Fault Diagnosis of Marine Engine Systems Using Optimized XGBoost with Grid Search Hyperparameter Tuning

Ekananda Sulistyo Putra[1], Nailul Muna[2], Teguh Junian Kuswanto[3], Thomas Brian[4]

[1,4] Marine Electrical Engineering Department, Shipbuilding Institute of Polytechnic Surabaya, Indonesia.
[2] Electrical Engineering Department, Electronic Engineering Polytechnic Institute of Surabaya, Indonesia.
[3] Design Construction, Shipbuilding Engineering Department, Shipbuilding Institute of Polytechnic Surabaya, Indonesia.

[1]ekanandaputra@ppns.ac.id , [2]nailulmuna@pens.ac.id, [3]tjunian@ppns.ac.id , [4]thomasbrian@ppns.ac.id

***Abstract*— The reliability of the main engine is a critical factor in ensuring vessel safety and operational effectiveness, particularly in the maritime industry, which requires continuous transportation services. Conventional maintenance approaches, such as time-based maintenance and corrective maintenance, remain widely applied; however, they often fail to provide early detection of potential failures. Disturbances such as overheating, lubrication degradation, fuel contamination, and turbocharger malfunction can significantly reduce engine performance and may result in unexpected operational shutdowns. Therefore, data-driven approaches based on machine learning are needed to improve monitoring capability and prediction of engine conditions. This study develops a ship engine condition classification model using the XGBoost algorithm based on multivariate data. The model training was conducted using five hyperparameter configurations with three train–test data split schemes (0.2, 0.3, and 0.5). Experimental results demonstrate stable performance, with accuracy ranging from 0.843 to 0.850, and improved performance as the proportion of training data increased. Models incorporating regularization showed better generalization capability when trained on larger datasets. Furthermore, hyperparameter optimization using Grid Search produced the optimal configuration for the final model. The findings indicate that XGBoost effectively handles complex sensor data and has strong potential to support reliable early fault detection in ship engines.**

Keywords*— *Fault Diagnosis, Grid Search, Marine Engine Condition Monitoring, Machine Learning, XGBoost.

I. INTRODUCTION

The main engine is a crucial component of a ship's propulsion system, as it is responsible for generating the power required to propel the vessel and support overall ship operations. The reliability of the main engine is therefore a critical aspect in the maritime industry, particularly for commercial and cargo vessels, since engine failure can significantly affect navigation safety and operational continuity. Moreover, main engine failure not only causes operational delays but also increases the risk of accidents and disruptions to other onboard systems [1].

In operational practice, conventional maintenance strategies such as time-based maintenance (TBM) and corrective maintenance are still widely implemented, especially in small- and medium-scale maritime industries [2], [3]. These approaches are generally based on predetermined time intervals without considering the real-time condition of the engine. Such maintenance patterns often fail to anticipate component degradation before failure occurs, potentially leading to unplanned downtime, increased maintenance costs, and reduced operational efficiency [1], [4].

Furthermore, ship engine failures, including overheating, lubrication problems, fuel contamination, and turbocharger malfunctions, represent common issues that frequently result in decreased engine performance and force vessels to cease operations unexpectedly, ultimately increasing repair costs and safety risks [5]. The limitations of conventional maintenance approaches highlight the need for engine condition monitoring systems capable of detecting fault indications at an early stage. Data-driven approaches based on machine learning enable the analysis of operational engine data patterns to identify abnormal conditions and detect potential failures in their early phases, thereby preventing more severe breakdowns [6], [7].

The limitations of conventional maintenance approaches highlight the need for a more adaptive engine condition monitoring system based on actual operational conditions. Advances in sensor technology and data acquisition systems in marine engines have enabled continuous collection of operational data, including parameters such as temperature, pressure, vibration, and fuel consumption. However, the availability of such data has not been fully utilized to optimally support predictive maintenance decision-making. Without appropriate analytical methods, operational data merely function as historical records and are unable to provide early warning signals of potential failures [8], [9].

In this context, data-driven approaches employing machine learning methods offer a promising solution to enhance early fault detection capabilities. Machine learning algorithms are capable of identifying complex patterns and nonlinear relationships among sensor variables that are difficult to analyze using conventional techniques. By developing a machine condition classification model based on multivariate data, the system can detect abnormal indications at an early stage before failures escalate into more severe damage.

Therefore, this study focuses on the development of a ship main engine condition classification model using the Extreme Gradient Boosting (XGBoost) algorithm to support a more accurate and reliable early fault detection system under real operational conditions.

II. METHOD

The research methodology in this study is carried out through several sequential stages, beginning with data preparation, followed by modeling, and ending with implementation and analysis. The overall workflow of the proposed method is illustrated in Figure 1.

