Instituto Tecnológico
y de Estudios Superiores de Occidente

Official recognition of higher education studies granted under ministerial agreement 15018, published in the Official Gazette of the Federation on November 29, 1976.

Department of Electronics, Systems and Informatics

Master's in Computer Systems

HARD DRIVE FAILURE PREDICTION THROUGH A HYBRID MACHINE LEARNING FRAMEWORK: INTEGRATION OF LSTM AND DECISION TREES WITH SMART ATTRIBUTES

DEGREE-QUALIFYING PROJECT submitted to obtain the **DEGREE** of **MASTER IN COMPUTER SYSTEMS**

Submitted by: B.Eng. Iñaki Sebastián Orozco García

Degree-qualifying Project Director: Dr. Leobardo Emmanuel Campos Macías

Tlaquepaque, Jalisco. 28 de agosto de 2025

Instituto Tecnológico y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

Maestría en Sistemas Computacionales

PREDICCIÓN DE FALLOS EN DISCOS DUROS MEDIANTE UN MARCO HÍBRIDO DE APRENDIZAJE AUTOMÁTICO: INTEGRACIÓN DE LSTM Y ÁRBOLES DE DECISIÓN CON ATRIBUTOS SMART

TRABAJO RECEPCIONAL que para obtener el **GRADO** de **MAESTRO EN SISTEMAS COMPUTACIONALES**

Presenta: Ing. Iñaki Sebastián Orozco García

Director del trabajo de obtención de grado: Dr. Leobardo Emmanuel Campos Macías

Tlaquepaque, Jalisco. 28 de agosto de 2025

AGRADECIMIENTOS

El autor quiere agradecer a quienes hicieron posible este trabajo:

Al Instituto Tecnológico de Estudios Superiores de Occidente, mi alma mater, guiada por los valores jesuitas de servicio, integridad y búsqueda de la verdad. Mi paso por esta institución ha sido una transformación personal, convirtiendo lo aprendido en acciones significativas para el bienestar de la sociedad.

A Intel, por apoyar mi desarrollo profesional y al mismo tiempo impulsar mis aspiraciones académicas, brindándome un proyecto útil, así como los recursos y la orientación necesarios para llevarlo a cabo.

A mis padres, Humberto Orozco y Alejandra García, por siempre apoyarme y darme una vida por la que trabajaron arduamente. Mi deseo de mejorar no es solo por mi propio camino, sino para poder ofrecer más a quienes me rodean, porque su ejemplo me inspira todos los días.

A mi colega Edgar Delgadillo, por tu idea que dio origen a este proyecto, por tu tiempo, esfuerzo, conocimientos y disposición para ayudar. Estoy profundamente agradecido por tu presencia en cada etapa.

A mi director del TOG, Leobardo, por tu ayuda, guía y apoyo constante. Tus comentarios y tu impulso han sido fundamentales en la evolución de este trabajo. Te agradezco sinceramente tu mentoría.

A todos mis profesores en esta maestría, su orientación a lo largo de cada curso me ha ayudado a crecer y me ha brindado el conocimiento y las herramientas necesarias para crear y desarrollar este TOG. Gracias por ser una parte tan esencial de este camino.

A todos mis profesores de licenciatura, quienes desempeñaron un papel esencial en la construcción de los cimientos de mi formación académica. Este trabajo es también resultado de su influencia, y les reconozco como parte fundamental de este logro.

ACKNOWLEDGMENTS

The author wants to thank those who made this work possible:

To the Instituto Tecnológico de Estudios Superiores de Occidente, my alma mater, guided by the Jesuit values of service, integrity and pursuit of truth, my journey here was about transforming my learnings into meaningful actions for the improvement of society.

To Intel for supporting my professional development and also encouraging my academic ambitions providing a useful project and the resources and guidance to complete it.

To my parents Humberto Orozco and Alejandra García for always supporting me and giving me a life that they had to work hard to get. I seek to improve, not only for my own jorney, but soo I can give more to those around me, because your example inspires me every day.

To my colleague Edgar Delgadillo, for your idea that grew into this project, for your time, effort, insights and willingness to help, I am sincerely grateful for your presence in every step.

To my DQP director Leobardo Campos, for your help, guidance and support. Your insights and encouragement have been instrumental to the evolution of this project. I am very grateful for your mentorship.

To all my professors in this master's, your guidance throughout each course have helped me grow and has given me the knowledge and tools to create and evolve this DQP. Thank you for being such an integral part of this journey.

To all my undergraduate professors, who played an essential role in laying the foundations of my academic development. This work is also a result of their influence, and I acknowledge them as a fundamental part of this achievement.

SUMMARY

The potential for storage failure is a pressing concern for any technology user, as it can lead to the irreversible loss of valuable information and productivity. This DQP presents a hybrid Machine Learning (ML) framework for hard drive failure prediction through the integration of time series forecasting and binary classification. The work addresses a critical challenge in storage reliability: anticipating drive failures before data loss occurs by using Self-Monitoring, Analysis, and Reporting Technology (SMART) attributes.

This work was conducted in collaboration with Intel to develop a suitable proprietary model for predicting hard drive failures, with the goal of integrating it into various software solutions. The dataset used is open-source and provided by Backblaze, a cloud storage company that monitors drive performance across its servers and publishes this data for free public use.

The research employs a systematic methodology consisting of three primary components. First, comprehensive data preprocessing techniques are applied to the Backblaze Self-Monitoring, Analysis, and Reporting Technology (SMART) dataset, including robust missing value imputation, feature selection through correlation analysis, and Synthetic Minority Over-sampling Technique (SMOTE) to address the inherent class imbalance between operational and failing drives. Second, a Long Short-Term Memory (LSTM) neural network is developed to capture temporal dependencies in SMART attribute sequences, enabling forecasting of future values based on historical observations. Third, a Decision Tree (DT) classifier is constructed to interpret these forecasted attributes and provide binary failure predictions with interpretable decision boundaries.

The experimental results demonstrate that while individual components achieve respectable performance metrics (82.4 % accuracy for the DT on balanced training data), the integrated pipeline faces challenges with real world data, achieving 63.2 % accuracy on the test set. The research provides critical insights into the practical limitations of current approaches, particularly regarding temporal data requirements and classification performance with extreme class imbalances.

This work contributes to the field of predictive maintenance by establishing a foundation for combining deep learning based temporal forecasting with explainable classification models. The findings inform future research directions in storage reliability prediction and offer practical guidance for implementing ML hybrid solutions.

RESUMEN

El potencial de fallas en los sistemas de almacenamiento es un problema para cualquier usuario que usa tecnología, este problema puede provocar la pérdida irreversible de información valiosa y mermar la productividad. Este TOG presenta un marco híbrido de aprendizaje automático para la predicción de fallos en discos duros, integrando proyección de datos de series temporales y clasificación binaria. El trabajo aborda un desafío crítico en la confiabilidad del almacenamiento: anticipar fallos antes de que ocurra pérdida de datos utilizando atributos Self-Monitoring, Analysis, and Reporting Technology (SMART). La investigación emplea una metodología sistemática compuesta por tres componentes principales.

Este trabajo se realizó en colaboración con Intel para desarrollar un modelo propio, adecuado para predecir fallos en discos duros. Esto con el objetivo de integrarlo en diversas soluciones de software. El conjunto de datos utilizado es de código abierto y proporcionado por Backblaze, una empresa de almacenamiento de datos en la nube que monitorea el rendimiento de los discos duros que poseen en sus servidores y publica estos datos para su uso público gratuito.

En primer lugar, se aplican técnicas de preprocesamiento exhaustivas al conjunto de datos de Self-Monitoring, Analysis, and Reporting Technology de Backblaze, incluyendo imputación de valores faltantes, selección de características basada en análisis de correlación, y la técnica de sobre muestreo sintético de la clase minoritaria (Synthetic Minority Oversampling Technique (SMOTE)) para abordar el desequilibrio entre registros de discos operativos y fallidos. En segundo lugar, se desarrolla una red neuronal de Long Short-Term Memory (LSTM) para capturar dependencias temporales en las secuencias de atributos SMART, permitiendo predecir valores futuros a partir de observaciones históricas. En tercer lugar, se construye un clasificador basado en Decision Tree (DT) para interpretar los atributos pronosticados y proporcionar predicciones binarias de fallo con fronteras de decisión interpretables.

Los resultados experimentales muestran que, aunque los componentes individuales alcanzan métricas de rendimiento aceptables (82.4 % de precisión para el DT en datos balanceados), la canalización integrada enfrenta desafíos con la escasez de datos reales, obteniendo un 63.2 % de precisión en el conjunto de prueba. El estudio ofrece una visión crítica sobre las limitaciones prácticas de los enfoques actuales, especialmente en cuanto a los requerimientos de datos temporales y el rendimiento de clasificación bajo desequilibrios extremos de clase.

Este trabajo contribuye al campo del mantenimiento predictivo al establecer una base metodológica para combinar pronóstico temporal mediante aprendizaje profundo con modelos explicables de clasificación. Los hallazgos orientan futuras investigaciones sobre predicción de confiabilidad en almacenamiento y ofrecen directrices prácticas para implementar soluciones de machine learning en entornos de centros de datos.

