Artificial Intelligence for Engineering Design, Analysis and Manufacturing

www.cambridge.org/aie

Predictive maintenance in aircraft engine maintenance using the C-MAPSS dataset: performance comparison and evaluation of machine learning classification algorithms

Hikmetcan Özcan

Computer Engineering, Kocaeli University, Türkiye

Research Article

Cite this article: Özcan H (2026). Predictive maintenance in aircraft engine maintenance using the C-MAPSS dataset: performance comparison and evaluation of machine learning classification algorithms. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **40**, e4, 1–22
https://doi.org/10.1017/S0890060426100249

Received: 27 December 2024
Revised: 02 October 2025
Accepted: 13 January 2026

Keywords:
aircraft engines; C-MAPSS; classification algorithms; machine learning; model comparison; performance analysis; predictive maintenance

Corresponding author:
Hikmetcan Özcan;
Email: hikmetcan.ozcan@kocaeli.edu.tr

Abstract

This study assesses classification-based predictive maintenance (PdM) for aircraft engines on the NASA Commercial Modular Aero-Propulsion System Simulation dataset and addresses the lack of wide-scope, unified benchmarks. PdM is cast as a short-term **binary** task – predicting whether an engine will fail within the **next 30 cycles** – and a comparison is conducted across **10** machine-learning models (Logistic Regression, Decision Tree, Random Forest, Support Vector Machine, k-Nearest Neighbor, Naïve Bayes, Extreme Gradient Boosting, LightGBM, CatBoost, and Gradient Boosting) and **3** deep-learning models (Multilayer Perceptron, Gated Recurrent Unit, and Long Short-Term Memory). A leakage-aware pipeline applies Min–Max scaling; class imbalance is handled with Synthetic Minority Over-sampling Technique where appropriate; hyperparameters are tuned via GridSearchCV/BayesSearchCV; and performance is reported with accuracy, precision, recall, F1-score, and receiver operating characteristic–area under the curve (ROC–AUC), complemented by Shapley Additive Explanations (SHAP) explainability and nonparametric significance tests. Sequence models delivered the strongest performance: **LSTM** achieved **Accuracy = 0.981** (Macro-F1 = 0.92; ROC–AUC = 0.96), and **GRU** achieved **ROC–AUC = 0.97** with **Accuracy = 0.975**. Among classical learners, **LightGBM** reached **Accuracy = 0.972** (Macro-F1 = 0.86; ROC–AUC = 0.93). These gains over weaker baselines were **statistically significant** across folds. Framing PdM as near-term failure classification yields operationally interpretable alerts. Models that explicitly capture temporal dependencies (GRU/LSTM) best track short-horizon failure dynamics, while gradient-boosted trees offer competitive, lightweight alternatives. The benchmark and analysis (including SHAP) provide a reproducible reference for model selection in aviation PdM.

Introduction

The aviation industry has become an important part of the global economy as a continuously growing and developing sector. In this industry, the ability of aircraft to operate safely and consistently is a fundamental requirement for both passenger and cargo transportation. Therefore, the maintenance of passenger and commercial aircraft is of great importance. Proper planning and management of maintenance can reduce operating costs and enhance the safety of operations.

Conventional maintenance strategies are generally time-based, requiring aircraft to undergo maintenance at regular intervals. However, this method can be expensive and result in redundant maintenance. Predictive maintenance (PdM) provides an alternative solution. Utilizing data analytics and machine learning (ML) methods, it is feasible to monitor aircraft engines and other vital components in real-time and forecast maintenance needs beforehand. This approach can lower maintenance expenses and reduce operational disruptions for aircraft.

PdM aims to predict future maintenance needs by utilizing time-series data from operational facilities, such as hangars. This approach's main goal is to accurately determine when an aircraft component requires servicing. Delaying maintenance can extend a component's usage period, but performing it too late might lead to unexpected failures, reducing the component's lifespan. Thus, improving the accuracy of predicting component lifespan has been a consistent priority. In this context, this study leverages ML, a subset of Artificial Intelligence, which focuses on analyzing large datasets to uncover complex patterns. Although the fundamental architecture and ideas were developed many generations ago, the necessary amount of data and computational power to make this possible has only recently become available.

The benefits of ML applications include reduced maintenance costs, fewer repair stations, decreased machine failures, extended spare part lifespan, reduced inventory, increased operator safety, enhanced production, and repair validation, as well as improvements in sensor accuracy, fuel level monitoring, ice detection, and overall profitability (EASA, 2015).

The primary objective of this study is to determine the effectiveness of data-driven PdM approaches over traditional methods in the maintenance of aircraft engines and to systematically compare the performance of widely used ML algorithms for this purpose. Efficient and reliable management of maintenance processes in the aviation industry is of great importance from both economic and operational perspectives. Traditional maintenance strategies are based on time-scheduled inspections carried out at regular intervals, which often result in unnecessary maintenance activities, leading to high costs and operational disruptions (Van den Bergh et al., 2013). In addition to increasing maintenance costs, this also poses critical safety risks due to the inability to detect potential failures in advance.

PdM methods aim to address these issues by analyzing real-time data collected from sensors to predict potential engine failures and schedule maintenance actions at the appropriate time. In this way, costs can be reduced, operational continuity improved, and safety risks minimized (Capodieci et al., 2020). Although various ML algorithms have been frequently studied in PdM processes for aircraft engines, the literature lacks clear and systematic comparisons that reveal which algorithms perform more effectively under which conditions and performance criteria (Celikmih et al., 2020; Singh et al., 2020).

To fill this gap, the present study comprehensively evaluates the performance of 10 different ML algorithms (Support Vector Machine [SVM], Decision Tree [DT], k-Nearest Neighbors [KNN], Extreme Gradient Boosting [XGBoost], Logistic Regression [LR], Random Forest [RF], Naïve Bayes [NB], LightGBM, CatBoost, and Gradient Boosting) and 3 learning models (MLP, GRU, and LSTM) on the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) dataset. Accordingly, the study analyzes algorithmic performance using objective metrics, such as accuracy, precision, recall, and F1-score, providing a scientifically grounded reference for PdM applications. Such systematic comparisons are critically important, as they serve as a valuable guide for optimizing maintenance strategies in a data-driven manner. Therefore, the study aims to contribute to the aviation industry both economically and in terms of safety by improving the efficiency of PdM processes.

The subsequent sections of this study will begin with Section "Motivation and contribution", which provides a comprehensive literature review, analyzing prior research and the current state of knowledge on the subject. Section "Background" will focus on the methodology by explaining the methods and analysis processes used in the research. Sections "Related work" and "Materials and methods" will present the findings and discuss the results, respectively. Section "Findings", the study's limitations, practical implications, and suggestions for future research will be presented. Finally, Section "Discussion and conclusion" will synthesize the main findings, interpret their theoretical and practical implications for predictive maintenance in aviation, and summarize the overall contributions of the study while outlining concluding remarks.

Motivation and contribution

Background on the economic and safety relevance of PdM is summarized in the Introduction; this section focuses on identified gaps and the study's specific contributions.

While existing research has made significant progress in applying ML techniques to PdM, most studies have primarily focused on the remaining useful life (RUL) estimation using regression-based approaches. These models aim to forecast the exact time-to-failure, but they often require extensive calibration and are sensitive to noise in continuous target values. Moreover, such models are typically evaluated using limited performance metrics – often relying solely on RMSE (Root Mean Square Error) or MAE (Mean Absolute Error) – without fully considering operational decision-making requirements.

More importantly, there remains a notable gap in the literature regarding classification-based PdM frameworks, especially those designed for binary failure prediction tasks, which are more aligned with real-world decision-making (e.g., "Will this component fail within the next 30 cycles?"). This binary framing reflects how maintenance teams make threshold-based interventions, such as replacing a part once the risk crosses a certain limit.

Additionally, prior works often:

- lack systematic hyperparameter tuning, leading to suboptimal performance;
- do not address class imbalance, which is a critical issue in rare failure detection tasks;
- focus on a narrow subset of ML models, omitting deep learning (DL) techniques; and
- use inconsistent preprocessing steps, making comparisons unreliable.

This study aims to bridge these gaps and offers several novel contributions to the field:

1. **Reframing PdM as a binary classification task:** Instead of predicting continuous RUL values, this study formulates the problem as a binary classification – predicting whether an engine will fail within the next 30 cycles – using the widely accepted C-MAPSS dataset. This approach offers practical interpretability and aligns more closely with aviation maintenance decision-making. The 30-cycle horizon reflects typical short-term maintenance planning windows used for shop-visit scheduling and part-replacement thresholds in practice.
2. **Comprehensive model benchmarking with DL integration:** This study compares 13 models – 10 traditional ML models (LR, DT, RF, SVM, KNN, NB, XGBoost, LightGBM, CatBoost, and Gradient Boosting) and 3 DL models (MLP, GRU, and LSTM) – to establish a broad, classification-focused PdM baseline.
3. **Advanced preprocessing and class imbalance handling:** The study applies Min–Max normalization and addresses the common but often overlooked issue of class imbalance by incorporating the Synthetic Minority Over-sampling Technique (SMOTE), particularly to enhance the sensitivity of models like RF in detecting minority (failure) classes.
4. **Systematic hyperparameter optimization using dual methods:** Both GridSearchCV and BayesSearchCV are used to optimize model performance. The study compares these methods in terms of both accuracy and computational efficiency, providing empirical guidance for future research on model tuning strategies in PdM contexts.
5. **Robust performance evaluation with multiple metrics:** Model performance is assessed using accuracy, precision, recall, and F1-score, offering a balanced view that captures both economic efficiency (via precision) and safety-critical concerns (via recall). Additionally, receiver operating characteristic–area under the curve (ROC–AUC) scores –are used to measure discriminative ability across thresholds.
6. **Model explainability via** Shapley Additive Explanations **(SHAP):** This study conducts a comprehensive SHAP analysis across all learner families, providing global interpretability that clarifies not only what is predicted but also why. The analysis supports auditability for safety-critical contexts and informs

https://doi.org/10.1017/S0890060426100249 Published online by Cambridge University Press

feature/sensor prioritization (see Section "Model explainability analysis" and Figure 3).
7. **Statistical significance of model differences:** This study applies a nonparametric repeated-measures protocol with a Friedman omnibus test followed, when significant, by Holm-corrected Wilcoxon signed-rank post-hoc comparisons, confirming that observed performance gaps are robust across folds (see Section "Statistical comparison protocol" and Tables 8–9).
8. **Cross-study benchmarking and reproducibility:** The results are systematically compared with findings from prior works (e.g., Soni et al., 2021; Al Hasib et al., 2023; Sharma et al., 2023, Melkumian, 2024), providing a performance-oriented benchmark. Unlike earlier studies, this paper ensures reproducibility by unifying preprocessing pipelines, using consistent metrics, and documenting all parameter configurations.

By addressing both technical (e.g., model variety, data balancing, and optimization) and methodological (e.g., problem framing and evaluation criteria) limitations of the existing literature, this study provides a robust and replicable framework for classification-based PdM in aviation. Its contributions support not only academic advancement in PdM research but also practical decision-making in safety-critical operational environments such as aircraft engine maintenance.

Background

The aircraft maintenance process is divided into two categories: traditional maintenance (corrective-preventive maintenance) and PdM.

Traditional maintenance

Aircraft maintenance involves routine tasks performed at specific intervals or after certain flight hours. While this ensures timely and scheduled maintenance, it can also lead to unnecessary procedures. Time-based maintenance includes four main types of checks: A, B, C, and D. These checks vary in their extent, duration, and how often they are needed.

The A-check, which is the most frequently performed, is required approximately every 65 flight hours, or about once a week. B-check maintenance, conducted every 300–600 flight hours, involves a detailed visual inspection and lubrication of all moving parts, such as ailerons and horizontal stabilizers. Major inspections, including C-check and D-check, occur every 4 years and necessitate taking the aircraft out of service for at least a month. Because of their infrequency, C-check and D-check inspections are often not included in regular maintenance planning, highlighting the dynamic nature of the aviation industry. Airlines primarily concentrate on adhering to A-check and B-check schedules within a 4-day maintenance cycle, with inspections and repairs typically conducted overnight until all required conditions are met (Van den Bergh et al., 2013).

Terminology note

In this paper, the maintenance terms are used consistently as follows: **corrective/reactive maintenance** = intervention after a failure; **preventive maintenance (PM)** = time−/cycle-based, scheduled intervention before failure; **PdM** = condition-based, prognostics-driven intervention dynamically triggered by estimated failure risk.

Corrective maintenance

This maintenance, also known as reactive maintenance, is unplanned and involves repairing equipment after a failure occurs. This maintenance type is often applied to simple equipment or components and is suitable for situations that do not impact safety but cause operational downtime (Eker, 2015). Items like home appliances and mobile devices frequently use corrective maintenance. Since the operational breakdown of these items typically does not lead to severe consequences and can be quickly repaired or replaced, it is usually more practical to address failures as they happen rather than prevent them (Xing and Marwala, 2018).

However, many complex industrial systems have various issues and weaknesses. For example, functional malfunctions in passenger aircraft are directly related to safety concerns, making corrective maintenance impractical after a failure. Therefore, proactive maintenance is crucial in complex industrial sectors, such as aviation, maritime, automotive, and large-scale industrial equipment.

Preventive maintenance

PM refers to scheduled, time- or cycle-based actions performed before any failure occurs, such as periodic inspections, adjustments, and part replacements planned on fixed intervals. The objective is to reduce the likelihood of unexpected faults and ensure safety and availability (e.g., routine A/B/C/D checks in aviation). Unlike corrective/reactive maintenance (carried out after a fault), PM is planned in advance; unlike PdM, PM does not rely on real-time condition-based prognostics to dynamically trigger interventions.

Predictive maintenance

PdM leverages sensor-driven diagnostics and prognostics to anticipate failures before they occur, thereby reducing unplanned downtime and maintenance costs. In this study, PdM is cast as a short-term, binary decision problem (failure within 30 cycles) to align with operational thresholds. The end-to-end pipeline conceptually comprises standard preprocessing, feature construction, model training, evaluation, and thresholding, and is model-agnostic – compatible with both classical ML and DL classifiers. This subsection provides only the conceptual background; prior art and methodological comparisons are synthesized in Section "Related work".

