machines

Article

Towards the Best Solution for Complex System Reliability: Can Statistics Outperform Machine Learning?

María Luz Gámiz [1], Fernando Navas-Gómez [1], Rafael Adolfo Nozal Cañadas [2] and Rocío Raya-Miranda [1,*]

[1] Department of Statistics and Operational Research, University of Granada, 18071 Granada, Spain; mgamiz@ugr.es (M.L.G.); fjnavasg@ugr.es (F.N.-G.)
[2] Department of Computer Science, UiT The Arctic University of Norway, 9037 Tromsø, Norway; rca015@uit.no
* Correspondence: rraya@ugr.es

Abstract: Studying the reliability of complex systems using machine learning techniques involves facing a series of technical and practical challenges, ranging from the intrinsic nature of the system and data to the difficulties in modeling and effectively deploying models in real-world scenarios. This study compares the effectiveness of classical statistical techniques and machine learning methods for improving complex system analysis in reliability assessments. Our goal is to show that in many practical applications, traditional statistical algorithms frequently produce more accurate and interpretable results compared with black-box machine learning methods. The evaluation is conducted using both real-world data and simulated scenarios. We report the results obtained from statistical modeling algorithms, as well as from machine learning methods including neural networks, K-nearest neighbors, and random forests.

Keywords: logistic regression; factorial analysis; isotonic smoothing; machine learning; supervised learning; unsupervised learning; ANN; KNN; RF

Citation: Gámiz, M.L.; Navas-Gómez, F.; Nozal Cañadas, R.A.; Raya-Miranda, R. Towards the Best Solution for Complex System Reliability: Can Statistics Outperform Machine Learning? *Machines* **2024**, *12*, 909. https://doi.org/10.3390/machines12120909

Academic Editor: Phuc Do

Received: 13 November 2024
Revised: 5 December 2024
Accepted: 9 December 2024
Published: 11 December 2024

1. Introduction

Reliability analysis of complex, multicomponent systems is crucial in engineering, manufacturing, and operations research. It involves understanding and quantifying the capacity of a system to perform its intended function over a specified period under given conditions. Addressing the reliability of such systems requires a multifaceted approach that integrates structural analysis, probabilistic modeling, and practical maintenance strategies. These methodologies help engineers design more reliable systems, predict potential failures, and develop effective maintenance plans to mitigate risks.

However, as the number of components increases, the lack of knowledge about the structure of the system leads to an estimation problem with an overwhelming number of features [1]. This results in exponential growth in dimensionality and sparse available data. As a result, high-dimensional dependability analysis continues to be extremely difficult because the majority of current techniques are plagued by the dimensionality curse [2].

The curse of dimensionality refers to the phenomenon where, in higher dimensions, a local neighborhood loses its traditional "local" properties. A neighborhood containing a fixed percentage of data points becomes disproportionately large, diverging from the intuitive concept of locality [3]. If a local neighborhood contains 10 data points along each axis, then there are 10^d data points in the corresponding d-dimensional neighborhood. As a consequence, much larger datasets are needed even when d is moderate, and such large datasets are often not available in practical situations. This makes smoothing techniques less effective in high-dimensional settings. In nonparametric regression, the curse of dimensionality significantly impacts the convergence rate of prediction error, which is proportional to $n^{-2/(2+d)}$. This illustrates that the required sample size n grows exponentially with d to maintain a low prediction error [4].

To address this challenge, various strategies have been proposed, all of which incorporate a dimensionality reduction step [3]. In this context, we propose an initial step for nonparametric fitting that reduces the dimensionality of the problem using a factor analysis model. This approach aims to mitigate the effects of high dimensionality and improve the practical utility of smoothing techniques.

Machine learning (ML), a form of applied statistics, focuses on using computational power to estimate complex functions, unlike traditional statistics, which prioritize confidence intervals and rigorous proofs of uncertainty [5]. ML employs algorithms to detect patterns in data, predicting and assessing performance and failure risks in systems. By doing so, it enhances design, maintenance, and safety. ML is defined as a collection of techniques used to identify patterns in data to predict future outcomes or support decision-making under uncertainty [6]. Over recent decades, ML has revolutionized fields such as control systems, autonomous systems, and computer vision. Similarly, reliability engineering and safety analysis are expected to follow this trend [7].

ML offers several advantages over traditional methods, including advanced predictive capabilities, the ability to handle large, high-dimensional datasets, real-time monitoring, and automation, all of which contribute to improved system reliability. Common applications of ML in reliability engineering include tasks such as estimating remaining useful life, detecting anomalies and faults, monitoring system health, planning maintenance, and assessing degradation. Studies like [7] highlight the potential of ML to transform reliability and safety analysis. However, challenges such as data quality and model interpretability must be addressed for ML to fully realize its potential (see [1] and references therein). Interdisciplinary collaboration and hybrid approaches also show promise [1,8,9]. For instance, ref. [8] proposes a state monitoring algorithm combining a convolutional neural network (CNN) with a random forest (RF) for data missing scenarios. The CNN algorithm is designed to extract the distributed fault information from the available signals and acquire the state features of the system, then the random forest algorithm processes the state features and judges the system state, while ref. [9] combines reliability analysis tools with ML to identify critical maintenance components and failure causes. Similarly, ref. [1] integrates supervised and unsupervised learning techniques to analyze system status.

When applying ML, the question of which model performs best often arises. In 1997, David Wolpert and William MacReady addressed this issue with the "No Free Lunch" (NFL) theorem [10], which demonstrated that without assumptions about the data, no model is universally superior. This implies that no model can consistently outperform others in all scenarios. It underscores the importance of inductive biases, suggesting that specialized algorithms are required for different types of data, as real-world problems often exhibit low complexity [11]. Recent explorations extend the NFL theorem to quantum learning protocols, revealing that quantum algorithms can achieve better sample complexity under certain conditions, thus showcasing the theorem's relevance in advanced learning contexts [12]. Conversely, while the NFL theorem suggests limitations, it also opens avenues for developing metalearning strategies that leverage correlations among algorithms, potentially leading to improved performance in specific contexts [13].

The use of ML for reliability analysis in large-scale, complex systems is a key application within Industry 4.0. In this context, the interconnection of sensors and smart devices generates large volumes of data, which can be analyzed to enhance system efficiency and safety. In systems with thousands of sensors, traditional reliability models based on statistical distributions become inefficient. Machine learning techniques, such as deep learning, are better suited for handling high-dimensional data. Algorithms like Random Forest and Bayesian classification models can extract valuable information from the vast datasets produced by sensors [14]. Industry 4.0 now includes diverse data sources, such as temperature, vibration, and pressure sensors, and the application of ML algorithms enables the integration and simultaneous analysis of these variables, detecting correlations that might not be apparent with traditional techniques [15].

Reliability analysis is fundamental in the design and maintenance of complex systems. One key question is that, traditionally, systems have been analyzed under the assumption that their components are independent, but this assumption becomes less valid as systems grow in complexity. In reality, system components are often interdependent; they are designed to work together to ensure the proper operation of the system. Addressing these interdependencies is crucial for improving the accuracy of reliability predictions. A complex system is characterized by the nonlinear interaction of its components, where the dynamics of a system cannot be understood simply as the sum of its parts. These systems exhibit emergent properties and adaptability and may have hierarchical structures. To model such complex interactions, advanced methods like ML can be employed, which allow for the consideration of nonlinear relationships between components. The analysis of complex systems in the context of reliability needs to address these interdependencies to improve the accuracy of reliability predictions. Traditional methods that assume independent components are insufficient, and this is where machine learning (ML) methods, such as neural networks, become valuable. These methods can model the complex interdependencies between the components of the system, providing more accurate predictions. This capability makes them ideal for reliability prediction, especially in systems with numerous components that interact in ways that traditional methods may not easily capture. Until 2006, we lacked the knowledge to train neural networks to outperform traditional methods, aside from a few specific cases [16]. In a neural network, we do not instruct the computer on how to solve our problem; rather, it learns from observational data, discovering its own solutions. The breakthrough in 2006 came with the advent of learning techniques for deep neural networks [17].

Nevertheless, ML methods have to be adapted according to the area of application, and it is necessary to incorporate the knowledge of experts to improve the accuracy and reliability of the predictions [18]. Below are some of the ML methods that can be applied to predict the reliability of a complex system with multiple components and possible correlations between them.

This paper conducts a comprehensive comparative analysis between classical statistical methods and widely used ML techniques to predict system reliability in complex systems characterized by high dimensionality. In this context, high dimensionality refers to systems with a large number of components, often grouped into independent blocks of correlated units due to physical, functional, or operational dependencies, a common scenario in engineering systems. These systems pose significant challenges to traditional reliability analysis methods. The contributions of this paper are summarized as follows:

1. Efficient method for reliability analysis: we present an easy-to-run and computationally efficient method specifically designed for reliability analysis in high-dimensional complex systems.
2. Classical statistical methods for effective reliability: the proposed method, based on classical statistical approaches, ensures interpretability, scalability, and robustness, even in the presence of complex correlation structures between system components.
3. Performance comparison with machine learning: we compare the performance of the FA-LR-IS algorithm, designed by the authors, with commonly used ML methods, including artificial neural networks (ANNs), k-nearest neighbors (KNN), and random forests.
4. Practical relevance via extensive evaluation: The evaluation includes an extensive simulation study and two real-world datasets. This ensures the practical relevance and provides insights into the strengths and limitations of each methodology.

The rest of this paper is organized as follows: Section 2 first defines the problem statement. Then it introduces the FA-LR-IS algorithm for estimating reliability in complex, high-dimensional systems and discusses machine learning solutions to this problem. Section 3 presents the traditional division of the data into the training and test sets, together with the metrics for evaluating the models. In addition, the particular evaluation procedure on the test sample for the FA-LR-IS algorithm is included. Section 4 presents numerical results from a simulation study and two real applications for the study of reliability in systems composed of sensors. Section 5 provides a brief discussion on the topic, and Section 6 ends with the conclusions.

2. Approaches to Reliability Analysis

2.1. Problem Statement

Let us consider a system that can be found in one of two states: operative (1) and failure (0). We denote Y the random variable representing the system state. The system is composed of p units, which interact and work together to assure the system's functioning. For $j = 1, \ldots, p$, unit j can be found in a state that is represented by a random variable X_j taking values in $[0, 1]$. The states of the p units are collected in a random vector $(X_1, X_2, \ldots, X_p)$ that we call the state vector. Depending on the underlying logic of the system, some configurations of the state vector lead to the good performance of the system, while for others the system is out of work. Then, our objective is to determine a function that links the states of the units to the state of the system.

The reliability of a system is the probability that the system will be operational for a particular configuration of the system. We consider the reliability of the system as a function of the state vector:

$$R(x_1, x_2, \ldots, x_p) = P[Y = 1 | X_1 = x_1, X_2 = x_2, \ldots, X_p = x_p], \tag{1}$$

That is, we assume that given $X_1 = x_1, X_2 = x_2, \ldots, X_p = x_p$, Y follows a binomial distribution with event probability $R(x_1, x_2, \ldots, x_p)$, with R an unknown function for which we do not assume any particular functional form. The only restriction is that R meets the coherence conditions of the system, i.e., $R(0, \ldots, 0) = 0$, $R(1, \ldots, 1) = 1$, and R is monotone in each argument.

Our main objective is to predict whether the system is operational given the state of the components. This problem will be solved using different algorithms in the following subsections. Some of the algorithms will only provide a classification of the system in one of two categories, functioning or failure, but in other cases an estimation of the probability that the system is working in terms of the components states is also provided. To learn our algorithms, we need to observe a sample of systems. Let $\{\mathbf{X}, \mathbf{Y}\}$ be the observed data, where $\mathbf{X}$ is a matrix of dimension $n \times p$. Each row of the matrix is a configuration of the component states of system i, with $i = 1, 2, \ldots, n$. $\mathbf{Y}$ is the n-dimensional vector of all systems of the sample. The objective is to predict whether the system is operational given the state of the components. In the following, we describe in detail the FA-LR-IS algorithm and the basic elements of the ML methods that we will apply in our numerical analysis in Section 4.

2.2. The FA-LR-IS Algorithm

As we have already mentioned, one of the primary goals in reliability analysis is to mathematically represent the logic underlying a system. Assessing system performance, even for simple structures, can be challenging, making it essential to develop effective methods for modeling the relationship between the state of the system and its components. Unlike some of the ML algorithms, the FA-LR-IS algorithm [1] not only classifies the system as operative or failed; it provides other useful information. Specifically, the main objective of the algorithm is to estimate the reliability function, that is, the probability that the system is functioning in terms of its unit states as in (1). Finally, a ranking of components can be established according to the importance of every component in the system state. Understanding the importance of a unit as the impact that a one-unit change in the component state has on the system behavior. In [19], we emphasize the importance of using nonparametric regression, which produces estimators with good asymptotic properties, although these estimators are not necessarily monotonic. In high-dimensional scenarios, a methodology is proposed that combines factor analysis, logistic regression, and isotonization. The combination of factor analysis and logistic regression was chosen for the following reasons: First, we consider factor analysis for dimensionality reduction, because in complex systems with a large number of components, the total number of variables in the regression problem we aim to solve is very high, many of them highly correlated [20–22]. In this context, factor analysis reduces dimensionality by extracting latent factors that summarize the original data. This prevents issues like multicollinearity and overfitting while maintaining the key relationships in the data. Second, in our practical application we are interested in predicting the probability that the system is either operative or in failure given a particular configuration of component states. As known, logistic regression is ideal for predicting binary outcomes. It is interpretable, efficient, and well suited for modeling the nonlinear relationships that are underlying in our problem. Finally, to ensure practical relevance and interpretability of results, the model is expressed in terms of the original variable by applying an inverse transformation, which is possible given the mathematical formulation of the combination of these two models.

