Smart Intrusion Detection: Continuous Learning and Multi-Model Collaboration

Mr. P Siva Srinivasa Rao
Cyber Security
Chalapathi
thrishiva123@gmail.com

Thimmareddygari Anjana Devi
Cyber Security
Chalapathi
anjanaroyal54@gmail.com

Pavan kalyan chekuri
Cyber Security
Chalapathi
pavankalyanchekuri18@gmail.com

Desamala Yaswanth
Cyber Security
Chalapathi
yaswanthluckyyaswanthlucky@gmail.com

Shaik Dulfiqaar
Cyber Security
Chalapathi
dulfiqaars@gmail.com

***Abstract*— With the rapid growth of digitalization and the increasing volume of data, the cybersecurity threat landscape is expanding at an alarming rate. Intrusion Detection Systems (IDS) have become crucial in conjunction with firewalls to safeguard networks from malicious activities. In this work, four well-known cybersecurity datasets—CIC IDS 2017, NSL KDD, KDD Cup, and CIC IDS 2018—are employed to evaluate the effectiveness of various techniques for intrusion detection. Feature selection is performed using Mutual Information to enhance the relevance of selected features. Data sampling techniques are also explored, including Original Data, Random Under Sampling, Random Over Sampling, and a combination of both under and over-sampling to address data imbalance. To further improve the detection performance, a refined approach utilizing a Stacking Classifier combining Random Forest (RF) and Decision Tree (DT) with a Bagging Classifier is implemented. The results show that this approach achieves high performance across all datasets and sampling techniques, demonstrating its effectiveness in accurately detecting network intrusions in dynamic cybersecurity environments.**

Keywords—* Incremental learning, network intrusion detection, machine learning, majority voting classifier, random sampling, Stacking classifier, Cyber Security.

I. Introduction

Intrusion detection plays a pivotal role in the current cybersecurity landscape as the number of evolving attacks, such as Distributed Denial of Service (DDoS), ransomware, and advanced persistent threats (APTs), continues to grow on a daily basis [1], [2], [3]. These attacks have become more sophisticated, causing significant damage to organizations' digital infrastructure, financial systems, and sensitive data. As the cyber threat landscape evolves, traditional security mechanisms often struggle to cope with these new and adaptive threats. Therefore, security systems are in dire need of robust components that can effectively prevent potential attacks and safeguard network integrity [4]. Intrusion Detection Systems (IDS) have emerged as indispensable tools for protecting systems against unauthorized access and mitigating the risks associated with malicious activities, such as unauthorized network access and data exfiltration [5]. IDS are critical in ensuring that systems remain protected, continuously monitoring network traffic and identifying anomalies that may indicate an ongoing attack.

To address these emerging threats, the field of intrusion detection has seen numerous studies employing both machine learning and deep learning approaches [6], [7], [8]. These studies have demonstrated favorable results in detecting a wide range of attack types, from common malware and phishing attacks to more sophisticated threats like zero-day exploits and APTs. Machine learning models, such as decision trees, support vector machines, and neural networks, have shown promise in identifying malicious activity by analyzing network traffic, user behaviors, and other system logs. Additionally, deep learning models, particularly those based on neural networks, have been used to extract high-level features from large-scale datasets and improve the classification of malicious activity. While these approaches offer enhanced detection capabilities, they also face significant challenges. Firstly, most existing IDS systems are static, meaning they lack the ability to adapt and learn in real-time. This hinders their ability to detect new, previously unseen attack vectors, and necessitates expensive and time-consuming retraining processes to keep up with the evolving threat landscape.

Another critical issue with current IDS is their reliance on large, labeled datasets for training, which can be both labor-intensive and costly to obtain. Additionally, these datasets often require significant storage space, creating logistical challenges for organizations with limited resources. The need for such vast datasets can delay the deployment of effective intrusion detection systems, especially in environments with rapidly changing data distributions and attack patterns. These challenges emphasize the need for dynamic, resource-efficient intrusion detection approaches capable of handling emerging threats without requiring extensive labeling efforts or large-scale data storage. In this context, incremental learning approaches have gained significant attention [10], [11] due to their ability to continuously learn and adapt without the need for a complete, ready-to-use dataset [12]. Incremental learning methods enable models to be initially trained with a small amount of data and updated as new data becomes available, offering a more flexible solution to the challenges of traditional IDS systems. This adaptability makes incremental learning particularly suitable for dynamic environments where both the nature of the data and the threats are continuously evolving.

