Comparative Evaluation of Machine Learning Techniques for Enhanced Intrusion Detection Systems Performance

Atef Bentahar[1], Sami Abdelli[2], Slimane Boukhalfa[3], Riad Saidi[4], Tarek Bentahar[5]
[1]Laboratory of Science for Mathematics, Computer Science and Engineering Applications
[1,2,3]Department of Informatics, Institute of Science
[1,2,3]University Center of Barika, Barika, 05400, Algeria
[4]LAAAS Laboratory
[4]Dept Commun Base Science and Technology
[4]Mustapha Ben Boulaid Batna-2 University, Batna, Algeria
[5]LABGET-laboratery
[5]Department of Electrical Engineering, Faculty of sciences and technology
[5]Larbi Tebessi University, Tebessa, 12002, ALGERIA
[1]atef.bentahar@cu-barika.dz, [2]samiabdelli44@gmail.com, [3]boukhalfaslimane264@gmail.com,
[4]r.saidi@univ-batna2.dz, [5]tarek.bentahar@univ-tebessa.dz
[1]https://orcid.org/0000-0003-3523-4459, [5]https://orcid.org/0000-0001-7976-9899

***Abstract*— In today's rapidly evolving digital landscape, safeguarding computer systems from increasingly sophisticated and frequent cyberattacks is more critical than ever. Intrusion detection plays a pivotal role in maintaining the integrity and confidentiality of sensitive data. Leveraging cutting-edge machine learning advancements, this paper explores the development of highly efficient and responsive security systems. We rigorously evaluate various machine learning techniques for intrusion detection. The results indicate that the Random Forest Classifier stands out for its exceptional precision and speed, making it an ideal choice for real-time applications. Additionally, our research benchmarks these models against existing relevant works, demonstrating the clear superiority of our implementation across multiple performance metrics.**

Keywords*— *Machine Learning, Intrusion Detection Systems, cybersecurity

I. INTRODUCTION

In today's digital age, securing information systems has become a critical priority. Cyberattacks are growing in both sophistication and frequency, posing serious threats to the confidentiality, integrity, and availability of data. In this environment, intrusion detection is a key component of cybersecurity, tasked with identifying and neutralizing malicious activities that compromise systems at both the software and hardware levels.

Recent advancements in machine learning provide new opportunities to enhance intrusion detection capabilities. By leveraging these techniques, it is possible to develop more effective and responsive detection systems that can quickly identify anomalies and suspicious behaviors within data streams.

Given the growing complexity of cyberattacks, several critical questions arise:

- How can machine learning techniques be effectively used for intrusion detection?
- What is the most suitable machine learning technique to achieve this goal?

Our paper aims to address these issues by exploring the potential of machine learning in intrusion detection and identifying best practices for improving the security of information systems.

The primary objective is to evaluate and compare the performance of various machine learning techniques for detecting intrusions. We aim to identify the most effective methods for detecting malicious activities and to understand the factors that influence the accuracy and responsiveness of intrusion detection systems (IDS).

Furthermore, we conduct a comparative analysis with related works to validate the efficiency of our model.

To achieve our objectives, we employ several machine learning techniques. These techniques were selected for their ability to handle complex data and deliver accurate results in IDS.

We evaluate the performance of our models using various metrics such as Accuracy, Precision, Recall, F1 Score, as well as training and prediction times, and other relevant criteria. These metrics enable us to assess the reliability and efficiency of each algorithm in detecting intrusions.

The rest of the paper is organized as follows: Section II presents the Related Works, providing an overview of previous research and approaches in intrusion detection using machine learning. Section III details the Architectures and Implementations, explaining the structure and configuration of the models used in this study. Section IV covers the Results and Discussion, where we analyze the performance of the evaluated models. Section V provides a Comparison with the Results of Other Works, highlighting the effectiveness of our implementations compared to existing solutions. Finally, Section VI concludes with the Conclusion, summarizing the key findings and potential future research directions.

II. RELATED WORKS

A. Ahmim [1] proposed a new IDS that combines multiple classifier approaches based on decision tree and rule-based concepts, specifically using the Reduced Error Pruning tree (REP tree) [2], JRip algorithm [3], and Forest PA [4]. The first two methods take dataset features as inputs and classify network traffic as either Attack or Benign. The third classifier uses both the initial dataset features and the outputs from the first two classifiers as inputs. Experimental results are obtained using the CICIDS 2017 dataset [5], in terms of accuracy, detection rate, false alarm rate, and system runtime.