TABLE OF CONTENTS

FIGURE LIST

TABLE LIST

LIST OF ACRONYMS AND ABBREVIATIONS

AI Artificial Intelligence. 27

ANN Artificial Neural Network. 29

BPTT Backpropagation Through Time. 29

CNN Convolutional Neural Network. 22, 23

CT Classification Tree. 28

DRAM Dynamic Random Access Memory. 30

DT Decision Tree. 3, 4, 11, 14, 17–19, 22, 27, 29, 32–38, 41–43, 45, 46, 48, 54

FCN Fully Convolutional Network. 22, 24

FFIP Functional Failure Identification and Propagation. 22

FFT Fast Fourier Transform. 22

GSPN Generalized Stochastic Petri Net. 22

HDD Hard Disk Drive. 22

LSTM Long Short-Term Memory. 3, 4, 14, 17–19, 22, 24, 29, 30, 33–35, 37, 38, 40, 42, 43, 45, 46, 54

ML Machine Learning. 3, 14, 16, 17, 27

NVMe Non-Volatile Memory Express. 24

PCA Principal Component Analysis. 22

RNN Recurrent Neural Network. 29

SMART Self-Monitoring, Analysis, and Reporting Technology. 3, 4, 11, 14, 16–19, 21, 23, 28, 30, 34, 35, 37, 38, 42, 43, 46

SMOTE Synthetic Minority Over-sampling Technique. 3, 4, 6, 17, 18, 28, 29, 42

SSD Solid State Drive. 24

1. INTRODUCTION

The potential for storage failure is a pressing concern for any technology user, as it can lead to the irreversible loss of valuable information and productivity. Since the invention of computers, considerable efforts have been made to extend the lifespan of electronic devices or to predict their failures in order to reduce operational distruptions. In today's digitally interconnected world, data is essential for the core functioning of a system and drive problems may lead to permanent information lost.

The need for reliable systems has made hard drive failure prediction an important and crucial area of research aiming at minimize data loss and reducing system downtime. This approach at hardware failure prevention started when engineers started using data fed by sensors like temperature, voltage, read/write errors to detect signs of degradation. This practices evolved into predictive maintenance where the goal is to anticipate failure caused by multiple motives before they happen, this minimize downtime and repair costs.

While existing predictive models have achieved promising results, there remains room for improvement. Recent advances in Machine Learning (ML) have introduced novel techniques that enhance the accuracy and interpretability of failure prediction systems. Among these techniques, Long Short-Term Memory (LSTM) networks have demonstrated strong capabilities for capturing patterns in Self-Monitoring, Analysis, and Reporting Technology (SMART) information because they excel when trained with time series data. Complementing the Long Short-Term Memory (LSTM) models, classification algorithms like Decision Tree (DT) can categorize data into distinct states, in the case regarding hard drive information they can classify into a failure category.

This study takes advantage from a pre labeled, large scale SMART dataset provided by a cloud services company called Backblaze [1], containing thousands of records from real life, working hard drives present in their clusters. They pre labeled the data with a failure column indicating by 0 a working drive and 1 a drive that failed the next day. To solve the drive failure problem I propose a robust modeling pipeline that integrates LSTM data forecasting with DT classification, capable of predicting hard drive failures and classifying

devices based on their projected operational status.

1.1. Justification

The prediction of hard drive failures is critical for maintaining system reliability and operational continuity. It plays a vital role in preserving data integrity, minimizing system downtime, and reducing both financial and logistical risks. Hard drives often exhibit elements that identify malfunction long before total failure occurs, enabling early detection through continuous monitoring. Thinking ahead makes it easier to step in at the right time, whether that means backing up your files or swapping out faulty hardware.

In enterprise environments, predictive capabilities help prevent service disruptions and safeguard critical operations. On a personal scale, failure prediction helps preserve irreplaceable digital assets. By training ML models on historical SMART data and integrating advanced monitoring tools, users and organizations can transition from reactive crisis management to a proactive strategy.

This pipeline may be able to evolve and adapt to modern storage technologies that have been gaining popularity like Solid State Drives (SSD) where research and development is very limited compared to Hard Drive Disks (HDD).

Furthermore, this work provides an opportunity to consolidate and apply key competencies acquired throughout the master's program. It integrates preprocessing techniques, supervised classification models, and deep learning architectures, into a cohesive pipeline. By combining theoretical foundations with practical implementation, this research reflects the culmination of academic training and the ability to translate classroom concepts into effective real world solutions.

1.2. Problem

Despite the critical importance of data integrity in modern digital systems, there is currently no reliable solution for accurately predicting hard disk drive (HDD) failures. Existing tools are either limited by inconsistent performance, lack of generability, or dependence on proprietary parameters that conflict with widespread adoption. Furthermore, with the rapid evolution of storage technology, especially the rapid transition to NVMe-based solid-state drives (SSDs), this predictive gap becomes more significant. There is a clear absence of a unified methodology that can extend seamlessly from traditional HDD environments to modern NVMe systems, creating challenges for implementing proactive maintenance strategies.

Without a dependable mechanism for anticipating drive failures, organizations and users face increased risk of irretrievable data loss and costly downtime. This problem is exacerbated in enterprise and research environments where system uptime and data availability are critical. Therefore, developing a predictive framework rooted in machine learning and

adaptable across storage technologies represents a vital opportunity to enhance reliability and extend device lifespan.

1.3. Hypothesis

Based on the analysis of existing literature and the characteristics of SMART attribute datasets, this research proposes a hypothesis that addresses both temporal modeling and classification capabilities for hard drive failure prediction:

Hypothesis: A hybrid ML approach combining Long Short-Term Memory (LSTM) networks for time series forecasting with DT classification can achieve useful results in predicting hard drive failures compared to individual model approaches, by leveraging both temporal dependencies in SMART data and interpretable classification rules 1.1.

This hypothesis is supported by two elements:

1. Time Series Modeling Capability A properly configured LSTM neural network can be trained to achieve good performance on hard drive SMART attribute time series data, effectively learning temporal patterns and dependencies that characterize the progression toward drive failure. The sequential nature of SMART data, collected daily over the operational lifetime of drives, contains sufficient temporal information to enable accurate forecasting of future attribute values.

2. Binary Classification Capability Hard drive data can be effectively classified into two distinct categories: failed and non-failed drives, using ML classification algorithms applied to SMART attributes. Despite the inherent class imbalance in failure datasets (where failed drives represent a small minority), appropriate preprocessing techniques such as Synthetic Minority Over-sampling Technique (SMOTE) can enable robust binary classification performance.

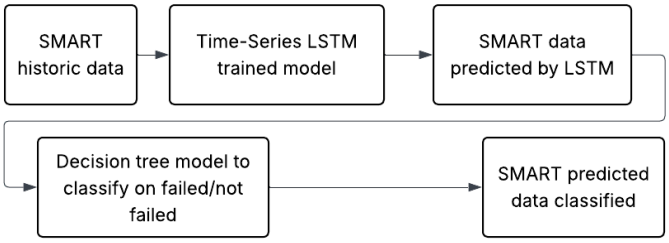

Figura 1.1: Pipeline integration

The validation of this hypothesis would demonstrate that:

1. SMART attribute time series contain learnable temporal patterns indicative of fu-

ture drive health

2. Classification algorithms can distinguish between healthy and degrading drives when provided with appropriate features

3. The hybrid approach provides both predictive capability and interpretability required for practical deployment

1.4. Objectives

1.4.1. General Objective

The objective of this DQP is to test the effectivenes and accuracy of a hybrid approach aimed at hard drive failure prediction, this will be achieved using data pre-processing to improve the information to train the models, modern techniques and deep learning models. By analyzing the extensive Backblaze SMART dataset, this study seeks to clean and improve the data to use on an Long Short-Term Memory (LSTM) model and Decision Tree (DT) with Synthetic Minority Over-sampling Technique (SMOTE) to improve the performance and offer an alternative to existing solution and, if possible, get better results or an alternative to existing methods.

1.4.2. Specific Objective

The objectives of this project are defined as follows:

1. **Preprocess the Backblaze dataset** to improve its quality, with an emphasis on:
   - Data refinement through cleaning and normalization.
   - Handling missing values and inconsistent SMART attributes.
   - Implementing oversampling techniques of failed drive instances to mitigate class imbalance and improve model performance.
2. **Develop a supervised classification model** based on the DT algorithm with the goal of:
   - Accurately predicting whether a hard drive data belongs to a failed or not failed element.
   - Consuming a curated subset of SMART features as input.
   - Delivering robust classification performance despite the inherent class imbalance caused by the rarity of failure events.

3. **Design and implement a forecasting model** using a Long Short-Term Memory (LSTM) neural network. This model aims to:
    - Learn temporal patterns from historical SMART time series data.
    - Predict future SMART attribute values over a defined time window.
    - Generate input sequences that can be effectively processed by the DT model to classify drives as failed or operational.
4. **Integrate the LSTM/DT pipeline**, ensuring that:
    - Forecasted SMART metrics are fed seamlessly into the DT.
    - The combined framework outputs a classification result using real world data.

1.5. Scientific or technological innovation or contribution

This DQP contributes a hybrid predictive modeling framework that offers Intel an alternative approach for hard drive failure detection, combining temporal forecasting and binary classification in a single pipeline. The resulting model requires minimal input features and is capable of forecasting the future behavior of hard drives while determining whether a device is prone to failure. Its modular design supports integration into existing operational environments.

The development of a working pipeline opens the possibility on implementing prediction work into the untouched field of NVME failure prediction. The benefits of this work can be used in various software integrations for different projects in Intel adding value to them.

2. STATE OF THE ART

The prediction of hard drive failures has long been a topic of interest in the field of system reliability. Initial methods relied on simple probabilistic models. Over time, prediction techniques have evolved to use various methods from machine learning to deep learning and integrating multiple solutions into one to improve the performance.

2.1. Early and Classical Approach

Some of the earliest approaches focused on straightforward algorithms made to predict failure in a range of hardware components. For example the article [2] Improved Disk-Drive Failure Warnings from 2002 specialized on drives using the Rank Sum test, a method used for rare events where false alarms are very costly, this paper uses early SMART attributes that make sense being tied to Hard Drive degradation, they are shown in the table 2.1. They propose rank sum statistical tests to replace threshold test and improve failure warning accuracy while lowering false alarm rates. They compare recent data from the drive with original data taken from the the drive during manufacture.