Here, we outline the algorithm described in [1] for constructing a statistical model that predicts system reliability in complex systems with many interdependent components.

Let $\{\mathbf{X}, \mathbf{Y}\}$ represent the observed data, where $\mathbf{X}$ is an $n \times p$ matrix. Each row of $\mathbf{X}$, for $i = 1, \ldots, n$, corresponds to a configuration of component states in a system with p dimensions. The state of each component is modeled by a random variable that takes values within the interval $[0, 1]$. The components are not assumed to be independent. The vector $\mathbf{Y}$ has dimension n, where for each $i = 1, 2, \ldots, n$, $Y_i = 1$ indicates that the system is operational, and $Y_i = 0$ otherwise.

2.2.1. Data Preprocessing

Center the matrix: $\mathbf{X}$: Define $U_j = (X_j - \mu_j)/\sigma_j$ for each $j = 1, 2, \ldots, p$. Let $\mathbf{\Sigma}_0$ be a diagonal matrix, where the jth element is σ_j. Define $\mathbf{M} = \mathbf{1}_p(\mu_1, \mu_2, \ldots, \mu_p)$, where $\mathbf{1}_p$ a unit column vector of size p. We denote $\mathbf{U} = (\mathbf{X} - \mathbf{M})\mathbf{\Sigma}_0^{-1}$, resulting in a matrix $\mathbf{U}$ with dimensions $n \times p$.

2.2.2. FA-RL-IS Algorithm

1. Apply factorial analysis (FA): we apply a FA algorithm to the scaled dataset $\mathbf{U}$ and transform the data into a reduced set denoted as $\{\mathbf{Z}, \mathbf{Y}\}$ with dimensions $n \times p_0$, where $p_0 < p$. Here, $\mathbf{Z} = \mathbf{U}\mathbf{\Gamma}$, and $\mathbf{\Gamma}$ is the FA coefficient-matrix of dimension $p \times p_0$.
2. Fit a local-logistic model: a local-logistic model is fitted using the reduced dataset $\{\mathbf{Z}, \mathbf{Y}\}$ with the leave-one-out-cross-validation (LOOCV) criterion for bandwidth selection.
   (a) Let h be the bandwidth, varying over the grid $\{h_1, h_2, \ldots, h_m\}$.
   (b) Construct the local-logistic model based on $\{\mathbf{Z}, \mathbf{Y}\}$ and estimate $\widetilde{R}_h(\mathbf{z}_i)$ for all $i = 1, 2, \ldots, n$.
   (c) Consider the leave-one-out (*loo*) dataset $\{\mathbf{Z}, \mathbf{Y}\}^{(-i)}$ for each $i = 1, 2, \ldots, n$, which exclude the ith observation. Define $\widetilde{R}_h^{(-i)}(\mathbf{z})$ as the fitted model using the *loo* dataset and estimate $\widetilde{R}_h^{(-i)}(\mathbf{z}_i)$ for all $i = 1, 2, \ldots, n$.
   (d) The cross-validation score is defined as

$$Q(h) = \sum_{i=1}^{n} \widetilde{R}_h(\mathbf{z}_i)^2 - 2\sum_{i=1}^{n} Y_i \widetilde{R}_h^{(-i)}(\mathbf{z}).$$

   (e) Define $h_{CV} = \arg\min_h \; Q(h)$.
3. Let $\mathbf{x}_0$ represent a specific configuration of component states. We compute $\mathbf{z}_0 = \big(\mathbf{x}_0 - (\mu_1, \mu_2, \ldots, \mu_p)\big)\mathbf{\Sigma}_0^{-1}\mathbf{\Gamma}$ and construct a local-logistic model. For any $\mathbf{z}$ such that $\|\mathbf{z} - \mathbf{z}_0\| < h_{CV}$, the fitted model is given by

$$\widetilde{R}_{CV}(\mathbf{z}) = \frac{e^{(1,(\mathbf{z}-\mathbf{z}_0)^t)\widehat{\mathbf{b}}}}{1 + e^{(1,(\mathbf{z}-\mathbf{z}_0)^t)\widehat{\mathbf{b}}}},$$

   with $\widehat{\mathbf{b}} = (\widehat{b}_0, \widehat{b}_1, \ldots, \widehat{b}_{p_0})^t$ vector of coefficients obtained using h_{CV}.

4. Let $\mathbf{x}_0$ be defined such that $\|(\mathbf{x}-\mathbf{x}_0)\mathbf{\Sigma}_0^{-1}\mathbf{\Gamma}\| < h_{CV}$. Under this condition, we estimate the local-logistic model within the state space of components using

$$\widehat{R}^*_{CV}(\mathbf{x}) = \frac{e^{(1,(\mathbf{x}-\mathbf{x}_0)^t)\widehat{\beta}}}{1+e^{(1,(\mathbf{x}-\mathbf{x}_0)^t)\widehat{\beta}}},$$

where the vector of estimated coefficients $\widehat{\beta}$ is given by

$$\begin{array}{rcl} \widehat{\beta}_0 & = & \widehat{b}_0, \\ \widehat{\beta}_j & = & \sigma_j^{-1}\mathbf{\Gamma}_{j\cdot}\widehat{\mathbf{b}}_{-0}, \qquad j = 1,2,\dots,p; \end{array}$$

where $\mathbf{\Gamma}_{j\cdot}$ denotes the jth row of matrix $\mathbf{\Gamma}$, and $\widehat{\mathbf{b}}_{-0} = (b_1,\dots,b_{p_0})^t$.

5. Define $\widehat{R}^*_{CV}(\mathbf{x}_i) = \frac{e^{b_0}}{1+e^{b_0}}$, for $i = 1,2,\dots,n$. Apply the isotonization algorithm outlined in [19] to adjust these estimated values, thereby obtaining the estimated reliability for the ith configuration of states of components, denoted as $\widehat{R}_{CV}(\mathbf{x}_i)$, $i = 1,2,\dots,n$. Let us denote $\mathbf{r}^*$ the vector with components $R^*_i = \widehat{R}_{CV}(\mathbf{x}_i)$, $i = 1,\dots,n$. The problem is to find $\mathbf{r} = (R_1,\dots,R_n) \in \mathbb{R}^n$ minimizing $\|\mathbf{r}^* - \mathbf{r}\|$ subject to $\mathbf{Ar} \geq 0$, for matrix $\mathbf{A}$ of dimension $(n-1)\times n$ and with elements $A(i,i) = -1, \quad A(i,i+1) = 1$, and $A(i,j) = 0$ for $j \neq i, i+1$. The set $C = \{\mathbf{r} \in \mathbb{R}^n : Az \geq 0\}$ is a polyhedral convex cone in $\mathbb{R}^n$. To solve this problem, we will use the *hinge* algorithm presented in [23].
6. Analyze the labeled data using the ROC curve to calculate the (AUC) as described below in Section 3.2 and evaluate the model.

Broadly speaking, the algorithm goes in this sense: The goal of the FA-LR-IS algorithm is to estimate the probability of the system functioning based on its components performance levels. However, with a large number of inputs, overfitting and higher prediction errors may arise. To address this, once the data have been normalized, the algorithm first reduces dimensionality using factor analysis (FA), which groups correlated variables into factors that share common variance.

Next, a local-logistic model is built in the latent space rather than using the original component states as inputs. The local regression model is constructed on the scores matrix generated by the FA algorithm. Nonparametric regression ensures estimators with desirable properties like consistency and normality, but these estimators are not inherently monotonic. In a coherent system, however, the response variable of the regression model (the system state) must be monotonic relative to the original variables (component states). Since the features used in the local-logistic model lack clear physical interpretation, we cannot assume monotonicity. Therefore, we propose an isotonization step after back-transformation to ensure the model is expressed in terms of the original variables.

Finally, the estimated probabilities generated by the logistic regression model are translated into classes or categories using the classification obtained from the ROC curve.

2.3. Artificial Neural Networks

Artificial neural networks (ANNs) are ML algorithms that simulate the learning mechanisms of biological organisms [17], having demonstrated success in a variety of areas, such as natural language processing, speech recognition, and image recognition. The application of neural networks in reliability estimation is based on their ability to model complex relationships between input data (e.g., historical failure data or operating conditions) and output results (such as failure probability, remaining useful life, etc.). Neural networks significantly improve the ability to anticipate failures, optimize maintenance, and ensure reliability in complex systems, especially in industries where the cost of a failure can be high [24]. An example of this can be seen in [25]; the authors propose the use of neural networks to predict the useful life of machines and components, highlighting the predictive capacity of these models and illustrating the methodology with two studies: repair of damaged units subjected to fatigue and a pump system in an industrial plant.

ANNs are made up of units called neurons, which are interconnected in a structure consisting of at least two layers: an input layer and an output layer. In the case of having a hidden layer, it must have at least one hidden layer. The input layer receives the data (relevant features such as operating conditions, runtime, sensor variables, etc.), while the hidden layers process the information in an intermediate manner. Each neuron in these layers performs a mathematical transformation based on the learned weights and biases. Finally, the output layer produces the reliability estimate, such as the probability of failure or the remaining lifetime. Each neuron takes a linear combination of the inputs and then applies a nonlinear activation function. This process can be described as follows:

$$z_j^{(l)} = \sum_{i=1}^{n} w_{ij}^{(l)} x_i^{(l-1)} + b_j^{(l)}$$

where the following is true:

- $z_j^{(l)}$ is the value of neuron j in layer l.
- $w_{ij}^{(l)}$ is the weight connecting neuron i in the previous layer $l-1$ to neuron j in the current layer l.
- $x_i^{(l-1)}$ is the output of neuron i in the previous layer.
- $b_j^{(l)}$ is the bias of neuron j in layer l.
- n is the number of neurons in the previous layer.

Then, an activation function g is applied to introduce nonlinearity, $a_j^{(l)} = g(z_j^{(l)})$; for more details, see [17,26]. Depending on the type of study, selecting the appropriate activation function allows the network to learn more complex patterns and perform more sophisticated tasks. For reliability classification problems, activation functions such as ReLU (Rectified Linear Unit) or the sigmoid are commonly used:

- ReLU: $g(z) = \max(0, z)$
- Sigmoid: $g(z) = \frac{1}{1+e^{-z}}$

The sigmoid function is especially popular in the output layer for binary classification problems, as it generates an output in the form of a probability [27]. This process is repeated in all the hidden layers. In the last layer (the output layer), the output value Y, which represents the probability of failure at a given time, is generated using an activation function such as the sigmoid in the case of classification problems.

The ANN requires a learning process to adjust the weights and biases of the connections, which incurs a computational cost. Algorithms such as gradient descent (GD), stochastic gradient descent (SGD), adaptive gradient descent (AdaGrad), root mean square propagation (RMSprop), and ADAM are commonly used to minimize the loss function [17,28]. The ADAM algorithm updates the parameters of the neural network according to the following expressions. For each iteration $r = 1, 2, \ldots, R$

$$w_{ij}^{(l),r+1} = w_{ij}^{(l),r} + \triangle w_{ij}^{(l),r+1} = w_{ij}^{(l),r} - \eta \frac{\partial L}{\partial w_{ij}^{(l)}} = w_{ij}^{(l),r} - \eta \frac{\widehat{m}^{(r)}}{\sqrt{\widehat{v}^{(r)}} + \epsilon},$$

$$b_j^{(l),r+1} = b_j^{(l),r} + \triangle b_j^{(l),r+1} = b_j^{(l),r} - \eta \frac{\partial L}{\partial b_j^{(l)}} = b_j^{(l),r} - \eta \frac{\widehat{m}^{(r)}}{\sqrt{\widehat{v}^{(r)}} + \epsilon},$$

where L is a loss function, η is the learning rate, and $\widehat{m}^{(r)}$ and $\widehat{v}^{(r)}$ are correction biases for the exponential moving average of the gradient and the exponential moving average of the squared gradient, respectively. The parameter ϵ is a smoothing term that avoids division by zero.

