Tejashree H Y, 2025, 15:5
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology

An Open Access Journal

Network Intrusion Detection Using Machine Learning: A Comparative Study of Logistic Regression, KNN, and Random Forest

[1]Tejashree H Y, [2]Komala R
[1]Department of MCA, India
[2]Assistant Professor, Department of MCA, India

Abstract- Network Intrusion Detection Systems (NIDS) play a critical role in defending networks against unauthorized access and cyber threats. This paper presents a real-time, web-enabled NIDS built using machine learning techniques to effectively identify and categorize network-based attacks. The system is trained on the NSL-KDD dataset, a refined alternative to earlier datasets, addressing issues like redundancy and class imbalance. We implement and evaluate three supervised learning algorithms—Logistic Regression, K-Nearest Neighbors (KNN), and Random Forest. The workflow includes comprehensive preprocessing, class balancing, and hyperparameter tuning via grid search with cross-validation. Among the models tested, Random Forest achieved the highest detection performance, showing excellent accuracy with minimal false positives. While KNN also produced reliable results, it was comparatively slower. Logistic Regression delivered quick and interpretable outcomes but struggled with complex intrusion patterns. This work contributes a practical, browser-accessible NIDS platform that brings together machine learning capabilities and real- time threat detection.

Keywords - Network Intrusion Detection, Machine Learning, Anomaly Detection, Cybersecurity, Real-Time Monitoring, Packet Analysis, Supervised Learning Based Anonymization, Big Data Privacy.

I. INTRODUCTION

In today’s digital age, ensuring cybersecurity has become a critical priority for individuals, enterprises, and governmental bodies. As networks expand in size, complexity, and interconnectedness, they simultaneously increase the surface area for potential cyberattacks [1]. Cybercriminals exploit vulnerabilities using a variety of tactics such as Distributed Denial-of-Service (DDoS) attacks, unauthorized access, privilege escalation, probing, and data exfiltration [9].As a result, traditional defenses like firewalls and antivirus software are insufficient on their own.In today’s evolving threat landscape, there is a critical need for smart and adaptive systems capable of continuously monitoring, identifying, and reacting to intrusions in real time [4], [9].

An Intrusion Detection System (IDS) serves as a vital tool for recognizing unauthorized or suspicious activities by monitoring events within a network or computing environment [5].Intrusion Detection Systems (IDSs) are typically divided into two primary categories: those that rely on known attack signatures and those that detect unusual or anomalous behavior. Signature-based IDS, such as Snort, rely on known pattern+s or "signatures" of attacks, making them effective against previously seen threats.However, these systems struggle to detect zero-day threats or even slightly altered versions of existing attacks.Furthermore, they oft en require constant updates to their signature database, adding to operational overhead [5].

Machine Learning (ML) techniques have emerged as an effective and scalable alternative to overcome the limitations of traditional signature-based intrusion detection systems [4], [6]. By learning from historical network activity, ML-driven IDSs can recognize novel attack behaviors through the detection of anomalies or deviations from typical traffic patterns [6]. These systems are well- suited for analyzing substantial volumes of network data, identifying relevant patterns, and constructing predictive models that perform reliably in real-world scenarios [7].

This study aims to design and develop a Network Intrusion Detection System (NIDS) using supervised machine learning techniques to detect and categorize different types of network intrusions. The fundamental objective is to incorporate intelligent models into the detection process ML models into a user-friendly, interactive web platform using the Django framework. This platform enables users to upload network traffic data (in CSV format) and receive immediate feedback on whether the traffic is malicious or benign, along with performance metrics of the ML classifiers.

The system utilizes the NSL-KDD dataset, an enhanced and more balanced iteration of the widely recognized KDD Cup 99 dataset [4]. The NSL-KDD dataset overcomes several limitations of the original KDD dataset, including duplicate entries and uneven class distribution, which enhances its reliability for practical model training and assessment [4].The dataset comprises 41 features extracted from network traffic and encompasses various attack types such as Denial of Service (DoS), Probe, Remote to Local (R2L),

and User to Root (U2R), in addition to legitimate or normal network traffic. This study utilizes three widely recognized supervised learning methods—Random Forest, K-Nearest Neighbors (KNN), and Logistic Regression—were selected for implementation and comparison in this project [6], [7], [5].These algorithms were chosen due to their proven capability in solving classification tasks and their widespread use in anomaly detection research. All three algorithms were trained and assessed based on key performance indicators such as accuracy, precision, recall, and F1-score to enable a thorough comparison. Among them, the Random Forest model delivered the highest performance, achieving an impressive 99.79% accuracy and surpassing the others by a notable margin [6].