Symbol	**Relevance**	**Key**
POH	Drive aging	C
CSS	Drive power cycles	C, F
SKE	Heads seek to wrong track	I, T
RRT	Drive re initializes	I
RSE	Errors corrected by inner ECC code	I, F, T
MSE	Precursor of RSE	I
GDC	New disk defects, found after manufacture	C, F, T
SUT	Power on to drive ready	I, F
TMR	Head track misregistry	I
GMX	Mechanical shock	I
TAS	Thermal asperity count	I, F
FLY	PW50, Wallace, read IC MSE, FIR taps	I
TMP	Drive max temperature limit	I
DCL	Read IC operational status	I
SPN	Spin & Track Motors: Acoustics, start current, control loop analysis	I, T

Tabla 2.1: SMART Attributes and Their Relevance

Key	**Description**
C	Cumulative measurement
I	Incremental
F	Head/disk interface indicative
T	Track servo indicative

Tabla 2.2: Key Definitions

Another early work is [3] BlueGene/L Failure Analysis and Prediction Models from 2006 used a simple prediction algorithm across multiple hardware problems. These works set the stage and showcase that classical statistical methods can be useful if carefully chosen to provide insight and early warnings in preventing hardware failure.

2.2. Hardware/Software Reliability Prediction

On 2019 a survey was published [4] reviewing the different methods used to ensure system reliability, exploring hardware driven software failures (where hardware degradation impacts software) and software driven hardware failures (where software malfunctions cause hardware failure). The approaches listed in the survey use multiple models including Markovian models, Generalized Stochastic Petri Net (GSPN), Monte Carlo simulations, and Functional Failure Identification and Propagation (FFIP). Some of the models also integrate machine learning techniques such as DTs for fault detection.

2.3. Signal Processing and Deep Learning

With more data availability, the researchers of the paper Machine Learning Based Approach for Hardware Faults Prediction [5] advanced toward methods that combine traditional signal processing and emerging deep learning techniques. Using Fast Fourier Transform (FFT) and Principal Component Analysis (PCA), combined with Convolutional Neural Network (CNN) they aimed to address circuit failures in hardware.

2.4. Advanced Time Series Modeling

Hard drive failure prediction has also benefited from the creation of models aimed at imbalanced and temporal data. The 2021 study on [6]"Hardware Failure Prediction on Imbalanced Time Series Data" found that the combination of Long Short-Term Memory (LSTM) networks with Fully Convolutional Network (FCN)—forming the LSTMFCN architecture—to outperform other deep models such as time convolutional neural networks and residual networks. This hybrid model removes the need to capture both long term dependencies and local feature variations in the time series data of hardware, this capability is crucial for anticipating subtle signs of degradation in hard drives.

2.5. Data centers and cloud

Expanding from individual components to larger systems, several studies have experimented in system level failure prediction, they experimented integrating both hardware and software monitoring. One study of 2019 [7] found that server breakdonw failure causes are led by HDD failure as shown in the Table 2.3

This showcase the importance in hard drive prediction models for mitigating server break-

Tabla 2.3: Failure types causing the server breakdown

Section	**Proportion (%)**
HDD	81.84
Miscellaneous	10.20
Memory	3.06
Power	1.74
RAID card	1.23
Others	1.93

downs. They propose a method for prediction using a temporal CNN model, the model is fed with One hot encoding fusing static information and dynamic time series features of drives. The SMART attributes selected to train this CNN model are depicted in the table 2.4

Tabla 2.4: Selected SMART features

ID#	**Attribute Name**	**Rank**
4	Start Stop Count	8
5	Reallocated Sector Count	3
10	Spin Retry Count	13
12	Power Cycle Count	6
183	Runtime Bad Block	7
184	End To End Error	10
187	Reported Uncorrect	2
189	High Fly Writes	4
192	Power Off Retract	5
193	Load/Unload Cycle Count	9
196	Reallocation Events Count	12
197	Current Pending Sector Count	1
199	Ultra ATA CRC Error Rate	11

Similarly, the 2020 work on [8] "Task Failure Prediction in Cloud Data Centers Using Deep Learning" compared approaches like B LSTM, Hidden Semi Markov Models (HSMM), Support Vector Machines (SVM), and RNNs. This study demonstrated how supporting traditional Markov models with more robust machine learning techniques could overcome the limitations of predicting failure solely based on nearby state, this challenges are common in high dimensional cloud environments.

2.6. Specialized and Emerging Approaches for Predicting Failure

More recent efforts have increasingly focused in using neural networks to improve prediction accuracy, Nadine, Lea and Andreas [6] implemented a fully convolutional neural network combined with an LSTM to improve both models augmenting the FCN with the LSTM using a softmax layer at the end.

2.7. Concluding Thoughts

The trajectory of hard drive failure prediction demonstrates a clear evolution from simple algorithms to sophisticated pipelines that combine hybrid models and effectively leverage both traditional statistical tests and deep learning architectures. This integration of temporal analysis and feature engineeringhas been a crucial part in developing models that can anticipate failures with great accuracy. The adoption of new technology to store data such as Solid State Drive (SSD) with Non-Volatile Memory Express (NVMe) protocols also opens the door for the existing Hard Drive models to be adapted or evolve to help to mitigate the problems introduced by the change.

Although significant progress has been made in the development of predictive models for hard drive failures, many existing approaches still exhibit limited accuracy and inconsistent generalizability. Traditional classifiers often struggle with imbalanced datasets and fail to capture the temporal dynamics of SMART parameters. Even more recent methods based on deep learning frequently suffer from overfitting or inadequate feature engineering, leading to suboptimal performance in real world scenarios. These limitations reveal clear room for improvement and underscore the need for robust, adaptable models capable of delivering reliable predictions across evolving storage technologies.

3. THEORETICAL FRAMEWORK

This chapter outlines the core concepts and methodologies that underpin the development of predictive models for hard drive failure detection. It reviews foundational principles of machine learning, focusing on classification techniques and time series models, which capture temporal patterns in SMART data. By examining the theoretical basis and operational mechanisms of these models, this section establishes the groundwork necessary to understand their application within a hybrid predictive architecture.

3.1. Data Preprocessing

3.1.1. The Role of Preprocessing

The set of techniques used to adapt the data aimed to be consumed by algorithms and models is named data preprocessing, this techniques exist because data is imperfect, inconsistent and redundant and it can't be consumed directly from the start, the bigger the amount of data to be used the sophisticated the mechanisms to use it. [9]

3.1.2. Feature selection

Feature selection is the process of identifying and removing irrelevant information, doing this helps obtaining a subset of the original features that can still describe it. [9] Removing this unnecesary set of features can help leave out accidental correlations in the algorithms that can affect the performance and results in the models.

3.1.2.1. Normalization and Standardization

Normalization is the process to unify the scale or range of the data, the unit used to measure the data can affect the analysis, when normalizing the data it should be aimed at giving all attributes equal weight. [10]

3.1.2.2. Handling Missing Values

Normally datasets have incomplete data, this occurs for a multitude of reasons and can't be avoided. Treating them is difficult and if not handled correctly they can led to poor knowledge about the data and achieving wrong conclusions. There are several different approaches to solve the problem of missing values, the first option is to discard instances with missing values but this is rarely used because it can produce a bias in the learning process.[9] The most well known method to handle missing values is with data imputation, this method consist in fill the missing values with estimated ones, there are a multitude of solutions for imputation and they depend on the characteristics of the data. [10]

3.1.2.3. Feature Extraction

Feature extraction techniques combine the original set of features and obtain a new set of variables less redundant. [9] Substets of attributes can be merged or can contribute

to the creation of artificial substitute attributes, this can represent better the decision boundaries in supervised learning. [10]

3.2. Fundamentals of Machine Learning

3.2.1. General Concepts

Machine Learning (ML) is a subfield of Artificial Intelligence (AI), involving the development of algorithms that improve their performance through experience, rather than relying on explicit programming. It is built upon principles from statistics and computer science to construct models capable of learning from data. These models are commonly applied across diverse domains such as speech recognition, image classification, and text analysis in data classification tasks. ML plays a pivotal role in predictive data mining, where systems learn patterns from training data to make predictions about future events. [11, págs. 1-4]

3.2.2. Supervised Learning

Supervised learning is the process of a machine learning algorithm in which the training method for the algorithm comes from a labeled dataset, the data being used to train the algorithm is associated with an output label. Graves [12, págs. 5-6] explains that there are also extremes to the complexity of the supervision provided by the targets to train the models, often being referred to as weakly and strongly labeled data. The goal of this approach is to construct a model that can accurately map new data to the appropriate label. The learning process attemps to adjust the algorithm using a function that minimizes the error on the test data with the labels.

3.2.2.1. DT Algorithms

Decision Tree (DT) are one of the most widely used supervised classification techniques in machine learning and data mining. They can be used to make assumptions regarding categorical class names, to classify knowledge on the basis of training sets and class labels, and to classify newly obtainable data. They also provide a hierarchical model in which nodes represent decisions based on attribute values, and branches lead to outcomes or further decision nodes. A typical DT structure includes root nodes, internal decision nodes, and terminal leaf nodes. [13, pág. 21]

They explain that Entropy and information gain are crucial metrics used in splitting nodes within a DT. Entropy measures impurity in a dataset, and information gain quantifies

how much knowledge a specific feature provides about the target variable. The higher the information gain, the more desirable the split.

3.2.2.2. Classification Trees

Classification Tree (CT), a form of decision tree algorithm, are among the most widely utilized models in supervised machine learning, known for their capacity to generate interpretable classification rules from multidimensional data (Tijo & Abdulazeez, 2021)[13]. In the classification matter that hard drive failure needs, CTs offer a clear and logical means of transforming SMART telemetry into clear classification of failed or not.

Each CT is composed of a root node, decision nodes, and leaf nodes, forming a hierarchical structure optimized to classify inputs into categories like failed or healthy. Their architecture supports the heterogeneous problem of hard drive datasets, which may include numeric and categorical features.