Loss functions are critical in the training and validation stages, as they minimize the difference between the system state predicted by the model and the actual state, allowing for the correct fit of the model parameters, i.e., weights and biases. For regression problems, the mean square error is typically used, while binary cross-entropy is an appropriate cost function for binary classification problems. The binary cross-entropy loss function is obtained by applying:

$$L = BCE = -\frac{1}{n}\sum_{i=1}^{n}\left[Y_i \cdot log(\hat{Y}_i) + (1 - Y_i) \cdot log(1 - \hat{Y}_i)\right]. \tag{2}$$

For our problem we consider a feedforward neural network with three layers that attempts classification. The input layer consists of p input neurons $(X_1, X_2, \ldots, X_p)$ with sample information, H neurons in the hidden layers, and one neuron in the output layer, Y. The model that generates the network is expressed as follows:

$$\hat{Y} = g_B\left(b_o + \sum_{h=1}^{H} w_{ho} g_A\left(b_k + \sum_{i=1}^{p} X_i w_{ih}\right)\right),$$

where w_{ih} are the connection weights of the input i with neuron h of the hidden layer, w_{ho} are the connection weights of the hidden layer with the output layer, b_o and b_k are the bias terms, and g_A y g_B are the activation functions in the hidden and output layers, respectively.

2.4. K-Nearest Neighbors

The K-nearest neighbors (KNN) algorithm is a supervised learning technique commonly used in classification and regression problems. In the context of reliability, KNN can be applied to make predictions related to the probability of failure or lifetime of a system based on historical data of similar failures. An example would be the prediction of the remaining lifetime of electronic devices under accelerated stress conditions, where machine learning models such as KNN are used to estimate the reliability of electronic components with high levels of accuracy [29]. In [30], a methodology is presented that combines active learning and the KNN algorithm to evaluate the reliability of engineering structures based on the assessment of the fracture probability of cracked structures.

Using KNN, a system is classified based on data from other similar systems (neighbors). If a majority of the nearby neighbors have failed under similar conditions, the algorithm predicts that the system is also at risk of failure. The aim is to assign an unclassified point $\mathbf{x}$, i.e., a given configuration of component states, to the class state of the system represented by a majority of its K-nearest neighbors. To do this, the distance between the point of interest $\mathbf{x}$ and each point $\mathbf{x}_i$ in the dataset, $d(\mathbf{x}, \mathbf{x}_i)$ is calculated. The K nearest neighbors are selected, that is, those points that minimize the distance $d(\mathbf{x}, \mathbf{x}_i)$. Finally, the majority class of the state of the system is determined among the nearest neighbors; the system state, $\hat{Y}$, for point $\mathbf{x}$ will be the one that appears most frequently among the K neighbors.

The choice of the parameter K has a significant influence on the performance of the model [31–33]. To choose the parameter K correctly, there are several alternatives, such as opting for an odd value of K, which would avoid possible ties in the proportions of membership in each class. The leave-one-out or K-fold cross-validation technique consists of dividing the dataset into K subsets, using $K - 1$ of these to train the model and the remaining subset to evaluate performance. This process is repeated K times, and the model performance is averaged for different values of K. Another option is to try different values of K, apply the method to sample points whose classification is known, and select the value of K that minimizes the classification error. Empirically, one can highlight the square root rule of the dataset size, which suggests that K should be approximately equal to the square root of the total size of the dataset.

The importance of normalizing the data is also highlighted since the KNN algorithm is sensitive to the scale of the variables. This algorithm is simple to implement and easy to understand [34], but it can have a high computational cost if working with large datasets due to the calculation of distances. In KNN algorithms, the choice of the distance metric directly affects how the proximity between points in the feature space is calculated, which in turn influences classification decisions [35]. The Euclidean distance is the most common metric; however, the Manhattan distance can also be used. Finally, we can mention the Minkowski distance, which generalizes the two previous ones and whose expression is given by

$$d(\mathbf{x}, \mathbf{x}_i) = \left(\sum_{i=1}^{n} |\mathbf{x} - \mathbf{x}_i|^q \right)^{\frac{1}{q}}. \tag{3}$$

When $q = 2$, it is equivalent to the Euclidean distance, and when $q = 1$, it is equivalent to the Manhattan distance.

For the problem we are considering, the method would be formulated according to the following steps. The objective of the KNN algorithm is to classify a new observation $\mathbf{x}_0$ into one of the classes $\hat{Y}$ using the information from the K nearest neighbors in the feature space:

1. Let a dataset with n observations, where each observation i has p input variables (features). This type of problem could involve features related to different system reliability metrics, such as time to failure, failure rate, system age, repair time, etc. We want to predict a class $Y_i \in \{0, 1\}$ for each observation.
2. Compute the distance between $\mathbf{x}_0$ and each observation $\mathbf{x}_i$ in the set using the chosen distance metric.
3. Choose the K neighbors with the smallest distances to $\mathbf{x}_0$.
4. The predicted class $\hat{Y}$ is the one with the majority of votes among the K nearest neighbors.

2.5. Random Forest

Random Forest (RF) is a powerful ML technique that has been applied in the field of reliability engineering to improve failure prediction, risk assessment, and maintenance decision-making. Its ability to handle complex, high-dimensional data, as well as providing insight into the most relevant factors affecting system reliability, makes it a particularly valuable tool in real-world applications.

Several scientific studies have explored the use of RF in reliability engineering; for example, to predict failures in semiconductor manufacturing equipment, demonstrating the high accuracy and robustness of the algorithm in handling complex operational data [36] to estimate product failures and warranty costs, highlighting the effectiveness of the model in analyzing warranty claims data and in projecting future costs based on historical failure data [37], and Ref. [38] proposes a model that integrates several machine learning techniques, including radio frequency, to predict failures in secondary power distribution networks. In particular, the model leverages the ability of radio frequency to identify relevant patterns in meteorological data and historical fault records, critical factors for predicting failures in power distribution systems.

In the field of reliability, RF operates by creating multiple independent and uncorrelated decision trees [39]. Each tree is trained on a randomly selected portion of the data, and at each split within the tree, a random subset of component states is used. This process, known as bagging or bootstrap aggregating, helps reduce model variance and prevent overfitting. Once all decision trees are trained, the RF algorithm makes predictions by combining the individual predictions from each tree. In classification problems, a majority voting strategy is used: the system state that receives the most votes among all trees is selected as the final prediction. The RF predictor is defined as the average of the predictions from B independent decision trees, each trained on a randomly selected dataset:

$$\hat{Y} = \frac{1}{B} \sum_{i=1}^{B} T_b(\mathbf{x}_i),$$

where B is the total number of trees in the forest, and T_b is the b-th decision tree. That is, if regression analysis is used, the probability that the system works will be the percentage of trees that have classified the system as working. If classification analysis is used, the estimate will be determined by majority vote. In this way, a final prediction is made that depends on the predictions resulting from the set of trees produced. This approach makes

RF particularly effective at handling large datasets with numerous features, providing efficient computational performance tailored to solving complex problems in ML [40].

For our problem, we consider the implementation of the following:

1. Bootstrap sampling: Training samples are created by randomly selecting, with replacement, a subset of component states for each tree. This procedure generates diverse datasets for building varied trees, introducing diversity into the model.
2. Node splitting: Within each tree, nodes are split using the best possible partition based on a random subset of features (or component states in this case). The goal is to minimize the impurity of child nodes after each split. The most common measure for classification is the Gini impurity or information gain (based on entropy).
$$G(p) = 1 - \sum_{k=1}^{K} p_k^2,$$
where p_k is the proportion of examples at the node that belongs to class k, and K is the total number of classes. The splitting of each node is performed in such a way as to minimize impurity.
3. Tree growth: trees are expanded to the maximum allowed size, without pruning except for explicit constraints such as maximum depth or minimum number of samples per leaf.
4. Aggregation of results: Once all trees are built, their predictions are combined to produce the final estimate. For classification, each tree makes a prediction for a given example. The class with the highest number of votes (majority voting) is selected.

The component state importance is based on the sum of the Gini impurity reduction, weighted by the number of samples arriving at each node, and averaged across all trees.

3. EvaluationFramework

3.1. Splitting the Sample for Training and Testing

The main objective of splitting the dataset is to evaluate the performance of an ML model objectively. The model is trained using only the training data, and its ability to generalize to new data is evaluated on the test set, which has not been previously used. Once the model has been trained on the training set, it is evaluated on the test set using appropriate metrics depending on the type of problem.

The process of splitting a dataset into training and test data is critical to evaluate the performance of an ML method. For this reason, to evaluate the proposed algorithm that combines unsupervised and supervised learning methods and the other ML methods previously introduced, we present the classic procedure that is common in this context, as shown in Figure 1.

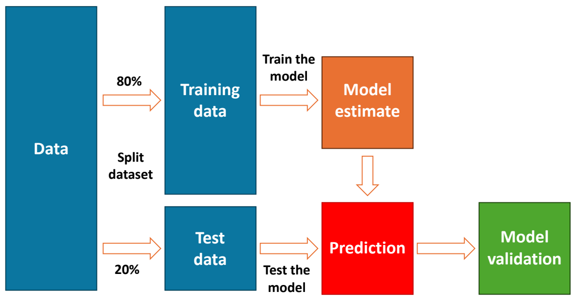

Figure 1. Splitting datasets.

3.2. Error Metrics for Model Evaluation

The most common error metrics in the context of ML algorithms for classification, as they are in our case, are based on a confusion matrix based on the values of the original response variable and the values predicted by the algorithm. The matrix has four entries that we call true positive (TP), true negative (TN), false positive (FP), and false negative (FN), respectively; see Figure 2. In our context we call a "positive" when the system is working, i.e., $Y = 1$, and a "negative" means that the system is in failure, $Y = 0$.

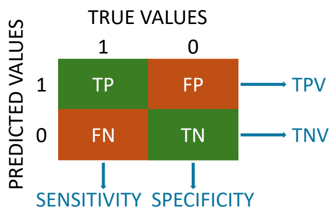

Figure 2. Confusion matrix.

Based on these measurements, the error metrics can be calculated as follows:

- Sensitivity: The systems are correctly classified as operative.

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP+FN}}$$

  It measures the probability with which the model correctly predicts that the system works.
- Specificity: The systems correctly classified as failed.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN+FP}}$$

  It measures the capacity (probability) of the model to detect failures in the system.
- Accuracy: The systems are correctly classified.

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+FP+TN+FN}}$$

  The probability that, given a configuration of components, the model correctly classifies the system.
- True Positive Value: The systems are correctly classified among all systems classified as operative.

$$\text{TPV} = \frac{\text{TP}}{\text{TP+FP}}$$

  The probability that the system is working when the model predicts it to be.
- F1-Score: This is the harmonic mean of TPV and sensitivity. Both metrics are weighted equally in the calculation, guaranteeing that the F1-Score accurately reflects the reliability of the model.

$$\text{F1-Score} = 2 \cdot \frac{\text{TPV} \cdot \text{Sensitivity}}{\text{TPV} + \text{Sensitivity}}$$

  A high F1-Score typically reflects a well-balanced performance, indicating that the model can achieve both a high TPV and high sensitivity simultaneously.

The area under the curve (AUC) is a measure used to evaluate the performance of a classification model. It is calculated as the probability that the predicted reliability for an operative system is higher than that of a failed system [19]. The dataset is divided into two groups: failed systems (denoted as A_F) and operative systems (denoted as A_O) with respective sample sizes n_F and n_O (where the total sample size $n = n_F + n_O$).

A linear regression model, or any other classification model, predicts the reliability $\hat{R}(\mathbf{x})$ for each system and assigns it to one of two classes, 0 for failed systems and 1 for operative systems. The AUC statistic quantifies the ability of the model to distinguish between these two classes, and it is defined as

$$\widehat{AUC} = \frac{1}{n_F \cdot n_O} \sum_{i \in A_F} \sum_{j \in A_O} I\big(\hat{R}(\mathbf{x}_i) < \hat{R}(\mathbf{x}_j)\big). \tag{4}$$

The $\widehat{AUC}$ value thus be used as an assessment of the discriminatory power of the classifier. The AUC ranges from 0 to 1, where the following is true:

- AUC=1, represents a perfect classifier with 100% discrimination power, meaning it can distinguish between failed and operative systems without any error.
- AUC=0.5, indicates a classifier with no discriminatory ability, equivalent to random guessing.
- $AUC < 0.5$, suggests that the classifier is performing worse than random guessing, potentially due to an inverted prediction mechanism.

The AUC is particularly useful for comparing multiple classifiers. It provides a single measure that summarizes the trade-off between true positive and false positive rates at different classification thresholds. This makes it threshold-independent, unlike measures such as accuracy, which depend on a specific decision boundary. The AUC is directly related to the Receiver Operating Characteristic (ROC) curve, which plots the true positive rate (sensitivity) against the false positive rate (1-specificity) for varying classification thresholds. The AUC represents the area under this curve, summarizing the overall performance of the model.

3.3. *The Particular Case of FA-LR-IS*

The various stages of the process are detailed below and illustrated in the flowchart in Figure 3.