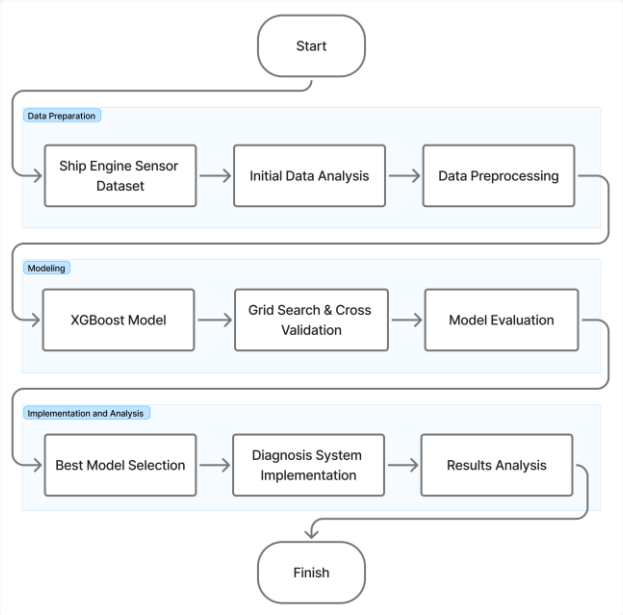

Figure 1. Research Methodology

Based in Figure 1, the research methodology is organized into three major stages, namely data preparation, modeling, and implementation with analysis, each consisting of several systematic processes to ensure the reliability and effectiveness of the proposed diagnostic approach.

A. Data Preparation

The study begins with the acquisition of ship engine sensor data obtained from a publicly available dataset on Kaggle. The dataset consists of multiple sensor-based operational parameters that represent the condition of the ship engine under both normal and faulty scenarios. These parameters typically include measurements such as Shaft RPM, Engine Load, Fuel Flow, Air Pressure, Ambient Temperature, Oil Temperature, Oil Pressure, Vibration X, Vibration Y, Vibration Z. In addition, the dataset contains labeled instances corresponding to different fault categories, enabling supervised learning for fault classification. Key characteristics of the dataset, such as the number of samples, feature dimensionality, class distribution, and potential class imbalance, are carefully examined to understand the data structure and its implications for model performance.

An initial exploratory data analysis is conducted to investigate statistical properties, including mean, standard deviation, and distribution patterns of each feature. Correlation analysis is also performed to identify relationships among variables and detect multicollinearity. Furthermore, anomaly detection techniques are applied to identify outliers or abnormal observations that may negatively impact model training. To further analyze the intrinsic structure of the data, dimensionality reduction and visualization techniques, namely Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE), are employed to analyze the underlying data structure, identify clustering patterns, and observe the separability among different fault classes.

Principal Component Analysis (PCA) is used as a linear dimensionality reduction technique to transform the original high-dimensional data into a smaller set of orthogonal components while retaining most of the variance information contained in the dataset. This method helps reveal dominant patterns, reduce redundancy among correlated features, and provide insight into the intrinsic structure of the data before model training [10]. Meanwhile, t-distributed Stochastic Neighbor Embedding (t-SNE) is applied as a nonlinear visualization technique that is particularly effective for representing complex high-dimensional data in a low-dimensional space. The method preserves local similarities between data points, making it suitable for identifying clusters and class separability that may not be visible through linear approaches [11]. The combination of PCA and t-SNE provides complementary perspectives in understanding the dataset characteristics prior to the modeling stage.

Following the exploratory analysis, a data preprocessing stage is conducted to improve data quality and ensure compatibility with machine learning algorithms. This process includes data cleaning to handle missing values and remove duplicate or inconsistent records, as well as outlier treatment to reduce the influence of abnormal data. Feature scaling, using normalization or standardization, is then applied to ensure that all variables are on a comparable scale.

Next, feature selection is performed to identify the most relevant variables for fault classification, helping to reduce dimensionality and improve model efficiency. If class imbalance is present, resampling techniques such as oversampling, undersampling, or synthetic data generation are applied to balance the dataset.

Finally, the dataset is divided into training and testing sets, and cross-validation may be used to ensure reliable model evaluation. This preparation ensures that the data is ready for the modeling stage and supports accurate performance assessment.

B. Modeling

In the modeling stage, the Extreme Gradient Boosting (XGBoost) algorithm is employed to construct the fault diagnosis model. XGBoost is an ensemble learning method based on gradient boosting decision trees that builds models in a sequential manner, where each new tree aims to correct the errors of the previous ones. The algorithm incorporates

regularization mechanisms, parallel processing, and optimized tree learning, which enable it to achieve high predictive performance while maintaining computational efficiency, particularly when dealing with nonlinear relationships and high-dimensional datasets [12].

