A Stacking Ensemble Approach for Predictive Maintenance of Automotive Engines

Edrich Darren A. Santuyo
College of Computing and Information Technologies
National University
Manila, Philippines
santuyoea@students.national-u.edu.ph

Danfred Martin D. Isip
College of Computing and Information Technologies
National University
Manila, Philippines
isipdd@students.national-u.edu.ph

Jonel Villaver
College of Computing and Information Technologies
National University
Manila, Philippines
villaverj@students.national-u.edu.ph

***Abstract*—Traditional vehicle maintenance such as scheduled maintenance and manual inspection often lead to expensive failure and unnecessary component replacement. To address this issue, data-driven approaches were applied. With the rapid advancement of Machine Learning, the study examines base models such as KNN, Logistic Regression, SVM, Random Forest, Decision Tree, XGBoost and Gradient Boosting. Upon strict evaluation, engine health can be assessed with the use of Stacking Ensemble integrated with Random Forest, KNN, Gradient Boosting and XGBoost, partnered with SMOTE for data balancing, feature scaling and domain-informed featured engineering. Model performance was appraised using accuracy, precision, recall and f1-score. To ensure reliability and stability, cross validation was applied with f1-macro metrics. Results showed that the stacked ensembles have an accuracy of 82% and an average f1-macro score of 0.812, surpassing all of baseline and individual ensemble models. This established its potential to reduce costly maintenance and risk of accident. While being tested with synthetic data, the study features potential application in real-world automotive systems and recommends using other non-linear models to further enhance the prediction.**

***Index Terms*—Predictive Maintenance (PdM), Machine Learning, Stacking Ensemble, Automotive Engine Health, Random Forest, Sensor Data Analytics**

I. Introduction

Automotive vehicles must operate reliably, yet conventional maintenance methods are ineffective. Reactive maintenance is undertaken only after an expensive failure, while periodic preventive maintenance often leads to the wasteful replacement of still-functional parts [1]. By using vehicle sensor data to predict faults before they happen, Predictive Maintenance (PdM) provides an improved, data-driven substitute [2]. Since dashboard indications and other existing systems are usually reactive to damage that has already occurred, the main difficulty is deciphering this complicated sensor data to deliver real preventative warnings.

By creating and assessing a variety of supervised machine learning models to categorize engine health, this study tackles this difficulty. We show that the most accurate results are obtained with a high-performance stacking classifier that combines the predictive capabilities of top-performing ensembles such as Random Forest and Gradient Boosting. A "just-in-time" maintenance system that greatly lowers unplanned downtime and enhances safety is made possible by this method. By reducing operating expenses and increasing vehicle availability, the suggested model is a useful, high-impact instrument for the automotive sector that provides significant advantages to both private car owners and commercial fleet managers [3].

II. Review of Related Literature

Traditionally, vehicle maintenance depended on fixed scheduling and manual inspection to check if the engine remained in a healthy condition, which was inefficient and inconvenient. However, as technology developed, advancements in vehicle sensors enabled continuous data collection, allowing early detection of anomalies within vehicle engines [4]-[6]. Jamal *et al.* [7] emphasized the continuous evolution of vehicle maintenance through machine learning and artificial intelligence (AI). Due to this shift, vehicle maintenance has transformed from schedule-based to proactive and data-driven methods, enhancing both safety and efficiency.

In addition, automotive systems' prediction has become an essential approach for evaluating engine health and anticipating potential failures. For instance, Arena *et al.* [1] highlighted the importance of data-driven predictive maintenance in boosting reliability and reducing downtime in automotive engines. Li *et al.* [8] used sensor-based data to forecast engine deterioration patterns, demonstrating the efficiency of machine learning in detecting latent deficiencies. Likewise, Patel *et al.* [9] applied Random Forest (RF) and Support Vector Machine (SVM) for assessing engine performance, achieving accurate detection of mechanical anomalies through multivariate sensor analysis. Furthermore, Zhang *et al.* [10] investigated data preprocessing and feature extraction techniques to improve interpretability and classification performance in predictive maintenance systems.

