SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V12I8P111

Volume 12 Issue 8, 123-137, August 2025

Original Article

Real-Time Intrusion Detection Using Adaptive Sliding Windows for Concept Drift

S. Meganathan[1], A. Sumathi[2], S. Sheik Mohideen Shah[3], R. Rajakumar[4]

[1,2,3,4]*Departmen of Computer Science and Engineering, Srinivasa Ramanujan Centre, SASTRA University, Tamilnadu, India.*

[4]*Corresponding Author : rajakumar@src.sastra.edu*

Received: 06 June 2025 Revised: 07 July 2025 Accepted: 08 August 2025 Published: 30 August 2025

Abstract *- Real-time intrusion detection has become crucial as a result of the rapid development of networks. This is because the distribution of the data and behaviors changes over time, a phenomenon known as concept drift. In order to address the data drift, this study suggests that an online adaptive sliding windowing method is used to tackle the concept drift, which gives a timely response for incoming data packets. Initially, collected different Intrusion Detection System (IDS) datasets like NSL KDD from the concerned repository and dataset evaluated by conventional machine learning models such as Logistic Regression(LR), Random Forest (RF), Decision Tree(DT), K-Nearest Neighbor (KNN), Gradient Boosting classifiers (GBT), and Light(GBM) and the results showed low detection rate 70.24%, 76.77%, 76.83%,77.20%,78.02% and 79.35% due to training and testing datasets consists of unequal class distribution and concept drift. Applying the oversampling, Undersampling, and SMOTE approaches, the accuracy was somewhat improved to 79.77% and 81.21% while using Synthetic Minority oversampling techniques (SMOTE) to address the majority and minority issues known as data imbalance. To further enhance the detection rate, developed a proposed model called Adaptive Drift-aware Windowing Intrusion Detection System with Optimization (ADWISE) was developed, combining adaptive sliding windows with random search hyperparameter tuning optimization. The proposed ADWISE framework achieves a top accuracy of 98.27% while effectively managing both class imbalance and concept drift..*

Keywords *- Concept Drift, Class Imbalance, Adaptive Machine Learning, Drift Detection, Streaming Data Analytics. Real-Time Learning.*

1. Introduction

Due to the rapid development of the digital landscape, securing network systems against malicious activity is a top priority. As cyber threats become more sophisticated, the need for real and responsive Intrusion Detection Systems (IDS) has grown significantly. Conventional IDS approaches, which mostly rely on offline or static data processing, are unable to handle the dynamic nature of modern network traffic. One of the major challenges in real-time IDS is the presence of concept drift, a phenomenon where the statistical characteristics of the input data vary over time, and degrade the performance of the degradation model. In streaming environments where attack patterns evolve and legitimate behavior may shift, this is especially problematic.

Class imbalance is another significant issue with IDS data**,** when benign traffic vastly outnumbers harmful instances. This imbalance can bias classifiers in favor of the majority class, reducing the system's ability to detect rare but critical attack events. Conventional machine learning models, when trained on such imbalanced and non-stationary data, tend to produce large false negative rates and poor detection rates, which restricts their use in practical situations.

In order to overcome these issues, this study suggests an online adaptive sliding window framework (ADWISE) that can handle idea drift and class imbalance while detecting intrusions in real-time. The approach leverages a dynamic windowing technique that adapts to changes in data distribution, enabling timely model updates when new data becomes available. The benchmark IDS dataset (NSL-KDD) was used to test standard classifiers such as LR, RF, DT, KNN, GBT and LightGBM. The results showed lower performance due to the aforementioned problems.

To improve classification accuracy, data balancing strategies such as oversampling, Undersampling, and the Synthetic Minority Over-sampling Technique (SMOTE) were investigated. While SMOTE produced moderate improvements, the study introduces a novel enhancement, an integrated framework combining the adaptive sliding window method with random search for hyperparameter tuning.

This proposed framework demonstrates robust performance in real-time environments, attaining noticeably improved detection accuracy while adjusting to changing attack patterns and addressing class imbalance effectively.

2. Related Works

Class imbalance and Concept drift in real-time or streaming data have been an emerging research challenge, especially in applications such as Intrusion Detection Systems (IDS). To meet the adaptive requirements of learning models in non-stationary environments, a variety of frameworks and algorithms have been put forth.

2.1. Concept Drift in Data Streams

Concept drift refers to the description of how the distribution of underlying data varies over time, leading to model degradation. A thorough analysis of concept drift adaptation techniques was provided by Gama et al. [17], who made a difference between passive and active mechanisms.

Zhang et al [6] presented a multilayer drift detection technique that uses model explainability to enhance interpretability and adaptability in dynamic scenarios. [16]. Similarly, Chen et al. [15] suggested a dual-layer variable sliding window method that can identify multiple kinds of drift in a frequent pattern mining context.

Adaptive windowing (ADWIN), a key invention for learning from evolving data streams, was first presented in seminal work by Bifet and Gavalda [10]. In a related effort, Wang et al. [2] proposed how dynamic resampling techniques increase learner robustness in imbalanced environments by proposing a framework that integrates concept drift detection with online class imbalance learning.

2.2. Class Imbalance in Streaming Data

Another major challenge in streaming environments, where minority classes can reflect crucial outcomes such as cyberattacks, is imbalanced datasets.

In their comparative analysis of undersampling, oversampling, and SMOTE, Wongvorachan et al. [9] offer insight into their effects in educational data mining, which can be generalized to other domains like fraud detection and security. In order to improve decision-making accuracy in real-time streaming tasks, Priya and Uthra [3] developed an ensemble framework based on deep learning that uses recurrent architectures to jointly tackle concept drift and class imbalance.

A similar hybrid ensemble strategy was implemented by S P. and R A. U. [1], who projected an ensemble concept drift detector for imbalanced streams, highlighting multi-learner collaboration for improved adaptability.

2.3. Intrusion Detection Systems and Adaptive Learning

Several studies in the cybersecurity domain have incorporated these techniques into the design of Intrusion Detection Systems (IDS). By employing a variable-length particle swarm optimization technique, Noori et al. [4] proposed a feature drift-aware intrusion detection system that demonstrated significant gains in attack detection accuracy. A genetic programming-based incremental learning method tailored to address concept drift and class imbalance in intrusion detection for streaming data was presented by Shyaa et al. [12].

In order to solve the scalability and latency issues associated with real-time data, Atbib et al. [11] designed a distributed intrusion detection system for Internet of Things scenarios. In order to ensure more reliable threat detection, Saeed [14] presented a hybrid IDS model that can adjust to evolving data distributions in a streaming setup.

Earlier foundational analyses by Lippmann et al. [6] and Axelsson [5] explored the limitations of benchmark datasets such as DARPA and the base-rate fallacy, respectively. Later, the KDD CUP 99 dataset-which remains widely used for IDS benchmarking-was thoroughly examined by Tavallaee et al. [19].

By generating synthetic samples of the minority class, Chawla et al. [21] proposed the Synthetic Minority Over-sampling Technique (SMOTE), a data-level approach for dealing with unbalanced datasets. Their study showed that combining SMOTE with majority class under-sampling enhanced classification performance. SMOTE remains a fundamental technique for improving the detection of uncommon but critical anomalies in the context of Intrusion Detection Systems (IDS), where malicious data is typically underrepresented.

A thorough taxonomy of performance-aware concept drift detectors was provided by Bayram et al. [22], who also demonstrated how model degradation can be utilized as a proxy to detect significant data changes. The techniques that track prediction performance in non-stationary environments were the focus of their survey. These techniques are very relevant to real-time intrusion detection systems, where it is crucial to maintain constant accuracy in the face of evolving threats.

The connection between Concept Drift, Feature Dynamics, and IDS was examined by Shyaa et al. [23]. They propose an integrated model that incorporates adaptive learners, dynamic feature selection, and continuous monitoring to address the lack of integration between drift-aware algorithms and IDS frameworks. By addressing concept and feature drift in the cybersecurity domain, this work fills a significant research gap.

Xiang et al. [24] concentrated on deep learning-based strategies for idea drift adaptation. They categorized the use of drift adaptation strategies in time-series and non-stationary environments, classifying them into discriminative, generative, and hybrid learning methods. The review provides valuable information on how to create robust IDS with deep models capable of handling evolving data distributions.

The Internet of Intelligent Things (IoIT) was introduced by Oliveira et al. [23], who combined edge computing, embedded systems, and TinyML. The growing importance of using lightweight, flexible machine learning models on devices with limited resources is highlighted by their survey. The discussion of on-device learning in dynamic environments is pertinent to drift-aware intrusion detection in IoT networks, while not exclusively focused on IDS.

3. Proposed Modelling

3.1. Dataset Description

The NSL-KDD dataset is an enhanced version of the original KDD Cup 1999 dataset, which was extensively used for assessing intrusion Detection Systems (IDS).

A number of problems with the original KDD dataset, such as duplicate records and unbalanced classes, affected the performance evaluation of machine learning algorithms.

Figures 2 and 3 visualize the distribution of attacks in the NSL-KDD dataset. It helps to understand how many samples belong to each attack type**,** highlight class imbalance (some attacks are very common, others rare), and help decide whether to do class grouping (e.g., DoS, R2L, etc).

