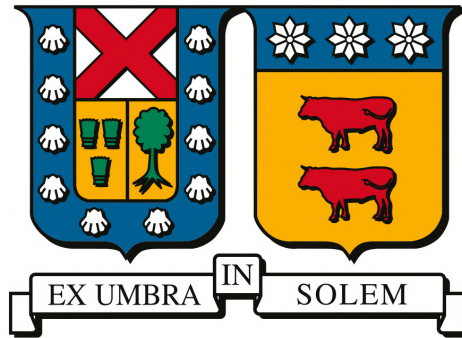




UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INDUSTRIAS

Análisis de Efectividad de Modelos de Machine Learning para la Estimación de RUL de Motor Turbofán N-CMAPSS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERIA CIVIL
INDUSTRIAL

Autor: Lucas Aaron Huito Fuentes
Profesor Guía: Fredy Kristjanpoller
Profesor Co-referente: Simón Gómez

26 de junio de 2026



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: Análisis de Efectividad de Modelos de Machine Learning para la Estimación de RUL de Motor Turbofán N-CMAPSS

Nombre del candidato(a): Lucas Aaron Huito Fuentes

Carrera / Grado: Ingeniero Civil Industrial

Campus: Casa Central

Departamento: Industrias

2.- VALIDACIÓN DEL PROFESOR GUÍA

Yo, Fredy Kristjanpoller Rodríguez, en mi calidad de profesor guía del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por (**marcar una opción**):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 26/06/2026

Firma:

Estudiante o Candidato(a): Lucas Aaron Huito Fuentes

Fecha: 26/06/2026

Firma:

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Índice general

1. Problema de Investigación	11
1.1. Contexto de la Inteligencia Artificial en la Industria	11
1.2. El Mantenimiento Predictivo en la Industria 4.0	11
1.3. La Brecha entre la Necesidad Operativa y la Realidad de Datos	12
1.4. El Conflicto Técnico: XGBoost frente a LSTM	13
2. Objetivos	5
2.1. Objetivo General	5
2.2. Objetivos Específicos	5
3. Marco Teórico	7
3.1. Inteligencia Artificial (IA)	7
3.2. Machine Learning (ML)	8
3.3. Deep Learning y Redes Neuronales Artificiales (ANN)	8
3.3.1. Arquitectura y Funcionamiento de una ANN	9
3.4. Mantenimiento Predictivo y Vida Útil Remanente (RUL)	10
3.4.1. Efectividad de la Estrategia Predictiva	11
3.4.2. Conceptualización de la Vida Útil Remanente (RUL)	12
3.4.3. La RUL como Problema de Modelado Estadístico	12
3.5. Modelos de Aprendizaje Supervisado	13
3.5.1. Extreme Gradient Boosting (XGBoost)	13

ÍNDICE GENERAL

3.5.1.1.	Fundamentos Arquitectónicos	13
3.5.1.2.	Ventajas Competitivas	13
3.5.1.3.	Formulación Matemática y Optimización	14
3.5.2.	Modelo Long Short-Term Memory (LSTM)	15
3.5.2.1.	Mecanismos de Control: Las Compuertas	16
3.5.2.2.	Funciones de Activación y su Rol Crítico	16
3.6.	Entorno de Simulación N-CMAPSS	17
3.6.1.	Proceso de Construcción de las Bases de Datos	17
3.6.2.	Arquitectura y Componentes de Interés	18
3.6.3.	Selección de Subconjuntos de Datos	19
3.6.4.	Taxonomía de Variables y Sensores	20
3.7.	Método de Evaluación de Modelos	23
4.	Metodología	25
4.1.	Selección y Caracterización de los Datos	25
4.2.	Preprocesamiento y Preparación de Datos	25
4.2.1.	Generación de Ventanas Temporales y RUL Rectificada	25
4.2.2.	Estrategia de Reducción de Datos (Simulación de Escasez)	26
4.3.	Implementación y Entrenamiento de los Modelos	27
4.3.1.	Entrenamiento Incremental de XGBoost	27
4.3.2.	Modelo de Aprendizaje Profundo (LSTM)	28
4.4.	Métricas de Evaluación	28
5.	Implementación y Arquitectura de Modelos	30
5.1.	Limitaciones Computacionales y Estrategias de Mitigación	30
5.2.	Ecosistema de Software y Librerías	31
5.3.	Configuración Arquitectónica de los Modelos	31