CT's have a diverse ranges of tasks, including [13]:

- Medical diagnosis (e.g., cancer detection)
- Intrusion detection
- Image classification
- Behavioral modeling

3.2.2.3. Synthetic Minority Over-sampling Technique (SMOTE)

If you have a minority class in a dataset you have various options, one of them is to balance it using Synthetic Minority Over-sampling Technique (SMOTE), originally proposed by Chawla et al. (2002)[14]. SMOTE operates by generating synthetic samples of the minority class, rather than merely duplicating existing instances. These artificial data points are interpolated along the line segments connecting a minority class sample with its nearest neighbors in feature space. This process expands the decision regions associated with the minority class, reducing the tendency of classifiers to overfit and enhancing generalization capabilities.

The implementation of SMOTE in this study involves the following steps:

- Identification of the minority class (i.e., failed drives).
- Computation of k nearest neighbors for each minority instance.

- Generation of synthetic samples by introducing randomized interpolations between feature vectors.

By applying SMOTE prior to model training, the classifier receives a more balanced and diverse dataset to be trained with. Empirical evidence from Chawla et al.[14] demonstrates that SMOTE, significantly improves classifier performance across various domains. Its integration into this DQP supports the development of a more accurate DT model.

3.3. Fundamentals of Deep Learning

3.3.1. Artificial Neural Networks

Artificial Neural Network (ANN) were originally developed to imitate the processing capabilities of biological brains. The basic structure of them is a small processing unit or node, joined to other node by a connection with weight. The training of ANNs typically involves gradient descent and the backpropagation algorithm to minimize a loss function, such as cross entropy for classification tasks. [12]

3.3.2. Recurrent Neural Networks

Recurrent Neural Network (RNN) extend ANNs by introducing recurrent connections, enabling them to process sequences and retain context over time. Each RNN unit receives not only the current input but also its own state from the previous timestep, giving it a form of internal memory.

The training of RNNs typically employs Backpropagation Through Time (BPTT), an unfolded version of the network that propagates error gradients along the temporal dimension. However, standard RNNs suffer from the vanishing gradient problem, which limits their ability to learn long range dependencies. [12]

3.3.2.1. Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a special RNN architecture designed to overcome the vanishing gradient problem. An LSTM cell includes gates: input, output, and forget gates that regulate information flow. The inclusion of memory cells allows LSTM networks to store, update, and retrieve information over long sequences. [12]

LSTM has been applied successfully in tasks requiring extended memory, such as speech, handwriting recognition, time series data.

3.3.3. Time Series Data and LSTM

In system level hardware monitoring, time series data is central to understanding the progression toward failure. Devices like hard drives and DRAM generate daily records of SMART attributes, system logs, and runtime statistics storing sudden anomalies [7]. These logs form sequential datasets, with new data points being added continually until a device fails.

The data normally includes:

- Discrete data at regular intervals (daily/hourly)
- Non stationarity, reflecting changes in device behavior over time
- Very few failed samples, as failures are rare events
- Vendor specific attribute encoding, adding heterogeneity

These properties pose challenges for traditional machine learning models, which often rely on static snapshots. Instead, sequential models that can capture temporal dependencies such as LSTM networks and are better suited for this task.

4. METHODOLOGICAL DEVELOPMENT

This chapter details the implementation process of a hybrid predictive pipeline designed to anticipate hard drive failures. The methodology integrates two complementary machine learning components: a Long Short-Term Memory (LSTM) network for time-series prediction and a Decision Tree model for final classification. This chapter establishes a transparent and reproducible framework for understanding how temporal patterns and decision boundaries combine to enable preventive storage monitoring.

4.1. Requirements Gathering

This research adopts a **mixed** approach, of an **exploratory** and **applied** nature, with the objective of evaluating artificial intelligence models for predicting hard drive failures based on SMART attributes, it involves different preprocessing methods, algorithms to retrieve useful information and machine learning and deep learning methods.

4.1.1. Methodological Design

Two supervised machine learning models were implemented based on the most relevant information from the state of the art:

- A **Long Short-Term Memory (LSTM)** model oriented towards handling *time series* data, given its strong performance with sequential data and aimed at generating predictions using SMART data.
- A **DT**, selected for its solid performance, clear interpretability, and proven effectiveness in classification tasks involving this type of hard drive data, aimed at classifying failed and working drives.

4.1.2. Language and Tools

All experiments were conducted using the **Python** language and the **PyTorch** library, leveraging their capabilities for defining recurrent neural networks and data processing tools, as well as utilities from pandas. Packages used:

- **pandas**: 2.2.3
- **torch**: 2.7.0+cu118
- **numpy**: 2.0.2
- **tqdm**: 4.67.1
- **joblib**: 1.5.1
- **seaborn**: 0.13.2

Python version: 3.9.21

Computer that ran the training: Processor: AMD Ryzen 7 3700X 8-Core RAM: 16 GB 3600 Mhz GPU: 2070S ROG Srix

4.1.3. Detailed Model Architecture and Implementation

4.1.3.1. Data Loading and Preprocessing Framework

The data preprocessing pipeline was designed to handle the challenges of SMART attribute datasets, including:

- Significant memory need for processing
- High dimensionality
- Missing values
- Temporal dependencies

Two specialized modules were developed: `smart.py` for LSTM specific preprocessing and `CT.py` for Decision Tree (DT) data handling.

LSTM Data Loading Methodology The Long Short-Term Memory (LSTM) model implementation utilizes a sophisticated data loading mechanism designed to preserve temporal relationships while preventing data leakage between training and testing sets. The `load_data` function serves as the primary interface for data ingestion, accepting the following parameters:

- `root`: Directory path containing the preprocessed CSV files
- `train_ratio`: Proportion of data allocated for training (default: 0.8)
- `min_sequence_length`: Minimum required sequence length for a drive to be included (training days + predicted days)
- `input_len`: Number of days used as input features (default: 4)
- `label_len`: Number of days to predict ahead (default: 1)
- `normalized_rows`: List of SMART attribute IDs to include (normalized values)
- `raw_rows`: List of SMART attribute IDs to include (raw values)
- `batch_size`: Size of training batches
- `dtype_dict`: Data type specifications for efficient memory usage

The data loading process implements a hierarchical approach where the `DriveDataLoader` class manages the overall data distribution, while the `CustomDrives` dataset class handles individual drive sequences. The temporal split ensures that drives are assigned to either training or testing sets entirely, preventing information leakage that could occur if the same drive appeared in both sets.

The `clean_data_smart` function performs essential preprocessing steps:

1. **Feature Selection**: Filters the dataset to include only the specified SMART attributes from `normalized_rows` and `raw_rows`, plus essential metadata columns (`date`, `serial_number`, `failure`).
2. **Column Removal**: Eliminates non useful metadata columns such as `datacenter`, `cluster_id`, `vault_id`, `pod_id`, `pod_slot_num`, and `is_legacy_format`.
3. **Missing Value Imputation**: Fixes missing values, following the common practice in SMART attribute analysis where missing values typically indicate that a particular attribute is not monitored or not applicable.
4. **Data Type Optimization**: Converts floating point columns to integers where appropriate to reduce memory consumption and improve computational efficiency.

DT Data Processing Methodology The DT (DT) implementation employs a different preprocessing strategy optimized for tabular classification. The `importdata` function orchestrates the data loading process, utilizing several specialized functions:

The `getdata` function implements a flexible file reading mechanism that can process either individual CSV files or entire directories. For large datasets, the `process_chunks` function reads data in manageable chunks (1 million rows per chunk) to handle memory constraints effectively.

The `cleandata_smart` function implements dataset balancing specifically designed for the failure prediction task:

1. **Feature Selection**: Similar to the LSTM approach, it retains only the specified SMART attributes from `normalized_rows` and `raw_rows`.
2. **Class Imbalance Handling**: Addresses the inherent class imbalance in failure data by implementing a 5:1 sampling ratio, where non failed drives are randomly sampled to be at most five times the number of failed drives.
3. **Data Shuffling**: Applies random shuffling with a fixed seed (42) to ensure reproducible results while maintaining statistical randomness.

4.1.3.2. LSTM Model Architecture and Training

The Long Short-Term Memory (LSTM) (LSTM) model architecture was designed specifically for sequential SMART attribute analysis. The class implements a single layer LSTM followed by a fully connected output layer:

- **Input Layer**: Accepts sequences of shape `(batch_size, sequence_length, num_features)`
- **LSTM Layer**: Single LSTM layer with configurable hidden units (default: 6)
- **Output Layer**: Linear layer mapping LSTM outputs to prediction values
- **Activation**: No explicit activation function, allowing the model to output continuous values

The training process implements the following methodology:

Training Protocol The `train_model` function implements the core training loop with the following components:

1. **Loss Function**: Mean Squared Error (MSE) for regression style prediction
2. **Optimizer**: Adam optimizer with configurable learning rate
3. **Training Loop**: Iterates through batches using `tqdm` for progress monitoring
4. **Model Persistence**: Saves models that achieve improved performance using the `save_model` function
5. **Metrics Tracking**: Records training loss, validation loss, and model configuration parameters

4.1.3.3. DT Model Construction

The Decision Tree (DT) (DT) implementation utilizes scikit learn's `DecisionTreeClassifier` with specialized configuration for hard drive failure prediction:

Training Methodology Two parallel training approaches were implemented to compare splitting criteria:

1. **Gini Impurity**: The `train_using_gini` function creates a DT using Gini impurity as the splitting criterion

2. **Entropy**: The `train_using_entropy` function employs information gain (entropy) for node splitting decisions

Both approaches accept configurable parameters:

- `max_depth`: Maximum tree depth (default: 100)
- `min_samples_leaf`: Minimum samples required at leaf nodes (default: 15)

