COBEM 2023 27TH INTERNATIONAL CONGRESS OF MECHANICAL ENGENEERING

ABCM Brazilian Society of Mechanical Sciences and Engineering
27th ABCM International Congress of Mechanical Engineering
December 04-08, 2023, Florianópolis-SC, Brazil

COB-2023-1672

MACHINE LEARNING APPROACHES APPLIED TO FAULT DETECTION OF TURBOCHARGED SPARK-IGNITED ENGINE SYSTEM

Lucas de Azevedo Takara
Department of Electrical Engineering (PPGEE), Federal University of Parana (UFPR), Curitiba, PR, Brazil
lucastakara@ufpr.br

Luiz Eduardo Thomaz
Mechanical Engineering Graduate Program (PPGEM), Pontifical Catholic University of Parana (PUCPR), Curitiba, PR, Brazil
luiz.thomaz@pucpr.edu.br

Viviana Cocco Mariani
Mechanical Engineering Graduate Program (PPGEM), Pontifical Catholic University of Parana (PUCPR), Curitiba, PR, Brazil
Department of Electrical Engineering, Federal University of Parana, Curitiba, PR, Brazil
viviana.mariani@pucpr.br

Leandro dos Santos Coelho
Department of Electrical Engineering (PPGEE), Federal University of Parana, Curitiba, PR, Brazil
Industrial Systems Engineering Graduate Program (PPGEPS), Pontifical Catholic University of Parana (PUCPR), Curitiba, PR, Brazil
leandro.coelho@pucpr.br

Abstract. *The continuous advancement of automotive technologies has led to an increase in the complexity of automobile functions and structures. As a result, fault diagnosis has become a popular and crucial topic in automotive engine systems, given the high susceptibility of their components to faults. While previous studies have utilized machine learning (ML) models, such as the multi-layer perceptron neural network, to detect misfire and pre-ignition faults in combustion engines, these approaches have limitations. They often fail to consider multiple faults, lack the ability to isolate them, and do not involve a comprehensive comparison of ML models. This study addresses these limitations by applying and comparing fourteen ML models to detect nine distinct fault scenarios in turbocharged spark-ignited engine systems. The models were trained using features derived from a simulation testbed specifically designed for a turbocharged spark-ignited engine system. Five features were extracted from the simulation variables and used as inputs for the models. Performance evaluation was conducted, demonstrating the models' effectiveness in identifying and classifying various fault scenarios. Among the evaluated algorithms, the Random Forest classifier hiperparameters were optimized by the Tree-structured Parzen Estimator algorithm and exhibited the best performance across all metrics. This model achieved accuracy, recall, precision, and F1 score of 97.61%, 97.61%, 97.68%, and 97.52%, respectively. These results are highly encouraging and make future investigations into the application of these methods to other engine systems.*

Keywords: *Fault Detection, Turbocharged Spark-Ignited Engine, Machine Learning, Artificial Intelligence.*

1. INTRODUCTION

Fault diagnosis in automotive systems is critical for safety and cost savings. As these systems grow in complexity, with more electronic components, the risk of faults increases. In modern engines, multiple components are monitored separately, pushing for innovative diagnostic methods that can pinpoint faults more precisely. As a result, machine learning (ML) has emerged as a valuable tool in fault diagnosis for combustion engines. It allows for the reduction of expensive sensors while maintaining efficiency. Recent studies have employed support vector machines (SVM) (Naveen Venkatesh *et al.*, 2022; Singh *et al.*, 2019) and deep artificial neural networks (Lima *et al.*, 2021; Singh *et al.*, 2022) for detecting engine misfire faults. However, a gap exists: few ML models have been deeply explored, and there's a lack of thorough comparison and evaluation for detecting flaws in spark-ignited engine systems. Moreover, the decision-making mechanisms of these models remain unexplored.

In response, our study evaluates ML classifier models across nine fault scenarios in a turbocharged spark-ignited engine system. We applied various algorithms, including Random Forest (RF), Extra Trees (ET), Category Boosting (CatBoost), Extreme Gradient Boosting (XGBoost), Light Gradient Boosting Machine (LightGBM), Gradient Boosting classifier (GBC), Decision Tree classifier (DT), Quadratic Discriminant Analysis (QDA), K-Nearest Neighbors classifier

(KNN), Logistic Regression (LR), SVM with a linear kernel, Linear Discriminant Analysis (LDA), Naive Bayes (NB), and Adaptative Boost (AdaBoost) classifier.

We assessed each algorithm using Stratified k-fold cross-validation with ten splits, ranking them by accuracy. We further enhanced our analysis with recall, precision, and F1 score metrics. The top-performing model was deeply analyzed using a confusion matrix, and we determined the most influential features contributing to its classifications.
The main contributions of this study are:

- Apply and compare the ML models, previously cited, to classify nine isolated fault scenarios of a simulated turbocharged spark-ignited engine system.
- Extensively investigate the best-performing algorithm in each class through the confusion matrix and evaluate each dataset's feature importance.

The remainder of this paper is organized as follows. Section 2 covers related work about this study. Section 3 provides an overview of the dataset description and explanatory data analysis. Section 4 presents the models applied to detect and classify faults, and accuracy, precision, recall, and F1-score metrics used to evaluate performance. In Section 5, the discussion of the results achieved by the algorithms are presented, followed by the extension of the best-performing model analysis. Finally, Section 6 concludes this study with final remarks and future directions.

2. RELATED WORK

Praveenkumar *et al.* (2014) addressed the use of vibration signal for automated fault diagnosis of gearbox. The authors extracted the statistical features from the acquired vibration signals, which served as an input to the SVM model for fault identification. The performance of the fault identification system using vibration signals were discussed and compared showing promising classification results.