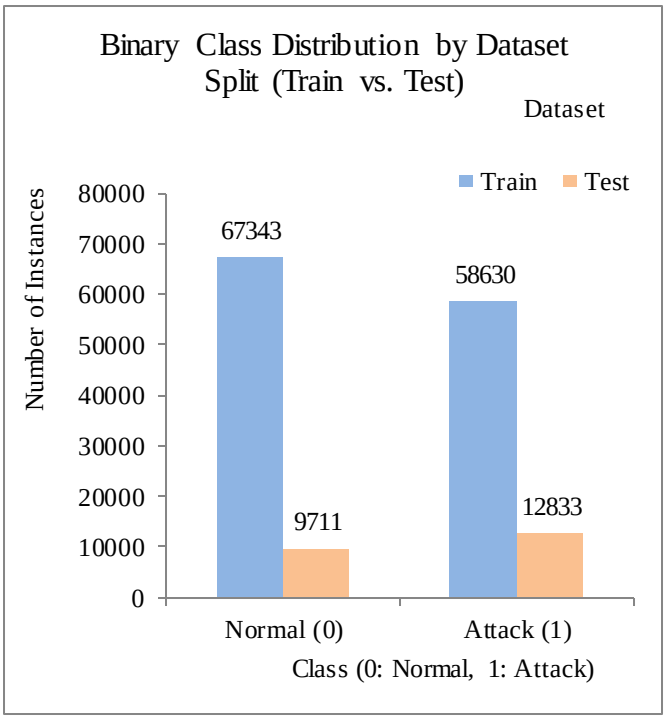

Fig. 1 Training and Test dataset distribution

Table 1. Attack categories

Categories	Types of Attacks
DoS	neptune, smurf, back, teardrop, etc.
Probe	satan, portsweep, nmap, ipsweep
R2L	guess_passwd, ftp_write, imap, warezclient
U2R	buffer_overflow, loadmodule, perl, rootkit

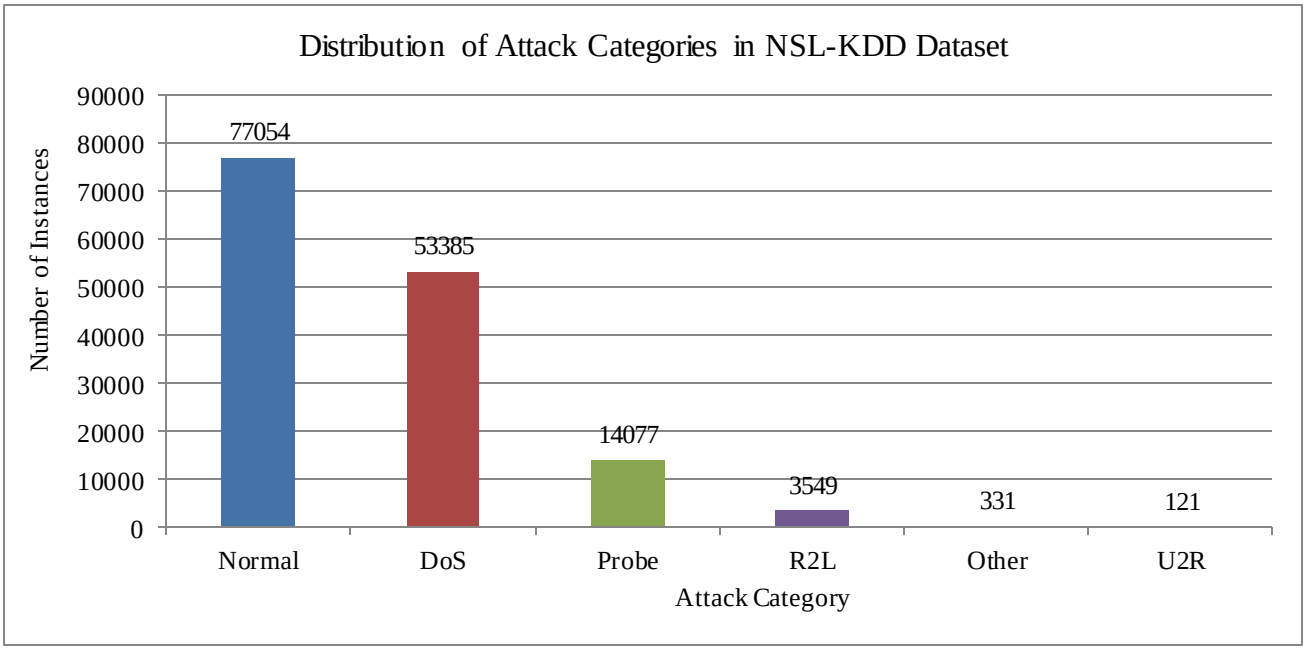

Fig. 2 Distribution of attacks

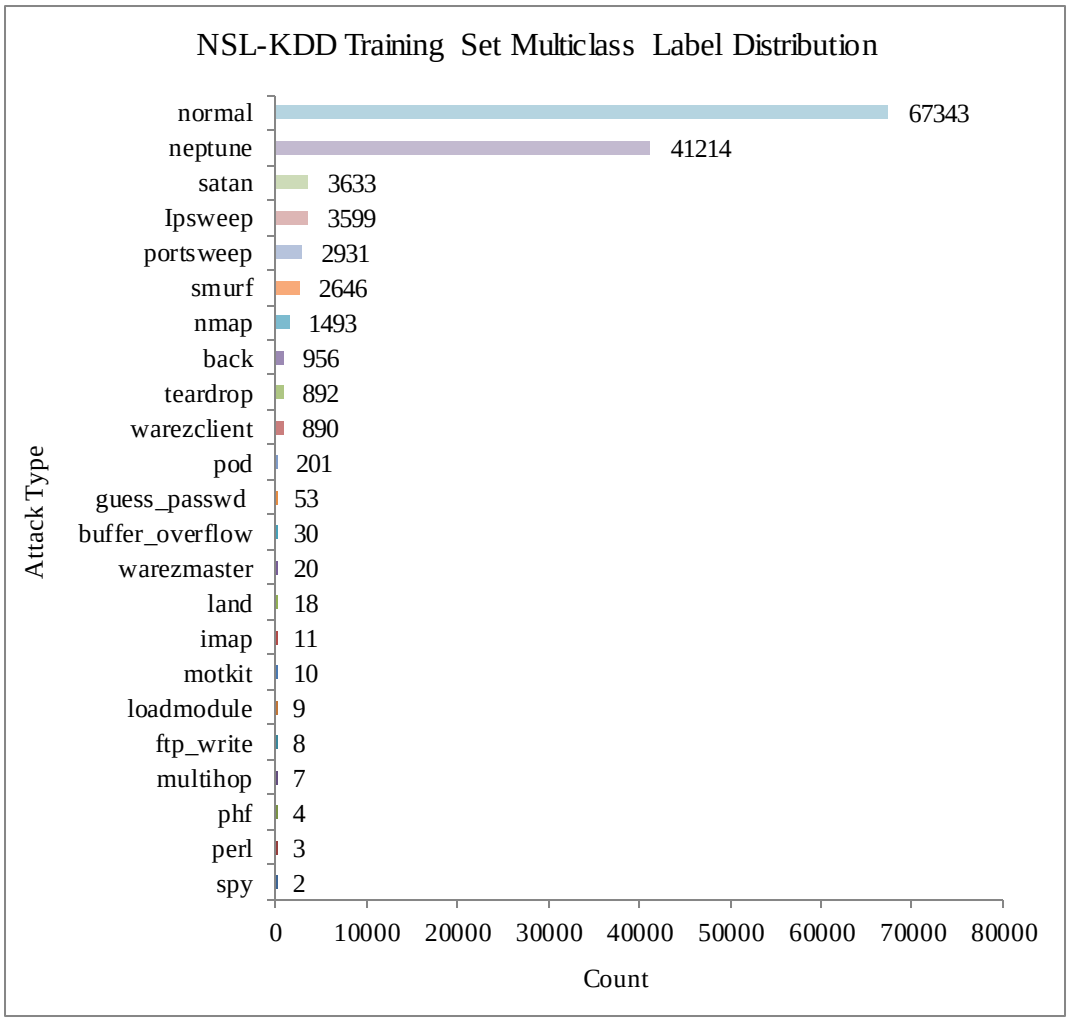

Fig. 3 Distribution of attacks (%)

Based on the normal or attack categories, we first decoded the target label of the NSL KDD dataset to 0 and 1. Attack type is classified as 1 and normal type as 0. The training and testing datasets in the NSL KDD dataset have a combined total of 125973 and 22544 records.

Figure 1 depicts the training and testing dataset distribution; 67343 records belong to the normal category, and 58630 records belong to the attack categories. There are 9711 records in the test dataset that belong to normal categories, and 12833 records belong to attack categories. Table 1 lists the attack categories according to which we have been classified. These categories include Normal, DoS, Probe, R2L, and U2R attacks.

In Figure 4, depicted as an outlier visualization using the combined Train and Test NSL KDD Datasets. It helps to understand and visualizing high-dimensional data in 2 dimensions, detect outliers or unusual patterns in any class, evaluating class separability, and understand data structure before training machine learning models.

