Detecting Attacks on Industrial Internet of Things (IIoT) Networks Using the Random Forest Algorithm with Feature Selection and Parameter Optimization Techniques

1st Diaz Sabat Dolly Silitonga
Department of Computer Science
Politeknik Negeri Batam
Batam City, Indonesia
diaz.4332201068@students.polibatam.ac.id

2nd Andy Triwinarko
Department of Computer Science
Politeknik Negeri Batam
Batam City, Indonesia
andytriwinarko@polibatam.ac.id

***Abstract*—The increasing adoption of interconnected industrial devices has created new cybersecurity challenges in Industrial Internet of Things (IIoT) environments. As communication among sensors, controllers, and monitoring systems becomes more intensive, industrial networks become more vulnerable to cyberattacks that may disrupt operational processes. Existing studies have primarily focused on limited attack categories or conventional IoT environments, creating a need for more effective multi-class attack detection models for complex IIoT networks. To address this issue, this research develops a multiclass intrusion detection framework using Random Forest and Decision Tree classifiers enhanced through feature reduction and hyperparameter optimization techniques. Experiments were conducted using the CICIIoT2025 dataset following data preprocessing, balancing, normalization, and feature selection procedures. The optimized Random Forest model achieved the strongest overall performance across all evaluation metrics. Statistical validation using the Wilcoxon Signed-Rank Test confirmed that the observed improvements were significant. These findings demonstrate that combining feature selection with parameter optimization can strengthen attack detection capabilities in IIoT networks.**

Keywords—Industrial Internet of Things, Attack Detection, Random Forest, Feature Selection, Parameter Optimization

I. INTRODUCTION

The integration of connected sensors, industrial devices, and communication infrastructures has transformed the way modern industrial systems operate. Through continuous data exchange and automated monitoring capabilities, IIoT technologies support more efficient production and decision-making processes. The implementation of IIoT offers various benefits, such as improved operational efficiency, productivity, system reliability, and reduced operational costs through the automation of industrial processes [1].

However, the increasing number of interconnected devices in IIoT environments also expands the attack surface that can be exploited by malicious actors. Various cybersecurity threats pose increasingly serious challenges to the operational continuity of industrial systems. Unlike conventional IoT environments, attacks on the IIoT environment not only impact data confidentiality and integrity but can also disrupt production processes, degrade service quality, and result in significant economic losses [2] [1].

Among the machine learning methods commonly applied to intrusion detection, Random Forest is widely recognized for its ability to combine multiple decision trees into a single ensemble model, thereby improving prediction stability and robustness [3]. This approach improves the model's generalization ability, reduces the risk of overfitting, and yields more stable classification performance compared to using a single Decision Tree [4], [5]. However, the performance of Random Forest is highly influenced by the quality of the features used and the configuration of the model parameters applied.

The large number of features in network traffic datasets can lead to increased computational complexity and the emergence of redundant features. Therefore, feature selection techniques are needed to retain relevant features and reduce information redundancy. To reduce feature redundancy before model training, a correlation-based filtering strategy was applied. Attributes exhibiting highly similar information were removed so that the resulting feature set remained compact while preserving important characteristics of the original dataset [6]. Additionally, parameter optimization is performed using GridSearchCV to obtain the best combination of hyperparameters so that model performance can be optimally improved [7].

Previous research conducted by Belinda Panjaitan and Hamdani Arif applied a Decision Tree algorithm with feature selection and parameter optimization techniques to detect attacks on the CICIoT2023 dataset. The results show that the resulting model achieved high accuracy, recall, and F1-score values in the classification process[8]. However, that study focused only on the classification of two data classes, namely benign and Mirai, and thus did not represent the complexity of the various types of attacks commonly found in modern Industrial Internet of Things environments. Additionally, the CICIoT2023 dataset was used to represent a general IoT environment, while current industrial infrastructure developments demand detection models capable of operating in IIoT environments with more complex network traffic characteristics and security threats [2]. Therefore, there remains a research gap in developing attack detection models capable of multiclass attack classification in IIoT

environments while effectively addressing feature redundancy and optimizing model parameters.

To provide a meaningful benchmark, Decision Tree was employed as the baseline classifier due to its simplicity, interpretability, and extensive use in cybersecurity and intrusion detection studies. Random Forest was selected as the comparison model because it extends the Decision Tree concept through an ensemble mechanism that aggregates predictions from multiple trees. Evaluating both models under the same experimental conditions makes it possible to assess the contribution of feature selection and hyperparameter optimization while minimizing the influence of architectural differences between classifiers.

The effectiveness of feature reduction and hyperparameter tuning was investigated by applying both techniques to Decision Tree and Random Forest classifiers within a multiclass IIoT intrusion detection framework. Model performance was evaluated using commonly adopted classification metrics.

II. MATERIALS AND METHODS

A. *CICIIoT2025 Research Dataset*

The experiments conducted in this study utilized the CICIIoT2025 dataset released by the Canadian Institute for Cybersecurity (CIC). This dataset contains network traffic collected from IoT and IIoT environments and includes both normal activities and various cyberattack scenarios. Its design reflects realistic attack conditions involving multiple device types and communication protocols commonly found in industrial ecosystems [9].

Table 1. Number of Attacks in the CICIIoT2025 Dataset

Label	Amount
Benign	400.672
Recon	105.848
DoS	57.736
DDoS	56.692
MITM	25.490
Malware (Mirai)	24.177
Web	9.040
BruteForce	6.016
Total	**685.671**

This study utilizes all classes in the CICIIoT2025 dataset, including DDoS, DoS, Recon, Web, Brute Force, MITM, as well as malware (Mirai), and Benign, thereby providing a more comprehensive representation of various attack patterns compared to previous studies that focused on only one type of attack [8].

B. *Research Workflow*

This study uses an experimental method to develop a machine learning model for detecting attacks on Industrial Internet of Things (IIoT) networks. The research workflow is shown in Figure 3.1.