In an innovative approach, Roy and Mohanty (2017) proposed a novel approach to replace the conventional direct measurement of in-cylinder pressure in engine firing detection. Their method involved the utilization of a piezo-electric sensor and a developed algorithm based on complementary ensemble empirical mode decomposition, followed by the short-time Fourier transform. By analyzing the frequency spectrum obtained from the sensor data, the algorithm was able to detect engine firing as well as rotational frequencies and harmonics. Experimental results demonstrated that the amplitude around the firing frequency was smaller in the laser vibrometer signal compared to the instantaneous angular speed signal. The proposed method was applied and evaluated on a single cylinder Honda G-300 gasoline engine. The results showcased the effectiveness of the algorithm in efficiently detecting engine firing. That research introduced a promising alternative to the traditional approach of in-cylinder pressure measurement, offering potential advancements in engine diagnostics and monitoring.

Hashim *et al.* (2020) presented the utilization of the wavelet packet technique on vibration signals from a spark ignition engine to diagnose defects. Vibration signals are captured at pistons 2 and 3 to ensure sufficient intensity, and the comparison between normal and faulty states is performed using wavelet packet analysis. The authors optimized the level of decomposition using the Shannon entropy method and calculate the maximum energy to Shannon entropy ratio for various wavelets. Finally, an artificial neural network and least square-SVM were employed, utilizing features such as maximum, minimum, mean, and standard deviation to accurately classify faults. That approach presents a comprehensive framework for defect diagnosis in spark ignition engines, combining wavelet packet analysis, wavelet selection based on maximum energy to Shannon entropy ratio, and classification models.

Turning to the issue of misfire faults, Singh *et al.* (2019) suggested a method for detecting cylinder misfire based on sound quality metrics of the radiated sound, which were measured near the cylinder block or exhaust tailpipe. The authors utilized the SVM classifier to classify metrics such as loudness, roughness, and engine and tailpipe fluctuation strength. The algorithm achieved a high level of accuracy, correctly predicting misfiring with 94% test accuracy. Additionally, the prediction of engine vibration statistical features achieved 82% test accuracy, while the prediction of engine/tailpipe sound pressure level achieved 85% test accuracy

Kn *et al.* (2020) undertook a study on the fault diagnosis of internal combustion engine gearboxes. They opted for a fusion of vibration signals, signal processing techniques, and machine learning models for diagnosis. By leveraging a J48 decision tree, their research effectively identified gearbox conditions with a classification accuracy of over 85%, indicating the promise of such a combined approach in fault detection.

3. MATERIALS

The simulation environment is centered around a 1.8L 4-cylinder single turbocharged spark-ignited (TCSI) petrol engine, following the design and specifications proposed by Ng *et al.* (2020). This engine is meticulously designed to replicate real-world scenarios, enabling accurate data collection and analysis. Figure 1 provides a detailed schematic view of the TCSI engine. In this depiction, the color red highlights the areas where hotter gases emerge as a result of combustion reactions taking place within the engine block. Conversely, the color blue designates areas operating at normal temperature.

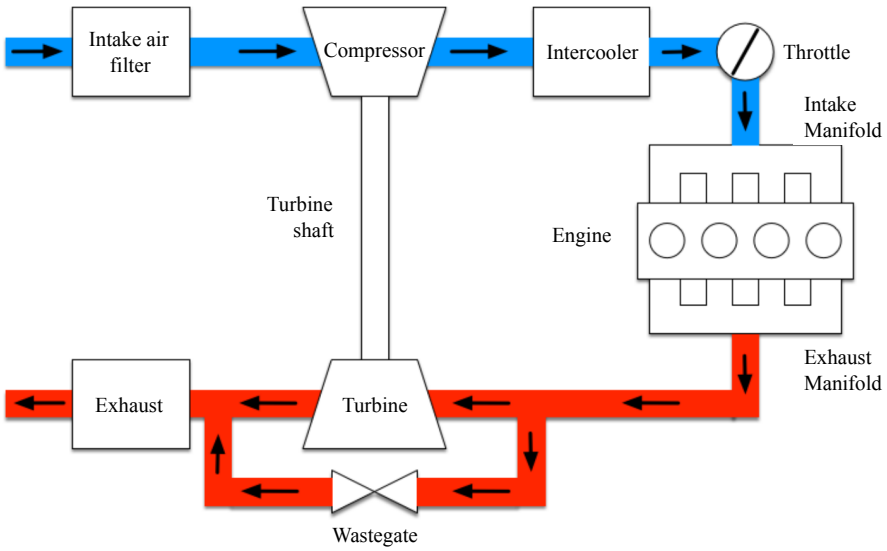


Figure 1: Schematic representation of TCSI petrol engine and subsystems: Visualizing heat distribution and combustion reactions in engine block.

The system consists of the following subsystems, air filter, compressor, intercooler, throttle, intake manifold, engine block, exhaust manifold, turbine, wastegate, and exhaust system. The engine system is modeled using dynamical equations that describe the airflow through the subsystems in the engine. These equations are derived based on the mean value engine model (MVEM) for a single turbocharged petrol engine, as reported in Eriksson (2007). The dataset considers sensor, actuator, and variable faults in different engine system parts. The original dataset includes 11 faults, from which 6 are variable faults ($f_{p_{af}}, f_{C_{vol}}, f_{W_{af}}, f_{W_{th}}, f_{W_c}, f_{W_{ic}}$), one is the actuator measurement fault ($f_{x_{th}}$), and four are sensor measurement faults ($fy_{W_{af}}, fy_{p_{im}}, fy_{p_{ic}}, fy_{T_{ic}}$) defined in Table 1.