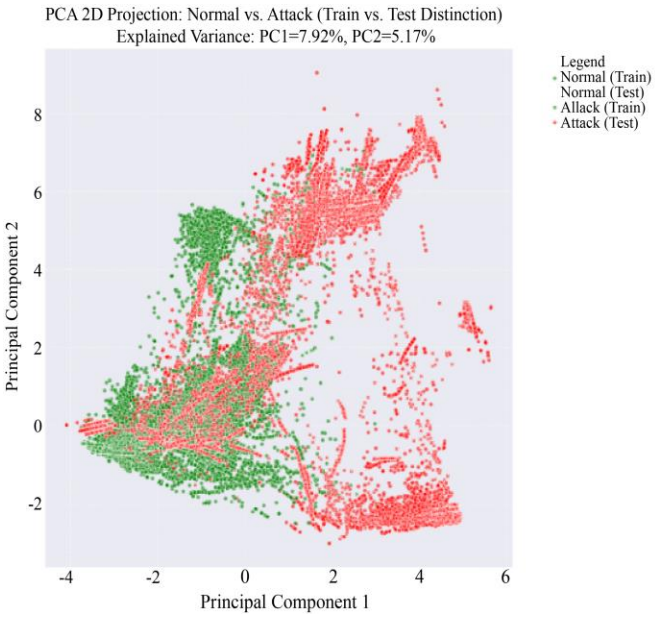

Fig. 4 PCA plot for outlier visualization

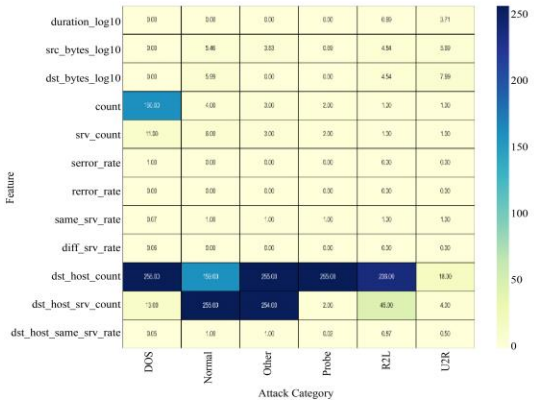

Fig. 5 Heatmap visualization in NSL-KDD

Figure 5 shows a heatmap visualisation using the combined Train and Test NSL KDD Datasets. It helps to visualize how different attack classes differ across selected features, identify discriminative features for classification, spot trends and patterns in network behavior for each attack type, and guide feature selection for machine learning models.

In Figure 6, shown as top features' visualization using combined Train and Test NSL KDD Datasets. It helps to understand and visualize ranks features based on their contribution to model performance. Additionally, it aids in feature selection, model simplification, and dimensionality reduction.

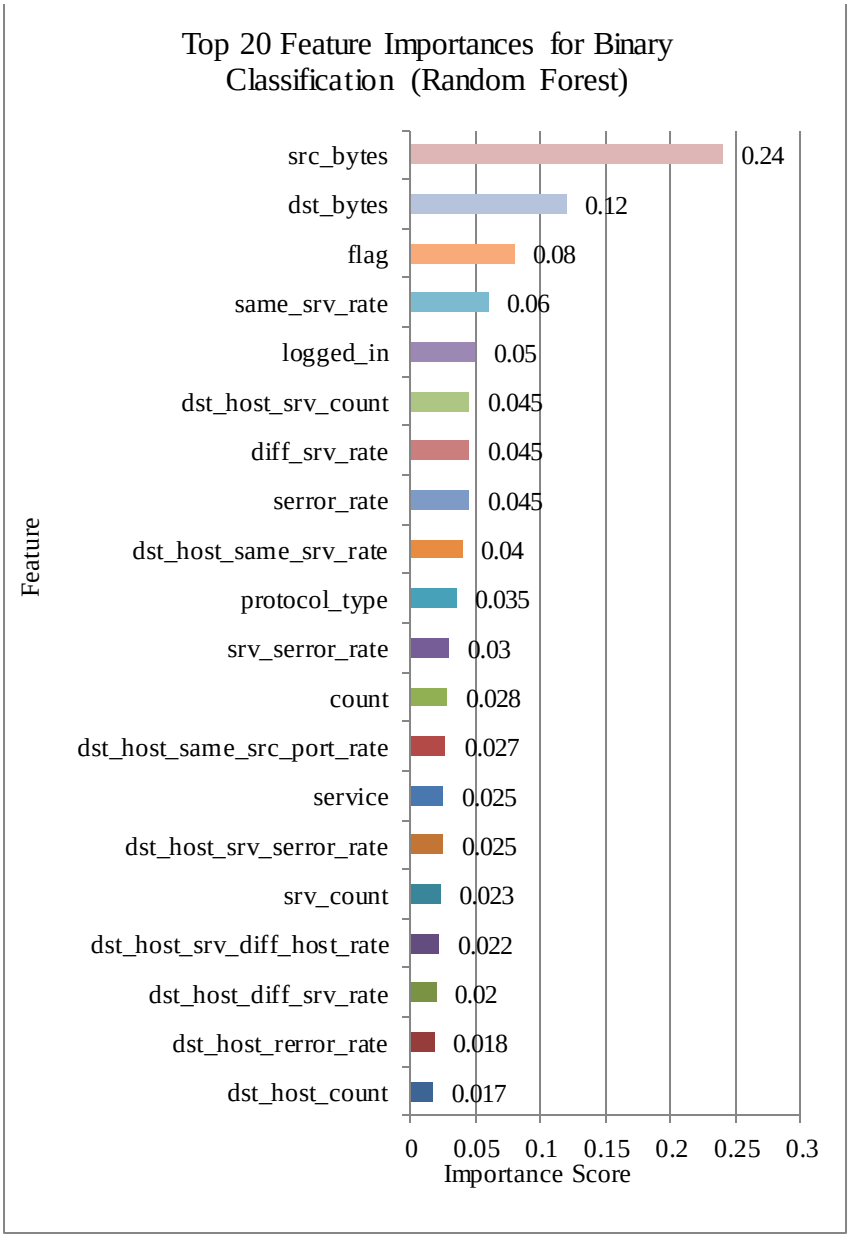

Fig. 7 Top 20 features importance using random forest

Figure 7 shows the distributions of major numerical features (violin plots) by dataset split and binary class. It facilitates comprehension of the distribution of important numerical features in the training and testing subsets of the NSL-KDD dataset across two classes (attack vs. normal). It is a powerful way to comprehend how class-specific features behave and how they are generalized across dataset splits.

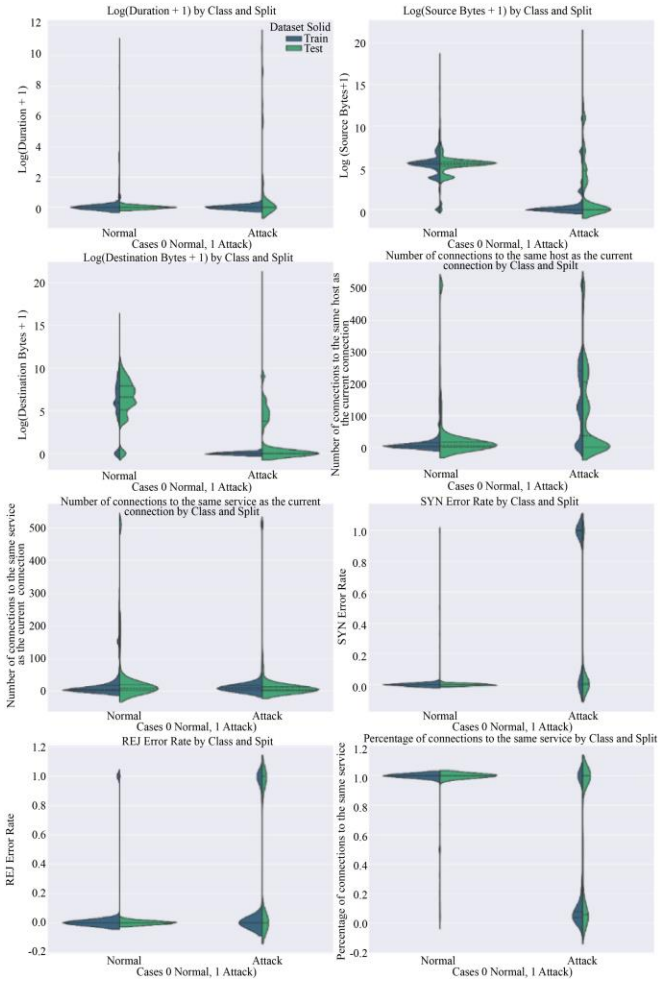

Fig. 7 Key numerical feature distributions

3.2. Conventional Classifiers Analysis

Following exploratory data analytics, conventional machine learning classifiers such as K-Nearest Neighbors (KNN), Logistic Regression (LR), Decision Tree (DT), Gradient Boosting (GBT), and lgb. LGBM (LGBM) and Random Forest (RF) were used to process offline static data.

The results of this analysis, Table 2, show that the accuracy of the training dataset was 99.73%, 87.71%, 99.99%, 99.6%, 99.99% and 99.83%, while the accuracy of the testing dataset was 77.20%, 70.24%, 79.35%, 78.84%, 76.24% and 79.35%. For this NSL KDD test dataset, the LGBM classifier offers a 79.35% higher accuracy than all other models' accuracy.