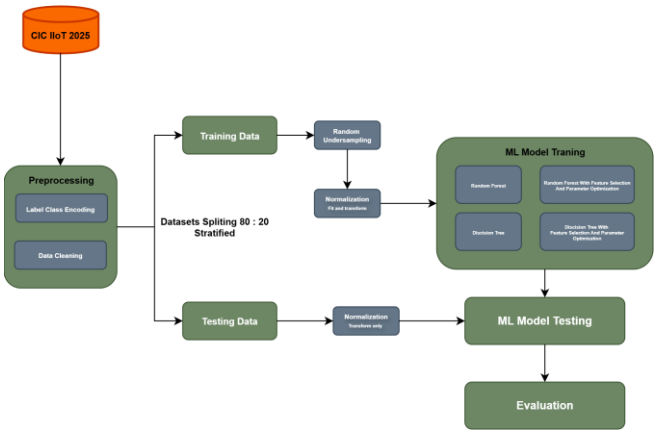


Figure 1. Research Workflow

The experimental workflow began with selecting the CICIIoT2025 dataset as the primary source of network traffic data. The class labels were subsequently transformed into numerical representations, followed by data cleaning procedures to eliminate duplicate records and address missing values. The processed dataset was then partitioned into training and testing subsets. Additional preprocessing stages included class balancing and feature scaling before model construction, optimization, and evaluation were performed.

C. *Research Scenario*

Following the workflow illustrated in Figure 3.1, two experiments in this study are divided into the following two scenarios:

1) The first scenario (baseline) involves developing an attack detection model without applying feature selection or parameter optimization.
2) The Scenario 2 (Optimized Model). The second experiment incorporated correlation-based feature selection together with GridSearchCV hyperparameter optimization before model training. The obtained results were then compared with those of the baseline experiment to determine the impact of these optimization techniques on predictive performance and computational efficiency.

D. *Datasets Splitting*

To obtain an unbiased assessment of model performance, the dataset was divided into training and testing subsets using an 80:20 ratio. Stratified sampling was employed during this process to ensure that the proportion of every attack category remained consistent in both subsets. This partitioning strategy provides sufficient data for model learning while reserving an adequate amount of unseen data for performance evaluation [10]. Furthermore, preserving the original class distribution helps reduce sampling bias and produces evaluation results that better reflect the characteristics of the complete dataset [11].

E. *Random UnderSampling*

The class frequencies in the dataset were not evenly distributed, resulting in a class imbalance problem. To address this issue, Random Under Sampling (RUS) was applied by reducing the number of instances belonging to majority classes. This approach decreases dataset size and computational requirements, although some information from the majority class may be discarded during the process [12], [13], [14].

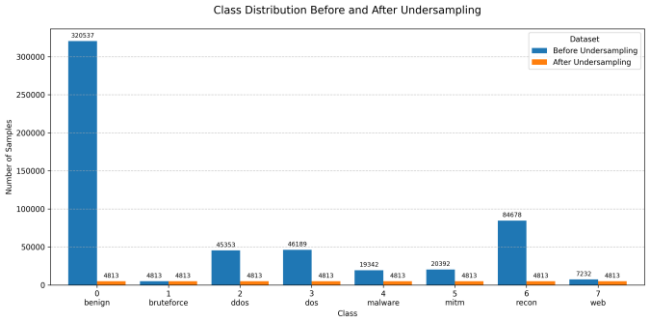


Figure 2. Class Distribution Before and After Undersampling

In this study, the Random Under-Sampling (RUS) process was performed after the dataset was split into a training set and a testing set. The undersampling technique was applied only to the training data as shown in Figure 2, while the testing data retained its original class distribution. This approach was chosen to prevent data leakage and ensure that the model evaluation was conducted on data that accurately represented real-world conditions.

Following the recommendation of Toleva (2026), Random Under Sampling was applied only to the training set after the train-test split. This strategy prevents data leakage and preserves the original class distribution in the testing set, resulting in a more reliable evaluation process [15]. Furthermore, Imbalanced-Learn notes that performing resampling before dataset splitting may lead to overly optimistic performance estimates because the testing data no longer reflects the original data distribution [16].

F. *Min Max Scaler - Normalization*

Feature normalization was carried out using the Min-Max scaling technique, which transforms numerical attributes into values ranging from 0 to 1. This preprocessing step minimizes the influence of variables with larger numerical ranges, allowing all features to contribute more equally during the learning process [17].

To avoid information leakage, the normalization parameters were estimated exclusively from the training dataset [18]. Specifically, the fit_transform() function was applied to the training data to compute the minimum and maximum values for each feature while simultaneously scaling the data. The resulting parameters were then reused through the transform() function to normalize the testing dataset without recalculating them. This procedure follows the Scikit-Learn recommendation that the fitting stage should only be performed on the training dataset [19].

Applying identical scaling parameters to both training and testing data preserves the validity of the evaluation process and prevents information from the testing set from influencing model development [20].

G. *Parameter Configuration*

The parameter optimization technique used in this study is GridSearchCV. The parameters optimized in this study include [21]:

1) The n_estimators parameter determines how many decision trees participate in the ensemble learning process. A larger number of trees generally improves prediction stability because the final decision is obtained by aggregating predictions from multiple trees. However, increasing this value also increases the computational cost during model training.

2) The criterion parameter specifies the rule used to select the best split at each node of a decision tree. It evaluates how effectively a candidate split separates the training data into more homogeneous groups, allowing the algorithm to construct a more informative tree structure.

3) The max_depth parameter restricts the growth of individual decision trees by defining the maximum allowable depth. Limiting tree depth prevents the model from becoming excessively complex, thereby reducing the likelihood of fitting noise contained in the training data.

4) This parameter controls when a node is eligible to be divided into additional branches. A split is performed only when the number of observations within the node satisfies the specified minimum requirement, helping to avoid unnecessary tree expansion.

5) The min_samples_leaf parameter determines the smallest number of training instances that must remain in every terminal node. Maintaining a sufficient number of samples in each leaf helps produce more stable decision boundaries and improves the model's ability to generalize to unseen data.

The hyperparameters included in the optimization process were selected because of their influence on model performance and generalization capability. Previous studies have shown that parameters can substantially affect classification outcomes. Therefore, several candidate values were evaluated to identify a configuration that provides the best performance for the intrusion detection task [5], [8], [22], [23]

Table 2. Overview Of Hyperparameter Tuning Processes

Model	Parameter	Values Explored	Best Value
	Number of estimators	10, 100, 240, 510	510
	Criterion	gini, entropy	gini
	Maximum depth	None, 20, 24	20

Random Forest	Minimum Samples Split	2, 4, 5	2
	Minimum Samples Leaf	1, 2	1
Decision Tree	Criterion	gini, entropy	gini
	Maximum depth	None, 20, 24	20
	Minimum Samples Split	2, 4, 5	2
	Minimum Samples Leaf	1, 2	1

Adjusting parameters via GridSearchCV has been shown to improve model performance compared to using default parameters, as well as enhance the model's ability to recognize complex patterns and reduce the risk of overfitting in Random Forest-based intrusion detection systems [5].