II. Related Work

A significant body of research has focused on advancing intrusion detection systems (IDS) to address the increasing complexity and volume of cybersecurity threats. M. Data and M. Aritsugi [13] proposed AB-HT, an ensemble incremental learning algorithm for IDS that dynamically updates detection models without retraining from scratch. AB-HT continuously learns from incoming data, making it adaptable to evolving attack patterns, while integrating ensemble techniques to enhance detection accuracy and reduce computational and storage costs.

B. A. Tama and S. Lim [14] conducted a systematic mapping study to explore ensemble learning in IDS. They demonstrated that combining multiple classifiers improves detection accuracy and robustness. Evaluation of various ensemble methods across benchmark datasets showed that models such as Random Forest and Adaboost reduce bias and variance, enhancing generalization and resilience against overfitting—a common issue in cybersecurity due to the variability of attack behaviors.

M. Torabi et al. [15] reviewed feature selection and ensemble techniques for IDS, emphasizing the importance of identifying relevant features to optimize machine learning performance. Methods such as Information Gain and Chi-square, when paired with ensemble algorithms like Random Forest and Bagging, improve detection rates while reducing dimensionality. This not only enhances computational efficiency but also strengthens robustness by focusing on the most influential attributes in attack data.

A. M. Bamhdi, I. Abrar, and F. Masoodi [16] introduced an ensemble approach using majority voting. In their framework, multiple classifiers, including K-Nearest Neighbors (KNN), Decision Trees, and Support Vector Machines (SVM), contribute to a final decision based on majority voting. This method leverages the strengths of individual classifiers, mitigating limitations such as high false positive rates and improving overall detection accuracy.

D. R. Patil and T. M. Pattewar [17] proposed a hybrid IDS integrating feature selection with majority voting. By selecting a minimal yet highly informative set of features to train various classifiers, their method reduces data and model complexity while maintaining high detection performance. It enhances scalability and real-time adaptability, allowing IDS to respond to new attack types without extensive retraining.

H. Xu and Y. Wang [18] developed a continual few-shot learning IDS using meta-learning techniques. This approach addresses scenarios with scarce or constantly evolving labeled data, enabling the detection of novel attacks from a small number of examples. Meta-learning improves generalization to unseen attack types, making it effective in real-time intrusion detection under data scarcity.

T. Wang et al. [19] introduced a few-shot class-incremental learning method, emphasizing incremental adaptation from small labeled datasets. This approach allows the IDS to learn continuously as new attack data becomes available, improving flexibility and scalability for emerging threats. It ensures that detection models remain current with evolving cybersecurity landscapes.

J. Zheng et al. [20] proposed a two-level ensemble learning IDS, integrating multiple classifiers in a hierarchical structure. Base classifiers first make independent predictions, which are then combined by a meta-classifier to improve final decision-making. This layered approach increases robustness and detection accuracy by incorporating classifier diversity and reducing the impact of misclassifications at the base level.

In summary, these studies collectively advance IDS by highlighting the significance of incremental learning, ensemble techniques, feature selection, and few-shot learning. AB-HT and incremental approaches [13,19] ensure adaptability to evolving threats, while ensemble methods [14,16,20] enhance detection accuracy and robustness. Feature selection [15,17] optimizes model efficiency and focus, and meta-learning approaches [18] address data scarcity challenges. The integration of these techniques offers a promising direction for developing IDS that are accurate, efficient, and resilient in dynamic cybersecurity environments.

III. Materials And Methods

The proposed system aims to enhance intrusion detection capabilities by evaluating various machine learning algorithms on well-known cybersecurity datasets, including CIC IDS 2017 [17], NSL KDD [21], KDD Cup [17], and CIC IDS 2018 [8]. Feature selection [15] is carried out using Mutual Information to ensure the most relevant features are utilized for training. Data sampling techniques, such as Original Data, Random Under Sampling, Random Over Sampling, and a combination of under and over-sampling, are applied to address class imbalance. The system incorporates several machine learning algorithms, including K-Nearest Neighbors (KNN) [6], Softmax Logistic Regression (LR) [7], Random Forest (RF) [8], HAT/Decision Tree (DT), and a Voting Classifier that combines KNN, LR, RF, and DT to boost classification performance. Additionally, a Stacking Classifier combining RF and DT with a Bagging Classifier is used to improve predictive accuracy. This comprehensive approach aims to optimize network intrusion detection across various datasets and sampling techniques, providing a robust solution for cybersecurity.