Table 2. Training and testing accuracy

Model	Train Acc_y (%)	Test Acc_y (%)	Precision	Recall	F1-Score
KNN	99.73	77.20	96.39	62.28	75.67
LR	87.71	70.24	88.09	55.19	67.86
DT	99.99	76.83	96.50	61.52	75.14
GBT	97.96	78.02	96.28	63.86	76.79
RF	99.99	76.77	96.59	61.37	75.05
LGBM	99.98	79.35	96.57	66.07	78.46

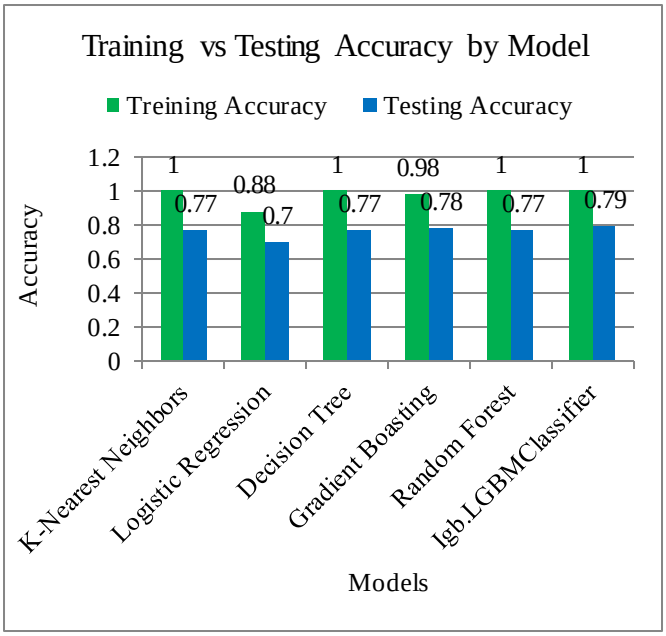

Fig. 8 Training and testing accuracy graph

Figure 8 depicts various models together with performance metrics such as recall, accuracy score, precision, and F1-measures. Because of differences in data distribution or statistical properties between the training and test datasets, all models have reduced accuracy. Therefore, more analysis is required for accuracy improvement.

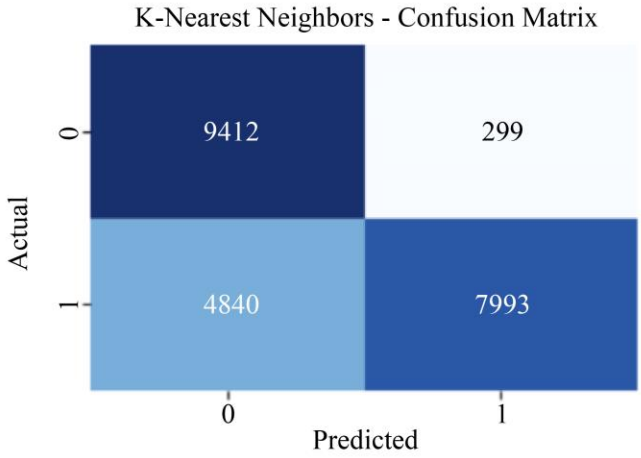

Fig. 9 KNN confusion matrix

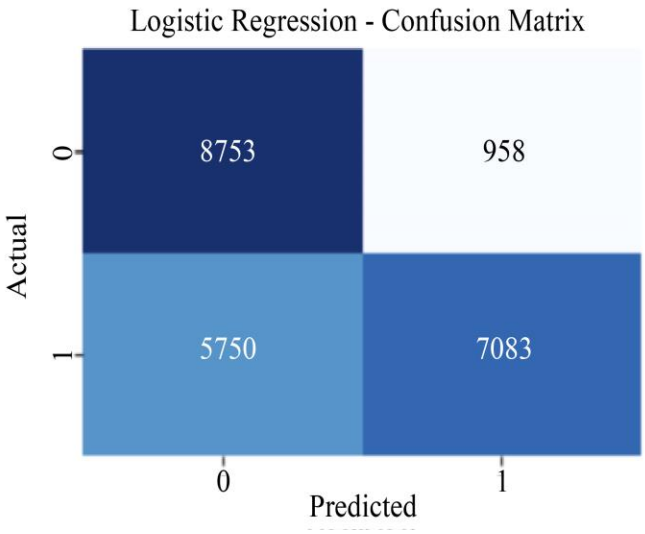

Fig. 10 LR confusion matrix

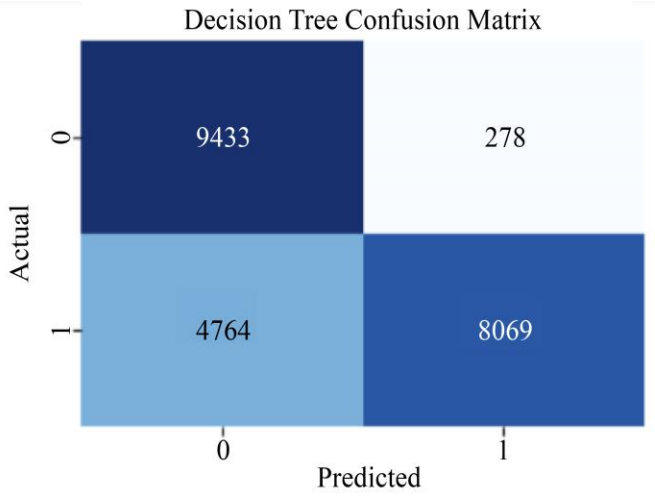

Fig. 11 DT confusion matrix

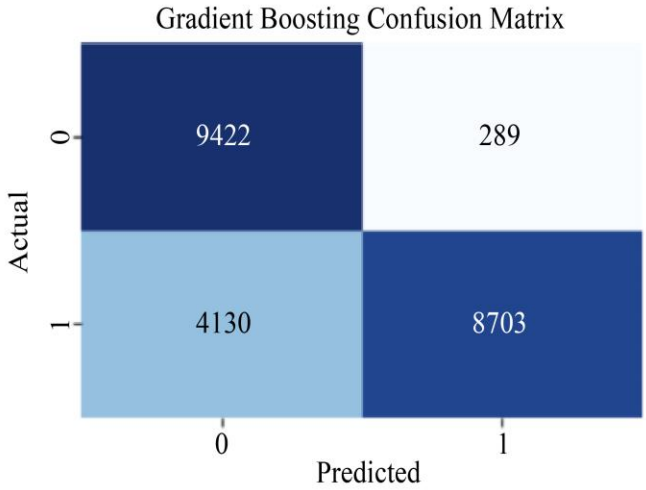

Fig. 12 GBT confusion matrix

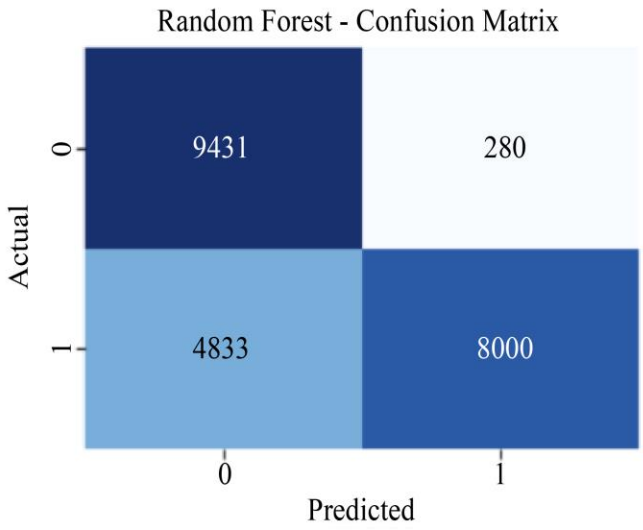

Fig. 13 RF confusion matrix

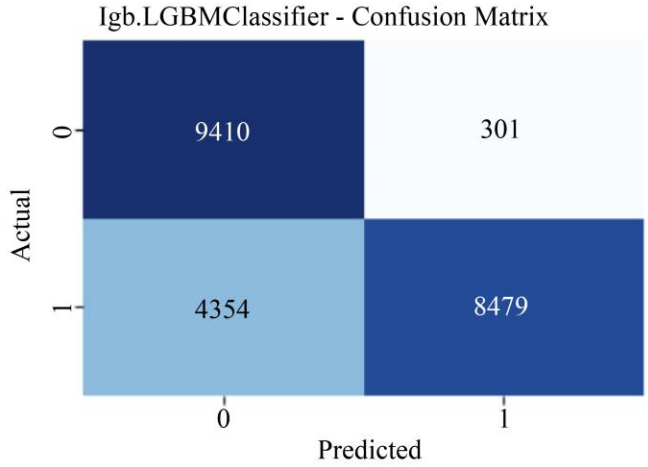

Fig. 14 RF confusion matrix

Figures 9 to 14 show the confusion matrix for various machine learning models, such as KNN, LR, DT, GBT, RF and LGBM.

This matrix compares predicted and true labels to assess how well a classification model performs.

3.3. Data Imbalance with Machine Learning Models

A significant challenge in machine learning, particularly in supervised learning issues, is data imbalance. It occurs when target classes' distributions are uneven, which causes model learning to be biased in favor of the majority class.

The model's performance may suffer as a result of this imbalance, especially when it comes to detecting instances of minority classes, which are frequently the most important in security-related tasks such as intrusion detection.

