Comparative Analysis of Weighted Ensemble and Majority Voting Algorithms for Intrusion Detection in OpenStack Cloud Environments

Pravin Patil, Geetanjali Kale, Nidhi Bivalkar, Agneya Kolhatkar
Department of Computer Engineering, Pune Institute of Computer Technology, Pune, Maharashtra, India

***Abstract*—In the ever-evolving landscape of cybersecurity, the detection of malicious activities within cloud environments remains a critical challenge. This research aims to compare the effectiveness of two ensemble algorithms, the weighted ensemble algorithm and the majority voting algorithm, in the context of intrusion detection within an OpenStack cloud environment. To conduct this study, a dataset was generated using a network of 10 virtual machines, simulating the complex dynamics of a real cloud infrastructure. Various attack scenarios were simulated, and system metrics including CPU usage, memory utilization, and network traffic were monitored and logged. The weighted ensemble algorithm combines the predictions of multiple individual models with varying weights, while the majority voting algorithm aggregates predictions from multiple models. Through a rigorous experimental setup, these algorithms were applied to the generated dataset, and their performance was evaluated using standard metrics such as accuracy, precision, recall, and F1-score. These findings provide valuable insights into the strengths and weaknesses of ensemble algorithms for intrusion detection in cloud environments. It highlights the importance of selecting appropriate algorithms based on specific security requirements and threat profiles. Different attack scenarios may require different algorithmic approaches to achieve optimal results. Overall, this study contributes to the understanding of ensemble techniques in cloud security and offers a foundation for further research in optimizing intrusion detection strategies within dynamic and complex cloud environments. By identifying the strengths and weaknesses of different ensemble algorithms, cybersecurity professionals can make informed decisions in selecting the most suitable approach to enhance the security of cloud environments.**

Keywords*—Intrusion detection; ensemble algorithms; cloud security; openstack; weighted ensemble; majority voting

I. Introduction

The advancement of computing has ushered in transformative technologies, with cloud environments being at the forefront. These digital ecosystems offer unparalleled convenience, scalability, and connectivity, fundamentally reshaping the storage and processing of data. However, along with these advantages comes the pressing challenge of securing data within interconnected cloud systems. These systems, while efficient, create pathways for a diverse range of cyber threats that require robust and adaptable intrusion detection mechanisms. In the pursuit of fortifying cloud security, traditional intrusion detection approaches have played a crucial role [1], [3]. Nevertheless, these methodologies face limitations as attackers continuously evolve their tactics. Rule-based systems prescribe inflexible attack patterns, signature detection relies on pre-identified attack signatures, and anomaly detection, while effective against new attacks, often yields high false positive rates. To overcome these limitations, the integration of ensemble techniques into intrusion detection systems (IDS) has emerged as a promising strategy. Ensembles amalgamate the insights of multiple models to enhance accuracy and robustness, enabling systems to adapt to evolving attack strategies. Within the realm of ensembles, two methods stand out prominently: the weighted ensemble and the majority voting algorithms.

This research embarks on a comprehensive exploration of these two algorithms within the dynamic framework of OpenStack cloud environments. OpenStack, renowned as an open-source cloud platform, provides an intricate architecture and susceptibility to real-world cyber threats, making it an ideal evaluation ground for ensemble techniques. At the heart of our investigation lies the question of which ensemble algorithm, between the weighted ensemble and majority voting, demonstrates superior performance in the field of intrusion detection within OpenStack environments. To address this question, our methodology involves meticulously simulating a wide range of attack scenarios within the OpenStack ecosystem. By creating a synthetic environment consisting of virtual machines that mimic the complexities of cloud ecosystems, we subject our algorithms to various cyber-attacks. This deliberate diversification encompasses different tactics and vectors, resulting in a comprehensive evaluation of the algorithms' resilience and adaptability. Furthermore, during these simulations, we diligently record and analyze intricate system metrics. This detailed dataset sheds light on how the algorithms behave under dynamic attack conditions, enhancing our understanding of their effectiveness and response. In essence, the main contribution of this research lies in the empirical evaluation of the weighted ensemble and majority voting algorithms. Through rigorous experimentation and thorough analysis of the results, we uncover valuable insights into their operational dynamics, strengths, and limitations. Additionally, our findings have practical implications for the real-world deployment of these algorithms within intrusion detection systems. Fig. 1 gives an overall research workflow.