Beyond model training and testing, this project emphasizes the importance of real-time integration and deployment. The system architecture enables real-time or near real-time anomaly detection by allowing users to periodically upload network traffic logs for analysis.The system incorporates user authentication mechanisms to restrict access and ensure that only verified users can interact with the platform.The backend infrastructure leverages Python’s joblib library for fast model loading and pandas for efficient data handling and preprocessing.The user interface is built with HTML, CSS, and Bootstrap, providing a clean and user-friendly design for easy interaction.Moreover, the system supports advanced features such as visualizing performance metrics (like confusion matrices), tracking prediction history, and logging suspicious activities.Its modular architecture supports scalability, making it adaptable for future enhancements such as integrating deep learning techniques like LSTM or Autoencoders [11], implementing real-time traffic analysis tools like Scapy, and deploying the system in containers via Docker to enhance maintainability and portability.

In conclusion, this study offers an integrated approach that connects machine learning techniques with real-world applications in cybersecurity. The proposed ML-based NIDS system not only enhances the detection of known and unknown attacks but also serves as a robust platform for further academic and industrial research. By combining intelligent analytics with real-time usability, this system provides a step forward in the development of modern, data-driven cybersecurity tools tailored for dynamic network environments [6], [4].

II. LITERATURE REVIEW

Intrusion detection research has advanced considerably, transitioning from fixed signature-based methods to more adaptive and intelligent techniques powered by machine learning. Early systems typically employed statistical anomaly detection or expert-defined rules.Although these techniques offered a certain level of effectiveness,These approaches frequently resulted in numerous false alarms and lacked the flexibility to effectively respond to emerging or evolving cyber threats [2], [5].

K-Nearest Neighbors (KNN) is one of the earliest machine learning classifiers applied in the field of intrusion detection. Its simplicity lies in comparing new data to nearby known instances, which makes it well-suited for identifying intrusions that share similarities with previously observed patterns. However, KNN can be computationally intensive during prediction, which can hinder its suitability for real- time applications [6].

Random Forest is a robust ensemble learning method that aggregates the outcomes of several decision trees to enhance predictive accuracy. It is particularly effective in managing complex and noisy data, making it suitable for intrusion detection scenarios. The model naturally performs feature selection and is less prone to overfitting, which contributes to its reliability in identifying threats like Denial of Service (DoS) and probe attacks while maintaining a low false positive rate [6], [7].

Logistic Regression, a classical statistical model, remains popular due to its simplicity, efficiency, and clear interpretability. It often serves as a baseline when benchmarking more complex models in network intrusion detection. The model's transparent decision boundaries help when understanding and explaining predictions is important [5], [7].

The introduction of the NSL-KDD dataset by Tavallaee et al. [1] marked a milestone in IDS research by resolving issues related to redundant records and imbalanced classes present in the original KDD Cup 99 dataset.This improved dataset offers a more practical benchmark for assessing machine learning models in real-world scenarios. Furthermore, extensive reviews, such as the one by Garcia- Teodoro et al. [2], provide valuable insights into the evolution and effectiveness of intrusion detection techniques.have highlighted the challenges in anomaly- based IDS and recommended hybrid approaches to combine the strengths of signature and anomaly detection.Expanding upon previous research, this study develops and evaluates KNN, Random Forest, and Logistic Regression models within a deployable web-based intrusion detection system, aiming to close the gap between academic research and real-world cybersecurity implementation.

III. METHODOLOGY

To design a robust Network Intrusion Detection System (NIDS), the project followed a systematic methodology including data acquisition, preprocessing, model development, evaluation, and deployment. These steps were essential to ensure that the system could accurately detect various types of network attacks and generalize to real- world scenarios [1], [3].

Dataset Description

This research utilizes the NSL-KDD dataset, an enhanced alternative to the original KDD Cup 99 dataset, which addresses major limitations such as data duplication and skewed class distribution. Due to these improvements, it serves as a more reliable standard for assessing intrusion detection systems. The dataset includes labeled samples that reflect both benign traffic and four key attack categories: Denial of Service (DoS), Probe, User to Root (U2R), and Remote to Local (R2L). Each record consists of 41 features, covering a mix of numerical, categorical, and discrete attributes such as protocol type, service type, connection status flags, and byte-level statistics for source and destination [1], [4].3.2 Data Preprocessing.