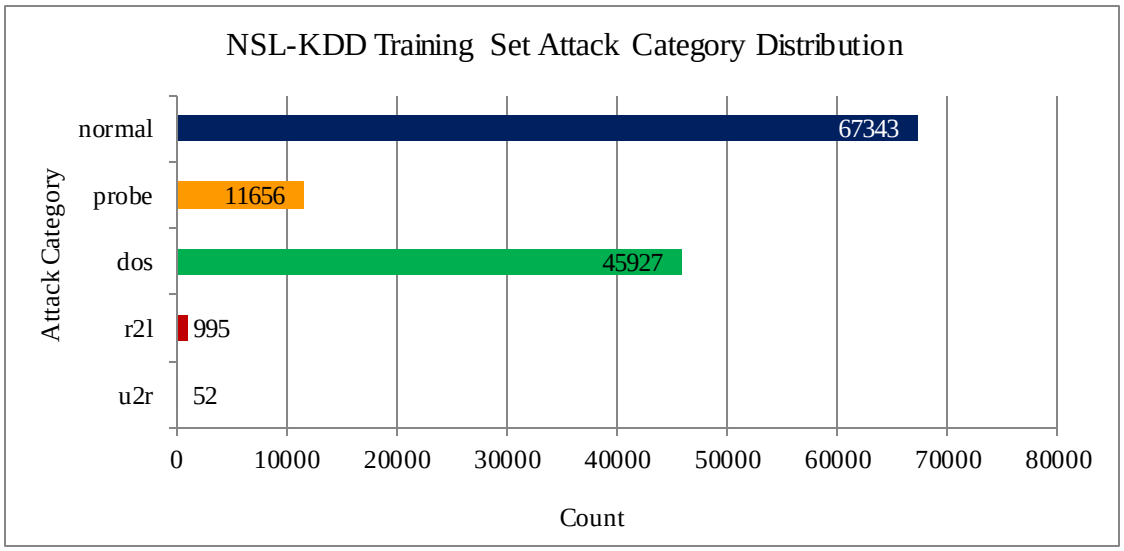

Fig. 15 Training dataset attack category distribution

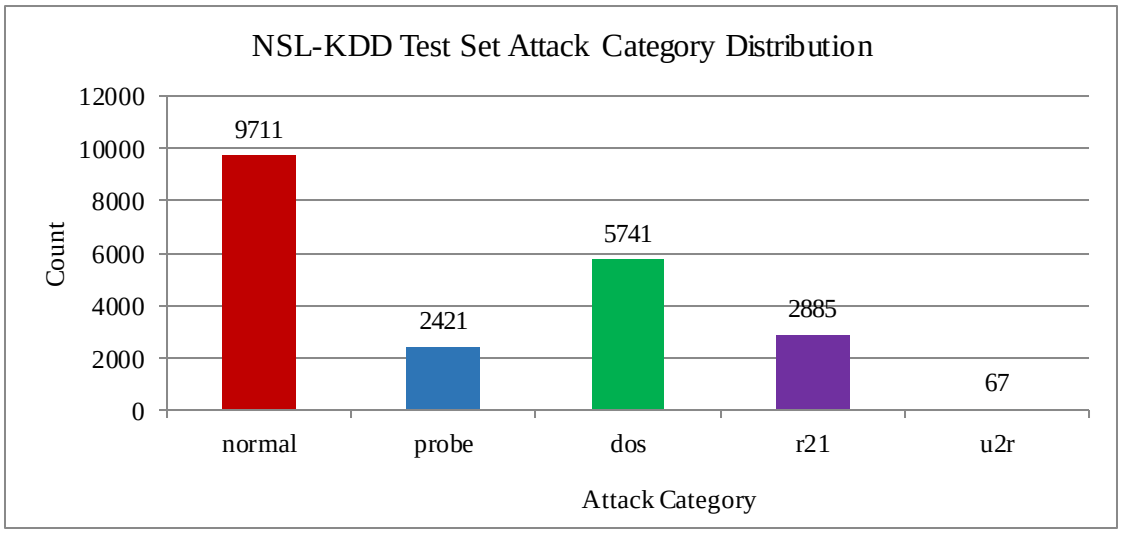

Fig. 16 Training dataset attack category distribution (%)

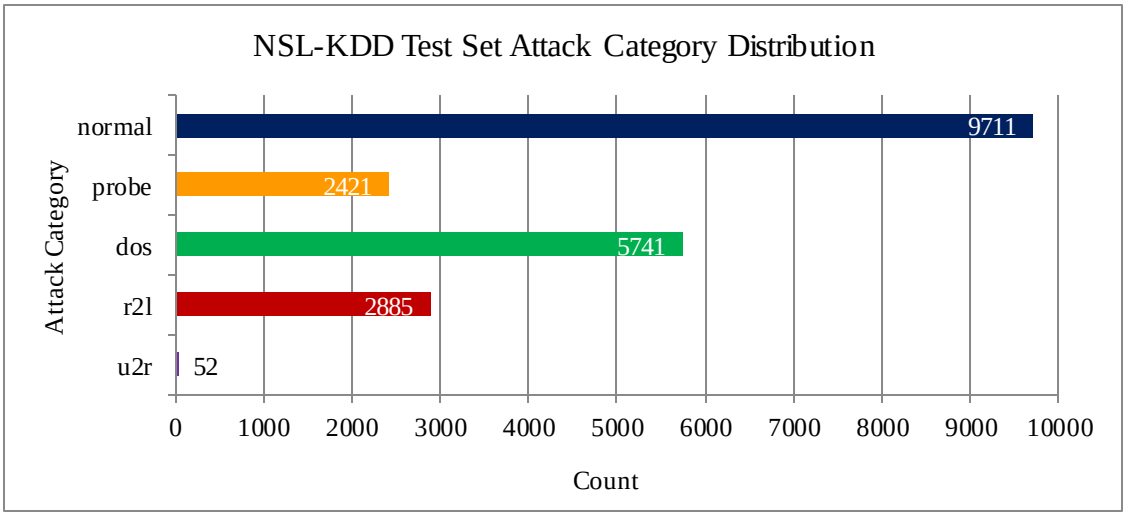

Fig. 17 NSL KDD test dataset attack category distribution

Figures 15 and 16 show the target class distribution for the NL KDD training dataset. Based on this data, 3% of instances belong to the normal category, 36% and 8.5% belong to the DoS and Probe attack categories. However, R2L and U2R only belong to this category at 0.8% and 0.04%, respectively. Therefore, the R2L and U2R categories are considered minority classes, while the regular, DoS, and Probe categories are treated as majority classes.

Figures 17 and 18, which depict the distribution of the NSL KDD Dataset test set, indicate that 53%, 36%, and 8.5% of the instances belong to the Normal, DoS, and Probe attack categories. Instances that belong to the R2L and U2R categories are 0.8% and 0.04%, respectively. Thus, this class's unequal distribution is also present in the test set.

This is the issue that results in NSL KDD and training and test datasets. For this NSL KDD Training dataset, conventional classifiers are therefore unable to provide superior accuracy. In order to address this data imbalance issue for this dataset, we will concentrate on resampling strategies.

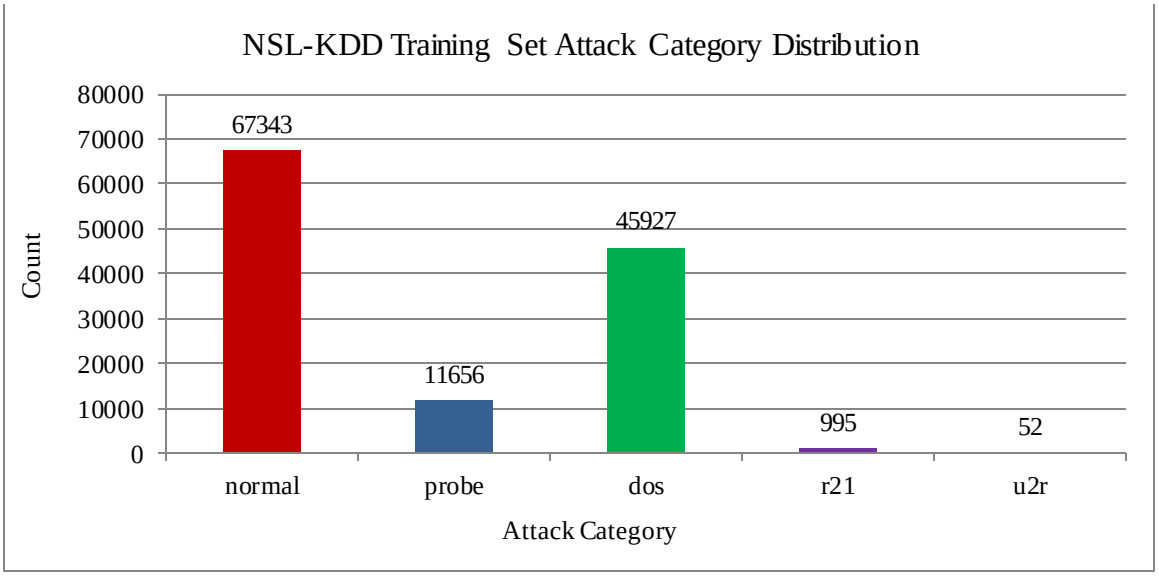

Fig. 18 NSL KDD test dataset attack category distribution (%)

3.3.1. Random Undersampling

Equations (1) to (6) Initially, we applied random undersampling to the NSL KDD dataset. It is a method for addressing class imbalance in datasets that is used in data preprocessing and machine learning. To balance the dataset, samples from the majority class are randomly removed. This helps prevent a model from becoming biased toward the majority class. DoS attacks dominate the attack classes, indicating a major imbalance in the dataset. There are very few samples for R2L and U2R attacks. Therefore, by reducing the number of samples in dominating classes (Normal, DoS, etc.), this method aids in addressing class imbalance.

Steps:

$$D = \{(x_i, y_i)\}_{i=1}^{N} \quad (1)$$

Be the dataset, where:

- $x_i \in \{C_1, C_2, \ldots . C_k\}$ *is the class label*
- N is the total number of samples in the dataset.
- C_j is the j-th class among k classes (e.g., 'Normal', 'DoS', 'R2L', 'U2R', 'Probe').