Related work

RUL-oriented regression

On turbofan and similar engine datasets, DL approaches for continuous RUL prediction are prevalent (Youness and Aalah, 2023). LSTM- and Convolutional Neural Network (CNN)–based models capture temporal dependencies and often outperform traditional methods (Sateesh Babu et al., 2016; Zheng et al., 2017). Likewise, similarity-based prognostics estimate RUL by matching degradation patterns (Wang et al., 2008). However, these studies primarily optimize regression metrics, such as RMSE, and do not align directly with operational decision thresholds (e.g., near-term failure windows). Broad classical-method surveys also exist (Mathew et al., 2017), but they remain largely within a regression framing.

Classification-based PdM

A second line of work frames risk as a binary or multiclass problem by labeling remaining cycles into ranges or by directly predicting failure/no failure, typically using RF, SVM, XGBoost, and related classifiers (Soni et al., 2021; Sharma et al., 2023). In the broader aircraft maintenance ecosystem, text and event logs have been used

https://doi.org/10.1017/S0890060426100249 Published online by Cambridge University Press

to anticipate high-priority defects (Yan and Zhou, 2017), and peripheral maintenance decisions (e.g., spare-part demand) have been modeled with conventional classifiers, such as NB and LR (Capodieci et al., 2020). Some studies apply MLP or SVM for equipment-failure prediction (Celikmih et al., 2020). Yet many of these efforts feature limited model diversity, minimal DL integration, or narrow comparisons; several focus on a small subset of classifier families (Singh et al., 2020). Comprehensive reviews of ML in manufacturing (Sarkon et al., 2022) seldom provide a systematic, side-by-side performance assessment tailored to aircraft maintenance.

Class imbalance and hyperparameter optimization

PdM datasets are typically imbalanced – early-failure examples are scarce – which calls for resampling (e.g., SMOTE), class weighting, cost-sensitive learning, or calibrated thresholds. Nevertheless, such techniques are not consistently or systematically applied, and hyperparameter search is often limited. For instance, some classification studies report results without DL components and without comprehensive Grid/Bayesian search (Soni et al., 2021; Sharma et al., 2023) even broad regression comparisons frequently treat optimization and balancing as secondary (Mathew et al., 2017). These gaps hinder alignment with operational objectives and limit generalizability across metrics such as precision, recall, F1-score, and ROC–AUC.

Synthesis and research gaps

- **GAP-1:** Within classification-based PdM, wide-scope benchmarks under a *unified* protocol are scarce. *This study response:* A single pipeline benchmarking 10 classical ML models (LR, DT, RF, SVM, KNN, NB, XGBoost, LightGBM, CatBoost, and Gradient Boosting) and 3 DL models (MLP, GRU, and LSTM) with shared preprocessing and evaluation.
- **GAP-2:** Limited DL integration together with systematic class balancing and hyperparameter optimization in comparative evaluations. *This study response:* Inclusion of MLP, *SMOTE* for class balance, dual tuning (*GridSearchCV* + *BayesSearchCV*), and multi-metric reporting (accuracy, precision, recall, F1-score, and ROC–AUC).

A consolidated side-by-side summary appears in Table 1. In contrast, the present study distinguishes itself through a comprehensive benchmarking framework involving 10 traditional ML models and 3 DL models. It incorporates advanced methodological elements, such as hyperparameter optimization via GridSearchCV and BayesSearchCV, class balancing with SMOTE, and multi-metric evaluation including accuracy, precision, recall, F1-score, and ROC–AUC. Additionally, the classification task is explicitly framed as short-term failure prediction within 30 cycles, offering an operationally meaningful and relatively underexplored perspective in PdM research.

Materials and methods

The methodology of this study is based on a systematic comparison of the performance of ML algorithms on the widely accepted C-MAPSS dataset, with the aim of predicting maintenance needs for aircraft engines. Initially, data normalization was performed using the Min–Max scaling method, and engine failure prediction was formulated as a binary classification problem. Dataset partitioning followed the official C-MAPSS train/test split, and hyperparameters were tuned via cross-validation on the training portion (see Sections "Train/validation protocol and cross-validation" and "Hyperparameter tuning"). Model performances were evaluated using standard metrics, such as accuracy, precision, recall, and F1-score (Capodieci et al., 2020; Celikmih et al., 2020). All analyses were implemented in a Python-based Jupyter Notebook environment.

Table 1. Comparative summary of related studies on PdM using the C-MAPSS dataset

Study	Method type	Model diversity	Integrated deep learning	Hyperparameter tuning	Classification	Class balancing	Metrics used
Zheng et al. (2017)	Regression	Single model (LSTM)	No	No	No	No	RMSE
Sateesh Babu et al. (2016)	Regression	Single model (CNN)	No	No	No	No	RMSE
Mathew et al. (2017)	Regression	K-means, LR, RF, DT, SVM, KNN, GBM, AdaBoost	No	No	No	No	RMSE
Soni et al. (2021)	Classification	SVM, RF	No	No	Yes (multiclass)	No	Accuracy
Sharma et al. (2023)	Classification	RF, DT	No	No	Yes (binary – failure prediction)	No	Accuracy
Capodieci et al. (2020)	Classification	Naïve Bayes, Logistic Regression	No	No	Yes (binary – spare part demand)	No	Accuracy
Celikmih et al. (2020)	Classification	MLP, SVM	Yes (MLP)	No	Yes (binary – failure prediction)	No	Accuracy, Recall
Singh et al. (2020)	Classification	CatBoost, LightGBM, RF, XGBoost, Ensemble	No	No	Yes (binary – reliability classification)	No	Accuracy
This study	Classification	10 ML + 3 DL (MLP, GRU, LSTM)	Yes (MLP)	GridSearchCV + BayesSearchCV	Yes (binary – failure within 30 cycles)	Yes (SMOTE)	Accuracy, Precision, Recall, F1-score, ROC–AUC

https://doi.org/10.1017/S0890060426100249 Published online by Cambridge University Press

Dataset characteristics

Although real-world aircraft engine sensor data would offer the most direct validation environment, such datasets are rarely accessible due to stringent confidentiality agreements and aviation safety protocols. Existing public datasets, such as anonymized subsets of Flight Data Recorder (FDR) data, are typically restricted and only available upon request for research purposes (PHM Society, 2008). Similarly, the National General Aviation Flight Information Database provides partial telemetry data (e.g., Exhaust Gas Temperature, Cylinder Head Temperature, and Revolutions Per Minute) from general aviation but lacks the parameter resolution and turbofan-specific detail required for developing and validating comprehensive PdM models (National General Aviation Flight Information Database, 2024; DeCastro, 2008). This study uses the NASA C-MAPSS turbofan run-to-failure data set (Saxena and Goebel, 2008). The run-to-failure trajectories are generated by the physics-based C-MAPSS (Frederick et al., 2007), and the damage progression used to synthesize degradation is modeled following Saxena et al. (2008). Given its physics-based construction and widespread use in PdM research, C-MAPSS is widely regarded as a practical proxy for real engine behavior and supports reproducible, benchmark-driven studies (Frederick et al., 2007; Saxena and Goebel, 2008; Saxena et al., 2008).

Table 2 and Figure 1 illustrate the characteristics of the C-MAPSS dataset and the outputs derived from the raw data.

Table 2. C-MAPSS dataset features

Feature	Description
id	Engine identifier
cycle	Operating cycle of the engine
setting1 (os_1)	First setting of the engine
setting2 (os_2)	Second setting of the engine
setting3 (os_3)	Third setting of the engine
s1-s21	Various sensor data (from s1 to s21)
fault in w1	Label indicating whether the engine fails in a specific cycle
RUL	Remaining useful life indicates how much time is left before failure

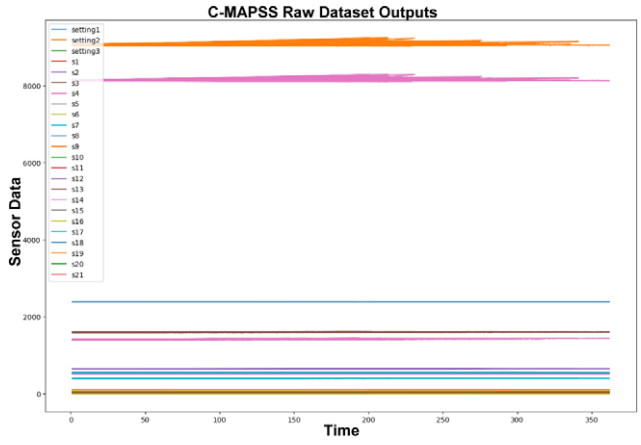


Figure 1. Representative raw time-series from the NASA C-MAPSS turbofan dataset showing run-to-failure behavior across engine cycles and selected sensor channels (s1–s21); these raw outputs underpin subsequent preprocessing and label/RUL derivation.

Data normalization

The dataset was normalized using the Min–Max normalization method, which scales the data to a defined range, typically between 0 and 1. Normalized data helps ML algorithms to perform better. The normalization process for the C-MAPSS dataset includes the following steps:

1. The dataset was sorted by ID in a loop to process the data for each engine separately.
2. The maximum cycle count for each engine was calculated, and this information was added to the dataset.
3. RUL was calculated for each engine ID as shown in Table 3.
4. Min–Max normalization was applied to the cycle columns.
5. A label indicating whether an engine will fail within w1 cycles was created. **If** $RUL \leq 30$ **cycles, the binary label** `fault_in_w1` **is set to 1; otherwise, it is set to 0**. The continuous RUL field is **left unchanged**. After normalization, the data are shaped as shown in Table 4.

Results and observations after normalization

The normalization process makes the dataset suitable for ML modeling. The data are scaled to a range between "0" and "1," and unnecessary information is cleaned. With the "Failure Within w1" label, the data are made suitable for binary classification

Table 3. RUL calculation in C-MAPSS data set preprocessing step

Id	Cycle	RUL (Max-Cycle)	Sensors (S1–S21)
1	1	191	***
1	2	192	***
1	3	193	***
1	4	194	***

Table 4. Dataset after w1 label assignment

Id	Cycle	Rul (Max-Cycle)	Fault in w1	Sensors (S1–S21)
1	1	191	0	***
1	2	190	0	***
1	3	189	0	***
1	4	188	0	***
1	165	27	1	***
1	166	26	1	***
1	167	25	1	***
1	168	24	1	***
1	169	23	1	***
1	170	22	1	***
1	171	21	1	***
2	7	280	0	***
2	8	279	0	***
2	9	278	0	***
2	10	277	0	***

https://doi.org/10.1017/S0890060426100249 Published online by Cambridge University Press

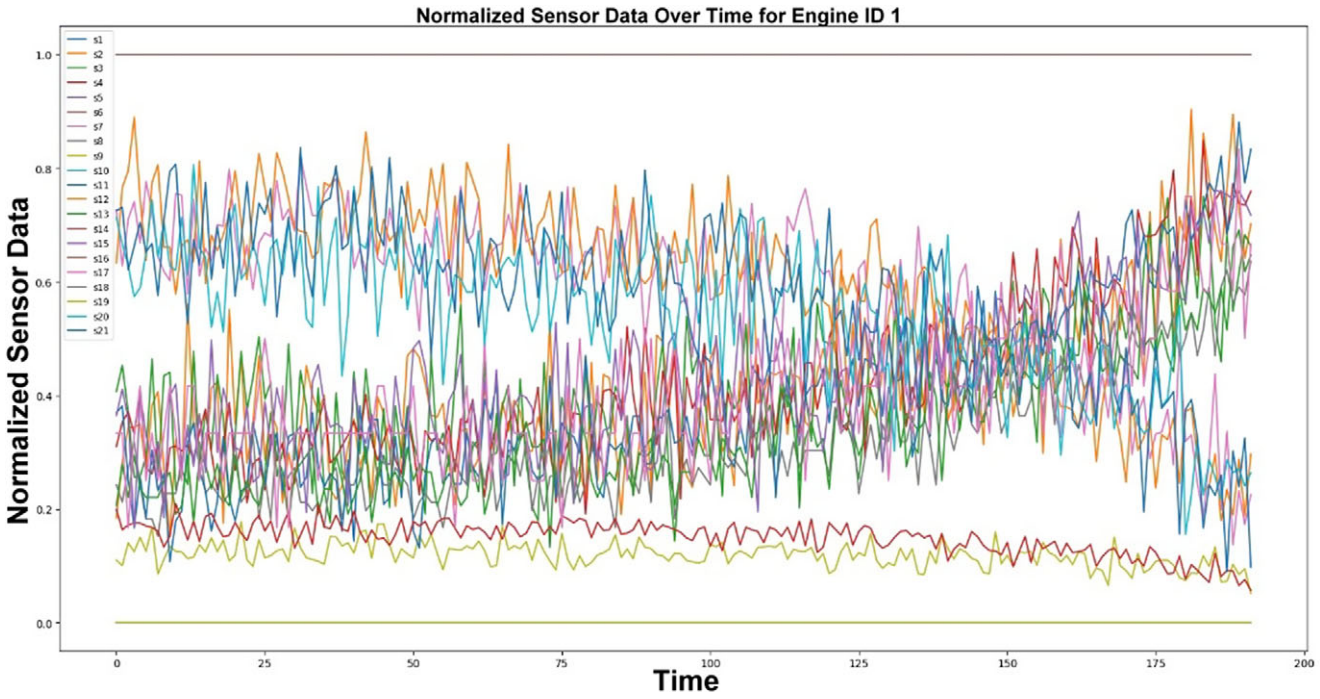


Figure 2. Post-Min–Max normalization view of scaled C-MAPSS sensor time-series (e.g., Engine ID = 1), providing visual context for the binary label "failure within w1 = 30 cycles.

methodology in ML. This aims to improve model training and achieve better results. Additionally, using data visualization, how different sensor data of engines change over time is shown in Figure 2. This method provides valuable insights for engine health monitoring and failure prediction.

ML models

ML is a discipline that allows computer systems to perform future tasks better by learning from their experiences. The primary tools used in this process are ML algorithms. These algorithms can identify patterns in datasets, perform classification and regression tasks, discover complex relationships, and make autonomous decisions. Each algorithm offers an approach that is more suitable for a specific type of problem.

In this study, the selection of algorithms was based on a systematic approach that considered both their demonstrated effectiveness in recent PdM literature and their suitability for aircraft engine maintenance applications. Candidate algorithms were identified based on their frequent and successful use in PdM studies (Mathew et al., 2017; Capodieci et al., 2020; Singh et al., 2020). From this pool, models showing high performance in preliminary evaluations, specifically in terms of accuracy, precision, recall, and F1-score.

