electronics

MDPI

Article

A Predictive–Prescriptive Framework for HPC Storage Maintenance via Explainable Artificial Intelligence

Álvaro Carrasco-Aguilar [1,*], José Javier Galán Hernández [2,3,*], Ziwei Shu [4] and Jorge de Andrés-Sánchez [5]

[1] Department of Business Economics, Rey Juan Carlos University, 28933 Madrid, Spain
[2] Faculty of Statistics, Complutense University, Puerta de Hierro, 28040 Madrid, Spain
[3] ENAE Business School, 30100 Murcia, Spain
[4] Department of Marketing, Faculty of Statistics, Complutense University of Madrid, 28040 Madrid, Spain; ziweishu@ucm.es
[5] Social and Business Research Laboratory, Department of Business Management, Universitat Rovira i Virgili, Campus Bellissens, 43204 Reus, Spain; jorge.deandres@urv.cat
* Correspondence: alvaro.carrasco@urjc.es (Á.C.-A.); josejgal@ucm.es (J.J.G.H.)

Abstract

As High-Performance Computing (HPC) architectures evolve towards the Exascale, storage infrastructure reliability has emerged as a critical operational challenge, with traditional reactive and static preventive maintenance strategies proving increasingly insufficient. This study addresses this gap by proposing a comprehensive methodological framework for the transition from predictive to predictive-prescriptive maintenance in large-scale storage environments. By integrating the CRISP-DM industry standard with a multi-layered eXplainable Artificial Intelligence (XAI) suite, we develop a system capable of isolating hardware degradation signals amidst massive volumes of routine telemetry. To validate our approach, we leveraged a publicly available disk failure dataset to evaluate multiple Machine Learning configurations, addressing the challenge of severe class imbalance through optimized oversampling and Gradient Boosting algorithms. The methodology employs global and local XAI techniques, including Permutation Feature Importance, SHAP, and surrogate decision trees, to translate probabilistic risk assessments into auditable hardware engineering rules. Our results demonstrate that this hybridization of robust predictive modeling with multi-layered explainability provides a transparent, evidence-based decision support system. Ultimately, we conclude that converting opaque risk predictions into technical justifications enables infrastructure managers to optimize hardware lifecycle management and minimize system downtime in mission-critical environments, establishing a viable pathway toward more resilient and auditable storage management.

Keywords: High-Performance Computing (HPC); predictive maintenance; eXplainable Artificial Intelligence (XAI); disk failure prediction; class imbalance; CRISP-DM; gradient boosting; prescriptive maintenance

Check for updates

Academic Editor: George Angelos Papadopoulos

Received: 30 April 2026
Revised: 8 June 2026
Accepted: 9 June 2026
Published: 17 June 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

High-Performance Computing (HPC) systems have become the backbone of modern scientific and business progress, supporting workloads that range from climate simulations to the training of large-scale artificial intelligence models [1]. As the architecture of these supercomputers grows in complexity and density, hardware reliability and availability emerge as critical challenges [2]. Within these infrastructures, the storage system consistently represents a bottleneck in terms of input/output performance and is also a

component vulnerable to mechanical and/or logical failures due to continuous operational stress [3,4].

Data center maintenance has long relied on reactive strategies, whereby faults are addressed only after they occur, or on preventive maintenance based on static heuristics, such as manufacturer alerts. Both paradigms are proving inefficient in modern architectures: the former results in unacceptable downtime, while the latter leads to the premature replacement of functional hardware [5]. To overcome these limitations, the scientific community has shifted toward predictive maintenance supported by Machine Learning (ML) algorithms. By monitoring telemetry, these models can identify imminent hardware failures and estimate the remaining useful life of critical components [6,7].

However, the implementation of ML in mission-critical HPC environments faces two major hurdles. First, a mathematical challenge: the telemetry data is characterized by a severe class imbalance, where hardware failures are rare events compared to routine operation cycles [8,9]. Second, a technological adoption challenge: high-performance models often operate as black boxes, lacking the transparency required for infrastructure managers to justify drastic actions, such as node isolation or data migration [10,11].

To address both challenges, this paper proposes a methodological framework that guides the transition from predictive to predictive-prescriptive maintenance in HPC storage architectures.

This proposal is based on the CRISP-DM industry standard, enriching it through deep integration of eXplainable Artificial Intelligence (XAI) techniques [10,12]. The scientific contribution of this work lies in the combination of a high-performance classification model (mathematically optimized to handle severe imbalance in cluster telemetry) with an explainability layer that translates probabilistic inference into hardware engineering rules. Using the operational logs of the Campaign Storage System documented in the Disk Failure Dataset by Ransom & George [13] as an empirical case study, we demonstrate how a Histogram-based Gradient Boosting algorithm (including techniques for handling imbalance) achieves high generalization power for anomaly detection. Simultaneously, a suite of global and local XAI tools is deployed to unravel the model's internal logic. Global techniques (including Permutation Feature Importance (PFI), Partial Dependence Plots (PDP), and surrogate decision trees) characterize the overall failure patterns, while local diagnostic tools—Individual Conditional Expectation (ICE) and Shapley Additive Explanations (SHAP)—reveal the nonlinear interactions between temporal stress and the spatial topology of the infrastructure. This predictive-prescriptive approach goes beyond failure prediction. It also reveals why a component is likely to fail and under which operational conditions, giving HPC operators the evidence needed to reschedule workloads or proactively isolate at-risk components. As detailed in Section 2, existing approaches typically address class imbalance and model interpretability in isolation, providing only partial insights insufficient to drive operational prescriptions; a unified methodology that rigorously addresses both simultaneously has not been documented in the HPC storage literature.

The remainder of this article is structured as follows: Section 2 reviews the state of the art in predictive maintenance and XAI applied to data centers; Section 3 formalizes the theoretical, mathematical, and algorithmic foundations underpinning the model; Section 4 details the iterative application of the CRISP-DM methodology to the case study; Section 5 presents and discusses the predictive and prescriptive results obtained; finally, Section 6 summarizes the most relevant conclusions and outlines future lines of research.

2. Related Works

The growing complexity of HPC systems and cloud data centers has driven both scientific and industrial communities to develop data-driven solutions to ensure hardware reliability. This trend of deploying advanced machine learning models is part of a broader paradigm shift across disciplines; for instance, deep learning architectures have demonstrated robust predictive capabilities in highly complex, non-linear physical contexts, such as forecasting space weather and ionospheric dynamics [14], while fault prediction methodologies have been successfully extended to critical mechanical structures through the integration of digital twin frameworks and deep transfer learning [15]. While these diverse applications underscore the wide-ranging potential of proactive diagnostics, a key challenge remains the systematic optimization of these pipelines to address the specific failure modes of large-scale computing infrastructures.

To establish a solid foundation for this study, a comprehensive literature search was conducted across three major academic databases: Scopus, Web of Science (WoS), and Google Scholar. The search strategy focused on recent advancements published after 2015, utilizing a combination of keywords related to the operational environment (e.g., "High-Performance Computing", "data centers", "server clusters"), the target problem (e.g., "predictive maintenance", "failure prediction", "anomaly detection"), and the analytical methodologies applied (e.g., "machine learning", "deep learning", "artificial intelligence").

The retrieved literature highlights three major research axes in this domain. The first axis concerns the selection of robust algorithms capable of processing complex telemetry logs, such as Self-Monitoring, Analysis, and Reporting Technology (SMART) attributes for storage devices. The second axis addresses the mitigation of severe class imbalance, given that failure events are extremely rare anomalies compared to millions of normal operational cycles. The third axis emphasizes the urgent need for XAI and white-box models to translate opaque, probabilistic predictions into auditable engineering rules.

Table 1 summarizes the most relevant contributions identified in the literature, categorizing them by the problem solved, the ML/DL algorithms applied, the data balancing strategies used, and the integration of XAI or intrinsically interpretable techniques.

Table 1. Summary of related works in HPC failure prediction: ML techniques, Balance strategies, and XAI.

Authors	Ref.	Solved Problem (HPC/Failure Prediction)	Analytical Techniques	Balancing Techniques	XAI/White-Box Techniques
Li et al. (2025)	[16]	Hard disk drive failure prediction focusing on transparency and trust for decision-makers.	XGBoost.	Random undersampling of operational drives	SHAP to visualize feature impact and extract explainable metrics.
Gebreyesus et al. (2024)	[10]	Interpretability and transparency of predictive maintenance models for Data Center operations.	Random Forest, XGBoost, and LSTM.		SHAP with TreeSHAP and DeepSHAP explainers; TSHAP (Temporal SHAP) for time-series analysis.
Gupta & Asirvatharaj (2025)	[17]	Drive failure prediction facing high data imbalance and non-linear S.M.A.R.T. relations.	BiLSTM with Multi-head Attention, XGBoost.	SMOTE and class-weighted loss function.	
Ahmed & Green II (2024)	[18]	Hard disk drive failure prediction dealing with extreme class imbalance.	LSTM networks	Modified focal loss function and Weighted Cross-Entropy (WCE) loss.	
Cornali et al. (2025)	[19]	Interpretable anomaly detection in multivariate time series for IT infrastructures (Tier-1 data center).	Graph Neural Networks, Generative Adversarial Networks, Autoencoders.		Interpretable deep learning mapping normal behavior via reconstruction errors and discriminator scores.

Table 1. *Cont.*

Authors	Ref.	Solved Problem (HPC/Failure Prediction)	Analytical Techniques	Balancing Techniques	XAI/White-Box Techniques
Guan et al. (2023)	[2]	Disk failure prediction facing simultaneous issues (e.g., minority samples and model aging).	Model-Agnostic Meta-Learning framework.	Multi-property Data Partitioning (MDP).	
Ji et al. (2022)	[20]	Fast and accurate disk failure prediction with extremely unbalanced positive/ negative samples.	LightGBM.	Under-sampling strategy via Mini Batch K-Means clustering and Focal Loss.	
Ahmed & Green II (2022)	[7]	Mitigation of severe class imbalance in HDD data to predict drive failures.	Tree-based methods, ensemble learning, Weighted Logistic Regression.	SMOTE and cost-sensitive learning.	
Zhang et al. (2022)	[21]	Minimization of false alarm rates and misclassification costs for disk failure prediction.	Meta-algorithm based on cost-sensitive learning, Genetic Algorithm.	Cost-sensitive learning framework.	
Chakraborttii & Litz (2020)	[22]	Prediction and explanation of solid-state drive (SSD) failures in hyperscale data centers.	Generative Adversarial Networks, 1-class Isolation Forest, LSTMs.	Ignoring the minority class for training (1-class anomaly detection approach).	1-class Autoencoders to enable model interpretability and reason about root causes.
Amram et al. (2021)	[8]	Interpretable predictive maintenance for hard drives to derive insights into underlying failure mechanisms.	Decision Trees.		White-box approaches (intrinsically interpretable models to provide insights on short/long-term health).
Our proposal		End-to-end predictive-prescriptive maintenance for HPC storage, bridging the gap between robust failure prediction and actionable, auditable maintenance decisions.	Model-agnostic competitive approach evaluating a broad set of ML algorithms (linear, kernel-based, tree-based, ensemble, and neural) under a unified temporal cross-validation protocol (TimeSeriesSplit), ensuring leakage-free model selection.	Comprehensive empirical benchmarking of all major balancing strategies within each fold: random undersampling, random oversampling, SMOTE, and cost-sensitive class weighting. Optimal strategy selected via AUC maximization.	Multi-layered XAI suite built on top of the winning predictive model to translate its probabilistic predictions into auditable engineering rules through two complementary inference levels: global explainability (PFI, PDP, Surrogate Decision Trees), which characterizes overall failure patterns, and local explainability (ICE, SHAP, Beeswarm), which provides instance-level diagnostics. Triangulating inferences across both levels enables the extraction of actionable operational thresholds.