Step 1: Compute Class Frequencies

For each class C_j, compute the number of instances:

$$n_j = |\{i: y_i = C_j\}| \quad (2)$$

This gives you:

- n_1=number of 'Normal' samples
- n_2=number of 'DoS' samples
- n_3=number of 'Probe' samples
- n_4=number of 'R2L' samples
- n5=n_5 =n5= number of 'U2R' samples

Step 2: Identify the Minority Class

Find the minimum number of instances across all classes:

$$n_{min} = \min_j n_j \quad (3)$$

This value determines the target number of samples per class after undersampling.

Step 3: Apply Random Undersampling

For each class C_j, randomly sample n_{min} instances without replacement:

$$D_j' = RandomSample\left(\{(x_i,x_j) \in D: y_i = C_j\}, n_{min}\right) \quad (4)$$

That is, for every class, they randomly pick n_{min} samples.

Step 4: Combine Resampled Classes

Combine the undersampled class subsets into one balanced dataset:

$$D' = \bigcup_{j=1}^{k} D_j' \quad (5)$$

Now, D′ contains $k \times n_{min}$ samples, with balanced classes.

Step 5: Shuffle the Dataset

Shuffle D′ to mix the classes randomly:

$$D'' = shuffle\,(D') \quad (6)$$

Now D″ is ready for training a machine learning model.

3.3.2. SMOTE Oversampling

Using the NSL-KDD dataset, SMOTE (Synthetic Minority Oversampling Technique) improve classification performance, especially for rare attack classes like R2L and U2R. SMOTE helps by creating synthetic samples for these minority classes instead of just duplicating them.

SMOTE Algorithm

1. Choose the number of synthetic samples to generate.
2. Find the k nearest neighbors for each minority instance.
3. Choose a neighbor at random.
4. To create a synthetic sample, interpolate between the instance and its neighbor.
5. Add the dataset with synthetic samples

Mathematical Steps

$$D = (x_i, y_i)_{i=1}^{N} \quad (7)$$

Where :

- $x_i \in R^d$: $feature\ vector$
- $y_i \in \{C_1, C_2 \ldots\ldots, C_k\}$: $class\ label$

Let $C_m \subset D$ be the minority class with n_m samples.

Synthetically create G new samples for class C_m, to equal the number of majority class samples.

Step 1: Choose the Number of Synthetic Samples

Decide how many synthetic samples you wish to generate:

$$G = n_{target} - n_m$$

Where:

- n_{target} : number of samples you wish the minority class to have after oversampling
- G: total number of synthetic samples to create.

Step 2: For Each Minority Sample

$$x_i \in C_m \quad (8)$$

Find its k nearest neighbors in feature space (typically k=5):

$$NN_k(x_i) = \{x_{i1}, x_{i2}, \ldots\ldots, x_{ik}\} \quad (9)$$

Using Euclidean distance or another metric:

$$dist(x_i, x_j) = || x_i - x_j || \quad (10)$$

Step 3: Randomly Pick Neighbors

For each sample x_i, randomly choose $N \leq k$ neighbors (often N=1).

Step 4: Generate Synthetic Samples

For each selected neighbor x_{ij}, create a synthetic sample x_{new} using interpolation:

$$x_{new} = x_i + \delta \,.\, (x_{ij} - x_i) \quad (11)$$

W here:

- $\delta \in [0,1]$is a random number chosen from a uniform distribution.
- This establishes a point along the line segment between x_i and its neighbor.

Repeat until you have G synthetic samples.

Step 5: Add Synthetic Samples to the Dataset

The synthetic data $D_s = \{(x_{new}, C_m)\}$ is added to the original dataset:

$$D' = D \cup D_s \quad (12)$$

Now the dataset D′ has a more balanced class distribution.

3.3.3. Undersampling and Oversampling Results and Discussion

To resolve the class imbalance and improve model accuracy in the NSL-KDD dataset, we applied resampling techniques, specifically SMOTE (Synthetic Minority Oversampling Technique) and random undersampling. The results show that SMOTE with oversampling attained an accuracy of 81.21%, while random undersampling attained 79.77%.

Table 3. Sampling techniques performance analysis

Metric	Test Acc$_y$ (%)	F1 score	Precision	Recall
Original	0.7873	0.7894	0.9044	0.7003
SMOTE	0.8121	0.8110	0.9487	0.7082
Undersampling	0.7977	0.8019	0.9063	0.7190

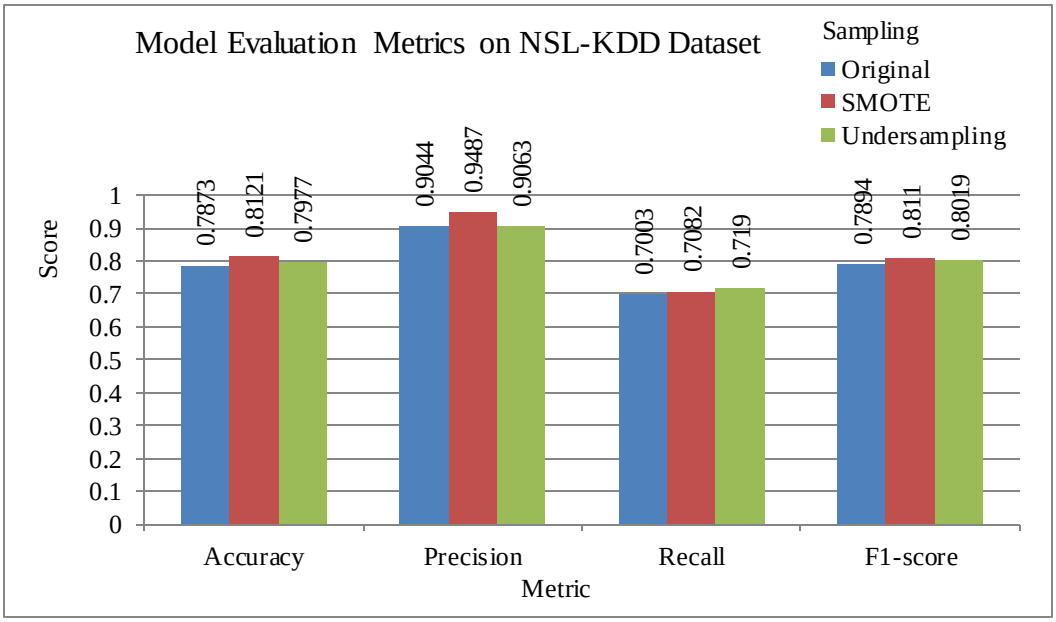

Fig. 19 Sampling techniques with NSL-KDD performance

Although SMOTE provided a slight improvement in accuracy compared to undersampling and conventional classifiers' accuracy, SMOTE did not considerably boost the model's overall performance to an optimal level. Table 3 and Figure 19. displays comprehensive evaluation measures for both approaches, such as accuracy, precision, recall, and F1-score.

3.4. Proposed Work

We have experimented with different models and imbalance techniques so far, but the performance in terms of accuracy of this NSL-KDD dataset has not increased. The statistical characteristics and behavior of the NSL KDD training and test datasets are hence the issue. The samples in the training set have a similar statistical distribution. However, when the statistical distribution of the test set is different from the training set, this kind of statistical distribution change is called concept drift. Therefore, the model that was trained on the training set is unable to identify assaults in the test set.

As a result, we created the Adaptive Drift-aware Windowing Intrusion Detection System with Optimization (ADWSE), a novel framework. It addresses handling unbalanced data and concept drift. In addition, we used random search to apply hyperparameter optimization to choose the optimal parameters for enhancing the framework's performance.

3.4.1. Major Contributions

- To create a concept drift detection and adaptation technique that uses the adaptive sliding windowing method to handle concept drift.
- To use the hyperparameter optimization techniques using random search to produce the optimal hyperparameters for the drift model.
- LightGBM should be used to create a real-time classifier that detects concept drift.

A Proposed ADWSE framework is depicted in Figure 20. To solve the majority and minority difficulties, we first collected data from the concern repository, then performed preprocessing techniques to normalize the data using resampling techniques.

The model's performance was then analyzed using traditional classifiers, and it produced lower accuracy. Hence, we have developed adaptive sliding windowing-based drift detection approaches with hyperparameter optimization, since the reason is concept drift in this dataset.

Lastly, we have classified the attacks and detected the drift using the LightGBM classifier. Finally, employing our proposed ADWSE framework, we achieved an improved accuracy of 98.27% in the test dataset.