H. *Filter Correlation - Feature Selection*

A correlation-based feature selection strategy was applied before model training to reduce redundant information among predictor variables. This method analyzes the relationships between features and removes attributes that exhibit very high correlation, thereby producing a more compact feature set while preserving relevant information for classification. Correlation coefficients range from −1 to +1, where values closer to either extreme indicate stronger linear relationships between variables [24].

The objective of correlation-based feature selection is to eliminate redundant attributes that provide overlapping information. By retaining only representative features, the dimensionality of the dataset can be reduced, leading to lower computational requirements and potentially improved classification efficiency [25].

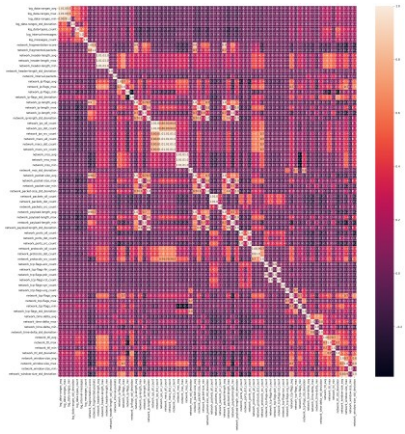


Figure 3. Heatmap Matrix Correlation

Table 3. Features removed above the 0.9 threshold

Removed features	
log_data-ranges_max	log_data-ranges_min
network_header-length_max	network_header-length_min
network_ips_dst_count	network_macs_all_count
network_macs_dst_count	network_macs_src_count
network_mss_max	network_mss_min
network_packet-size_avg	network_packet-size_max
network_packet-size_std_deviation	network_packets_dst_count
network_payload-length_avg	network_payload-length_max
network_payload-length_min	network_payload-length_std_deviation
network_ports_src_count	network_protocols_dst_count
network_tcp-flags-psh_count	network_tcp-flags-rst_count

Table 4. Number of Features Before and After Feature Selection

Number of Features	After Feature Selection
71	49

A correlation threshold of 0.90 was adopted to identify highly correlated feature pairs. When the absolute correlation coefficient between two attributes exceeded this threshold, one of the attributes was removed to minimize redundancy. This strategy helps simplify the feature space while preserving the most informative characteristics of the dataset. The selection of a threshold of 0.90 is based on the study by Koukaras and Tjortjis, who applied the Correlation Filter using a value of $\alpha = 0.90$ to identify highly correlated feature groups and remove redundant features [26]. This approach is also supported by other studies stating that correlations above 0.90 have the potential to cause multicollinearity and that removing one of the highly correlated features can reduce the feature dimension without significantly reducing information [26].

I. *Statistical Significance Test*

A Wilcoxon Signed-Rank Test was employed to examine whether the performance differences between the baseline models and the optimized models were statistically meaningful. This non-parametric test was selected because it is suitable for comparing paired observations without requiring the assumption of normal data distribution. Statistical significance was evaluated using a significance level of 0.05. A p-value below this threshold indicates sufficient evidence to reject the null hypothesis, whereas a larger p-value suggests that the observed differences may have occurred by chance.[27].

J. *Evaluation Metrics*

In general, the evaluation of classification models in machine learning can utilize a confusion matrix, as follows [28]:

1) Accuracy

Instead of evaluating individual classes separately, accuracy summarizes the overall classification

capability of a model. It is obtained by comparing the total number of correct predictions with the total number of evaluated observations. Larger accuracy values indicate that the classifier makes fewer prediction errors across the dataset.

$$\boldsymbol{Accuracy} = \frac{(\mathbf{TP} + \mathbf{TN})}{(\mathbf{TP} + \mathbf{FP} + \mathbf{TN} + \mathbf{FN})}$$

2) Precision

Precision focuses on the correctness of positive predictions generated by the classifier. This metric indicates how frequently samples predicted as positive truly belong to the positive class, making it useful for assessing the occurrence of false positive errors.

$$\boldsymbol{Precision} = \frac{\mathbf{TP}}{(\mathbf{TP} + \mathbf{FP})}$$

3) Recall

Recall evaluates how effectively a model detects all relevant positive instances. A classifier with high recall successfully identifies most positive samples while minimizing the number of missed detections.

$$\boldsymbol{Recall} = \frac{\mathbf{TP}}{(\mathbf{TP} + \mathbf{FN})}$$

4) F1-Score

Neither precision nor recall alone is sufficient to describe classification performance. Therefore, the F1-score combines both measures into a single indicator using their harmonic mean, providing a more comprehensive evaluation when both false positives and false negatives are important.

$$\boldsymbol{F1 - Score} = \frac{(\mathbf{2} \times \mathbf{precision} \times \mathbf{recall})}{(\mathbf{precision} + \mathbf{recall})}$$

5) ROC Curve

The ROC curve illustrates the classification behavior of a model under different decision thresholds. By comparing the true positive rate against the false positive rate, this curve helps determine how well the classifier distinguishes between different classes.

III. Results and Discussion

The experimental findings presented in Table 5 indicate that both Random Forest configurations achieved strong classification performance across the evaluated metrics. The optimized Random Forest model produced the highest scores, demonstrating the positive impact of feature selection and hyperparameter tuning.

Relative to the baseline configuration, the optimized model exhibited consistent improvements across all evaluation measures, suggesting that the proposed optimization strategy contributed positively to attack detection performance.

These results indicate that the combination of the Correlation Filter and GridSearchCV is capable of improving the performance of both algorithms in classifying attacks in an IIoT environment.

