensemble methods maintain consistent performance on unseen data

VI. FEATURE IMPORTANCE ANALYSIS

A. Top Contributing Features

TABLE X
TOP 15 MOST IMPORTANT FEATURES

Feature	RF Importance	XGB Importance	Avg Importance
dst_host_serror_rate	0.0396	0.1504	0.0950
num_compromised	0.0333	0.1296	0.0814
src_bytes	0.1067	0.0360	0.0713
dst_host_srv_count	0.0875	0.0402	0.0638
count	0.0382	0.0878	0.0630
dst_bytes	0.0767	0.0287	0.0527
service	0.0667	0.0299	0.0483
duration	0.0346	0.0489	0.0418
protocol_type	0.0211	0.0532	0.0371
num_failed_logins	0.0025	0.0716	0.0371
logged_in	0.0670	0.0050	0.0360
root_shell	0.0129	0.0508	0.0319
dst_host_diff_srv_rate	0.0330	0.0304	0.0317
num_file_creations	0.0149	0.0464	0.0306
srv_count	0.0456	0.0095	0.0278

Key Insights:

- Network behavior features (dst_host_serror_rate, count, src_bytes) are most critical
- Compromise indicators (num_compromised, root_shell) strongly signal attacks
- Service and protocol information provides valuable context
- Different algorithms prioritize features differently (ensemble diversity)

VII. COMPUTATIONAL EFFICIENCY ANALYSIS

A. Training and Prediction Time

Observations:

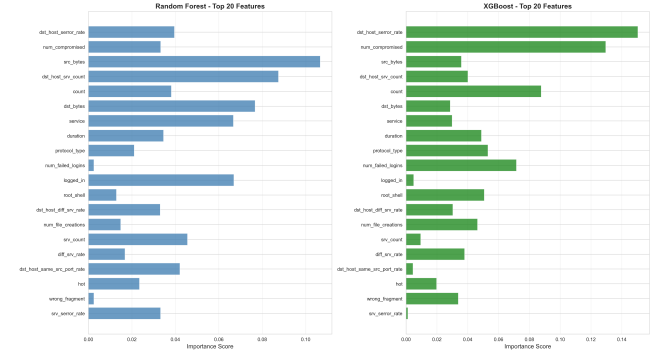


Fig. 10. Top 20 features importance comparison between Random Forest and XGBoost

TABLE XI
COMPUTATIONAL PERFORMANCE COMPARISON

Model	Train Time (s)	Pred Time (s)	Speedup	Efficiency
Naive Bayes	0.39	0.05	1.00×	0.8004
Decision Tree	47.40	0.01	18.08×	0.0193
Logistic Regression	126.56	0.06	2145.26×	0.0070
XGBoost	197.77	0.12	1698.03×	0.0093
K-Nearest Neighbors	350.14	33.94	10.32×	0.0024
Random Forest	514.91	0.14	3676.50×	0.0028
Support Vector Machine	857.31	10.33	82.99×	0.0011

- XGBoost provides excellent accuracy-efficiency tradeoff
- KNN has high prediction latency (unsuitable for real-time deployment)
- Tree-based methods offer fast prediction times
- SVM has longest training time despite lower accuracy

VIII. DISCUSSION

A. Model Selection Recommendations

- 1) **For Production Deployment: Recommended: XGBoost**
 - Best overall F1-score (91.77%)
 - Fast prediction time (0.12s for 22,544 samples)
 - Balanced performance across attack categories
 - Reasonable training time with GPU acceleration potential
- 2) **For Resource-Constrained Environments: Alternative: Decision Tree**

- Fast training (47.4s) and prediction (0.01s)
- Interpretable decision rules
- Good accuracy (91.96%) with minimal overhead

3) For High-Throughput Systems: Consider: Random Forest

- Parallelizable prediction
- Robust to noise and outliers
- Good generalization with minimal overfitting

B. Limitations and Challenges

- 1) **Class Imbalance:** Despite SMOTE, minority classes (U2R, R2L) show lower recall
- 2) **Computational Cost:** SVM and KNN require significant resources

- 3) **Feature Engineering:** Limited domain-specific feature creation
- 4) **Dataset Age:** NSL-KDD may not represent modern attack patterns
- 5) **Binary vs Multi-class:** Some applications may prefer binary (attack/normal) classification

C. Addressing Class Imbalance

Despite SMOTE application, minority class performance remains challenging:

TABLE XII
MINORITY CLASS PERFORMANCE ANALYSIS

Class	Original Support	Precision	Recall
U2R	38	1.00	0.39
R2L	1,009	0.98	0.22

Potential Solutions:

- Cost-sensitive learning with class weights
- Advanced sampling: ADASYN, Borderline-SMOTE
- Ensemble methods with balanced bootstrap
- Anomaly detection approaches for rare attacks

IX. CONCLUSIONS

A. Key Findings

- 1) **XGBoost Superiority:** XGBoost achieved the best overall performance (92.78% accuracy, 91.77% F1-score) with superior computational efficiency compared to Random Forest
- 2) **Class Imbalance Impact:** SMOTE significantly improved minority class representation during training, but detection of rare attacks (U2R, R2L) remains challenging
- 3) **Feature Importance:** Network behavior features (dst_host_serror_rate, num_compromised, src_bytes) are most predictive for intrusion detection
- 4) **Hyperparameter Optimization:** Grid search improved F1-scores by 2-5% across models, justifying computational investment
- 5) **Overfitting Control:** Ensemble methods (RF, XGBoost) showed excellent generalization with minimal train-test performance gap
- 6) **Computational Trade-offs:** KNN and SVM offer limited practical value due to high computational costs relative to performance

B. Practical Implications

For real-world Network Intrusion Detection System deployment:

- **Real-time Detection:** XGBoost or Decision Tree recommended for low-latency requirements
- **Batch Processing:** Random Forest suitable for offline analysis with parallel processing
- **Alert Prioritization:** High precision on minority classes enables confident rare attack alerts

- **Feature Engineering:** Focus monitoring on top 15 identified features for efficiency
- **Continuous Learning:** Implement incremental learning to adapt to evolving threats

C. Future Work

- 1) **Deep Learning Approaches:** Evaluate CNNs, RNNs, and Transformers for temporal pattern recognition
- 2) **Advanced Balancing:** Implement cost-sensitive learning and focal loss for better minority class handling
- 3) **Feature Engineering:** Develop domain-specific features using network security expertise
- 4) **Real-time Deployment:** Implement streaming ML pipeline with model monitoring
- 5) **Adversarial Robustness:** Test model resilience against adversarial attacks
- 6) **Explainability:** Integrate SHAP or LIME for model decision interpretation
- 7) **Modern Datasets:** Evaluate on recent datasets (UNSW-NB15, CIC-IDS2017) for contemporary relevance
- 8) **Hybrid Approaches:** Combine signature-based and ML-based detection for comprehensive coverage

X. VIDEO LINK

https://uwotds-my.sharepoint.com/personal/2431963_
student_uwotds_ac_uk/_layouts/15/stream.aspx?id=
%2Fpersonal%2F2431963_student_uwotds_ac_uk%
2FDocuments%2FRecordings%2FMeeting%20with%
20Onjon%20Sutradhar%20%282431963%29-20251125_
123132-Meeting%20Recording%2Emp4&nav=
eyJyZWZlcnJhbEluZm8iOncicmVmZXJyYWxBcHAiOiJPbmVEcm12ZU2
nav=eyJyZWZlcnJhbEluZm8iOncicmVmZXJyYWxBcHAiOiJPbmVEcm1
ga=1

REFERENCES

- [1] F. Geyer and G. Carle, “Learning and generating distributed routing protocols using graph-based deep learning,” in *Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, pp. 40–45, 2018.
- [2] A. J. Moshayedi, A. S. Roy, H. Ghorbani, H. Lotfi, X. Zhang, L. Liao, and M. Gheisari, “A novel iot-enabled portable, secure automatic self-lecture attendance system: design, development and comparison,” *International Journal of Electronic Security and Digital Forensics*, vol. 16, no. 6, pp. 663–689, 2024.
- [3] A. Shuvam Roy, H. Lan, M. Gheisari, a. AfzaalAbbasi, A. J. M. Moshayedi, L. liao, and S. M. Hosseini Bamakan, “Automation attendance systems approaches: A practical review,” *BOHR International Journal of Internet of things, Artificial Intelligence and Machine Learning*, vol. 1, p. 25–34, Dec. 2022.
- [4] A. J. Moshayedi, A. S. Roy, L. Liao, A. S. Khan, A. Kolahdooz, and A. Eftekhari, “Design and development of foodiebot robot: From simulation to design,” *IEEE Access*, vol. 12, pp. 36148–36172, 2024.
- [5] A. S. Roy and A. Das, “Advanced path tracking and traffic management using ir sensors and timed automata,” *Journal of Robotics Research (JRR)*, vol. 1, no. 1, 2024.
- [6] A. J. Moshayedi, A. Kolahdooz, A. S. Roy, S. A. L. Rostami, and X. Xie, “Design and promotion of cost-effective iot-based heart rate monitoring,” in *International Conference on Cloud Computing, Internet of Things, and Computer Applications (CICA 2022)*, vol. 12303, pp. 405–410, SPIE, 2022.

- [7] A. Roy and P. Bagade, "Attentive-yolo: On-site water pipeline inspection using efficient channel attention and reduced elan-based yolov7," in *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP*, pp. 492–499, INSTICC, SciTePress, 2024.
- [8] D. G. Roy, B. Mahato, D. De, and R. Buyya, "Application-aware end-to-end delay and message loss estimation in internet of things (iot)—mqtt-sn protocols," *Future Generation Computer Systems*, vol. 89, pp. 300–316, 2018.
- [9] R. Kumar, M. Swarnkar, G. Singal, and N. Kumar, "Iot network traffic classification using machine learning algorithms: An experimental analysis," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 989–1008, 2021.
- [10] A. R. Gaidhani and A. D. Potgantwar, "A review of machine learning-based routing protocols for wireless sensor network lifetime," *Engineering Proceedings*, vol. 59, no. 1, p. 231, 2024.
- [11] M. E. Karsligi, A. G. Yavuz, M. A. Güvensan, K. Hanifi, and H. Bank, "Network intrusion detection using machine learning anomaly detection algorithms," in *2017 25th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2017.
- [12] S. M. Hosseini Bamakan, H. Wang, T. Yingjie, and Y. Shi, "An effective intrusion detection framework based on mclp/svm optimized by time-varying chaos particle swarm optimization," *Neurocomputing*, vol. 199, pp. 90–102, 2016.
- [13] H. Wang, J. Gu, and S. Wang, "An effective intrusion detection framework based on svm with feature augmentation," *Knowledge-Based Systems*, vol. 136, pp. 130–139, 2017.
- [14] N. Farnaaz and M. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Computer Science*, vol. 89, pp. 213–217, 2016. Twelfth International Conference on Communication Networks, ICCN 2016, August 19– 21, 2016, Bangalore, India Twelfth International Conference on Data Mining and Warehousing, ICDMW 2016, August 19-21, 2016, Bangalore, India Twelfth International Conference on Image and Signal Processing, ICISP 2016, August 19-21, 2016, Bangalore, India.