Although these studies represent important advances in hardware failure prediction, they tend to address class imbalance and model interpretability separately, or through a narrow methodological lens. For instance, works proposing robust solutions for extreme data imbalance [7,17,18,20] rarely incorporate explainability frameworks, leaving the predictive models as opaque black boxes. Conversely, studies focusing on interpretability or white-box approaches [8,10,16,19,22] tend to rely on a single XAI technique (such as solely using SHAP [10,16] or isolated autoencoders [19,22]). Relying on a single metric often provides only partial insights and lacks the depth required to formulate actionable prescriptive maintenance strategies.

In contrast, and as made explicit in the last row of Table 1, our methodology distinguishes itself by addressing all three research axes simultaneously. At the predictive level, a model-agnostic competitive approach evaluates a broad set of ML algorithm families (linear, kernel-based, tree-based, ensemble, and neural) under a unified temporal cross-validation protocol, ensuring leakage-free model selection. At the balancing level, all major strategies are benchmarked empirically within each fold, with the optimal treatment identified via AUC maximization rather than assumed a priori. At the explainability level, a multi-layered XAI suite is built on top of the winning predictive model to translate its probabilistic output into auditable engineering rules through two complementary inference levels: global explainability (PFI, PDP, Surrogate Decision Trees), which characterizes overall failure patterns, and local explainability (ICE, SHAP, Beeswarm), which provides instance-level diagnostics. Combining both levels allows each finding to be confirmed from multiple analytical perspectives, providing the confidence required to translate model outputs into concrete maintenance protocols.

3. Theoretical Foundations

The development of an advanced system for predictive and prescriptive maintenance in HPC storage infrastructures requires the convergence of robust ML algorithms [23], data bias mitigation strategies, and XAI techniques [10,11,24,25]. This section formalizes the mathematical and algorithmic concepts underpinning the proposed methodology.

3.1. Formal Definition and Notation

We define the predictive maintenance problem as a binary classification task. Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ be a historical dataset, where N is the total number of recorded events.

- $\mathbf{x}_i \in \mathbb{R}^d$ represents the feature vector associated with the nth event i, $\mathbf{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,d}]^\top$, capturing d dimensions that include hardware and usage characteristics.
- $y_i \in \{0, 1\}$ denotes the target variable, where $y_i = 0$ represents a normal operation or reset cycle (cycle), and $\mathrm{y_i} = 1$ indicates a critical failure or degradation requiring physical replacement of the disk (replacement).

The objective of the predictive model is to learn a mapping function

$$f : \mathbb{R}^d \to [0, 1] \text{ such that } f(\mathbf{x}_i) = P(y_i = 1 | \mathbf{x}_i), \tag{1}$$

that is, to estimate the probability of imminent failure given the current state of the hardware. Based on this probability, an optimal decision threshold τ is applied to issue the final prediction

$$\hat{y}_i = \mathbb{I}(f(\mathbf{x}_i) \geq \tau). \tag{2}$$

3.2. Handling Class Imbalance

In the context of HPC storage infrastructure, hardware failures ($y = 1$) constitute anomalous and rare events compared to the volume of normal operations ($y = 0$). We define $\mathcal{D}_0 = \{(x_i, y_i) \in \mathcal{D} | y_i = 0\}$ as the subset of data corresponding to normal operations, and $\mathcal{D}_1 = \{(x_i, y_i) \in \mathcal{D} | y_i = 1\}$ as the subset of critical failures. The disproportion in the cardinality of these subsets, such that $N_1 = |\mathcal{D}_1| \ll N_0 = |\mathcal{D}_0|$, generates a highly imbalanced dataset. If left unaddressed, this bias causes ML algorithms to learn to distinguish the majority class with high accuracy but fail to classify the minority class. The fundamental goal of these mitigation strategies is to transform the original training set into a new balanced set $\mathcal{D}_{train}^{bal} = \mathcal{S}_0 \cup \mathcal{S}_1$ where $|\mathcal{S}_0| \approx |\mathcal{S}_1|$, forcing the learning algorithm to optimize the loss function over an equitable class distribution. The main approaches are as follows [26]:

- Undersampling: This involves extracting a random subset $\mathcal{S}_{under} \subset \mathcal{D}_0$ such that $|\mathcal{S}_{under}| = N_1$, resulting in a balanced set $\mathcal{D}_{train}^{bal} = \mathcal{S}_{under} \cup \mathcal{D}_1$ where $|\mathcal{S}_{under}| = |\mathcal{D}_1|$.
- Oversampling: It reshapes the data space by randomly duplicating instances of the minority class with replacement until a set $\mathcal{S}_{over}$ is created where $|\mathcal{S}_{over}| = N_0$. The resulting set is $\mathcal{D}_{train}^{bal} = \mathcal{D}_0 \cup \mathcal{S}_{over}$.
- SMOTE (Synthetic Minority Oversampling Technique): Generates new synthetic samples $\mathcal{S}_{syn}$ through interpolation in the feature space. For a minority sample $\mathbf{x}_i \in \mathcal{D}_1$, a nearby neighbor $\mathbf{x}_{zi}$ is selected, and a new point $\mathbf{x}_{new}$ is created as follows:

$$\mathbf{x}_{new} = \mathbf{x}_i + \lambda(\mathbf{x}_{zi} - \mathbf{x}_i) \tag{3}$$

  where $\lambda \sim \mathcal{U}(0, 1)$ is a random factor. The final training set is $\mathcal{D}_{train}^{bal} = \mathcal{D}_0 \cup \mathcal{D}_1 \cup \mathcal{S}_{syn}$.
- Cost-Sensitive Learning (Class Weights): Unlike the previous methods, this approach does not generate a new set $\mathcal{D}_{train}^{bal}$, but rather retains the original $\mathcal{D}_{train}$. Balancing

is achieved by modifying the algorithm's loss function, introducing weights w_0 and w_1 inversely proportional to the class frequencies (N_0 and N_1), which asymmetrically penalizes errors during optimization.

3.3. Predictive Model: Optimized Gradient Boosting

Following the exhaustive experimentation phase described in Section 4, the Gradient Boosting algorithm demonstrated technical superiority in generalization ability. Gradient Boosting is an ensemble model based on the sequential and additive construction of weak estimators, typically decision trees. The raw estimate $F_M(\mathbf{x})$ after M iterations is defined as:

$$F_M(\mathbf{x}) = \sum_{m=1}^{M} \gamma_m h_m(\mathbf{x}) \tag{4}$$

where $h_m(\mathbf{x})$ is the tree fitted in the iteration m to model the pseudo-residual (the negative gradient of the loss function L with respect to the previous prediction), and γ_m is the learning rate. To link this estimate to the probability function $f(\mathbf{x}) \in [0,1]$ defined in Section 3.1, the output $F_M(\mathbf{x})$ (log-odds) is transformed using the sigmoid logistic function:

$$f(\mathbf{x}) = \frac{1}{1 + e^{-F_M(\mathbf{x})}} \tag{5}$$

To address the scalability required in HPC environments, the selected implementation employs the histogram-based Gradient Boosting technique. This method groups continuous features $\mathbf{x} \in \mathbb{R}^d$ into b discrete bins (typically $b = 255$). This transformation reduces the computational complexity of finding the best split in the nodes of the decision tree from $\mathcal{O}(N\log N)$ to $\mathcal{O}(b)$, enabling high-performance training without a significant loss in predictive performance.

3.4. XAI for Prescriptive Maintenance

The transition from predictive to prescriptive maintenance is achieved through XAI techniques, which break down the model (our mapping function $f(\mathbf{x})$) that has proven most predictive—often a black-box model—into explanations that enable decision-making [10,11,24,25].

A. Global Explainability. Defines the macroscopic behavior of the model, that is, in our case, the general failure guidelines:

- Permutation Feature Importance (PFI): Measures the model's dependence on a specific feature x_j (where $j \in \{1, \ldots, d\}$). It is calculated by evaluating the empirical drop in a performance metric (such as AUC-ROC) when the values of x_j are randomly permuted in the dataset, destroying their relationship with the target variable y_i. The importance I_j is defined as

$$I_j = s_{orig} - \frac{1}{K}\sum_{k=1}^{K} s_{perm,j}^{(k)} \tag{6}$$

  where s_{orig} is the value of the performance metric evaluated on the original, intact dataset; K is the total number of permutation repetitions (necessary to ensure statistical robustness against randomness); and $s_{perm,j}^{(k)}$ is the performance obtained in the k-th iteration with the variable x_j permuted.
- Partial Dependence Plots (PDP): Visualizes the average marginal effect of a subset of features of interest $\mathbf{x}_S$ on the predicted probability $f(\mathbf{x})$. If we partition the original

feature vector into $\mathbf{x} = (\mathbf{x}_S, \mathbf{x}_C)$, where $\mathbf{x}_C$ represents the complementary features, the partial dependence function is

$$\bar{f}_S(\mathbf{x}_S) = \int f(\mathbf{x}_S, \mathbf{x}_C)\, d\mathbb{P}(\mathbf{x}_C) \approx \frac{1}{N}\sum_{i=1}^{N} f\left(\mathbf{x}_S, \mathbf{x}_C^{(i)}\right) \tag{7}$$

where $\mathbb{P}(\mathbf{x}_C)$ is the marginal probability distribution of the complementary features. In practice, the integral is approximated empirically by averaging the predictions over the N instances in the evaluation set, keeping $\mathbf{x}_S$ constant and using the observed values $\mathbf{x}_C^{(i)}$ for each instance i.

- Surrogate Models (Decision Trees): This involves training an intrinsically interpretable auxiliary model $g(\mathbf{x}) : \mathbb{R}^d \to [0, 1]$ (typically a decision tree with a maximum shallow depth) using the set of pairs $(\mathbf{x}_i, f(\mathbf{x}_i))$. By approximating the complex model's probabilistic predictions rather than the actual labels y_i, the surrogate model distills the complex decision boundaries of Gradient Boosting into Boolean rules that are readable by infrastructure operators.

B. Local Explainability. Provides granular justifications for specific alerts on individual disks:

- Individual Conditional Expectation (ICE): Unlike PDP, which collapses information by showing only the average effect, ICE disaggregates the dependency. It plots a separate curve $f\left(\mathbf{x}_S, \mathbf{x}_C^{(i)}\right)$ for each individual instance i in the dataset. This allows for the identification of hidden heterogeneous interactions, revealing cases where an increase in a feature raises the risk of failure in certain physical clusters but lowers it in others.
- SHAP (SHapley Additive exPlanations): Based on cooperative game theory, SHAP breaks down the exact probability of failure for an individual disk as the sum of the marginal contributions of each feature relative to the baseline (the model's average prediction $E[f(\mathbf{x})]$). The Shapley values ϕ_j ensure mathematical accuracy through the following equation:

$$\phi_j(\mathbf{x}) = \sum_{S \subseteq \mathcal{F} \setminus \{j\}} \frac{|S|!(|\mathcal{F}| - |S| - 1)!}{|\mathcal{F}|!} [f_x(S \cup \{j\}) - f_x(S)] \tag{8}$$

where $\mathcal{F} = \{1, \ldots, d\}$ is the set of indices of all features, $|\mathcal{F}| = d$ is the total number of dimensions, and S is any subset of features that does not include j, with cardinality $|S|$. The fraction represents the combinatorial weight of the possible permutations in which the feature can be added. Finally, $f_x(S)$ represents the expected value of the model conditional on the fact that only the features included in the subset S are known for that instance. Its global aggregation also allows for the construction of directional importance diagrams (Beeswarm Graphs).