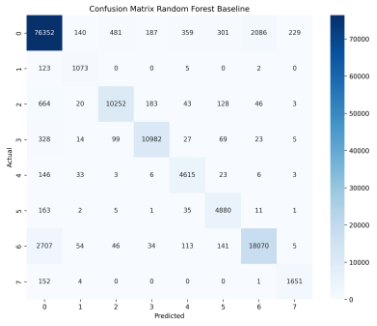
Table 5. Performance Measures of Various Classifiers

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Random Forest	93.25	93.27	93.25	93.23
Random Forest With Feature Selection and Parameter Optimization Techniques	94.08	94.12	94.08	94.05
Decision Tree	91.36	91.49	91.36	91.39
Decision Tree With Feature Selection and Parameter Optimization Techniques	92.12	92.22	92.12	92.14

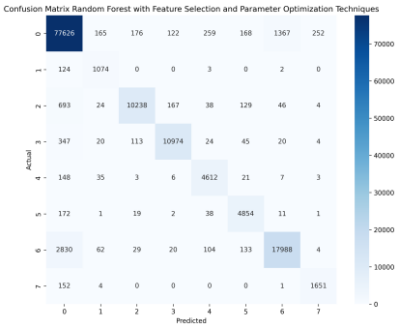
Table 6. Performance Measures of Multi Classifiers

Model	Label	Precision (%)	Recall (%)	F1-Score (%)	Support
Random Forest	Benign	94	95	94	80135
	Brute Force	80	89	84	1203
	DDoS	94	90	92	11339
	DoS	96	95	95	11547
	Malware	88	95	92	4835
	MITM	88	95	91	5098

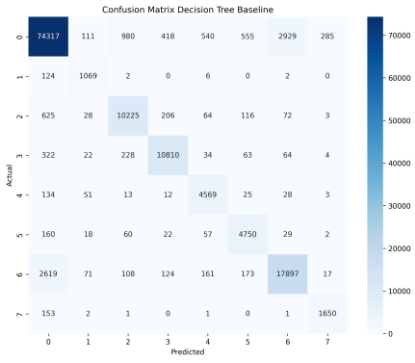
	Recon	89	85	87	21170
	Web	87	91	89	1808
Random Forest With Feature Selection and Parameter Optimization Techniques	Benign	94	96	95	80135
	Brute Force	77	89	82	1203
	DDoS	96	90	93	11339
	DoS	97	95	96	11547
	Malware	90	95	93	4835
	MITM	90	95	92	5098
	Recon	92	84	88	21170
	Web	86	91	88	1808
Decision Tree	Benign	94	92	93	80135
	Brute Force	77	88	83	1203
	DDoS	88	90	89	11339
	DoS	93	93	93	11547
	Malware	84	94	89	4835
	MITM	83	93	88	5098
	Recon	85	84	84	21170
	Web	84	91	87	1808
Decision Tree With Feature Selection and Parameter Optimization Techniques	Benign	94	93	94	80135
	Brute Force	78	88	83	1203
	DDoS	90	89	90	11339
	DoS	93	93	93	11547
	Malware	83	93	88	4835
	MITM	89	93	91	5098
	Recon	86	84	85	21170
	Web	80	91	85	1808



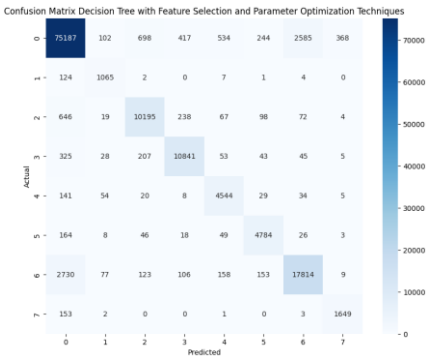
(a) Random Forest Baseline



(b) Random Forest Optimized



(c) Discision Tree



(d) Discision Tree Optimized

Figure 4. Confusion Matrix of Various Classifiers

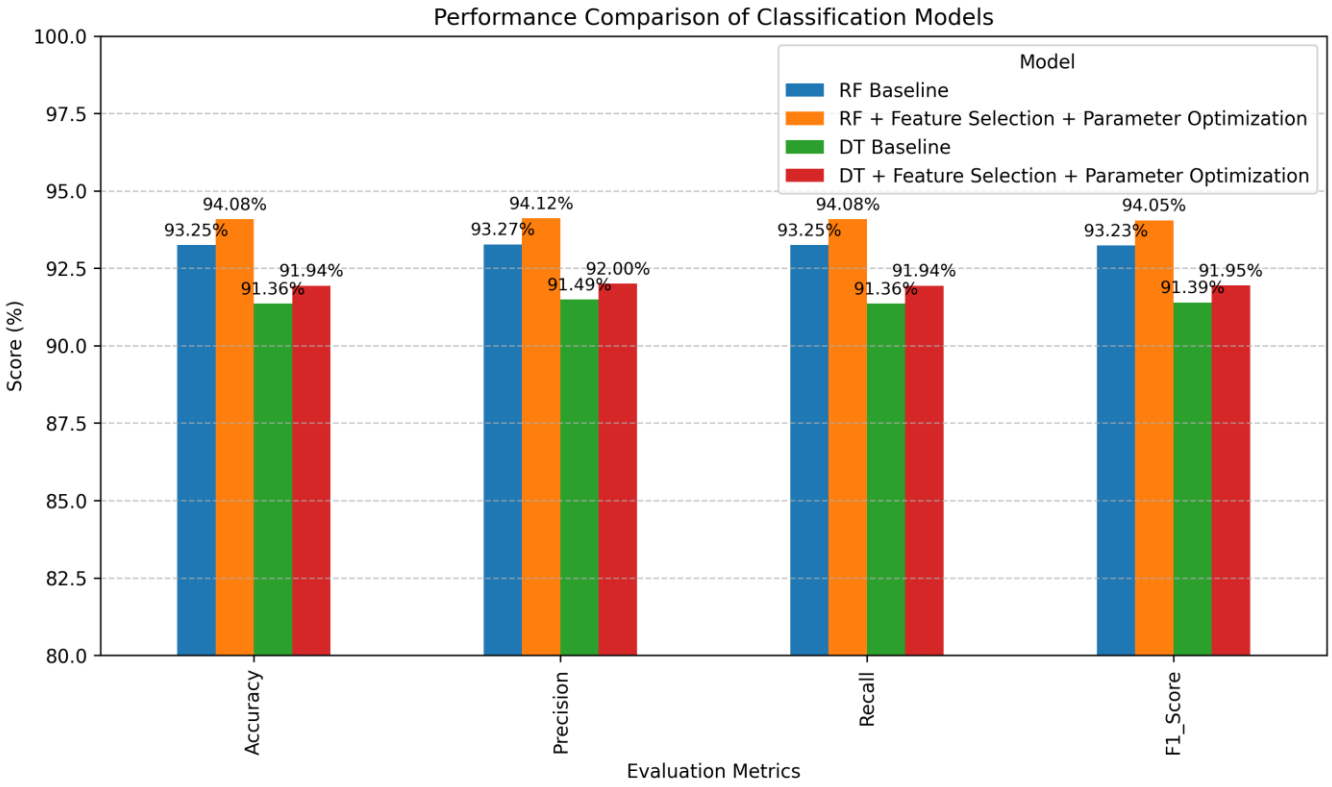
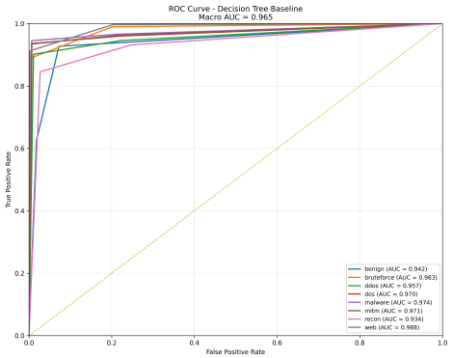
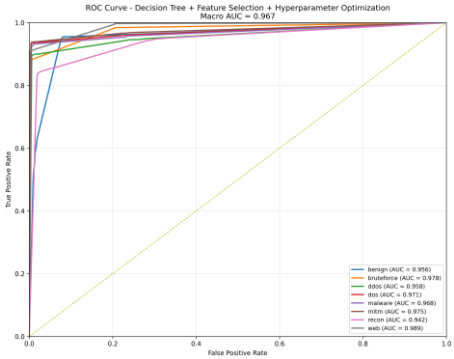