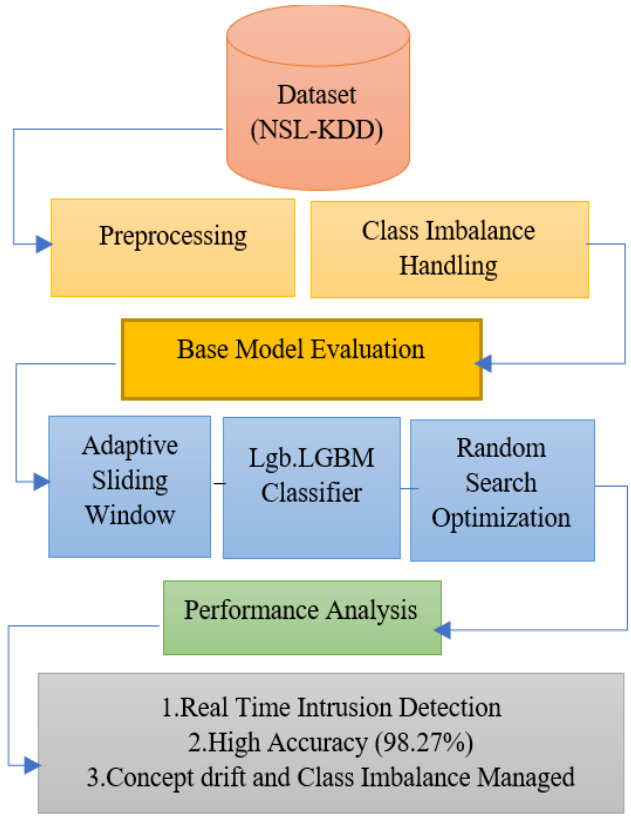

Fig. 20 Proposed ADWISE framework

3.4.2. Adaptive Sliding Window Algorithm

First, we created an adaptive sliding window algorithm that is used for concept drift detection and streamprocessing. It dynamically adjusts the size of the data window based on changes in data characteristics, making it highly useful for real-time applications like intrusion detection

Mathematical Steps

Input:

- NSL KDD Dataset converted as data stream D= {x_1, x_2…., x_t}
- Confidence parameter δ
- window size Wmin
- Dynamically resized window

Output:

- Change detection alerts (if any)

1. Initialization
Set window W= []
Set δ, the confidence threshold (e.g., 0.01)

2. For each new data point x_t:

- Append x_t to the end of the window: $W = W \cup \{x_t\}$

3. While the window can be split into two parts, W_1 and W_2, such that:

$$W = W_1 + W_2 \quad (13)$$

$$|W_1| \geq W_{min}, |W_2| \geq W_{min} \quad (14)$$

Do the following

a. Compute:

$$\mu_1 = mean\ (W_1), \mu_2 = mean\ (W_2) \quad (15)$$

$$n_1 = |W_1|, n_2 = |W_2| \quad (16)$$

b. Calculate threshold ϵ using Hoeffding's bound:

$$\epsilon = \sqrt{\frac{1}{2}.ln\left(\frac{4.log_2(n)}{\delta}\right).\left(\frac{1}{n_1}+\frac{1}{n_2}\right)} \quad (17)$$

Where $n=n_1+n_2$

c. If:

$$|\mu_1 - \mu_2| > \epsilon \quad (18)$$

- Drift Detected
- Remove the oldest portion W1
- Reset window

$$W = W_2$$

4. Repeat for the next data point

3.4.3. Hyper Parameter Optimization using Random Search

Random Search is a simple yet effective strategy for hyperparameter optimization. Random search samples hyperparameters from established distributions and assesses performance for each combination instead of grid search, which searches across a predefined grid exhaustively. Especially in high-dimensional spaces, it often identifies appropriate combinations more quickly and effectively than grid search.

Random Search Algorithm

Step 1: Define the Objective Function

Let f(θ) represent your objective, for example, the validation loss or 1 - accuracy of a machine learning model.

θ ∈ Θ is a vector of hyperparameters,
e.g.: θ= [a, b, win1, win2]

Find the configuration θ^* that minimizes the objective function:

$$\theta^* = arg \min_{\theta\ \varepsilon\ \Theta} f\ (\theta) \quad (19)$$

Step 2: Define the Search Space **Θ**
Specify the range of each hyperparameter you want to optimize

Mathematically:

$$for\ i = 1\ to\ N: \theta^{(i)} \sim \mathrm{P}(\theta), f^{(i)} = \mathrm{f}(\theta^{(i)}) \qquad (20)$$

Example:

Table 4. Search space

Parameter	Range	Type
a	[0.95, 0.99]	Continuous
b	[0.90, 0.98]	Continuous
Win1	[200, 1000]	Integer
Win2	[1000, 5000]	Integer

Step 3: Choose a Sampling Distribution

- For continuous variables → use a uniform distribution over the range.
- For integers → use random integers within the range.

So:

- A ~ u (0.95,0.99)
- B ~ u (0.90,0.98)
- win1~UniformInt (200,1000)
- win2~UniformInt (1000,5000)

Step 4: Random Sampling and Evaluation

Perform the following steps for N trials (e.g., 20 or 50):

1. Randomly sample one combination of parameters $\theta^{(i)} \sim P(\theta)$
2. Evaluate the model using $f(\theta^{(i)})$
3. Record the result and compare with the previous best

Step 5: Select the Best Hyperparameters

After N iterations, select the best-performing configuration:

$$\theta^{*} = \theta^{(i^{*})}\ whe \downarrow * = \ arg \min_{i} f^{(i)} \qquad (21)$$

Step 6: Use the Optimal Configuration

Train your final model using the best hyperparameters θ^{*} found during search.

Table 5. Best parameters

Parameter	Range
a	0.973
b	0.904
Win1	618
Win2	2759

Applying this random search space method, we were able to extract the optimal parameters from the search space that was initially used. These parameters will be used for the Adaptive Sliding Windowing approach; ultimately, the proposed AWSE framework attained an enhanced accuracy of 98.27%.

Figure 21 illustrates that when the test set begins, the current accuracy starts to decline significantly due to the statistical distribution change in the dataset.

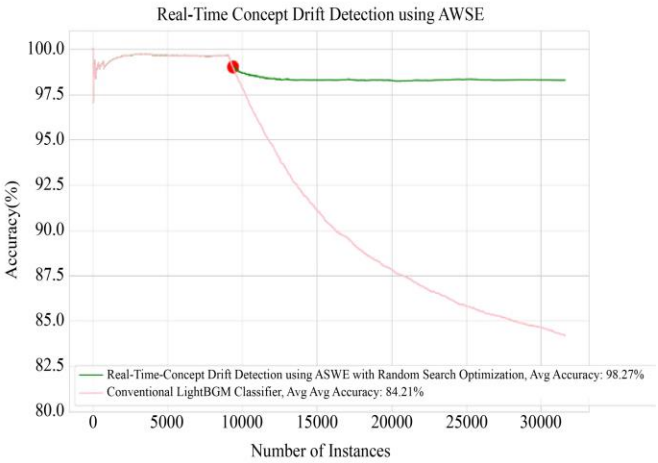

Fig. 21 ADWSE framework-drift detection

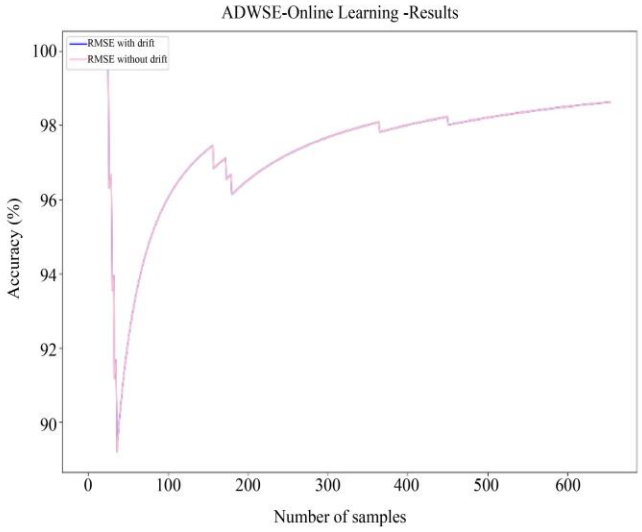

Fig. 22 ADWSE - real-time drift detection and model accuracy

Real-time intrusion detection with accuracy changes is shown in Figure 22. With the use of Python river packages, we first converted the data into a stream and fed it into the ADWSE model, detecting the outcomes as real-time accuracy change detection.

4. Results and Discussions

Initially, we used the NSL KDD Dataset to apply conventional classifiers. Test accuracy was 70.24%, 76.77%, 76.83%, and 77.20%.LR, RF, DT, KNN, GBT, and LightGBM produced 78.02% and 79.35%, respectively.

Utilizing sampling techniques such as SMOTE oversampling and undersampling, the model's accuracy was slightly increased to 79.77% and 81.21%, respectively. Lastly, we used the proposed framework ADWISE, which obtained a superior accuracy of 98.27% for this dataset.

