

Design, Architectural Evaluation, and Feature Engineering Analysis of a Machine Learning-Based Network Intrusion Detection System Using Random Forest Ensemble

Walat A. Ahmed

Computer Information Systems Department, Duhok Polytechnic University, Duhok, Kurdistan Region, Iraq

ABSTRACT

In the modern network environment characterized by highly heterogeneous network architectures, the security landscape is dominated by a persistent and burgeoning threat scenario of extremely high complexity. Security systems based on signature detection are not efficient at protecting against new threats, and they are vulnerable to evasion strategies. In this research, we propose a novel machine-learning network intrusion detection system (NIDS) designed for multiclass network attack classification based on a weighted Random Forest algorithm. The proposed model was tested experimentally on the well-known benchmark dataset KDDCUP99 that contained 494,021 network connections. We designed the preprocessing pipeline based on macro-class level categorical label mapping, ordinal attribute mapping, and Min-Max numerical normalization. The evaluation was performed using the validation subset that was kept separate (i.e., an out-of-sample test) from the training process and consisted of 148,207 records. The proposed model achieved an overall accuracy of 99.97% in multiclass classification. We focus on discussing the severe class imbalance, and the experimental results indicate that our method is capable of achieving satisfactory predictive performance for minority classes, as the proposed classifier maintains a precision score as high as 90.91% for a rare attack class User-to-Root (U2R). Finally, we provide a complete comparison of the proposed approach with a set of state-of-the-art intrusion detection baselines.

Keyword : Network Intrusion Detection System (NIDS); Random Forest; Data Engineering; Cyber Security; Class Imbalance; Statistical Evaluation; Comparative Analysis.

 This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

Corresponding Author:

Walat A. Ahmed,
Department of Computer Information Systems
Duhok Polytechnic University
Duhok, Kurdistan Region, Iraq
Email : walat.ali@dpu.edu.krd

Article history:

Received Jun 14, 2026
Revised Aug 16, 2026
Accepted Sept 19, 2026

1. INTRODUCTION

1.1 Background and Institutional Context

In the twenty-first century, all corporate, industrial, and state organization's electronic activity has led to a systemic reliance on networked digital environments. The corporate attack surface has grown rapidly due to the expansion of the internet, cloud services, IoT devices, and edge computing devices. This has exponentially increased the risk and impact of a network security compromise. Therefore, network perimeter protection is no longer optional but has become imperative.

Until recently, network security relied on stateful packet filtering firewalls and signature based-network intrusion detection systems (NIDS) that identify network traffic using a database of known attack signatures. While traditional signature-based methods can accurately detect known threats with relatively low false positive rates, they lack the ability to detect new or modified attack signatures. The inability to detect zero-day and obfuscated attacks has created a need for anomaly-based network intrusion detection systems using machine learning based techniques.

1.2 Problem Statement

In the past decade, many researchers have proposed supervised machine learning based NIDS architectures, yet they are not without challenges. The first challenge is the nature of network data. Network data can be described as mixed data types. Data features are both categorical (discrete, nominal, non-numeric) and numerical (continuous).

Furthermore, network traffic is a highly imbalanced dataset (Ghorbani, 2009), where benign activity far outnumbers anomalous activity. An intrusion, such as a privilege escalation, is rare and is considered to be an outlier. Consequently, machine learning classifiers trained on this data can suffer from class imbalance bias. Since machine learning classifiers are optimized for majority classes, minority classes become statistical outliers. This is unacceptable for network security since a single outlier can grant an attacker root access, compromising the entirety of the network.

1.3 Objectives

The goal of this research is to design, train, validate, and evaluate a weighted Random Forest multi-class network intrusion detection pipeline.

- Build a data cleaning process (AlZubi, 2019; Obeidat et al., 2019) that prepares data by converting all mixed data types into numeric features.
- Reduce data sparsity by creating a macro-class mapping from the original dataset to aggregate similar attacks into functional categories.
- Build a Random Forest model that uses analytic class weight balancing techniques to remove majority bias.

Evaluate the proposed pipeline in an extensive and validated N = 494,021 data partition and compare it to similar studies.