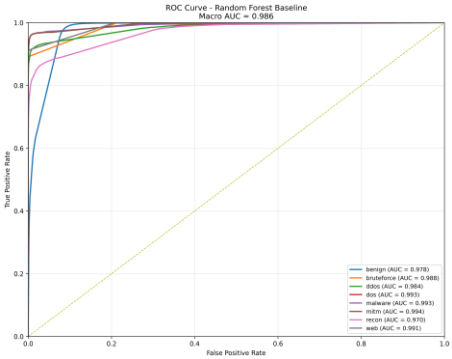
Figure 5. Performance charts by model



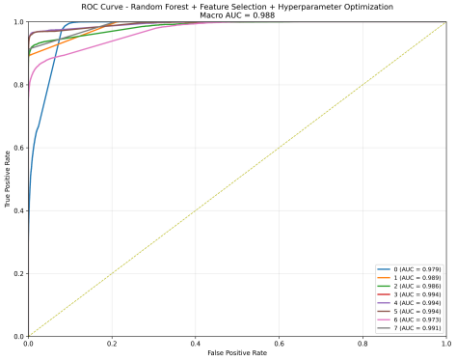
(a) Decision Tree Baseline



(b) Decision Tree Optimized

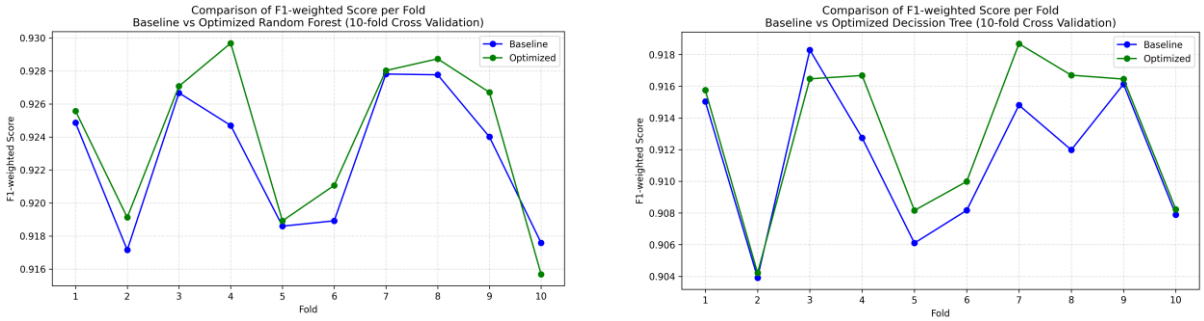


(c) Random Forest Baseline



(b) Random Forest Optimized

Figure 6. Multiclass ROC Curve by Model



(a) Statistical Test of the Baseline and Optimized Random Forest and Decision Tree

```
from scipy.stats import wilcoxon
import numpy as np

stat, p_value = wilcoxon(
    baseline_scores,
    optimized_scores,
    alternative="two-sided"
)

print("Random Forest")

print(f"Wilcoxon Statistic : {stat}")
print(f"P-value            : {p_value:.6f}")

alpha = 0.05

if p_value < alpha:
    print("\nDecision   : Reject H0")
    print("Conclusion : There is a statistically significant difference between the two models.")
else:
    print("\nDecision   : Fail to Reject H0")
    print("Conclusion : There is no statistically significant difference between the two models.")

mean_baseline = np.mean(baseline_scores)
mean_optimized = np.mean(optimized_scores)

print(f"\nMean Baseline  : {mean_baseline:.6f}")
print(f"Mean Optimized : {mean_optimized:.6f}")

if mean_optimized > mean_baseline:
    print("The optimized model demonstrates better performance.")
elif mean_optimized < mean_baseline:
    print("The baseline model demonstrates better performance.")
else:
    print("Both models demonstrate equivalent performance.")
```

```
Random Forest
Wilcoxon Statistic : 6.0
P-value            : 0.027344

Decision   : Reject H0
Conclusion : There is a statistically significant difference between the two models.

Mean Baseline  : 0.922804
Mean Optimized : 0.924056
The optimized model demonstrates better performance.
```