To further improve prediction accuracy, ensemble-based approaches have been widely employed in recent studies. Reddy *et al.* [11] demonstrated that applying stacking by combining multiple classifiers significantly increases predictive performance and model stability compared to using individual models. Similarly, Ahmad *et al.* [12] conducted a comparative study of ensemble learning methods and discovered that

stacking ensembles achieve higher generalization in industrial anomaly detection. These findings support the application of stacking ensemble methods in automotive engine prediction to enhance accuracy in identifying deficiencies and improving overall model performance.

Moreover, metrics such as the F1-score are commonly used to measure performance in predictive maintenance (PdM), which has progressed from simple rule-based or statistical approaches to sophisticated machine learning techniques [2]. Although baseline models often rely on fundamental methods, advanced tree-based ensembles—such as Random Forests and gradient boosting algorithms like XGBoost—dominate state-of-the-art classification on tabular data [13]. Deep learning models, such as Long Short-Term Memory (LSTM) networks, can automatically learn temporal dependencies to predict a component's Remaining Useful Life (RUL), representing the cutting edge of prognostics using time-series data [8].

Several researchers have attempted to address the problem of predicting engine health through stacked ensemble approaches. Among them, Isinka *et al.* [14] improved prediction accuracy by combining multiple deep learning models into a stacked ensemble, achieving accuracy levels above 94%. However, their approach required significant computational resources and lacked interpretability.

In comparison, the present study utilizes a stacking ensemble to address limitations of previous research in terms of data imbalance, model complexity, and limited interpretability. By incorporating Random Forest, K-Nearest Neighbors (KNN), Gradient Boosting, and XGBoost as base models—with Random Forest serving as the final estimator—combined with SMOTE-based data balancing and domain-informed feature engineering, this study aims to provide a more efficient and interpretable solution that aligns with recent advances in predictive maintenance research while attaining over 80% classification accuracy.

III. Methodology

A. Data Collection

The data for this study is from the "Automotive Vehicles Engine Health Dataset, " a collection of synthetically generated sensor readings. The dataset, created by user Parv Modi and last updated in 2023, is publicly available on the Kaggle data science community platform. The original dataset consists of 19,535 rows, each with seven columns. These include six predictor features representing raw sensor measurements and a target variable named engine condition. The target is labeled with '1' representing a healthy engine and '0' denoting an at-risk engine. The dataset is imbalance, containing 12,317 healthy instances and 7,218 at-risk instances.

B. Data Pre-Processing

Before training our model, we did several preprocessing steps to make sure the data was balanced and clean to use. Since our dataset had more healthy engines (12,317) and unhealthy engines (7,218), we used both RandomOverSampler and SMOTE to compromise on the imbalance. RandomOverSampler duplicates samples from the smaller class, while SMOTE creates new synthetic samples based on similar data points. This helped make the dataset fairer and prevented the model from favoring just one side of the dataset. Next we applied StandardScaler to make all features have a similar scale, which is important for algorithms like KNN, LabelEncoder was used to turn categorical labels into numbers so models could process them. On top of everything else mentioned, we checked the dataset for any missing or duplicate values to make sure the data was clean.

C. Experimental Setup

Initially, Google Colab was utilized as the primary development environment, however, it was later replaced by Visual Studio Code (VS Code) due to the time-consuming nature of output generation. To accomplish the objectives, a dedicated Python environment was configured with the following libraries and versions: `imbalanced-learn` 0.0, `matplotlib` 3.10.7, `NumPy` 2.3.3, `pandas` 2.3.3, `scikit-learn` 1.7.2, `seaborn` 0.13.2, and `XGBoost` 3.0.5.

An imbalance was observed between the classes, which led to biased and inaccurate predictions. In order to solve this problem, the `RandomOverSampler` technique was used to duplicate instances of the minority class until they matched the majority class, balancing the dataset. 24,634 data samples were produced as a result of this procedure. Following that, the dataset was split 80:20 between training and testing, yielding 19,707 training samples and 4,927 testing samples.