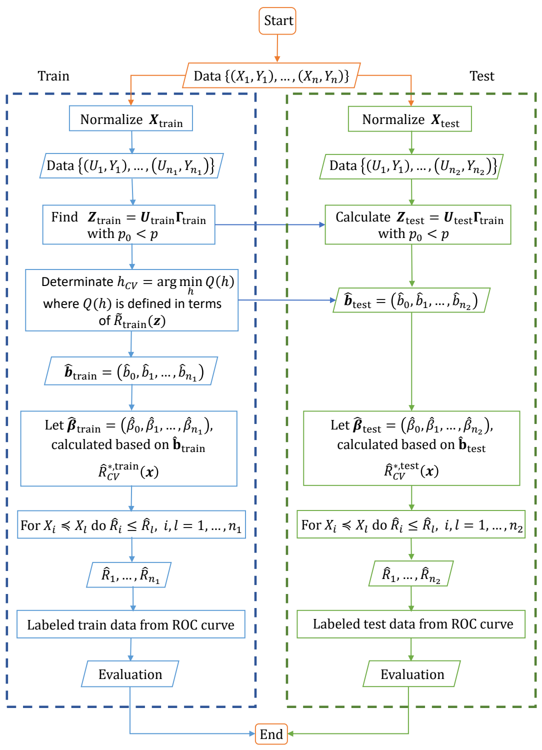
Start
Data $\{(X_1,Y_1),\ldots,(X_n,Y_n)\}$
Train
Test
Normalize $\boldsymbol{X}_{\text{train}}$
Normalize $\boldsymbol{X}_{\text{test}}$
Data $\{(U_1,Y_1),\ldots,(U_{n_1},Y_{n_1})\}$
Data $\{(U_1,Y_1),\ldots,(U_{n_2},Y_{n_2})\}$
Find $\boldsymbol{Z}_{\text{train}} = \boldsymbol{U}_{\text{train}}\boldsymbol{\Gamma}_{\text{train}}$ with $p_0 < p$
Calculate $\boldsymbol{Z}_{\text{test}} = \boldsymbol{U}_{\text{test}}\boldsymbol{\Gamma}_{\text{train}}$ with $p_0 < p$
Determinate $h_{CV} = \arg\min_h Q(h)$ where $Q(h)$ is defined in terms of $\tilde{R}_{\text{train}}(\boldsymbol{z})$
$\hat{\boldsymbol{b}}_{\text{test}} = (\hat{b}_0, \hat{b}_1, \ldots, \hat{b}_{n_2})$
$\hat{\boldsymbol{b}}_{\text{train}} = (\hat{b}_0, \hat{b}_1, \ldots, \hat{b}_{n_1})$
Let $\hat{\boldsymbol{\beta}}_{\text{train}} = (\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_{n_1})$, calculated based on $\hat{\mathbf{b}}_{\text{train}}$ $\hat{R}_{CV}^{*,\text{train}}(\boldsymbol{x})$
Let $\hat{\boldsymbol{\beta}}_{\text{test}} = (\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_{n_2})$, calculated based on $\hat{\mathbf{b}}_{\text{test}}$ $\hat{R}_{CV}^{*,\text{test}}(\boldsymbol{x})$
For $X_i \preccurlyeq X_l$ do $\hat{R}_i \leq \hat{R}_l,\ i,l = 1,\ldots,n_1$
For $X_i \preccurlyeq X_l$ do $\hat{R}_i \leq \hat{R}_l,\ i,l = 1,\ldots,n_2$
$\hat{R}_1,\ldots,\hat{R}_{n_1}$
$\hat{R}_1,\ldots,\hat{R}_{n_2}$
Labeled train data from ROC curve
Labeled test data from ROC curve
Evaluation
Evaluation
End

Figure 3. Training and testing stages of the FA-LR-IS algorithm.

3.3.1. Data Preprocessing

Center the matrix: $\mathbf{X}_{\text{test}}$: Define $U_j = (X_j - \mu_j)/\sigma_j$ for each $j = 1, 2, \ldots, p$. Let $\boldsymbol{\Sigma}_0$ be a diagonal matrix where the jth element is σ_j. Define $\mathbf{M}_{\text{test}} = \mathbf{1}_p(\mu_1, \mu_2, \ldots, \mu_p)$, where $\mathbf{1}_p$ is a unit column vector of size p. We denote $\mathbf{U}_{\text{test}} = (\mathbf{X}_{\text{test}} - \mathbf{M}_{\text{test}})\boldsymbol{\Sigma}_0^{-1}$, resulting in a matrix $\mathbf{U}_{\text{test}}$ with dimensions $n_2 \times p$.

3.3.2. Model Estimation

1. Transform the data to a reduced set denoted $\{\mathbf{Z}_{\text{test}}, \mathbf{Y}_{\text{test}}\}_{n_2 \times p_0}$, with $p_0 < p$, and $\mathbf{Z}_{\text{test}} = \mathbf{U}_{\text{test}}\boldsymbol{\Gamma}$, where $\boldsymbol{\Gamma}$ is the FA coefficient-matrix of dimension $p \times p_0$ obtained with the scaled dataset $\mathbf{U}_{\text{train}}$.
2. Let $\mathbf{x}_{0,\text{test}}$ represent a specific configuration of component states in the test dataset. We compute $\mathbf{z}_{0,\text{test}} = \big(\mathbf{x}_{0,\text{test}} - (\mu_1, \mu_2, \ldots, \mu_p)\big)\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\Gamma}$ and construct a local-logistic model. For any $\mathbf{z}_{\text{test}}$ such that $\|\mathbf{z}_{\text{test}} - \mathbf{z}_{0,\text{test}}\| < h_{CV}$, the fitted model is given by

$$\widetilde{R}_{CV}(\mathbf{z}_{\text{test}}) = \frac{e^{(1,(\mathbf{z}_{\text{test}} - \mathbf{z}_{0,\text{test}})^t)\widehat{\mathbf{b}}_{\text{test}}}}{1 + e^{(1,(\mathbf{z}_{\text{test}} - \mathbf{z}_{0,\text{test}})^t)\widehat{\mathbf{b}}_{\text{test}}}},$$

with $\widehat{\mathbf{b}}_{\text{test}} = (\widehat{b}_0, \widehat{b}_1, \ldots, \widehat{b}_{p_0})^t$ vector of coefficients obtained using h_{CV}, the optimal bandwidth selected with the LOOCV criterion applied to the train dataset.
3. Let $\mathbf{x}_{0,\text{test}}$ be defined such that $\|(\mathbf{x}_{\text{test}} - \mathbf{x}_{0,\text{test}})\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\Gamma}\| < h_{CV}$. Under this condition, we estimate the local-logistic model within the state space of components using

$$\widehat{R}^*_{CV}(\mathbf{x}_{\text{test}}) = \frac{e^{(1,(\mathbf{x}_{\text{test}} - \mathbf{x}_{0,\text{test}})^t)\widehat{\beta}_{\text{test}}}}{1 + e^{(1,(\mathbf{x}_{\text{test}} - \mathbf{x}_{0,\text{test}})^t)\widehat{\beta}_{\text{test}}}},$$

where the vector of estimated coefficients $\widehat{\beta}_{\text{test}}$ is

$$\begin{aligned} \widehat{\beta}_0 &= \widehat{b}_0, \\ \widehat{\beta}_j &= \sigma_j^{-1}\boldsymbol{\Gamma}_{j\cdot}\widehat{\mathbf{b}}_{-0}, \qquad j = 1, 2, \ldots, p; \end{aligned}$$

where we denote $\boldsymbol{\Gamma}_{j\cdot}$ the jth row of matrix $\boldsymbol{\Gamma}$, and $\widehat{\mathbf{b}}_{-0} = (b_1, \ldots, b_{p_0})^t$.
4. Define $\widehat{R}^*_{CV}(\mathbf{x}_{i,\text{test}}) = \frac{e^{b_0}}{1+e^{b_0}}\cdot$i, for $i = 1, 2, \ldots, n_2$. Apply the isotonization algorithm outlined in [19] to adjust these estimated values, thereby obtaining the estimated reliability for the ith configuration of states of components, that is, $\widehat{R}_{CV}(\mathbf{x}_{i,\text{test}})$, $i = 1, 2, \ldots, n_2$.
5. Analyze the labeled test data using the ROC curve to calculate the (AUC) as in (4) and evaluate the model.

4. Numerical Results

This section presents a simulation study of multiple systems, followed by an application to two real-world datasets. These analyses demonstrate the strong performance of the FA-LR-IS algorithm compared with the supervised learning methods mentioned above.

4.1. Simulations

To assess our method, we carried out a simulation study using systems with varying configurations. Many scientific fields involve static or dynamic systems composed of multiple components, which can be grouped into distinct, interacting blocks. We assume that the structural logic of the system can be represented using a block diagram. Figure 4 presents a graphical representation of the four cases analyzed in this section.

The data for each case were generated as follows: We simulate M = 500 samples with size $n = 125$. Let p represent the number of components in the system, with p taking values of 9, 10, 15, and 25 in our examples. The resulting data are organized into a matrix with $p + 1$ columns. The first p columns correspond to the states of the components, denoted as $X_1, X_2, \ldots, X_p$, where each $X_k \in [0, 1]$, $k = 1, \ldots, p$. The $(p + 1)$th column represents the system state, Y, which is 1 if the system is operational and 0 otherwise. Then the components are combined in blocks, denoted as F_j, for $j = 1, 2, 3$. for example, in System 1. We assume that $cor(X_k, X_{k'}) = 0.9$ if components k and k' are disposed of in the same block F and $cor(X_k, X_{k'}) = 0$ otherwise. This relationship is illustrated in Figure 4, where the blocks containing dependent components are marked with dashed lines.

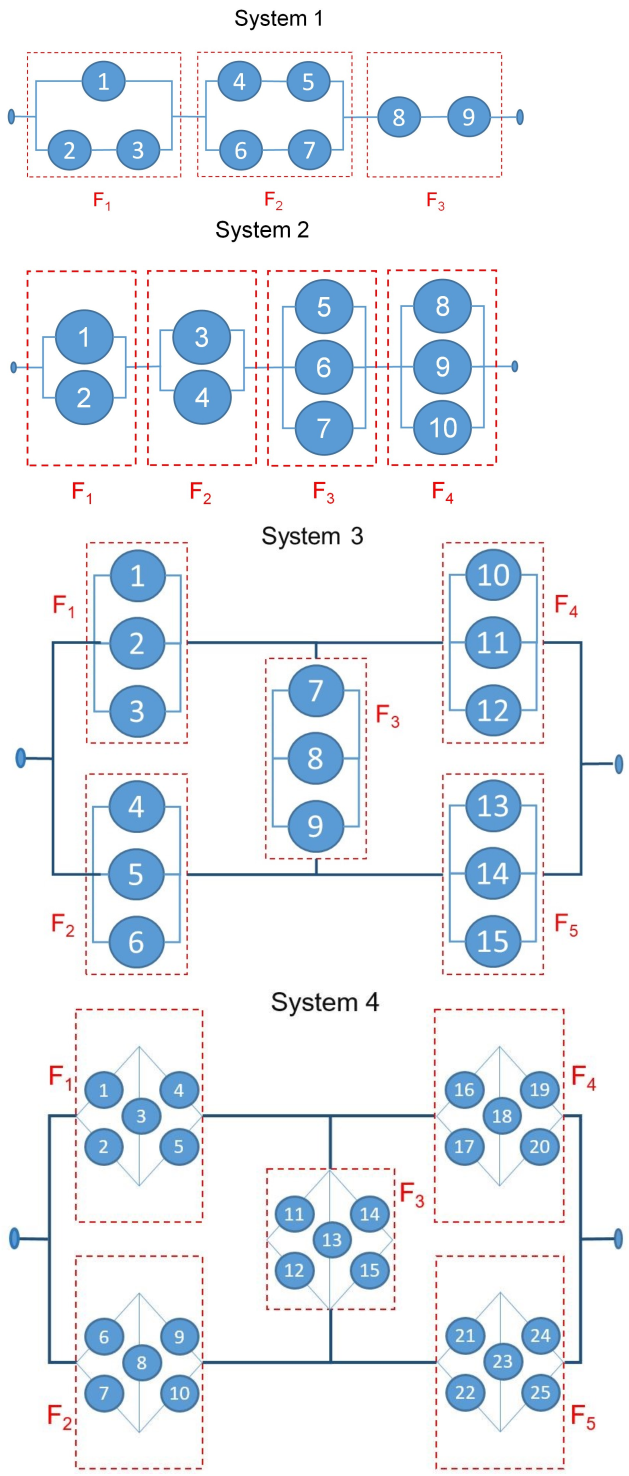

Figure 4. Configurations for the simulated systems.

The state of the system is simulated by incorporating a latent variable that is not directly observable, represented as $Y \to N(\Phi(\mathbf{X}), \sigma)$, with $\mathbf{X} = (X_1, X_2, \ldots, X_p)$ denoting a specific configuration of the state vector. We set $\sigma = 0.2$, and the information regarding the state of system Y is simulated from a binomial distribution, where the event probability is defined by $R(\mathbf{x}) = P(Y > y_0)$, with $y_0 = 0.5$.

The structure function of each model is given in the following:

- System 1. We examine a series-parallel system consisting of $p = 9$ components, as depicted in Figure 4 (top plot). The system is organized into three blocks connected in series. The first two blocks are arranged in parallel, containing three and four components, respectively. The third block is made up of two components connected in series. The structure function of this system is expressed as follows:

$$\phi(\mathbf{x}) = \min(\max(x_1, \min(x_2, x_3)), \max(\min(x_4, x_5), \min(x_6, x_7)), \min(x_8, x_9)),$$

where x_j denotes the state of the jth component, $j = 1, 2, \ldots, 9$.