4. Application of the Model to the Use Case: HPC Storage Architecture

The empirical validation of the theoretical foundations described in Section 3 is structured in strict accordance with the industry-standard CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology. The iterative nature of this framework allows the data lifecycle to be adapted to the specific characteristics of HPC architectures.

4.1. Business Understanding

In HPC environments, mass storage is among the most critical and failure-prone components. Disk management has traditionally followed a reactive approach (post-

mortem replacement) or static heuristics (generic manufacturer alerts), leading to systemic performance drops and high operational costs.

This article proposes a paradigm shift toward predictive and prescriptive maintenance. The challenge lies in learning a mapping function $f : \mathbb{R}^d \to [0, 1]$ capable of isolating the hardware degradation signal within a large volume of routine telemetry records. To this end, we use the Campaign Storage System as a case study, a mass storage infrastructure documented in the Disk Failure Dataset [13].

To address the problem from a predictive-prescriptive perspective, we must understand the nature of the system not as a set of static states, but as a continuous stream of telemetry events over time. Formally, we define the historical record as a temporal sequence of events $\mathcal{E} = \{e_1, e_2, \ldots, e_N\}$, where each event at a time i is defined by the tuple:

$$e_i = (t_i, c_{s,i}, c_{r,i}, c_{e,i}, c_{d,i}, y_i), \tag{9}$$

where t_i is the absolute timestamp, $c_{s,i}, c_{r,i}, c_{e,i}, c_{d,i}$ represent the exact topological coordinates of the hardware (Server, Rack, Enclosure, and Drive slot), and $y_i \in \{0 = \text{cycle}, 1 = \text{replacement}\}$ represents the triggered technical action.

The challenge consists of projecting this raw time sequence $\mathcal{E}$ into the standardized Euclidean space defined in Section 3.1. That is, we must transform each historical event e_i into an independent feature vector $x_i \in \mathbb{R}^d$ that contains information not only about the exact moment of failure but also the disk's degradation history. Based on this vector, the analytical objective is to predict whether the event will culminate in a routine cycle ($y_i = 0$) or in a physical replacement of the disk due to a critical failure ($y_i = 1$). Furthermore, algorithmic explainability must be provided to improve decision-making by infrastructure managers.

4.2. Initial Understanding and Data Preparation

The original dataset $\mathcal{D}$ consists of $N = 1389$ historical disk event records. Each event is initially defined by a feature vector e_i as described in the previous section.

The initial Exploratory Data Analysis (EDA) of these events ($\mathcal{E}$) confirmed the absence of missing values. However, the univariate analysis revealed a challenge characteristic of this mathematical domain: severe class imbalance. The majority class ($y_i = 0$) accounts for 82.2% of the events, while the critical class ($y_i = 1$) constitutes only 17.8%, as shown in Figure 1. Therefore, we can assume in this problem that $N_1 \ll N_0$, so the conditions are met to apply techniques for handling class imbalance (see Section 3.2).

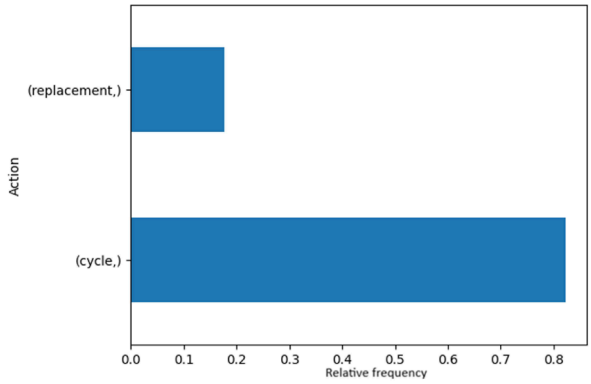


Figure 1. Distribution of the target variable y_i showing the class imbalance between normal operational cycles and critical physical replacements.

The direct use of these raw infrastructure events ($\mathcal{E}$) would introduce high cardinality and a severe risk of overfitting, since the variables Server, Rack, Enclosure, and Drive slot

identify specific elements of the current infrastructure that are not easily generalizable. Therefore, a thorough feature engineering process is necessary to replace these variables. This process transforms the raw event tuple e_i into the vector $\mathbf{x}_i$, extracting knowledge from the following operational dimensions:

1. Time window for slot degradation.
   - hours_since_last_event_slot: Measures the elapsed time (in hours) between event e_i and the most recent preceding event recorded in the exact same physical slot (identified jointly by Server, Rack, Enclosure, and Drive slot). The dataset is first sorted in strictly ascending chronological order. The feature is then computed as follows:
   $$\Delta t_i = (t_i - t_{prev})/3600 \tag{10}$$
   where t_{prev} is the timestamp of the immediately preceding event in the same slot, and the division by 3600 converts seconds to hours. If event e_i is the first recorded occurrence for that slot, the feature value is undefined and the cold start flag is activated (see flag_no_info_slot below), with the numerical value set to zero prior to scaling to avoid distorting the normalization step. Ties between events sharing the same timestamp within the same slot are resolved by their original record order in the raw log. Values close to zero indicate that a second fault has occurred very shortly after a previous one in the same location, reflecting a progressive deterioration of the drive's components (when $y_i = 1$).
   - flag_no_info_slot: Boolean variable introduced to address the cold start issue, indicating whether there is insufficient information for the event e_i to calculate the previous variable.
2. Weekly environment usage window. Failures in data centers are not isolated events; they are often correlated with thermal or electrical anomalies in the enclosure or rack. To capture this cross-stress, rolling windows were implemented and evaluated over the $[t_i - 7 \text{ days}, t_i)$ interval:
   - events_last_7d_enclosure: Count of all events occurring in the same enclosure ($c_{e,i}$) during the 7-day look-back window preceding event e_i. Formally:
   $$C_i^{enc} = |\{e_j : t_i - 7 \text{ days} \leq t_j < t_i \wedge c_{e,j} = c_{e,i}\}| \tag{11}$$
   - events_last_7d_rack: Analogous count aggregated at the rack level ($c_{r,i}$):
   $$C_i^{rack} = |\{e_j : t_i - 7 \text{ days} \leq t_j < t_i \wedge c_{r,j} = c_{r,i}\}| \tag{12}$$
   The strict exclusion of t_i itself ($t_j < t_i$) is critical: it ensures that the current event never contributes to its own window, preventing data leakage. Overlapping windows across different events on the same component are handled naturally—each event independently looks back exactly 7 days from its own timestamp, so windows overlap without interference. Events with duplicate timestamps within the same group are processed in original record order.
   - flag_no_info_7d: A binary flag that mitigates temporal bias for the first events in the dataset $i \to 1$, where the 7-day window does not yet have sufficient statistical history.
3. Historical risk of the rack infrastructure. The failure propensity was calculated at the rack level:
   - rack_historical_fail_rate: Cumulative rate of physical replacements relative to the total number of events occurring in the rack ($c_{r,i}$) up to (but not including)

event e_i. The use of a cumulative shift ensures that the rate at time t_i is computed exclusively from past observations, preserving temporal integrity.

- flag_no_info_rack: Boolean variable that is activated when there is no information available to calculate the above.

4. Time cycles. The absolute timestamp t_i was abstracted using cyclic transformations to detect the influence of workloads in the computing environment:
   - time: Continuous numerical variable (range 0–24) representing the exact hour of event t_i in decimal format (hours + minutes/60 + seconds/3600), allowing the model to detect operational interactions during peak periods of computational stress.
   - dayofweek: Nominal categorical variable encoding the day of the week (0 = Monday, 6 = Sunday).

4.3. Understanding the Preprocessed Data

Bivariate analysis of the transformed dataset shows that the distribution of temporal variables differs noticeably depending on whether the event corresponds to a routine cycle or a physical failure, as shown in Figure 2.

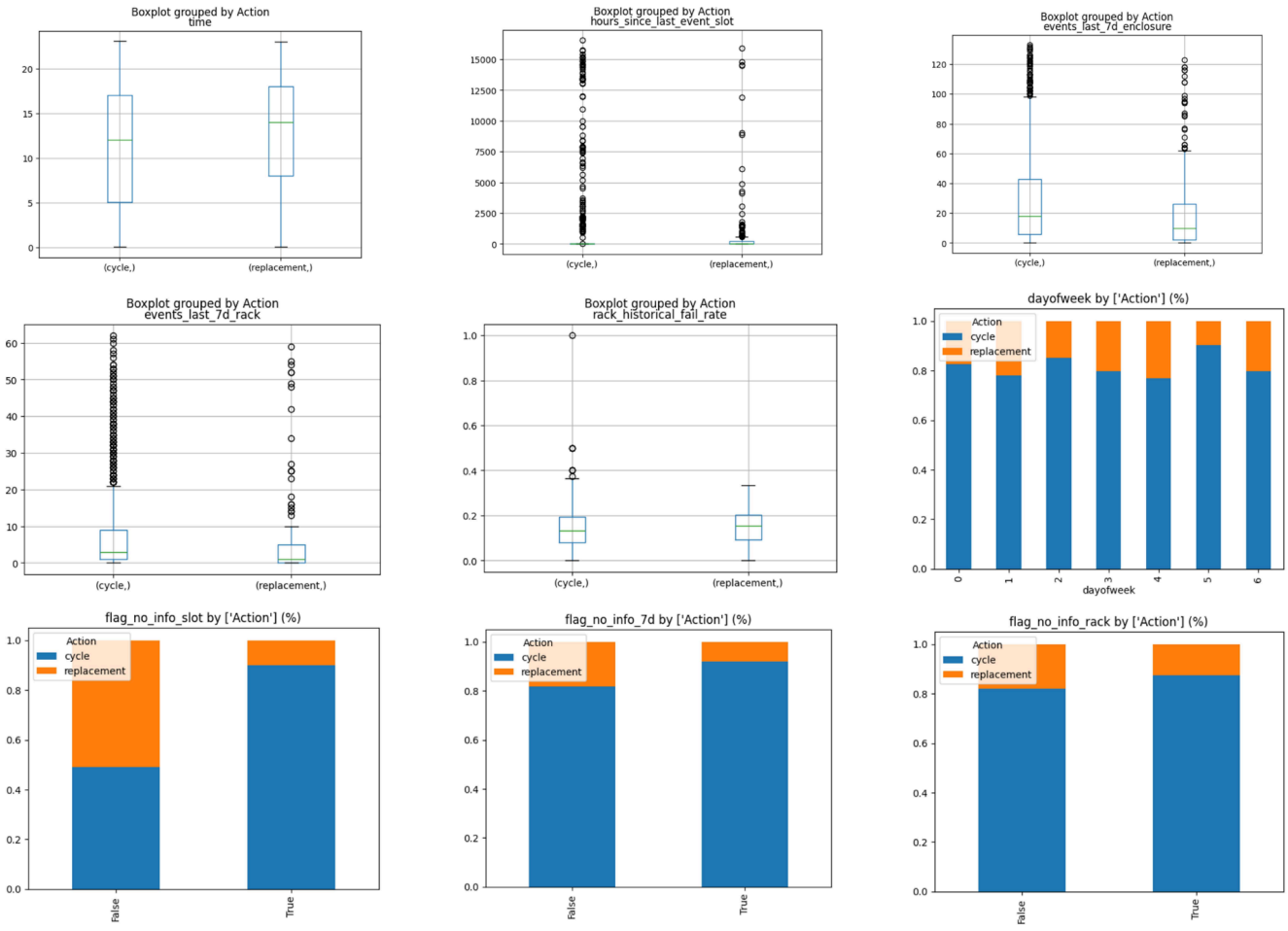


Figure 2. Distribution of independent variables grouped by the target class to compare routine cycles and hardware failures.