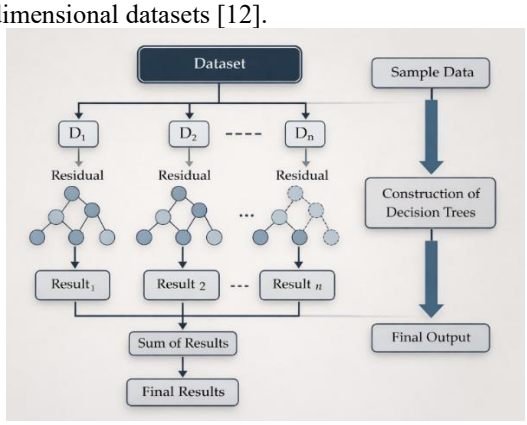

Figure 2. General Training Process of the Extreme Gradient Boosting (XGBoost) Algorithm

Figure 2 illustrates show The model is constructed in a sequential and additive manner, where each decision tree is trained to minimize the prediction error of the preceding ensemble.

The process begins with the input dataset, which serves as the training data. At the initial stage, a base prediction is generated by minimizing the specified loss function, typically resulting in a constant value such as the mean of the target variable for regression tasks or the log-odds for classification problems [13]. Based on this initial prediction, pseudo-residuals are computed using the gradient of the loss function with respect to the current predictions.

Unlike conventional gradient boosting methods that rely primarily on first-order gradients, XGBoost employs both the first-order (gradient) and second-order (Hessian) derivatives of the loss function through a second-order Taylor expansion of the objective function [14]. These derivatives guide the optimization process, enabling more accurate split finding and stable leaf weight estimation.

Subsequently, a decision tree is constructed to model the negative gradients by optimizing a regularized objective function. The resulting tree produces an intermediate output that serves as a corrective adjustment to the previous prediction. The model prediction is then updated by adding the scaled output of the newly constructed tree, controlled by a learning rate parameter. This procedure is repeated iteratively, yielding an additive ensemble of trees $f_1, f_2 \ldots . f_n$. Each tree contributes incrementally to improving the overall model performance.

To achieve optimal predictive accuracy, hyperparameter tuning is performed using the Grid Search method integrated with k-fold cross-validation. Grid Search systematically explores combinations of predefined hyperparameter values to identify the configuration that yields the best model performance based on validation results.

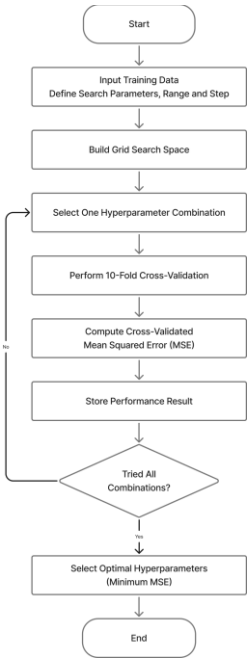

Figure 3. Grid Search Flowchart

The flowchart illustrates (Figure 3) the hyperparameter optimization procedure using a grid search strategy combined with 10-fold cross-validation. Initially, the training dataset is prepared and the search parameters—including the predefined ranges and step sizes for each hyperparameter—are specified to construct the grid search space, a systematic tuning approach widely adopted in machine learning research and shown to robustly explore parameter configurations [15]. Each possible hyperparameter combination is then systematically evaluated through 10-fold cross-validation, in which the dataset is partitioned into ten subsets and the model is iteratively trained and validated to obtain a reliable performance estimate and reduce bias in performance estimation. For every configuration, the cross-validated Mean Squared Error (MSE) is calculated and recorded. This exhaustive evaluation continues until all combinations within the search space have been assessed. The optimal hyperparameters are subsequently selected based on the minimum MSE value. By integrating exhaustive search with cross-validation, this approach ensures a robust, unbiased, and reproducible model selection process while mitigating overfitting risk and improving generalization performance, consistent with findings on rigorous hyperparameter optimization and cross-validation strategies applied to predictive modeling in healthcare domains [16].

C. Implementation and Analysis

Once the optimal model configuration is obtained, the best-performing model is selected and implemented into a diagnostic system framework to simulate real-world application scenarios in ship engine monitoring. This implementation aims to assess the practical feasibility of the proposed method in detecting and classifying engine faults under operational conditions. The results obtained from the implementation are then analyzed to evaluate model reliability, robustness, and effectiveness. The analysis also includes

interpretation of model behavior, identification of strengths and limitations, and comparison with expected diagnostic performance. Finally, conclusions are formulated based on the experimental findings, highlighting the contribution of the proposed approach and providing recommendations for future research and system enhancement.