(b) P-Value Random Forest

```
from scipy.stats import wilcoxon
import numpy as np

stat, p_value = wilcoxon(
    baseline_scores,
    optimized_scores,
    alternative="two-sided"
)

print("Decision Tree")

print(f"Wilcoxon Statistic : {stat}")
print(f"P-value            : {p_value:.6f}")

alpha = 0.05

if p_value < alpha:
    print("\nDecision   : Reject H0")
    print("Conclusion : There is a statistically significant difference between the two models.")
else:
    print("\nDecision   : Fail to Reject H0")
    print("Conclusion : There is no statistically significant difference between the two models.")

mean_baseline = np.mean(baseline_scores)
mean_optimized = np.mean(optimized_scores)

print(f"\nMean Baseline  : {mean_baseline:.6f}")
print(f"Mean Optimized : {mean_optimized:.6f}")

if mean_optimized > mean_baseline:
    print("The optimized model demonstrates better performance.")
elif mean_optimized < mean_baseline:
    print("The baseline model demonstrates better performance.")
else:
    print("Both models demonstrate equivalent performance.")
```

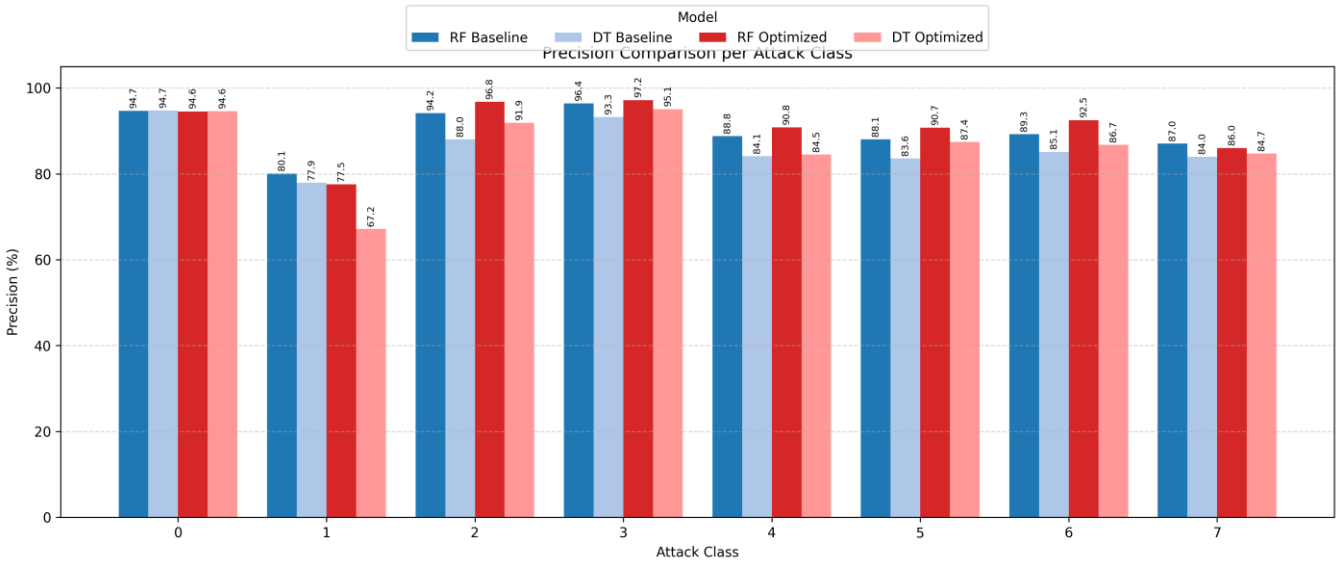
```
Decision Tree
Wilcoxon Statistic : 5.0
P-value            : 0.019531

Decision   : Reject H0
Conclusion : There is a statistically significant difference between the two models.

Mean Baseline  : 0.911505
Mean Optimized : 0.913132
The optimized model demonstrates better performance.
```

(c) P-Value Descision Tree

Figure 7. Statistical Test and P- Value of the Baseline Model and Optimized Model