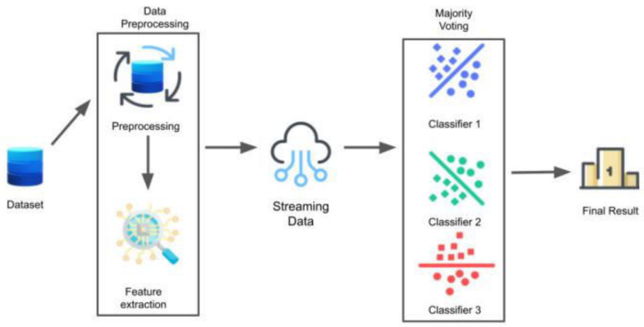

Fig. 1. System Architecture

This system (fig. 1) processes four datasets (CIC IDS 2017, NSL-KDD, KDD Cup, and CIC IDS 2018) through data cleaning and visualization. Label encoding prepares data for feature extraction and selection. Four sampling techniques create training and testing sets. Models (KNN, Softmax LR, RF, DT, Voting Classifier and Stacking Classifier) are trained and evaluated using performance metrics (Accuracy, Precision, Recall, F1-Score).

A) Dataset Collection

a) CIC IDS 2017: The dataset used for this project is the CIC IDS 2017 [17], which contains 2,044,217 entries with 78 features, capturing network traffic data such as packet lengths, flow statistics, and flag counts. After feature selection, the dataset was reduced to 44,697 entries and 20 relevant features. The "Label" column indicates whether a sample corresponds to a specific type of intrusion or attack, making it suitable for intrusion detection system (IDS) analysis.

	Protocol	Flow Duration	Total Fwd Packets	Total Backward Packets	Fwd Packets Length Total	Bwd Packets Length Total	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean
0	6	4	2	0	12	0	6	6	6.00000
1	6	1	2	0	12	0	6	6	6.00000
2	6	3	2	0	12	0	6	6	6.00000
3	6	1	2	0	12	0	6	6	6.00000
4	6	609	7	4	484	414	233	0	69.14286

5 rows × 78 columns

Fig 2 NSL KDD Dataset

b) NSL-KDD: The NSL-KDD [21] dataset, containing 125,972 entries with 43 features, is a benchmark dataset for intrusion detection systems. It includes features such as protocol type, service, flag, source and destination bytes, and various traffic statistics. After feature selection, the dataset is reduced to 26,047 entries with 11 significant features, including "attack" as the target label. It is widely used for evaluating anomaly detection and network intrusion classification methods.

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_same_srv_rate
0	0	udp	other	SF	146	0	0	0	0	0	...	0.00
1	0	tcp	private	S0	0	0	0	0	0	0	...	0.10
2	0	tcp	http	SF	232	8153	0	0	0	0	...	1.00
3	0	tcp	http	SF	199	420	0	0	0	0	...	1.00
4	0	tcp	private	REJ	0	0	0	0	0	0	...	0.07

5 rows × 43 columns

Fig.3 NSL-KDD Dataset

c) KDD Cup: The KDD Cup [17] dataset consists of 125,973 entries with 42 features, used for intrusion detection in networks. It includes attributes like protocol type, service, flag, and traffic details. After feature selection, it is reduced to 26,047 entries with 11 critical features, including "labels" as the target. This dataset is widely employed for evaluating machine learning models in anomaly detection and intrusion prevention systems.

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count
0	0	tcp	ftp_data	SF	491	0	0	0	0	0	...	25
1	0	udp	other	SF	146	0	0	0	0	0	...	1
2	0	tcp	private	S0	0	0	0	0	0	0	...	26
3	0	tcp	http	SF	232	8153	0	0	0	0	...	255
4	0	tcp	http	SF	199	420	0	0	0	0	...	255

5 rows × 42 columns

Fig.4 KDDCUP Dataset

d) CIC IDS 2018: The CIC IDS 2018 dataset [8] contains 3,550,129 entries and 78 features, designed for analyzing network traffic and detecting intrusions. Key features include packet lengths, flow durations, and flag counts. After feature selection, it is reduced to 30,594 entries with 20 essential features, including "Label" as the target. This dataset is widely used in machine learning research for building robust intrusion detection systems, focusing on performance optimization and attack pattern recognition.