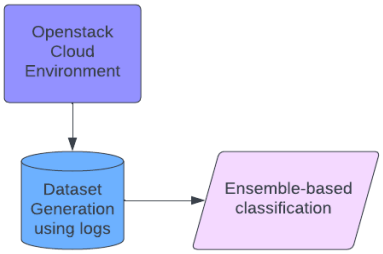

Fig. 1. Workflow.

This research paper unfolds in several sequential sections following the introduction. The Related Work section delves into existing intrusion detection systems, thereby establishing the groundwork for ensemble techniques. Following this, the Data Collection section provides an in-depth account of the creation of a synthetic dataset, designed to simulate a wide range of cyber-attacks within an OpenStack environment. Subsequently, in the Classification Algorithms section, ten different supervised learning models are introduced, serving as the foundation for subsequent evaluations of ensemble techniques. The obtained results yield a comprehensive performance analysis, comparing various metrics such as accuracy, precision, recall, and F1 score and further implications of those results are discussed. Finally, the Conclusion synthesizes the findings, highlighting the strengths of the research and proposing potential avenues for future investigations.

II. Related Work

The continuous pursuit of enhancing intrusion detection systems has propelled researchers to explore a diverse range of methodologies. Traditional approaches encompass rule-based systems, signature detection, and anomaly detection. Rule-based systems prescribe static attack patterns, yet struggle to accommodate the dynamic nature of evolving attack strategies.

Signature detection relies on predefined attack signatures, rendering it ineffective against novel attacks that evade established patterns [1], [4]. Wie et al. and W. Hu et al. used the AdaBoost Classifier for a Network IDS [5], [6]. A. Rai et al. designed optimized IDS using deep neural networks and the GradientBoost Classifier [7].

Ensemble methods leverage the collective insights of multiple models to enhance accuracy and resilience, enabling systems to adapt to emerging attack tactics. Notably, the weighted ensemble and majority voting algorithms have emerged as formidable contenders within the realm of ensemble-based intrusion detection. The weighted ensemble algorithm hinges on the principle of model weighting, dynamically assigning importance to individual model predictions [8]. This adaptability empowers the algorithm to excel across diverse attack scenarios, optimally adjusting the influence of each model based on its performance characteristics.

Conversely, the majority voting algorithm capitalizes on the synthesis of predictions from multiple models [9]. By establishing a consensus among models, this approach fosters robustness, mitigating the impact of errors arising from individual models. Several studies have explored the applications of ensemble techniques for intrusion detection [8], [2] [10], [9], [11], [12].

There has also been significant research on the applications of these classifiers on multi-tenant systems [8], [9]. While these previous studies have enriched the discourse on ensemble-based intrusion detection, the comparative assessment of the weighted ensemble and majority voting algorithms remains a relatively unexplored area, particularly within the dynamic context of OpenStack cloud environments [4]. It is within this realm that our research finds its foundation, systematically evaluating the performance of these algorithms in a relevant cloud security landscape. Previous studies have shed light on the potential of ensemble techniques. However, these studies often lack a comprehensive analysis of the strengths and weaknesses of the algorithms. The specific limitations of each method, particularly within the OpenStack environment, have not been thoroughly addressed. To bridge this gap, our research undertakes an extensive comparative analysis of the weighted ensemble and majority voting algorithms. By subjecting these algorithms to attack scenarios within OpenStack cloud environments, we aim to discern their nuanced responses and understand their operational dynamics in the face of complex threats. Through this exploration, our study strives to offer practical insights into the adaptability and effectiveness of these algorithms, enabling informed decision-making for intrusion detection in multi-tenant distributed systems.