Before training, the data underwent several preprocessing steps:

- Data Cleaning: Although mostly clean, the dataset was checked for missing or invalid entries, which were handled appropriately to avoid compromising model performance [1].
- Label Encoding: To make them compatible with machine learning algorithms, categorical attributes like protocol type, service, and flag were converted into numerical format using label encoding [3].
- Feature Scaling:Numerical attributes were scaled
- using the Min-Max normalization technique to standardize their values within the range of 0 to 1, facilitating faster and more stable training [3].
- Attack Label Grouping: To simplify classification, detailed attack types were mapped to their broader categories (DoS, Probe, U2R, R2L), while maintaining "normal" as a separate class [1].

Addressing Class Imbalance

The dataset exhibits imbalance, with fewer samples representing rare attacks such as U2R and R2L compared to the abundant normal traffic. To mitigate this:

- To preserve the original class distribution within the dataset, stratified sampling was employed during the train-test split process [3].
- While more advanced techniques like SMOTE (Synthetic Minority Over-sampling Technique) were taken into account, this study opted for traditional oversampling due to its simplicity and ease of interpretation [8].

Model Selection and Implementation

Three supervised machine learning algorithms were developed and tested using the scikit-learn library in Python:

- Logistic Regression (LR): A linear, interpretable model serving as a baseline classifier [5].
- K-Nearest Neighbors (KNN): An instance-based method that classifies samples based on nearest neighbors, useful for complex decision boundaries [6].
- Random Forest (RF): An ensemble of decision trees that combines predictions to reduce overfitting and improve accuracy, especially suited for high- dimensional data [7].

Hyperparameter Tuning

To optimize model performance, Grid Search combined with 5-fold Cross-Validation was employed, exploring parameters such as:

- For Logistic Regression: regularization strength and solver algorithm [5].
- For KNN: number of neighbors, distance metrics, and weighting schemes [6].
- For Random Forest: number of trees, tree depth, and minimum samples per split [7].

Evaluation Metrics

Models were evaluated on:

- Accuracy, measuring overall correct classification rate [3].
- Precision, reflecting correctness of positive predictions [3].
- Recall (sensitivity), indicating ability to detect actual attacks [3].
- F1-score, which represents the harmonic average of precision and recall, is particularly valuable when dealing with datasets that have class imbalance.
- Confusion matrices, to analyze class-wise performance and common misclassifications [3].

Random Forest demonstrated superior results across all metrics, showing robustness and high sensitivity to various attack types. KNN showed competitive accuracy but slower inference, while Logistic Regression, despite its speed and simplicity, was less effective in capturing complex attack patterns [5], [6], [7].

Dataset Description

The performance of a machine learning-powered Intrusion Detection System (IDS) largely depends on the quality and suitability of the dataset employed for training and testing the models. In this work, the NSL-KDD dataset is used, which is a well-established benchmark in the domain of network intrusion detection [1], [4].

Overview of NSL-KDD

The NSL-KDD dataset is an enhanced version of the original KDD Cup 1999 dataset, developed to overcome its known limitations, specifically designed to overcome major shortcomings present in the initial version [1]:

- Redundancy Reduction: The initial KDD'99 dataset included a significant number of repeated entries,which could bias the learning process and evaluation results. NSL-KDD removes these duplicates to provide a more balanced training and testing environment [1].
- Balanced Class Distribution: The dataset corrects for the uneven representation of attack types seen in KDD'99, where some attacks were overrepresented and others underrepresented. This adjustment helps improve the learning capability for less frequent but significant attack categories [1].

By eliminating redundancies and balancing class frequencies, NSL-KDD offers a more realistic and reliable benchmark for intrusion detection research [1], [4].

Classes of Network Activity
Each record in NSL-KDD is labeled either as "normal" network traffic or as an attack, with attack instances grouped into four main categories [1]:

- (DoS):These attacks target system resources with the intent of exhausting them, thereby preventing access for legitimate users. Examples include smurf, neptune, and back attacks.
- Probe: Surveillance or reconnaissance attacks that scan the network for vulnerabilities or open ports. Common examples are satan and nmap.
- Remote to Local (R2L):This type of attack involves an external attacker trying to obtain unauthorized access to a system from a remote location. Common examples are warezclient and guess_passwd.
- User to Root (U2R): These attacks involve a user with restricted local access attempting to elevate their privileges to gain root or superuser control. Examples include buffer_overflow and rootkit.
- This classification supports both binary classification tasks (normal vs. attack) and multiclass classification involving the different attack types [1].