	Protocol	Flow Duration	Total Fwd Packets	Total Backward Packets	Fwd Packets Length Total	Bwd Packets Length Total	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean
0	6	141385	9	7	553	3773.0	202	0	61.444443
1	6	281	2	1	38	0.0	38	0	19.000000
2	6	279824	11	15	1086	10527.0	385	0	98.727272
3	6	132	2	0	0	0.0	0	0	0.000000
4	6	274016	9	13	1285	6141.0	517	0	142.777771

5 rows × 78 columns

Fig.5 CICIDS2018 Dataset

B) Data Processing:

In the pre-processing step, we focus on preparing the dataset for modeling. This includes cleaning the data, visualizing key relationships, encoding categorical labels, performing feature extractionandsampling techniques to ensure high-quality input for the prediction model.

a) Data Processing: The data preprocessing involves handling missing and duplicate entries. Initially, the dataset is assessed for null values, which are then removed to ensure the data remains clean and complete. Duplicate entries are identified to eliminate redundancy and prevent bias in analysis. By carefully addressing both issues, the dataset is refined for consistency and accuracy, supporting more reliable and efficient machine learning model training. These steps enhance data quality, ensuring robust outcomes in subsequent analysis and predictions.

b) Data Visualization: The data visualization focuses on analyzing the distribution of the target variable, providing insights into the balance between the classes within the dataset. By presenting the counts of each class, it helps identify any potential class imbalance, which is critical for ensuring fair and effective machine learning model training. Understanding the representation of each target class aids in evaluating the dataset's characteristics, guiding strategies such as resampling or adjusting evaluation metrics to enhance the model's performance on imbalanced data.

c) Label Encoding: Label encoding is applied to transform categorical variables into numerical values, making them suitable for machine learning algorithms. In this process, the target variable, along with other categorical features like protocol type, service, and flag, are converted into integer representations. This technique assigns a unique integer to each category, facilitating the model's ability to handle these variables. By encoding categorical data, the dataset is prepared for more efficient processing, improving model compatibility and performance, especially when dealing with algorithms that require numerical inputs.

d) Feature Extraction: Feature extraction involves selecting the relevant input data (X) and the target variable (y) from the dataset. In this case, the features are extracted by removing the 'labels' column from the data, which is the target variable, while the remaining columns are stored as input features (X). The target variable (y) is then set as the 'labels' column, which the model will predict.

e) Feature Selection: Feature selection using mutual information helps identify the most relevant features for the prediction task. In this process [15], the mutual information classifier evaluates the relationship between each feature and the target variable. By using the SelectPercentile method, the top 25% of features with the highest mutual information scores are selected. The selected features are then transformed into a reduced dataset, and the relevant columns are extracted and listed, ensuring that only the most informative features are retained for model training.

f) Sampling the data: Sampling the data involves modifying the dataset to address class imbalances. The original data is examined first to assess the distribution. Random under-sampling reduces the majority class by randomly removing instances, while random over-sampling increases the minority class by duplicating instances. Combining both techniques creates a balanced dataset by adjusting both classes, ensuring that the model receives an equal representation of each class. This approach helps improve the model's ability to learn from both classes without being biased toward the majority class.

C) Training &Testing

The dataset is divided into training and testing sets to evaluate the model's performance. A portion of the data, typically 20%, is reserved for testing, while the remaining 80% is used for training the model. This split ensures that the model learns patterns from the majority of the data while being tested on unseen examples to assess its generalization capability. The random state is fixed to maintain consistency in the data partitioning across different runs of the model.

D) Algorithms

KNN: K-Nearest Neighbors is applied to classify data based on the proximity of feature values to labeled examples. It [6] helps detect patterns and classify instances by majority voting from the nearest neighbors.

$$distance(x, X_i) = \sqrt{\sum_{j=1}^{d} \left(x_j - X_{i_j}\right)^2} \quad (1)$$

Softmax LR: Softmax Logistic Regression is used to handle multiclass classification [7], transforming logits into probability distributions, enabling the model to predict the likelihood of each class.

$$P(y = k \mid x) = \frac{e^{w_k^T x + b_k}}{\sum_{j=1}^{K} e^{w_j^T x + b_j}} \quad (2)$$