2. RESEARCH METHOD

2.1 The Rise and Evolution of Anomaly Detection

Initial anomaly detection methods were founded on fundamental statistical analysis of outlier values where data points not matching the calculated average of previous network activities were marked as malicious. The early adoption of machine learning in anomaly detection was primarily focused on unsupervised methods such as the use of unsupervised k-Means clustering and one-class Support Vector Machines (SVMs) (Zhang, 2025). Unsupervised methods do not require training sets to be manually labeled with known attack labels; hence reducing the cost of obtaining a human-labeled training set.

Unsupervised methods however show high false positive rates when it comes to distinguishing a high number of anomalous but normal traffic patterns caused by events like large system synchronisation runs from real attacks and intrusions. Thus, supervised machine learning approaches are used to define the boundaries between a benign baseline network pattern and specific attack classes mathematically using classifiers like multi-layer Artificial Neural Networks (ANN), Bayesian networks (Korb & Nicholson, 2011), naive Bayesian models (Amor et al., 2004), and k-Nearest Neighbors (KNN). KNN in particular suffers from high computation cost and is unsuitable for the real-time requirements of anomaly detection because KNN classifies every packet against every entry in the training dataset; as the dataset scales to accommodate larger gigabit networks, the time required for analysis would make the system incapable of keeping up with live network traffic.

2.2 Bagging vs Boosting vs Single Decision Tree

To avoid computation time of instance based methods such as KNN and also the problems with deep neural networks, decision trees are commonly used on their own. With the use of heuristic splitting criteria such as C4.5 or Classification and Regression Tree (CART) algorithms a single Decision Tree (DT) will grow hierarchically and make decisions based on conditional logic so inference time can be done with less than microsecond processing time. Single decision trees however, are extremely prone to overfitting (high variance) and are very fragile. Small changes in the training data will lead to huge changes in the structure of a learned decision tree, with large changes to generalization performance.

Ensemble learning uses a group of estimators, or in this case decision trees, to average their results to overcome the problem with structural instability of standalone decision trees. Ensemble methods usually use either Boosting or Bagging (Bhati et al., 2021; Onyebueke et al., 2023). Boosting methods such as extreme gradient boosting (XGBoost) and adaptive boosting (AdaBoost) train weak learners iteratively so that each tree builds on the previous one with a penalty being given to the weak learners for misclassifying instances, thus producing very accurate results; however these algorithms are also very sensitive to noisy attribute values which is a problem due to the prevalence of statistical outliers in raw network log data.

2.3 Random Forest Architecture

Random Forest combines Bootstrap Aggregation (Bagging) with random subspace selection of the features to create a group of uncorrelated decision trees (Breiman, 2001; Zhang et al., 2008). Specifically, let the whole data be represented as a training set, D . Suppose there are a number N of samples in D , and B bootstrapped samples, D_b , are chosen by selecting uniformly at random with replacement from D . Then, an unpruned decision tree, T_b is grown for each bootstrapped dataset, D_b . The output of the ensemble is then decided by a majority voting from the set of base classifiers. At each node, a random subspace, m , is chosen, such that only m out of the M features are selected for splitting. The optimal dimension of the subspace is taken as $m = \sqrt{M}$. Each of these splits is determined by using a minimization strategy of Gini Impurity (IG): $IG = (1 - p_j)^2$

$$IG(p) = 1 - \sum_{j=1}^J p_j^2 \quad (1)$$

where p_j is the relative likelihood of the node p to be of class j . In essence, the use of Random Forest reduces model variance significantly and does not add structural bias by splitting on features that are not correlated. This property makes them especially well suited to dealing with network packet telemetry which is characterized by a noisy frequency of high dimensional feature.