In [6], Aydin et al. proposed a hybrid Intrusion Detection System (IDS) that integrates two anomaly-based detection methods: i) Packet Header Anomaly Detection (PHAD) and ii) Network Traffic Anomaly Detection (NETAD), in combination with the signature-based IDS Snort. Tested on the IDEVAL dataset [7], the hybrid system demonstrated a substantial improvement in detecting attacks compared to conventional signature-based systems.

Chung and Wahid [8] addressed the challenge of decision rule generation by employing a rough set of rules based on swarm intelligence (IDS-RS) for feature selection and a Simplified Swarm Optimization with Weighted Local Search (SSO-WLS) for data classification. Their work presented a complete system solution to enhance the rule extraction process by weighting three predefined constants in the SSO search. Experimental results on the KDD CUP 1999 dataset [9] indicated that the proposed hybrid network IDS achieved a strong overall performance with an average accuracy of 93.3% across 20 runs.

In [10], Wang et al. proposed an intrusion detection approach called FC-ANN, which is based on Artificial Neural Networks (ANN) and fuzzy clustering. The FC-ANN approach comprises three key modules: the fuzzy clustering module, the ANN module, and the fuzzy aggregation module. The fuzzy clustering module partitions the dataset into clusters, while the ANN module learns patterns for each subset. The fuzzy aggregation module aggregates the results from the various ANNs to reduce detection errors. FC-ANN was tested on the KDD CUP 1999 dataset and proved effective against infrequent attack types, such as R2L and U2R attacks.

Meanwhile, Kim et al. [11] integrated both misuse detection and anomaly detection models into a decomposition framework. Their study utilized the C4.5 decision tree algorithm [12] and several one-class SVM models. Experimental results on the NSL-KDD dataset [13] showed that the proposed hybrid IDS outperformed conventional methods in terms of detection performance, training time, and testing time.

A. Ahmim [1] achieved promising results in terms of accuracy, detection rate, and false alarm rate using the CICIDS2017 dataset. However, the proposed system may present increased complexity in configuration and result interpretation.

Aydin et al. [6] improved the detection of attacks compared to signature-based systems, but their approach may require substantial resources to maintain rule databases and manage configurations.

Chung and Wahid [8] developed a method that offers good accuracy in intrusion detection, yet it may be limited by its reliance on predefined constants and requires significant expertise for proper tuning.

Wang et al. [10] introduced the FC-ANN approach for intrusion detection. While effective against certain types of attacks, this method demands considerable computational power and complex parameter configurations to achieve optimal results.

Kim et al. [11] proposed a method with high accuracy, but it has limitations due to the complexity of implementation and the resources needed for training and testing the models.

This paper presents the implementation of various machine learning techniques to improve the performance of IDS by overcoming previous limitations and determining the most effective machine learning algorithm for this purpose.

III. Architectures and Implementations

This section details the steps that led to the results of our study. We begin by presenting the UNSW-NB15 database [14] used in our work, followed by the necessary preprocessing steps for its utilization.

A. UNSW-NB 15 dataset Description

The UNSW-NB15 dataset is a network intrusion detection dataset created by the University of New South Wales (UNSW) in Australia. It simulates real-world network traffic, containing both normal and malicious traffic. The dataset includes various types of attacks, such as denial-of-service (DoS), reconnaissance, backdoors, and more.

Below some key features of this dataset are presented:

- Binary classification: The dataset labels traffic as either normal (0) or an attack (1).
- Size: It contains a total of 2,540,044 records, distributed across four CSV files (UNSW-NB15_1.csv, UNSW-NB15_2.csv, UNSW-NB15_3.csv, UNSW-NB15_4.csv).
- Training and testing sets: A part of the dataset is specifically divided into training and test sets (UNSW-NB15_training-set.csv and UNSW-NB15_testing-set.csv). The test set consists of 82,332 records.
- This dataset also includes 9 different types of attacks. The complete dataset contains 82,332 instances, of which 45,332 are malicious attacks and 37,000 are benign (legitimate/normal) network traffic instances.
- UNSW-NB15 dataset contains 45 system-related features and additional labeling features. These labeling features help classify the data based on different security scenarios: (i) Binary Data, (ii) Categorical Data

To ensure the construction of an accurate model, it is essential to conduct thorough exploratory analyses on the dataset and its features before preprocessing for machine learning.