For the modeling phase, the dataset $\mathcal{D}$, already transformed, was split via sampling into a training set ($\mathcal{D}_{train}$) and a test set ($\mathcal{D}_{test}$). The test set consisted of the 25% of records that were most recent in terms of time, while the remaining 75% was used for model

training. Numeric variables were normalized using *Z-scores*, and categorical variables were encoded using one-hot encoding, expanding the feature space $d = 18$ dimensions.

4.4. Modeling

4.4.1. Generation and Training

As defined in Section 3, a large set of algorithms was configured and tested under the same conditions. The following algorithms were used: Logistic Regression (*LR*), Multi-Layer Perceptron (MLP), Decision Trees (DTs), K-Nearest Neighbors (KNNs), Support Vector Machines (SVMs), Random Forest (RF), and Gradient Boosting (GB), with the aim of comparing their performance. To address the imbalance mentioned in Section 3.2, each algorithm was combined with different strategies: undersampling (Under), oversampling (Over), SMOTE, the application of weights in the cost function (Weighted) for compatible models, and control configurations without any treatment for the imbalance (Base). Furthermore, for each of these variants, both the fine-tuning of the most common hyperparameters (Tuned) and the use of default parameters were evaluated. The exhaustive combination of all these approaches generated a total grid of 66 configurations. To prevent data leakage (leakage of future information into the past), resampling techniques were applied exclusively within the training partition of a temporal cross-validation scheme (TimeSeriesSplit with $k = 10$), respecting the chronological order of events at all times and leaving the validation fold intact and representative of reality. Appendix A provides a more detailed explanation of this process.

4.4.2. Selection of the Winning Model

The chosen optimization metric was the Area Under the ROC Curve (AUC), as it is insensitive to directional imbalance. The comparative analysis empirically determined that the Gradient Boosting configuration with oversampling and hyperparameter tuning (GB_Tuned_Over) maximized the performance of this metric, as can be seen in Figure 3.

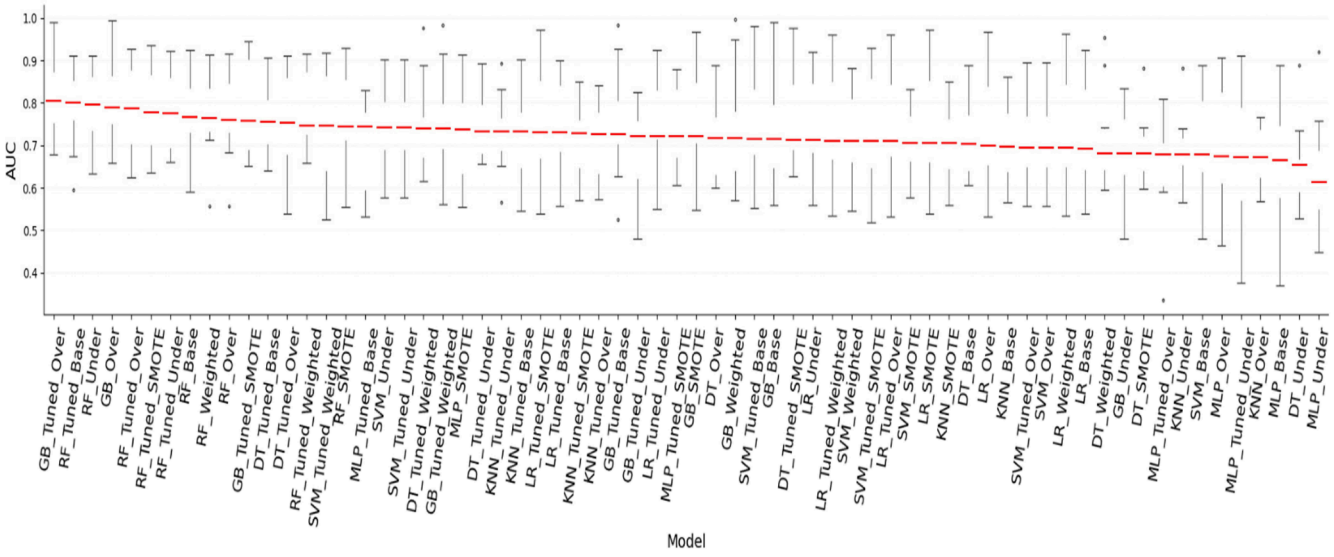


Figure 3. Comparative performance of the evaluated machine learning configurations to identify the optimal predictive model.

A comparative analysis of all configurations reveals that ensemble-based models (GB and RF) consistently outperform the rest of the evaluated algorithms, establishing themselves as the most robust and generalizable architectures for predictive management of the HPC infrastructure in our use case.

Therefore, we selected the model (labeled GB_Tuned_Over) that uses the Gradient Boosting technique (introduced in Section 3.3) in conjunction with the oversampling strategy (see Section 3.2) and a fine-tuning of its hyperparameters (for more details, see Appendix A). Compared to the use of asymmetric weights, the geometric balancing of the dataset through synthetic duplication of the minority class, combined with the optimization of tree depth and learning rate, proved to be mathematically more effective for isolating and propagating the weak failure signal ($y = 1$) in the ensemble function $F_M(\mathbf{x})$.

4.4.3. Threshold Optimization

To translate the model's predictive value into a real-world operational environment, it is necessary to establish an action protocol that converts the probabilities generated by the system into actionable maintenance decisions. The primary objective is to proactively identify which disks have a failure probability high enough to warrant technical intervention, thereby minimizing the impact on the storage system's availability. While the model provides a continuous output reflecting risk, daily operations require a binary decision: whether to keep the device in service or proceed with its immediate physical replacement. This choice represents a risk management exercise, where it is necessary to carefully balance the operational costs resulting from false alarms against the criticality of undetected failures. Thus, to transition from a probability $f(\mathbf{x}_i) \in [0, 1]$ to a concrete decision $\hat{y}_i \in \{0, 1\}$, Youden's statistical index J was applied (maximizing the sum of sensitivity and specificity). Iterative analysis of the ROC curve during validation mathematically determined an optimal threshold $\tau_{opt} \approx 0.1782$, deviating from the standard $\tau = 0.5$.

While Youden's J provides a mathematically objective baseline, in a production setting with a defined cost matrix (assigning explicit financial and operational weights to false positives, false negatives, true positives, and true negatives), metrics that prioritize recall, such as the F-Beta score, would be highly desirable to better align with the strict operational reality of storage maintenance. In HPC environments, the cost of a missed failure (false negative)—catastrophic data loss and extended system downtime—far exceeds that of a false alarm (false positive)—a preventive inspection that proves unnecessary.

4.5. Evaluation

In this phase, the selected model (GB_Tuned_Over) is evaluated, taking into account the calculated threshold τ_{opt}, on the test dataset $\mathcal{D}_{test}$ ($N = 348$ samples). This dataset remained strictly isolated throughout the training and selection process, allowing us to simulate the model's behavior with new data and validate its generalization ability in a production environment. The results obtained on the test set, as shown in Figure 4, have been satisfactory and will be analyzed in greater depth in Section 5.1.

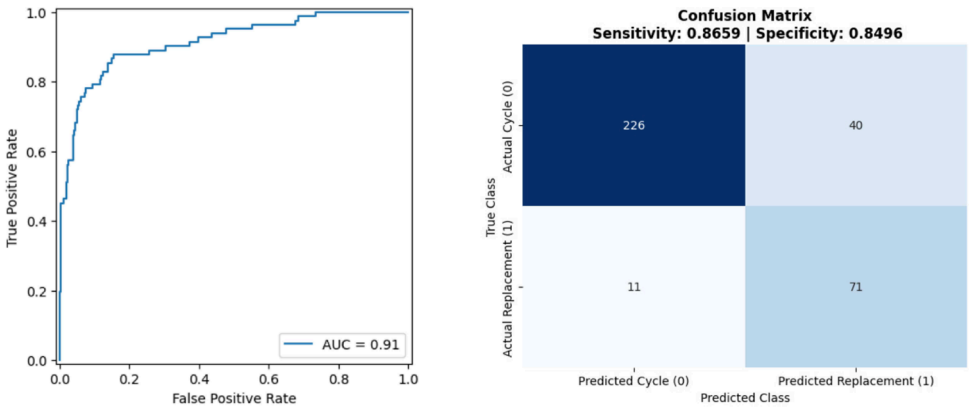


Figure 4. ROC curve and confusion matrix (with $\tau_{opt} \approx 0.1782$) of the selected model on the test set.

4.6. XAI Deployment: Prescriptive Maintenance

The deployment phase, the final stage of the CRISP-DM methodology, goes beyond the mere implementation of a model to focus on its effective integration into the organization's operational workflows. In HPC environments, automation based exclusively on predictions is insufficient; it is essential to establish a Trusted AI framework that allows for rigorous auditing of algorithmic decisions. With this objective in mind, the proposed deployment strategy is built upon the XAI fundamentals described in Section 3.4. Through this approach, it is possible to transition from a predictive maintenance model to a prescriptive paradigm, transforming the opaque logic of the black box (in this case, the GB_Tuned_Over model) into interpretable technical recommendations and operational engineering rules. To this end, we present the results of applying XAI techniques to our use case:

A. Global Structural Importance: PFI and Surrogate Decision Tree.

Quantifying the importance of the variables using the I_j index (defined in Section 3.4) reveals that the risk of failure is dominated by the four factors shown in Figure 5. As shown in this figure, the risk of failure is dominated by degradation factors and workload cycles. Specifically, the time elapsed since the last anomaly in a slot (hours_since_last_event_slot) is by far the most critical factor in the system (causing a drop in the AUC of more than 0.149 when permuted), followed by the time of the event (time). At a second level of importance, cross-stress at the infrastructure level emerges, evidenced by the relevance of events_last_7d_rack and events_last_7d_enclosure (Figure 5).

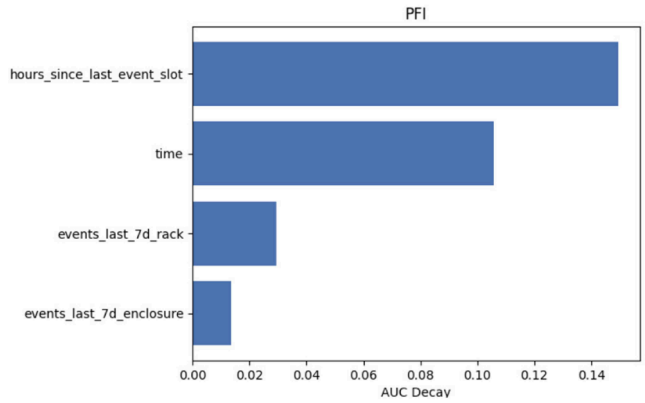


Figure 5. Feature importance ranking measured via PFI to identify key predictors of failure.

Additionally, a decision tree was trained as a surrogate model over the prediction space $\hat{y}$ generated by the selected model (GB_Tuned_Over). This approach allowed for the derivation of readable decision rules from the black-box model, maintaining 95.11% fidelity to the master model, as shown in Figure 6.