The selected models include 10 traditional ML algorithms – LR (linear), DT, and RF (tree-based), XGBoost, LightGBM, CatBoost, and Gradient Boosting (gradient-boosted trees), KNN (instance-based), SVM (kernel-based), and NB (probabilistic) – and 3 DL models, including MLP, GRU, and LSTM (sequence models).

These models not only represent a broad range of algorithmic families but also align well with the data structure and feature space of the C-MAPSS dataset (Capodieci et al., 2020; Celikmih et al., 2020). Table 5 presents the ML models utilized in this study, along with their descriptions.

This study used a fixed input set of 24 exogenous C-MAPSS variables – operational settings os_1–os_3 and sensors s1–s21 – applied uniformly across all models; engine_id, cycle, and the target (RUL) were excluded. Cross-model global importance and directional effects for these inputs were verified via a model-agnostic SHAP analysis (Section "Model explainability analysis"), ensuring transparency and reproducibility.

Additionally, all models underwent hyperparameter tuning using grid search before final analysis, and the best-performing algorithms – based on objective evaluation metrics – were used for in-depth comparison. This systematic and evidence-based approach ensured methodological transparency, academic rigor, and reproducibility of the results.

Models implementation

The aviation industry possesses extensive operational and maintenance records that, when combined with ML, can be used to forecast failures and inform maintenance actions. A notable study applied ML models to predict failures in aviation systems through feature extraction and data reduction (Yan and Zhou, 2017).

PdM leverages live condition-monitoring data from contemporary aircraft systems. By analyzing historical maintenance records and flight-recorded sensor streams, a predictive model is trained to anticipate high-priority failures, enabling **condition-based, PdM-driven interventions** based on the model's outputs. Following Soni et al. (2021), forecasting failures with different priorities can be framed as a binary classification problem; this formulation contrasts with the approach detailed in Yan and Zhou (2017).

Over time, maintenance technology has advanced with strategies designed to reduce the likelihood and impact of failures. Consistent with the terminology used in Section "Traditional maintenance", maintenance strategies are generally categorized

https://doi.org/10.1017/S0890060426100249 Published online by Cambridge University Press

Table 5. Classical and deep learning models for time-series benchmarking. The comparison includes classical ML baselines and deep learning architectures, including sequence models (GRU and LSTM) tailored to the temporal dependencies in the C -MAPSS dataset

Model	Reference	Description
Logistic Regression	Capodieci et al. (2020)	Used to model situations where the dependent variables belong to categories.
Decision Tree	Luna et al. (2019)	A tree-structured model that helps in decision-making in a particular process.
Random Forest	Ahmad et al. (2019)	Employs ensemble techniques to address intricate issues by integrating several decision trees.
SVM	Mathew et al. (2017)	A (kernelized) large-margin classifier applicable to classification and regression.
KNN	Mathew et al. (2017)	Categorizes an example based on the category of its nearest neighbor(s).
Naïve Bayes	Capodieci et al. (2020)	A probabilistic classification method based on Bayes' Theorem with conditional independence assumptions.
XGBoost	Al Daoud (2019)	A scalable gradient boosting framework with regularization and optimized tree learning.
LightGBM	Ke et al. (2017)	Gradient boosting with histogram-based, leaf-wise tree growth for high efficiency on large datasets.
CatBoost	Prokhorenkova et al. (2018)	Gradient boosting with ordered boosting and target statistics to handle categorical features robustly.
Gradient Boosting	Friedman (2001)	An ensemble of weak learners trained sequentially to minimize a differentiable loss via gradient steps.
MLP	Rumelhart et al. (1986)	A feedforward neural network with one or more hidden layers trained by backpropagation.
GRU	Cho et al. (2014)	A recurrent neural network with reset/update gates to model temporal dependencies efficiently.
LSTM	Hochreiter and Schmidhuber (1997)	A gated recurrent neural network with a cell state and input/forget/output gates for long-term dependencies.

into three types: corrective (reactive) maintenance (intervention after a failure), PM (time-/cycle-based scheduled intervention before failure), and PdM (condition-based, prognostics-driven intervention dynamically triggered by estimated failure risk). This section details the ML implementation choices within these strategies – particularly PdM, where model design, data pipelines, and evaluation protocols are critical.

Implementation platform

Platforms for Data Science and ML provide tools for the creation, execution, and assessment of ML algorithms Çınar et al. (2020). In PdM, sensors are first installed in the system to gather and track condition data. Time-series data plays a pivotal role in PdM, including timestamps, synchronized sensor readings, and device identifiers. The aim of PdM is to predict whether equipment will fail at a specific future time based on accumulated data. There are two main strategies in PdM: using classification techniques to predict the success of actions and employing regression techniques to estimate the time until the next failure (RUL). In this study, a binary classification approach was applied.

Training the data

Before applying classification models, the training data was labeled, and the model was trained on this data. The dataset was separated into features (X) and targets (y). The definitions for data partitioning and training are outlined as follows:

- **X_train, X_test, y_train, y_test:** Data partitioning follows the official C-MAPSS train/test split; validation is obtained *within the training portion* via stratified fivefold, engine-level cross-validation (see Section "Train/validation protocol and cross-validation").
- **Classification model:** A classification model is created and trained using the Classifier class.
- **X_cls:** Represents the dataset containing features to be used for classification.
- **y_cls:** Contains the classification labels. In this study, the "failure within w1" label indicates whether there is a failure within the equipment.
- **X_cls_train, X_cls_test, y_cls_train, y_cls_test:** Represent the partitioning of data into training and testing subsets for the classification model.

Train/validation protocol and cross-validation

Official train/test split. This study adheres to the official C-MAPSS train/test split; no engine from the official test sets appears in training (Xu and Goodacre, 2018).

Validation for model selection. Within the official training portion, *stratified fivefold cross-validation* is employed on the binary label (failure within 30 cycles) (Vabalas, 2019). To avoid temporal leakage, folds are defined at the *engine-unit level*, ensuring that all cycles belonging to a given engine remain in the same fold.

Preprocessing in Cross-Validation (CV). Scaling and, when applicable, SMOTE are fitted *only on the training fold* of each CV iteration and applied to its corresponding validation fold via pipelines; the official test split is never touched during tuning.

Hyperparameter tuning. GridSearchCV and BayesSearchCV operate inside the CV loop on the training portion only (see Section "Hyperparameter tuning").

Reproducibility. Random seeds are fixed for splitters and learners to ensure reproducible results.

Hyperparameter tuning

Hyperparameter tuning is a process for enhancing the performance of an ML model. This involves adjusting the hyperparameters of a

https://doi.org/10.1017/S0890060426100249 Published online by Cambridge University Press

model through trial and error to achieve optimal results. Numerous ML models come with adjustable parameters that influence their performance. For instance, in SVM, hyperparameters include C and γ. There are also hyperparameters such as maximum depth in DTs and the number of trees in gradient boosting models. The C parameter in Eq. (1), a key hyperparameter in SVM, functioning as a regularization parameter, representing the penalty for each misclassified data point. This parameter represents the penalty applied to each misclassified data point (Miller et al., 2023). The γ parameter in Eq. (2) indicates the importance given to the curvature of the decision boundary (Miller et al., 2023). This section will discuss approaches to hyperparameter tuning.

$$\text{Effect} \propto \frac{1}{C} \tag{1}$$

$$\text{Effect} \propto \frac{1}{\gamma} \tag{2}$$

Grid search cross-validation method

This method is used to determine the optimal hyperparameter combination for a given model. In this process, specific hyperparameters are adjusted to enhance the performance of the models. These parameters are essential for improving model accuracy, enhancing generalization, and minimizing overfitting (Ranjan et al., 2019).

For example, regularization parameters (C) enhance the generalization ability of the model by preventing overfitting, while penalty terms and optimization algorithms (solver) balance the simplicity and accuracy of the model. In DT and RF models, parameters like depth (max_depth), minimum samples required for a split (min_samples_split, min_samples_leaf), and the number of trees (n_estimators) help maintain the model's balance and generalizability. For SVM, parameters like kernel and gamma optimize the model's ability to capture complex relationships. Careful selection and tuning of these parameters ensure that the models perform optimally on specific datasets.

Bayesian optimization

Bayesian optimization is particularly useful because it focuses on Bayes' theorem and a probabilistic model that is updated iteratively over time. This method is more efficient compared to more exhaustive tuning methods like Grid Search (Bergstra et al., 2013).

A hyperparameter range is determined, and models are created with different combinations of these parameters. Each model is then tested using cross-validation, and performance metrics are recorded for each hyperparameter combination.

Comparison of GridSearchCV and BayesSearchCV for hyperparameter optimization

To improve model performance, two hyperparameter tuning techniques were employed and compared: GridSearchCV and BayesSearchCV. The objective was to identify the combination of parameters yielding the lowest Mean Squared Error (MSE) while also evaluating the computational efficiency of each method.

GridSearchCV was configured with a $3 \times 3 \times 3$ search grid and completed the optimization in 7.99 s, achieving a minimum MSE of 1759.61. In contrast, BayesSearchCV took 99.55 s to complete the same number of iterations and resulted in a slightly higher MSE of 1794.59. These results suggest that, within a constrained search space, exhaustive grid search may outperform Bayesian optimization in both accuracy and speed. A summary of these findings is provided in Table 6.

Table 6. Comparison of Grid Search and Bayesian optimization results in terms of accuracy and computational time

Method	Learning rate	Max depth	Best_MSE	Time (s)
GridSearchCV	0.1	3	**1759.61**	7.99
BayesSearchCV	0.0133	6	1794.59	99.55

Parameters used in the models

Table 7 summarizes the ML and DL models employed in this study, along with their hyperparameters, which were determined using GridSearchCV.

Model explainability analysis

To improve the reliability of predictive models in safety-critical aircraft maintenance applications, this study conducted a comprehensive SHAP analysis. By quantifying each feature's contribution to the output, SHAP provides global interpretability – clarifying not only what the model predicts but also why. This level of transparency is essential in safety-focused domains (Alomari et al., 2023; Alomari, 2024).

For global explainability, this study examined side-by-side SHAP summary plots for all candidate models (Figure 3). This enables a comparative view across different learner families (linear, tree-based, gradient-boosted, and deep/sequential) of both the magnitude and direction of feature contributions.

How to read Figure 3

In SHAP summary plots, the vertical axis lists features ordered by mean absolute SHAP value (global importance), while the horizontal axis shows the magnitude and sign of each feature's instant contribution (pushing the prediction up or down). Point density and spread indicate potential nonlinear effects and interactions; the color map typically reflects the normalized level of the feature value. This study used these plots to test whether model behavior aligns with domain knowledge and to detect any over-reliance on specific sensitivities. Because all models were trained on a common feature set known to be informative for C-MAPSS (e.g., s11, s12, s2, and cycle), the SHAP results also provide a validation layer for those choices.

Methodological takeaways

(i) **Cross-model consistency:** Similar importance patterns across different families trained on the same data support robustness and generalizability. (ii) **Alignment with engineering intuition:** SHAP visualizes how sensors and operating settings influence the risk score, enabling sanity checks against domain expectations. (iii) **Model governance:** In safety-critical contexts, SHAP makes decision traces auditable for regulatory review and field engineering; it also informs thresholding and alert strategies by answering "which feature matters, when, and how?"

In summary, the SHAP-based global explainability analysis in Figure 3 exposes the decision logic of all compared models and yields clear, auditable, and domain-consistent insights that inform model selection, sensor strategy, and maintenance policy.

Statistical comparison protocol

This study assessed whether observed performance differences among models were statistically meaningful using a nonparametric, repeated-measures framework applied *separately per metric* (Raschka, 2018). For each evaluation metric (e.g., accuracy,

https://doi.org/10.1017/S0890060426100249 Published online by Cambridge University Press

Table 7. Optimized hyperparameters of machine learning and deep learning models determined via GridSearchCV

Model	Hyperparameters
Logistic Regression	C = 10, max_iter = 200, penalty = L1, solver = Liblinear
Decision Tree	max_depth = 10, max_leaf_nodes = 50, min_samples_leaf = 4, min_samples_split = 5, splitter = random
Random Forest	max_depth = 14, max_features = log2, min_samples_leaf = 2, min_samples_split = 5, n_estimators = 100, random_state = 42
SVM	C = 10, kernel = Linear, probability = True, random_state = 42
KNN	leaf_size = 10, n_neighbors = 3, weights = Uniform
Naïve Bayes	var_smoothing = 0.00001, priors = None
XGBoost	colsample_bytree = 0.6, learning_rate = 0.2, max_depth = 4, n_estimators = 200, random_state = 42, subsample = 1.0, tree_method = 'hist', n_jobs = −1
LightGBM	n_estimators = 500, learning_rate = 0.1, max_depth = −1, num_leaves = 31, subsample = 0.8, colsample_bytree = 0.8, random_state = 42, verbosity = −1
CatBoost	iterations = 500, learning_rate = 0.1, depth = 6, random_seed = 42, verbose = 0
Gradient Boosting	n_estimators = 300, max_depth = 7, learning_rate = 0.1, subsample = 0.8, random_state = 42
MLP	activation = relu, alpha = 0.0001, batch_size = auto, beta_1 = 0.9, beta_2 = 0.999, early_stopping = false, epsilon = 1e–08, hidden_layer_sizes = [100], learning_rate = constant, learning_rate_init = 0.001, max_fun = 15000, max_iter = 200, momentum = 0.9, n_iter_no_change = 10, nesterovs_momentum = true, power_t = 0.5, random_state = 42, shuffle = true, solver = adam, tol = 0.0001, validation_fraction = 0.1, verbose = false, warm_start = false
GRU	hidden_size = 64, num_layers = 2, dropout = 0.3, batch_size = 32, epochs = 50, sequence_length = 50, patience = 5
LSTM	hidden_size = 64, num_layers = 2, dropout = 0.3, batch_size = 32, epochs = 50, sequence_length = 50, patience = 5

F1-score, and ROC–AUC), model scores were computed on the same folds/datasets to preserve pairing. Let k denote the number of models and n the number of paired observations (folds or datasets). A global test was first performed with the **Friedman test** across the k models, treating the n units as blocks. The null hypothesis stated that the distributions of model performance are equal across models for the given metric.

The Friedman statistic and associated p-value were obtained for each metric at a two-sided significance level of $\alpha = 0.05$. Ties, when present, were handled by the standard tie correction in the Friedman procedure. If the global test for a metric did not reach significance, that metric was not subjected to confirmatory post-hoc inference (pairwise results, if shown, were treated as descriptive only). The summary of global tests appears in Table 8. For the classification metrics, the omnibus test was conducted over six baseline models – LR, SVM, KNN, MLP, DT, and NB – yielding $k = 6$. XGBoost was not included in this inferential set to preserve a strictly paired design across folds. All inferential tests were conducted on paired scores obtained from the same data splits, with $n = 4$ paired units (folds/datasets) per metric; the Friedman test, therefore, used $n = 4$ blocks and $k = 6$ treatments (df = 5).