III. Data Collection

The research methodology encompasses a seamless continuum from data collection to model training. System metrics are meticulously collected from a simulated OpenStack cloud, replicating real-world dynamics during attacks. The raw data undergoes thorough preprocessing, including cleaning and feature engineering, transforming the metrics into meaningful attributes for insightful analysis. The refined dataset is used to train weighted ensemble and majority voting models. The models undergo iterative adjustments and fine-tuning to optimize their intrusion detection capabilities. This integrated process establishes a robust evaluation framework for ensemble algorithms within the intricate context of OpenStack cloud scenarios.

A. Dataset Generation

In our pursuit of conducting a comprehensive evaluation, we systematically undertook the task of creating a virtual cloud environment utilizing the OpenStack framework. This task utilized the computational resources of two laptops, one with 8GB of RAM and the other with a substantial 32GB. Within this setup, we deployed ten strategically distributed virtual machines (VMs) across the laptops, each serving as a control node or a compute node. These roles mirrored the dynamics of a real-world cloud ecosystem and facilitated a highly realistic evaluation [13]. To further enhance the realism of our evaluation, we simulated cyber-attacks on this virtual cloud environment, with a focus on the control node, a vital component of the cloud infrastructure. The attacks

encompassed a wide range of threats, including Cross-Site Request Forgery (CSRF), XML External Entity (XXE) Injection, Brute Force, Cross-Site Scripting (XSS), Open Redirect, Directory Traversal, SQL Injection, Command Injection, and Remote Code Execution. Each attack category had the potential to impact various critical parameters within the cloud environment. Each attack category had distinct implications for the cloud environment, leading to specific performance metrics. Each attack category had the potential to influence these system performance metrics, resulting in nuanced impacts on the operational landscape of the cloud environment.

1) CSRF could compromise data integrity and disrupt user interactions, potentially affecting transaction success rates and request latencies [14].

2) XEE could impact system behaviour and data confidentiality, potentially influencing system response times and memory utilization [15].

3) Brute Force attacks on authentication mechanisms could have system-wide consequences, impacting authentication failure rates and overall system availability.

4) XSS could undermine user interactions and data integrity, posing threats to user session durations and engagement metrics.

5) Open Redirect vulnerabilities could impact user navigation experiences, potentially affecting click-through rates and user satisfaction levels.

6) Directory Traversal exploits could influence file access rates and disk I/O operations.

7) SQL Injection could jeopardize data confidentiality and system availability, influencing query execution times and database throughput.

8) Command Injection could manipulate system commands, potentially affecting CPU usage and system response times.

9) Remote Code Execution posed severe risks to system integrity and availability, potentially impacting memory usage and network traffic rates.

The execution of these attacks was randomized to ensure a diverse range of threat scenarios. To capture the dynamics of the cloud environment during attacks, we utilized the Netdata REST API service to collect real-time system metrics. These metrics were meticulously organised into a structured CSV format for subsequent analysis. The extent of our evaluation is evident in the execution of a total of 10,000 attack instances, showcasing the rigorous and dedicated nature of our study. This comprehensive exploration serves as a robust foundation for analyzing the performance of ensemble algorithms in real-world scenarios.

The distribution of attack and non-attack instances is clearly shown in Table I.

The dataset utilized in this study comprises a diverse range of system metrics and attributes collected from a simulated OpenStack cloud environment. With a total of 63 distinct parameters, each column represents a specific system parameter or feature [16]. These parameters encompass various measurements related to CPU utilization, memory consumption, disk activity, network behaviour, process behaviour, firewall activity, and more. Each row in the dataset corresponds to a specific time instance during simulated attack scenarios. A value of 1 or 0 is assigned to categorise the instances, where 1 denotes an attack instance and 0 represents a non-attack instance. For attack instances, an additional column specifies the type of attack executed, providing valuable insights into the nature of each attack scenario.

TABLE I. DATASET DISTRIBUTION

Attack Instances	10000
Non-attack instances	90000
Total instances	100000

B. Oversampling and Undersampling

The ratio of attack to non-attack instances is 1:9 causing the dataset to be slightly imbalanced. To improve the performance of classification algorithms, we employed two essential techniques: Synthetic Minority Over-sampling Technique (SMOTE) and Random Under-sampling. SMOTE generates synthetic instances for the minority class by interpolating existing instances, effectively balancing class proportions [17]. This technique mitigates the risk of the model favouring the majority class due to its higher representation. Conversely, Random Under-sampling reduces the majority of class instances randomly, aligning class distributions [18]. By employing SMOTE and Random Under-sampling, we aimed to strike a harmonious balance between class representations, enabling the models to train on a more equitable dataset.