III. RESULTS AND DISCUSSION

This section presents the results of the proposed fault diagnosis model along with an analysis of its performance. The discussion includes data analysis results, model training and optimization, and evaluation based on selected performance metrics to assess the effectiveness of the approach in detecting and classifying ship engine faults.

The initial analysis examines the distribution of the dataset to understand the characteristics of the ship engine sensor data used in this study. This analysis aims to identify the proportion of samples across different fault classes, detect potential class imbalance, and observe the statistical behavior of each feature. Understanding the data distribution is important to ensure data quality and provide a foundation for interpreting the results of the subsequent modeling process.

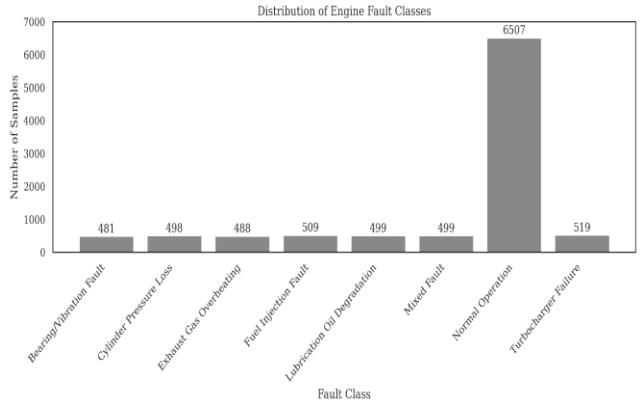

Figure 4. Distribution of Dataset

The figure 4 shows the distribution of samples across different engine fault classes in the dataset. It can be observed that most fault categories have a relatively similar number of samples, ranging from approximately 480 to 520 instances per class, indicating a generally balanced distribution among fault types. However, the Normal Operation class contains a significantly higher number of samples (6507), making it the dominant class in the dataset. This imbalance suggests that the dataset is skewed toward normal operating conditions, which may influence the model training process and should be considered when interpreting classification performance.

After analyzing the data distribution, Principal Component Analysis (PCA) is applied to reduce the dimensionality of the dataset while preserving important variance information. This technique helps visualize data patterns and observe the separability among different engine fault classes before the modeling stage.

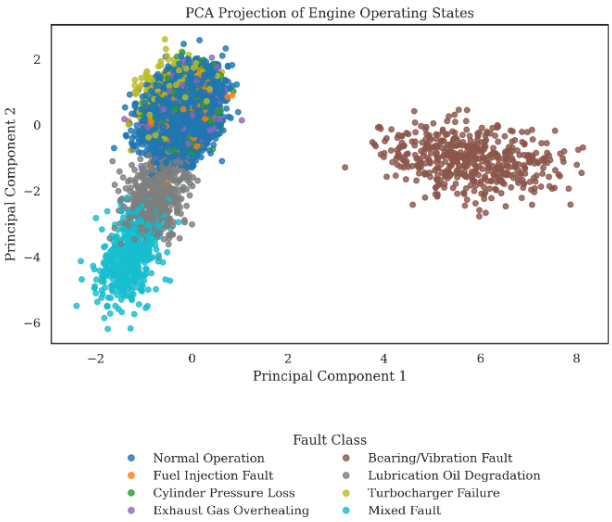

Figure 5. PCA Visualization

The figure 5 shows the PCA projection of engine operating states into two principal components to visualize the data structure in a lower-dimensional space. It can be observed that the Normal Operation class forms a clearly separated cluster from the fault conditions, indicating strong distinguishable characteristics between normal and faulty states. Meanwhile, several fault classes appear partially overlapping, suggesting similarities in their sensor patterns and potentially increasing the difficulty of classification among certain fault types. Overall, the PCA visualization demonstrates that the dataset contains meaningful separability, supporting the feasibility of machine learning–based fault diagnosis.

To obtain a more detailed visualization of the data structure, t-distributed Stochastic Neighbor Embedding (t-SNE) is further applied as a nonlinear dimensionality reduction technique.