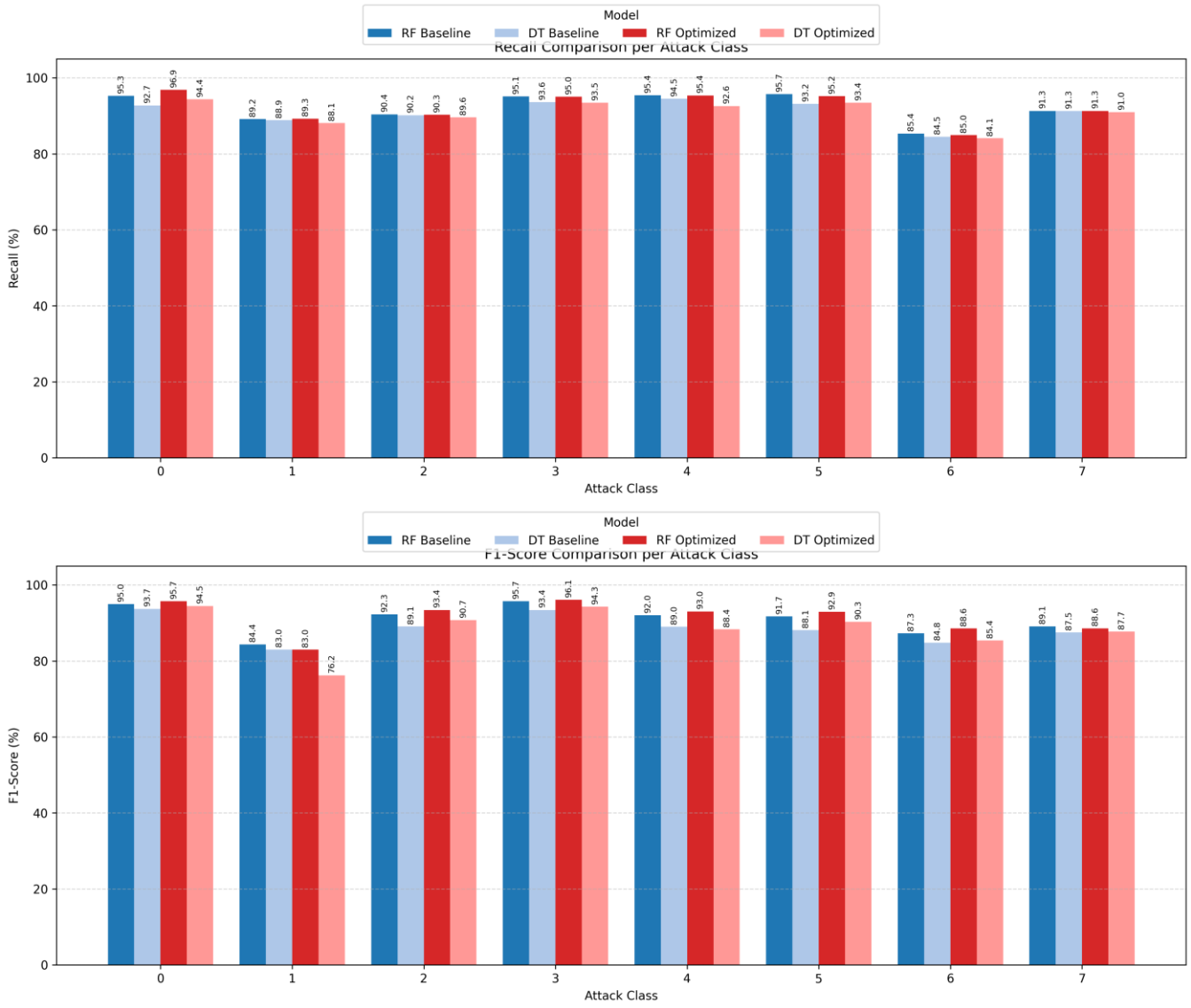


Figure 8. Graphs of Various Multi-Class Classifications by Model

Figure 4 illustrates the confusion matrices generated by each classification model, providing a detailed view of prediction outcomes across the evaluated attack categories. As shown in Figure 4(a), The baseline Random Forest classifier successfully identified most observations in each traffic category, indicating strong overall classification capability. However, several misclassifications were still observed, particularly in the Recon, MITM, and Brute Force categories. After applying Correlation-Based Feature Selection and GridSearchCV hyperparameter optimization, as illustrated in Figure 4(b), the number of predictions located on the main diagonal increased, indicating a higher number of correctly classified instances and an overall improvement in classification performance.

The detailed class-wise performance is reported in Table 6, while Figure 8 illustrates the corresponding comparison across all traffic categories. In general, all evaluated models achieved strong performance on the Benign and DoS classes, with F1-scores exceeding 91%. This result indicates that the traffic patterns associated with these classes were relatively distinguishable from those of other categories. In contrast, the Brute Force class consistently produced the lowest F1-score among all attack categories. This may be attributed to the relatively smaller number of samples available for this class as well as the similarity of its traffic characteristics to other categories, making it more difficult for the models to learn discriminative patterns and accurately distinguish Brute Force attacks from other network activities.

Among all evaluated models, the Random Forest model combined with Feature Selection and Hyperparameter Optimization achieved the highest F1-scores across most attack categories, particularly for the DDoS, Recon, and Benign classes. The observed performance gain may be explained by the ensemble-based structure of Random Forest, which combines predictions from multiple trees to improve robustness and generalization. Furthermore, the application of Correlation-Based Feature Selection reduced feature redundancy by removing highly correlated features, enabling the model to focus on more informative attributes. The GridSearchCV optimization process also contributed to performance improvement by identifying a more suitable hyperparameter configuration for the classification task.

The multiclass ROC analysis shown in Figure 6 provides additional evidence of the discriminative capability of each

classification model. In general, all ROC curves were located close to the upper-left corner of the graph, indicating strong classification capability and high discriminative power. The Random Forest model with Feature Selection and Hyperparameter Optimization achieved the highest Area Under the Curve (AUC) values among the evaluated models, demonstrating its superior ability to separate attack categories from benign traffic and from one another. These findings are consistent with the accuracy, precision, recall, and F1-score results presented in Table 5.

To further validate the effectiveness of the optimization process in Figure 7, a Wilcoxon Signed-Rank Test was conducted to determine whether the observed performance differences between baseline and optimized models were statistically significant. For the Random Forest modelThe statistical analysis yielded a p-value below the predefined significance threshold of 0.05, demonstrating that the observed performance improvement was unlikely to have occurred by chance. The optimized model also achieved a higher mean performance score (0.924056) than the baseline model (0.922804), further confirming the effectiveness of the proposed optimization approach.

Similarly, the Wilcoxon Signed-Rank Test conducted on the Decision Tree models yielded a p-value of 0.019531, which was also below the significance threshold of 0.05. As a result, the null hypothesis was rejected, indicating a statistically significant difference between the baseline and optimized Decision Tree models. The optimized Decision Tree model achieved a higher mean performance score (0.913132) compared to the baseline model (0.911505). These findings suggest that the improvements observed in both algorithms were not caused by random variation but were attributable to the application of Correlation-Based Feature Selection and GridSearchCV hyperparameter optimization.

Overall, the experimental results demonstrate that the proposed approach effectively improves multiclass attack detection performance in Industrial Internet of Things environments. The combination of Correlation-Based Feature Selection and hyperparameter optimization contributed to reducing feature redundancy while enhancing classification capability. Furthermore, the superior performance achieved by Random Forest indicates that ensemble learning provides additional benefits for handling the complexity of IIoT network traffic. These findings address the limitations identified in previous studies that focused primarily on binary attack classification and confirm the effectiveness of the proposed approach for multiclass intrusion detection in IIoT environments.

IV. Conclusion

This research investigated the impact of combining Correlation-Based Feature Selection and GridSearchCV optimization on Decision Tree and Random Forest models for multiclass intrusion detection in IIoT environments. Using the CICIIoT2025 dataset, the feature selection process successfully reduced the number of features from 71 to 49, thereby decreasing data dimensionality while preserving classification performance.

Experimental results demonstrated that the optimized Random Forest configuration delivered the best overall classification performance among all evaluated models.

Furthermore, the multiclass classification results demonstrate that the proposed approach can effectively classify benign traffic and multiple attack categories in IIoT environments. Although some misclassification remains among classes with similar traffic characteristics, particularly Brute Force attacks, the overall findings indicate that Correlation Based Feature Selection and Parameter Optimization Based GridSearchCV effectively address feature redundancy and parameter optimization challenges while improving attack detection performance.

Overall, the findings indicate that integrating feature selection with hyperparameter optimization can substantially enhance the effectiveness of Random Forest for intrusion detection tasks in Industrial Internet of Things networks.