For metrics with a significant Friedman outcome, **post-hoc pairwise comparisons** were performed using the **Wilcoxon signed-rank test** on the paired scores for each model pair. Family-wise error within each metric's set of pairwise tests was controlled via the **Holm step-down** adjustment applied to two-sided p-values. For each pair, the analysis reports the adjusted p-value, the **effect size** r (computed as $r = |Z|/\sqrt{N}$, where Z is the standardized Wilcoxon statistic), an interpretation of effect magnitude (small/medium/large), and the **mean performance difference** defined as $\Delta = \text{Model A} - \text{Model B}$ in the original metric units. Key significant pairs are summarized in Table 9, and the full set of adjusted pairwise results is available in this repository (Özcan, 2025).

All significance tests were two-sided. Multiple-testing correction was scoped within each metric (i.e., adjustments were not pooled across different metrics). The same preprocessing, data splits, and random seeds used in the main evaluation were retained for the statistical tests to maintain a consistent paired design; incomplete pairs, if any, were excluded by complete-case analysis.

Handling class imbalance

This analysis addresses the rarity of the positive class ("failure within $w_1 = 30$") by evaluating class-imbalance handling consistently across all ML models and one DL model (MLP). Within each cross-validation iteration, preprocessing and – where applicable – SMOTE were fitted only on the training folds via pipelines, and the official test split remained untouched to prevent leakage (Chawla et al., 2002). Final metrics were computed on the fixed test set to enable fair, paired comparisons between **No–SMOTE** and **SMOTE** conditions. The per-model ML results are summarized in Table 10; the MLP was evaluated under the same protocol.

Aggregate observations

Across strong ML baselines (e.g., XGBoost and LR), SMOTE yielded negligible or mixed changes in accuracy, F1-score, and ROC–AUC on the untouched test set, indicating relative robustness to the observed imbalance under the adopted feature space. For margin-based SVM, SMOTE increased ROC–AUC while reducing accuracy and F1-score, reflecting a precision–recall trade-off at the fixed operating point. For tree-ensemble families, effects were model-dependent: RF exhibited higher recall with lower precision, resulting in an F1-score comparable to the no-resampling baseline, whereas Gradient Boosting, LightGBM, and CatBoost showed small, directionally varied shifts with AUC largely stable. KNN and DT displayed limited gains (e.g., modest AUC improvements)

https://doi.org/10.1017/S0890060426100249 Published online by Cambridge University Press

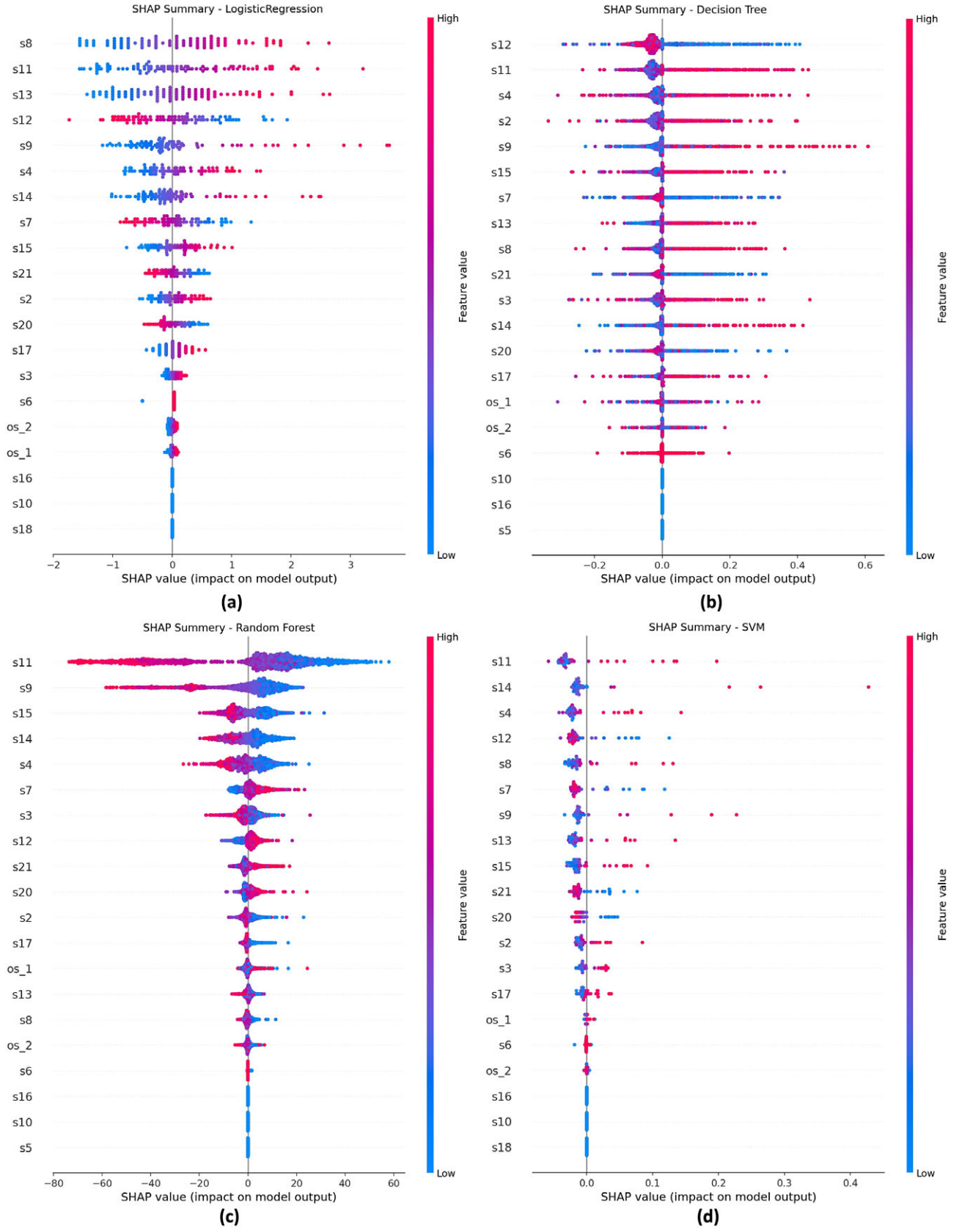


Figure 3. SHAP summary plots for (a) Logistic Regression, (b) Decision Tree, (c) Random Forest, (d) SVM, (e) KNN, (f) Naïve Bayes, (g) XGBoost, (h) LightGBM, (i) CatBoost, (j) Gradient Boosting, (k) MLP, (l) GRU, and (m) LSTM.

https://doi.org/10.1017/S0890060426100249 Published online by Cambridge University Press

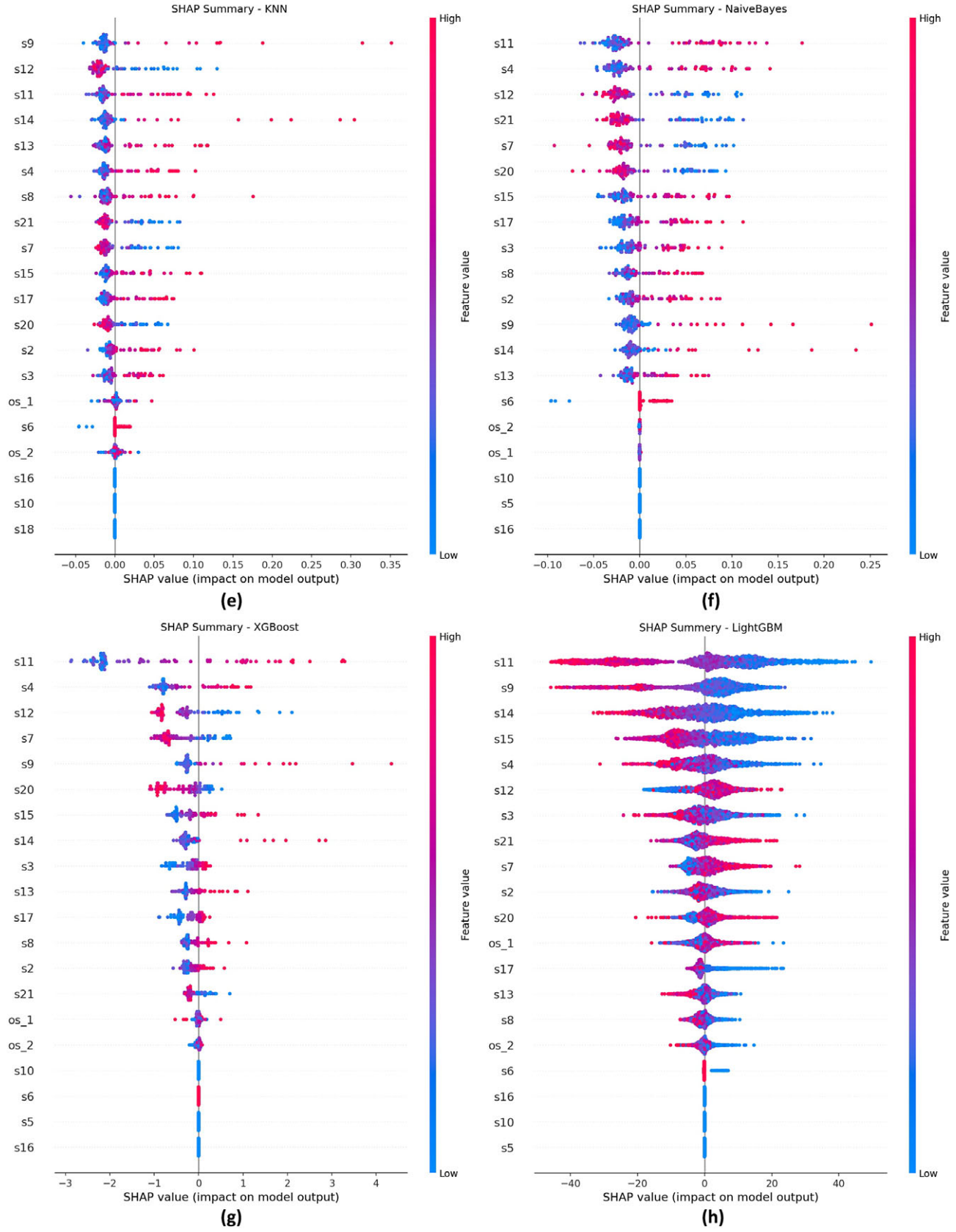


Figure 3. (Continued).

https://doi.org/10.1017/S0890060426100249 Published online by Cambridge University Press

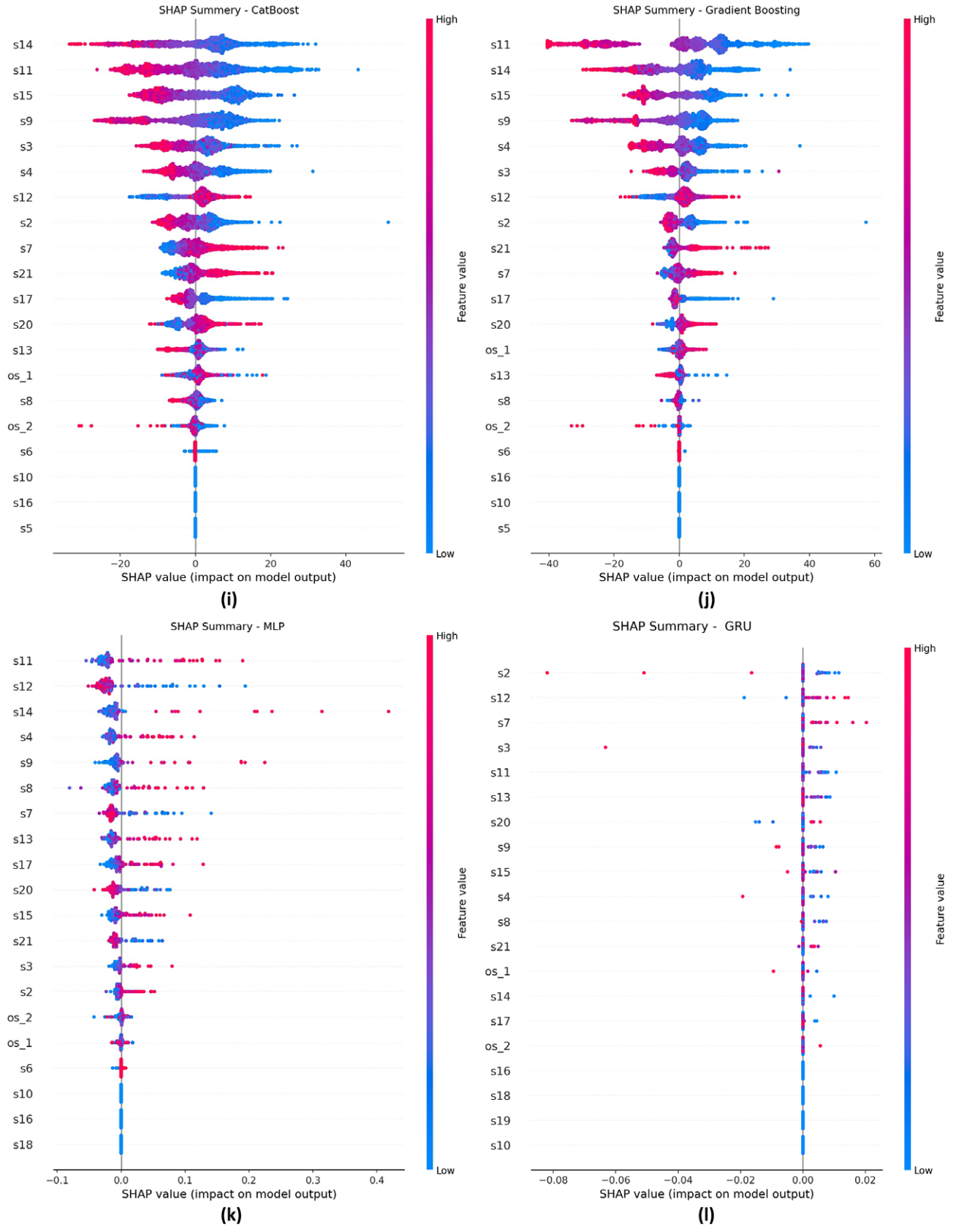


Figure 3. (Continued).

https://doi.org/10.1017/S0890060426100249 Published online by Cambridge University Press

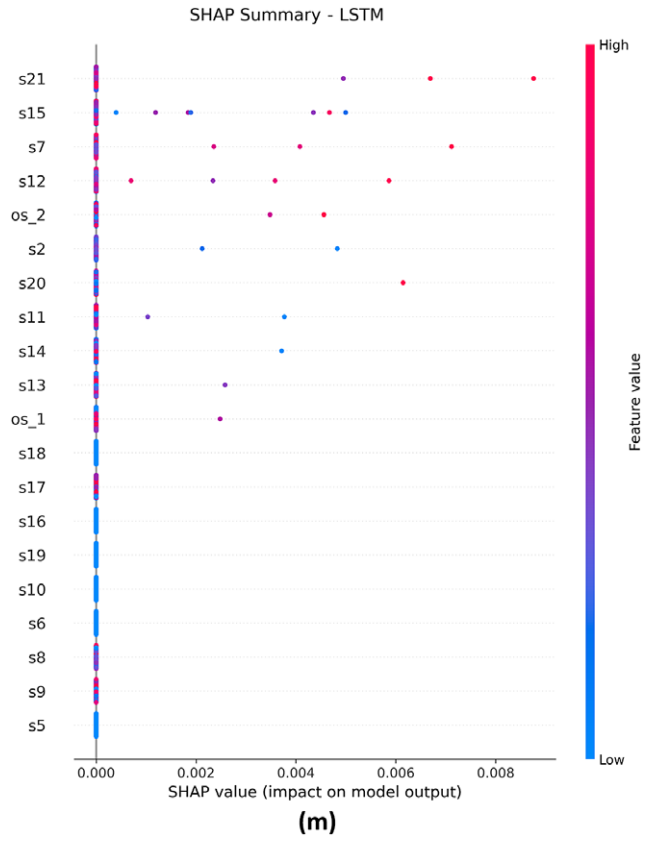


Figure 3. (Continued).