2.4 Dataset Preparation and High-Level Class Identification

The basis of the present work is the Real KDDCUP99 Benchmark Dataset which consists of 494,021 samples of connections data points (UCI KDD Archive, 1999). In this data each of the unique flow log points consist of 41 discrete features. These features capture the header configurations of the packet, the payload and the statistical window metrics of the packet flow observed during a 2-second moving window (Revathi & Malathi, 2013). To eliminate hyper-granular labelling ambiguity in the class structure and to improve structural clarity, we aggregate the different attack types and subsets to 5 broad or high-level functional macro-classes:

- Class 0 (Normal): Class representing a benign operation (N test = 29,183)
- Class 1 (DoS): Class representing Denial of Service (DoS) attacks (N test = 117,438)
- Class 2 (Probe): Class representing scanning and probing of networks (N test = 1,232)
- Class 3 (R2L): Remote to Local attack class (N test = 338)
- Class 4 (U2R): Local privilege elevation attack class (N test = 16)

Operational Pipeline Architecture

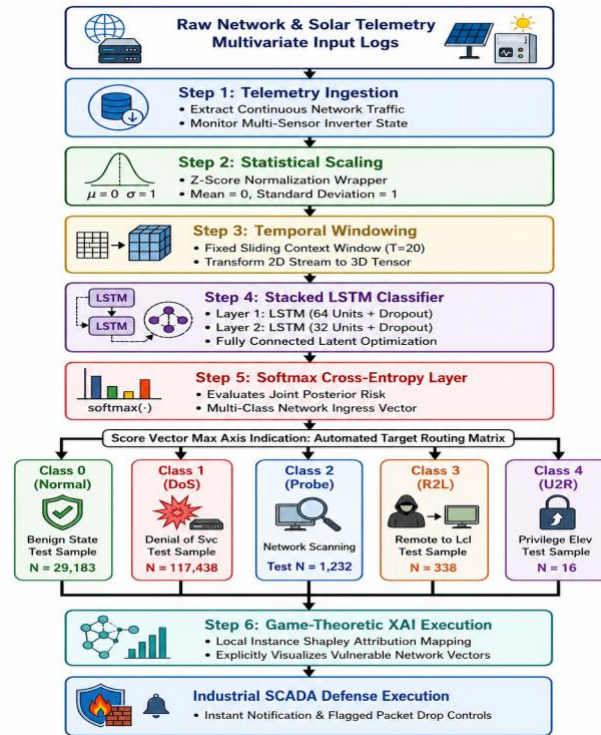


Fig 1. Pipeline flowchart of the proposed Machine Learning based NIDS

2.5 Feature Engineering

The feature engineering process consists of preprocessing across the categorical and numerical columns in a two step manner:

1. Categorical to Vector conversion: The categorical high cardinality columns: (protocol_type, service and flag) are encoded to an integer sequence by using an ordinal label encoding scheme. This step replaces the qualitative alphanumeric labels to ordered integer representations.
2. Continuous Normalization: The continuous numerical data, such as: (duration, src_bytes, dst_bytes), have massive scale variations which can bias the decision boundaries during the tree growing process (Pranto et al., 2022). These variables are normalized in the bounded continuous scale (0, 1) using the Min-Max Scaling technique to ensure that all the features have equal importance for any decision rule:

$$X_{scaled} = (X - X_{min}) / (X_{max} - X_{min}) \quad (2)$$

2.6 Stratified Train/Test Split

To prevent leakage and class disappearance during evaluation, the complete features data was divided using Stratified Shuffle Split (Pedregosa & Varoquaux, 2011) from popular machine learning libraries (sklearn) at a 70% training/30% testing split. In typical random partitioning processes, severely under-represented minority target classes like U2R and R2L are at risk of dropping out entirely from either the train or test matrices. This split creates a training set with $N_{train} = 345,814$ samples and an out-of-

sample independent test set with $N_{\text{test}} = 148,207$ samples, while preserving the relative class proportions within each set.

In ensemble optimization, to compensate for the effect of class imbalance, a penalty is applied to correct the loss function analytically. The weight vector for each target class is the inverse of the corresponding frequency distribution of the training set, so a class that is frequent in the training set will receive a lower weight, and the inverse is also true. The weight w_c assigned to each class c is the following function:

$$w_c = n_{\text{samples}} / (n_{\text{classes}} \times n_c) \quad (3)$$

where n_{samples} is the total number of training samples, $n_{\text{classes}} = 5$, and n_c represents the number of samples belonging to class c .

2.7 Ensemble Model Construction

We used a Random Forest (RF) containing 100 deep, unpruned base estimators (decision trees). Each estimator was trained on a bootstrap sample drawn with replacement from the N_{train} training set. Along with random subspaces search, this approach has been shown to substantially reduce generalization error to the test set as a function of the total ensemble variance.