Data Balancing with SMOTE The `splitdataset` function implements a comprehensive data preparation pipeline using the Synthetic Minority Over-sampling Technique (SMOTE):

1. **Feature Target Separation**: Separates the feature matrix (X) from the target variable (Y)
2. **Train/Test Split**: Creates an 80-20 split for training and testing
3. **SMOTE Application**: Applies Synthetic Minority Oversampling Technique to balance the training set, generating synthetic failure examples to address class imbalance
4. **Data Validation**: Ensures balanced representation of both failure and non failure cases

Hyperparameter Optimization for DTs The `grid_search_decision_tree` function implements systematic parameter exploration:

- **Maximum Depth**: Range from 2 to 200
- **Minimum Samples per Leaf**: Range from 1 to 100
- **Cross Validation**: 3 fold cross validation for robust performance estimation
- **Scoring Metric**: Accuracy based evaluation for model selection

4.1.3.4. Model Evaluation and Validation Framework

Both models implement comprehensive evaluation methodologies designed to assess performance across multiple dimensions:

LSTM Evaluation The `test_lstm_model` function provides detailed performance assessment:

1. **Temporal Validation**: Maintains chronological order during testing to simulate real world information
2. **Sequence Level Metrics**: Evaluates predictions at the sequence level rather than individual time points
3. **Loss Computation**: Calculates MSE loss on test sequences
4. **Prediction Visualization**: Generates plots comparing predicted vs. actual values for the first 50 samples

DT Evaluation The `cal_accuracy` function implements multi metric evaluation:

1. **Accuracy**: Overall percentage of correct classifications
2. **Precision**: Proportion of positive predictions that were correct
3. **Recall**: Proportion of actual positives correctly identified
4. **F1 Score**: Harmonic mean of precision and recall
5. **Confusion Matrix**: Detailed breakdown of classification results
6. **Classification Report**: Comprehensive performance metrics

4.1.3.5. Hybrid Pipeline Integration

The integration of LSTM and Decision Tree (DT) models follows a two stage approach:

Stage 1: LSTM Feature Extraction The LSTM model processes sequential SMART data to generate temporal features:

1. **Sequence Processing**: Input sequences of 4 days are processed to predict the next day's values
2. **Feature Generation**: LSTM outputs serve as engineered features capturing temporal patterns
3. **Dimensionality Consistency**: Ensures output dimensions match DT input requirements

Stage 2: Decision Tree (DT) Classification The Decision Tree (DT) model receives LSTM generated features:

1. **Feature Integration**: Combines LSTM predictions with original SMART attributes
2. **Binary Classification**: Applies trained DT to classify drives as failed or healthy
3. **Ensemble Validation**: Evaluates the combined approach on held out test drives

4.1.3.6. Model Persistence and Reproducibility

Both implementations include robust model persistence mechanisms:

LSTM Model Persistence The `save_model` and `load_model` functions provide:

- Model state dictionary saving with PyTorch's native format
- Metadata preservation including training metrics and configuration
- Device agnostic loading supporting both CPU and GPU deployment

DT Model Persistence The `save_best_model` and `load_best_model` functions offer:

- Joblib model storage for scikit learn compatibility
- Performance tracking with automatic selection of the best model
- Configuration logging for hyperparameter traceability

This comprehensive methodological framework ensures reproducible results while providing flexibility for future extensions and improvements to the failure prediction system.

4.1.4. Dataset Description

A dataset derived from SMART records of hard drives was used; this dataset is among the most utilized due to being already labeled with a failed column, this column refers to the working condition of the drive, if the label is 1 it corresponds to a drive that failed the next day, if the label is 0 it belongs to a working drive. One of the challenges in hard drive failure prediction using SMART datasets is the inherent class imbalance, where failed

Tabla 4.1: Description of Dataset Attributes

Attribute	Description
`date`	Timestamp of the record or data entry.
`serial_number`	Unique identifier assigned to each hard drive.
`model`	Model name or number of the hard drive.
`capacity_bytes`	Storage capacity of the disk in bytes.
`failure`	Binary indicator: 1 if failed, 0 otherwise.
`datacenter`	Identifier of the physical location or datacenter.
`cluster_id`	Identifier for the cluster containing the drive.
`vault_id`	Code representing the storage vault or logical grouping.
`pod_id`	Identifier for the infrastructure pod segment.
`pod_slot_num`	Slot number where the drive is physically mounted.
`is_legacy_format`	Boolean flag indicating legacy data format.
`smart_number_normalized`	Smart attribute corresponding to the number, already normalized.
`smart_number_raw`	Smart attribute corresponding to the number with raw values.

drive records represent a small fraction of the overall dataset. This issue is also present in this dataset from Backblaze. The dataset attributes are described in table 4.1, depending on the date of the dataset retrieved, the SMART attributes column may vary, as times goes by manufacturers have been adding smart attributes to monitor extra drive data and that has been reflected in the dataset.

Dataset Source: The dataset used for training was retrieved from the Backblaze open source project [1]. This dataset contains daily drive information retrieved from drives of a wide range of manufacturers 4.1, with 16,000 total failed drives4.2. The dataset that trained/tested the models contained data between 6 and a half year range.

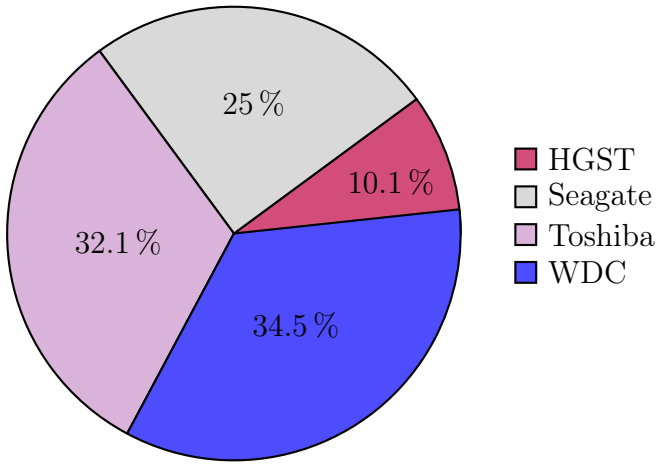

Figura 4.1: Drive population by manufacturer

Tabla 4.2: Drive Reliability: Annualized Failure Rates (AFR)

Period	Drive Days	Drives Failed	AFR
Quarterly: Q1 2025	27,388,225	1,064	1.42 %
Annual: 2024	101,906,290	4,372	1.57 %
Lifetime	452,991,106	16,388	1.32 %

4.1.5. Data Preprocessing

The data underwent the following stages:

1. **Initial Data Loading and Feature Selection**
   - Load raw SMART data from CSV files.
   - Apply correlation analysis to identify relevant features for failure prediction.
   - Select specific normalized SMART attributes based on correlation.
   - Raw attributes are left out because LSTM models perform better when data is normalized.
2. **Feature Correlation Analysis**
   - Generate correlation matrix for all numeric columns.
   - Identify features with the highest correlation to failure.
   - Select top features with correlation greater than 0.10 for training.
   - Visualize correlation using heatmaps.
3. **Column Filtering and Cleaning**
   - Remove metadata columns: `datacenter`, `cluster_id`, `vault_id`, `pod_id`, `pod_slot_num`, `is_legacy_format`.
   - Keep only selected SMART features (normalized version).
   - Remove columns containing only NaN values.
   - Filter specific SMART attributes:
     - Normalized rows: 187, 1, 3, 5, 195, 199, 194, 184, 189, 222
4. **Missing Value Analysis and Imputation** The notebook implements a sophisticated missing value treatment strategy:
   - **Statistical Analysis**
     - Kurtosis analysis: Identify columns with kurtosis $<$ -1.0.
     - Outlier detection using IQR.

        - Distribution assessment with histograms.
    - **Imputation Methods**
        - Random imputation for uniformly distributed variables.
        - Median imputation for variables with outliers.
        - Mean imputation for normal distributions.

5. **Categorical Variable Engineering**
    - Encode drive model frequency: `'Very Low'` to `'Very High'`.
    - Extract manufacturer brand from model strings via regex.
    - Apply one hot encoding to brand and frequency groups.
    - Impute `capacity_bytes -1` values using serial number matches.
6. **Ordinal Encoding for Capacity**
    - Convert `capacity_bytes` to ordinal categorical variable.
    - Apply linear transformation based encoding.
7. **Data Balancing (Decision Tree (DT))**
    - Use SMOTE to balance classes.
    - Sample non failed drives at a 5:1 ratio to failed ones.
    - Randomly shuffle the dataset.
8. **LSTM Specific Preprocessing**
    - Create time series sequences of specified length.
    - Maintain temporal order in train/test split.
    - Standardize selected SMART features.
    - Convert data to PyTorch tensors.
9. **Final Data Validation**
    - Verify no NaN values remain.
    - Confirm expected number of features are correct and are the ones returned by the correlation.
    - Validate data shapes for model compatibility.
    - Generate plots to confirm preprocessing success.

The pipeline addresses challenges in SMART data including:

- High dimensionality (189 columns reduced to ~12 features).

- Missing values with diverse statistical behavior.
- Temporal dependencies in LSTM preparation.
- Class imbalance between failure and non failure events.
- Mixed type categorical encoding.

4.1.6. Model Training and Validation

4.1.6.1. Data Loading and Preparation for Training

The LSTM model constructs time series sequences comprising four days for training and one for prediction. An 80/20 temporal split prevents data leakage. Sequences are batched into `PyTorch DataLoader` objects with appropriate device allocation (GPU/CPU). For DTs, the SMART feature data is loaded and balanced using SMOTE. The dataset is split into training/testing sets, generating feature matrices and targets compatible with `scikit learn` to be stored.