In order to determine the best possible model combinations, hyperparameter tuning was applied to each model, centered on the parameter grid that is specific to that model. For SVM and Logistic regression, important parameters like regularization strength (C), kernel type, and gamma were adjusted; for KNN, `n_neighbors` and distance metrics; for Random Forest and Decision Tree, `n_estimators`, `max_depth`, and `min_samples_split`; and, finally, for XGBoost and Gradient Boosting, `learning_rate`, `n_estimators`, and `max_depth`. To guarantee balanced accuracy and stability across models, the optimized configurations were subsequently implemented into a stacking ensemble.

D. Algorithm

Starting with well-known baseline models and ending with a high-performance stacking ensemble, this study assesses a wide variety of supervised machine learning techniques. In order to thoroughly investigate the predictive patterns found in the engine sensor data, the models were selected to reflect several learning paradigms, such as distance-based, and tree-based approaches.

K-Nearest Neighbors (KNN)

As an instance-based, non-parametric technique, K-Nearest Neighbors classifies a new data point according to the majority class of its 'k' nearest neighbors [15]. Because of its simplicity

and capacity to identify local, non-linear patterns without assuming anything about the distribution of the underlying data, KNN was chosen. The model was tuned for this investigation, and the Euclidean distance metric was used to make predictions.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \quad (1)$$

Equation 1: *Euclidean Distance Formula*

Where:

- $d(p,q)$ is the distance between points p and q.
- p_i and q_i are the values for the i-th feature of each point.

Random Forest

During training, Random Forest, an ensemble learning technique, builds a large number of decision trees and outputs the class that is the mode of the classes predicted by the individual trees [16]. Because of its excellent accuracy, resilience to overfitting, and inherent capacity to provide feature importance rankings, this model was selected. In order to provide a varied and potent prediction ensemble, it builds each tree on a random portion of the data and only takes into account a random subset of features for each split.

Gradient Boosting (XGBoost and GBC)

Gradient Boosting is a progressive ensemble technique in which previous models' flaws are corrected by adding new models [17]. We made use of two well-known implementations: XGBoost and Scikit-learn's `GradientBoostingClassifier` (GBC). These models were selected due to their extraordinarily high predicted accuracy and reflect the state-of-the-art for tabular data. They can discover intricate and nuanced patterns in the data because they are trained by iteratively minimizing a loss function (such as Log Loss) via gradient descent.

Stacking Ensemble Classifier

The Stacking (Stacked Generalization) Ensemble Classifier is the last model put forth in this investigation. Heterogeneous machine learning models are combined in a sophisticated process called stacking to synthesize their predictions and attain better performance [18]. In order to test the hypothesis that a combination of multiple cutting-edge models could produce a more accurate result than any one model alone, we selected this design. There are two levels to the architecture:

1) Level-0 (Base Models): The dataset is used to train four excellent models: Random Forest, KNN, XGBoost, and `GradientBoostingClassifier`.
2) Level-1 (Meta-Model): A final meta-model (a Random Forest) is trained using the original features (`passthrough=True`) and the predictions from the Level-0 models. In order to arrive at the final categorization, this meta-model learns the best strategy to aggregate the outputs from the basic models. To prevent data leakage, a 10-fold internal cross-validation process is used during training.

E. Training Procedure

To make sure our model training and evaluation were as reliable as possible, we used several strategies throughout the process, we started by splitting the dataset into 80% training and 20% testing data. This allowed us to train the models on one portion and then evaluate their performance on unseen data, helping to detect overfitting early on.