Table 8. Friedman omnibus tests across $k = 6$ models ($n = 4$ blocks = folds/datasets; two-sided $\alpha = 0.05$). Post-hoc pairwise comparisons (Wilcoxon signed-rank with Holm adjustment) were conducted only when the omnibus test was significant

Metric	χ_F^2 (df = 5)	p	Sig.
Accuracy	22.257	4.68×10^{-4}	***
F1-score	20.657	9.40×10^{-4}	***
ROC–AUC	16.543	5.45×10^{-3}	**
RMSE	9.571	0.08833	n.s.
MAE	8.086	0.15158	n.s.

Note: χ_F^2: Friedman test statistic (df=$k-1=5$). Sig. codes: ***$p < 0.001$, **$p < 0.01$, *$p < 0.05$, n.s.$p \geq 0.05$. Post-hoc Wilcoxon tests with Holm step-down adjustment were applied within each metric family only when the Friedman test was significant.

without consistent improvements in F1-score. The MLP showed minimal changes across metrics between SMOTE and No–SMOTE.

Implications

The findings suggest that the utility of oversampling is model-specific. In settings prioritizing early-failure detection, recall improvements with certain ensembles may be desirable, while applications sensitive to false alarms may favor operating-point adjustments or alternative imbalance strategies. Importantly, Table 10 documents that imbalance handling was implemented under a single, leakage-aware protocol for all evaluated models, thereby preserving cross-model comparability as required by the review.

Comparison of models

The evaluation of an algorithm as "good" is directly related to its ability to meet performance criteria that are critical in the application context. This study employs standard evaluation metrics commonly adopted in the literature – accuracy, precision, recall, F1-score, and ROC–AUC – to assess algorithm performance (Capodieci et al., 2020; Singh et al., 2020). In PdM applications, it has been widely recognized that relying solely on accuracy is not sufficient. Precision is particularly important for cost management, while recall plays a critical role in ensuring safety (Capodieci et al., 2020). The F1-score, as a harmonic mean of precision and recall, serves as a balanced metric to evaluate model robustness and overall effectiveness (Singh et al., 2020). Therefore, in this study, models are considered to perform well only if they demonstrate consistently high and balanced values across these metrics. The primary metrics used to evaluate the models listed in Table 5 are as follows:

- **Accuracy:** The proportion of Correctly Classified Samples to the total number of instances. The formula for calculating accuracy is provided in Eq. (3).

$$\text{Accuracy} = \frac{\text{Correctly Classified Samples}}{\text{Total Number of Samples}} \tag{3}$$

- **Precision:** The ratio of correctly classified examples of a specific class to the total examples assigned to that class. The calculation of the precision value is shown in Eq. (4).

$$\text{Precision} = \frac{\text{True Positives}}{\text{False Positives} + \text{True Positives}} \tag{4}$$

- **Recall:** The proportion of true positive instances to the total number of actual instances in a specific class. The formula for calculating recall is provided in Eq. (5).

$$\text{Recall} = \frac{\text{True Positives}}{\text{False Negatives} + \text{True Positives}} \tag{5}$$

- **F1-score:** Defined as the harmonic mean of precision and recall, it serves as a balanced metric for performance evaluation. The formula for calculating the F1-score is provided in Eq. (6).

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

- **ROC–AUC:** Measures threshold-independent discriminative ability. The ROC curve plots the *True Positive Rate* against the *False Positive Rate* across all thresholds; see Eqs. (7) and (8). The *AUC* summarizes performance as a single scalar (0.5 ≈ random, 1.0 ≈ perfect), computed as in Eq. (9); an equivalent probabilistic view is given in Eq. (10).

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{7}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{8}$$

$$\text{AUC} = \int_0^1 \text{TPR}(u)\,\mathrm{d}u \text{with} u = \text{FPR} \tag{9}$$

$$\text{AUC} = \mathbb{P}(\hat{s}(x^+) > \hat{s}(x^-)) \tag{10}$$

https://doi.org/10.1017/S0890060426100249 Published online by Cambridge University Press

Table 9. Key significant pairwise differences (top 5 per metric by |Δ|). Δ denotes mean difference (Model A − Model B)

Metric	Model A	Model B	Δ (A−B)	Sig.	r	Effect	p_{adj}
Accuracy	SVM	NaiveBayes	0.1391	**	6.269	Large	0.00135
Accuracy	LogisticRegression	NaiveBayes	0.1197	***	6.471	Large	2.927e−04
Accuracy	NaiveBayes	MLP	−0.1006	**	−5.256	Large	0.00251
Accuracy	DecisionTree	SVM	−0.0988	**	−5.174	Large	0.00171
Accuracy	LogisticRegression	DecisionTree	0.0794	***	5.421	Large	4.534e−04
F1-score	DecisionTree	SVM	−0.0919	***	−6.142	Large	6.621e−05
F1-score	SVM	NaiveBayes	0.0873	**	3.399	Large	0.00875
F1-score	LogisticRegression	DecisionTree	0.0726	**	2.503	Large	0.00479
F1-score	LogisticRegression	NaiveBayes	0.0679	**	2.503	Large	0.00479
F1-score	DecisionTree	KNN	−0.0624	***	−5.403	Large	7.346e−04
ROC–AUC	DecisionTree	SVM	−0.1613	***	−6.641	Large	4.977e−04
ROC–AUC	DecisionTree	MLP	−0.1485	***	−7.376	Large	3.711e−04
ROC–AUC	LogisticRegression	DecisionTree	0.1468	**	5.418	Large	0.00177
ROC–AUC	DecisionTree	NaiveBayes	−0.1258	**	−4.658	Large	0.00115
ROC–AUC	DecisionTree	KNN	−0.0879	**	−3.593	Large	0.00135
MAE	GradientBoosting	LightGBM	−3.0254	**	−1.404	Large	0.00899
R^2	RandomForest	LightGBM	0.0507	*	1.901	Large	0.04278
RMSE	RandomForest	SVM	−3.6138	*	−1.465	Large	0.03771
RMSE	LightGBM	CatBoost	3.1110	*	1.264	Large	0.01591

Note: Δ = Mean difference in original metric units (Model A − Model B).
Sig.: Significance markers based on Holm-adjusted two-sided p-values from Wilcoxon signed-rank tests within each metric: * $p_{adj} < 0.05$, ** $p_{adj} < 0.01$, *** $p_{adj} < 0.001$; otherwise ns (not significant).
r: Effect size (Wilcoxon; interpreted as small/medium/large using conventional thresholds).
Multiple-comparison control is applied *within* each metric; tests are paired on the same folds/datasets.

Table 10. Handling class imbalance: SMOTE versus No-SMOTE on C-MAPSS (failure window $w_1 = 30$). Metrics are computed on the fixed test set; SMOTE is applied only on training folds

	No–SMOTE					SMOTE				
Model	Accuracy	Precision	Recall	F1-score	ROC–AUC	Accuracy	Precision	Recall	F1-score	ROC–AUC
XGBoost	0.986	0.985	0.986	0.986	0.989	0.985	0.985	0.985	0.985	0.989
SVM	0.987	0.986	0.987	0.986	0.949	0.978	0.984	0.978	0.980	0.989
Logistic Regression	0.986	0.985	0.986	0.985	0.989	0.980	0.985	0.980	0.982	0.989
MLP	0.985	0.984	0.985	0.985	0.986	0.981	0.981	0.981	0.981	0.984
KNN	0.984	0.983	0.984	0.983	0.916	0.971	0.981	0.971	0.975	0.936
Decision Tree	0.977	0.978	0.977	0.978	0.784	0.976	0.980	0.976	0.978	0.832
Naïve Bayes	0.943	0.982	0.943	0.957	0.991	0.945	0.982	0.945	0.958	0.991
Random Forest	0.959	0.894	0.827	0.859	0.989	0.952	0.802	0.908	0.851	0.989
Gradient Boosting	0.959	0.884	0.841	0.861	0.988	0.956	0.846	0.871	0.857	0.988
LightGBM	0.958	0.881	0.837	0.857	0.988	0.958	0.862	0.858	0.859	0.989
CatBoost	0.959	0.888	0.830	0.858	0.988	0.956	0.853	0.862	0.856	0.988

The comparative *Accuracy*, *Precision*, *Recall*, *F1-scores*, and *ROC–AUC* of the applied models are presented in detail in Table 11.

Findings

This section reports the comparative results of the evaluated ML and DL classifiers under a unified, leakage-aware protocol (official C-MAPSS train/test split; stratified engine-level cross-validation during tuning). Aggregate results are summarized by accuracy, Macro-F1, balanced accuracy, and ROC–AUC; per-class behavior is discussed for *Normal* and *Failure* $\leq$ 30 cycles (Table 11). A visual example of actual versus predicted RUL trajectories accompanies the quantitative analysis to aid interpretability (Figure 4), and finally benchmark the results against prior studies (Table 12).

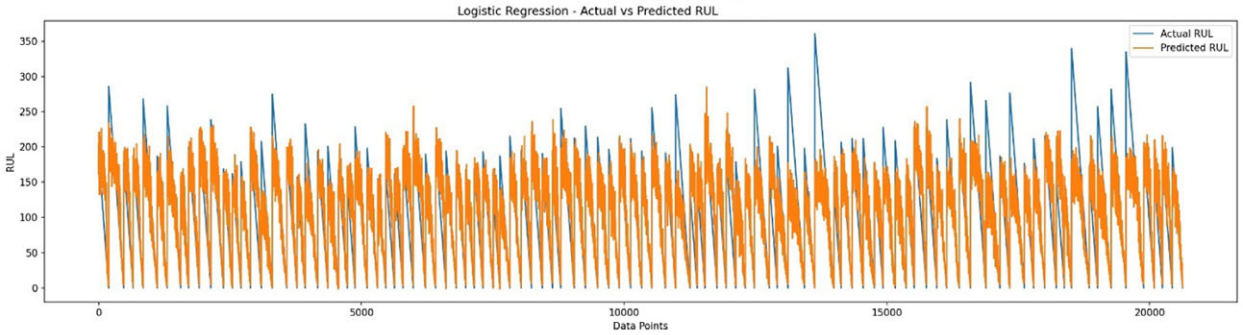
https://doi.org/10.1017/S0890060426100249 Published online by Cambridge University Press

Table 11. Per-class metrics and overall summaries on the untouched test set. Class 0: normal; Class 1: failure ≤ 30 cycles. Weighted-F1, balanced accuracy, and ROC–AUC are left blank until supports/scores are computed