ÍNDICE GENERAL

5.3.1. XGBoost: Aprendizaje Incremental y Aplanamiento de Datos	31
5.3.2. LSTM: Arquitectura Profunda y Secuencialidad	32
5.3.3. Resumen de Hiperparámetros	35
6. Desempeño de los Modelos en las Distintas BBDD	38
6.1. Resultados obtenidos	38
6.1.1. Resultados LSTM	39
6.1.2. Resultados XGBoost	47
7. Análisis Diferencial de las Capacidades Predictivas	55
7.1. Comparación de Exactitud Predictiva	55
7.2. Análisis de Estabilidad del Desempeño	58
7.3. Modelos LSTM y XGBoost de mejor desempeño en las tres BBDD	62
8. Aplicaciones Industriales y Hoja de Ruta de Implementación	66
8.1. Propuesta de Adopción Escalonada	68
8.1.1. Casos de Aplicación por Sector Industrial	68
9. Conclusión	70
9.1. Resiliencia vs. Complejidad Arquitectónica	70
9.2. El Umbral Crítico de Viabilidad Operativa	71
9.3. Estabilidad, Sesgo y Gestión del Riesgo	71
9.4. Reflexión Final y Adopción en la Industria 4.0	72

Problema de Investigación

1.1. Contexto de la Inteligencia Artificial en la Industria

La Inteligencia Artificial (IA) ha permeado transversalmente diversas áreas del conocimiento, consolidándose como una herramienta estratégica para la optimización de procesos en múltiples sectores económicos. No obstante, dada su naturaleza técnica, es imperativo establecer definiciones fundamentales antes de profundizar en aplicaciones específicas. Según Mhlanga (2023, p. 3), la IA puede definirse como una disciplina que permite la resolución de problemas mediante la combinación de ciencias de la computación y extensos conjuntos de datos.

Dentro de este ecosistema, el Machine Learning (ML) o aprendizaje automático emerge como un subconjunto crítico. Su principio operativo no reside en la programación de reglas estáticas, sino en el desarrollo de modelos que identifican patrones autónomamente para alcanzar un objetivo predictivo. En el ámbito industrial, esta capacidad de aprendizaje es el motor que impulsa la transición hacia sistemas autónomos y eficientes.

1.2. El Mantenimiento Predictivo en la Industria 4.0

Una de las aplicaciones más disruptivas del ML es el mantenimiento predictivo (PdM, por sus siglas en inglés). Esta práctica se ha erigido como un pilar fundamental de la Cuarta Revolución Industrial o Industria 4.0, la cual se fundamenta en la convergencia de

1.3 La Brecha entre la Necesidad Operativa y la Realidad de Datos

tecnologías digitales y operativas. La relevancia de este enfoque radica en su capacidad para transformar la gestión de activos:

Uno de los pilares de esta última industria es el mantenimiento predictivo, que trata de aplicar las últimas tecnologías para cambiar los escenarios y evitar cualquier tipo de avería o anomalía mediante la predicción, la reducción de costos y de tiempos de parada (Achouch et al., 2022, p. 3).

1.3. La Brecha entre la Necesidad Operativa y la Realidad de Datos

A pesar de los beneficios teóricos, existe una brecha significativa en la implementación práctica. De acuerdo con una encuesta global realizada por MaintainX (2024), donde el 31 % de los 1,165 profesionales consultados pertenecen al sector manufacturero, el 22 % identifica las fallas de los equipos como la causa principal del tiempo de inactividad no planificado (MaintainX, 2024, p. 17). No obstante, el 57 % de las organizaciones aún depende de un mantenimiento reactivo (intervenir solo cuando el equipo falla) frente a un incipiente 30 % que ha logrado adoptar estrategias predictivas (MaintainX, 2024, p. 23).

Esta resistencia a la adopción no se debe a una falta de interés, sino a una barrera de entrada técnica: la dependencia de grandes volúmenes de datos. Los modelos de ML requieren una base de datos (BBDD) robusta para capturar las relaciones no lineales entre variables; a mayor complejidad del fenómeno industrial, mayor es la cantidad de datos requerida para evitar errores de generalización. Este desafío es particularmente agudo en organizaciones de menor escala. Tawil et al. (2024) documentaron esta limitación en su estudio sobre el sector de impresión 3D:

Una de las principales limitaciones que encontramos fue la insuficiencia de datos de fabricación... esto no implica que no existan datos, sino que podrían no ser

suficientes para implementar modelos de aprendizaje automático supervisado (p. 10).

1.4. El Conflicto Técnico: XGBoost frente a LSTM

Ante la escasez de datos históricos, surge la necesidad de evaluar qué arquitectura de ML es más resiliente. Esta investigación propone un análisis comparativo entre dos paradigmas distintos:

- XGBoost (Extreme Gradient Boosting): Un modelo basado en árboles de decisión que destaca por su eficiencia computacional y su capacidad para manejar datos tabulares con alta precisión, incluso en muestras moderadas.
- LSTM (Long Short-Term Memory): Una arquitectura de red neuronal recurrente diseñada específicamente para capturar dependencias temporales en series de tiempo, pero que suele requerir volúmenes de datos considerables para converger sin sobreajuste (*overfitting*).