Feature Set
Each connection record in the dataset is described by 41 features, which can be grouped as follows [1], [4]:

- Basic Features:These features are extracted from packet headers without analyzing the payload and include attributes like duration, protocol type, and source bytes. These features provide general information about the connection.
- Content Features: Extracted by examining the payload contents of packets, these features detect suspicious activity like the number of failed login attempts (e.g., num_failed_logins, hot, logged_in).
- Traffic Features: Calculated using statistics over a sliding window of connections related to the same host or service, helping identify patterns of frequent or unusual traffic (e.g., count, srv_count, dst_host_count).

Features include both numeric and categorical data; categorical features require preprocessing (e.g., label encoding) before use in machine learning models [3].

Dataset Partitions
The NSL-KDD dataset is divided into two main parts for evaluation purposes [1]:

- KDDTrain+: This subset is used for training models and contains a balanced mix of normal traffic and various attack samples.
- KDDTest+: The test subset contains attack types that are absent from the training data, allowing assessment of the model's capability to identify previously unseen or zero-day threats.

Such partitioning reflects real-world scenarios where IDS must effectively recognize both known and previously unseen threats [1].

Evaluation Measures
To evaluate the performance of the proposed machine learning-driven Network Intrusion Detection System (NIDS),we utilized several standard classification metrics. These metrics help assess not only the overall effectiveness of the models but also their capability to accurately identify and distinguish between different types of network attacks [4][6].

Accuracy

Accuracy refers to the proportion of total predictions the model correctly classifies, both for normal and attack categories. It is calculated as:

Accuracy is calculated using the formula:

Accuracy = (True Positives + True Negatives) / (True Positives + True Negatives + False Positives + False Negatives)

where:

TP = True Positives, TN = True Negatives, FP = False Positives,

FN = False Negatives.. While accuracy offers an overall view of a model's performance, it may not be reliable when dealing with imbalanced datasets, as the model might favor the dominant class in its predictions [1][2].

Precision

Precision indicates the proportion of alerts classified as attacks that were actually threats. Within the scope of a NIDS,it indicates how effectively the system reduces incorrect classifications of normal traffic as attacks, it indicates how many of the flagged attacks were actual intrusions:

Precision = TP / (TP + FP)

High precision is crucial in real-time detection systems to avoid false alerts that may overwhelm security teams [4].

Recall (Sensitivity)

Recall evaluates the model's ability to detect actual positives, i.e., how many real attacks were correctly identified:

Recall = TP / (TP + FN)

A high recall value suggests that the model successfully

identifies the majority of intrusion attempts, reducing the likelihood of missing genuine threats [4], [6].

F1-Score

The F1-score merges precision and recall into one metric by computing their harmonic average. It is especially effective in scenarios with imbalanced classes:

F1-Score = 2 × (Precision × Recall) / (Precision + Recall)

This measure is crucial when assessing how well a model identifies less frequent attack types, as it emphasizes minimizing both false alarms and missed detections.

Confusion Matrix

The confusion matrix presents a detailed summary of the model's classification outcomes across all categories— Normal, DoS, Probe, R2L, and U2R—highlighting both correct predictions and instances of misclassification for each class. It reveals how well the classifier distinguishes between different categories and helps pinpoint specific weaknesses. For instance, a high number of false negatives in the R2L class would indicate that the model struggles to detect these subtle intrusions [3].

ROC Curve and AUC

Beyond the conventional evaluation metrics, this study also utilized the Receiver Operating Characteristic (ROC) curve along with the Area Under the Curve (AUC) to assess model performance in binary classification scenarios and its proficiency in identifying diverse attack types. These graphical tools help visualize the balance between true positive and false positive rates at varying threshold levels, where a higher AUC value signifies stronger classification ability [5].

By utilizing these comprehensive evaluation metrics, we were able to gain a robust understanding of the effectiveness ofthe classification models: Logistic Regression, K-Nearest Neighbors (KNN), and Random Forest. This informed our final model selection and optimization strategies for deployment in a real-time network monitoring environment.