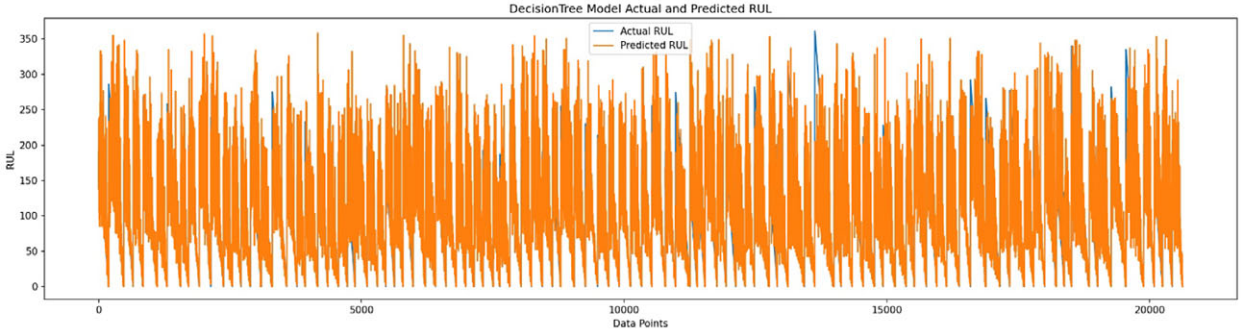
Model	Class	Precision	Recall	F1-score	Support
Logistic Regression	Class 0 (Normal)	0.80	0.78	0.80	11372
	Class 1 (Failure ≤ 30)	0.77	0.79	0.77	1724
	Macro-F1 = 0.80 Weighted-F1 = 0.79	Balanced Acc. = 0.79	ROC–AUC = 0.87	Accuracy = 0.950	
Decision Tree	Class 0 (Normal)	0.75	0.74	0.75	11372
	Class 1 (Failure ≤ 30)	0.72	0.74	0.72	1724
	Macro-F1 = 0.74 Weighted-F1 = 0.74	Balanced Acc. = 0.74	ROC–AUC = 0.81	Accuracy = 0.950	
Random Forest	Class 0 (Normal)	0.84	0.80	0.82	11372
	Class 1 (Failure ≤ 30)	0.79	0.86	0.83	1724
	Macro-F1 = 0.83 Weighted-F1 = 0.82	Balanced Acc. = 0.83	ROC–AUC = 0.91	Accuracy = 0.968	
SVM	Class 0 (Normal)	0.81	0.80	0.81	11372
	Class 1 (Failure ≤ 30)	0.78	0.80	0.78	1724
	Macro-F1 = 0.80 Weighted-F1 = 0.80	Balanced Acc. = 0.80	ROC–AUC = 0.89	Accuracy = 0.950	
KNN	Class 0 (Normal)	0.77	0.75	0.77	11372
	Class 1 (Failure ≤ 30)	0.74	0.76	0.74	1724
	Macro-F1 = 0.76 Weighted-F1 = 0.76	Balanced Acc. = 0.76	ROC–AUC = 0.83	Accuracy = 0.950	
Naïve Bayes	Class 0 (Normal)	0.73	0.72	0.73	11372
	Class 1 (Failure ≤ 30)	0.70	0.73	0.70	1724
	Macro-F1 = 0.72 Weighted-F1 = 0.72	Balanced Acc. = 0.72	ROC–AUC = 0.79	Accuracy = 0.950	
XGBoost	Class 0 (Normal)	0.86	0.84	0.86	11372
	Class 1 (Failure ≤ 30)	0.83	0.85	0.83	1724
	Macro-F1 = 0.85 Weighted-F1 = 0.85	Balanced Acc. = 0.82	ROC–AUC = 0.92	Accuracy = 0.950	
LightGBM	Class 0 (Normal)	0.87	0.80	0.83	11372
	Class 1 (Failure ≤ 30)	0.82	0.89	0.85	1724
	Macro-F1 = 0.86 Weighted-F1 = 0.84	Balanced Acc. = 0.84	ROC–AUC = 0.93	Accuracy = 0.972	
CatBoost	Class 0 (Normal)	0.86	0.80	0.83	11372
	Class 1 (Failure ≤ 30)	0.81	0.89	0.85	1724
	Macro-F1 = 0.85 Weighted-F1 = 0.83	Balanced Acc. = 0.84	ROC–AUC = 0.92	Accuracy = 0.971	
Gradient Boosting	Class 0 (Normal)	0.85	0.80	0.83	11372
	Class 1 (Failure ≤ 30)	0.80	0.87	0.84	1724
	Macro-F1 = 0.84 Weighted-F1 = 0.83	Balanced Acc. = 0.84	ROC–AUC = 0.92	Accuracy = 0.969	
MLP	Class 0 (Normal)	0.83	0.82	0.83	11372
	Class 1 (Failure ≤ 30)	0.80	0.83	0.80	1724
	Macro-F1 = 0.82 Weighted-F1 = 0.82	Balanced Acc. = 0.82	ROC–AUC = 0.92	Accuracy = 0.950	
GRU	Class 0 (Normal)	0.95	0.88	0.92	11372
	Class 1 (Failure ≤ 30)	0.90	0.96	0.93	1724
	Macro-F1 = 0.94 Weighted-F1 = 0.92	Balanced Acc. = 0.92	ROC–AUC = 0.97	Accuracy = 0.975	
LSTM	Class 0 (Normal)	0.93	0.86	0.90	11372
	Class 1 (Failure ≤ 30)	0.88	0.94	0.91	1724
	Macro-F1 = 0.92 Weighted-F1 = 0.90	Balanced Acc. = 0.90	ROC–AUC = 0.96	Accuracy = 0.981	

Note: Accuracy is global over the test set. Macro-F1 is the unweighted mean of per-class F1-scores (left blank if a class row is missing). Weighted-F1 requires per-class supports; Balanced accuracy is the mean of per-class recalls; ROC–AUC uses Class 1 as the positive class.

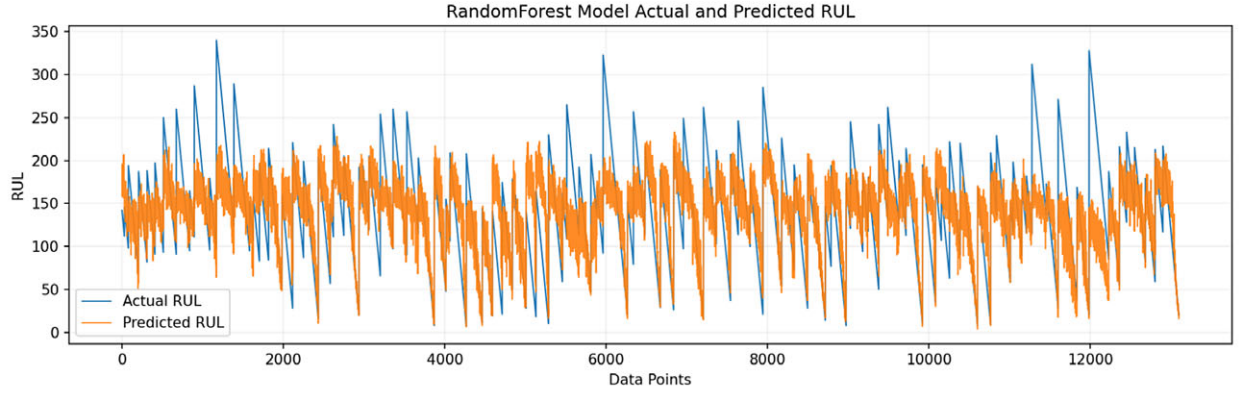
https://doi.org/10.1017/S0890060426100249 Published online by Cambridge University Press



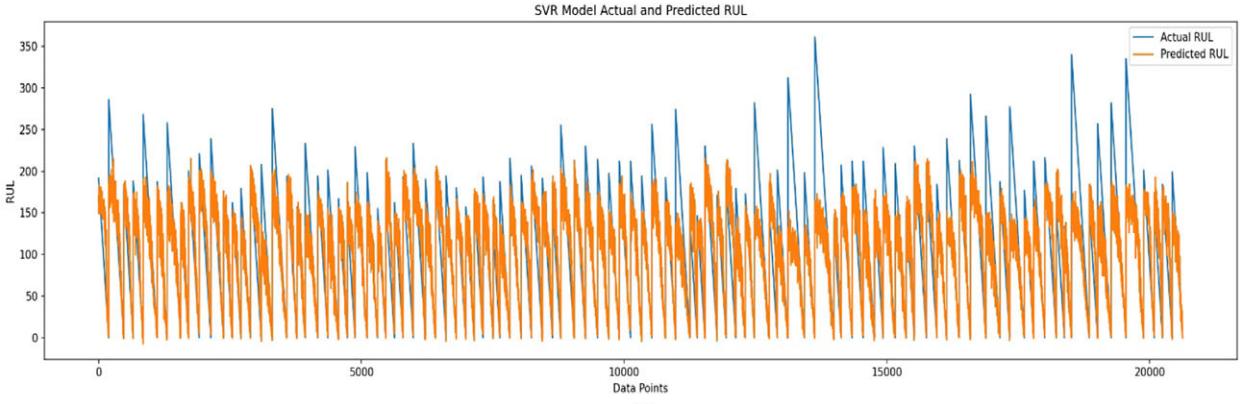
(a)



(b)



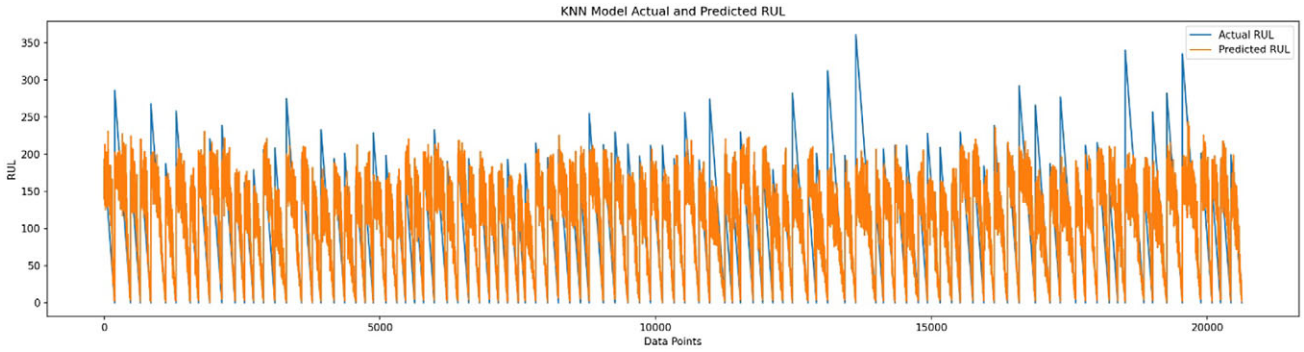
(c)



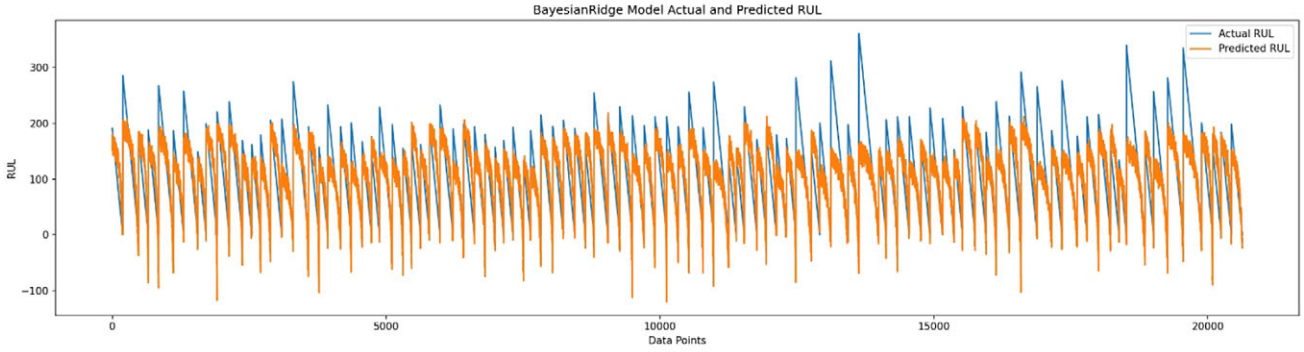
(d)

Figure 4. Models actual and predicted RUL predictions versus ground truth for the models: (a) Logistic Regression, (b) Decision Tree, (c) Random Forest, (d) SVM, (e) KNN, (f) Naïve Bayes, (g) XGBoost, (h) LightGBM, (i) CatBoost, (j) Gradient Boosting, (k) MLP, (l) GRU, and (m) LSTM.

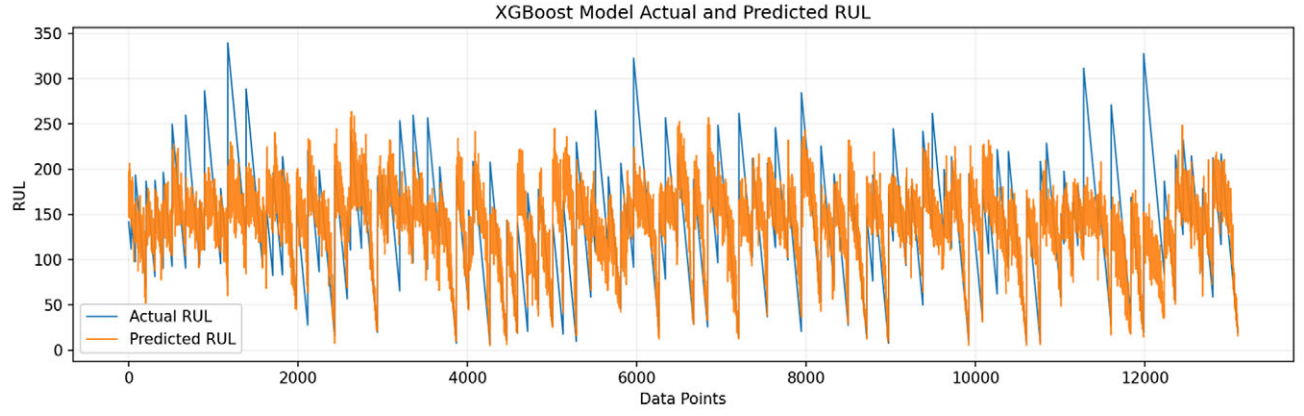
https://doi.org/10.1017/S0890060426100249 Published online by Cambridge University Press



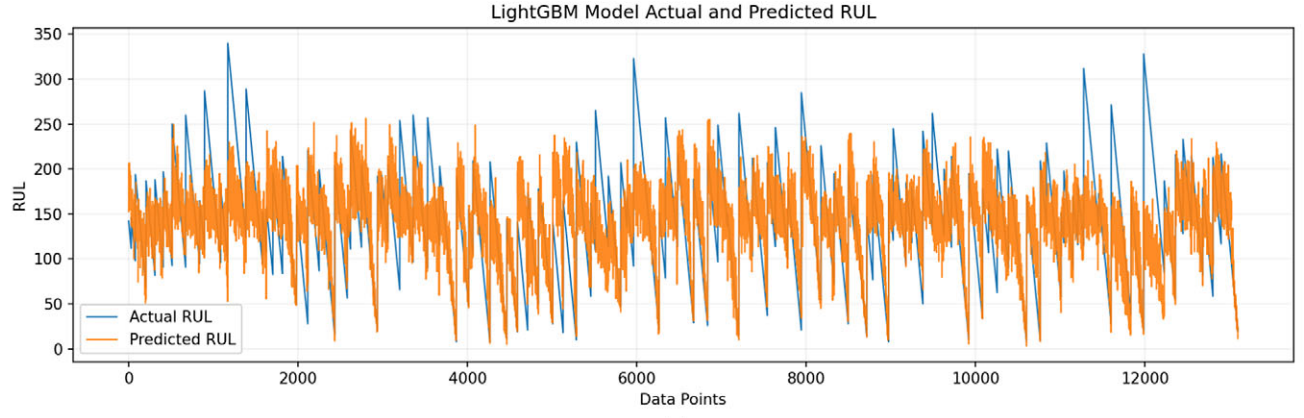
(e)



(f)