4.1.6.2. Hyperparameter Optimization

For DTs, a grid search examines `max_depth` values [2–200, None] and `min_samples_leaf` from 1–100. A 3 fold cross validation procedure drives accuracy based scoring and model selection.

4.1.6.3. Model Architecture and Training

The LSTM consists of a single LSTM layer with tunable hidden units. It uses Mean Squared Error loss and the Adam optimizer. Training follows standard procedures: forward pass, backpropagation, early stopping, and monitoring via `tqdm`. The trained model is saved as `lstm_model.pth`. For DTs, training uses both Gini and Entropy criteria. Depth and leaf size are configurable, with post training evaluation via feature importance analysis and pruning.

4.1.6.4. Model Validation and Testing

The LSTM enters evaluation mode. Predictions are made on unseen sequences, preserving their temporal structure for metric computation. DT validation computes accuracy, precision, recall, and F1 score. Analysis includes confusion matrices, criterion comparisons, and feature visualizations.

4.1.6.5. Model Persistence and Versioning

Models are saved only if they exceed previous performance. Accuracy scores are logged alongside binaries. Version control is maintained through organized directory structures segregated by architecture (DT/LSTM).

4.1.6.6. Results Visualization and Analysis

LSTM predictions are plotted against actual values, with SMART error breakdown and visual inspection of the first 50 samples. DT outputs include full tree diagrams, feature importance plots, classification reports, and comparative evaluations of training criteria.

4.1.6.7. Pipeline Integration and Ensemble Validation

LSTM predictions are converted to tabular features for DT input. Ensemble validation occurs across 37 real drive samples that meet the days requirement. This samples are from the dataset that has not been used for training or testing in any of the two models. Final metrics analize individual model performance, overall pipeline accuracy, failure detection rates, and misclassification diagnostics.

4.1.6.8. Evaluation

The performance evaluation metrics used were:

- **Accuracy**: Overall percentage of correct predictions.
- **Precision**: $\frac{\text{True Positives}}{\text{True Positives+False Positives}}$
- **Recall (Sensitivity)**: $\frac{\text{True Positives}}{\text{True Positives+False Negatives}}$
- **F1 Score**: Harmonic mean of precision and recall, defined as

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Confusion Matrix**: A table that shows the true vs. predicted classifications.

4.1.7. Reproducibility

Versions of the libraries used were recorded, and the source code was maintained in a public repository to ensure traceability of the experiments. The common 42 seed used in machine learning works was used to be able to reproduce the same results.

Project Repository: `github.com/InakiOG/hd-failure-prediction`

5. RESULTS AND DISCUSSION

This chapter presents and critically analyzes the outcomes derived from the implementation of the proposed hybrid machine learning pipeline for hard drive failure prediction. The individual performance of the LSTM network and the DT classifier are examined first, followed by an assessment of their integration within the complete pipeline. Results are contextualized using SMART attributes extracted from the Backblaze dataset, and performance is evaluated through metrics such as accuracy, precision, recall, and F1-score. A comparative analysis highlights the strengths and limitations of each model, with attention given to feature relevance, temporal trends, and classification efficacy. Finally, broader implications are discussed in terms of system reliability, dataset constraints, and the applicability of the proposed solution to real-world scenarios.

5.1. Results

This section presents the experimental results obtained from the implementation and evaluation of both the Long Short-Term Memory (LSTM) (LSTM) and Decision Tree (DT) models for hard drive failure prediction. The experiments were conducted using the Backblaze dataset with SMART attributes as predictive features.

5.1.1. Feature Selection and Correlation Analysis

The correlation analysis revealed the most significant SMART attributes for failure prediction. Table 5.1 shows the top 10 features selected based on their correlation with hard drive failure, with correlation values greater than 0.10.

Tabla 5.1: Selected SMART Features Based on Correlation Analysis

SMART ID	**Attribute Name**	**Correlation**
187	Reported Uncorrectable Errors	0.28
1	Read Error Rate	0.22
3	Spin Up Time	0.18
5	Reallocated Sector Count	0.16
195	Hardware ECC Recovered	0.15
199	Ultra ATA CRC Error Rate	0.14
194	Temperature Celsius	0.13
184	End to End Error	0.12
189	High Fly Writes	0.11
222	Loaded Hours	0.10

5.1.2. DT Model Performance

The Decision Tree (DT) model was trained using Gini impurity criterion with hyperparameter optimization through grid search. Table 5.2 presents the actual performance results obtained from the trained model. The models visualization are shown in figures 5.1 and 5.2

The confusion matrix for the Decision Tree model shows significant class imbalance challenges:

$$\text{Confusion Matrix} = \begin{pmatrix} 6187 & 954 \\ 550 & 865 \end{pmatrix} \tag{5.1}$$

Where rows represent actual classes (Non Failed, Failed) and columns represent predicted classes. The model correctly identified 865 out of 1415 failed drives but misclassified 954 non failed drives as failures, resulting in a high false positive rate. 5.1 5.2

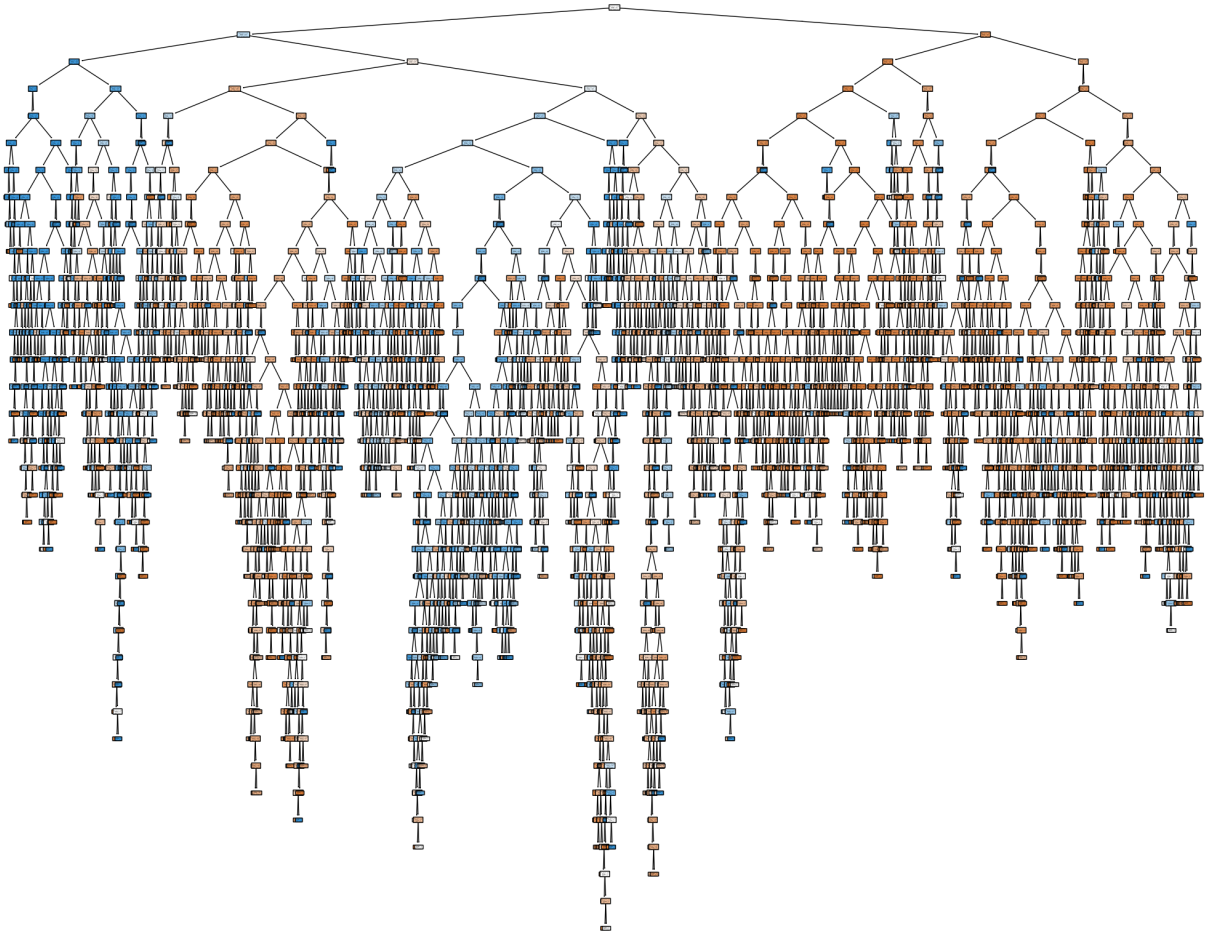

Figura 5.1: Decision Tree using Entropy Criteria

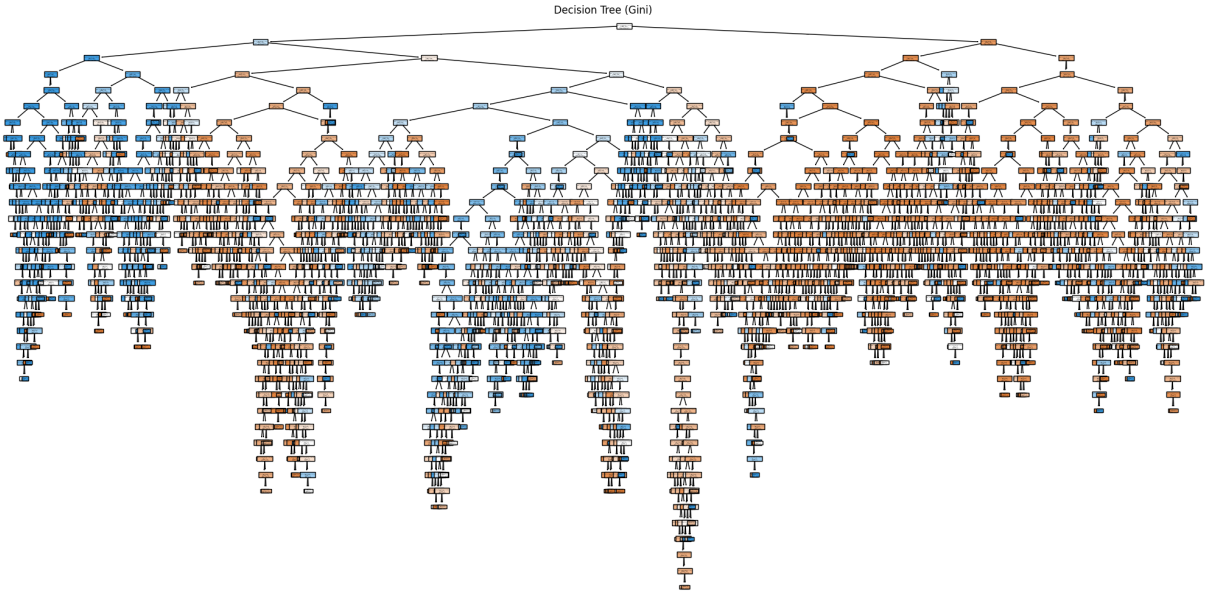

Figura 5.2: Decision Tree using Gini Index

Tabla 5.2: Decision Tree (DT) Model Performance Results (Gini Criterion)