To simulate all fault scenarios, Ng *et al.* (2020) developed a graphical user interface using MATLAB/Simulink, utilizing the environmental protection agency federal test procedure (FTP-75) as the standard driving cycle in the industrial setting. The simulation generated various arrays that encompassed the following data:

- Reference engine speed, ω_{eREF} [N.m];
- Reference engine torque, Tq_{eREF} [N.m];
- Five actuator measurements to the engine: control inputs for throttle position area [m^2] and wastegate ([0 . . . 1]), engine speed [rad/s], ambient temperature [K], and pressure [Pa];
- Nine sensor measurements from the engine: Temperatures [K] for the compressor, intercooler, and intake manifold, pressures for the compressor [Pa], intercooler, intake manifold, and exhaust manifold), air filter mass flow [kg/s], and engine torque [rad/s]);
- Thirteen states of the engine system, temperatures [K] and pressures [Pa] for the air filter, compressor, intercooler, intake manifold, exhaust manifold, and turbine, as well as, the turbine speed [rad/s];
- Normalized data of the faults, the selected induced defect would have non-zero data, except when a 'Fault-free' scenario is specified where all faults would have data of value zero;
- Residuals based on the current sensors setup, $rT_c, rp_c, rT_{ic}, rp_{ic}, rT_{im}, rp_{im}, rW_{af}, rTq_e, rp_{em}$.

The air filter plays a crucial role in the engine system by allowing ambient air to enter while preventing the ingress of abrasive particulate matter that could harm the engine block. Once inside, the filtered air undergoes compression to enhance its volumetric flow, pressure, and temperature. To ensure optimal performance, the compressed air is then cooled in the intercooler component while maintaining its mass flow velocity. The throttle serves as a control mechanism, regulating the intake manifold pressure and determining the quantity of fuel injected into the engine. The resulting mixture of air and fuel is evenly distributed among the four cylinders through the intake manifold. Within the engine block, this combustion mixture is ignited, generating the necessary torque for mechanical work. The gases produced as a byproduct of the combustion process are directed towards the turbine and the wastegate. The turbine harnesses energy from these exhaust gases in the exhaust manifold, utilizing it to power the compressor. In parallel, the wastegate valve provides a means to bypass the turbine, allowing for control over the power delivered by the turbocharger. Finally, the exhaust system facilitates the safe discharge of the gases from the engine system to the ambient environment.

3.1 Exploratory Data Analysis

The individual arrays were combined into a single data frame, excluding the $f_{W_{af}}$ and $f_{yp_{im}}$ faults due to their continuous values that couldn't be categorized. To ensure uniformity across the dataset, all features were normalized using the z-score normalization method. This technique adjusts the values of each feature such that they have a mean of zero and a standard deviation of one, which standardizes the dataset. The z-score normalization is defined by the following equation:

$$Z = \frac{x - \mu}{\sigma}, \tag{1}$$

where Z is the normalized value, x is the observed value of the feature, μ represents the mean of the samples in the feature, and σ represents the standard deviation.

After normalizing the data, a sliding window size of 500 was set, and within each window, the mean, skewness, kurtosis, maximum, and minimum values were computed. Furthermore, the fault category was included as the target variable, resulting in the generation of the final dataset. To facilitate further analysis, label encoding was applied to the nine different fault types. This technique transformed non-numerical labels into numerical representations. Table 1 presents the resulting target mapping, along with descriptions and severity levels, for each fault scenario.

Table 1: Simulation results: Fault types, descriptions, and severity in TCSI engine.

Class	Fault scenario	Description	Severity
0	No fault	Engine operating under normal conditions	None
1	$f_{C_{\text{vol}}}$	Intake-valve timing stuck at an arbitrary position	High
2	f_{W_c}	Air leakage between the compressor and the intercooler	High
3	$f_{W_{\text{ic}}}$	Air leakage between the intercooler and the throttle	High
4	$f_{W_{\text{th}}}$	Air leakage after the throttle in the intake manifold	High
5	$f_{p_{\text{af}}}$	Loss of pressure in the air filter	Medium
6	$f_{x_{\text{th}}}$	Throttle position actuator error	Medium
7	$f_{yT_{\text{ic}}}$	Intercooler temperature sensor fault	Low
8	$f_{yW_{\text{af}}}$	Air filter flow sensor fault	Low
9	$f_{yp_{\text{ic}}}$	Intercooler pressure sensor fault	Low

The label encoder resulted in ten classes, encompassing nine fault types and one class denoting normal operating conditions.

- Class zero corresponds to the previously mentioned scenario, indicating no fault severity.
- Class one represents a high-severity fault characterized by the intake-valve timing being stuck at an arbitrary position, resulting in sudden pulses occurring for 30 seconds every 150 seconds.
- Class two denotes a high-severity fault related to air leakage between the compressor and the intercooler, activated when the leakage accounts for at least 20% of the airflow.
- Similarly, class three denotes a high-severity fault associated with air leakage between the intercooler and the throttle.
- Class four pertains to a high-severity fault concerning air leakage after the throttle in the intake manifold, concluding the analysis of high-severity faults.
- Class five represents a medium-severity fault activated when there is a loss of pressure in the air filter.
- Class six corresponds to a medium-severity fault associated with a throttle position actuator error, triggered when a 20% flow error is detected.
- Class seven denotes a low-severity fault that occurs when there is a flaw in the intercooler temperature sensor, causing a 20-K offset.
- Class eight signifies a low-severity fault related to air leakage between the air filter and the compressor, activated by a 20% flow error in the air filter.
- Finally, class nine represents a low-severity fault triggered by a 20% pressure deviation in the intercooler pressure sensor.

4. METHODOLOGY

This section delves into the comprehensive methods employed for fault detection and classification. We outline the procedural steps and provide a detailed overview of the experimental setup. Furthermore, we discuss the evaluation metrics that were strategically chosen to offer a comparison and evaluation of the algorithms' performance.