The topology of this surrogate tree reveals a fundamental prescriptive finding: the model divides cluster management into two major operational regimes, segregated from the root node based on the activation of the cold start variable, that is, without known prior history regarding the slot (flag_no_info_slot):

1. Known prior history. When the system already has historical information about a physical slot (flag_no_info_slot = False), the decision depends entirely on the frequency of the anomaly. If the new alert occurs less than 1040 h (approx. 43 days and 8 h) after the previous event, the terminal node classifies it as a critical failure (class: 1), regardless of the enclosure's stress level. This rule demonstrates that bursts of errors are the dominant predictor of mechanical failure: any disk that reports anomalies again within this timeframe requires immediate replacement. Conversely, if the disk

has exceeded 1040 h of stability, the event is consistently classified as a routine cycle (class: 0), assuming that the component has passed the critical degradation period.

2. Unknown prior history. When the disk has no history in its slot (flag_no_info_slot = True), the model shifts its strategy toward a diagnosis based on time windows and topological risk metrics. Instead of evaluating the disk's recurrence, the model audits the time and calendar of the event. Thus, on any day of the week except Friday (dayofweek_4 = False), events are mostly inferred as routine cycles (class: 0), unless they occur within a highly precise time window: strictly between 4:05 PM and 5:04 PM (time > 16.05 and time $\leq$ 17.04), which triggers a replacement alert (class: 1). Similarly, on Fridays (dayofweek_4 = True), the criteria are even stricter: if the rack shows absolute operational inactivity (events_last_7d_rack $\leq$ 0.50), detected anomalies are directly marked as critical failures (class: 1). This partition suggests that the model has detected a specific vulnerability of the new disks in relation to workload routines or scheduled maintenance at the end of the operational week.

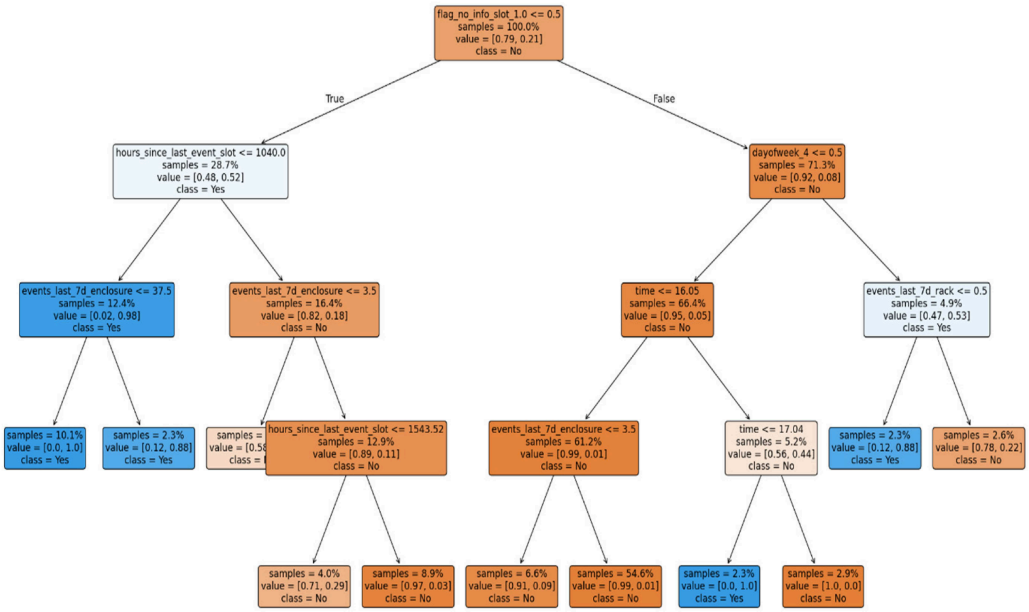


Figure 6. Surrogate decision tree of the winning model, translating complex predictions into readable decision rules.

PDPs visualize the average marginal functional relationship between each feature and the predicted probability of failure, as shown in Figure 7. The results describe distinct behaviors in the model's response:

- hours_since_last_event_slot: Shows an extremely high failure probability (greater than 0.9) at the initial time, followed by a sharp and drastic drop over the first 1000 h, until it asymptotically stabilizes around 0.2. This profile indicates that the risk is critical immediately after an incident and decreases rapidly over time.
- time: Shows a highly nonlinear and oscillatory dependency profile. The probability is not constant throughout the day but exhibits multiple peaks and troughs, indicating that the model has learned to associate specific times of day with a higher or lower propensity for failure, reaching its maximum values (close to 0.7) during late-night hours.
- events_last_7d_rack: Shows an inverse decreasing trend: the probability of failure reaches its maximum value (~0.35) when the counter for previous events in the rack is zero, and drops to a stable baseline level (close to 0.175) as the number of recorded events increases.

- events_last_7d_enclosure: Exhibits behavior similar to the previous one: the dependency value is higher (~0.32) when the event count is minimal, decreasing rapidly as events in the enclosure increase, and maintaining minor fluctuations at low probability levels.

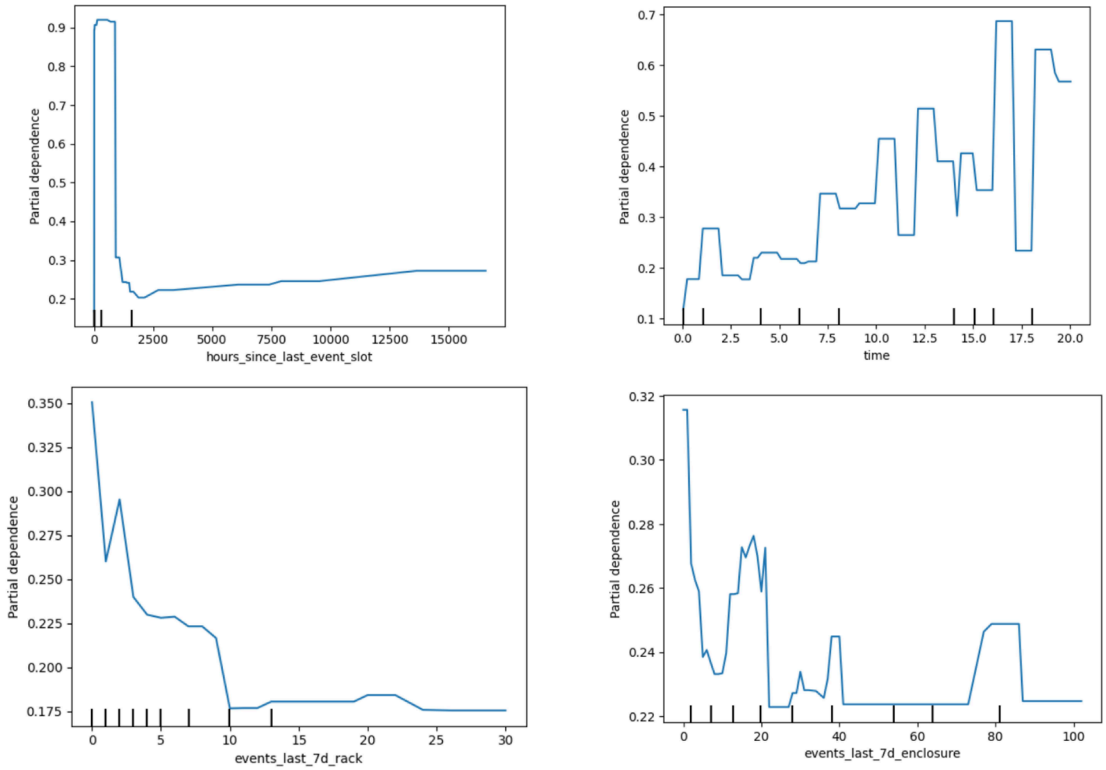


Figure 7. PDP illustrating the average marginal effect of the most significant variables on failure probability.

B. Local Explainability: SHAP and ICE.

Figure 8 shows the probability with which each variable contributes to predicting the target variable. This graph confirms the importance of the four variables determined by the PFI (Figure 5).

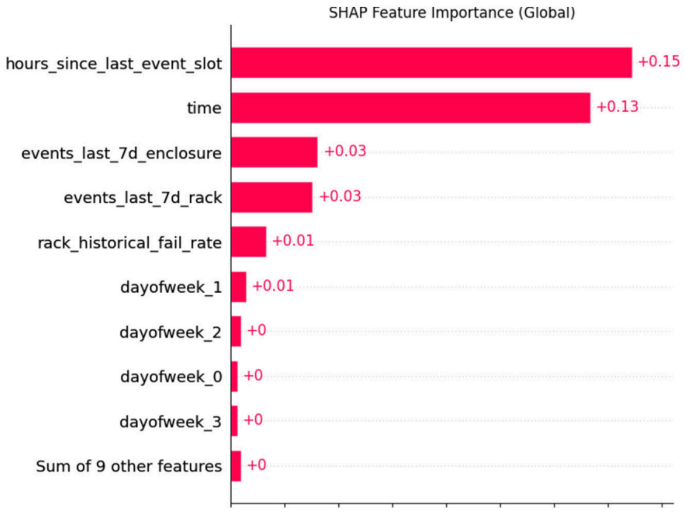


Figure 8. SHAP global feature importance ranking showing the overall contribution of each variable to model predictions.

Additionally, the SHAP beeswarm plot is also used to determine the value of the variables in terms of their impact on predicting the target variable, as shown in Figure 9. In this figure, each point represents the inference of a disk in the feature space. The color represents the feature value (blue for low, red for high), and the horizontal position indicates its impact on the model's output (moving to the right increases the probability of failure, while moving to the left decreases it). As mentioned, since the dataset is small, it was decided to generate this plot using all cases from the test set. Analysis of this diagram confirms that hours_since_last_event_slot acts as the variable with the greatest discriminatory power: low values (blue) are significantly concentrated toward the right, implying a substantial increase in the predicted probability of failure. On the other hand, variables such as events_last_7d_enclosure and events_last_7d_rack show that high values (red) have a positive influence on the failure prediction.

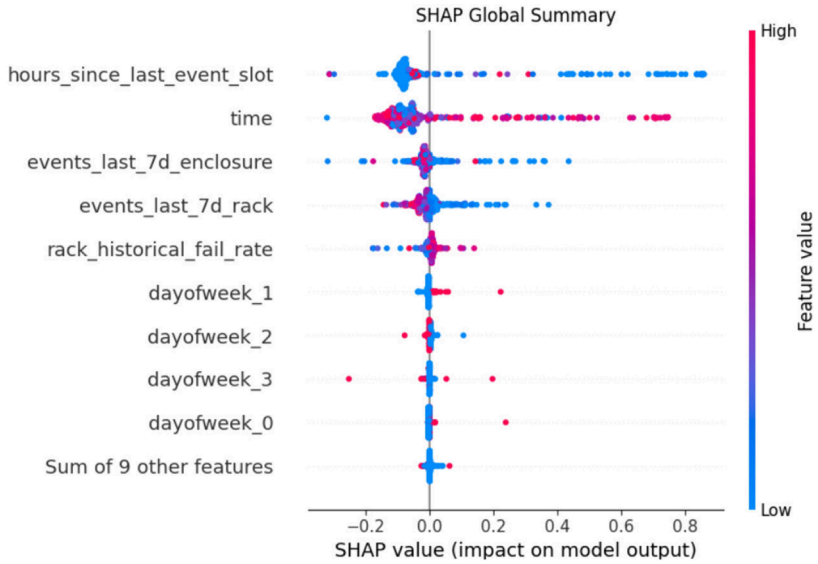


Figure 9. SHAP beeswarm plot showing the distribution and directional impact of feature values on predicted failure risk.