B. Selection of Learning Algorithm

The selection of a learning algorithm for our study is a crucial yet often complex decision. Typically, an empirical approach is required, involving the testing and evaluation of multiple algorithms to identify the most suitable one. We decided to test several algorithms commonly used in the literature to compare their performance and determine which one best meets our objectives.

Here is the list of algorithms we will evaluate and compare to make an informed decision:

- Random Forest Classifier
- Extra Trees Classifier
- Decision Tree Classifier
- Multilayer Perceptron (MLP)

- K-Nearest Neighbors (KNN)
- Logistic Classifier
- Gaussian Naive Bayes

C. *Evaluation Criteria*

The goal of any classification algorithm is to build a model using labeled data that can accurately identify the class of new, unlabeled data. Several performance metrics are commonly used to evaluate a model's effectiveness, including Accuracy, Precision, Recall, and the F1 score. Additionally, practical factors like training time and prediction time are important to consider.

*1) Accuracy :*represents the percentage of correct predictions made by the model out of all predictions.

2) Precision :measures how many instances predicted as "Normal" are actually correct (True Positives) out of all instances predicted as "Normal" (both correct and incorrect). A low precision score indicates the model misclassifies many "Anomaly" cases as "Normal."

*3) Recall :*is the ratio of correctly predicted "Normal" instances out of all actual "Normal" instances (even if some were misclassified as "Anomaly"). A low recall score suggests the model incorrectly classifies many "Normal" cases as "Anomaly."

In imbalanced datasets, it is possible to have high accuracy but low precision or recall, especially when the model favors the majority class and neglects the minority class.

4) The F1 score : is the harmonic mean of precision and recall. It provides a balanced evaluation, especially useful when both precision and recall are important, giving a better sense of overall performance in classifying positive instances.

5) Training time : refers to how long it takes to train the model on the training data. A short training time indicates that the model can quickly converge to an optimal solution, which is beneficial when retraining with new data or tuning hyperparameters.

6) Prediction time : is the time the model takes to classify new data after being trained. A short prediction time is crucial for real-time applications or those requiring fast performance.

D. *Experimental Hardware and Software Environment*

Our work is conducted in a local environment, using tools integrated with Anaconda and Jupyter Notebook to manage and analyze the UNSW-NB15 dataset, which is stored locally on our hard drive.

The hardware setup consists of a DESKTOP PC, equipped with an Intel Core i3-7020U CPU @ 2.30GHz and 8 GB of RAM. This configuration is sufficient for efficiently handling data preprocessing tasks and running machine learning models without the need for GPU or other hardware acceleration.

Using Python in a Jupyter Notebook environment allows us to import, visualize, and analyze data directly on the local machine. This setup provides maximum flexibility for real-time testing of different configurations and parameters for machine learning algorithms.

Operating in a local environment gives us full control over the data analysis process, ensuring that system performance is optimized for our specific setup, without cloud-based limitations.

IV. RESULTS AND DISCUSSION

In this section, we will review and interpret the results from our experiences. We will start by displaying the confusion matrices for each algorithm, as illustrated in the following figures (Fig. 1, 2, 3, 4, 5, 6 and 7).