Featured engineered the following:

- Oil_to_coolant_temp_ratio
- Temp_difference
- Oil_to_fuel_pressure_ratio
- Pressure_difference
- Engine_stress_index

Then handled the class imbalances using:RandomOverSampler, also applying StandardScaler for feature normalization. These methods helped us balance the dataset, reducing bias towards the majority data and improving the model's ability to generalize new data. Then we used LabelEncoder to achieve the target variable. This allowed us to train the models on one portion and then evaluate their performance on unseen data, helping to detect overfitting early on. For more reliable performance check, we also decided to use Stratified K-Fold Cross-Validation with 5 splits. Stratification kept the class ratios consistent across all folds, which was important since our dataset was imbalanced. Each model was trained and validated on different models as we used the average performance across folds as a much fairer measure of accuracy.

To fine-tune the model performance, we used GridSearchCV together with StratifiedKFold. This method systematically tested different combinations of hyperparameters to find the most effective setup for each algorithm, improving both accuracy and generalization for the following models:

- Logistic Regression
- K-Nearest Neighbors
- Support Vector Machine
- Random Forest
- Decision Tree
- XGBoost
- Gradient Boosting Classifier

Lastly we then implemented ensemble learning through a stacking model that combined multiple models such as Random Forest, KNN, XGBoost, and Gradient Boosting. A random Forest meta-model was used to learn how to best combine their predictions. Cross-validation was applied within the stacking process to minimize overfitting and ensure robust performance. Overall, by combining cross-validation, grid search, ensemble learning, and resampling techniques, we were able to build a much better training strategy to get an accurate score.

F. Evaluation Metrics

For model evaluation and comparison, *Accuracy*, *Precision*, *Recall*, *F1-Score*, and *F1-Macro* were applied. Since the dataset was already balanced using the Random Over Sampling technique, Accuracy was used to determine the proportion of correct predictions. Although Accuracy is sufficient, it is also important to understand how well the model predicts the positive class. Hence, Precision was included to determine how many of the positive predictions were actually correct, while Recall measures how many of the actual positive samples were correctly identified. The F1-Score provides a balance between Precision and Recall.

On the other hand, F1-Macro was used during cross-validation as the scoring method, since relying solely on Accuracy can be misleading even for balanced datasets. F1-Macro treats all classes equally by providing equal weight through averaging the F1-Score across all classes, ensuring fair evaluation of model performance.

In addition, utilizing *GridSearchCV* was significant for hyperparameter tuning to further determine which models should be included as base estimators. The scoring metric used was Accuracy, identifying which parameter combination yields the most promising results. As a result, the selected models for experimentation were Random Forest (RF), k-Nearest Neighbors (KNN), Extreme Gradient Boosting (XGBoost), and Gradient Boosting Classifier (GBC).

G. Baselines and Comparative Models

A comprehensive collection of baseline and comparative models were trained and evaluated on the same dataset to fully evaluate our proposed Stacking Ensemble Classifier's performance. To provide a multi-tiered benchmark, various models were chosen, ranging from simple interpretable models to the most sophisticated models currently on the market. The performance comparison of each model justifies the complexity and effectiveness of our final ensemble.

We used the following individual models for comparison:

1) Simple Baseline Models: A standard Decision Tree and Logistic Regression were implemented to establish a foundational performance baseline. These models offer high interpretability and serve as a reference point to ensure more complex models provide a tangible performance benefit.
2) Standard Machine Learning Models: K-Nearest Neighbors (KNN) and Support Vector Machine (SVM) were evaluated as they represent different and powerful classification paradigms (distance-based and kernel-based).
3) Advanced Ensemble Models: A standalone Random Forest, XGBoost Classifier (XGB), and Scikit-learn's GradientBoostingClassifier (GBC) were included in our comparison. As these models themselves represent the state-of-the-art for tabular data, they serve as the most critical benchmark. The primary goal of our stacking classifier is to outperform these powerful, individual ensembles.

Each of these comparative models was subjected to the same data preprocessing, feature engineering, and evaluation procedures as our final model to ensure a fair and direct comparison.

IV. Discussion

A. Baselines and Comparative Models

These are the baseline and comparative models used for testing out the threshold of what accuracy scores we can get from different models.