Random Forest: A decision tree [8] ensemble method that aggregates multiple trees to improve accuracy, robustness, and handle overfitting. It is used for reliable classification and feature importance evaluation.

$$y = \arg\max_k \sum_{t=1}^{T} 1\left(h_t(x) = k\right) \quad (3)$$

HAT/DecisionTree: Decision Tree or Hierarchical Agglomerative Tree (HAT) is utilized for classification tasks, providing an interpretable, tree-based model that splits data based on feature thresholds.

$$H(X) = -\sum p(x) \log_2 p(x) \quad (4)$$

Voting Classifier: A combination of KNN, LR, RF, and DT, where each model votes and the class with the most votes is selected. This ensemble approach improves classification performance by combining strengths of different models.

$$\hat{y} = argmax_c \left(\sum_{i=1}^{n} II\ (\hat{y}_i = c) \right) \quad (5)$$

Stacking Classifier: A meta-model built from RF and DT with Bagging Classifier, which leverages predictions from base learners to improve accuracy. It allows the model to correct biases and errors made by individual classifiers.

$$\hat{y} = g(Y_{base}) = g\left(f_1(x), f_2(x), \dots, f_m(x)\right) \quad (6)$$

IV. Experimental Results

A) Evaluation Metrics

Accuracy: The accuracy of a test is its ability to differentiate the patient and healthy cases correctly. To estimate the accuracy of a test, we should calculate the proportion of true positive and true negative in all evaluated cases. Mathematically, this can be stated as:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (7)$$

Precision: Precision evaluates the fraction of correctly classified instances or samples among the ones classified as positives. Thus, the formula to calculate the precision is given by:

$$Precision = \frac{\text{True Positive}}{\text{True Positive } + \text{False Positive}} \quad (8)$$

Recall: Recall is a metric in machine learning that measures the ability of a model to identify all relevant instances of a particular class. It is the ratio of correctly predicted positive observations to the total actual positives, providing insights into a model's completeness in capturing instances of a given class.

$$Recall = \frac{TP}{TP + FN} \quad (9)$$

F1-Score: F1 score is a machine learning evaluation metric that measures a model's accuracy. It combines the precision and recall scores of a model. The accuracy metric computes how many times a model made a correct prediction across the entire dataset.

$$F1\ Score = 2 * \frac{Recall \text{ X } Precision}{Recall + Precision} * 100 \quad (10)$$

B) Performance Evaluation Tables

Table.1 Performance Evaluation for Random Under/Over Sampling (Combine) – CICIDS2017

ML Model	Accuracy	Precision	Recall	F1 Score
KNN	0.993	0.993	0.993	0.993
Softmax LR	0.331	0.676	0.331	0.404
Random Forest	0.989	0.989	0.989	0.989
HAT/Decision Tree	0.960	0.966	0.960	0.961
Majority Voting	0.995	0.995	0.995	0.995
Stacking Classifier	0.994	0.994	0.994	0.994

Table.2 Performance Evaluation for Random Under/Over Sampling (Combine) – CICIDS2018

ML Model	Accuracy	Precision	Recall	F1 Score
KNN	0.995	0.995	0.995	0.995
Softmax LR	0.429	0.568	0.429	0.453
Random Forest	0.997	0.997	0.997	0.997
HAT/Decision Tree	0.996	0.996	0.996	0.996

Majority Voting	0.996	0.996	0.996	0.996
Stacking Classifier	0.996	0.996	0.996	0.996

The tables (3.1 to 3.4) presents the performance evaluation of several machine learning models on the KDD Cup dataset. The results highlight that the Stacking Classifier achieved the highest accuracy across all data sampling strategies.

Table.3 Performance Evaluation for Random Under/Over Sampling (Combine) – KDDCUP

ML Model	Accuracy	Precision	Recall	F1 Score
KNN	0.997	0.997	0.997	0.997
Softmax LR	0.386	0.584	0.386	0.463
Random Forest	0.999	0.999	0.999	0.999
HAT/Decision Tree	0.992	0.992	0.992	0.992
Majority Voting	0.998	0.998	0.998	0.998
Stacking Classifier	1.000	1.000	1.000	1.000

Table.4 Performance Evaluation for Random Under/Over Sampling (Combine) – NSL-KDD