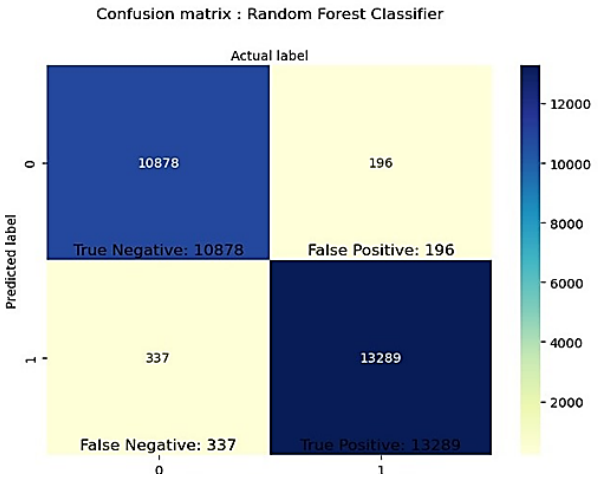

Fig. 1 Confusion Matrix for Random Forest Classifier

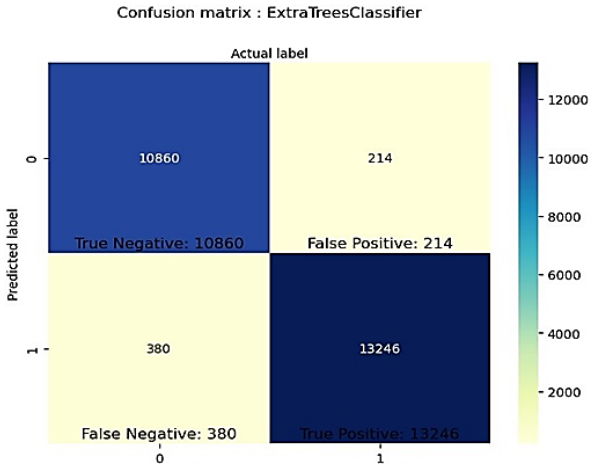

Fig. 2 Confusion Matrix for Extra Trees Classifier

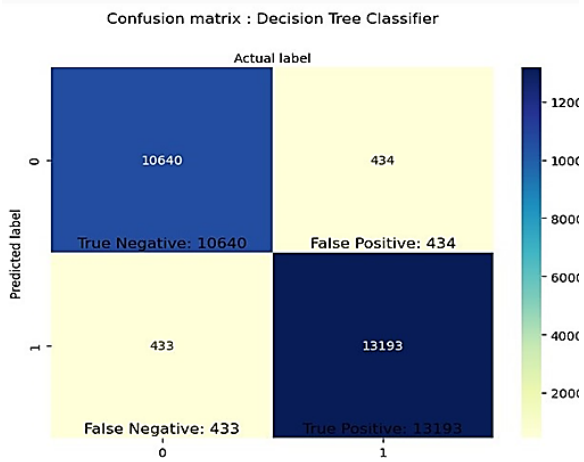

Fig. 3 Confusion Matrix for Decision Trees Classifier

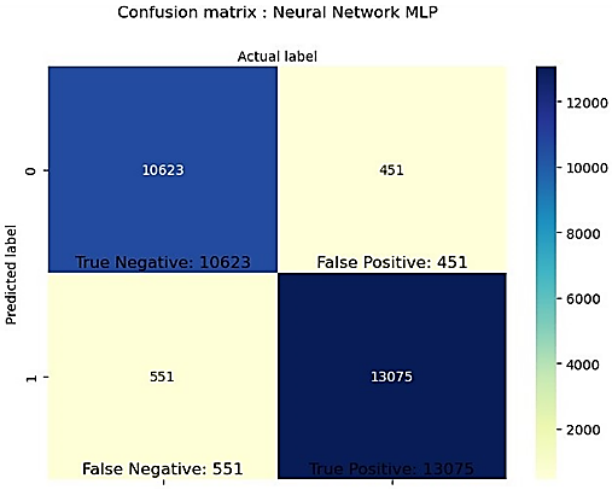

Fig. 4 Confusion Matrix for Neural Network MLP

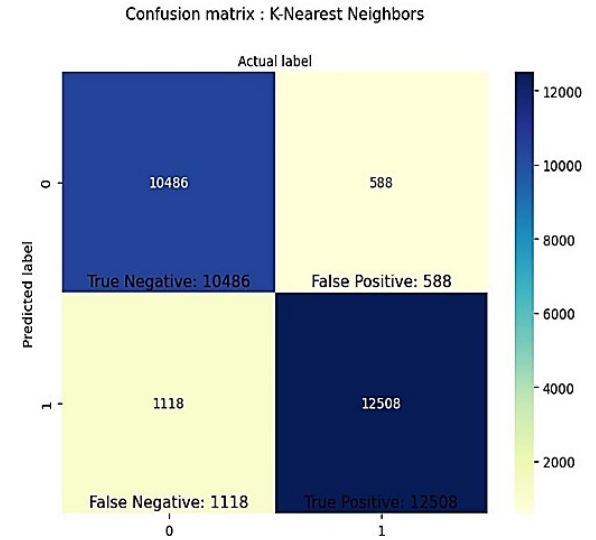

Fig. 5 Confusion Matrix for K-Nearest Neighbors

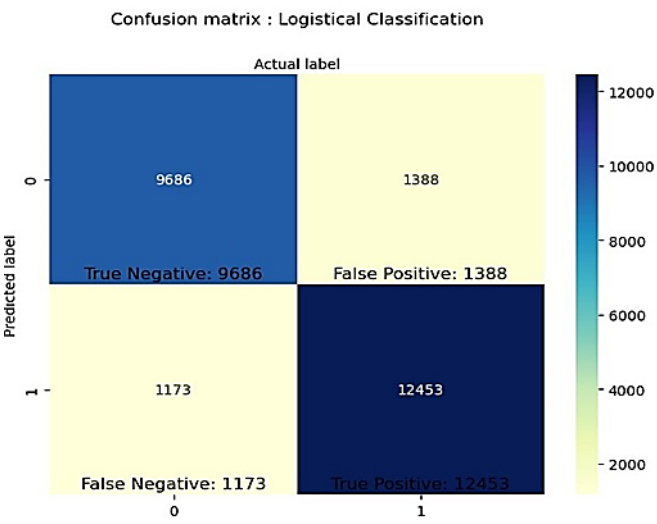

Fig. 6 Confusion Matrix for Logistical Classification

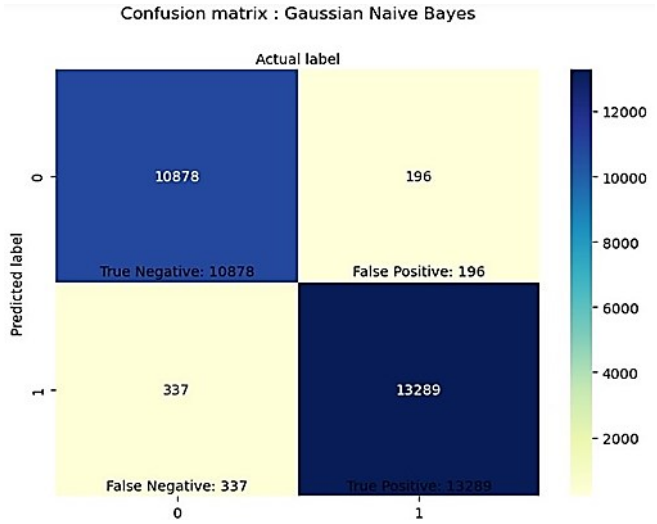

Fig. 7 Confusion Matrix for Gaussian Naive Bayes

From these confusion matrices, we can calculate accuracy, precision, recall, and the F1 score. By also factoring in the training score, training time, prediction time, and total time, we obtain the values presented in Table I, shown below.

TABLE I. PERFORMANCE OF OUR VARIOUS IMPLEMENTATIONS USING DIFFERENT ALGORITHMS

	Evaluation criteria							