4.1 Classical machine learning approaches

Introduced by Boser *et al.* (1992), the Support Vector Machine (SVM) algorithm separates data points of different classes using a hyperplane. In two dimensions, this hyperplane reduces to a line, known as the decision boundary. It's described by:

$$\mathbf{w} \cdot \mathbf{x} + b = 0, \tag{2}$$

where $\mathbf{w}$ represents the weight vector, $\mathbf{x}$ is a data point, and b is the bias. The margin M is expressed as $\frac{2}{\|\mathbf{w}\|}$. The aim is to maximize this margin by fine-tuning $\mathbf{w}$ and b.

K-Nearest Neighbors (KNN), as devised by Cover and Hart (1967), assigns a class label to a given query point based on its nearest neighbors. The Euclidean distance between a query point q and another point p is:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}, \tag{3}$$

where $d(\mathbf{p}, \mathbf{q})$ indicates the distance between $\mathbf{p}$ and $\mathbf{q}$, and p_i and q_i are their respective i-th coordinates. n represents the dimensionality of the data.

Leo Breiman (1984) proposed the Classification and Regression Decision Trees (CART) algorithm, suitable for both classification and regression. CART seeks to minimize node impurity, often employing the Gini impurity, which for binary classification is:

$$G(p) = 2p(1 - p), \tag{4}$$

where p denotes the probability of selecting an item from one class.

Ensemble methods combine multiple algorithms to improve prediction accuracy and come in two main types: parallel and sequential. RF (Breiman, 2001) and ET (Geurts *et al.*, 2006) represent parallel approaches, with the former using bootstrapped samples and bagging, and the latter applying distinct samples with randomized features. Sequentially, AdaBoost (Freund and Schapire, 1996) develops models by adapting earlier ones to influence subsequent models. GBC (Mason *et al.*, 1999) enhances this adaptiveness by using any differentiable loss function. XGBoost (Chen and Guestrin, 2016) fine-tunes predictions through gradient-boosted trees, while CatBoost (Prokhorenkova *et al.*, 2018) focused on categorical features using "ordered boosting". LightGBM (Ke *et al.*, 2017), on the other hand, optimizes computations via Gradient-Based One Side Sampling (GOSS).

LDA, QDA (McLachlan, 1992), and NB (Rish, 2001) represent generative techniques, modeling class-specific input distributions. While LDA linearly amalgamates features for class separation, QDA envisions each class through unique Gaussian distributions. NB employs Bayes' theorem for estimating class probabilities, given by:

$$P(C_k|\mathbf{x}) = \frac{P(C_k|\mathbf{x})P(C_k)}{P(x)}, \tag{5}$$

and LDA's decision boundary is:

$$\mathbf{w}^T\mathbf{x} + b = 0. \tag{6}$$

Logistic Regression (Dreiseitl and Ohno-Machado, 2002), a supervised probabilistic classifier, calculates event probabilities as:

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}, \tag{7}$$

where $P(y = 1|\mathbf{x})$ represents the probability of class 1 given input $\mathbf{x}$, and $\mathbf{w}$ is the weight vector.

Lastly, the Ridge Classifier (Hoerl and Kennard, 1970) incorporates regularization to mitigate overfitting, optimizing:

$$J(\mathbf{w}) = \sum_{i=1}^{n} (y_i - \mathbf{w}^T\mathbf{x}_i)^2 + \lambda||w||^2, \tag{8}$$

where $J(\mathbf{w})$ is the cost function, y_i the true label of the i-th data point, $\mathbf{x}_i$ the i-th data point, and λ the regularization hyperparameter.

4.2 Classification metrics

Many works use the accuracy, precision, recall and F1 score as metrics to measure performance in classification tasks. (Piratelo *et al.*, 2022; Hicks *et al.*, 2022; Neupane and Seok, 2020). Accuracy is the most common metric used to evaluate performance and is defined by:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{9}$$

where accuracy is calculated by the sum of the true positives (TP) with the true negatives (TN), divided by the exact sum of the previous terms with the sum of the FP false positives (FP) and false negatives (FN).

Related to the previous metric, the Precision is a measure of statistical variability that describes how close the measurements are to each other, given by:

$$Precision = \frac{TP}{TP + FP}, \tag{10}$$

where Precision is calculated by the TP samples divided by the sum of TP with FP samples. In addition, Recall metrics stands for the fraction of relevant instances that were retrieved and is given by:

$$Recall = \frac{TP}{TP + FN}, \tag{11}$$

where the Recall is calculated by the TP samples divided by the sum of TP with FN samples. Lastly, the F1 score metric represents both precision and recall in one value, hence it is the harmonic mean of the precision and recall, and is given by:

$$F1\ \ score = \frac{2 * Precision * Recall}{Precision + Recall}. \tag{12}$$

4.3 Experimental design setup

The experiments were carried out utilizing the PyCaret (Ali, 2023) machine learning library in the Python 3.10 environment, hosted on the Google Colab platform. For the tuning of hyperparameters, we turned to the Optuna (Akiba *et al.*, 2019) software framework, a prominent tool for automatic hyperparameter optimization. The precise search space for hyperparameters is detailed in the tables provided in the appendix. Specifically, we employed the Sampler with the Tree-structured Parzen Estimator (Ozaki *et al.*, 2020) algorithm to efficiently explore the hyperparameter space. To robustly evaluate the performance of our algorithms, we implemented Stratified *k*-fold cross-validation with ten distinct folds. It's worth noting that the data partitioning was done to allocate 80% (equivalent to 3,469 samples) for training, while the remaining 20% (or 868 samples) was reserved strictly for testing purposes.