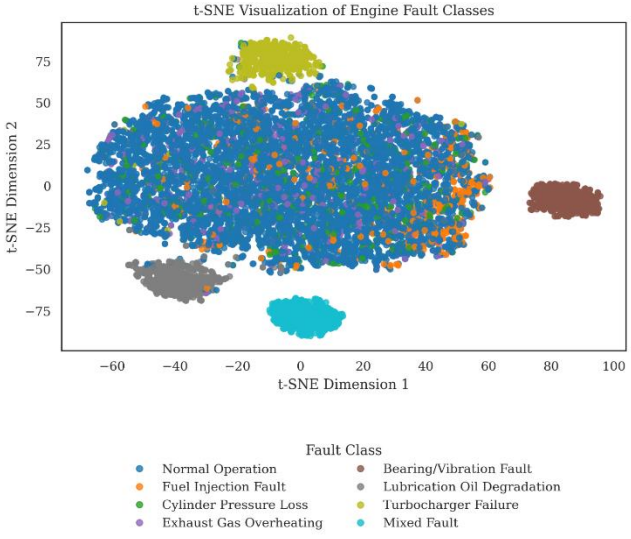

Figure 6. t-SNE Visualization

The figure 6 presents the t-SNE visualization of engine fault classes in a two-dimensional space, highlighting the local

structure and clustering patterns of the dataset. Compared to PCA, t-SNE provides clearer separation for several classes, where some fault types form distinct clusters, indicating strong similarities within the same class. However, a large central region shows overlapping points among multiple classes, suggesting that certain fault conditions share similar sensor characteristics, which may increase classification complexity. Overall, the visualization confirms the presence of meaningful patterns in the data while also indicating potential challenges in distinguishing closely related fault types.

To evaluate the influence of different parameter configurations on model performance, five experimental cases are designed using the XGBoost algorithm with varying hyperparameter settings. The first case represents the baseline model, while the remaining cases explore the effects of learning rate adjustment, tree depth variation, number of estimators, and regularization. The experimental configurations are summarized in Table 1.

TABLE 1.
HYPERPARAMETER CONFIGURATION

Case	n estimators	learning rate	max depth	reg lambda	reg alpha
Case1	200	0.10	6	-	-
Case2	300	0.03	6	-	-
Case3	250	0.05	10	-	-
Case4	500	0.05	6	-	-
Case5	300	0.05	6	2.0	1.0

In addition to the variation of hyperparameter configurations, the experiments are further conducted using different train–test split ratios to evaluate the robustness and generalization capability of the model under varying data proportions. Three data partition scenarios are considered, namely 80:20, 70:30, and 50:50 for training and testing, respectively. For each data split scenario, all five experimental cases described in Table X are applied independently. Consequently, a total of fifteen training experiments are performed to comprehensively analyze the impact of both parameter configurations and data partitioning on the model performance. This experimental design enables a more reliable comparison and provides deeper insight into the stability of the proposed approach across different training conditions.

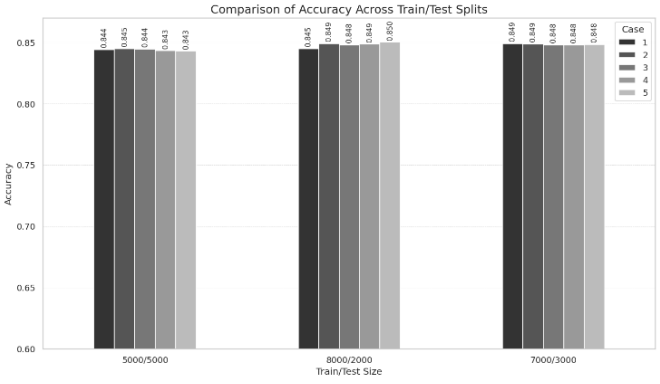

Figure 7. Comparison of model accuracy

Figure 7 illustrates the comparison of model accuracy across different train–test split scenarios for five hyperparameter configurations (Case 1–5). The results show that the accuracy values remain relatively consistent within the range of approximately 0.843 to 0.850 across all experimental settings. Among the evaluated splits, the 8000/2000 (80% training – 20% testing) configuration achieved the highest accuracy, reaching approximately 0.850. This finding indicates that increasing the proportion of training data contributes positively to the model's learning capability and overall predictive performance. In contrast, the 5000/5000 (50% training – 50% testing) split produced slightly lower accuracy values, suggesting that a reduced training dataset limits the model's ability to capture underlying patterns effectively. Furthermore, the performance differences among the five hyperparameter cases within each split are relatively small, implying that variations in parameter settings within the tested range have a limited impact compared to the influence of training data size. Overall, these results highlight the importance of sufficient training data in achieving stable and optimal classification performance.