To verify the directional impact and possible local interactions of this variable, we applied the ICE technique. To do this, 100 random samples (individual disks) were selected from the test set, and their respective curves were calculated for these variables, as shown in Figure 10. Analysis of these graphs reveals notable heterogeneity in the model's response, going beyond the simplified view provided by the marginal averages of the PDPs. While for the variable hours_since_last_event_slot the curves exhibit highly convergent behavior (confirming that the 1040 h threshold is a universal inflection point toward stability for all disks), a marked divergence is observed in the temporal and topological context variables. In the graph corresponding to time, the disk trajectories intersect and exhibit varying risk peaks, demonstrating that the model does not apply a rigid hourly pattern to all devices, but rather dynamically adjusts its prediction based on unobserved local characteristics. Likewise, in the graphs for events_last_7d_rack and events_last_7d_enclosure, a clear segmentation is identified among the disk population: while a majority remains at baseline probability levels, a specific subset exhibits persistently high susceptibility to failure.

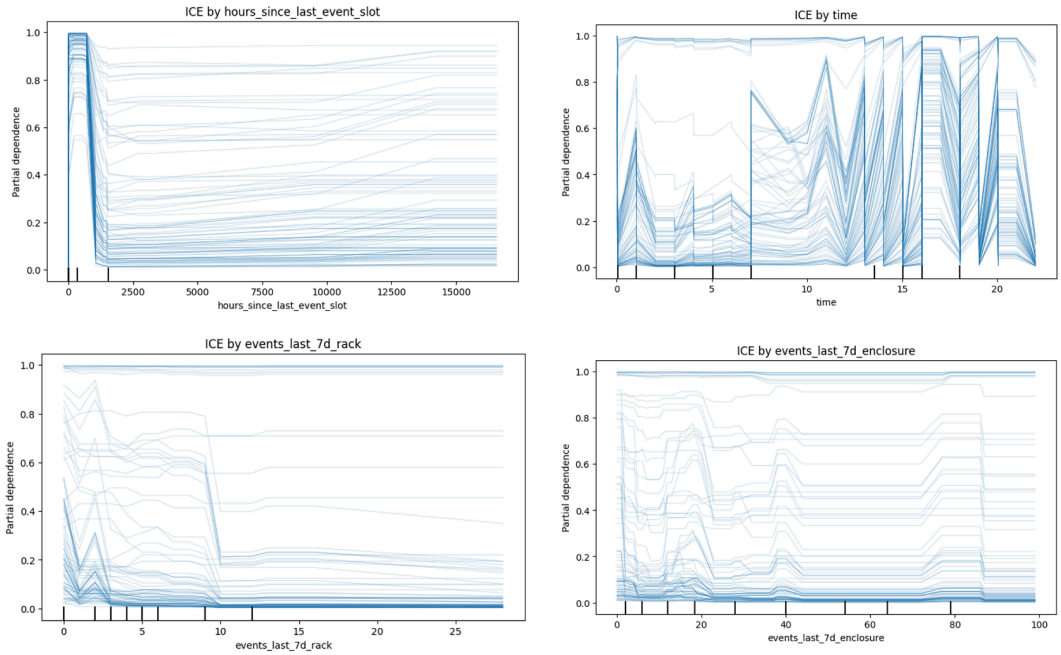


Figure 10. ICE plots for the most significant variables, based on 100 random samples from the test set.

5. Results and Discussion

The empirical implementation of the CRISP-DM methodology has enabled the development of an analytical system that not only anticipates physical failures in the HPC infrastructure but also provides an auditable explanatory framework. The findings are detailed below in two dimensions: the predictive robustness of the algorithm and the extraction of prescriptive knowledge via XAI.

5.1. Predictive Performance: Modeling and External Validation

To ensure that the model captures the underlying failure signal without overfitting to routine telemetry noise, the evaluation was structured into an internal cross-validation (CV) phase and a testing phase on a strictly isolated dataset ($\mathcal{D}_{test}$). During the modeling phase, the winning configuration (GB_Tuned_Over) demonstrated robust stability. The algorithm, combined with oversampling techniques and hyperparameter tuning, achieved an average AUC of 0.8153 in cross-validation, with a sensitivity of 61.84% and a specificity of 90.61%.

Final validation was performed on the test set ($N = 348$), simulating a production deployment. When evaluating the continuous probabilities $f(\mathbf{x}_i)$ generated by the model against this unseen data, an AUC of 0.9143 was obtained (see Figure 4). This performance on isolated data confirms the ensemble's robust generalization ability. To optimize the prescriptive operation, the probabilistic cut-off point (τ_{opt}) was not set by default at 0.5 but was mathematically derived by maximizing the Youden statistic J on the ROC curves. The resulting threshold, $\tau_{opt} \approx 0.1782$, reflects the asymmetry of risk in HPC: it is operationally preferable to slightly increase preventive inspections rather than tolerate the high cost of a catastrophic failure. Applying this τ_{opt} to $\mathcal{D}_{test}$, the model achieved a sensitivity of 86.59% and a specificity of 84.96%. This performance, with a (macro) F1-Score of 0.82, establishes an exceptional operational balance, ensuring high detection capability despite the severe initial imbalance in the dataset.

5.2. Prescriptive Analytics: From Global Structure to Local Interaction

The operational accuracy demonstrated by the winning model validates its predictive capability, but the value of this solution in mission-critical environments lies in its ability

to provide evidence-based decision support. The prescriptive analytics presented here transform the model's probabilistic output into operational maintenance rules. To this end, an XAI framework is deployed that enables the transition from probabilistic prediction to the technical justification of each intervention, ensuring that hardware isolation decisions are traceable and robust.

Thus, by integrating the results of variable importance using two different methods (Figures 5 and 8) with the surrogate tree structure (Figure 6), an operational convergence becomes evident: the risk of failure is not a random phenomenon, but a stochastic progression strongly linked to the recent temporal behavior of the hardware and its operational context.

The 1040 h rule (approx. 43 days) identified in the surrogate tree is not an arbitrary limit; the PDPs (Figure 7) validate that the model has learned to capture a real inflection point in the mechanical degradation of the disks. The vertical drop in partial dependency around 1040 h reflects how the model drastically minimizes the probability of failure once the component survives the initial burst of errors. This interpretation is supported by the SHAP values (Figure 9), where the impact of the variable hours_since_last_event_slot in blue tones (low values) is strongly projected to the right, confirming that bursts are the dominant predictor of mechanical failure. Therefore, from a prescriptive perspective, the model establishes a clear protocol: any disk exhibiting a recurrence of errors in less than 43 days must be replaced immediately, while those exceeding this threshold can be classified as operational, thereby reducing the rate of premature replacements.

The physical interpretation of this threshold is grounded in hardware reliability engineering: a low value of hours_since_last_event_slot indicates that a second fault has occurred shortly after a previous one in the same physical slot, which reflects a progressive deterioration of the drive's mechanical or electronic components and is a well-documented precursor to physical replacement. Once a component survives approximately 1040 h without recurrence, the model consistently classifies it as stable, suggesting that the critical degradation window has passed. This empirical threshold should not be confused with the manufacturer's MTBF (Mean Time Between Failures), which typically spans millions of hours. Instead, it represents an empirical "risk quarantine" and stabilization period following initial hardware stress. This aligns perfectly with pragmatic data center maintenance schedules, where flagged hardware is kept under strict observation for approximately a month and a half (about 43 days) before being cleared for standard operation, though a direct validation against manufacturer specifications is not possible here, as the dataset [13] does not include drive model metadata.

This comparison makes more sense when contrasted with the cold start procedure (disks with no history). While the PDP of the time variable (Figure 7) reveals high hourly volatility, the cross-plot with the surrogate tree (Figure 6) allows us to identify that this volatility is not noise, but a direct response to the supercomputer's workload. The model has determined that high-criticality windows (e.g., late-afternoon peaks) and weekly schedules (such as Fridays) are strongly predictive of failure for "cold-start" drives. While the dataset lacks direct job-scheduling logs to confirm this categorically, these temporal patterns plausibly correlate with typical HPC operational stress events, such as massive batch job scheduling, system-wide file cleanups, or scheduled backup operations. These routines act as systemic stress tests that undiagnosed or already vulnerable hardware cannot withstand. This could help infrastructure managers distinguish between a failure due to intrinsic degradation (the first regime) and a weakness induced by the operating environment (the second regime).

Finally, the integration of SHAP with ICE curves (Figure 10) provides the system with granular diagnostic capabilities. The heterogeneity revealed in the ICE curves

confirms that the model does not apply a rigid pattern to all disks, but rather dynamically adjusts its prediction using topological modulators such as events_last_7d_rack and events_last_7d_enclosure. The peaks in the ICE curves demonstrate that a disk with a good historical performance may see its risk increase if it is located in a rack with operational inactivity or a high accumulation of previous events, acting as an environmental catalyst. We hypothesize that this highly susceptible subset corresponds to disks located in micro-environments with physical anomalies (such as rack thermal hot-spots) or those assigned to high I/O logical partitions (e.g., scratch nodes). Consequently, the actionable insight is that disks exhibiting this specific topological susceptibility should bypass the standard 1040 h "quarantine" observation and be flagged for immediate proactive replacement or thermal relocation.

In conclusion, this comprehensive analytics approach allows systems engineers to move away from reactive maintenance toward justified intervention: the recommendation to take a node offline or replace a disk is based on a three-part diagnosis that combines the criticality of the error burst, the hourly context of the workload, and the topological state of the rack, ensuring evidence-based storage management. To bridge the gap from these diagnostic insights to prescriptive actions within a production workflow, infrastructure managers could map these rules to dynamic alert severities. For instance, a disk violating the 1040 h temporal stability rule could trigger a "critical alert" that automatically initiates data migration protocols and schedules an immediate replacement. Conversely, isolated anomalies (or those occurring well beyond this timeframe) could flag a "minor alert" to launch automated diagnostic scripts without disrupting the node's availability.

6. Conclusions and Future Work

The main objective of this article was to propose and validate a comprehensive methodological framework for the transition from reactive to prescriptive maintenance in HPC storage architectures. Through the iterative application of the CRISP-DM standard and the deep integration of XAI, it has been empirically demonstrated that it is possible to anticipate physical failures and decode the underlying logic of "black box" algorithms in mission-critical environments.

Comparing the results with the initial objectives, the main conclusions are as follows:

1. Predictive robustness and handling of class imbalance: It has been confirmed that the severe class imbalance intrinsic to HPC systems can be effectively mitigated through oversampling strategies and fine-tuning of hyperparameters. The winning configuration demonstrated outstanding generalization capability on unseen data, achieving a competitive AUC. Technical optimization of the decision threshold allowed for the establishment of an optimal operational balance between the sensitivity needed to detect imminent failures and the specificity required to keep the false alarm rate under control, meeting the requirements of minimizing system downtime without driving up operational costs.
2. From prediction to prescription via XAI: The goal of overcoming algorithmic opacity was achieved through a hierarchical concatenation of XAI techniques. The applied methodology demonstrated that failures in the Campaign Storage System are not stochastic events, but rather phenomena linked to degradation patterns and usage cycles. At the global level, importance analysis confirmed that risk is contingent on temporal and workload variables. At the intermediate level, rule extraction using the surrogate model revealed complex dynamics: risk emerges from the interaction between the disk's temporal stability (the operational "quarantine" threshold) and the work cycles scheduled in the data center. Finally, at the local level, SHAP analysis provided auditable justification for each alert, allowing operators to distinguish

between a failure due to mechanical wear and a vulnerability induced by system load testing.