ML Model	Accuracy	Precision	Recall	F1 Score
KNN	0.998	0.998	0.998	0.998
Softmax LR	0.403	0.617	0.403	0.485
Random Forest	0.998	0.998	0.998	0.998
HAT/Decision Tree	0.995	0.995	0.995	0.995
Majority Voting	0.997	0.997	0.997	0.997
Stacking Classifier	1.000	1.000	1.000	1.000

C) Comparison Graphs

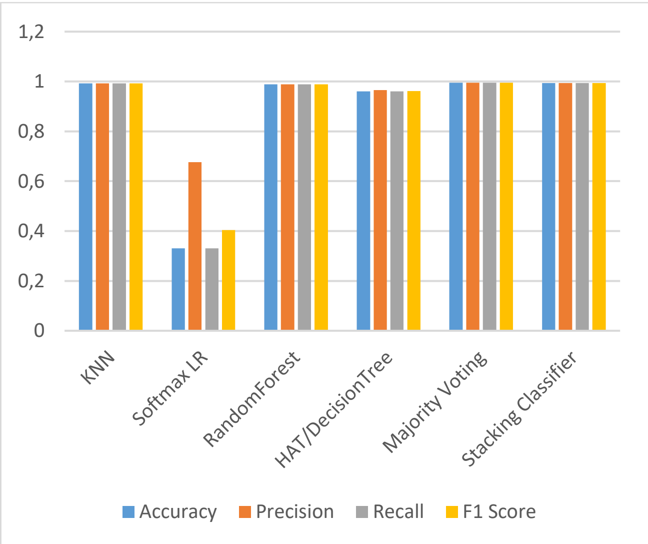

Fig.5 Comparison Graphs for Random Under/Over Sampling (Combine) – CICIDS2017

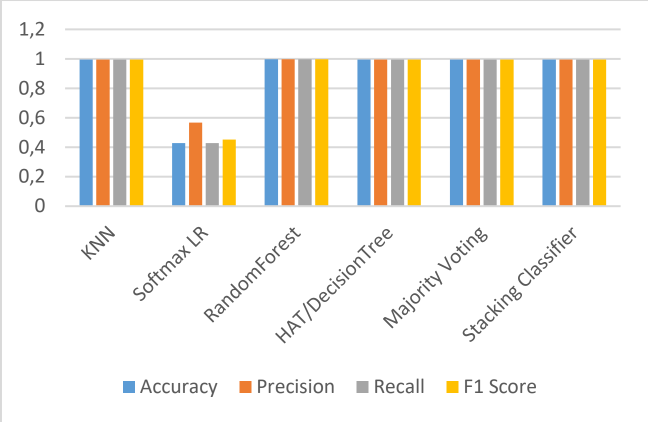

Fig.6 Comparison Graphs for Random Under/Over Sampling (Combine) – CICIDS2018

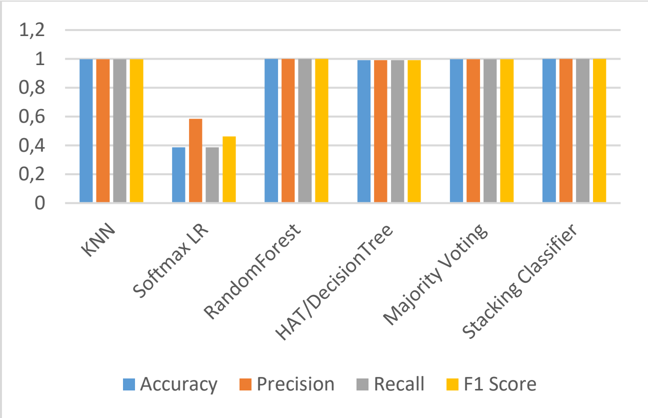

Fig.7 Comparison Graphs for Random Under/Over Sampling (Combine) – KDDCUP

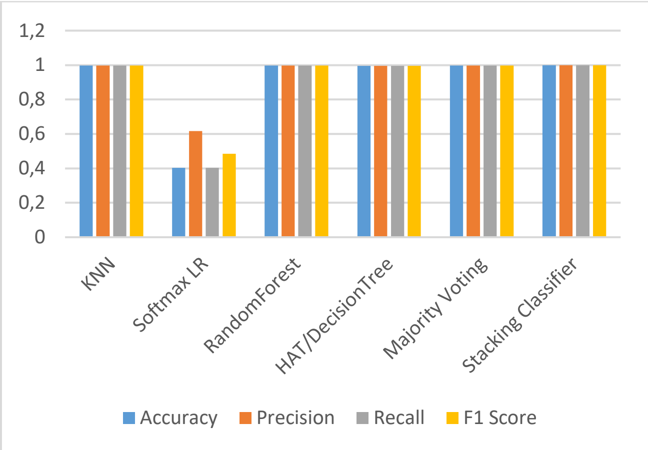

Fig.8 Comparison Graphs for Random Under/Over Sampling (Combine) – NSL-KDD

D) Discussion

The performance evaluation demonstrates that the Stacking Classifier consistently outperforms other machine learning models across all four datasets—CICIDS2017, CICIDS2018, KDDCUP, and NSL-KDD—when using the combined under/over sampling strategy. Metrics such as accuracy, precision, recall, and F1-score indicate that ensemble learning effectively captures complex patterns in imbalanced network traffic, reducing false negatives and improving overall detection rates.

Additionally, comparison graphs highlight the stability of the Stacking Classifier compared to individual models like KNN, Random Forest, and Decision Trees. These results confirm that data balancing combined with ensemble methods significantly enhances intrusion detection performance, making it a robust approach for cybersecurity applications.

V. Conclusion

In conclusion, this study demonstrates the significant potential of advanced machine learning techniques in improving Intrusion Detection Systems (IDS) for network security. By utilizing four widely recognized cybersecurity datasets—CIC IDS 2017, NSL KDD, KDD Cup, and CIC IDS 2018—we have effectively evaluated various approaches for intrusion detection. The application of feature selection via Mutual Information proved to be beneficial in reducing irrelevant features, thereby enhancing model performance. Furthermore, the exploration of data sampling techniques, including Random Under Sampling, Random Over Sampling, and a combination of both, effectively addressed data imbalance issues, ensuring more reliable predictions. The implementation of a Stacking Classifier, which combines the strengths of Random