- System 2. We examine a system consisting of a series–parallel combination with $p = 10$ components, as shown in Figure 4 (second plot). The system is arranged with four parallel blocks connected in series. The first two blocks each contain two components, while the last two blocks consist of three components each. The structure function for this system is represented by the following expression:

$$\phi(\mathbf{x}) = \min(\max(x_1, x_2), \max(x_3, x_4), \max(x_5, x_6, x_7), \max(x_8, x_9, x_{10})),$$

where x_j denotes the state of the jth component, $j = 1, 2, \ldots, 10$.

- System 3. We analyze a bridge system consisting of $p = 15$ components, as shown in the bottom plot of Figure 4 (third plot). The basic bridge structure has been modified to incorporate redundancy, where each component is replaced by a block of three parallel-connected units. The structure function of this system is expressed as follows:

$$\phi(\mathbf{x}) = \max(\min(\mathbf{x}_1, \mathbf{x}_4), \min(\mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_5), \min(\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4), \min(\mathbf{x}_2, \mathbf{x}_5))$$

where $\mathbf{x}_k = \max(x_{3k-2}, x_{3k-1}, x_{3k})$, for $k = 1, \ldots, 5$, and x_j denoting the state of the jth component, $j = 1, 2, \ldots, 15$.

- System 4. We consider a bridge structure with $p = 25$ components as displayed in the bottom plot of Figure 4 (bottom plot). Again, a simple bridge structure has been modified, introducing redundancy, but in this case, each node has been replaced by a block consisting of a bridge structure with five components. The structure function for this system is represented by the following expression:

$$\phi(\mathbf{x}) = \max(\min(\mathbf{x}_1, \mathbf{x}_4), \min(\mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_5), \min(\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4), \min(\mathbf{x}_2, \mathbf{x}_5))$$

where

$$\begin{aligned}\mathbf{x}_j = \max(&\min(x_{5j-4}, x_{5j-1}), \min(x_{5j-4}, x_{5j-2}, x_{5j}),\\ &\min(x_{5j-3}, x_{5j-2}, x_{5j-1}), \min(x_{5j-3}, x_{5j})),\end{aligned}$$

for $j = 1, \ldots, 5$, and x_k denoting the state of the kth component, $k = 1, 2, \ldots, 25$.

Each sample $\chi = \{(x_i, y_i), i = 1, \ldots, n\}$ is split into a training set and a test set, as illustrated in Figure 1. We then have $\chi = \{\chi_1 \cup \chi_2\}$, with the corresponding set of indices $I = \{1, 2, \ldots, 125\}$, similarly divided into $I = \{I_1 \cup I_2\}$, where I_1 represents the indices of the training set and I_2 represents the indices of the test set. Accordingly,

$$\begin{aligned}\text{car}(I_1) &= 0.8 \times n = n_1\\ \text{car}(I_2) &= n - n_1 = n_2\end{aligned}$$

The results presented below were obtained using the estimated reliability for each system (S = 1,2,3,4) with a sample size of $n = 125$, ensuring that the sample split remains consistent to facilitate reproducibility across methods; this guarantees that all algorithms are applied to the same datasets.

1. The FA-LR-IS algorithm begins by reducing the number of features using a factor analysis (FA) approach. This step identifies optimal factors representing the system. To determine the appropriate number of factors (or blocks) required for dimensionality

reduction, we employed the testing procedure implemented in the *factanal* function in the R-4.4.2 software, as described in [1]. The goal is to maintain an effective reduction in dimensionality without significant loss of information. Additionally, bandwidth, another critical parameter for this method, was selected using cross-validation techniques, as explained in Section 2.2.

2. For the artificial neural network (ANN) model, we employed a feedforward neural network with three layers, aimed at classifying the system state (0: failure, 1: operative). The ANN architecture includes 50 input neurons, two hidden layers with 15 and 80 neurons, respectively (both using ReLU activation), and a final output neuron with sigmoid activation for binary classification. The network was optimized using the ADAM algorithm with the binary cross-entropy loss function (2). The ADAM algorithm, which combines AdaGrad and RMSprop to dynamically adjust the learning rates of the parameters based on the first and second derivatives of the gradient, was chosen due to its adaptability, ease of implementation, and computational efficiency [28]. We ran the model for 125 epochs with a batch size of 64. The architecture (layers and neurons) was selected by evaluating various brute-force combinations.
3. We used a K-nearest neighbors (KNN) classifier with the number of neighbors set to 20, a value determined to be optimal through brute-force search. The distance metric used was Minkowski distance, given in (3).
4. For RF we used a random forest classifier with 100 decision trees, which were chosen through trial and error to balance performance without overfitting.

Below, we present a series of tables and graphs with different measures to compare the FA-LR-IS algorithm with other ML methods.

First, AUC was used to assess the goodness of fit; a higher value of this measure means that the model achieves better overall performance. Table 1 shows the mean and standard deviation (SD) of the AUC calculated over all repetitions in the experiment. All values are displayed in Figure 5; each system is color-coded for clarity. The FA-LR-IS algorithm, with the highest values in the table, consistently outperforms the other methods across all system configurations, demonstrating superior discriminatory power as a classification method.

Table 1. Mean and SD of area under the ROC curve (AUC), defined in (4).

Models	**System 1**	**System 2**	**System 3**	**System 4**
	Mean (SD)	**Mean (SD)**	**Mean (SD)**	**Mean (SD)**
FA-LR-IS	**0.8218** (0.1338)	**0.7708** (0.1273)	**0.7599** (0.1045)	**0.7674** (0.0992)
ANN	0.6902 (0.1423)	0.6702 (0.1373)	0.6907 (0.0970)	0.6998 (0.0941)
KNN	0.5763 (0.1212)	0.5510 (0.0905)	0.6428 (0.0981)	0.6754 (0.0929)
RF	0.6315 (0.1395)	0.5928 (0.1110)	0.6632 (0.1021)	0.6894 (0.0933)

For each system the highest value appears in bold.

Next, we consider the Mean Squared Error (MSE) to evaluate the accuracy of a model. Then, for each system, given a particular sample of a test dataset $\chi_2 = \{(\mathbf{x}_i^m, y_i^m); i = 1, \dots, n_2\}$, with $m = 1, \dots, M$, the MSE_m was calculated using the formula:

$$MSE^{\bullet}_{S,m} = \frac{1}{n_2} \sum_{i=1}^{n_2} \left(\widehat{R}^{\bullet,m}_{S,i} - R_S(\mathbf{x}_i^m) \right)^2, \tag{5}$$

where $R_S(\mathbf{x}_i^m)$ represents the true reliability function for the structure S, and $\widehat{R}^{\bullet,m}_{S,i}$ denotes the reliability estimated using the corresponding method. Table 2 presents the average values and standard deviations (SD) of $MSE^{\bullet}_{S,m}$ across M repetitions; a lower MSE indicates that the model predictions are very close to the actual values. Table 2 shows that FA-LR-IS outperforms all other methods for Systems 3 and 4, achieving the lowest error, while ANN performs comparably for Systems 1 and 2; although it can be seen in Figure 6 that FA-LR-IS and ANN perform similarly concerning MSE for the four systems considered, the values are very similar.

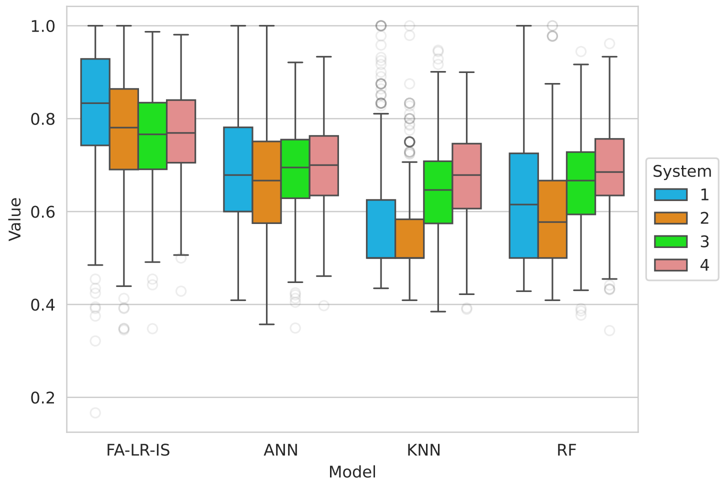

Figure 5. Boxplot comparing *AUC* between models.

Table 2. Mean and SD of *MSE* for each system and model defined in (5).

Models	**System 1**	**System 2**	**System 3**	**System 4**
	Mean (SD)	**Mean (SD)**	**Mean (SD)**	**Mean (SD)**
FA-LR-IS	0.0202 (0.0107)	0.0298 (0.0139)	**0.0487** (0.0165)	**0.0456** (0.0163)
ANN	**0.0199** (0.0122)	**0.0278** (0.0159)	0.0515 (0.0214)	0.0565 (0.0245)
KNN	0.3150 (0.0521)	0.2787 (0.0494)	0.1406 (0.0340)	0.1273 (0.0319)
RF	0.3307 (0.0519)	0.3009 (0.0490)	0.1568 (0.0379)	0.1503 (0.0381)

For each system the lowest value appears in bold.

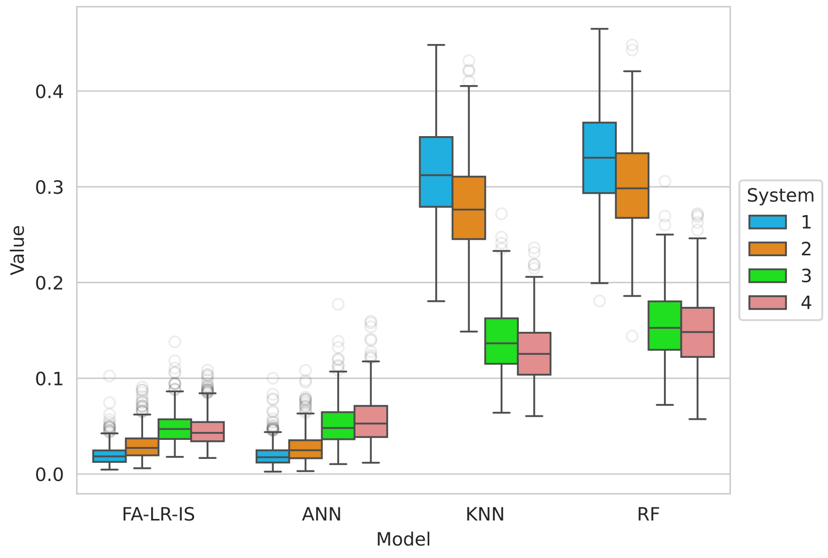

Figure 6. Boxplot comparing *MSE* between models.

Table 3 provides the mean and SD for the accuracy of the estimator measured in terms of predictive capacity as defined in Section 3.2 across repetitions. The FA-LR-IS algorithm outperforms the other methods for Systems 3 and 4, while ANN performs best for System 1 and KNN for System 2. That is, our algorithm has been shown to achieve a greater predictive capacity in the case of a more complex system than if it is simpler. All values obtained are displayed in Figure 7.

Table 3. Accuracy, measured as indicated in Section 3.2.

Models	System 1	System 2	System 3	System 4
	Mean (SD)	**Mean (SD)**	**Mean (SD)**	**Mean (SD)**
FA-LR-IS	0.8166 (0.1214)	0.7786 (0.1166)	**0.7523** (0.0891)	**0.7570** (0.0845)
ANN	**0.8482** (0.0752)	0.8259 (0.0764)	0.7245 (0.0854)	0.6990 (0.0941)
KNN	0.8440 (0.0746)	0.8250 (0.0763)	0.7010 (0.0902)	0.6743 (0.0930)
RF	0.8482 (0.0730)	**0.8263** (0.0763)	0.7068 (0.0919)	0.6880 (0.0927)

For each system the highest value appears in bold.

In Figure 7, the accuracy of the estimator, measured in terms of predictive capacity as defined in Section 3.2, is shown.

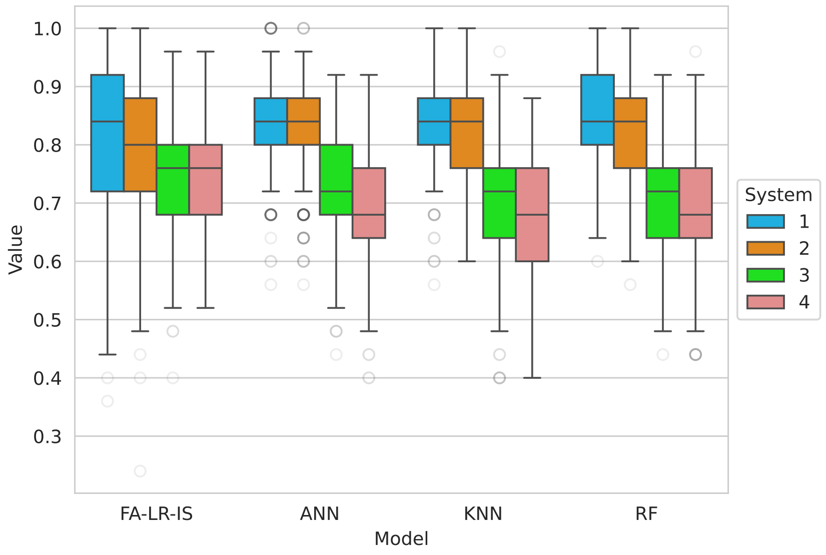

Figure 7. Boxplot comparing predictive capacity between models.