5. RESULTS AND DISCUSSION

This section presents the results from the experiments conducted in section 4, including each classifier's average performance and in each test fold. In addition, to further analyze the best-performing model, we extracted the confusion matrix and the feature importance to have further insights into the best-performing model's performance. Table 2 presents the machine learning models' mean performance considering the accuracy, recall, precision, and F1 score metrics in descending order.

Table 2: ML models' accuracy, recall, precision, and F1 score mean metrics calculated under the 10 Stratified *k*-folds cross-validation. The mean of the accuracy measure was used for ranking the models.

Symbol	Model	Accuracy	Recall	Precision	F1 score
RF	**Random Forest Classifier**	$\mathbf{0.9761 \pm 0.0072}$	$\mathbf{0.9761 \pm 0.0072}$	$\mathbf{0.9768 \pm 0.0072}$	$\mathbf{0.9752 \pm 0.0076}$
ET	Extra Trees Classifier	0.9755 ± 0.0078	0.9755 ± 0.0078	0.9763 ± 0.0076	0.9744 ± 0.9744
XGBoost	Extreme Gradient Boosting	0.9746 ± 0.0086	0.9746 ± 0.0086	0.9754 ± 0.0085	0.9737 ± 0.0090
CatBoost	CatBoost Classifier	0.9746 ± 0.0070	0.9746 ± 0.0070	0.9750 ± 0.0069	0.9733 ± 0.0075
LightGBM	Light Gradient Boosting Machine	0.9743 ± 0.0106	0.9743 ± 0.0106	0.9750 ± 0.0104	0.9733 ± 0.0112
GBC	Gradient Boosting Classifier	0.9703 ± 0.0077	0.9703 ± 0.0077	0.9715 ± 0.0070	0.9689 ± 0.0086
DT	Decision Tree Classifier	0.9686 ± 0.0083	0.9686 ± 0.0083	0.9689 ± 0.0086	0.9672 ± 0.0090
QDA	Quadratic Discriminant Analysis	0.9294 ± 0.0100	0.9294 ± 0.0100	0.9339 ± 0.0091	0.9291 ± 0.0093
KNN	K Neighbors Classifier	0.9265 ± 0.0155	0.9265 ± 0.0155	0.9287 ± 0.0160	0.9245 ± 0.0196
LR	Logistic Regression	0.8838 ± 0.0148	0.8838 ± 0.0148	0.8855 ± 0.0177	0.8742 ± 0.0166
SVM	SVM - Linear Kernel	0.8443 ± 0.0162	0.8443 ± 0.0162	0.8364 ± 0.0184	0.8325 ± 0.0153
LDA	Linear Discriminant Analysis	0.8412 ± 0.0100	0.8412 ± 0.0100	0.8355 ± 0.0102	0.8343 ± 0.0084
NB	Naive Bayes	0.7489 ± 0.0162	0.7489 ± 0.0163	0.7606 ± 0.0166	0.7467 ± 0.0144
Ridge	Ridge Classifier	0.7091 ± 0.0093	0.7091 ± 0.0093	0.6522 ± 0.0056	0.6461 ± 0.0110
AdaBoost	Ada Boost Classifier	0.5515 ± 0.1100	0.5515 ± 0.1100	0.6199 ± 0.0429	0.5500 ± 0.0945
Dummy	Dummy Classifier	0.3851 ± 0.0015	0.3851 ± 0.0015	0.1483 ± 0.0015	0.2142 ± 0.0015

The results obtained from the experimental section showcase a significant disparity in the performance of different classifiers, as illustrated in Table 2. Evaluating the quantitative aspects of the performance metrics, we observe that the RF classifier consistently outperforms other models, boasting an accuracy of 97.61% with a minimal standard deviation of 0.0072, suggesting a robust and stable performance. The model with the widest spread in performance is AdaBoost, with a significant standard deviation of 0.1100 in accuracy.

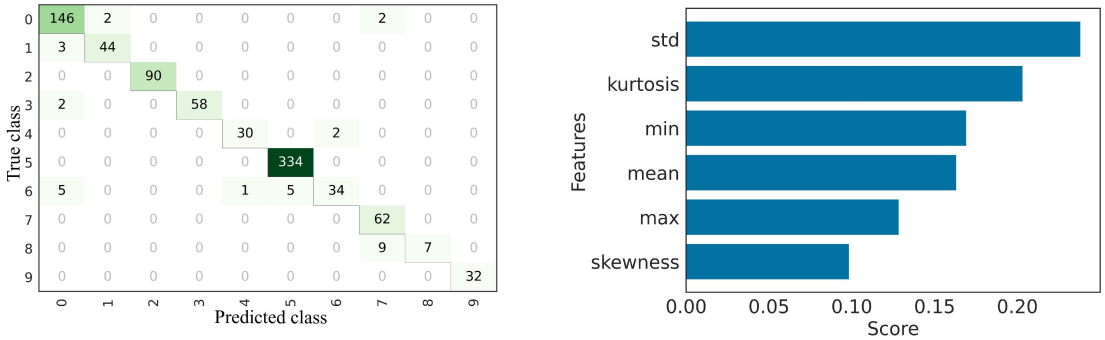
Elucidating further, the classifiers with the highest mean accuracies are ensemble models: RF, ET, XGBoost, LightGBM, and GBC. The deviation from the top performer, the Random Forest, is not drastically significant, as the difference in accuracy is a mere 0.061% with Extra Trees and 0.15% with XGBoost. However, there's a noteworthy observation concerning the F1 scores. The Decision Tree classifier, despite its 6th rank in accuracy, exhibits an F1 score that is 0.82% lower than that of Random Forest, indicating possible imbalances in its precision and recall. This implies that while accuracy might be comparable, the quality of predictions can vary.