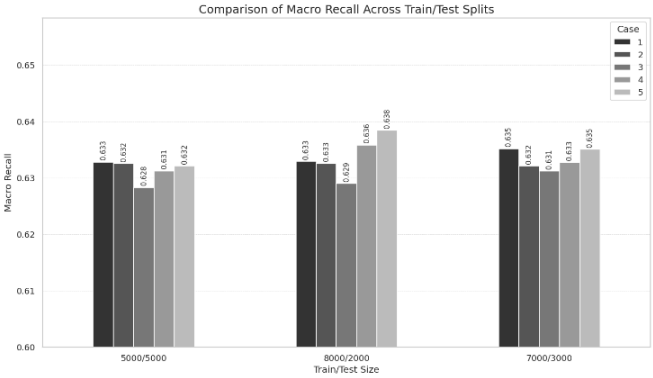

Figure 8. Comparison of macro recall

Figure 8 presents the comparison of macro recall values across different train–test split scenarios for the five hyperparameter configurations (Case 1–5). In contrast to the accuracy results, the macro recall values are relatively lower and range approximately between 0.628 and 0.638 across all experiments. The highest macro recall is observed under the 8000/2000 split, reaching approximately 0.638, indicating that increasing the proportion of training data also improves the model's ability to correctly identify instances across all classes.

The 5000/5000 split produces slightly lower macro recall values, suggesting that a smaller training set limits the model's sensitivity in detecting minority or less represented fault classes. The 7000/3000 split yields intermediate performance, reinforcing the trend that a larger training dataset contributes positively to recall performance. Additionally, the differences among the five hyperparameter cases within each split scenario remain relatively small, indicating that variations in hyperparameter configurations have limited impact compared to the influence of training data proportion.

Overall, these results suggest that while the model achieves stable overall accuracy, improvements in macro recall are more sensitive to data distribution and class balance, highlighting the importance of sufficient and representative training data for enhancing multi-class fault detection performance.

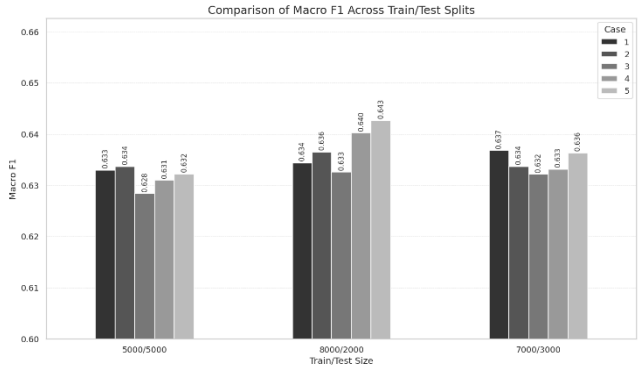

Figure 9. Comparison of macro F1-scores

Figure 9 illustrates the comparison of macro F1-scores across different train–test split scenarios for the five hyperparameter configurations (Case 1–5). The macro F1 values range approximately between 0.628 and 0.643, indicating moderate and consistent multi-class classification performance across all experimental settings. Similar to the accuracy and macro recall results, the highest macro F1-score is obtained under the 8000/2000 split, reaching approximately 0.643. This finding confirms that a larger proportion of training data improves the balance between precision and recall across all classes.

The 5000/5000 split produces slightly lower macro F1 values, reflecting reduced capability in simultaneously maintaining class-wise precision and recall when the training data size is limited. The 7000/3000 split shows intermediate performance, further reinforcing the positive impact of increased training data on overall classification quality. Differences among the five hyperparameter configurations within each split scenario remain relatively small, suggesting that within the tested parameter ranges, model performance is more influenced by the proportion of training data than by hyperparameter variation.

Overall, the macro F1 results indicate that while the model achieves stable overall classification performance, improvements in balanced multi-class detection are closely associated with data availability and class representation.

After the initial training phase and performance evaluation using the training curves, which demonstrated convergence and model stability, the next stage involved hyperparameter optimization using the Grid Search method. This approach was applied to identify the optimal parameter combination to further improve model performance. Grid Search systematically evaluates predefined hyperparameter sets through cross-validation, resulting in a model with improved generalization capability. The parameters optimized included key components of the XGBoost algorithm, such as the learning rate, number of estimators, maximum tree depth (max_depth), and regularization terms. The optimal configuration obtained from this process was then used as the final model for evaluation on the testing dataset. This tuning stage aims to achieve a better balance between bias and variance, thereby enhancing the accuracy, stability, and robustness of the model in detecting ship engine fault conditions