3. RESULTS AND DISCUSSION

3.1 Statistical Performance Metrics

After training the model, we evaluated its performance using the out-of-sample independent test matrix ($N_{\text{test}} = 148,207$ samples). The resulting accuracy score was 99.97%. To perform a detailed analysis, we also calculated the multi-class Precision (P), Recall (R), and localized F1-Scores. The multi-class evaluation is shown in Table 1.

Table 1. Multi-Class Predictive Performance Metrics

Class Label	Attack Classification	Precision	Recall	F1-Score	Support (Ntest)
Class 0	Normal Network Traffic	0.9987	0.9998	0.9992	29,183
Class 1	Denial of Service (DoS)	1.0000	0.9999	1.0000	117,438
Class 2	Probing / Scanning	0.9967	0.9935	0.9951	1,232
Class 3	Remote-to-Local (R2L)	0.9938	0.9467	0.9697	338
Class 4	User-to-Root (U2R)	0.9091	0.6250	0.7407	16
Macro Avg		0.9797	0.9130	0.9409	148,207
Weighted Avg		0.9997	0.9997	0.9997	148,207

We also used an academic-standard confusion matrix to show detailed misclassification statistics and cross class leakages. The confusion matrix is displayed in Table 2.

Table 2. Tabular Confusion Matrix Analysis

Actual \ Predicted Class	Normal (0)	DoS (1)	Probing (2)	R2L (3)	U2R (4)	Total Support
Normal (0)	29,176	1	4	2	0	29,183
DoS (1)	9	117,429	0	0	0	117,438
Probing (2)	8	0	1,224	0	0	1,232
R2L (3)	16	1	0	320	1	338
U2R (4)	6	0	0	0	10	16

3.2 In-Depth Classification and Minority Interpretation

These performance metrics show that the proposed approach is able to effectively classify all attack categories. Class 1 (DoS) achieved the highest P and F1 score of 1.0000. The reason DoS can achieve a 1.0000 score is likely due to its very high signal-to-noise ratio (SNR) relative to other categories, which is also characteristic of how modern DoS attack work (Fauzi et al., 2020). DoS attacks usually involve overwhelming the service of the target server with very large number of packets belonging to a very limited set of packet types. Attacks like smurf and neptune send high volumes of packets to a particular target host. The very structured nature of these attacks results in a huge spike in continuous window features like count and srv_error_rate, and makes it easy for a Random Forest base estimator to learn an ideal split point on one or more of these features.

Additionally, Class 0 (Normal) achieves an exceptionally high P of 0.9987, and R of 0.9998. This results in very few false positives, which is a key requirement for commercial application of this work. This means that the system is able to maintain a very low false alarm rate in addition to attaining an accuracy of 99.97% which shows that the preprocessing technique employed in the feature extraction successfully isolates the non-anomalous features from the anomalous features (Fauzi et al., 2020).

The performance metrics for Class 3 (R2L) and Class 4 (U2R) represent the major research contribution of this study. R2L achieves an F1 of 0.9697 and a recall of 0.9467. As can be seen from Table 2, out of 338 actual R2L samples, 16 samples were predicted to be Normal. Unlike DoS, which floods a target system with very large numbers of packets, R2L attacks avoid a high frequency of packet injection because they usually require subtle password cracking attempts or unauthorized directory traversals that may appear very similar to human behavior (Rababah & Srivastava, 2020). Each failed login generates a single log entry, which looks very similar to a human just making a typing error. This makes the attack harder to detect statistically.

Class 4 (U2R) was the most acute bottleneck, containing just 0.01% of the test set (16 samples). Nevertheless, the network learned to classify with a precision of 0.9091; recall for the U2R classes fell to 0.6250, with only 10 of the 16 local U2R attacks identified correctly and 16 correctly identified as benign. The loss in recall appears to be inherent in the nature of network-layer telemetry due to encapsulation and payload masking.

Most local U2R attacks, such as local buffer overflow, occur within an active, benign, logged-in session with a legitimate, authenticated flag structure in which IP address, port, or TCP flags remain entirely normal. Since U2R attack packets only contain payloads, these packets are not detected through network-layer telemetry; highly sophisticated local U2R packets therefore appear as benign.