Further down the rank, more traditional models like SVM, LR, and QDA show reduced performance. The SVM, for instance, though ranked 10th in accuracy, has a precision lower by approximately 14.08% compared to RF. Such a deviation might indicate that while SVM can correctly classify a substantial portion of instances, it might also be misclassifying a significant number of instances as false positives.

Interestingly, the Naive Bayes classifier, known for its simplicity and efficiency, manifests a substantial gap of 23.28% in accuracy from the Random Forest. This could be attributed to the underlying assumption of feature independence in Naive Bayes, which might not hold in this particular dataset. The Dummy classifier serves as a baseline, with its notably poor performance underscoring the efficacy of other models. Its remarkably low F1 score of 21.42% further emphasizes the model's inadequacy in handling both precision and recall.

The quantitative nuances highlight the superiority of ensemble models in this context, with their ability to merge predictions from multiple models, leading to enhanced generalization. However, the deviations in precision, recall, and F1 scores across models emphasize the importance of a comprehensive evaluation rather than a sole reliance on accuracy.

In order to extensively study the best performing algorithm's performance, Figure 2 shows the confusion matrix and feature importance plots of the model.



(a) (b)

Figure 2: Random forest algorithm classification performance metrics: (a) confusion matrix and (b) feature importance.

According to the confusion matrix plot, the algorithm most accurately predicted fault types for classes 2, 5, 7, and 9, also denoted as $f_{p_{\text{af}}}$, f_{W_c}, $f_{yT_{\text{ic}}}$, and $f_{yp_{\text{ic}}}$. These fault types were detected with a 100% success rate, with occurrences in the dataset numbering 90, 334, 62, and 32 respectively. Conversely, the algorithm found class 8 to be the most challenging to classify; of its 16 occurrences, only seven were correctly identified. Following this, class 6 was the next most difficult fault type for the model, with only 34 correct identifications out of 45 occurrences. Classes 1, 3, and 4 comprised 47, 60, and 32 samples respectively. For class 1, the random forest model correctly identified 44 samples, but confused three samples with class 0, which represents normal conditions. Similarly, class 3 saw 58 correct identifications, with two samples mistakenly classified as class 0. For class 4, 30 samples were correctly identified, while two were misclassified as belonging to class 6.

In terms of feature importance, XGBoost was employed to compute the significance of each feature. This importance was averaged across all folds, with the score representing the cumulative change in loss for each fold across all trees. This was computed using a single round, as opposed to multiple rounds in the previous analysis. Referring to Figure 2(b), the standard deviation emerged as the most critical feature with a score of 0.2380, followed closely by the kurtosis of the distribution, which scored 0.2031. The variability in the signal, as indicated by the standard deviation, was the most impactful feature in the detection of all fault types. Additionally, the minimum and mean value features ranked third and fourth in importance with scores of 0.1691 and 0.163, respectively. In contrast, the maximum value and skewness proved to be the least significant in the predictive model, with scores of 0.1284 and 0.0983, respectively.

From the analysis, the standard deviation, which emerged as the most pivotal feature, underlines the importance of variability in the signal. Such variability can be a clear indicator of any anomalies or faults in the system. A consistent system under no-fault conditions would generally exhibit a low standard deviation, whereas any disruptions or unexpected behaviors would naturally increase this variability.

The kurtosis of the distribution, as the second most crucial feature, indicates the nature of the fault. The presence of outliers or sudden spikes in readings is typical of abrupt and significant faults. Such sharp changes can easily distort the tail behavior of a distribution, and thus, kurtosis becomes a key identifier.

The mean and minimum values being ranked third and fourth reflect the general behavior and lower extremities of the dataset. A fault might cause an overall change in the mean behavior or might result in occasional low dips – indicating issues in the system's performance. The importance of these features shows that the model does not solely rely on sudden, anomalous changes but also on the overall behavior and trends in the data.

6. CONCLUSION

This study introduced and evaluated 14 machine learning model to detect nine distinct faults in a simulated 1.8L 4-cylinder single TCSI engine: $f_{C_{\text{vol}}}$, f_{W_c}, $f_{W_{\text{ic}}}$, $f_{W_{\text{th}}}$, $f_{p_{\text{af}}}$, $f_{x_{\text{th}}}$, $f_{yT_{\text{ic}}}$, $f_{yW_{\text{af}}}$, and fyp_{ic}. To extract features for our models, we employed z-score normalization and adopted a sliding window of size 500. From this window, we derived attributes such as standard deviation, mean, skewness, kurtosis, maximum, and minimum values. These features trained algorithms including RF, ET, XGBoost, LightGBM, GBC, DT, QDA, KNN, LR, SVM (with a linear kernel), LDA, naive Bayes, ridge, and AdaBoost.

Evaluation metrics such as accuracy, recall, precision, and F1 score revealed the Random Forest as the standout algorithm, achieving 97.61% in both accuracy and recall, 97.68% in precision, and an F1 score of 0.9752. These metrics were appraised using a Stratified *k*-fold cross-validation method with ten partitions. Notably, extra trees and XGBoost secured the 2nd and 3rd positions in accuracy performance, achieving 97.55% and 97.46%, respectively. These results underscore the capability of the presented models in detecting and categorizing the enumerated faults with over 95% accuracy.

A deeper exploration into the Random Forest model using the confusion matrix disclosed its aptitude in classifying faults for classes 2, 5, 7, and 9 with 100% accuracy. However, out of 47 samples for class 1, three were misclassified, and two samples from class 3 and nine from class 8 were also mistaken. The model faced difficulties with class 6, often confusing it with classes 0, 4, and 5, making it the most challenging class for classification. Utilizing XGBoost to determine feature importance pinpointed the standard deviation as the paramount feature, scoring 0.2381, followed by the kurtosis feature at 0.2031. The Gini index facilitated the selection of split points.