Forest and Decision Tree models with a Bagging Classifier, demonstrated superior performance in accurately detecting intrusions. This approach consistently outperformed other methods across all datasets, showcasing its robustness and adaptability in dynamic cybersecurity environments. The results underscore the importance of integrating multiple strategies, such as feature selection, data sampling, and ensemble learning, to develop effective IDS solutions capable of detecting evolving threats in real-time network traffic.

Future work could focus on enhancing the performance of IDS by incorporating more advanced feature selection methods, exploring deep learning models for anomaly detection, and evaluating additional datasets for broader generalization. Additionally, investigating the use of real-time data streaming and online learning algorithms could improve IDS responsiveness to emerging threats. Furthermore, integrating threat intelligence feeds and developing hybrid systems combining multiple ensemble techniques may lead to even more robust and accurate intrusion detection capabilities, adapting to evolving cybersecurity challenges.

References

[1] M. Mijwil, O. J. Unogwu, Y. Filali, I. Bala, and H. Al-Shahwani, ''Exploring the top five evolving threats in cybersecurity: An in-depth overview,'' Mesopotamian J. Cyber Secur., vol. 2023, pp. 57–63, Mar. 2023.

[2] T. Fadziso, U. Thaduri, S. Dekkati, V. Ballamudi, and H. Desamsetti, ''Evolution of the cyber security threat: An overview of the scale of cyber threat,'' Digitalization Sustainability Rev., vol. 3, no. 1, pp. 1–12, 2023.

[3] R.Dillon, P. Lothian, S. Grewal, D. Pereira, and A. Kuah, ''Cyber security: Evolving threats in an ever changing world,'' in Digital Transformation in a Post-Covid World: Sustainable Innovation, Disruption and Change. Boca Raton, FL, USA: CRC Press, 2021, pp. 129–154.

[4] P. Vanin, T. Newe, L. L. Dhirani, E. O'Connell, D. O'Shea, B. Lee, and M. Rao, ''A study of network intrusion detection systems using artificial intelligence/machine learning,'' Appl. Sci., vol. 12, no. 22, p. 11752, Nov. 2022, doi: 10.3390/app122211752.

[5] H. Albasheer, M. Md Siraj, A. Mubarakali, O. Elsier Tayfour, S. Salih, M. Hamdan, S. Khan, A. Zainal, and S. Kamarudeen, ''Cyber attack prediction based on network intrusion detection systems for alert correlation techniques: A survey,'' Sensors, vol. 22, no. 4, p. 1494, Feb. 2022.