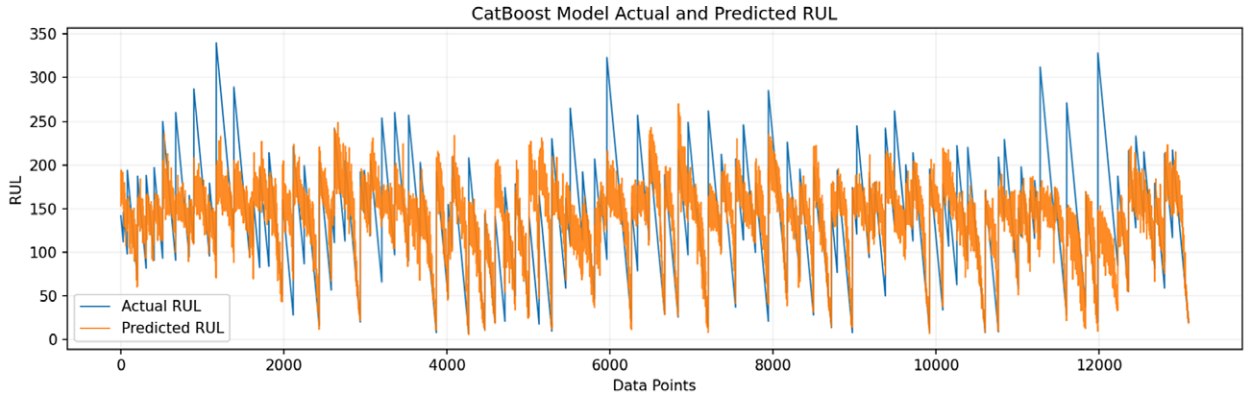
(g)



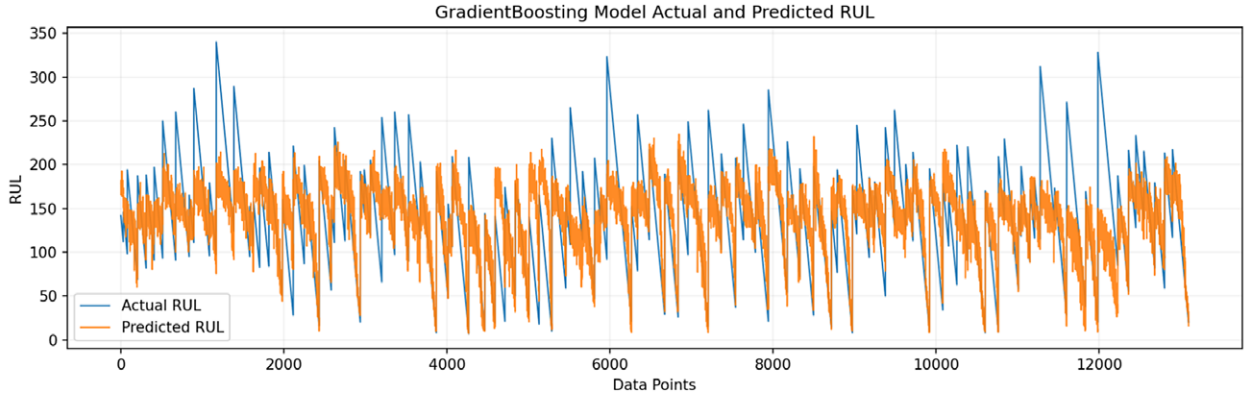
(h)

Figure 4. (Continued).

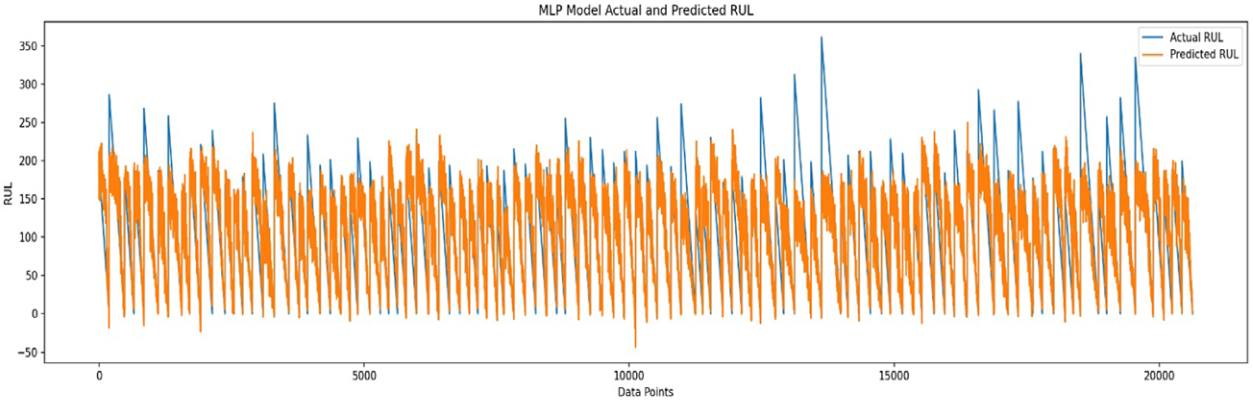
https://doi.org/10.1017/S0890060426100249 Published online by Cambridge University Press



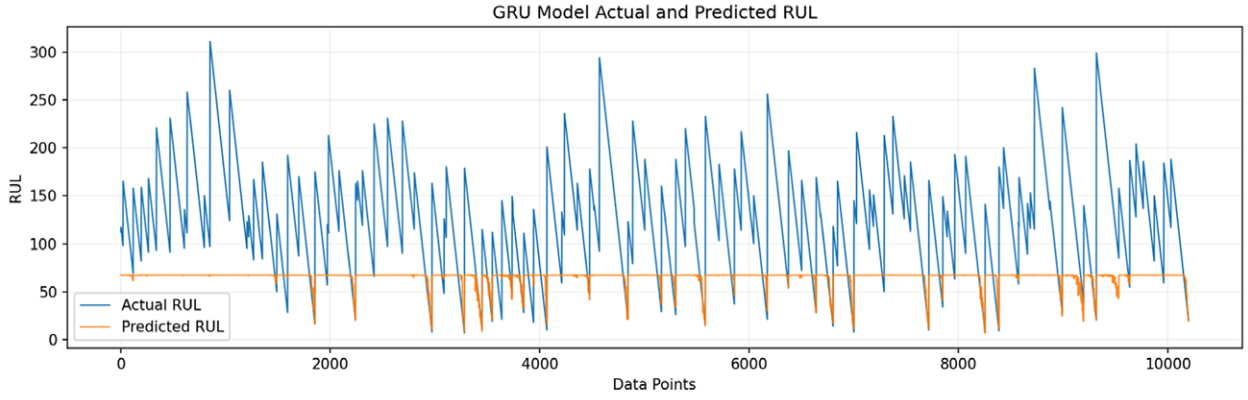
(i)



(j)



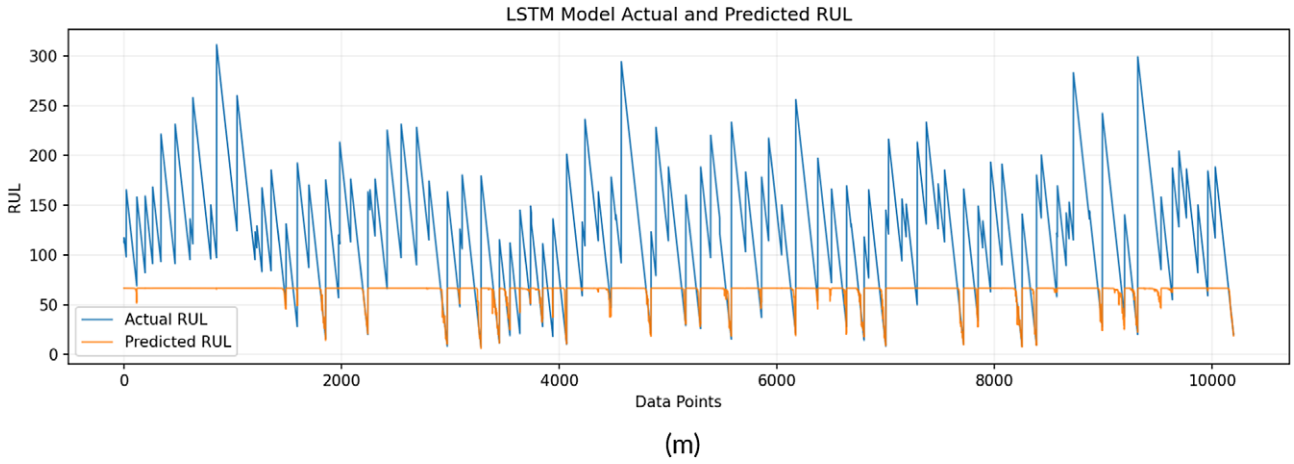
(k)



(l)

Figure 4. (Continued).

https://doi.org/10.1017/S0890060426100249 Published online by Cambridge University Press



(m)

Figure 4. (Continued).

Table 12. Comparison of accuracy results of ML/DL models from this study (Soni et al., 2021; Al Hasib et al., 2023; Sharma et al., 2023; Melkumian, 2024)

Model	This study	Sharma et al. (2023)	Soni et al. (2021)	Al Hasib et al. (2023)	Melkumian (2024)
Logistic Regression	0.950	0.88	0.80	N/A	N/A
Decision Tree	0.950	0.97	0.77	N/A	N/A
Random Forest	0.968	0.98	0.81	0.961	0.88
SVM	0.950	0.88	N/A	N/A	N/A
KNN	0.950	0.95	N/A	0.973	N/A
Naïve Bayes	0.950	0.97	N/A	0.930	N/A
XGBoost	0.950	N/A	N/A	N/A	0.89
LightGBM	0.972	N/A	N/A	N/A	N/A
CatBoost	0.971	N/A	N/A	N/A	N/A
Gradient Boosting	0.969	N/A	N/A	0.956	N/A
MLP	0.950	N/A	N/A	N/A	N/A
GRU	0.975	N/A	N/A	0.978	N/A
LSTM	0.981	N/A	N/A	0.9642	0.86

Overall and per-model performance

Table 11 provides per-class precision/recall/F1 together with global summaries on the untouched test set. Among classical ML models, gradient-boosted tree families were consistently competitive: LightGBM (Accuracy = 0.972, Macro-F1 = 0.86, ROC–AUC = 0.93), CatBoost (Accuracy = 0.971, Macro-F1 = 0.85, ROC–AUC = 0.92), Gradient Boosting (Accuracy = 0.969, Macro-F1 = 0.84, ROC–AUC = 0.92), and RF (Accuracy = 0.968, Macro-F1 = 0.83, ROC–AUC = 0.91). Linear/instance-based baselines (LR, SVM, and KNN) yielded solid but comparatively lower Macro-F1 values (0.76–0.80 range).

Sequence models achieved the strongest overall performance: GRU attained Accuracy = 0.975, Macro-F1 = 0.94, ROC–AUC = 0.97; LSTM delivered the top accuracy at 0.981 with Macro-F1 = 0.92 and ROC–AUC = 0.96. These results indicate that architectures explicitly modeling temporal dependencies better capture short-horizon failure dynamics in C-MAPSS.

Per-class behavior and safety–cost trade-offs

Failure-class (*Class 1*) sensitivity is essential for safety-critical PdM. The best recalls for *Failure* ≤ 30 were observed with GRU (Recall = 0.96, Precision = 0.90) and LSTM (Recall = 0.94, Precision = 0.88), followed among ML models by LightGBM and CatBoost (Recall = 0.89 for Class 1 in both, with Class 0/1 precision–recall balanced around 0.83–0.87). RF emphasized recall for the minority class (Class 1 Recall = 0.86) at a modest precision cost (Class 1 Precision = 0.79), which may be desirable when missed detections are costlier than false alarms.

Robustness to class imbalance

To quantify sensitivity to class imbalance, this study compared No–SMOTE versus SMOTE conditions within cross-validation pipelines and reported metrics on the fixed test set (Table 11). Oversampling effects were model-specific: for margin-based

https://doi.org/10.1017/S0890060426100249 Published online by Cambridge University Press

SVM, SMOTE increased ROC–AUC while slightly reducing Accuracy/F1; for RF, SMOTE shifted the precision–recall balance toward higher recall with comparable F1; boosted trees showed small, directionally mixed changes with largely stable AUC. Overall, top-ranked models remained robust under the adopted feature space and labeling scheme.

Statistical comparison

A nonparametric repeated-measures analysis was conducted per metric across baselines. Friedman omnibus tests were significant for Accuracy, F1-score, and ROC–AUC ($p < 0.01$), supporting nonequal performance distributions across models. Post-hoc Wilcoxon signed-rank tests with Holm adjustment revealed multiple large-effect differences (e.g., linear models outperforming DTs; DTs underperforming vs. kernel/linear baselines in ROC–AUC), corroborating the ranking trends observed in Table 11. These findings strengthen the claim that temporal DL models and modern boosted ensembles provide statistically meaningful gains over weaker baselines under this study protocol.

Visualization of prediction behavior

Figure 4 presents example trajectories of actual versus predicted RUL across representative models, illustrating error structure and temporal bias patterns. While the classification target focuses on a 30-cycle failure horizon, these RUL plots are informative for understanding how regression-style signals relate to the classifier's operating point and for diagnosing under/over-prediction regimes that may impact threshold selection in deployment.

Cross-study benchmarking

To contextualize these findings, Table 12 compares accuracy figures with prior studies (Soni et al., 2021; Al Hasib et al., 2023; Sharma et al., 2023; Melkumian, 2024). Consistent with the literature, boosted ensembles (RF and Gradient Boosting) and sequence models (GRU and LSTM) reside near the accuracy frontier. Notably, this study's LSTM (0.981) and GRU (0.975) accuracies are on par with or exceed results reported for comparable settings, while LightGBM and CatBoost compare favorably where published baselines exist. Differences across studies are plausibly attributable to horizon labeling choices, feature engineering, and hyperparameter search scopes.

Study limitations and directions for future research

Reliance on simulated data and external validity

This study evaluates model performance exclusively on the NASA C-MAPSS turbofan simulation, which – while physics-based and widely used – remains a proxy for operational engines. As such, the reported metrics may overestimate real-world effectiveness due to domain shift between simulated and field telemetry (e.g., sensor noise characteristics, calibration drift, flight-phase distributions, maintenance actions, and fleet/engine heterogeneity). Labeling in C-MAPSS (run-to-failure trajectories with a well-defined horizon) is also more pristine than operational labels derived from work orders or deferred defect logs, which can introduce timestamp uncertainty and class noise. Consequently, generalizability to airline ACMS/FDR data and to different engine families should be interpreted with caution.