For our feature extraction process, we settled on a window size of 500. Future endeavors could explore varied window sizes to discern any impact on classification performance. Additionally, novel features, such as peak-to-peak, variance, and mean root square, might be incorporated to enrich the dataset. On the algorithmic front, delving into deep learning models—like 1-dimensional convolutional neural networks and long short-term memory layers—may offer avenues to further refine fault classification.

7. REFERENCES

Akiba, T., Sano, S., Yanase, T., Ohta, T. and Koyama, M., 2019. "Optuna: A next-generation hyperparameter optimization framework". In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Association for Computing Machinery, Anchorage, United States of America, p. 2623–2631.

Ali, M., 2023. *PyCaret: An open source, low-code machine learning library in Python*. URL `https://www.pycaret.org`. PyCaret version 3.0.0.

Boser, B.E., Guyon, I.M. and Vapnik, V.N., 1992. "A training algorithm for optimal margin classifiers". In *Proceedings of the 5th Annual Workshop on Computational Learning Theory*. Association for Computing Machinery, Pittsburgh, United States of America, p. 144–152.

Breiman, L., 2001. "Random forests". *Machine Learning*, Vol. 45, No. 1, pp. 5–32.

Chen, T. and Guestrin, C., 2016. "Xgboost: A scalable tree boosting system". In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, New York, NY, United States of America, p. 785–794.

Cover, T. and Hart, P., 1967. "Nearest neighbor pattern classification". *IEEE Transactions on Information Theory*, Vol. 13, No. 1, pp. 21–27.

Dreiseitl, S. and Ohno-Machado, L., 2002. "Logistic regression and artificial neural network classification models: a methodology review". *Journal of Biomedical Informatics*, Vol. 35, No. 5, pp. 352–359.

Eriksson, L., 2007. "Modeling and control of turbocharged si and di engines". *Oil & Gas Science and Technology*, Vol. 62, No. 4, pp. 523–538.

Freund, Y. and Schapire, R.E., 1996. "Experiments with a new boosting algorithm". In *Proceedings of the 13th Interna-*

tional Conference on International Conference on Machine Learning. Morgan Kaufmann Publishers Inc., Bari, Italy, p. 148–156.

Geurts, P., Ernst, D. and Wehenkel, L., 2006. "Extremely randomized trees". *Machine Learning*, Vol. 63, No. 1, pp. 3–42.

Hashim, M., Nasef, M., Kabeel, A. and Ghazaly, N.M., 2020. "Combustion fault detection technique of spark ignition engine based on wavelet packet transform and artificial neural network". *Alexandria Engineering Journal*, Vol. 59, No. 5, pp. 3687–3697.

Hicks, S.A., Strümke, I., Thambawita, V., Hammou, M., Riegler, M.A., Halvorsen, P. and Parasa, S., 2022. "On evaluation metrics for medical applications of artificial intelligence". *Scientific Reports*, Vol. 12, No. 1, p. 5979.

Hoerl, A.E. and Kennard, R.W., 1970. "Ridge regression: Biased estimation for nonorthogonal problems". *Technometrics*, Vol. 12, No. 1, pp. 55–67.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. and Liu, T.Y., 2017. "Lightgbm: A highly efficient gradient boosting decision tree". In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Curran Associates Inc., Red Hook, United States of America, p. 3149–3157.

Kn, R., Kumar, H., Gn, K. and Kn, G., 2020. "Fault diagnosis of internal combustion engine gearbox using vibration signals based on signal processing techniques". *Journal of Quality in Maintenance Engineering*, Vol. 27, No. 2, p. 385 – 412.

Leo Breiman, Jerome Friedman, C.J.S.R.O., 1984. *Classification and Regression Trees*. Chapman and Hall/CRC, Boca Raton, United States of America.

Lima, T.L.d.V., Filho, A.C.L., Belo, F.A., Souto, F.V., Silva, T.C.B., Mishina, K.V. and Rodrigues, M.C., 2021. "Non-invasive methods for fault detection and isolation in internal combustion engines based on chaos analysis". *Sensors*, Vol. 21, No. 20, p. 6925.

Mason, L., Baxter, J., Bartlett, P. and Frean, M., 1999. "Boosting algorithms as gradient descent". MIT Press, Denver, United States of America, p. 512–518.

McLachlan, G.J., 1992. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley Interscience, Hoboken, United States of America.

Naveen Venkatesh, S., Chakrapani, G., Senapti, S.B., Annamalai, K., Elangovan, M., Indira, V., Sugumaran, V. and Mahamuni, V.S., 2022. "Misfire detection in spark ignition engine using transfer learning". *Computational Intelligence and Neuroscience*, Vol. 2022, p. 7606896.

Neupane, D. and Seok, J., 2020. "Bearing fault detection and diagnosis using case western reserve university dataset with deep learning approaches: A review". *IEEE Access*, Vol. 8, pp. 93155–93178.

Ng, K.Y., Frisk, E., Krysander, M. and Eriksson, L., 2020. "A realistic simulation testbed of a turbocharged spark-ignited engine system: A platform for the evaluation of fault diagnosis algorithms and strategies". *IEEE Control Systems Magazine*, Vol. 40, No. 2, pp. 56–83.

Ozaki, Y., Tanigaki, Y., Watanabe, S. and Onishi, M., 2020. "Multiobjective tree-structured parzen estimator for computationally expensive optimization problems". In *Proceedings of the Genetic and Evolutionary Computation Conference*. Association for Computing Machinery, Cancún, Mexico, p. 533–541.