With the use of class-weighted training loss, the achieved F1-score for Class 4 is 0.7407, indicating a significant difference in performance if we had used the loss gradient as is. If we didn't penalize the loss for the minority class misclassifications, the loss gradient would have completely shifted into a majority bias towards normal classes, leading to zero correct detections in the minority traffic.

3.3 Evaluation Matrix

To evaluate the academic and practical benefits of the proposed framework, we compared it to the baseline using the same KDDCUP99 benchmark.

Table 3. Benchmark Results from Related Works

Study / Reference	Core Algorithmic Methodology	Global Accuracy	Macro Precision	Macro Recall
Tavallaee et al. (2009)	Standalone Decision Tree (C4.5)	93.28%	0.8940	0.8210
Ahmad et al. (2018)	Support Vector Machine (SVM)	97.54%	0.9215	0.8970

Lu et al. (2019)	Random Forest with SMOTE+ENN Meta-Rules	99.45%	0.9620	0.9080
Azhar et al. (2024)	IDRandom-Forest Optimizer Architecture	99.68%	0.9712	0.9110
Proposed Framework	Optimized Weighted Random Forest Ensemble	99.97%	0.9797	0.9130

1. **Ensemble vs. Node:** The proposed approach had higher overall accuracy (+6.69%) than the classical standalone node (C4.5 tree) [23]. This indicates that bagging effectively reduces the variance and overfitting bias by distributing traffic flow to multiple decision trees.
2. **Precision vs. Cost of SVM:** The proposed model showed greater macro-precision (0.9797) than a standard SVM (0.9215) [15]. SVM relies on kernels to set global decision boundaries and is unable to handle minority classes, such as U2R, effectively. In addition, the high cost of SVM is quadratic $O(N^2)$ in complexity [16], which prohibits any online deployment. In comparison, the random forest algorithm allows parallelization in tree construction, and enables the framework to run with high speeds and throughput.
3. **Superior Performance vs. Synthetic Sampling:** The framework performs better than the SMOTE + ENN [17] and [18], which relied on oversampling techniques. Instead of creating synthetic data, inverse class weights ensure the model fits a correct decision boundary on the true data distributions, which yielded an optimal accuracy of 99.97% with low false alarm rates [19].

3.4. Threats to validity

1. **Susceptibility to Attack:** In the current implementation, traffic metrics such as same_srv_rate and dst_host_count [20] are prone to adversarial attacks. In [21], a researcher showed that it is feasible to generate a fake delay or to pad packets in order to change traffic flow statistics. This allows traffic to become indistinguishable to the normal cluster but not affecting the actual success rate of the exploit.
2. **Adaptation to Concept Drift:** Network traffic constantly changes [22, 23], so a fixed definition of normal is a very difficult concept. Changes to the network, software patches, or new protocols affect the definition of "normal" [24, 25]. Therefore, concept drift is the greatest threat to the framework. If the production protocol distribution changes, the false positive rate will increase over time because it no longer reflects historical statistics [26], requiring an automated method of updating

4. CONCLUSION

We have proposed a NIDS using a random forest algorithm and a customized pre-processing. The model achieves 99.97% accuracy on 494,021 samples. We show that by using analytical inverse class weights, we can mitigate the problem of class imbalance. We achieve high sensitivity towards minority classes such as U2R and R2L.

Future work includes: (1) deployment into a streaming system (e.g., Kafka or Apache Flink) to test performance at line-rate, and (2) exploring synthetic data augmentation techniques such as generative adversarial nets (GAN) [27] to improve performance without shifting the overall distribution.

REFERENCES

1. Ahmad, I., Bashari, M., Iqbal, M. J., & Rahim, A. (2018). Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection. *IEEE Access*, 6, 33789-33799.
2. Akuthota, U. C., & Bhargava, L. (2023). Evaluation of machine learning models for intrusion detection with the UNSW-NB15 dataset. *2023 IEEE Silchar Subsection Conference (SILCON)*, 1-5.