ML Algorithm	***Accuracy***	***Precision***	***Recall***	***F1_score***	***Training score***	***Training time***	***Prediction time***	***Total time***
Random Forest Classifier	97.84%	98.55%	97.53%	0.980340	0.999913	11.502875	0.376489	11.879364
Extra Trees Classifier	97.60%	98.41%	97.21%	0.978070	0.999913	4.130365	0.234373	4.364768
Decision Tree Classifier	96.49%	96.82%	96.82%	0.968187	0.999913	0.891612	0.000000	0.891612
Neural Network MLP	95.94%	96.67%	95.96%	0.963097	0.963978	23.735153	0.015631	23.750784
K-Nearest Neighbors	93.09%	95.51%	91.80%	0.936157	0.964395	0.005997	26.855669	26.861687
Logistical Classification	89.64%	89.97%	91.39%	0.906761	0.896325	1.048819	0.000000	1.048819
Gaussain Naive Bayes	78.64%	83.75%	76.05%	0.799108	0.783245	0.046824	0.031246	0.07807

A. Models Accuracy

For Random Forest Classifier, the high accuracy (97.84%) and F1 score (0.980340) indicate excellent performance, balancing both precision and recall. The random state ensures reproducibility, and its ensemble method reduces overfitting while maintaining strong generalization.