Baseline: Logistic Regression and Decision Tree Standard ML Models: KNN and SVM Advanced Ensemble Models: Random Forest, XGBoost Classifier (XBC), Gradient Boosting Classifier (GBC).

TABLE I
COMPARATIVE MODELS

Model	Type	Accuracy Score	Precision Score	Recall Score	F1 Score
Logistic Regression	Baseline	0.64	0.68	0.64	0.66
Decision Tree	Baseline	0.73	0.66	0.78	0.71
KNN	Standard ML	0.73	0.59	0.82	0.69
SVM	Standard ML	0.65	0.61	0.68	0.64
Random Forest	Advanced Ensemble	0.77	0.72	0.81	0.77
XGBoost Classifier	Advanced Ensemble	0.75	0.69	0.79	0.74
Gradient Boosting Classifier	Advanced Ensemble	0.68	0.64	0.70	0.67

The baseline models like Logistic Regression and Decision Tree gave us a simple starting point, but couldn't handle the complex patterns in data. KNN performed better as seen from the table above[] by picking up local trends, while SVM struggled due to the data's non-linearity and imbalance. The ensemble models Random Forest, XBC, GBC did much better, showing higher accuracy and stronger generalization. In the end, the stacking ensemble combined several of the models achieved the best performance of 82%, proving that blending different strong models works better than one singular model.

B. Hyperparameter Tuning of each Model

To further optimize our models, we used Hypertuning. It is conducted to find the most optimal/best configuration that the algorithm can use in order to produce optimal or the best results possible. By implementing Hypertuning it ensured that each model operated under its most effective setup, providing a fair basis

TABLE II
BEST MODEL PARAMETERS AND CROSS-VALIDATION SCORES

Model	Best Parameters	CV Score
Logistic Regression	C = 1, Solver = 'lbfgs'	0.63
KNN	N_neighbors = 23, Metric = 'euclidean', Weights = 'distance'	0.71
SVM	Kernel = 'rbf', C = 100, Gamma = 'scale'	0.65
Decision Tree	Criterion = 'gini', Max_depth = 10	0.72
Random Forest	N_estimators = 200, Max_depth = None, Min_samples_split = 5, Min_samples_leaf = 1	0.77
XGBoost	Colsample_bytree = 0.8, Gamma = 0, Lr = 0.2, Max_depth = 7, Min_child_weight = 1, N_estimators = 300, Subsample = 1.0	0.75
Gradient Boosting	Lr = 0.2, Max_depth = 4, Min_samples_split = 4, Min_samples_leaf = 1, N_estimators = 150, Subsample = 1.0	0.67

After tuning the models, Random Forest and XGBoost game out on top with the best CV scores, showing they handled the complex engine data the best. Simple models like Logistic

Regression and SVM did okay, but struggled with non-linear patterns. KNN and Decision Tree got better after tuning, but couldn't match the ensemble models.

C. Stacked Ensemble Model

TABLE III
CLASSIFICATION REPORT FOR STACKED ENSEMBLE (RF + KNN + XGB + GBC)

Class	Precision	Recall	F1	Acc	CV (f1-macro)
Unhealthy(0)	0.71	0.91	0.79		
Healthy(1)	0.93	0.77	0.84	0.82	0.81

According to the data presented on Table III, the model accurately predicted the healthy engine as healthy due to precision being 0.93. With 0.82 accuracy. However, the model is sensitive to unhealthy engines due to high recall, which is beneficial for early detection, but there's a risk of detecting a healthy engine as unhealthy.

TABLE IV
CROSS VALIDATION RESULTS FOR THE STACKED ENSEMBLE

Fold	1	2	3	4	5
f1-macro	0.818	0.806	0.810	0.810	0.815

Upon evaluating the model through Cross Validation, the results demonstrated that the model is reliable and stable as the average f1-macro is 0.812 with a deviation of 0.005. This portrays that the base model combined in stacking performs consistently and does not cause overfitting heavily.