Para resolver esta incertidumbre, se utilizará el conjunto de datos *N-CMAPSS* (Turbofan Engine Degradation Simulation) de la NASA. Aunque este *dataset* es exhaustivo, se empleará como un laboratorio controlado para simular escenarios de datos limitados y relaciones complejas de degradación. El desempeño se medirá estrictamente bajo dos métricas fundamentales de error:

- Raíz del Error Cuadrático Medio (RMSE)
- Error Absoluto Medio (MAE)

En virtud de lo expuesto, la presente investigación busca dar respuesta a la siguiente interrogante:

1.4 El Conflicto Técnico: XGBoost frente a LSTM

- ¿Cuál de los modelos, XGBoost o LSTM, presenta un desempeño óptimo en términos de precisión ($RMSE$ y MAE) al enfrentarse a un historial de datos limitado en un contexto de degradación industrial?

2

Objetivos

2.1. Objetivo General

Evaluar comparativamente el desempeño y la robustez de los modelos XGBoost y LSTM en la predicción de la Vida Útil Remanente (*Remaining Useful Life*, RUL) de sistemas mecánicos complejos, analizando el impacto de la reducción progresiva del volumen de datos de entrenamiento provenientes del *dataset* N-CMAPSS, con el fin de determinar la viabilidad de ambas arquitecturas en entornos industriales con baja disponibilidad de información.

2.2. Objetivos Específicos

- **Caracterizar** técnicamente los subconjuntos de datos de la base N-CMAPSS, identificando las particularidades de los perfiles de vuelo, la cantidad de unidades (motores) y la complejidad de los modos de falla registrados.
- **Configurar** las arquitecturas de los modelos XGBoost y LSTM mediante la definición de hiperparámetros basada en criterios de ingeniería y literatura especializada, asegurando una capacidad predictiva competitiva de cada modelo bajo condiciones de disponibilidad total de datos.
- **Cuantificar** la sensibilidad del desempeño de los modelos, utilizando las métricas

2.2 Objetivos Específicos

$RMSE$ y MAE , frente a la disminución sistemática del historial de datos disponible para el proceso de entrenamiento.

- **Determinar** los umbrales de robustez de cada modelo, estableciendo un marco comparativo que identifique qué arquitectura (aprendizaje profundo vs. aprendizaje basado en árboles) mantiene una mayor precisión y estabilidad en escenarios de escasez de datos.

3

Marco Teórico

Para fundamentar el desarrollo y la evaluación de los modelos predictivos propuestos, es indispensable establecer una base conceptual sobre la Inteligencia Artificial (IA) y sus ramificaciones. Esta sección explora la evolución desde los sistemas basados en reglas hasta las arquitecturas de aprendizaje profundo.

3.1. Inteligencia Artificial (IA)

Una definición operativa y contemporánea de la IA se centra en la creación de agentes diseñados para alcanzar objetivos específicos definidos por y para el ser humano. Esta perspectiva es fundamental, pues subraya la alineación de valores entre la máquina y el usuario. Como señalan Russell y Norvig (2022, p. 23), el propósito no es desarrollar máquinas que posean una inteligencia autónoma para perseguir sus propios fines, sino agentes que actúen de manera demostrablemente beneficiosa para la humanidad. La complejidad de estos agentes permite clasificarlos en dos grandes categorías según su mecanismo de inferencia:

1. **IA de Conocimiento Elaborado:** Basada en la programación explícita de reglas lógicas que estandarizan el comportamiento del sistema. El éxito del resultado depende directamente de la exhaustividad de las reglas definidas por el programador.

2. **Machine Learning (Aprendizaje Automático):** Un enfoque donde el sistema extrae conocimiento directamente de los datos, permitiendo una mayor flexibilidad y capacidad de generalización ante problemas complejos.

3.2. Machine Learning (ML)

La distinción fundamental entre los sistemas tradicionales y el ML reside en el origen del conocimiento. Allen (2020, p. 7) destaca que, mientras los sistemas de conocimiento elaborado reciben instrucciones humanas en forma de reglas manuales, los sistemas de ML tienen la capacidad de generar sus propias reglas a partir del análisis de patrones en los datos. Este proceso se rige por algoritmos que procesan bases de datos (BBDD) para transformar información de entrada en resultados predictivos. Dentro del ML, coexisten modelos clásicos, frecuentemente basados en árboles de decisión o ensambles, y modelos de alta complejidad denominados *Deep Learning* (DL).