Finally, given that FA-LR-IS and ANN were the most competitive methods, we conducted direct bootstrap comparisons between them. For each measure (*AUC*, *MSE*, or accuracy), the following procedure was implemented:
Steps

1. For the 500 sample realizations of the measure obtained for each method (FA-LR-IS and ANN), calculate the difference between the means.
2. Combine all results from both methods and draw two samples with replacements from this combined data, each of size 500. Calculate the means of these samples and then compute the difference between them.
3. Repeat Steps 1 and 2 a total of 10,000 times to obtain an asymptotic bootstrap distribution for the difference in means.
4. Calculate the p-value as the proportion of bootstrap differences that are less than or equal to the absolute value of the observed difference in means from the initial 500 observations.

Table 4 shows the p-values for hypothesis tests assessing the equality of the measurements obtained with each method. For *AUC* and *MSE*, FA-LR-IS significantly outperforms ANN (p-value < 0.0001). Although ANN achieved better results in accuracy for Systems 1 and 2, the differences were not statistically significant. In contrast, FA-LR-IS significantly outperformed ANN in Systems 3 and 4. Notably, when FA-LR-IS was superior, the difference was statistically significant, whereas the advantage of ANN was not significant in the cases where it performed better.

Table 4. Comparing the *p*-value for each of the metrics for FA-LR-IS and ANN.

Metrics	System 1	System 2	System 3	System 4
AUC	<0.0001	<0.0001	<0.0001	<0.0001
MSE	<0.0001	<0.0001	<0.0001	<0.0001
Accuracy	0.5986	0.0304	0.0198	<0.0001

4.2. Scalability Analysis

We have calculated the execution time based on sample sizes of 50, 100, 250, 500, and 1000, simulated from System 4 shown in Figure 4. On the one hand, the FA-LR-IS algorithm has been implemented using a 3.60-GHz Intel Core i5-8600K processor. On the other hand, the simulations for the ML methods have been performed using a 2.20-GHz Intel Xeon processor. The simulation results are shown in Table 5. We observe that the execution time for all the methods grows with the sample size, with the AF-RL-IS algorithm standing out. The cross-validation method used for bandwidth selection makes the time to increase exponentially. However, this algorithm yields the best goodness of fit, as can be seen from the AUC, MSE, and accuracy results for System 4 shown in Tables 1–3. As for the ML techniques, we see that the ANN algorithm provides longer execution time than RF and KNN, the latter being the one that yields the smallest execution time and worse results for the AUC, MSE, and accuracy measures. The KNN method is sensitive to data scaling; therefore, it exhibits low execution times due to the variables being normalized beforehand. In this context, we can assert that, for System 4, the models that take the longest to train are those that yield the best goodness-of-fit results. However, despite the comparison of the different methods studied, it should be noted that their recorded execution times are not directly comparable, as they were not performed under the same conditions. The ML techniques have been implemented using Python 3.10.12, with optimized functions that have been tested by a large team of professionals. In contrast, the FA-LR-IS method has been implemented in R-4.4.2, utilizing functions developed by the authors that are not fully optimized for computational efficiency. Additionally, this software has more limited resources for performing complex calculations with large datasets. In this regard, we propose the creation of an R package in the short term, where the functions will be optimized. The package will be made available on the CRAN repository and on our personal website, www.reliastat.com.

Table 5. Recorded execution time (in seconds) for the training dataset with the different algorithms for System 4, for sample size n.

Sample Size	FA-LR-IS	ANN	KNN	RF
50	16.5942	4.8824	0.0009	0.1699
100	44.9332	5.0662	0.0046	0.1974
250	193.1672	6.2501	0.0050	0.1676
500	692.8524	7.4171	0.0046	0.1842
1000	2508.6760	8.9800	0.0081	0.2221

4.3. First Real Case Study: A Water Pump Sensor Monitoring System

We analyze a dataset concerning an industrial structure; in particular, we have performance measurements of a water pump installed in a small area. The dataset was sourced from the data platform www.kaggle.com (accessed on 1 February 2024), and a statistical analysis of this data is presented in [41]. The sample information was collected by a set of sensors monitoring various components of the water pump over time. Specifically, 50 sensors measured parameters such as temperature, pressure, vibration, load capacity, volume, and flow density, among others, every minute from April 1st, 2018, to August 31st, 2018. In total, there are 220.320 observations. Limited information is available regarding the behavior of the sensors, but according to the study of one of the experts who have analyzed the data (available on the data website), the intermediate group of sensors (sensor16–sensor36) corresponds to the performance of two impellers, while the first 14 sensors monitor aspects related to the engine.

The data consist of a longitudinal follow-up of a single system, with observations taken at one-minute intervals between consecutive data points. To ensure that the systems in the sample are independent, we did not use all the available records. Instead, we increase the time gap between the data points we analyze. In [1] we considered a small sample with all records taken in the first minute of every day and focused the study on building a ranking of components in terms of the effect that changes in each component had on the reliability of the system. In this case, we consider sampled data at one-hour intervals, resulting in a sample size of $n = 3672$, and we applied the same algorithms used in previous simulations. Specifically, we implemented the FA-RL-IS algorithm following this procedure:

1. We split the sample into training (80%) and test (20%) sets (Step 1).
2. To train the model, we proceeded as follows:
   (a) Data normalization: since the scales of the sensor measurements vary, it was necessary to normalize the data to avoid the influence of variables with larger scales and ensure comparability.
   (b) Factor analysis: this step involved:
      i. Examine possible correlations. The correlation matrix with all variables is shown in Figure 8. Some sensor groups, such as the intermediate group, show high positive correlations within the group.

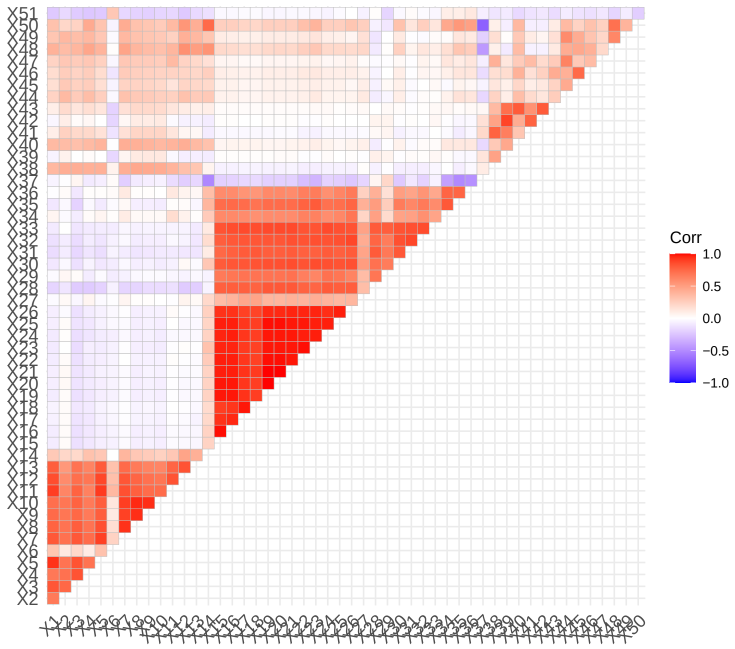

Figure 8. Correlation matrix for water pump sensor monitoring system.

      ii. Determine the appropriate number of factors. Using the R package *psych* [42], we determined the number of factors to extract, based on a *scree* plot (Figure 9).
      iii. Conduct the factor analysis. We used the *fa* function from the *psych* package to perform an exploratory factor analysis of latent variables using maximum likelihood. The correlation matrix was decomposed into eigenvalues and eigenvectors, estimating the commonalities for each variable across the first five factors. Factor loadings and interfactor correlations were also obtained.
   (c) Local-logistic estimation: We then fitted a local-logistic model in the space of the first five factors, $p_0 = 5$, and back-transformed the results to the original

feature space, $p = 50$. The bandwidth parameter was estimated through cross-validation. We now had a model that predicts the probability that the machine is functioning (reliability function) based on the sensor values.

(d) Classification: After estimating the probabilities with the logistic regression model, we translated these probabilities into classes or categories. For this, we used the classification obtained via the $\widehat{AUC}$, see (4).

3. Test the model:
   (a) Normalize the data.
   (b) Transform the data to a reduced set.
   (c) Local-logistic estimation. We applied the same model used to predict the reliability function to the test dataset.
   (d) Classification. We classified the observations in the test set using the same method as with the training set.
4. Metrics: All previous steps generated results to compute error metrics on the training dataset. We calculated these error metrics using the test dataset, fitting the local-logistic model, and applying the bandwidth obtained in Step 2(c) on the training data. The test set was classified using the model obtained with the training set.

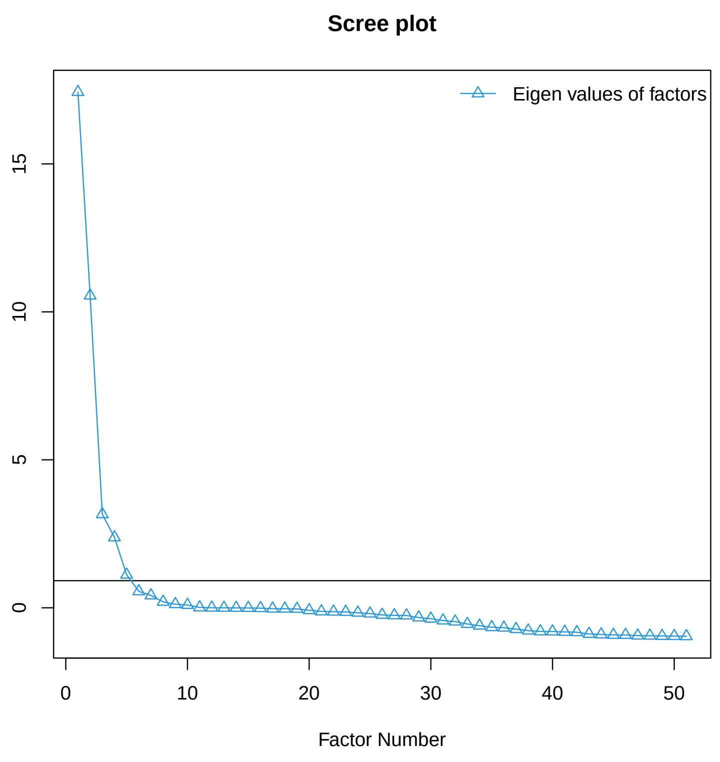

Figure 9. Scree plot for water pump sensor monitoring system.

In addition to the proposed algorithm, we also executed the ANN, KNN, and RF algorithms using the same sample split and parameters as in the simulations. Table 6 shows the error metrics calculated on the test dataset. We observe that the KNN algorithm yields the highest specificity, accuracy, TPV, and F1-Score, while ANN shows the highest sensitivity and the same accuracy and F1-Score as KNN. The proposed algorithm shows high and competitive values comparable to the other ML algorithms, except for specificity, where it underperforms relative to the other methods.

Table 6. Error metrics defined in Section 3.2 calculated on test water pump sensor dataset.

	FA-LR	**ANN**	**KNN**	**RF**
Sensitivity	0.9845	**0.9942**	0.9927	0.9927
Specificity	0.9167	0.9792	**1**	0.9792
Accuracy	0.9809	**0.9932**	**0.9932**	0.9918
TPV	0.9941	0.9985	**1**	0.9985
F1-Score	0.9898	**0.9963**	**0.9963**	0.9956

For each error metric the highest value appears in bold.

4.4. A Second Real Case Study: Condition Monitoring of Hydraulic Systems

We analyze a dataset containing information collected from a hydraulic system. The dataset is available on the website archive.ics.uci.edu (accessed on 20 September 2024). The data contain useful information for developing predictive models to detect faults early, contributing to the safety and operational efficiency of these systems. These data have been analyzed in several works, exploring the use of advanced data analysis techniques, such as multivariate statistics, sensor fault compensation, and automatic feature extraction, to improve condition monitoring in complex hydraulic systems [43–45].

The data were collected using a hydraulic test rig. This rig features two circuits: a primary working circuit and a secondary cooling-filtration circuit, both linked through an oil tank. The system performs cyclic load tests of 60 seconds, during which it records process parameters such as pressures, flow rates, and temperatures. Simultaneously, the conditions of four key hydraulic components (cooler, valve, pump, and accumulator) are varied in a controlled manner. A total of 2205 observations are recorded, involving 17 sensors. The target condition value recoded was the “stable flag”: 1 if conditions were stable and 0 if static conditions might not have been reached yet.