[6] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, ''Network intrusion detection system: A systematic study of machine learning and deep learning approaches,'' Trans. Emerg. Telecommun. Technol., vol. 32, no. 1, p. e4150, Jan. 2021.

[7] H. Liu and B. Lang, ''Machine learning and deep learning methods for intrusion detection systems: A survey,'' Appl. Sci., vol. 9, no. 20, p. 4396, Oct. 2019, doi: 10.3390/app9204396.

[8] L. Shahbandayeva, U. Mammadzada, I. Manafova, S. Jafarli, and A. Z. Adamov, ''Network intrusion detection using supervised and unsupervised machine learning,'' in Proc. IEEE 16th Int. Conf. Appl. Inf. Commun. Technol. (AICT), Oct. 2022, pp. 1–7.

[9] S. Gamage and J. Samarabandu, ''Deep learning methods in network intrusion detection: A survey and an objective comparison,'' J. Netw. Comput. Appl., vol. 169, Nov. 2020, Art. no. 102767. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804520302411

[10] K. S. Adewole, T. T. Salau-Ibrahim, A. L. Imoize, I. D. Oladipo, M. AbdulRaheem, J. B. Awotunde, A. O. Balogun, R. M. Isiaka, and T. O. Aro, ''Empirical analysis of data streaming and batch learning models for network intrusion detection,'' Electronics, vol. 11, no. 19, p. 3109, Sep. 2022.

[11] D. Bhosale and R. Ade, ''Intrusion detection using incremental learning from streaming imbalanced data,'' Int. J. Manag. Public Sector Inf. Commun. Technol., vol. 6, no. 1, pp. 9–20, Mar. 2015.

[12] M. R. Mohamed, A. A. Nasr, I. F. Tarrad, and M. Z. Abdulmageed, ''Exploiting incremental classifiers for the training of an adaptive intrusion detection model,'' Int. J. Netw. Secur., vol. 21, no. 2, pp. 275–289, 2019.

[13] M. Data and M. Aritsugi, ''AB-HT: An ensemble incremental learning algorithm for network intrusion detection systems,'' in Proc. Int. Conf. Data Sci. Appl. (ICoDSA), Jul. 2022, pp. 47–52.

[14] B. A. Tama and S. Lim, ''Ensemble learning for intrusion detection systems: A systematic mapping study and cross-benchmark evaluation,'' Comput. Sci. Rev., vol. 39, Feb. 2021, Art. no. 100357. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013720304573

[15] M. Torabi, N. I. Udzir, M. T. Abdullah, and R. Yaakob, ''A review on feature selection and ensemble techniques for intrusion detection system,'' Int. J. Adv. Comput. Sci. Appl., vol. 12, no. 5, pp. 6–7, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:236317529

[16] A.M. Bamhdi, I. Abrar, and F. Masoodi, ''An ensemble based approach for effective intrusion detection using majority voting,'' Telkomnika, vol. 19, no. 2, pp. 664–671, Apr. 2021.

[17] D.R. Patil and T.M. Pattewar, ''Majority voting and feature selection based network intrusion detection system,'' EAI Endorsed Trans. Scalable Inf. Syst., vol. 9, no. 6, p. e6, 2022.

[18] H. Xu and Y. Wang, ''A continual few-shot learning method via meta learning for intrusion detection,'' in Proc. IEEE 4th Int. Conf. Civil Aviation Saf. Inf. Technol. (ICCASIT), Oct. 2022, pp. 1188–1194.

[19] T. Wang, Q. Lv, B. Hu, and D. Sun, ''A few-shot class-incremental learning approach for intrusion detection,'' in Proc. Int. Conf. Comput. Commun. Netw. (ICCCN), Jul. 2021, pp. 1–8.

[20] J. Zheng, X. Ni, L. Li, K. Yu, and J. Zhang, ''An ensemble learning-based two-level network intrusion detection method,'' in Proc. Int. Conf. Comput. Eng. Artif. Intell. (ICCEAI), 2022, pp. 571–575.

[21] T. Wang, Q. Lv, B. Hu, and D. Sun, ''A few-shot class-incremental learning approach for intrusion detection,'' in Proc. Int. Conf. Comput. Commun. Netw. (ICCCN), Jul. 2021, pp. 1–8.