Decision Tree Classifier provides good performance with simplicity. While less effective than ensemble models, its transparency makes it easier to interpret decisions.

Neural Network (MLP) Performed well due to hidden layer configuration (20, 20), 'relu' activation to avoid vanishing gradients, and 'adam' solver for efficient training. However, it did not outperform Random Forest or Extra Trees, which showed better accuracy and precision.

Using 3 neighbors in KNN balanced precision and recall, minimizing overfitting. However, it performed worse due to sensitivity to input data and lack of explicit generalization.

Lower performance of Logistic Regression compared to other models. However, it is suitable for binary classification with linear relationships, offering simplicity and ease of interpretation.

B. Execution Times

This section, highlights the training and prediction times for various machine learning algorithms and compares their performance:

Tree-based algorithms (Random Forest, Extra Trees, and Decision Tree) have relatively fast training times, with Decision Tree being the quickest.

Neural Networks (MLP) have the longest training time.

For prediction time, tree-based algorithms remain efficient, with Decision Tree offering the fastest response. In contrast, K-Nearest Neighbors is significantly slower.

Logistic Regression and Gaussian Naive Bayes show low execution times for both training and prediction, making them efficient choices for fast computation.

In summary, Random Forest Classifier is the most suitable algorithm for intrusion detection due to:

- High accuracy, which is critical for minimizing false alerts and maximizing threat detection.
- Fast prediction time, making it ideal for real-time applications that require immediate responses to threats.
- Effective handling of large datasets without needing dimensionality reduction, preserving key features for attack prediction.
- Robustness to imperfect data, making it capable of handling missing or incomplete network data efficiently.

C. Precision-Recall (PR) Curve

The Precision-Recall (PR) curve analysis, with an Average Precision (AP) of 0.97 as shown in fig. 8, indicates that the Random Forest Classifier achieves an excellent balance between precision and recall. The curve's proximity to the upper right corner of the graph confirms that the model consistently performs well across various threshold values. This high AP value demonstrates the classifier's strong ability to accurately classify positive samples while correctly identifying the majority, making it highly effective for tasks involving imbalanced data where identifying positive instances is critical.

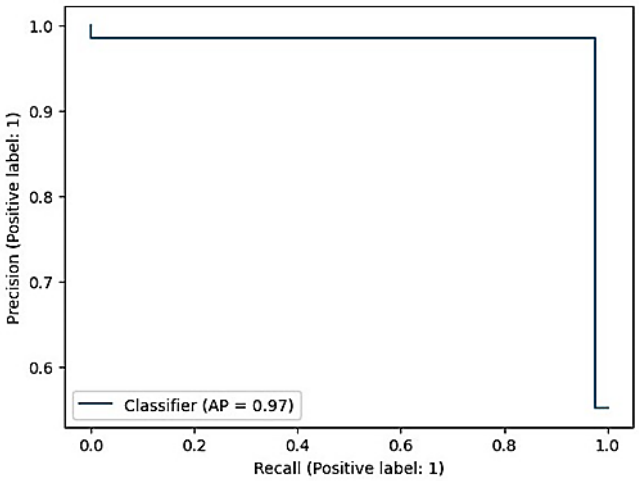

Fig. 8 Precision-Recall Curve for Random Forest Classifier

D. Receiver Operating Characteristic (ROC) Curve

The ROC (Receiver Operating Characteristic) curve evaluates the performance of a binary classification model by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) across various threshold values. The AUC (Area Under the Curve) measures the model's ability to distinguish between positive and negative classes, where an AUC value of 1 indicates perfect performance. In this case, the AUC is 1, meaning the model accurately distinguishes all positive instances (TPR = 1) without incorrectly classifying any negative instances as positive (FPR = 0). This reflects flawless classification performance, as illustrated in fig. 9.