3. Validation of the conceptual framework: It is confirmed that the hybridization of state-of-the-art ML algorithms with XAI techniques provides data center administrators with an evidence-based engineering tool. Preventive shutdown decisions no longer depend on generic lifespan estimates but rather on a dynamic and auditable diagnosis. While this case study has inherent limitations due to the volume of the dataset, the methodology has proven to be scalable. Applying this framework in Big Data environments will allow for the full analytical potential of XAI to be realized, extracting prescriptive rules of even greater complexity and precision.

While the results demonstrate the technical maturity of the proposed system, the implementation of this paradigm opens up new and compelling lines of research:

- The primary limitation of this study concerns the size of the dataset. The Disk Failure Dataset constitutes the complete operational record of the Campaign Storage System, and its relatively modest volume is therefore inherent to the case study rather than a methodological choice. Although multiple safeguards were adopted to ensure robust generalization (temporal cross-validation, strict test set isolation, and constrained tree complexity...), some of the fine-grained patterns identified by the model, such as the specific hourly windows or the day-of-week effects, may partially reflect the operational routines of this particular infrastructure. It should be noted, however, that the proposed framework is dataset-size agnostic and has proven to be theoretically scalable. The same methodology applies regardless of data volume, with parameters such as fold size and hyperparameter search grids adapted to the scale of each use case. Specifically, the use of histogram-based Gradient Boosting optimizes split-finding, making the training phase highly efficient. Empirically, the preprocessing and model-training phases were executed using cloud computing resources (Google Colab), where the most computationally demanding tasks, such as grid-search hyperparameter tuning, could be easily parallelized to support the framework's scalable performance. For application in Big Data environments, the proposed architecture is designed to be fully compatible with distributed processing pipelines (e.g., Apache Spark 3.5.8 MLlib). The primary computational bottleneck in such massive environments would be the exact extraction of local interpretability metrics; however, this can be mitigated using efficient approximations like TreeSHAP. To support this scalability, the implementation was built generically, with all configurable settings externalized into a YAML file (the full training protocol is documented in Appendix A), so that adapting the pipeline to a larger dataset requires only targeted parameter adjustments rather than changes to the core codebase. Ultimately, applying this framework in larger environments and validating it across diverse HPC datasets, such as those available from other national computing facilities, constitutes a priority for future research, allowing for the extraction of prescriptive rules of even greater complexity and precision.
- Agent-based prescriptive support via LLMs: Currently, interpreting XAI artifacts requires technical expertise. The natural next step is to feed the model's structured outputs (logical rules and impact metrics) into language models (LLMs) via dynamic prompts to generate automatic prescriptions in natural language, facilitating immediate decision-making by operators.
- Inference on real-time data streams: The model has been evaluated in a batch processing paradigm. The future architecture must adapt these components to process telemetry in real time using streaming technologies (e.g., Apache Kafka or Flink), requiring approximate Shapley calculations or iterative PFI updates with millisecond latencies.

- Multi-component expansion: Scaling the XAI methodology toward a holistic predictive model of the entire HPC node, including accelerators (GPUs/TPUs), RAM, and cooling systems, using graphs to analyze spatial correlations between components in relation to the risk of cascading failures.
- Feature portability and cross-facility validation: While this study focuses on a specific storage architecture, the temporal variables (timestamps) and topological metadata (Node, Rack, and Enclosure IDs) used to construct our features are ubiquitous standards within data center telemetry. Since these data points are readily available in virtually all modern HPC monitoring and scheduling systems (e.g., SLURM, Ganglia, Nagios, and Prometheus), our data engineering pipeline is highly portable. Consequently, a direct line of future work involves deploying and validating this framework across different storage systems and computing facilities to establish generalized, cross-platform prescriptive maintenance protocols.

Author Contributions: Conceptualization, Á.C.-A., J.J.G.H., Z.S. and J.d.A.-S.; methodology, Á.C.-A., J.J.G.H., Z.S. and J.d.A.-S.; software, Á.C.-A.; validation, Á.C.-A., J.J.G.H., Z.S. and J.d.A.-S.; formal analysis, Á.C.-A., J.J.G.H., Z.S. and J.d.A.-S.; investigation, Á.C.-A., J.J.G.H., Z.S. and J.d.A.-S.; resources, Á.C.-A., J.J.G.H., Z.S. and J.d.A.-S.; data curation, Á.C.-A.; writing—original draft preparation, Á.C.-A., J.J.G.H., Z.S. and J.d.A.-S.; writing—review and editing, Á.C.-A., J.J.G.H., Z.S. and J.d.A.-S.; visualization, Á.C.-A., J.J.G.H., Z.S. and J.d.A.-S.; supervision, Á.C.-A.; project administration, Á.C.-A., J.J.G.H., Z.S. and J.d.A.-S.; funding acquisition, Á.C.-A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received institutional and academic support from the Telefonica and the Telefonica Chair on Smart Cities of the Universitat Rovira i Virgili and Universitat de Barcelona (project number: 42. DB.00.18.00), and from ENAE Business School, Center of Excellence in Digitalization (grant DIGIT–062026–JJGH). No specific funding was received to cover publication costs.

Data Availability Statement: The original data presented in the study are openly available at https://www.osti.gov/biblio/2446874 (accessed on 15 April 2026).

Acknowledgments: The authors gratefully acknowledge the Universidad Complutense de Madrid, Universidad Rey Juan Carlos, ENAE Business School, and Universitat Rovira i Virgili for their continued support.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Model Configuration Grid and Training Protocol

This appendix provides a complete specification of the experimental design described in Section 4.4.1, intended to ensure full reproducibility of the reported results. The full implementation was developed in Python 3, leveraging the scikit-learn [27] and imbalanced-learn [28] libraries for model construction, pipeline orchestration, resampling, and evaluation. The exhaustive configuration grid comprises 66 model variants arising from the factorial combination of seven algorithm families (Logistic Regression, Decision Tree, K-Nearest Neighbors, Support Vector Machine, Random Forest, Histogram-based Gradient Boosting, and Multi-Layer Perceptron) with five imbalance treatment strategies (no treatment, random undersampling, random oversampling, SMOTE, and cost-sensitive class weighting) and two parameterization regimes (default and tuned hyperparameters). Not all combinations are applicable: cost-sensitive weighting is only available for algorithms that natively support asymmetric class penalties, which accounts for the asymmetry in the number of configurations per family. All experimental parameters are externalized in a YAML configuration file, and a global random seed of 123 is applied uniformly to all stochastic components (model initialization, resampling operations, and cross-validation splitting) to guarantee deterministic reproducibility.

Each configuration is executed through a unified sequential training pipeline. Where resampling is required, the corresponding sampler (random undersampling, random oversampling, or SMOTE, each initialized with the global seed) is chained immediately before the estimator and applied exclusively within the training partition of each fold, ensuring that synthetic data generation or subsampling is never exposed to validation data. For cost-sensitive configurations, asymmetric class penalties are set directly on the estimator. Feature scaling via Z-score normalization is applied upstream for distance- and margin-sensitive algorithms (Logistic Regression, KNN, SVM, and MLP); tree-based and ensemble methods (Decision Tree, Random Forest, and Gradient Boosting) consume raw features. For Tuned variants, hyperparameter selection is performed via exhaustive grid search nested within the same temporal cross-validation scheme as the outer evaluation loop, with AUC-ROC as the scoring metric, ensuring that no validation-fold information influences the hyperparameter search.

The cross-validation protocol follows a temporal splitting strategy (TimeSeriesSplit) with $k = 10$ folds that strictly preserves the chronological ordering of events: each fold's training partition contains exclusively observations that precede the validation fold in time, preventing any leakage of future information into the learning process. Within each fold, the pipeline is independently cloned and fitted, and AUC, sensitivity, and specificity are recorded. Upon completion of all folds, the best-performing pipeline is refitted on the full training partition to produce the final deployable estimator. Each trained configuration is subsequently persisted as a serialized artifact storing both the fitted model object and the cross-validation metric vectors, enabling post hoc threshold optimization on a strictly isolated test set without retraining. The complete configuration details for all 66 variants are presented in Table A1.

Table A1. Complete model configuration grid. For each of the evaluated configurations, the table specifies the algorithm family, the imbalance treatment strategy applied within each cross-validation fold, whether Z-score feature scaling was applied, and the hyperparameter search space explored for Tuned variants. Configurations with default parameters use scikit-learn factory settings.

Model	Imbalance Treatment	Scaled	Hyperparameter Search Space
LR—LogisticRegression			
LR_Base	None (default parameters)	Yes	Default parameters
LR_Weighted	Cost-sensitive (balanced weights)	Yes	Default parameters
LR_Under	Random undersampling	Yes	Default parameters
LR_Over	Random oversampling	Yes	Default parameters
LR_SMOTE	SMOTE	Yes	Default parameters
LR_Tuned_Base	None	Yes	C: {0.01, 0.1, 1, 10}; penalty: {l1, l2}
LR_Tuned_Weighted	Cost-sensitive (balanced weights)	Yes	C: {0.01, 0.1, 1, 10}; penalty: {l1, l2}
LR_Tuned_Under	Random undersampling	Yes	C: {0.01, 0.1, 1, 10}; penalty: {l1, l2}
LR_Tuned_Over	Random oversampling	Yes	C: {0.01, 0.1, 1, 10}; penalty: {l1, l2}
LR_Tuned_SMOTE	SMOTE	Yes	C: {0.01, 0.1, 1, 10}; penalty: {l1, l2}
DT—DecisionTreeClassifier			
DT_Base	None (default parameters)	No	Default parameters
DT_Weighted	Cost-sensitive (balanced weights)	No	Default parameters
DT_Under	Random undersampling	No	Default parameters
DT_Over	Random oversampling	No	Default parameters
DT_SMOTE	SMOTE	No	Default parameters
DT_Tuned_Base	None	No	criterion: {gini, entropy}; max_depth: {5, 10, 15, None}; min_samples_split: {2, 10, 20}