IV. ARCHITECTURE EXPLANATION

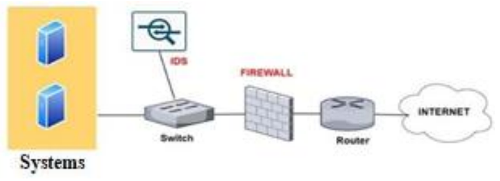

Figure 1: Architecture Design of the Proposed Network Intrusion Detection System (NIDS)

The proposed Machine Learning-based Network Intrusion Detection System (NIDS) is structured using a modular and layered architecture to ensure scalability, maintainability, and real-time responsiveness [1][4]. The system is divided into three main components: the frontend interface, the backend server, and the machine learning models. Each component is designed to perform specific tasks while seamlessly integrating with the others to form a cohesive and efficient pipeline.

Frontend Interface

The frontend is developed using HTML, CSS, and Django's template engine. It serves as the primary interaction point for users, providing a clean and intuitive web interface. Users can upload network traffic files (in CSV format), which may contain records of normal behavior or various types of attacks. The interface also provides feedback to users by displaying the prediction results, enabling them to understand whether the uploaded data points are classified as normal or potentially malicious [6]. User experience is enhanced through the use of simple form validation and alert messages based on prediction outcomes.

Backend Server

- The backend is built using Django, a high-level Python framework known for its efficiency and structured design. When a user uploads a file via the frontend interface, the backend is responsible for managing several essential operations:
- Data Validation: Ensures that the uploaded file adheres to the expected format and contains valid records.
- Preprocessing: Applies transformations to the input data, including encoding of categorical variables,
- feature scaling, and alignment with the trained model's feature schema [2].
- Model Invocation: Based on user input or system default, the backend loads the relevant machine learning model and feeds it the preprocessed data for prediction.
- Response Generation: After prediction, the backend packages the results and sends them to the frontend for display. These results can include labels (e.g., Normal, DoS, R2L) and associated confidence scores or probabilities.

This modular backend design promotes ease of maintenance and enables future integration with external APIs, real-time data streams, or authentication systems for broader applications [3][5].

Machine Learning Models

The machine learning models—Logistic Regression, K- Nearest Neighbors (KNN), and Random Forest—were pre- trained on the NSL-KDD dataset, an enhanced variant of the original KDD'99 dataset, which eliminates redundancy and addresses inherent class imbalance issues [1], [2]. Each model is serialized using Joblib, a Python utility for saving Python objects efficiently. This allows the system to load models dynamically during runtime without retraining.The system allows either automated or user-defined model selection, making it possible to perform comparative analysis of prediction results.During inference, the selected model processes the incoming data and classifies each record into categories such as Normal,Denial of Service (DoS), Probe, Remote to Local (R2L), and User to Root (U2R) attack types.The modularity of this setup ensures that additional models or updates to existing models can be integrated with minimal changes to the backend logic [6].

V. EXPERIMENTAL RESULTS AND ANALYSIS

To assess the effectiveness of the proposed Network Intrusion Detection System (NIDS), a series of detailed experiments were conducted using the NSL-KDD dataset— an established benchmark in the field of intrusion detection [1], [4]. The system was developed and evaluated using three different classification algorithms: Logistic Regression (LR), K-Nearest Neighbors (KNN), and Random Forest (RF), allowing for a comparative performance analysis.

Model Evaluation

Out of the three classification models, Random Forest achieved the best results across all evaluation criteria such as accuracy, precision, recall, and F1-score. A summary of these results is presented in Table 1.provides a summary of the per-class classification metrics for selected classes, including both frequent and minority attack types.

Table 1: Classification Report (Representative Classes)

Class Label	Precision	Recall	F1-Score
Normal	0.96	0.98	0.97
Neptune	0.50	0.11	0.18
Portsweep	0.99	0.99	0.99
Guess_Password	1.00	1.00	1.00
Spy	1.00	0.75	0.86
Warezclient	0.97	0.99	0.98
Smurf	0.99	1.00	1.00

These results indicate that the system is highly reliable in detecting common attacks, with some drop in recall on underrepresented classes such as neptune and spy. This is expected due to inherent class imbalance in the NSL-KDD dataset [2][4].