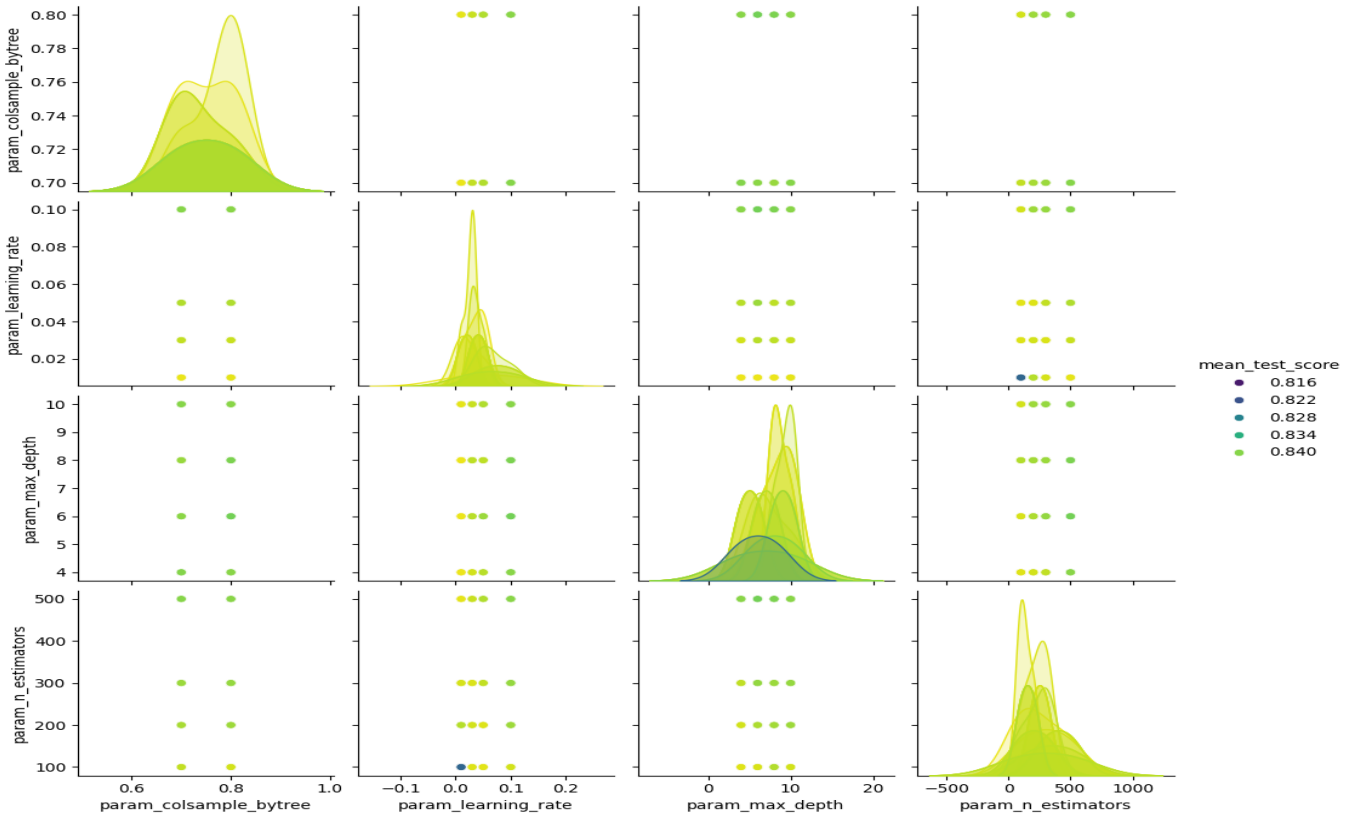

Figure 10. Pairwise Visualization of Grid Search Hyperparameter Results

Figure 10 presents the pairwise visualization of the Grid Search results, illustrating the relationship between selected hyperparameters and the mean cross-validation score. The color gradient represents the mean_test_score, where lighter shades indicate higher accuracy. The highest accuracy achieved was 0.840, obtained using a configuration characterized by a low learning rate (approximately 0.01–0.03), a higher number of estimators (around 500), a moderate-to-high max_depth (approximately 8–10), and colsample_bytree values in the range of 0.8–0.9.

The results indicate that improved performance is primarily associated with lower learning rates combined with a larger number of estimators, reflecting the typical boosting trade-off in which smaller learning rates require more iterations but enhance generalization capability. In addition, moderate to deeper tree structures contribute positively to classification accuracy, suggesting that sufficient model complexity is required to capture nonlinear patterns in the sensor data. In contrast, variations in colsample_bytree exhibit a comparatively minor influence on performance. Overall, the analysis confirms that learning rate and tree depth are the most influential hyperparameters in determining model accuracy, while an increased number of estimators further stabilizes and improves classification performance.