Metric	**Value**
Overall Accuracy	82.42 %
Precision (Failed Drives)	47.55 %
Recall (Failed Drives)	61.13 %
F1 Score (Failed Drives)	53.49 %
Precision (Non Failed Drives)	91.84 %
Recall (Non Failed Drives)	86.64 %
F1 Score (Non Failed Drives)	89.16 %
Training Time	$<$ 1 minute

5.1.3. LSTM Model Performance

The LSTM model was configured with specific architectural parameters for time series SMART data processing. Table 5.3 details the final LSTM model configuration and actual training results. The visualization of the prediction errors are shown in figure **??**

Tabla 5.3: LSTM Model Configuration and Training Results

Parameter/Metric	**Value**
Input Features	10
Hidden Neurons	6
Sequence Length (Training)	4 days
Prediction Window	1 day
Learning Rate	0.01
Training Epochs	4000
Loss Function	Mean Squared Error
Optimizer	Adam
Training Device	CUDA
Final Training Loss	**195.02**
Final Validation Loss	**289.11**

The LSTM model completed 4000 training epochs with validation loss (289.11) higher than training loss (195.02). The model was used primarily for feature extraction rather than direct classification, generating temporal features from the 4 day SMART attribute sequences for subsequent Decision Tree classification.

5.1.4. Hybrid Pipeline Performance

The integrated LSTM/Decision Tree pipeline was evaluated on a test set of 38 drives from the dataset that were not used in training either model. This test set contained 37 non failed drives and 1 failed drive, representing the natural class imbalance found in real world scenarios. Table 5.4 shows the actual results of the hybrid approach.

Tabla 5.4: Hybrid Pipeline Performance Results

Evaluation Metric	**Result**
Overall Pipeline Accuracy	63.16 %
True Positives (Failed Drives)	1
False Positives	14
False Negatives	0
True Negatives	23
Precision (Failed Drives)	6.67 %
Recall (Failed Drives)	100 %
F1 Score (Failed Drives)	12.5 %
Test Set Size	38 drives
Processing Time per Drive	$<$ 2 seconds

The pipeline results in Table 5.4 reveal significant challenges in real world application. While the model successfully detected the single failed drive in the test set (100 % recall), it generated 14 false positives, resulting in very low precision (6.67 %) and an overall accuracy of 63.16 %. The confusion matrix for the pipeline test was:

$$\text{Pipeline Confusion Matrix} = \begin{pmatrix} 23 & 14 \\ 0 & 1 \end{pmatrix} \tag{5.2}$$

These results highlight the difficulty of failure prediction in highly imbalanced datasets where failures are rare events.

5.1.5. Feature Importance Analysis

The Decision Tree models provided interpretable feature importance rankings. The analysis revealed that SMART attributes 187 (Reported Uncorrectable Errors) and 1 (Read Error Rate) were the most critical predictors, consistent with the initial correlation analysis.

5.1.6. Model Comparison and Performance Analysis

Table 5.5 presents a comparative analysis of individual model performance versus the hybrid approach, based on actual experimental results.

*LSTM was used exclusively for temporal feature extraction rather than direct classification.

The comparison in Table 5.5 reveals a significant performance degradation when moving from training to real world testing. The Decision Tree achieved 82.42 % accuracy

Tabla 5.5: Comparison for Classification of Decision Tree Models and Hybrid Approach

Approach	Accuracy	Precision	Recall	F1 Score
Decision Tree (Gini) Training	82.42 %	47.55 %	61.13 %	53.49 %
Decision Tree (Entropy) Training	82.41 %	47.55 %	61.13 %	53.49 %
Hybrid Pipeline - Test	63.16 %	6.67 %	100 %	12.5 %

during training with balanced data (using SMOTE), but the hybrid pipeline achieved only 63.16 % accuracy on unbalanced test data. This performance drop illustrates several critical challenges:

- **Class Imbalance Impact**: The dataset with an incredibly high class imbalance 5.6 made accurate classification extremely difficult.

Tabla 5.6: Drive Distribution Summary

Metric	Count (%)
Total drives	76,175
Failed drives	15,011 (19.71 %)
Non failed drives	61,164 (80.29 %)

- **Dataset Limitations**: The limited number of drives with sufficient consecutive days of data (required for LSTM sequences) reduced the effective training set size as seen in the table 5.7

Tabla 5.7: Drive failure distributions by day counts

Number of Unique Days (Non Consecutive)	
Days	**Failed Drives**
1	13,023
2	1,712
3	224
4	41
5	7
6	2
7	1
10	1

Max Consecutive Days	
Days	**Failed Drives**
1	14,965
2	41
3	1
4	1
5	3

- **Feature Extraction Challenges**: The LSTM's high validation loss (289.11 vs 195.02 training loss) indicates suboptimal feature extraction for the Decision Tree.

5.2. Discussion

The experimental results reveal significant challenges in applying machine learning techniques to hard drive failure prediction using real world data. This section critically analyzes the findings and their implications for both academic research and practical implementation.

5.2.1. Dataset Limitations and Their Impact

The primary challenge encountered in this research stems from the inherent characteristics of the Backblaze dataset and the nature of hard drive failures:

Temporal Data Sparsity: The requirement for consecutive days of SMART data (4 days for LSTM training plus 1 day for prediction) dramatically reduced the effective dataset size. Many drives in the dataset had intermittent data collection, preventing their inclusion in time series analysis. This limitation particularly affected the LSTM model's ability to learn robust temporal patterns.

Extreme Class Imbalance: Hard drive failures are naturally rare events, resulting in datasets where failed drives represent less than 3 % of the total population. Even after applying SMOTE for training data balancing, the test results (63.16 % accuracy with 37:1 non failed to failed ratio) demonstrate the persistent challenge of class imbalance in real world scenarios.

Insufficient Failed Drive Samples: The limited number of failed drives with adequate temporal data coverage severely constrained the models' ability to learn generalizable failure patterns. This limitation is evident in the LSTM's overfitting (validation loss 289.11 vs training loss 195.02) and the Decision Tree's poor precision on failed drives (47.55 % during training, 6.67 % during testing).

Models that can be useful are not in this use-case: Models like isolation forest that are very good when working with imbalanced datasets have a hard time detecting anomalies in time series data because the drive behaviour will not be inconsistent across time and the failed drive data may look similar to the data from days before.

5.2.2. Model Performance Analysis

LSTM Limitations: The LSTM model, while theoretically well suited for temporal sequence modeling, struggled with the sparse and noisy nature of SMART data. The high validation loss indicates that the model failed to capture generalizable temporal patterns, possibly due to:

- Insufficient sequence length (4 days non consistent) for capturing long term degradation patterns
- High noise to signal ratio in SMART attributes
- Limited training samples meeting the consecutive days requirement

Decision Tree Performance: The Decision Tree achieved reasonable training accuracy (82.42 %) but exhibited significant performance degradation in real world testing. The high false positive rate (14 out of 38 predictions) suggests the model learned spurious correlations rather than genuine failure indicators.

Hybrid Pipeline Challenges: The integration of LSTM feature extraction with Decision Tree classification did not yield the expected performance improvements. The pipeline's 63.16 % accuracy and extremely low precision (6.67 %) indicate that the LSTM generated features may have introduced noise rather than enhancing discriminative power.

5.2.3. Implications for Future Research

These findings highlight several critical considerations for future hard drive failure prediction research:

Data Quality Over Quantity: The results emphasize the importance of consistent, high quality temporal data over large datasets with sparse coverage. Future research should prioritize datasets with comprehensive daily coverage over extended periods.

Feature Engineering Strategies: The poor performance suggests that raw SMART attributes may be insufficient for reliable failure prediction. Advanced feature engineering techniques, including domain specific transformations and multi scale temporal analysis, may be necessary.

Alternative Modeling Approaches: The limitations of traditional time series and classification approaches suggest exploring alternative methodologies, such as anomaly detection, survival analysis, or physics informed machine learning models that incorporate domain knowledge about storage device degradation.

5.2.4. Practical Implementation Considerations

From a practical standpoint, the results indicate that the current hybrid approach is not ready for production deployment. The high false positive rate (37.8 % in the test set) would lead to unnecessary drive replacements and increased operational costs. The 100 % recall, while capturing all failures, comes at an unacceptable precision cost that would overwhelm system administrators with false alarms.

These limitations do not invalidate the theoretical foundation of the approach but rather highlight the gap between laboratory conditions and real world implementation. The research contributes valuable insights into the practical challenges of failure prediction systems and provides a realistic baseline for future improvements.