D. Evaluation of Stacked Model with Synthetic Data

To further examine the model capability and reliability to predict automotive engine condition. It was tested on six synthetic data classified as healthy and unhealthy respectively. Before prediction, the data is preprocessed to ensure normalization and the model will accept it.

TABLE V
EVALUATION DATA AND RESPECTIVE PREDICTIONS ON SAMPLE CASES

FEATURES	Case 1	Case 2	Case 3	Case 4
ENGINE_RPM	**750**	720	1210	**946**
Lub_oil_pressure	**3.1**	2.95	4.05	**3.84**
Fuel_pressure	**12.0**	**12.3**	6.82	**6.70**
Coolant_pressure	**3.0**	2.6	2.30	**4.27**
lub_oil_temp	**82.0**	**83.1**	76.9	**77.6**
Coolant_temp	**80.5**	**81.6**	75.9	**79.4**
oil_to_coolant_temp_ratio	**1.019**	**1.018**	1.016	**0.978**
temp_difference	**1.5**	**1.5**	0.95	**-1.77**
oil_to_fuel_pressure_ratio	**0.258**	**0.240**	0.607	**0.574**
pressure_difference	**9.0**	**9.7**	1.68	**-0.429**
engine_stress_index	**2909.9**	**2950.7**	2060.2	**1648.2**
Actual Label	**1**	**1**	**0**	**0**
Model Prediction	**1**	**1**	**1**	**0**

The results in Table V show that the ensemble model is effective at identifying healthy engines, as demonstrated in Case 1 and Case 2. However, the table also highlights a critical behavior: the model's tendency to misclassify less severe or "borderline" unhealthy engines. This is perfectly illustrated in Case 3, where the actual label is 'unhealthy' (0), but the model incorrectly predicted it as 'healthy' (1).

In contrast, the model correctly identifies the unhealthy engine in Case 4. A comparison of the feature values suggests why: the engine in Case 4 presents more pronounced anomalies, such as a negative pressure_difference, which the model correctly flags as a sign of risk. This indicates that the model may require a stronger, more obvious pattern of failure to make an 'unhealthy' prediction, causing it to misclassify the more ambiguous case. This behavior aligns with the model's high overall recall for at-risk engines, as it is tuned to catch the most critical failures.

V. CONCLUSION

This study went beyond the usual reactive and preventive maintenance models to look at the important problem of sudden, expensive breakdowns in the automotive industry. The main goal was to make and test a machine learning model that could correctly guess engine health from sensor data so that "just-in-time" repair could happen before it was needed. A careful comparison int the work demonstrated that while many individual models performed at a basic level, a stacking ensemble classifier had the highest expected accuracy, reaching 82%. The idea that combining the different predictive capabilities of several high-performing models, like Random Forest and XGBoost, could lead to a more stable and better result than any single model working by itself was proven by this result.

The main thing this work adds is that it proves that this complex ensemble design can be used to accurately guess how healthy car engines are. We think the results are important because they show that advanced machine learning models can be used in the real world, specifically in the automotive industry. One of the real benefits is that it can be used in fleet management and car diagnostic systems. Making this change can lower costs, keep cars from breaking down, and make driving safer for both business and personal drivers. In support of the move towards data-driven maintenance, our study provides a guide for using cutting-edge ensemble methods.

There are, however, some issues with this study. The primary issue is the utilization of a synthetic dataset. Its performance on real-world sensor data, which is frequently noisy and incomplete, remains unknown, despite its efficacy for controlled experiments. Second, the model couldn't learn from temporal trends like the rate of change in sensor readings because the data is made up of single snapshots instead of time series data. The stacking model that has been proposed is a "black box", which presents a challenge in terms of comprehending its predictions. This might make it unsuitable for use in circumstances where safety is of the utmost importance.

REFERENCES

[1] F. Arena, M. Collotta, L. Luca, M. Ruggieri, and F. G. Termine, "Predictive Maintenance in the Automotive Sector: A Literature Review,"

Mathematical and Computational Applications, vol. 27, no. 1, p. 2, 2022, doi: 10.3390/mca27010002.