IV. CONSLUSSION

This study demonstrates that the XGBoost model with manually selected hyperparameters achieved an accuracy of 0.85, while hyperparameter optimization using Grid Search produced a highest mean cross-validation accuracy of 0.840. These findings indicate that the tuning process did not significantly improve absolute performance compared to the initial configuration, although it provided a more systematically validated set of parameters. Overall, all experimental scenarios yielded relatively consistent accuracy values within the range of 0.84–0.85, reflecting the model's stability in classification tasks. This consistency further suggests that the model has approached the predictive limit that can be extracted from the available data representation, which is likely influenced by overlapping distributions between fault and normal conditions, inherent noise in sensor measurements, and the complexity of engine operational patterns that are not fully separable within the feature space. Therefore, the achieved performance can be considered a realistic representation of the model's predictive capacity given the structure and quality of the dataset, indicating that the proposed model is sufficiently stable and reliable for supporting early fault detection in ship main engines under practical operational conditions.

REFERENCES

[1] W. Busse, N. Siswantoro, and M. N. Bintang, "Dynamic Information System for Failure Analysis with It's Application on Ship Main Engine," *International Journal of Marine Engineering Innovation and Research*, vol. 8, no. 3, p. 570, 2023.

[2] A. Ali and A. Abdelhadi, "Condition-Based Monitoring and Maintenance: State of the Art Review," Jan. 01, 2022, *MDPI*. doi: 10.3390/app12020688.

[3] Y. Gholipour, M. Zare, M. Vaziri Sereshk, and Y. Gholipour, "A Comprehensive Review of Maintenance Strategies: From Reactive to Proactive Approaches," *Central Asia and the Caucasus*, vol. 26, 2025.

[4] R. Gagana Erwin Asmara, T. Sistem Perkapalan, F. Teknik dan Ilmu Kelautan, and U. Hang Tuah Surabaya, "Analisa Kegagalan Sistem Bahan Bakar Kapal Dengan Menggunakan Metode Preliminary Hazard Analysis (Pha) Dan Fault Tree Analysis (Fta)," *HEXAGON*, vol. 3, 2022.

[5] P. Prasetyo, A. Seno, and P. Pelayaran Sumatera Barat, "Jurnal Cakrawala Bahari Analisis Penyebab Terjadinya Overheat pada Main Engine di Kapal Self Propelled Oil Barge Tirta Samudra XVIII," *Jurnal Cakrawala Bahari*, vol. 5, no. 2, pp. 5–10, 2022, [Online]. Available: http://jurnal.poltekpelsumbar.ac.id/index.php/jcb

[6] M. H. Park and W. J. Lee, "Comprehensive review of shipboard maintenance management strategies," Sep. 01, 2025, *Elsevier B.V.* doi: 10.1016/j.rineng.2025.106671.

[7] M. Mursidi and A. Sarjito, "Implementation Strategy of Ship Engine Maintenance Management System to Improve Operational Efficiency," *Jurnal Aplikasi Pelayaran Dan Kepelabuhanan*, vol. 15, no. 2, pp. 201–214, Feb. 2025, doi: 10.30649/japk.v15i2.137.

[8] A. S. Kalafatelis, N. Nomikos, A. Giannopoulos, G. Alexandridis, A. Karditsa, and P. Trakadas, "Towards Predictive Maintenance in the Maritime Industry: A Component-Based Overview," Mar. 01, 2025, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/jmse13030425.

[9] T. Kirketerp-Møller, M. W. Hyldgaard, J. Cai, A. I. Dodis, and N. G. M. Rytter, "Data-driven predictive maintenance for two-stroke marine diesel engines using machine learning and MLOps," *Journal of Ocean Engineering and Science*, 2025, doi: 10.1016/j.joes.2025.11.011.

[10] I. T. Jollife and J. Cadima, "Principal component analysis: A review and recent developments," Apr. 13, 2016, *Royal Society of London*. doi: 10.1098/rsta.2015.0202.

[11] L. Van Der Maaten and G. Hinton, "Visualizing Data using t-SNE," 2008.

[12] Z. Qinghe, X. Wen, H. Boyan, W. Jong, and F. Junlong, "Optimised extreme gradient boosting model for short term electric load demand forecasting of

regional grid system," *Sci. Rep.*, vol. 12, no. 1, Dec. 2022, doi: 10.1038/s41598-022-22024-3.

[13] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machin," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[14] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 13-17-August-2016, pp. 785–794, Aug. 2016, doi: 10.1145/2939672.2939785.

[15] Q. A. Hidayaturrohman and E. Hanada, "A Comparative Analysis of Hyper-Parameter Optimization Methods for Predicting Heart Failure Outcomes," *Applied Sciences (Switzerland)*, vol. 15, no. 6, Mar. 2025, doi: 10.3390/app15063393.

[16] P. Calle *et al.*, "Integration of nested cross-validation, automated hyperparameter optimization, high-performance computing to reduce and quantify the variance of test performance estimation of deep learning models," *Comput. Methods Programs Biomed.*, vol. 272, Dec. 2025, doi: 10.1016/j.cmpb.2025.109063.