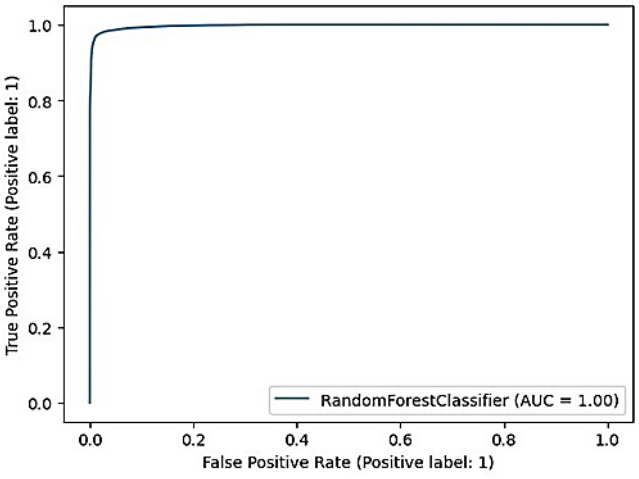

Fig. 9 ROC Curve for Random Forest Classifier

V. Comparison with the Results of Other Works

In this section, we compare our results, particularly the performance of our top model (Random Forest Classifier), with those from other relevant works to validate the effectiveness of our approach.

A. Comparison criteria

We selected the criteria in Table II because they are widely used in other studies, facilitating comparison by standardizing these measures. The key performance metrics are:

- Accuracy: Represents the overall effectiveness of the model in correctly classifying both normal and attack records.
- False Positives (FP): Number of normal records mistakenly classified as attacks.
- False Negatives (FN): Number of attack records incorrectly classified as normal.
- Total Normal (TN): Total number of records classified as normal.
- Total Attacks (TA): Total number of records classified as attacks.
- Detection Rate (DR): Indicates the percentage of correctly detected attacks.

$$DR = [(TA - FN)/TA] * 100 \quad (1)$$

- False Alarm Rate (FAR): Measures the percentage of normal records incorrectly classified as attacks.

$$FAR = [FP/TN] * 100 \quad (2)$$

TABLE II. COMPARISON WITH THE RESULTS OF OTHER WORKS

	Relevant Works							
Comparison criteria	***Our Random Forest***	***REP Tree [15]***	***MLP [15]***	***REP_JRip ForestPA [1]***	***WISARD [16]***	***J48 Con-Solidated [17]***	***LIBSVM [18]***	***FURIA [15]***
FAR	1.76%	4.83%	7.35%	1.14%	2.86%	6.64%	5.13%	3.16%
DR	97.52%	91.64%	77.83%	94.47%	48.17%	92.02%	54.59%	90.5%
Accuracy	97.84%	93.40%	85.24%	96.66%	72.65%	92.68%	74.73%	93.66%
Temp	11.50s	2.73s	942.98s	159.5s	13.47s	105.46s	318.6s	234.98s
Apprentissage	0.37s	0.52s	1.61s	2.27s	243.28s	0.61s	343.96s	0.96s

B. Comparison analysis

A comparative analysis of our model (Random Forest) against other well-known classification models highlights its advantages in terms of accuracy, training time, and prediction time:

Random Forest (Our Implementation): It has a fast training time of 11.50 seconds and a prediction time of 0.37 seconds, making it highly efficient for real-time applications. It achieves an exceptional accuracy of 97.84%, with a high detection rate (DR) of 97.52% and a low false alarm rate (FAR) of 1.76%. This makes it a top choice compared to other models.

REP_JRip_forestPA [1]: With a training time of 159.5 seconds, prediction time of 2.27 seconds, and an accuracy of 96.66%, this model shows good performance but is outperformed by Random Forest in all metrics.

J48 Consolidated [17]: Though it has decent performance with a training time of 105.46 seconds and prediction time of 0.61 seconds, it is slower compared to Random Forest.

LIBSVM [18]: This model has the longest training time (318.6 seconds) and high prediction time (343.96 seconds), making it unsuitable for fast applications.

REP Tree [15]: While quick (2.73 seconds training, 0.52 seconds prediction), it has lower accuracy (93.40%), DR (91.64%), and higher FAR (4.83%) than Random Forest.

MLP[15] : Despite good accuracy, its long training time (942.98 seconds) and slower prediction time (1.61 seconds) make it less competitive.

WISARD [16]: Shows poor performance with a low DR (48.17%) and high FAR (2.86%).

FURIA [15]: Offers a good DR (90.5%) but has a higher FAR (3.16%) compared to our Random Forest model.