IV. CLASSIFICATION ALGORITHMS

The dataset is split into training and testing data in a ratio of 1:4. 10 different supervised learning algorithms are trained on this data comprising 17998 non-attack instances and 2002 attack instances.

A. Supervised Classification Algorithms

1) Logistic regression: Logistic Regression is a linear classifier that estimates instance probabilities by identifying an optimal hyperplane separating different class data points. It is particularly essential when assuming a linear relationship between input features and the outcome.

2) Support Vector Machine (SVM): Support Vector Machine aims to find the hyperplane that maximizes the margin between data points of different classes, thereby enhancing the separation between classes. SVM is particularly effective in handling complex datasets and can handle non-linear decision boundaries through kernel transformations.

3) k-Nearest Neighbours (KNN): KNN is a powerful instance-based classification algorithm that focuses on the local neighbourhood of data points. It assigns a class label to a new instance based on the majority class of its k closest neighbours. This algorithm is beneficial for capturing the local characteristics of data, making it effective for tasks with irregular data distributions and localized patterns.

4) Random forest: Random Forest is a versatile ensemble learning technique that addresses overfitting and improves predictive performance. It constructs multiple decision trees during training and combines their outputs to make predictions. By doing so, it mitigates the risk of overfitting associated with individual decision trees and provides robust results across a variety of datasets.

5) Gradient boosting: Gradient Boosting is a powerful ensemble algorithm that constructs a strong predictive model by iteratively improving upon the errors of previous iterations. It sequentially builds a series of weak learners, often decision trees, and places emphasis on instances that were misclassified earlier. This approach is particularly advantageous for capturing complex relationships and delivering high predictive accuracy.

6) Adaptive Boosting (AdaBoost): AdaBoost is a boosting algorithm that iteratively trains weak learners and combines them into a robust ensemble model. What makes AdaBoost instrumental is its ability to adapt to difficult instances by assigning higher weights to misclassified instances in the previous iteration. This adaptability enhances the model's performance and is particularly beneficial when dealing with imbalanced class distributions [5].

7) Naive bayes: Naive Bayes is a probabilistic classification algorithm that makes an instrumental simplifying assumption: feature independence given the class. Despite this assumption, it's highly efficient, making it suitable for large datasets. Naive Bayes excels in text classification and tasks where the independence assumption approximately holds, showcasing its instrumental role in such contexts.

8) Decision tree: Decision Trees are simple yet effective models that recursively partition data based on feature attributes. They're instrumental in understanding the hierarchy of decisions within a model. However, they're prone to overfitting, which can be mitigated through ensemble methods like Random Forest or Gradient Boosting, making them an essential foundational element in machine learning.

9) Multi-Layer Perceptron (MLP): Multi-Layer Perceptron is an instrumental neural network architecture composed of multiple layers of interconnected neurons. Its ability to capture complex patterns in data makes it highly versatile across various domains. However, its instrumental application requires thoughtful architecture design and hyperparameter tuning to prevent overfitting and ensure effective learning [19].

10)XGBoost: XGBoost is an advanced gradient boosting library instrumental for improved model performance. It incorporates regularization techniques, optimized tree pruning, and effective handling of missing data [20]. XGBoost's instrumental role lies in its capability to provide robust results with minimal overfitting, making it a go-to choice for boosting-based ensemble methods.

All classification algorithms have been trained on the same data with default parameters. The data has been scaled using the standard scaler. The decision function shape for the SVM algorithm is One-vs-Rest by default and max iterations for Logistic Regression have been set to 1000 to account for the size of the dataset. Now let us explore the two ensemble-based classification algorithms in consideration for this comparative analysis.