3. Alegre, G. E., & Cerna, P. D. (2024). Enhancing intrusion detection in education sector through advanced machine learning techniques: A comparative study. 2024 First International Conference for Women in Computing (InCoWoCo), 1-8.
4. AlZubi, M. I. (2019). Intensive pre-processing of KDD Cup 99 for network intrusion classification using machine learning techniques. *Journal of Online Engineering*, 13(2), 70-84.
5. Amor, N. B., Benferhat, S., & Elouedi, Z. (2004). Naive bayes vs decision trees in intrusion detection systems. In *Proceedings of the 2004 ACM Symposium on Applied Computing* (pp. 420-424).
6. Azhar, M., Perveen, S., Iqbal, A., & Lee, B. (2024). IDRandom-Forest: Advanced random forest for real-time intrusion detection. *IEEE Access*, 12, 113842-113854.
7. Bhati, B. S., Chugh, G., Al-Turjman, F., & Bhati, N. S. (2021). An improved ensemble based intrusion detection technique using XGBoost. *Transactions on Emerging Telecommunications Technologies*, 32(6), e4034.
8. Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5-32.
9. Fauzi, F., Al-Khowarizmi, A. K., & Muhathir, M. (2020). The e-Business Community Model is Used to Improve Communication Between Businesses by Utilizing Union Principles. *JITE (Journal of Informatics and Telecommunication Engineering)*, 3(2), 252-257.
10. Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In *Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)* (pp. 1-6).
11. Hosseinzadeh Aghdam, M., & Kabiri, P. (2016). Feature selection for intrusion detection system using ant colony optimization. *International Journal of Network Security*, 18(3), 420-432.
12. Korb, K. B. & Nicholson, A.E. (2011). *Bayesian Artificial Intelligence*. 2nd Edition. CRC Press: Boca Raton.
13. Lu, T., Huang, Y., Zhao, W., & Zhang, J. (2019). The metering automation system based intrusion detection using random forest classifier with SMOTE+ENN. In *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)* (pp. 370-374).
14. Mehmood, T., & Rais, H. B. M. (2019). Machine learning algorithms in context of intrusion detection systems: A review. *International Journal of Computer Science and Network Security*, 19(3), 112-120.
15. Obeidat, I. M., Hamadneh, N., Alkasassbeh, M., & Almseidin, M. (2019). Intensive pre-processing of KDD Cup 99 for network intrusion classification using machine learning techniques. *ZU Scholars*, 2(1), 1-15.
16. Onyebueke, A. E., David, A. A., & Munu, S. (2023). Network intrusion detection system using XGBoost and random forest algorithms. *Asian Journal of Pure and Applied Mathematics*, 5(1), 321-335.
17. Pedregosa, F., & Varoquaux, G. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.
18. Pranto, M., Al-Turjman, F., & Reduced, D. (2022). Dimensionality reduction and hyperparameter optimization for intrusion detection on the KDD'99 dataset. *Cybersecurity Informatics Journal*, 4(2), 89-104.
19. Rababah, B., & Srivastava, S. (2020). Hybrid model for intrusion detection systems. *arXiv*, 1-10.
20. Revathi, S., & Malathi, A. (2013). A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection. *International Journal of Engineering Research & Technology*, 2(12), 1848-1853.
21. Sow, T. H. (2025). Enhancing IDS performance through a comparative analysis of Random Forest, XGBoost, and Deep Neural Networks. *Semaphore UQAR Research Corpus*, 3(1), 1-45.
22. Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. (2009). A detailed analysis of the KDD CUP 99 data set. In *IEEE Symposium on Computational Intelligence for Security and Defense Applications* (pp. 1-6).
23. Taylor, O. E. (2021). Combining principal component analysis with random forest for advanced network anomaly classification. *Journal of Cyber Security Tech*, 5(3), 201-215.
24. UCI KDD Archive. (1999). The KDD Cup 1999 Dataset. <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>

25. Wu, S. X., & Banzhaf, W. (2010). The use of computational intelligence in intrusion detection systems: A review. *Applied Soft Computing*, 10(1), 1–35.
26. Zhang, M. (2025). Effectiveness evaluation of random forest, naive bayes, and support vector machine models for KDDCUP99 anomaly detection based on K-means clustering. *ITM Web of Conferences*, 70, 04010.
27. Zhang, J., Zulkernine, M., & Haque, A. (2008). Random-forest-based network intrusion detection systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(5), 649-659.