In summary, Random Forest stands out as the most balanced and efficient model in terms of accuracy, detection rate, and speed, making it the best choice for real-time and high-accuracy applications.

VI. CONCLUSION

Recent advances in machine learning offer new opportunities to enhance detection systems, enabling the development of more effective and responsive defenses. This paper aims to evaluate and compare the performance of various machine learning techniques for intrusion detection, identifying the most effective approaches and benchmarking them against existing research. Techniques such as Random Forest, Extra Trees, Decision Tree, Neural Network MLP, K-Nearest Neighbors, Logistic Regression, and Gaussian Naive Bayes were used due to their ability to handle complex data and deliver accurate results.

Our models' performance was evaluated using metrics such as accuracy, precision, recall, F1 score, as well as training and prediction times. The Random Forest model stood out, offering superior precision, fast training and prediction times, and a low false alarm rate, making it ideal for real-time detection systems. When compared to other research, Random Forest consistently outperformed other models across almost all evaluation criteria.

Future research in machine learning for intrusion detection could focus on new techniques, real-time online learning adaptation, and behavioral analysis for anomaly detection. Additional areas of exploration include improving model robustness against adversarial attacks, promoting data sharing among researchers, and integrating active cyber defense into intrusion detection systems.

REFERENCES

[1] A. Ahmim, L. Maglaras, M. A. Ferrag, M. Derdour and H. Janicke, "A Novel Hierarchical Intrusion Detection System Based on Decision Tree and Rules-Based Models," in 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS), 2019.

[2] I. H. W. E. Frank, "Reduced-error pruning with significancetests," 1999.

[3] W. W. Cohen, "Fast Effective Rule Induction," in Machine Learning Proceedings 1995, Elsevier, 1995, p. 115–123.

[4] M. N. Adnan and M. Z. Islam, "Forest PA : Constructing a decision forest by penalizing attributes used in previous trees," Expert Systems with Applications, vol. 89, p. 389–403, December 2017.

[5] I. Sharafaldin, A. Habibi Lashkari and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in Proceedings of the 4th International Conference on Information Systems Security and Privacy, 2018.

[6] M. A. Aydın, A. H. Zaim and K. G. Ceylan, "A hybrid intrusion detection system design for computer network security," Computers & Electrical Engineering, vol. 35, p. 517–526, May 2009.

[7] IDEVAL, MIT Lincoln Laboratories network traffic data, 1999, https://archive.ll.mit.edu/ideval/data/index.html

[8] Y. Y. Chung and N. Wahid, "A hybrid network intrusion detection system using simplified swarm optimization (SSO)," Applied Soft Computing, vol. 12, p. 3014–3022, September 2012.

[9] K. D. D. Cup, KDD Cup 1999 Data, 1999, https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[10] G. Wang, J. Hao, J. Ma and L. Huang, "A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering," Expert Systems with Applications, vol. 37, p. 6225–6232, September 2010.

[11] G. Kim, S. Lee and S. Kim, "A novel hybrid intrusion detection method integrating anomaly detection with misuse detection," Expert Systems with Applications, vol. 41, p. 1690–1700, March 2014.

[12] S. L. Salzberg, "C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993," Machine Learning, vol. 16, p. 235–240, September 1994.

[13] NSL-KDD, A modified version of KDD'99 data set, https://www.kaggle.com/datasets/hassan06/nslkdd

[14] A. D. F. A. at UNSW Canberra, The UNSW-NB15 Dataset, https://research.unsw.edu.au/projects/unsw-nb15-dataset

[15] J. Hühn and E. Hüllermeier, "FURIA: an algorithm for unordered fuzzy rule induction," Data Mining and Knowledge Discovery, vol. 19, p. 293–319, April 2009.

[16] M. De Gregorio and M. Giordano, "An experimental evaluation of weightless neural networks for multi-class classification," Applied Soft Computing, vol. 72, p. 338–354, November 2018.

[17] I. Ibarguren, J. M. Pérez, J. Muguerza, I. Gurrutxaga and O. Arbelaitz, "Coverage-based resampling: Building robust consolidated decision trees," Knowledge-Based Systems, vol. 79, p. 51–67, May 2015.

[18] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," ACM Transactions on Intelligent Systems and Technology, vol. 2, p. 1–27, April 2011.