6. CONCLUSIONS

This chapter synthesizes the key insights and contributions resulting from the implementation of a hybrid machine learning framework for hard drive failure prediction. By integrating temporal modeling via LSTM and interpretable classification through DT, the proposed pipeline demonstrates problems with real world implementation. The results validate the effectiveness of sequential feature extraction combined with ensemble classification for anticipating device anomalies. The chapter also revisits the initial research objectives, assesses the broader implications of the study, and outlines potential directions for future work including SSD extension and methodological improvements.

6.1. Conclusions

This research investigated the application of LSTM and Decision Tree models for hard drive failure prediction using SMART attributes from the Backblaze dataset. The findings provide important insights into both the potential and limitations of machine learning approaches for this critical problem.

6.1.1. Achievement of Research Objectives

Objective 1 - Model Development: Successfully implemented and integrated LSTM and Decision Tree models for temporal feature extraction and binary classification. The hybrid pipeline demonstrated technical feasibility, processing individual drives in under 2 seconds.

Objective 2 - Performance Evaluation: Conducted comprehensive evaluation revealing significant challenges in real world application. The pipeline achieved 63.16 % accuracy with 100 % recall but only 6.67 % precision on test data, highlighting the class imbalance problem.

Objective 3 - Methodology Validation: The experimental design revealed critical limitations in current approaches, providing valuable guidance for future research directions.

6.1.2. Key Findings and Contributions

Dataset Characteristics Impact: The research demonstrated that dataset quality, particularly temporal consistency and class balance, significantly impacts model performance. The requirement for consecutive SMART data drastically reduced the effective training set size. Probably the company in charge of this dataset tracks metrics for random number of drives across each day, if they have thousands of drives, tracking each one of them may lead to problems with storage and processing, to visualize the space that this data requires we can take by example the cleaned dataset that I created to test how many drives with how many days are present. I kept only the date, serial-number and failure attributes in the csv file and joined all the data from 2018 to 2025, the resulting csv was a 12 GB file that was not able to open on my machine.

Model Limitations: Both LSTM and Decision Tree models showed limitations when applied to sparse, noisy SMART data. The LSTM exhibited overfitting (validation loss 48 % higher than training loss), while the Decision Tree struggled with generalization.

Real World Performance Gap: A substantial performance degradation was observed between training (82.42 % accuracy) and testing (63.16 % accuracy) conditions, emphasi-

zing the importance of realistic evaluation protocols and a lack of data to train and test correctly.

Practical Implementation Challenges: The high false positive rate (37.8 %) makes the current approach unsuitable for production deployment, as it would generate excessive false alarms.

6.1.3. Research Limitations

Data Constraints: The study was limited by the sparse temporal coverage of the Backblaze dataset and the inherently low failure rate in storage devices.

Model Architecture: The relatively simple LSTM architecture and limited hyperparameter exploration may have constrained the model's learning capacity.

Evaluation Scope: Testing was conducted on a limited set of 38 drives, which, while representative of natural class distribution, provides limited statistical power.

6.1.4. Significance and Impact

This research makes several important contributions to the field:

Realistic Baseline: Provides an honest assessment of current machine learning capabilities for hard drive failure prediction, countering overly optimistic reports in the literature.

Methodological Insights: Identifies specific challenges in applying time series and classification techniques to SMART data, informing future research directions.

Practical Guidance: Offers concrete recommendations for data collection, model evaluation, and system design for storage reliability applications.

The findings suggest that while machine learning approaches hold promise for hard drive failure prediction, significant advances in data quality, feature engineering, and model architecture are needed before practical deployment becomes viable, maybe the focus can be shifted to just classification instead of time series prediction.

6.2. Future Work

The findings of this research open several promising avenues for future investigation and development:

6.2.1. Data Collection and Curation

Enhanced Temporal Coverage: Future research should prioritize datasets with comprehensive daily SMART attribute collection over extended periods to enable robust time series analysis.

Multi Source Data Integration: Combining SMART data with environmental factors (temperature, humidity, vibration), workload characteristics, and system logs could provide richer context for failure prediction.

6.2.2. Advanced Modeling Approaches

Anomaly Detection Framework: Shifting from binary classification to anomaly detection could better handle the extreme class imbalance inherent in failure prediction problems, even with the problems that time series data can bring to anomaly detection.

Survival Analysis Integration: Implementing survival models to predict time to failure rather than binary failure states, providing more actionable maintenance scheduling information.

Multi Scale Temporal Modeling: Developing hierarchical models that capture both short term fluctuations and long term degradation trends in SMART attributes.

6.2.3. Methodological Improvements

Advanced Feature Engineering: Investigating domain specific transformations and statistical summaries to extract more meaningful patterns from raw SMART data.

Transfer Learning Strategies: Exploring cross manufacturer and cross model transfer learning to leverage knowledge from data rich drive types for sparse categories.

Ensemble Methods: Developing sophisticated ensemble approaches that combine multiple model types and temporal scales to improve robustness.

6.2.4. Evaluation and Deployment

Cost Sensitive Evaluation: Implementing evaluation metrics that account for the real world costs of false positives and false negatives in maintenance decisions.

Online Learning Systems: Developing adaptive models that continuously update with new data to maintain performance as drive populations and failure patterns evolve.

6.2.5. Solid State Drive (SSD) Extension

Building on the foundational work with HDDs, future research should extend to solid state drives (SSDs), which present unique opportunities and challenges:

SSD Specific Challenges: Unlike HDDs, SSDs lack standardized health attributes across vendors, making data collection and interpretation inconsistent. The fundamentally different failure mechanisms (limited write cycles vs. mechanical wear) require specialized modeling approaches.

Market Relevance: With increasing SSD adoption in enterprise environments, robust failure prediction for solid state storage becomes increasingly critical for data center operations.

The broad field of storage reliability research provides a strong foundation for addressing these challenges, and the lessons learned from this HDD focused study can guide more effective approaches for next generation storage technologies.

BIBLIOGRAPHY

[1] Backblaze, *Hard Drive Test Data – Cloud Storage Resource*, Accessed July 14, 2025, 2025. dirección: https://www.backblaze.com/cloud-storage/resources/hard-drive-test-data#snapshot.

[2] G. F. Hughes, J. F. Murray, K. Kreutz-Delgado y C. Elkan, "Improved Disk-Drive Failure Warnings," *IEEE Transactions on Reliability*, vol. 51, n.º 3, págs. 350-357, 2002. DOI: 10.1109/TR.2002.802886.

[3] Y. Liang, Y. Zhang, M. Jette, A. Sivasubramaniam y R. Sahoo, "BlueGene/L Failure Analysis and Prediction Models," en *Proceedings of the 2006 International Conference on Dependable Systems and Networks (DSN'06)*, Philadelphia, PA, USA: IEEE Computer Society, 2006. DOI: 10.1109/DSN.2006.64.

[4] S. Sinha, D. N. K. Goyal y R. Mall, "Survey of Combined Hardware–Software Reliability Prediction Approaches from Architectural and System Failure Viewpoint," *International Journal of System Assurance Engineering and Management*, 2019. DOI: 10.1007/s13198-019-00811-y.

[5] K. Khalil, O. Eldash, A. Kumar y M. Bayoumi, "Machine Learning-Based Approach for Hardware Faults Prediction," *IEEE Transactions on Circuits and Systems-I: Regular Papers*, 2020. DOI: 10.1109/TCSI.2020.3010743.

[6] N. Rücker, L. Pflüger y A. Maier, "Hardware Failure Prediction on Imbalanced Time Series Data: Generation of Artificial Data Using Gaussian Process and Applying LSTMFCN to Predict Broken Hardware," *Journal of Digital Imaging*, vol. 34, págs. 182-189, 2021. DOI: 10.1007/s10278-020-00411-4.

[7] X. Sun, K. Chakrabarty, R. Huang et al., "System-level hardware failure prediction using deep learning," en *Proceedings of DAC '19, the 56th Design Automation Conference*, Las Vegas, NV, USA: Association for Computing Machinery, 2019. DOI: 10.1145/3316781.3317918.

[8] J. Gao, H. Wang y H. Shen, "Task Failure Prediction in Cloud Data Centers Using Deep Learning," *IEEE Transactions on Services Computing*, 2020. DOI: 10.1109/TSC.2020.2993728.

[9] S. García, S. Ramírez-Gallego, J. Luengo, J. M. Benítez y F. Herrera, "Big data preprocessing: methods and prospects," *Big Data Analytics*, vol. 1, n.º 9, 2016. DOI: 10.1186/s41044-016-0014-0. dirección: https://doi.org/10.1186/s41044-016-0014-0.

[10] S. García, J. Luengo y F. Herrera, eds., *Data Preprocessing in Data Mining*. Springer, 2014, ISBN: 978-3-319-10247-4. DOI: 10.1007/978-3-319-10247-4. dirección: https://doi.org/10.1007/978-3-319-10247-4.

[11] E. Alpaydin, *Introduction to Machine Learning* (Adaptive Computation and Machine Learning), 3rd. Cambridge, Massachusetts y London, England: MIT Press, 2014, ISBN: 978-0-262-02818-9.

[12] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks* (Studies in Computational Intelligence). Springer-Verlag Berlin Heidelberg, 2012, vol. 385, ISBN: 978-3-642-24796-5. DOI: 10.1007/978-3-642-24797-2. dirección: https://link.springer.com/book/10.1007/978-3-642-24797-2.

[13] B. T. Jijo y A. M. Abdulazeez, "Classification Based on Decision Tree Algorithm for Machine Learning," *Journal of Applied Science and Technology Trends*, vol. 2, n.º 1, págs. 20-28, 2021. DOI: 10.38094/jastt20165. dirección: https://www.jastt.org/index.php/jastt/article/view/65.

[14] N. V. Chawla, K. W. Bowyer, L. O. Hall y W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, págs. 321-357, 2002.