For future research, further comparisons with advanced machine learning and deep learning algorithms, such as XGBoost, LightGBM, CatBoost, and Neural Networks, are recommended. In addition, future studies may explore various feature selection, feature engineering, data balancing, and hyperparameter optimization techniques to further improve classification accuracy and model robustness. Furthermore, implementing the proposed model in a real-time intrusion detection environment would provide a more comprehensive evaluation of its practical applicability in industrial networks.

Acknowledgment

The authors would like to thank Politeknik Negeri Batam for providing support and facilities that contributed to the completion of this research. The authors also would like to thank Mr. Andy Triwinarko for his valuable guidance, suggestions, and support during the completion of this research.

References

[1] A. Firouzi, S. Dadkhah, S. A. Maret, and A. A. Ghorbani, “DataSense: A Real-Time Sensor-Based Benchmark Dataset for Attack Analysis in IIoT with Multi-Objective Feature Selection,” *Electronics*, vol. 14, no. 20, p. 4095, Oct. 2025, doi: 10.3390/electronics14204095.

[2] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, “CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment,” *Sensors*, vol. 23, no. 13, p. 5941, Jun. 2023, doi: 10.3390/s23135941.

[3] L. Breiman, “Random Forests,” 2001.

[4] M. W. Nugroho, “Analisis Performa Algoritma Random Forest dalam Mengatasi Overfitting pada Model Prediksi,” *jtik*, vol. 9, no. 4, pp. 1562–1571, Oct. 2025, doi: 10.35870/jtik.v9i4.4236.

[5] N. Zhu, C. Zhu, L. Zhou, Y. Zhu, and X. Zhang, “Optimization of the Random Forest Hyperparameters for Power Industrial Control Systems Intrusion Detection Using an Improved Grid Search Algorithm,” *Applied Sciences*, vol. 12, no. 20, p. 10456, Oct. 2022, doi: 10.3390/app122010456.

[6] L. Yu and H. Liu, “Efficient Feature Selection via Analysis of Relevance and Redundancy,” 2004.

[7] I. K. A. Jayaditya, “Implementasi Random Forest pada Klasifikasi Penyakit Kardiovaskular dengan Hyperparameter Tuning Grid Search,” vol. 2, 2023.
[8] B. Panjaitan and H. Arif, “Deteksi Serangan Malware pada Jaringan Internet of Things (IoT) Menggunakan Algoritma Decision Tree dengan Teknik Seleksi Fitur dan Optimasi Parameter,” 2025.
[9] “IIoT Dataset 2025 | Datasets | Research | Canadian Institute for Cybersecurity | UNB.” Accessed: Apr. 02, 2026. [Online]. Available: https://www.unb.ca/cic/datasets/iiot-dataset-2025.html
[10] A. Gholamy, V. Kreinovich, and O. Kosheleva, “Why 70/30 or 80/20 Relation Between Training and Testing Sets: A Pedagogical Explanation”.
[11] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow*. 2022.
[12] Y. SEOK JEON and A. D. JOON LIM, “PSU: Particle Stacking Undersampling Method for Highly Imbalanced Big Data,” 2020.
[13] A. Kim and I. Jung, “Optimal selection of resampling methods for imbalanced data with high complexity,” 2023.
[14] S. Bagui and K. Li, “Resampling imbalanced data for network intrusion detection datasets,” *J Big Data*, vol. 8, no. 1, p. 6, Dec. 2021, doi: 10.1186/s40537-020-00390-x.
[15] B. Toleva, “Is There a Preferred Cross-Validation Type for Train/Test Split in Imbalanced Classification,” 2026.
[16] The imbalanced-learn developers, “9. Common pitfalls and recommended practices.” [Online]. Available: https://imbalanced-learn.org/stable/common_pitfalls.html
[17] L. B. V. de Amorim, G. D. C. Cavalcanti, and R. M. O. Cruz, “The choice of scaling technique matters for classification performance,” *Applied Soft Computing*, vol. 133, p. 109924, Jan. 2023, doi: 10.1016/j.asoc.2022.109924.
[18] A. Apicella, F. Isgrò, and R. Prevete, “Don’t push the button! Exploring data leakage risks in machine learning and transfer learning,” *Artif Intell Rev*, vol. 58, no. 11, p. 339, Aug. 2025, doi: 10.1007/s10462-025-11326-3.
[19] scikit-learn developers, “12. Common pitfalls and recommended practices.”
[20] L. Sasse *et al.*, “Overview of leakage scenarios in supervised machine learning,” *J Big Data*, vol. 12, no. 1, p. 135, May 2025, doi: 10.1186/s40537-025-01193-8.
[21] “scikit-learn, sklearn.ensemble.RandomForestClassifier.html,” scikit-learn. Accessed: Apr. 02, 2026. [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[22] B. Alabdullah and S. Sankaranarayanan, “Optimized ensemble machine learning model for cyberattack classification in industrial IoT,” *Front. Artif. Intell.*, vol. 8, p. 1685376, Jan. 2026, doi: 10.3389/frai.2025.1685376.
[23] O. E. Akinbowale, A. T. Adesina, M. F. Zerihun, and P. Mashigo, “Machine learning based approach to intrusion detection in internet of things environments,” *Front. Artif. Intell.*, vol. 9, p. 1760137, Apr. 2026, doi: 10.3389/frai.2026.1760137.
[24] A. Mangal and E. A. Holm, “A Comparative Study of Feature Selection Methods for Stress Hotspot Classification in Materials,” *Integr Mater Manuf Innov*, vol. 7, no. 3, pp. 87–95, Sep. 2018, doi: 10.1007/s40192-018-0109-8.
[25] I. Guyon and A. Elisseeff, “An Introduction to Variable and Feature Selection”.
[26] Naveed Ahmed, Md Asri Ngadi, Muhammad Siraj Rathore, and Azhar Mahmood, “PCM-RF a Hybrid Feature Selection Mechanism for Intrusion Detection System in IoT,” 2025, [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/spy2.499
[27] Frank Wilcoxon, “Individual Comparisons by Ranking Methods,” 1945.
[28] S. Sathyanarayanan and B. R. Tantri, “Confusion Matrix-Based Performance Evaluation Metrics,” *AJBR*, pp. 4023–4031, Nov. 2024, doi: 10.53555/AJBR.v27i4S.4345.