B. Ensemble-based Classification Algorithms

1) Weighted ensemble classifier: A weighted ensemble classifier is a machine learning technique that combines the predictions of multiple base classifiers, each with its own assigned weight (see Fig. 2). The weights reflect the strengths and weaknesses of each base classifier, and they determine the contribution of each classifier's prediction to the final ensemble prediction. By giving more weight to the predictions of more accurate or reliable classifiers, and less weight to those of less accurate ones, the weighted ensemble aims to improve the overall performance of the ensemble. The weights of the classifiers were decided using cross-validation f1 scores and came out to be in the range of 0.0872 for KNN to 0.1049 for AdaBoost.

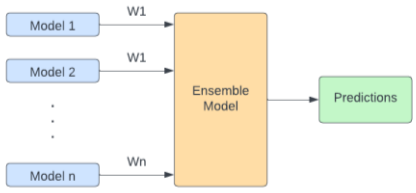

Fig. 2. Weighted ensemble classifier.

2) Majority voting ensemble classifier: The Majority Voting ensemble method is a powerful technique used to enhance the accuracy and robustness of machine learning models [9]. It involves combining the predictions of multiple individual models to make a final prediction. In Majority Voting, each model's prediction is considered a "vote" for a specific class label. The class label that receives the most votes becomes the ensemble's final prediction. This method leverages the wisdom of the crowd by aggregating the predictions of multiple models, which can lead to more accurate and reliable results. Fig. 3 diagrammatically shows its working. One of the key advantages of Majority Voting is its ability to reduce variance and errors. Even if some individual models make incorrect predictions, the ensemble can still provide accurate results if the majority of the models are correct. In cases where there is a tie in the votes, various strategies can be employed to handle it.

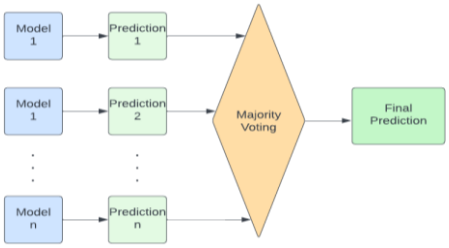

Fig. 3. Majority voting classifier.

V. Results and Discussion

In this section, we will conduct a detailed examination of the Weighted Ensemble and Majority Voting algorithms, specifically in the context of identifying potential attacks within OpenStack cloud configurations and being able to extend them to broader multi-tenant environments in the future. We will evaluate the effectiveness of these algorithms using various metrics such as accuracy, F1 score, precision, recall, and confusion matrices. By analyzing these metrics, we aim to gain insights into how well these algorithms distinguish between attack and non-attack instances in an OpenStack cloud environment. Furthermore, this section also discusses the reasoning behind choosing Majority Voting and Weighted Ensemble as the two algorithms in this comparison.

A. Performance Metrics

1) Confusion matrix: A confusion matrix provides a detailed breakdown of a model's predictions by comparing them against actual class labels. It includes four metrics: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). True positives represent the instances that were correctly predicted as positive by the model, while true negatives represent the instances that were correctly predicted as negative. False positives are instances incorrectly predicted as positive, and false negatives are those incorrectly predicted as negative. From these metrics, other evaluation metrics can be derived.

2) Accuracy: Accuracy is the ratio of correctly predicted instances to the total instances in a dataset, offering an overall performance assessment across all classes, particularly effective for balanced datasets. Nevertheless, in imbalanced class scenarios, accuracy might be skewed by the majority class. In such cases, metrics like precision, recall, and F1 score offer deeper insights into performance, particularly for identifying attacks in intricate multi-tenant setups.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (1)$$

3) Precision: Precision evaluates the proportion of true positive predictions out of all positive predictions made by the model (see Eq. (2)). It is an important metric when the cost of false positives is high, as it indicates how trustworthy the positive predictions are. A high precision value indicates a low rate of false alarms.

$$Precision = \frac{TP}{(TP + FP)} \quad (2)$$

4) Recall: Recall calculates the proportion of true positive predictions from all actual positive instances in the dataset. It is valuable when the cost of false negatives is high, as it measures how effectively the model captures all actual positives. A high recall value indicates that the model is good at identifying positives. However, a high recall value may come at the cost of more false positives.