We have also applied to this dataset the ML algorithms considered previously, i.e., ANN, KNN, and RF, as well as our FA-LR-IS algorithm. The main results are summarized in the following:

1. We split the sample into training (80%) and test (20%) sets (Step 1).
2. To train the model, we proceeded as follows:
   (a) Data normalization.
   (b) Factor analysis: This step involved the following:
      i. Examine possible correlations. The correlation that exists between the variables is remarkable, as can be seen in Figure 10.
      ii. Determine the appropriate number of factors to extract, based on (Figure 11).
      iii. Conduct the factor analysis.
   (c) Local-logistic estimation. We then fitted a local-logistic model in the space of the first three factors, $p_0 = 3$, and back-transformed the results to the original feature space, $p = 17$. The bandwidth parameter was estimated through cross-validation.
   (d) Classification. After estimating the probabilities with the logistic regression model, we translated these probabilities into classes or categories via the $\widehat{AUC}$; see (4).
3. Test the model: do steps 3(a–d) as in the previous example.
4. Metrics: We calculated the error metrics using the test dataset, fitting the local-logistic model and applying the bandwidth obtained in Step 2(c) on the training data. The test set was classified using the model obtained with the training set.

Finally, we executed the ANN, KNN, and RF algorithms using the same sample split. Table 7 shows the error metrics calculated on the test dataset. We observe that, in this case, the RF algorithm yields the highest values in almost all metrics, followed by the

KNN algorithm. The FA-LR-IS algorithm shows high values comparable to the other ML algorithms, proving to be a strong competitor.

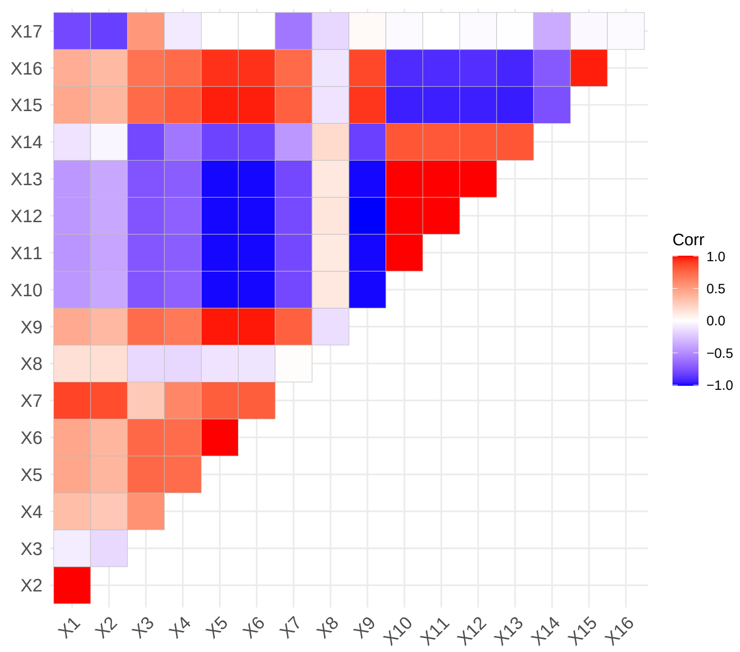

Figure 10. Correlation matrix for hydraulic system data.

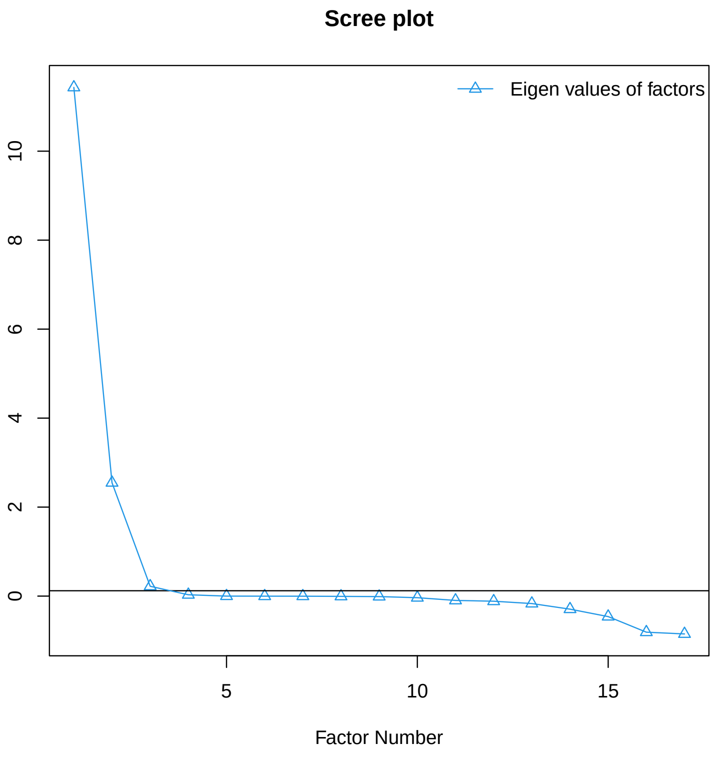

Figure 11. Scree plot for hydraulic system data.

Table 7. Error metrics defined in Section 3.2 calculated on test hydraulic system dataset.

	FA-LR	**ANN**	**KNN**	**RF**
Sensitivity	0.9542	0.6725	**0.9613**	0.919
Specificity	0.7962	0.9108	0.8344	**0.9172**
Accuracy	0.8980	0.7574	0.9161	**0.9184**
TPV	0.8944	0.9317	0.913	**0.9526**
F1-Score	0.9223	0.7812	**0.9365**	**0.9365**

For each error metric the highest value appears in bold.

5. Discussion

One significant limitation of our study is the inability to train a deep learning model for comparison with our custom statistical model. While deep learning models typically outperform ANNs, they require substantial datasets to learn effectively [46]. Unfortunately, our datasets are insufficient in size to support the training of such a model. Additionally, the computational resources necessary for training deep learning models are beyond our current capabilities. The need for multiple data sources, such as labeled and unlabeled data in semi-supervised learning, adds another layer of complexity to determining the right dataset size. This requires careful consideration of how to balance different types of data [47]. Designers must also consider performance targets, collection costs, and penalties for failing to meet these targets, which complicates the decision-making process regarding dataset size. Training large models with very large datasets can take months of computational power [48], although several strategies have been proposed to curtail the trial and error time [49].

A key advantage of classical statistical models over ML models is interpretability. While ML models often operate as black boxes, classical statistical models are explicitly defined through mathematical equations, which are more objectively quantifiable.

Although some ML methods attempt to overcome this limitation using procedures like SHAP for all ML models or MDI for random forests, these methods require fine-tuning numerous hyperparameters to achieve optimal performance. In contrast, in our probabilistic approach, implemented through the FA-LR-IS algorithm, the only tuning parameter is the bandwidth or smoothing parameter, which is determined via cross-validation, making it entirely data-driven and not sensitive to user-selected options. Our statistical methodology is fully non-parametric, allowing it to be flexible enough to capture complex relationships among variables, such as non-linearity and interactions.
That being said, we acknowledge the high predictive power of ML methods, which operate with lower computational complexity even with large amounts of data compared with our statistical model, as demonstrated in Table 5, included in the new version of the paper. For this reason, one of our future research directions is to propose a hybrid method that combines the computational power of ML methods with the objectivity and clear formulation of a statistical model. Table 8 summarizes the strengths and weaknesses of each method.

An interesting avenue for future research could involve measuring the performance of FA-LR-IS against a deep learning model over time. Because of the simulations, we hypothesize that our statistical model will initially outperform deep learning models in scenarios where the system is new and data availability is limited, due to its lower dependency on large volumes of data. However, as the system matures and accumulates more data, the deep learning model may exhibit superior performance.

We aim to explore several questions in future work: At what point does this shift occur? How does it differ from system to system? Is the improvement worth the cost of deploying a deep learning model?

There are also models of ANNs that allow for feedback loops, known as recurrent neural networks (RNNs). While RNNs have been less influential than feedforward networks, partly because their learning algorithms are less powerful to date, they are still extremely interesting. RNNs more closely resemble the way our brains operate, and they may be capable of solving important problems that feedforward networks can only tackle with great difficulty. RNNs are more effective for large and complex datasets, while classical methods are better suited for smaller, univariate datasets. RNNs consistently outperform ARIMA and exponential smoothing in terms of accuracy, especially for seasonal time series, as demonstrated in various studies. RNNs exhibit lower Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) compared with classical methods, indicating superior forecasting capabilities. RNNs are particularly advantageous for long-term forecasts, whereas ARIMA may perform better in short-term predictions [50,51].

Table 8. Comparison of strengths and weaknesses of FA-LR-IS algorithm with ML methods.

Methods	Data Requirements	Computational Cost	Overfitting	Interpretation
AF-LR-IS	Overcome other methods with small datasets	Longer execution time	Bandwidth parameter can be tuned to avoid overfitting	Easy to interpret
ANN	Requires large datasets	Intensive training and resources	Overfitting due to hyperparameter	Difficult to interpret due to complex structure
KNN	Requires large datasets	Limited scalability with large datasets	Similar to AF-LR-IS	Relatively easy to interpret
RF	Requires large datasets	Moderate, but the ensemble nature increases complexity	Lower risk of overfitting due to ensemble approach	Harder to interpret because of the ensemble approach

6. Conclusions

In this paper, we refine a previously introduced probabilistic algorithm to align with the standard ML framework of training and testing. This refinement allows us to systematically evaluate its performance, showing it outperforms leading ML methods such as RF and ANN in the following sense. Unlike black-box ML models, our probabilistic approach is fully interpretable, providing meaningful insights into the system state and quantifying individual component importance. By combining the procedural strengths of ML with the transparency of probabilistic modeling, we present a robust, hybrid algorithm that is both powerful and practical for engineering applications. To apply ML techniques, it is essential to carry out a thorough analysis of the data beforehand and reflect on which approach would be most appropriate, whether using ANN, KNN, or another algorithm. Additionally, we explore the use of ML methods like RF for reliability estimation, an area traditionally dominated by probabilistic parametric approaches, and conduct an extensive comparison of classical statistical methods versus ML techniques in the reliability context.

To illustrate the benefits of our method, we carried out a simulation study and applied it to two real datasets. In all instances, the FA-LR-IS algorithm yielded favorable results regarding model accuracy, making it a strong competitor for any ML method. The algorithm even is able to identify internal dependency structures in the system. In summary, the FA-LR-IS algorithm can be considered a versatile and effective option when there is uncertainty about which method to use, acting as a general solution in situations of uncertainty.

In the simulations, KNN and RF have been shown to offer less accurate estimations of the reliability of the system, providing considerably poorer results than the FA-LR-IS algorithm and the ANN. This is because these estimates are only averages and are not formulated with a mathematical expression that depends on the state of the system.

This study suggests that the FA-LR-IS model is not intended to be a definitive solution but rather an intermediate step while better resources are being developed. This idea is reinforced by the conclusion that there is no single method to solve all reliability estimation problems. The FA-LR-IS is presented as a versatile and effective solution in scenarios with uncertainty, aligning with the need to use diverse tools depending on the context.

Future work will focus on investigating the flexibility of the FA-LR-IS algorithm in quantifying the localized effects of specific units on system performance, comparing these findings with the machine learning methodologies discussed in this paper, as well as other methods. It is also proposed to extend machine learning techniques, such as random forests (RFs), to address the problem of reliability estimation while ensuring that system consistency conditions are met. In this context, the approach will focus on analyzing methods to optimize the selection of input samples for tree growth. Additionally, the study will explore more accurate methods for determining the optimal criteria for splitting the nodes of the regression trees used in building the random forest model. A measure will also be developed to quantify the impact of each individual component on the overall system performance, allowing for a ranking of components within the system structure. Managing the weakest areas would allow preventing failures that result in significant economic losses. This approach would facilitate the design of preventive maintenance policies aimed at mitigating the probability of failures originating in the most vulnerable components.

Author Contributions: Conceptualization and supervision, methodology, software, validation, investigation, resources, writing—original draft preparation, writing—review and editing, visualization, M.L.G., F.N.-G., R.A.N.C. and R.R.-M. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been partially supported by ERDF/Spanish Ministry of Science and Innovation—State Research Agency, under grant PID2020-120217RBI00, and the IMAG-María de Maeztu grant CEX2020-001105/AEI/10.13039/501100011033.

Data Availability Statement: Data are publicly available through www.kaggle.com (accessed on 1 February 2024).