Metric Comparison: LR vs KNN vs RF

The effectiveness of the three models is assessed using the following evaluation metrics:

- Accuracy: Random Forest (99.8%) outperformed KNN (99.4%) and Logistic Regression (99.3%) in overall classification accuracy.
- Precision: Random Forest achieved the highest precision (0.79), indicating fewer false positives and better correctness in positive predictions compared to KNN (0.73) and Logistic Regression (0.66).
- Recall: With a recall of 0.74, Random Forest again led the models, effectively capturing more true positives than KNN (0.69) and Logistic Regression (0.61).
- F1-Score: Random Forest achieved the highest F1- score (0.76), balancing precision and recall better than KNN (0.70) and Logistic Regression (0.63), demonstrating stronger generalization across classes.

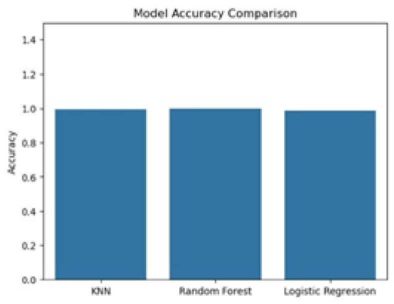

Figure 2: Bar chart comparing Accuracy of LR, KNN, RF

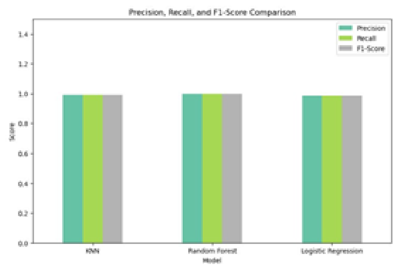

Figure 3: Precision, Recall, and F1-Score Comparison Chart

Figure 2 and Figure 3 illustrate that Random Forest is the most balanced and accurate classifier across all evaluated metrics.

VI. CONCLUSION

This paper presents a comprehensive approach to designing, developing, and evaluating a robust Machine Learning- based Network Intrusion Detection System (NIDS). Utilizing the NSL-KDD dataset [1], the study conducts a comparative analysis of three commonly adopted classification techniques: Logistic Regression, K-Nearest Neighbors (KNN), and Random Forest.

Among these, the Random Forest algorithm consistently achieved the best results across key evaluation criteria, including accuracy, precision, recall, and F1-score. The results highlight the effectiveness of ensemble learning techniques [2] in managing the challenges posed by complex and imbalanced intrusion detection data. Beyond algorithmic comparison, our key contribution lies in the practical deployment of these models within a real- time, web-based platform developed using Django. The system offers a complete suite of functionalities that reflect real-world security needs, including user input-based detection, role-based access control, real-time packet monitoring, and dynamic dashboards for visualizing trends. It incorporates advanced capabilities such as anomaly scoring [3], protocol-specific model switching, IP geolocation mapping [4], automatic retraining, ensemble modeling, and integration with threat intelligence APIs. These features not only improve detection accuracy but also enhance operational usability, scalability, and responsiveness to modern attack vectors.
The platform is secured through role-based authentication, and deployment is simplified using Docker [5], making it portable and adaptable across environments. The inclusion of real-time alerting mechanisms, exportable reports, and an intuitive UI with dark mode support ensures that the system is not only technically effective but also user-friendly and ready for enterprise adoption.

To improve the system in the future, deep learning architectures such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks can be incorporated to enhance detection capabilities [6], implementing real-time traffic capture and stream analysis, and scaling the deployment using cloud-native tools like Kubernetes [7]. Further integration with SIEM tools [8] and the addition of online learning [9] for continual improvement will further elevate the platform's capabilities.

In summary, this research delivers a practical, intelligent, and extensible solution for intrusion detection by bridging machine learning theory with real-world cybersecurity applications, and it sets a strong foundation for future innovations in proactive threat mitigation.

REFERENCES

1. Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, 1–6.
2. Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32.
3. Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys (CSUR), 41(3), 1–58.
4. MaxMind Inc. (2024). GeoIP2 Databases. [Online]. Available: https://dev.maxmind.com/geoip
5. Merkel, D. (2014). Docker: Lightweight Linux containers for consistent development and deployment. Linux Journal, 2014(239), 2.
6. Kim, Y. (2014). Convolutional neural networks for sentence classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1746–1751.
7. Hightower, K., Burns, B., & Beda, J. (2019). Kubernetes: Up and Running (2nd ed.). O'Reilly Media.
8. Scarfone, K., & Mell, P. (2007). Guide to intrusion detection and prevention systems (IDPS). NIST Special Publication 800-94.
9. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. ACM Computing Surveys (CSUR), 46(4), 1–37.