$$Recall = \frac{TP}{(TP + FN)} \quad (3)$$

5) F1 Score: The F1 score is a balanced metric that takes into account both precision and recall. It is particularly valuable when the dataset is imbalanced and there is a need to consider false positives and false negatives. The F1 score is calculated as the harmonic mean of precision and recall, helping to strike a balance between them and providing a more comprehensive assessment of a model's performance.

$$F1\ Score = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)} \quad (4)$$

B. Performance

The performance results obtained from the Weighted Ensemble and Majority Voting classifiers based on the above performance five metrics are given in Table II.

The Weighted Ensemble Classifier holds a slight performance edge over the Majority Voting Classifier in accuracy, recall, and F1 score. Although the latter has higher precision, the difference between the two precision values is not as significant as that of the two recall values. It is also noteworthy that the Majority Voting Classifier incurred an extra 1161-second runtime compared to the Weighted Ensemble. Consequently, the Weighted Ensemble Classifier emerges as the more efficient and effective choice overall.

Table III and Table IV show the confusion matrices of both classifiers for a more in-depth review of the results.

TABLE II. Results

Algorithms	Performance Metrics			
	Accuracy	Precision	Recall	F1 Score
Weighted Ensemble	0.9942	0.9876	0.9535	0.9703
Majority Voting	0.9926	0.9920	0.9336	0.9619

TABLE III. Confusion Matrix for Weighted Ensemble

Weighted Ensemble	Actual		
	20000 logs	Non-attack	Attack
Prediction	Non-attack	17974	24
	Attack	93	1909

TABLE IV. Confusion Matrix for Majority Voting

Majority Voting	Actual		
	20000 logs	Non-attack	Attack
Prediction	Non-attack	17983	15
	Attack	133	1869

TABLE V. F1 Scores after Oversampling

Algorithm with oversampling	F1 Score
Weighted Ensemble	0.9711
Majority Voting	**0.9702**

From these matrices, one can see that the two algorithms performed very similarly. The Weighted Ensemble could classify more attack instances correctly whereas the Majority Voting Classifier could classify more non-attack instances correctly. The Weighted Ensemble does have one disadvantage which is that it requires the calculation of weights as an extra step before the classification process. Furthermore, we have also extracted performance results after using SMOTE and RandomUndersampler on the data to get a balanced dataset [17], [18].

Table V clearly shows that oversampling reduced the performance gap between the two algorithms but the Weighted Ensemble still has slightly higher performance. Add to this, the fact that the Weighted Ensemble did not require the extra pre-processing step of Oversampling and Undersampling.

C. Discussion

Now, we delve into the evaluation of classification algorithms, and their comparison with Weighted Ensemble and Majority Voting with a particular emphasis on the stability of predictions. To simulate scenarios involving unseen data, random test-train dataset splits are utilized. A key metric used to assess the consistency of algorithmic performance is the standard deviation of F1 scores for each split.

Notably, the KNN algorithm stands out as it exhibits the lowest standard deviation (see Fig. 4), demonstrating a remarkable level of consistency across diverse dataset splits. However, its performance pales in comparison to other algorithms as shown in Fig. 5. From Fig. 4 it is evident that both the Weighted Ensemble and Majority Voting techniques prove to be the strongest contenders, displaying low standard deviations and highlighting their stability. One could argue that the Random Forest algorithm displays similar levels of stability but Fig. 5 clearly shows the lower F1 score as compared to Majority Voting and Weighted Ensemble.

This discussion provides valuable insights into the realm of intrusion detection algorithms. These algorithms carry the dual responsibility of accurately classifying threats while avoiding the pitfall of overfitting specific threat types.