Mitigations and future validation path

To strengthen external validity, future work should (i) perform external validation on independent real-engine datasets across multiple fleets and operating environments; (ii) explicitly quantify domain shift and apply domain-adaptation techniques (e.g., feature alignment or adversarial adaptation) and/or physics-informed augmentation to bridge sensor/operating-regime differences; (iii) evaluate robustness under realistic noise, missingness, and sensor dropout; (iv) calibrate decision thresholds using cost-sensitive criteria that reflect airline safety and false-alarm costs; and (v) report uncertainty (e.g., confidence/credibility intervals and calibration curves) alongside point estimates. Mapping C-MAPSS variables to airline telemetry, harmonizing failure-within-horizon labels with actual shop-visit or removal events, and conducting prospective pilots with maintenance stakeholders would provide a clearer assessment of deployability.

Discussion and conclusion

This study systematically compared the strengths and weaknesses of diverse ML/DL approaches for aircraft-engine PdM on the C-MAPSS dataset, framed as a binary classification task – "failure within the next 30 cycles." Sequence-based deep architectures that explicitly model temporal dependencies (GRU/LSTM) achieved the highest overall accuracy and sensitivity for the positive class (≤30 cycles to failure); LSTM delivered the top performance, with GRU closely behind. Gradient-boosted tree families (LightGBM, CatBoost, and Gradient Boosting) and RF trailed only narrowly, especially on the safety-critical Class-1 recall. Taken together, these findings indicate that capturing temporal structure is decisive for short-horizon risk prediction, while well-tuned boosted trees remain a competitive and operationally attractive alternative.

On model explainability, SHAP analyses showed that feature importances were consistent in both magnitude and direction across candidate models and aligned with engineering intuition. This transparency provides actionable input for sensor strategy, threshold design, and alerting, and helps satisfy auditability requirements in safety-critical settings by furnishing a traceable decision rationale for model selection and maintenance-policy calibration.

Regarding class imbalance (the rarity of the positive class), SMOTE offered model-dependent benefits: for margin-based SVMs, AUC tended to increase even when accuracy/F1 saw limited trade-offs; for RF, recall often improved as precision decreased; with strong tree baselines and MLPs, metrics were generally stable. Practically, organizations that prioritize missed-failure avoidance can justifiably favor recall-oriented tuning (and/or cost-sensitive thresholds), whereas those facing high false-alarm costs may prefer precision-oriented operating points.

A rigorous statistical comparison framework (Friedman omnibus tests followed by Holm-corrected pairwise Wilcoxon tests on significant metrics) confirmed that the observed performance differences across models are statistically meaningful. In particular, linear/kernel methods and boosted trees outperform weaker baselines, while sequence models emerge as overall leaders. This supports that the reported rankings are not attributable to random variation.

Cross-study triangulation with the literature shows a convergent pattern: sequence-based DL and boosted trees approach the "accuracy frontier" for this problem class. Performance gaps across studies remain sensitive to choices in forecasting-horizon labeling,

https://doi.org/10.1017/S0890060426100249 Published online by Cambridge University Press

feature engineering, and hyperparameter search breadth – differences that should be considered when interpreting absolute scores.

Threats to validity and external generalizability

All findings were obtained on NASA's simulated C-MAPSS data. While physically grounded and widely adopted, these simulations can differ from operational telemetry (e.g., sensor noise, calibration drift, flight-phase distributions, and fleet/engine heterogeneity). Absolute metrics may therefore be optimistic in real-world fleets. Future work should address this with external validation on independent fleet data, domain adaptation techniques, robustness to noise and missingness, cost-sensitive thresholding, and principled uncertainty reporting.

Practical implications

(i) On routes where safety dominates cost (e.g., ETOPS-critical operations), deploy GRU/LSTM or recall-optimized RF/LightGBM, and set alert thresholds to minimize false negatives. (ii) Where false-alarm cost is high (e.g., expensive unscheduled removals), prefer LightGBM/CatBoost for balanced precision–recall and emphasize probability calibration and monitoring. (iii) Use explainability outputs to guide sensor-investment priorities and data-quality controls by elevating information-rich channels.

Conclusion

By reframing PdM as a near-term failure-risk classification problem (≤ 30 cycles) and comparing 10 classical ML and 3 DL models under a leakage-averse protocol, this work shows that sequence models lead overall, while boosted trees offer a strong, interpretable alternative. Stability under class imbalance, together with consistent explainability findings, suggests that operational decision-making for aircraft-engine health can be grounded in scientifically defensible evidence. Nevertheless, fleet-scale external validation and integration with cost-sensitive calibration remain essential to simultaneously maximize safety and efficiency.

Data availability and code statement. The data supporting the findings of this study are openly available in the NASA Ames Prognostics Data Repository at https://www.nasa.gov/content/prognostics-center-of-excellence-data-set-repository. The code used for the analysis is openly available at https://github.com/hkmtcn/cmapss-binary-failure-classification.

Author contribution. Hikmetcan Özcan was responsible for the conceptualization, methodology design, data analysis, model implementation, result interpretation, and manuscript writing.

Competing interests. The author declares none.

References

Ahmad WMTW, Ghani NLA and Drus SM (2019) Data mining techniques for disease risk prediction model: A systematic literature review. In *Recent Trends in Data Science and Soft Computing: Proceedings of the 3rd International Conference of Reliable Information and Communication Technology (IRICT 2018)*. Cham: Springer, pp. 40–46.

Al Daoud E (2019) Comparison between xgboost, lightgbm and catboost using a home credit dataset. *International Journal of Computer and Information Engineering* **13**(1), 6–10.

Al Hasib A, Rahman A, Khabir M and Shawon MTR (2023) An interpretable systematic review of machine learning models for predictive maintenance of aircraft engine. Available at https://arxiv.org/abs/2309.13310.

Alomari Y (2024) Shap-based insights for aerospace phm: Temporal feature importance, dependencies, robustness, and interaction analysis. *Results in Engineering* **21**, 101834. https://doi.org/10.1016/j.rineng.2024.101834.

Alomari AA, Andó R and Baptista M (2023) Advancing aircraft engine rul predictions: An interpretable integrated approach of feature engineering and aggregated feature importance. *Scientific Reports*, **13**(1), 13240. https://doi.org/10.1038/s41598-023-40315-1; Available at https://www.nature.com/articles/s41598-023-40315-1.

Bergstra J, Yamins D and Cox D (2013) Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of Machine Learning Research (PMLR)*. Atlanta, Georgia, USA: PMLR, pp. 115–123.

Capodieci A, Caricato A, Carlucci AP, Ficarella A, Mainetti L, Vergallo C and **ATI Associazione Termotecnica Italiana** (2020) Using different machine learning approaches to evaluate performance on spare parts request for aircraft engines. *E3S Web of Conferences* **197**, 11014.

Celikmih K, Inan O and Uguz H (2020) Failure prediction of aircraft equipment using machine learning with a hybrid data preparation method. *Scientific Programming* **2020**(1), 8616039.

Chawla NV, Bowyer KW, Hall LO and Kegelmeyer WP (2002) Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research* **16**, 321–357.

Cho K, van Merriënboer B, Bahdanau D and Bengio Y (2014) Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics (ACL), pp. 1724–1734.

Çınar ZM, Abdussalam Nuhu A, Zeeshan Q, Korhan O, Asmael M and Safaei B (2020) Machine learning in predictive maintenance towards sustainable smart manufacturing in industry 4.0. *Sustainability* **12**(19), 8211.

DeCastro JA (2008) A modular aero-propulsion system simulation of a large commercial aircraft engine. In *NASA Technical Reports (NTRS)/Related Tech Paper*. Available at https://ntrs.nasa.gov/api/citations/20080043619/downloads/20080043619.pdf.

EASA (2015) *EASA Artificial Intelligence Roadmap 1.0 Published.* Available at https://www.easa.europa.eu/en/newsroom-and-events/news/easa-artificial-intelligence-roadmap-10-published (accessed 20 December 2024).

Eker ÖF (2015) *A Hybrid Prognostic Methodology and its Application to Well-Controlled Engineering Systems*. Cranfield, Bedfordshire, United Kingdom: Cranfield University.

Frederick DK, DeCastro JA and Litt J (2007) *User's Guide for the Commercial Modular Aero-Propulsion System Simulation (c-mapss)*. NASA. NASA/TM-2007-215026.

Friedman JH (2001) Greedy function approximation: A gradient boosting machine. *Annals of Statistics* **29**(5), 1189–1232. https://doi.org/10.1214/aos/1013203451.

Hochreiter S and Schmidhuber J (1997) Long short-term memory. *Neural Computation* **9**(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

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. *Advances in Neural Information Processing Systems* **30**, 3146–3154.

Luna JM, Gennatas ED, Ungar LH, Eaton E, Diffenderfer ES, Jensen ST, Simone CB 2nd, Friedman JH, Solberg TD and Valdes G (2019) Building more accurate decision trees with the additive tree. *Proceedings of the National Academy of Sciences* **116**(40), 19887–19893.

Mathew V, Toby T, Singh V, Rao BM and Kumar MG (2017) Prediction of remaining useful lifetime (rul) of turbofan engine using machine learning. In *2017 IEEE International Conference on Circuits and Systems (ICCS)*. Thiruvananthapuram, India: IEEE, pp. 306–311.

Melkumian SA (2024) Predictive maintenance analysis of turbofan engine sensor data. *The Journal of Purdue Undergraduate Research*, **14**(1). https://doi.org/10.7771/2158-4052.1708; Available at https://docs.lib.purdue.edu/jpur/vol14/iss1/8/.

Miller J, Seegmiller C, Masoum MA, Shekaramiz M and Seibi AC (2023) Hyperparameter tuning of support vector machines for wind turbine detection using drones. In *2023 Intermountain Engineering, Technology and Computing (IETC)*. Provo, Utah, USA: IEEE, pp. 55–60.

https://doi.org/10.1017/S0890060426100249 Published online by Cambridge University Press

National General Aviation Flight Information Database (2024) *NGAFID: National General Aviation Flight Information Database*. Available at https://www.ngafid.org (accessed 29 June 2025).

Özcan H (2025) *Cmapss-Binary-Failure-Classification: Binary Failure Classification on Nasa c-mapss with ml & dl*. GitHub Repository, Commit 78f6213. Available at https://github.com/hkmtcn/cmapss-binary-failure-classification (accessed 10 February 2025).

Prokhorenkova L, **Gusev G**, **Vorobev A**, **Dorogush AV and Gulin A** (2018) Catboost: Unbiased boosting with categorical features. *Advances in Neural Information Processing Systems* **31**, 6638–6648.

Ranjan G, **Verma AK and Radhika S** (2019) K-nearest neighbors and grid search cv based real time fault monitoring system for industries. In *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*. Bombay, India: IEEE, pp. 1–5.

Raschka S (2018) *Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning*. CoRR, abs/1811.12808. Available at http://arxiv.org/abs/1811.12808.

Rumelhart DE, **Hinton GE and Williams RJ** (1986) Learning representations by back-propagating errors. *Nature* **323**(6088), 533–536. https://doi.org/10.1038/323533a0.

Sarkon GK, **Safaei B**, **Kenevisi MS**, **Arman S and Zeeshan Q** (2022) State-of-the-art review of machine learning applications in additive manufacturing; from design to manufacturing and property control. *Archives of Computational Methods in Engineering* **29**(7), 5663–5721.

Sateesh Babu G, **Zhao P and Li XL** (2016) Deep convolutional neural network based regression approach for estimation of remaining useful life. In *Database Systems for Advanced Applications: 21st International Conference, DASFAA 2016*. Cham: Springer, pp. 214–228.

Saxena A and Goebel K (2008) Turbofan engine degradation simulation data set. *NASA AMES Prognostics Data Repository* **18**, 878–887.

Saxena A, **Goebel K**, **Simon D and Eklund N** (2008) Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 International Conference on Prognostics and Health Management (PHM)*. Denver, Colorado, USA: IEEE, pp. 1–9.

Sharma DB, **Kodipalli A**, **Rao T**, **BR R**, **Sripradha and Nikita** (2023) Machine predictive maintenance classification using machine learning. In *2023 International Conference on Computational Intelligence for Information, Security and Communication Applications (CIISCA)*. Bengaluru, India: IEEE, pp. 308–313.

Singh D, **Kumar M**, **Arya K and Kumar S** (2020) Aircraft engine reliability analysis using machine learning algorithms. In *2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS)*. RUPNAGAR, India: IEEE, pp. 443–448.

Soni P, **Khan MA**, **Zubair M and Garg SK** (2021) Multiclass classification for predicting remaining useful life (rul) of the turbofan engine. In *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. Noida, India: IEEE, pp. 1023–1028.

Vabalas A, **Gowen E**, **Poliakoff E and Casson AJ** (2019) Machine learning algorithm validation with a limited sample size. *PLoS One* **14**(11), e0224365.

Van den Bergh J, **De Bruecker P**, **Beliën J and Peeters J** (2013) Aircraft maintenance operations: State of the art. *HUB Research Paper* **9**, 7–8.

Wang T, **Yu J**, **Siegel D and Lee J** (2008) A similarity-based prognostics approach for remaining useful life estimation of engineered systems. In *2008 International Conference on Prognostics and Health Management*. Denver, Colorado, USA: IEEE, pp. 1–6.

Xing B and Marwala T (2018) *Smart Maintenance for Human–Robot Interaction*. Springer.

Xu Y and Goodacre R (2018) On splitting training and validation set: A comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning. *Journal of Analysis and Testing* **2**(3), 249–262.

Yan W and Zhou JH (2017) Predictive modeling of aircraft systems failure using term frequency-inverse document frequency and random forest. In *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. Singapore: IEEE, pp. 828–831.

Youness G and Aalah A (2023) An explainable artificial intelligence approach for remaining useful life prediction. *Aerospace*, **10**(5). https://doi.org/10.3390/aerospace10050474; Available at https://www.mdpi.com/2226-4310/10/5/474.

Zheng S, **Ristovski K**, **Farahat A and Gupta C** (2017) Long short-term memory network for remaining useful life estimation. In *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*. Dallas, Texas, USA: IEEE, pp. 88–95.

Hikmetcan Özcan received his BSc, MS, and PhD degrees from the Computer Engineering Department at Kocaeli University, Kocaeli, Türkiye. He is currently an Assistant Professor in the Computer Engineering Department at Kocaeli University. His current research interests include image machine learning, artificial intelligence, cryptography, computer software and software engineering, database management, and human–computer interaction.

https://doi.org/10.1017/S0890060426100249 Published online by Cambridge University Press