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

Abbreviations

The following abbreviations are used in this manuscript:

$R(x_1, x_2, \ldots, x_p)$	Reliability of the system as a function of the components states.
n, n_1, n_2	Sample sizes of the observed dataset, training set and test set.
p, p_0	Numbers of components. Optimal number of factors.
$\mathbf{X}$	Matrix of the components states of system.
$\mathbf{Y}$	Vector of system states.
$\mathbf{X}_{\text{train}}, \mathbf{X}_{\text{test}}$	Data matrix of training and test set.
$\mathbf{Y}_{\text{train}}, \mathbf{Y}_{\text{test}}$	Response vector of training and test set.
$\mathbf{\Sigma}_0$	Diagonal matrix whose elements are standard error.
$\mathbf{M}_{\text{train}}, \mathbf{M}_{\text{test}}$	Mean matrix of training and test set.
$\mathbf{1}_p$	Unit column vector of size p.
$\mathbf{U}_{\text{train}}, \mathbf{U}_{\text{test}}$	Centered matrix of training and test set.
$\mathbf{Z}_{\text{train}}, \mathbf{Z}_{\text{test}}$	FA score-matrix of training and test set.
$\mathbf{\Gamma}_{\text{train}}$	FA coefficient-matrix of training set.
h, h_{CV}	Bandwidth. Optimal bandwidth based on cross-validation.
$\widetilde{R}_h$	Estimated reliability based on reduced dataset with bandwidth h.
$\{\mathbf{Z}_{\text{train}}, \mathbf{Y}_{\text{train}}\}^{(-i)}$	Leave-one-out dataset.
$\widetilde{R}_h^{(-i)}$	Fitted model using the leave-one-out dataset.
$Q(h)$	Cross-validation score.
$\mathbf{x}_{0,\text{train}}$	Specific configuration of component states in the training set.
$\mathbf{z}_{0,\text{train}}$	Specific configuration of reduced data in the training set.
$\widehat{R}_{CV}$	Fitted model using optimal bandwidth.
$\widehat{\mathbf{b}}_{\text{train}}$	Vector of coefficients based on optimal bandwidth within reduced space in the training set.
$\widehat{R}^*_{CV}$	Fitted model within the original state space of components.
$\widehat{\beta}_{\text{train}}$	Vector of coefficients within the original state space of components in the training set.

$\widehat{R}_{CV}$	Isotonized fitted model.
$\mathbf{x}_{0,\text{test}}$	Specific configuration of component states in the test set.
$\mathbf{z}_{0,\text{test}}$	Specific configuration of reduced data in the test set.
$\widehat{\mathbf{b}}_{\text{test}}$	Vector of coefficients based on optimal bandwidth within reduced space in the test set.
$\widehat{\beta}_{\text{test}}$	Vector of coefficients within the original state space of components in the test set.
A_F, A_O	Failed and operative systems sample.
n_F, n_O	Failed and operative systems sample size.
$cor(X_k, X_{k'})$	Correlation between components k y k'.
I, I_1, I_2	Set of indices of the observed data, training data and test data.
$R_S, \widehat{R}_S$	True and estimated reliability function for the structure S.
MSE	Mean square error.

References

1. Gámiz, M.L.; Navas-Gómez, F.; Nozal-Cañadas, R.; Raya-Miranda, R. Unsupervised and supervised learning for the reliability analysis of complex systems. *Qual. Reliab. Eng. Int.* **2023**, *39*, 2637–2658. [CrossRef]
2. Li, M.; Wang, Z. Deep learning for high-dimensional reliability analysis. *Mech. Syst. Signal Process.* **2020**, *139*, 106399. [CrossRef]
3. Fan, J.; Gijbels, I. *Local Polynomial Modelling and Its Applications*; Taylor & Francis: Milton Park, UK, 1996.
4. Hastie, T.; Tibshirani, R.; Friedman, J.; Franklin, J. The elements of statistical learning: Data mining, inference and prediction. *Math. Intell.* **2005**, *27*, 83–85.
5. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
6. Murphy, K.P. *Machine Learning: A Probabilistic Perspective*; MIT Press: Cambridge, MA, USA, 2012.
7. Xu, Z.; Saleh, J.H. Machine learning for reliability engineering and safety applications: Review of current status and future opportunities. *Reliab. Eng. Syst. Saf.* **2021**, *211*, 107530. [CrossRef]
8. Xu, Y.; Sun, K.; Zhang, Y.; Chen, F.; He, Y. A State Monitoring Algorithm for Data Missing Scenarios via Convolutional Neural Network and Random Forest. *IEEE Access* **2024**, *12*, 137080–137088. [CrossRef]
9. Daya, A.A.; Lazakis, I. Systems reliability and data driven analysis for marine machinery maintenance planning and decision making. *Machines* **2024**, *12*, 294. [CrossRef]
10. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
11. Goldblum, M.; Finzi, M.; Rowan, K.; Wilson, A.G. The no free lunch theorem, Kolmogorov complexity, and the role of inductive biases in Machine Learning. *arXiv* **2024**, arXiv:2304.05366.
12. Wang, X.; Du, Y.; Liu, K.; Luo, Y.; Du, B.; Tao, D. Separable power of classical and quantum learning protocols through the lens of no-free-lunch theorem. *arXiv* **2024**, arXiv:2405.07226.
13. Wolpert, D.H. The implications of the no-free-lunch theorems for meta-induction. *J. Gen. Philos. Sci.* **2023**, *54*, 421–432.
14. Aikhuele, D.; Nwosu, H.; Ighravwe, D. Data-driven model for the evaluation of the reliability of sensors and actuators used in IoT system architecture. *J. Reliab. Intell. Environ.* **2022**, *9*, 135–145. [CrossRef]
15. Choi, W.H.; Kim, J. Unsupervised Learning approach for anomaly detection in industrial control systems. *Appl. Syst. Innov.* **2024**, *7*, 18. [CrossRef]
16. Paluszek, M.; Thomas, S.; Ham, E. *Practical MATLAB Deep Learning*, Last ed.; Apress, Berkeley, CA, USA, 2022.
17. Aggarwal, C.C. *Neural Networks and Deep Learning*; Springer: New York, NY, USA, 2018.
18. Hussain, M.; Zhang, T.L.; Chaudhry, M.; Jamil, I.; Kausar, S.; Hussain, I. Review of prediction of stress corrosion cracking in gas pipelines using machine learning. *Machines* **2024**, *12*, 42. [CrossRef]
19. Gámiz, M.L.; Navas-Gómez, F.; Raya-Miranda, R. A machine learning algorithm for reliability analysis. *IEEE Trans. Reliab.* **2021**, *70*, 535–546. [CrossRef]
20. Afanaseva, O.; Afanasyev, M.; Neyrus, S.; Pervukhin, D.; Tukeev, D. Information and Analytical System Monitoring and Assessment of the Water Bodies State in the Mineral Resources Complex. *Inventions* **2024**, *9*, 115. [CrossRef]
21. Kozlowski, E.; Mazurkiewicz, D.; Sep, J.; Zabinski, T. The use of principal component analysis and logistic regression for cutter state identification. In Proceedings of the International Conference on Engineering, Technology and Innovation, Kuala Lumpur, Malaysia, 19–20 October 2021; pp. 396–405.
22. Zuo, J.; Gan, H.; Li, H. A study of ancient glass classification problem based on multiple logistic regression. *Highlights Sci. Eng. Technol.*, **2022**, *22*, 265–269. [CrossRef]
23. Meyer, M.C. Semi-parametric additive constrained regression. *J. Nonparametric Stat.* **2013**, *25*, 715–730. [CrossRef]
24. Li, X.; Ding, Q.; Sun, J.Q. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliab. Eng. Syst. Saf.* **2018**, *172*, 1–11. [CrossRef]
25. Herzog, MA.; Marwala, T.; Heyns, P.S. Machine and component residual life estimation through the application of neural networks. *Reliab. Eng. Syst. Saf.* **2009**, *94*, 479–489. [CrossRef]
26. Liu, J.; Wu, Q.; Sui, X.; Chen, Q.; Gu, G.; Wang, L.; Li, S. Research progress in optical neural networks: Theory, applications and developments. *PhotoniX* **2021**, *2*, 5. [CrossRef]

27. Singh, Y.; Saini, M.; Savita. Impact and performance analysis of various activation functions for classification problems. In Proceedings of the IEEE International Conference on Contemporary Computing and Communications (InC4), Bangalore, India, 21–22 April 2023; pp. 1–7.
28. Reyad, M.; Sarhan, A.; Arafa, M. A modified Adam algorithm for deep neural network optimization. *Neural Comput. Appl.* **2023**, *35*, 17095–17112. [CrossRef]
29. Qiu, Y.; Li, Z. Neural network-based approach for failure and life prediction of electronic components under accelerated life stress. *Electronics* **2024**, *13*, 1512. [CrossRef]
30. Guo, K.; Yan, H.; Huang, D.; Yan, X. Active learning-based KNN-Monte Carlo simulation on the probabilistic fracture assessment of cracked structures. *Int. J. Fatigue* **2022**, *154*, 106533. [CrossRef]
31. Ghosh, A.K. On optimum choice of k in nearest neighbor classification. *Comput. Stat. Data Anal.* **2006**, *50*, 3113–3123. [CrossRef]
32. Li, J.; Zhang, J.; Zhang, J.; Zhang, S. Quantum KNN classification with K value selection and neighbor selection. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2024**, *43*, 1332–1345. [CrossRef]
33. Li, Y.; Yang, Y.; Che, J.; Zhang, L. Predicting the number of nearest neighbor for kNN classifier. *IAENG Int. J. Comput. Sci.* **2019**, *46*, 662–669.
34. Tiwari, S.P.; Manohar, M.; Shukla, S.K. A reliable protection scheme for high resistance fault detection in wind generator-integrated HVDC transmission system using ensemble of kNN. In *Electrical Engineering*; Springer: New York, NY, USA, 2024.
35. Abu Alfeilat, H.A.; Hassanat, A.B.A.; Lasassmeh, O.; Tarawneh, A.S.; Alhasanat, M.B.; Eyal Salman, H.S.; Prasath, V.B.S. Effects of distance measure choice on K-Nearest Neighbor classifier performance: A review. *Big Data* **2019**, *7*, 221–248. [CrossRef]
36. Puggini, L.; Doyle, J.; McLoone, S. Fault detection using Random Forest similarity distance. *IFAC Pap. Online* **2015**, *48*, 583–588. [CrossRef]
37. Kizito, R.; Scruggs, P.; Li, X.; Kress, R.; Devinney, M.; Berg, T. The application of Random Forest to predictive maintenance. In Proceedings of the IISE Annual Conference, Orlando, FL, USA, 9–22 May 2018.
38. Makota, D.T.; Shililiandumi, N.; Iddi, H.U.; Bagile, B.B. A big data-based ensemble for fault prediction in electrical secondary distribution network. *Cogent Eng.* **2024**, *11*, 2340183. [CrossRef]
39. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32.:1010933404324. [CrossRef]
40. Payette, M.; Abdul-Nour, G. Machine learning applications for reliability engineering: A review. *Sustainability* **2023**, *15*, 6270. [CrossRef]
41. Alagarsamy, P. Predict Pump Failure Before It Happens Using Deep Learning Model. 2021. Available online: https://becominghuman.ai/predict-pump-failure-before-it-happens-using-deep-learning-model-dc886bfa073e (accessed on 15 July 2024).
42. Revelle, W. psych: Procedures for Psychological, Psychometric, and Personality Research. *R Package Version* **2024**, *2*, 9.
43. Helwig, N.; Pignanelli, E.; Schütze, A. Condition monitoring of a complex hydraulic system using multivariate statistics. In Proceedings of the IEEE International Instrumentation and Measurement Technology Conference, Pisa, Italy, 11–14 May 2015; paper PPS1-39.
44. Helwig, N.; Schütze, A. Detecting and compensating sensor faults in a hydraulic condition monitoring system. In Proceedings of the 17th International Conference on Sensors and Measurement Technology, Nuremberg, Germany, 19–21 May 2015.
45. Schneider, T.; Helwig, N.; Schütze, A. Automatic feature extraction and selection for classification of cyclical time series data. *Tech. Mess.* **2017**, *84*, 198–206. [CrossRef]
46. Bansal, A.; Sharma, R.; Kathuria, M. A systematic review on data scarcity problem in deep learning: Solution and applications. *ACM Comput. Surv.* **2022**, *54*, 1–29. [CrossRef]
47. Mahmood, R.; Lucas, J.; Álvarez, J.M.; Fidler, S.; Law, M.T. Optimizing data collection for Machine Learning. *arXiv* **2022**, arXiv:2210.01234.
48. Hestness, J.; Narang, S.; Ardalani, N.; Diamos, G.F.; Jun, H.; Kianinejad, H.; Patwary, M.A.; Yang, Y.; Zhou, Y. Deep Learning scaling in predictable, empirically. *arXiv* **2017**, arXiv:1712.00409.
49. Calvo-Pardo, H.F.; Mancini, T.; Olmo, J. Optimal deep neural networks by maximization of the approximation power. *Comput. Oper. Res.* **2023**, *156*, 106264. [CrossRef]
50. Zou, X.; Wang, K.; Lu, J.; Wu, D. Time series forecasting of emission trends using recurrent neural networks. *Comput. Life* **2024**, *12*, 12–18. [CrossRef]
51. Rani, S.; Kaur, R.; Desai, C. Enhancing time series forecasting accuracy with deep learning models: A comparative study. *Int. J. Adv. Res.* **2024**, *12*, 315–324. [CrossRef]