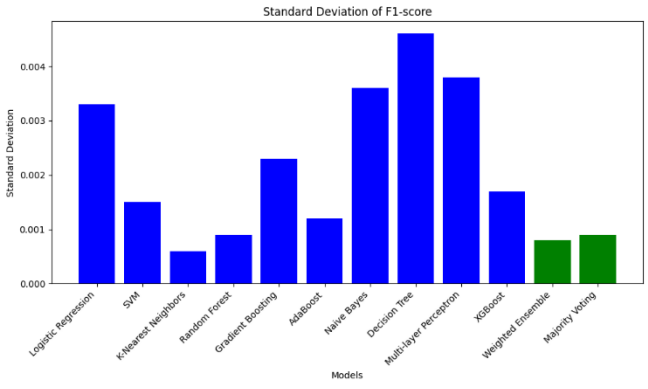

Fig. 4. Standard deviation of f1 scores of algorithms.

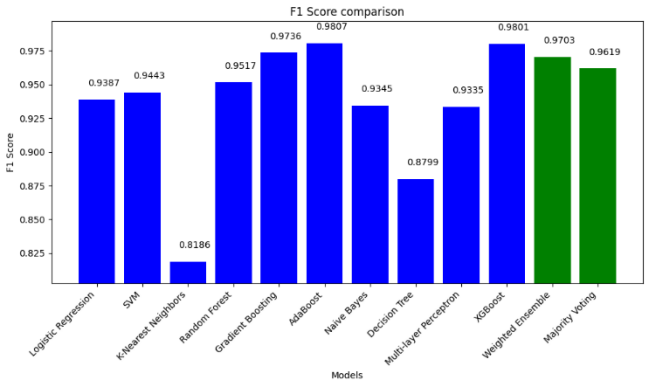

Fig. 5. Comparison of f1 scores of algorithms.

VI. Conclusion

Thus, we comprehensively compared the Weighted Ensemble and Majority Voting algorithms for intrusion detection in OpenStack cloud environments. Our analysis aimed to assess their ability to identify attacks and non-attacks and their real-world applicability. Through extensive evaluation, both algorithms demonstrated strong performance in distinguishing between attack and non-attack instances, highlighting their effectiveness as ensemble-based intrusion detection methods. While the Weighted Ensemble algorithm showcased a slight edge in terms of accuracy, recall, and F1 score, it is important to note that both algorithms demonstrated comparable performance. Additionally, the runtime analysis revealed that the Weighted Ensemble algorithm exhibited faster processing times compared to the Majority Voting algorithm, highlighting its potential efficiency advantage during real-time intrusion detection scenarios. However, it is essential to consider that real-time performance is influenced by various dynamic factors specific to the operational environment. Although one outperforms the other, both these algorithms display high performance along with stability which underscores their resilience in addressing challenges posed by unseen security threats.

Our study contributes to the ongoing discourse on enhancing cybersecurity within multi-tenant cloud environments. The findings underscore the role of ensemble techniques as valuable tools for bolstering intrusion detection capabilities. As the landscape of cyber threats evolves, future research could explore further optimizations and extensions to ensemble algorithms, aiming to refine their performance in real-world cloud environments. In conclusion, our investigation advances the understanding of ensemble-based intrusion detection, facilitating more informed decisions for ensuring the security and resilience of cloud infrastructures.

References

[1] A. Valdes, K. Skinner, Adaptive, “Model-Based Monitoring for Cyber Attack Detection”, International Workshop on Recent Advances in Intrusion Detection, Berlin, Heidelberg, 2000.

[2] J. J. Shirley and M. Priya, "A Comprehensive Survey on Ensemble Machine Learning Approaches for Detection of Intrusion in IoT Networks," 2023 International Conference on Innovations in Engineering and Technology (ICIET), Muvattupuzha, India, 2023, pp. 1-10, doi: 10.1109/ICIET57285.2023.10220795.

[3] M. Yassin, H. Ould-Slimane, C. Talhi and H. Boucheneb, "Multi-tenant intrusion detection framework as a service for SaaS," in IEEE Transactions on Services Computing, doi: 10.1109/TSC.2021.3077852.

[4] B. I. Santoso, M. R. S. Idrus and I. P. Gunawan, "Designing Network Intrusion and Detection System using signature-based method for protecting OpenStack private cloud," 2016 6th International Annual Engineering Seminar (InAES), Yogyakarta, Indonesia, 2016, pp. 61-66, doi: 10.1109/INAES.2016.7821908.