Piratelo, P.H.M., de Azeredo, R.N., Yamao, E.M., Bianchi Filho, J.F., Maidl, G., Lisboa, F.S.M., de Jesus, L.P., Penteado Neto, R.d.A., Coelho, L.d.S. and Leandro, G.V., 2022. "Blending colored and depth cnn pipelines in an ensemble learning classification approach for warehouse application using synthetic and real data". *Machines*, Vol. 10, No. 1.

Praveenkumar, T., Saimurugan, M., Krishnakumar, P. and Ramachandran, K., 2014. "Fault diagnosis of automobile gearbox based on machine learning techniques". *Procedia Engineering*, Vol. 97, pp. 2092–2098.

Prokhorenkova, L.O., Gusev, G., Vorobev, A., Dorogush, A.V. and Gulin, A., 2018. "Catboost: unbiased boosting with categorical features". In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018*. Montréal, Canada, pp. 6639–6649.

Rish, I., 2001. "An empirical study of the naive bayes classifier". In *IJCAI workshop on empirical methods in artificial intelligence*. Seattle, United States of America, Vol. 3, pp. 41–46.

Roy, S.K. and Mohanty, A., 2017. "Use of rotary optical encoder for firing detection in a spark ignition engine". *Measurement*, Vol. 98, pp. 60–67.

Singh, E., Kuzhagaliyeva, N. and Sarathy, S.M., 2022. "Chapter 9 - using deep learning to diagnose preignition in turbocharged spark-ignited engines". In J. Badra, P. Pal, Y. Pei and S. Som, eds., *Artificial Intelligence and Data Driven Optimization of Internal Combustion Engines*, Elsevier, pp. 213–237.

Singh, S., Potala, S. and Mohanty, A.R., 2019. "An improved method of detecting engine misfire by sound quality metrics of radiated sound". *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, Vol. 233, No. 12, pp. 3112–3124.

8. APPENDIX

Tables 3 and 4 present a delineation of the hyperparameters explored for each algorithm.

Table 3: ML models hyperparameter search space (Part 1)

Model	**Hyperparameter Search Space**
RF	*n_estimators*: [10, 300], *max_depth*: [1, 11], *min_impurity_decrease*: [1e-09, 0.5], *max_features*: [0.4, 1], *min_samples_split*: [2, 10], *min_samples_leaf*: [2, 6], *bootstrap*: [True, False], *criterion*: [gini, entropy]
ET	*n_estimators*: [10, 300], *max_depth*: [1, 11], *min_samples_split*: [2, 10], *min_samples_leaf*: [1, 5], *max_features*: [0.4, 1], *min_impurity_decrease*: [1e-09, 0.5], *criterion*: [gini, entropy], *bootstrap*: [True, False]
CatBoost	*eta*: [1e-06, 0.5], *depth*: [1, 11], *n_estimators*: [10, 300], *random_strength*: [0, 0.8], *l2_leaf_reg*: [1, 200]
LightGBM	*num_leaves*: [2, 256], *learning_rate*: [1e-06, 0.5], *n_estimators*: [10, 300], *min_split_gain*: [0, 1], *reg_alpha*: [1e-10, 10], *reg_lambda*: [1e-10, 10], *feature_fraction*: [0.4, 1], *bagging_fraction*: [0.4, 1], *bagging_freq*: [0, 7], *min_child_samples*: [1, 100]
GBC	*n_estimators*: [10, 300], *learning_rate*: [1e-06, 0.5], *subsample*: [0.2, 1], *min_samples_split*: [2, 10], *min_samples_leaf*: [1, 5], *max_depth*: [1, 11], *min_impurity_decrease*: [1e-09, 0.5], *max_features*: [0.4, 1]

The RF model considers hyperparameters such as the number of estimators, which is set between 10 to 300, the maximum depth of the tree, ranging from 1 to 11, and several other specifications like minimum impurity decrease and maximum features. It also factors in criteria options like "gini" and "entropy". Similarly, the ET model hyperparameters are quite analogous to the RF, covering the same range for the number of estimators and the maximum depth, among others.

Table 4: ML models hyperparameter search space (Part 2)

Model	**Hyperparameter Search Space**
DT	*max_depth*: [1, 16], *max_features*: [0.4, 1], *min_samples_leaf*: [2, 6], *min_samples_split*: [2, 10], *min_impurity_decrease*: [1e-09, 0.5], *criterion*: [gini, entropy]
QDA	*reg_param*: [0, 1]
KNN	*n_neighbors*: [1, 51], *weights*: [uniform, distance], *metric*: [minkowski, euclidean, manhattan]
LR	*C*: [0.001, 10]
SVM	*l1_ratio*: [1e-10, 0.999], *alpha*: [1e-10, 0.999], *eta0*: [0.001, 0.5], *penalty*: [elasticnet, l2, l1], *fit_intercept*: [True, False], *learning_rate*: [constant, invscaling, adaptive, optimal]
LDA	*shrinkage*: [0.0001, 1], *solver*: [lsqr, eigen]
NB	*var_smoothing*: [1e-09, 1]
Ridge	*alpha*: [0.001, 10]
AdaBoost	*n_estimators*: [10, 300], *learning_rate*: [1e-06, 0.5], *algorithm*: [SAMME, SAMME.R]

For the CatBoost model, the learning rate is configured to be anywhere from 1e-06 to 0.5. This model also considers the depth of trees, the total number of estimators, the degree of random strength, and the regularization term L2 leaf regulation. The LightGBM, on the other hand, works on a more expansive range. Notably, it considers the number of leaves, which has a range set from 2 to 256, and a learning rate ranging between 1e-06 to 0.5. hyperparameters like minimum split gain, feature fraction, bagging fraction, and frequency are also accounted for. The GBC has a defined search space that covers hyperparameters such as the number of estimators, learning rate, subsample, and several hyperparameters related to the depth and impurity of the trees.