[2] A. K. S. Jardine, D. Lin, and D. Banjevic, "A review on machinery diagnostics and prognostics implementing condition-based maintenance," *Mechanical Systems and Signal Processing*, vol. 20, no. 7, pp. 1483–1510, Oct. 2006, doi: 10.1016/j.ymssp.2005.09.012.

[3] A. Leslie and D. Murray, *An Analysis of the Operational Costs of Trucking: 2024 Update*. Arlington, VA, USA: American Transportation Research Institute (ATRI), July 2024.

[4] Y. Zhang, X. Li, and H. Chen, "Predictive maintenance for intelligent vehicles using real-time sensor data analytics," *IEEE Access*, vol. 9, pp. 11245–11258, 2021, doi: 10.1109/ACCESS.2021.3051234.

[5] O. Olatunji, A. Hussain, and J. Lee, "Vehicle health monitoring and fault diagnosis using IoT-based sensor systems," *Sensors*, vol. 22, no. 3, p. 1015, 2022, doi: 10.3390/s22031015.

[6] R. Tiwari and P. Gupta, "A review on predictive maintenance techniques in automotive systems," *Journal of Intelligent Manufacturing and Systems*, vol. 34, no. 2, pp. 143–158, 2023.

[7] H. Jammal, J. E. Houssainy, F. Srour, M. Tormos, R. Sakr, and R. Achkar, "Prediction of automotive vehicles engine health using MLP and LR," in *Proc. 3rd Int. Conf. Innovations in Computing Research (ICR'24)*, K. Daimi and A. Al Sadoon, Eds., Lecture Notes in Networks and Systems, vol. 1058. Cham: Springer, 2024, pp. 98–109, doi: 10.1007/978-3-031-65522-7_9.

[8] Y. Li, J. Wang, and C. Zhao, "Machine learning-based health prediction of automotive engines using multivariate sensor data," *Mechanical Systems and Signal Processing*, vol. 159, p. 107819, Mar. 2021.

[9] R. Patel, A. Sharma, and K. S. Rajan, "Comparative study of machine learning algorithms for internal combustion engine fault detection," *Int. J. Prognostics and Health Management*, vol. 12, no. 2, pp. 44–53, 2021.

[10] T. Zhang, Q. Xu, and S. Zhou, "Feature extraction and selection for predictive maintenance in vehicle engines," *Applied Soft Computing*, vol. 108, p. 107495, Sep. 2021.

[11] S. Reddy, H. Kumar, and R. Bhat, "Stacking ensemble approach for predictive maintenance of industrial systems," *IEEE Access*, vol. 9, pp. 146712–146723, 2021.

[12] A. Ahmad, F. Ali, and M. Hussain, "Evaluation of ensemble learning methods for anomaly detection in mechanical systems," *Expert Systems with Applications*, vol. 185, p. 115634, Dec. 2021.

[13] A. C. C. de Souza, J. C. D. Santos, A. M. A. F. Guimarães, and F. A. D. M. Ghijs, "Machine learning for predictive maintenance in industrial IoT: A comparative study of algorithms and applications," *Int. J. Adv. Eng. Manage. (IJAEM)*, vol. X, no. Y, pp. ZZZ–ZZZ, Dec. 2024. [Online]. Available: https://ijaem.net/issue_dcp/Machine%20Learning%20for%20Predictive%20Maintenance%20in%20Industrial%20Iot%20A%20Comparative%20Study%20of%20Algorithms%20and%20Applications.pdf

[14] J. Isinka, N. Zaman, M. A. Rahim, and P. Pillai, "An ensemble deep learning model for vehicular engine health prediction," *IEEE Access*, vol. PP, no. 99, pp. 1–1, Jan. 2024, doi: 10.1109/ACCESS.2024.3395927.

[15] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. New York, NY, USA: Springer, 2009.

[16] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[17] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[18] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.