[5] Wei Hu and Weiming Hu, "Network-based intrusion detection using Adaboost algorithm," The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), Compiegne, France, 2005, pp. 712-717, doi: 10.1109/WI.2005.107.

[6] W. Hu, W. Hu and S. Maybank, "AdaBoost-Based Algorithm for Network Intrusion Detection," in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 38, no. 2, pp. 577-583, April 2008, doi: 10.1109/TSMCB.2007.914695.

[7] A. Rai, "Optimizing a New Intrusion Detection System Using Ensemble Methods and Deep Neural Network," 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), Tirunelveli, India, 2020, pp. 527-532, doi: 10.1109/ICOEI48184.2020.9143028.

[8] P. Patil and R. Ingle, "Meta-ensemble based classifier approach for attack detection in multi-tenant distributed systems," 2020 International Conference for Emerging Technology (INCET), Belgaum, India, 2020, pp. 1-6, doi: 10.1109/INCET49848.2020.9154077.

[9] Pravin Patil*, Dr. Geetanjali Kale. (2022). Stacked Anomaly Detector Guided Side Channel Attacks Detection in Multi Tenant Distributed Systems. Scandinavian Journal of Information Systems, 34(2), 17–26.

[10] E. Roponena and I. Polaka, "Classifier Selection for an Ensemble of Network Traffic Analysis Machine Learning Models," 2022 63rd International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS), Riga, Latvia, 2022, pp. 1-6, doi: 10.1109/ITMS56974.2022.9937116.

[11] E. Roponena and I. Polaka, "Classifier Selection for an Ensemble of Network Traffic Analysis Machine Learning Models," 2022 63rd International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS), Riga, Latvia, 2022, pp. 1-6, doi: 10.1109/ITMS56974.2022.9937116.

[12] V. Timčenko and S. Gajin, "Ensemble classifiers for supervised anomaly based network intrusion detection," 2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 2017, pp. 13-19, doi: 10.1109/ICCP.2017.8116977.

[13] Z. Chen, W. Dong, H. Li, P. Zhang, X. Chen and J. Cao, "Collaborative network security in multi-tenant data center for cloud computing," in Tsinghua Science and Technology, vol. 19, no. 1, pp. 82-94, Feb. 2014, doi: 10.1109/TST.2014.6733211.

[14] W. H. Rankothge and S. M. N. Randeniya, "Identification and Mitigation Tool For Cross-Site Request Forgery (CSRF)," 2020 IEEE 8th R10 Humanitarian Technology Conference (R10-HTC), Kuching, Malaysia, 2020, pp. 1-5, doi: 10.1109/R10-HTC49770.2020.9357029.

[15] R. Shahid, S. N. K. Marwat, A. Al-Fuqaha and G. B. Brahim, "A Study of XXE Attacks Prevention Using XML Parser Configuration," 2022 14th International Conference on Computational Intelligence and Communication Networks (CICN), Al-Khobar, Saudi Arabia, 2022, pp. 830-835, doi: 10.1109/CICN56167.2022.10008276.

[16] B. W. Masduki and K. Ramli, "Improving intrusion detection system detection accuracy and reducing learning time by combining selected features selection and parameters optimization," 2016 6th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Penang, Malaysia, 2016, pp. 397-402, doi: 10.1109/ICCSCE.2016.7893606.

[17] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, ” SMOTE: Synthetic Minority Over-sampling Technique”, arXiv:1106.1813 [cs.AI]

[18] R. Mohammed, J. Rawashdeh and M. Abdullah, "Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results," 2020 11th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 2020, pp. 243-248, doi: 10.1109/ICICS49469.2020.239556.

[19] J. Esmaily, R. Moradinezhad and J. Ghasemi, "Intrusion detection system based on Multi-Layer Perceptron Neural Networks and Decision Tree," 2015 7th Conference on Information and Knowledge Technology (IKT), Urmia, Iran, 2015, pp. 1-5, doi: 10.1109/IKT.2015.7288736.

[20] Tianqi Chen, Carlos Guestrin, ”XGBoost: A Scalable Tree Boosting System”, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Pages 785-794.