Table A1. *Cont.*

Model	Imbalance Treatment	Scaled	Hyperparameter Search Space
DT_Tuned_Weighted	Cost-sensitive (balanced weights)	No	criterion: {gini, entropy}; max_depth: {5, 10, 15, None}; min_samples_split: {2, 10, 20}
DT_Tuned_Under	Random undersampling	No	criterion: {gini, entropy}; max_depth: {5, 10, 15, None}; min_samples_split: {2, 10, 20}
DT_Tuned_Over	Random oversampling	No	criterion: {gini, entropy}; max_depth: {5, 10, 15, None}; min_samples_split: {2, 10, 20}
DT_Tuned_SMOTE	SMOTE	No	criterion: {gini, entropy}; max_depth: {5, 10, 15, None}; min_samples_split: {2, 10, 20}
KNN—KNeighborsClassifier			
KNN_Base	None (default parameters)	Yes	Default parameters
KNN_Under	Random undersampling	Yes	Default parameters
KNN_Over	Random oversampling	Yes	Default parameters
KNN_SMOTE	SMOTE	Yes	Default parameters
KNN_Tuned_Base	None	Yes	n_neighbors: {3, 5, 7, 9}; weights: {uniform, distance}
KNN_Tuned_Under	Random undersampling	Yes	n_neighbors: {3, 5, 7, 9}; weights: {uniform, distance}
KNN_Tuned_Over	Random oversampling	Yes	n_neighbors: {3, 5, 7, 9}; weights: {uniform, distance}
KNN_Tuned_SMOTE	SMOTE	Yes	n_neighbors: {3, 5, 7, 9}; weights: {uniform, distance}
SVM—SVC			
SVM_Base	None (default parameters)	Yes	Default parameters
SVM_Weighted	Cost-sensitive (balanced weights)	Yes	Default parameters
SVM_Under	Random undersampling	Yes	Default parameters
SVM_Over	Random oversampling	Yes	Default parameters
SVM_SMOTE	SMOTE	Yes	Default parameters
SVM_Tuned_Base	None	Yes	C: {0.1, 1, 10}; kernel: {linear, rbf}
SVM_Tuned_Weighted	Cost-sensitive (balanced weights)	Yes	C: {0.1, 1, 10}; kernel: {linear, rbf}
SVM_Tuned_Under	Random undersampling	Yes	C: {0.1, 1, 10}; kernel: {linear, rbf}
SVM_Tuned_Over	Random oversampling	Yes	C: {0.1, 1, 10}; kernel: {linear, rbf}
SVM_Tuned_SMOTE	SMOTE	Yes	C: {0.1, 1, 10}; kernel: {linear, rbf}
RF—RandomForestClassifier			
RF_Base	None (default parameters)	No	Default parameters
RF_Weighted	Cost-sensitive (balanced weights)	No	Default parameters
RF_Under	Random undersampling	No	Default parameters
RF_Over	Random oversampling	No	Default parameters
RF_SMOTE	SMOTE	No	Default parameters
RF_Tuned_Base	None	No	n_estimators: {50, 100, 150}; max_depth: {10, 20, None}
RF_Tuned_Weighted	Cost-sensitive (balanced weights)	No	n_estimators: {50, 100, 150}; max_depth: {10, 20, None}
RF_Tuned_Under	Random undersampling	No	n_estimators: {50, 100, 150}; max_depth: {10, 20, None}
RF_Tuned_Over	Random oversampling	No	n_estimators: {50, 100, 150}; max_depth: {10, 20, None}
RF_Tuned_SMOTE	SMOTE	No	n_estimators: {50, 100, 150}; max_depth: {10, 20, None}

Table A1. *Cont.*

Model	Imbalance Treatment	Scaled	Hyperparameter Search Space
GB—HistGradientBoostingClassifier			
GB_Base	None (default parameters)	No	Default parameters
GB_Weighted	Cost-sensitive (balanced weights)	No	Default parameters
GB_Under	Random undersampling	No	Default parameters
GB_Over	Random oversampling	No	Default parameters
GB_SMOTE	SMOTE	No	Default parameters
GB_Tuned_Base	None	No	learning_rate: {0.01, 0.05, 0.1}; max_iter: {50, 100, 200}; max_depth: {3, 5, 10}
GB_Tuned_Weighted	Cost-sensitive (balanced weights)	No	learning_rate: {0.01, 0.05, 0.1}; max_iter: {50, 100, 200}; max_depth: {3, 5, 10}
GB_Tuned_Under	Random undersampling	No	learning_rate: {0.01, 0.05, 0.1}; max_iter: {50, 100, 200}; max_depth: {3, 5, 10}
GB_Tuned_Over	Random oversampling	No	learning_rate: {0.01, 0.05, 0.1}; max_iter: {50, 100, 200}; max_depth: {3, 5, 10}
GB_Tuned_SMOTE	SMOTE	No	learning_rate: {0.01, 0.05, 0.1}; max_iter: {50, 100, 200}; max_depth: {3, 5, 10}
MLP—MLPClassifier			
MLP_Base	None (default parameters)	Yes	Default parameters
MLP_Under	Random undersampling	Yes	Default parameters
MLP_Over	Random oversampling	Yes	Default parameters
MLP_SMOTE	SMOTE	Yes	Default parameters
MLP_Tuned_Base	None	Yes	hidden_layer_sizes: {[50], [100], [50,25]}; activation: {relu, tanh}; alpha: {0.0001, 0.001}
MLP_Tuned_Under	Random undersampling	Yes	hidden_layer_sizes: {[50], [100], [50,25]}; activation: {relu, tanh}; alpha: {0.0001, 0.001}
MLP_Tuned_Over	Random oversampling	Yes	hidden_layer_sizes: {[50], [100], [50,25]}; activation: {relu, tanh}; alpha: {0.0001, 0.001}
MLP_Tuned_SMOTE	SMOTE	Yes	hidden_layer_sizes: {[50], [100], [50,25]}; activation: {relu, tanh}; alpha: {0.0001, 0.001}

References

1. Lu, Y.; Zhang, W.; Yang, L.; Dai, Y.; Wu, H.; Wu, Z.; Tian, H.; Li, Y.; Chen, J.; Liu, C.; et al. A survey of anomaly detection in HPC systems using machine learning. *CCF Trans. High Perform. Comput.* **2026**, *8*, 352–376. [CrossRef]
2. Guan, Y.; Liu, Y.; Zhou, K.; Li, Q.; Wang, T.; Li, H. A disk failure prediction model for multiple issues. *Front. Inf. Technol. Electron. Eng.* **2023**, *24*, 964–979. [CrossRef]
3. Lima, A.L.C.D.; Aranha, V.M.; Carvalho, C.J.L.; Nascimento, E.G.S. Smart predictive maintenance for high-performance computing systems: A literature review. *J. Supercomput.* **2021**, *77*, 13494–13513. [CrossRef]
4. Jian, L.; Ren, X.; Zhang, Z.; Shi, S.; Huang, Y.; Zhang, C. Research on Maintenance Fault Diagnosis System for E-class High-Performance Computer. *Jisuanji Gongcheng/Comput. Eng.* **2022**, *48*, 24–37. [CrossRef]
5. Jiang, T.; Huang, P.; Zhou, K. Cost-efficiency disk failure prediction via threshold-moving. *Concurr. Comput. Pract. Exp.* **2020**, *32*, e5669. [CrossRef]
6. Mohammed, B.; Awan, I.; Ugail, H.; Younas, M. Failure prediction using machine learning in a virtualised HPC system and application. *Clust. Comput.* **2019**, *22*, 471–485. [CrossRef]

7. Ahmed, J.; Green, R.C., II. Predicting severely imbalanced data disk drive failures with machine learning models. *Mach. Learn. Appl.* **2022**, *9*, 100361. [CrossRef]
8. Amram, M.; Dunn, J.; Toledano, J.J.; Zhuo, Y.D. Interpretable predictive maintenance for hard drives. *Mach. Learn. Appl.* **2021**, *5*, 100042. [CrossRef]
9. Anantharaman, P.; Qiao, M.; Jadav, D. Large Scale Predictive Analytics for Hard Disk Remaining Useful Life Estimation. In Proceedings of the 2018 IEEE International Congress on Big Data (BigData Congress), Seattle, WA, USA, 2–7 July 2018; pp. 251–254. [CrossRef]
10. Gebreyesus, Y.; Dalton, D.; De Chiara, D.; Chinnici, M.; Chinnici, A. AI for Automating Data Center Operations: Model Explainability in the Data Centre Context Using Shapley Additive Explanations (SHAP). *Electronics* **2024**, *13*, 1628. [CrossRef]
11. Pinciroli, R.; Yang, L.; Alter, J.; Smirni, E. Lifespan and Failures of SSDs and HDDs: Similarities, Differences, and Prediction Models. *IEEE Trans. Dependable Secur. Comput.* **2023**, *20*, 256–272. [CrossRef]
12. Monje, L.; Carrasco, R.A.; Sánchez-Montañés, M. Machine learning XAI for early loan default prediction. *Comput. Econ.* **2025**, *67*, 4033–4062. [CrossRef]
13. Ransom, G.W.; George, A. *Disk Failure Dataset from the Campaign Storage System*; Oak Ridge National Laboratory (ORNL): Oak Ridge, TN, USA; Oak Ridge Leadership Computing Facility (OLCF): Oak Ridge, TN, USA; Los Alamos National Laboratory (LANL): Los Alamos, NM, USA, 2024. [CrossRef]
14. Liu, Y.; Yang, K.; Sun, L.; Wang, J.; Smirnov, A.; Xiong, C. Modeling and prediction of the ionosphere with deep learning: A review. *Intell. Robot.* **2026**, *6*, 68–96. [CrossRef]
15. Lu, Q.; Li, M. Fault Prediction Method Towards Rolling Element Bearing Based on Digital Twin and Deep Transfer Learning. *Appl. Sci.* **2025**, *15*, 12509. [CrossRef]
16. Li, W.; Zhou, H.; Radhakrishnan, S.; Kamarthi, S. Explainable time series features for hard disk drive failure prediction. *Eng. Appl. Artif. Intell.* **2025**, *152*, 110674. [CrossRef]
17. Gupta, S.; Asirvatharaj, G. A Novel Two-Stage BiLSTM-XGBoost Hybrid Model for Drive Failure Prediction in Data Center. In Proceedings of the INDISCON 2025—IEEE 6th India Council International Subsections Conference, Rourkela, India, 21–23 August 2025. [CrossRef]
18. Ahmed, J.; Green, R.C., II. Cost aware LSTM model for predicting hard disk drive failures based on extremely imbalanced S.M.A.R.T. sensors data. *Eng. Appl. Artif. Intell.* **2024**, *127*, 107339. [CrossRef]
19. Cornali, R.; Ronchieri, E.; Zurlo, G.; Scarponi, L.B.; Cesini, D.; dell'Agnello, L. Graph neural network solutions for anomaly detection in time series. In Proceedings of the Annual Hawaii International Conference on System Sciences, Honolulu, HI, USA, 15 June–17 August 2025; pp. 1002–1011. [CrossRef]
20. Ji, S.; Zuo, X.; Huang, H. Disk failure prediction via Lightgbm model. In *Proceedings of SPIE—The International Society for Optical Engineering*; SPIE: Washington, DC, USA, 2022; Volume 12331, p. 123313G. [CrossRef]
21. Zhang, X.; Shant, K.; Tan, Z.; Feng, D. CSLE: A Cost-sensitive Learning Engine for Disk Failure Prediction in Large Data Centers. In Proceedings of the 2022 Design, Automation and Test in Europe Conference and Exhibition, DATE 2022, Antwerp, Belgium, 14–23 March 2022; pp. 478–483. [CrossRef]
22. Chakraborttii, C.; Litz, H. Improving the accuracy, adaptability, and interpretability of SSD failure prediction models. In Proceedings of the SoCC 2020—2020 ACM Symposium on Cloud Computing, Santa Cruz, CA, USA, 19–21 October 2020; pp. 120–133. [CrossRef]
23. Fernández-Avilés, G.; Montero, J.M. *Fundamentals of Data Science with R*; McGraw-Hill Education: Madrid, Spain, 2021; pp. 1–320.
24. Monje, L.; Carrasco, R.A.; Rosado, C.; Sánchez-Montañés, M. Deep learning XAI for bus passenger forecasting: A use case in Spain. *Mathematics* **2022**, *10*, 1428. [CrossRef]
25. Wilkinson, C.; Yawney, J.; Gadsden, S.A. Explaining explainability: A comprehensive survey on explainable artificial intelligence and relevant industry applications. *Intell. Syst. Appl.* **2026**, *30*, 200647. [CrossRef]
26. Altalhan, M.; Algarni, A.; Alouane, M.T.H. Imbalanced data problem in machine learning: A review. *IEEE Access* **2025**, *13*, 13686–13699. [CrossRef]
27. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
28. Lemaître, G.; Nogueira, F.; Aridas, C.K. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *J. Mach. Learn. Res.* **2017**, *18*, 1–5. Available online: http://jmlr.org/papers/v18/16-365.html (accessed on 4 April 2026).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.