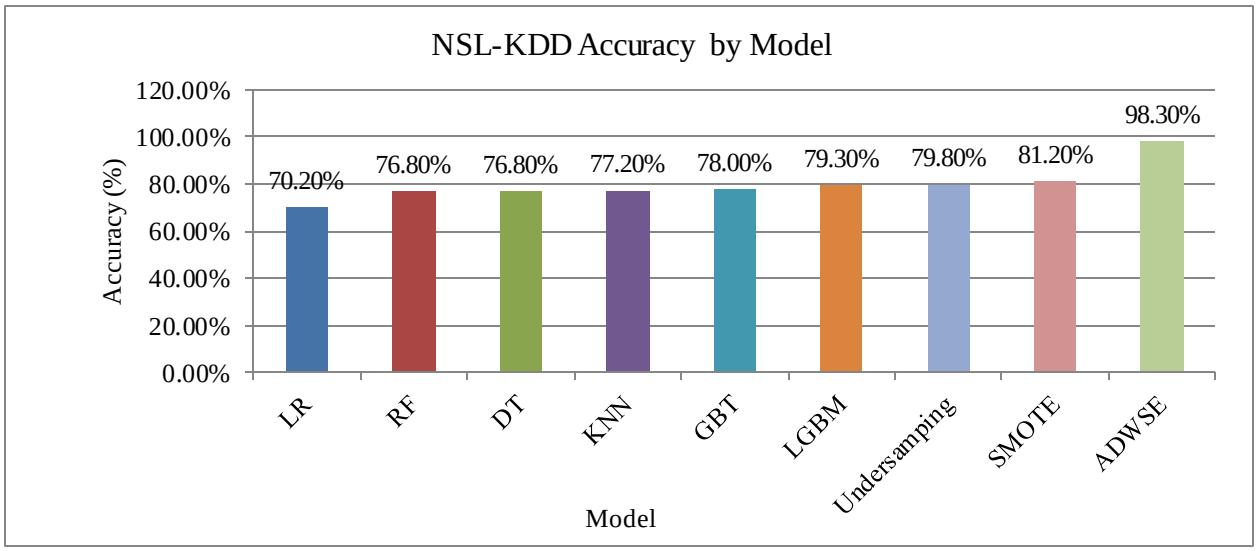

Fig. 23 Performance analysis

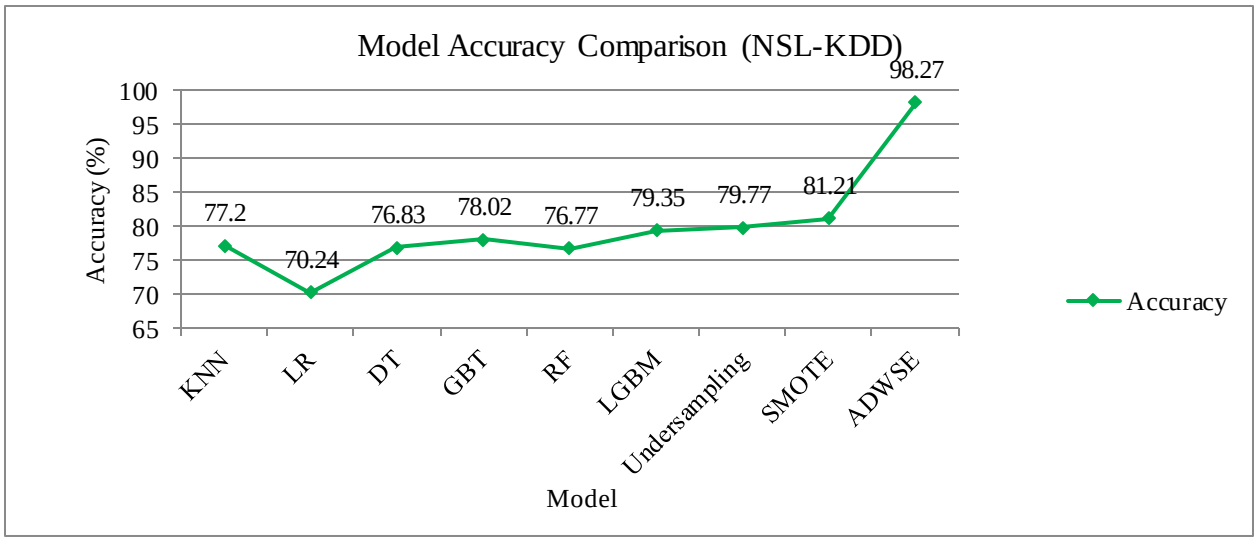

Fig. 24 Model-wise performance trends

Figures 23 and 24 display the accuracy and trends of the model, while Table 6 displays the accuracy (%) of the model's performance. Hence, the above study performed better when using the adaptive sliding windowing method with random search hyperparameter tuning algorithms.

Table 6. Overall model performance (%)

Model	Accuracy (%)
ADWSE	98.27
SMOTE	81.21
Undersampling	79.77
LGBM	79.35
GBT	78.02
KNN	77.20
DT	76.83
RF	76.77
LR	70.24

5. Conclusion

The suggested ADWISE framework, which combines an adaptive sliding window technique with random search hyperparameter tuning, significantly outperformed any other models. This outcome shows that dynamic data segmentation and optimal learning are excellent ways to enhance model performance.

In order to assess ADWISE's usefulness in streaming data conditions, we want to deploy it in real-time systems in subsequent research. In order to validate the framework's generalizability, we also want to evaluate it across a range of domains.

In larger, more complex datasets, scalability and adaptability will be supported by additional research into deep learning integration and computational efficiency optimization.

References

[1] S. Priya, and R. Annie Uthra, "Ensemble Framework for Concept Drift Detection and Class Imbalance in Data Streams," *Multimedia Tools and Applications*, vol. 84, pp. 8823-8837, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[2] Shuo Wang et al., "Concept Drift Detection for Online Class Imbalance Learning," *The 2013 International Joint Conference on Neural Networks (IJCNN)*, Dallas, TX, USA, pp. 1-10, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[3] S. Priya, and R. Annie Uthra, "Deep Learning Framework for Handling Concept Drift and Class Imbalanced Complex Decision-Making on Streaming Data," *Complex & Intelligent Systems*, vol. 9, pp. 3499-3515, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[4] Mustafa Sabah Noori et al., "Feature Drift Aware for Intrusion Detection System Using Developed Variable Length Particle Swarm Optimization in Data Stream," *IEEE Access*, vol. 11, pp. 128596-128617, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[5] Stefan Axelsson, "The Base-Rate Fallacy and the Difficulty of Intrusion Detection," *ACM Transactions on Information and System Security*, vol. 3, no. 3, pp. 186-205, 2000. [CrossRef] [Google Scholar] [Publisher Link]

[6] R.P. Lippmann et al., "Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation," *Proceedings DARPA Information Survivability Conference and Exposition, DISCEX'00*, Hilton Head, SC, USA, pp. 12-26, 2000. [CrossRef] [Google Scholar] [Publisher Link]

[7] K. Ashok Kumar, "Optimized Bayesian Regularization-Back Propagation Neural Network using Data-Driven Intrusion Detection System in Internet of Things," *IEEE Access*, vol. 13, no. 2, 249-263. [CrossRef] [Publisher Link]

[8] P. García-Teodoro et al., "Anomaly-based Network Intrusion Detection: Techniques, Systems and Challenges," *Computers & Security*, vol. 28, no. 1-2, pp. 18-28, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[9] Tarid Wongvorachan, Surina He, and Okan Bulut, "A Comparison of Undersampling, Oversampling, and SMOTE Methods for Dealing with Imbalanced Classification in Educational Data Mining," *Information*, vol. 14, no. 1, pp. 1-15, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[10] Albert Bifet, and Ricard Gavaldà, "Learning from Time-Changing Data with Adaptive Windowing," *Proceedings of the 2007 SIAM International Conference on Data Mining*, pp. 443-448, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[11] Souad Atbib, Chaimae Saadi, and Habiba Chaoui, "Design of A Distributed Intrusion Detection System for Streaming Data in IoT Environments," *2023 9th International Conference on Optimization and Applications (ICOA)*, AbuDhabi, United Arab Emirates, pp. 1-6, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[12] Methaq A. Shyaa et al., "Enhanced Intrusion Detection with Data Stream Classification and Concept Drift Guided by the Incremental Learning Genetic Programming Combiner," *Sensors*, vol. 23, no. 7, pp. 1-34, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[13] Mingyuan Zang, and Ying Yan, "Machine Learning-Based Intrusion Detection System for Big Data Analytics in VANET," *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, Helsinki, Finland, pp. 1-5, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[14] Mozamel M. Saeed, "A Real-Time Adaptive Network Intrusion Detection for Streaming Data: A Hybrid Approach," *Neural Computing and Applications*, vol. 34, pp. 6227-6240, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[15] Jing Chen et al., "Multi-type Concept Drift Detection under a Dual-Layer Variable Sliding Window in Frequent Pattern Mining with Cloud Computing," *Journal of Cloud Computing*, vol. 13, pp. 1-19, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[16] Haolan Zhang et al., "Multilayer Concept Drift Detection Method Based on Model Explainability," *IEEE Access*, vol. 12, pp. 190791-190808, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[17] João Gama et al., "A Survey on Concept Drift Adaptation," *ACM Computing Surveys*, vol. 46, no. 4, pp. 1-37, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[18] Guolin Ke et al., "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Long Beach California, USA, pp. 3149-3157, 2017. [Google Scholar] [Publisher Link]

[19] Mahbod Tavallaee et al., "A Detailed Analysis of the KDD CUP 99 Data Set," *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, ON, Canada, pp. 1-6, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[20] Bobak Shahriari et al., "Taking the Human Out of the Loop: A Review of Bayesian Optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148-175, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[21] Nitesh V. Chawla et al., "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321-357, 2002. [CrossRef] [Google Scholar] [Publisher Link]

[22] Firas Bayram, Bestoun S. Ahmed, and Andreas Kassler, "From Concept Drift to Model Degradation: An Overview on Performance-Aware Drift Detectors," *Knowledge-Based Systems*, vol. 245, pp. 1-19, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[23] Franklin Oliveira et al., "Internet of Intelligent Things: A Convergence of Embedded Systems, Edge Computing and Machine Learning," *Internet of Things*, vol. 26, pp. 1-20, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[24] Methaq A. Shyaa et al., "Evolving Cybersecurity Frontiers: A Comprehensive Survey on Concept Drift and Feature Dynamics Aware Machine and Deep Learning in Intrusion Detection Systems," *Engineering Applications of Artificial Intelligence*, vol. 137, pp. 1-34, 2024. [CrossRef] [Google Scholar] [Publisher Link]