En inteligencia artificial (IA), el aprendizaje automático es la capacidad de adaptarse automáticamente con poca o ninguna intervención humana, y el aprendizaje profundo es un subconjunto del aprendizaje automático que utiliza redes neuronales para simular el procedimiento de aprendizaje del cerebro humano. (Taye, 2023, p. 3)

3.3. Deep Learning y Redes Neuronales Artificiales (ANN)

El *Deep Learning* se fundamenta en las Redes Neuronales Artificiales (ANN, por sus siglas en inglés), las cuales se han consolidado como el método más eficaz y popular en la investigación científica desde la década de 1980 (Das et al., 2017, p. 6).

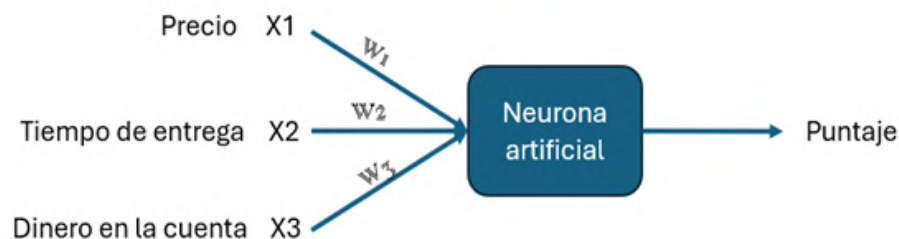
3.3.1. Arquitectura y Funcionamiento de una ANN

Una ANN es un modelo computacional que emula la estructura biológica del cerebro mediante la interconexión de neuronas artificiales dispuestas en capas. Su composición estándar incluye tres niveles críticos:

- **Capa de entrada:** Compuesta por nodos que reciben los datos brutos y los distribuyen a la red.
- **Capas ocultas:** Una o varias capas intermedias donde ocurre el procesamiento. Cada neurona en estas capas recibe información, le aplica una transformación matemática y la transmite a la siguiente fase.
- **Capa de salida:** Recibe los datos procesados y aplica un criterio final para retornar la predicción o clasificación.

El mecanismo de aprendizaje de una neurona individual se basa en la asignación de pesos (w). Como se ilustra en la Figura 3.1, una neurona puede recibir múltiples entradas (x_1, x_2, x_3). Cada entrada es multiplicada por un peso que define su importancia relativa; por ejemplo, en una decisión financiera, el "dinero en cuenta" podría tener un peso superior al "tiempo de entrega" para determinar el puntaje final.

Figura 3.1: Diagrama del funcionamiento de una neurona artificial.

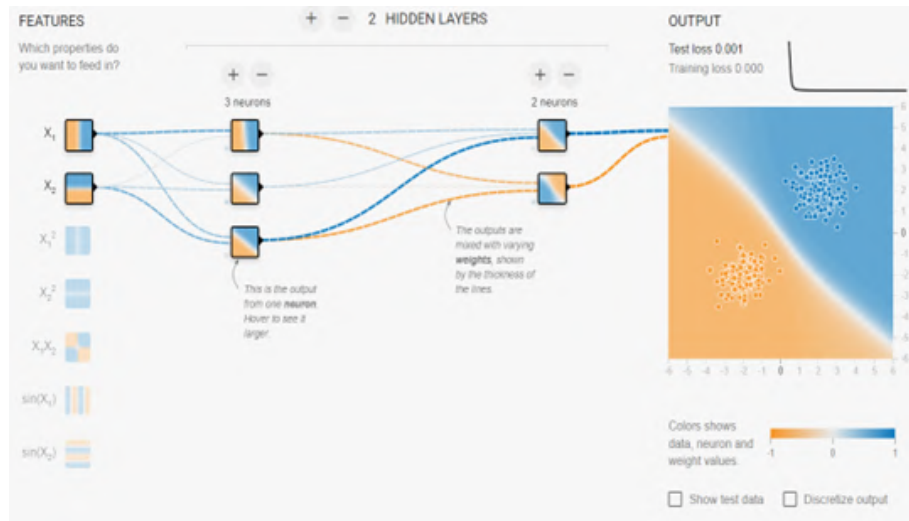


Nota. La neurona procesa las variables de entrada multiplicándolas por pesos específicos para generar un valor de salida.

3.4 Mantenimiento Predictivo y Vida Útil Remanente (RUL)

Cuando estas neuronas se organizan en múltiples capas, la red adquiere la capacidad de resolver problemas de alta complejidad, como la separación de conjuntos de datos no lineales (ver Figura 3.2). Este proceso de ajuste constante de pesos es lo que permite a modelos como LSTM capturar dependencias temporales en sistemas industriales complejos (Smilkov & Carter, 2016).

Figura 3.2: Representación de una red neuronal de dos capas ocultas para clasificación.



Nota. Ilustración de la separación de conjuntos mediante nodos de entrada, capas ocultas y una salida gráfica. Creado por Smilkov y Carter (2016).

3.4. Mantenimiento Predictivo y Vida Útil Remanente (RUL)

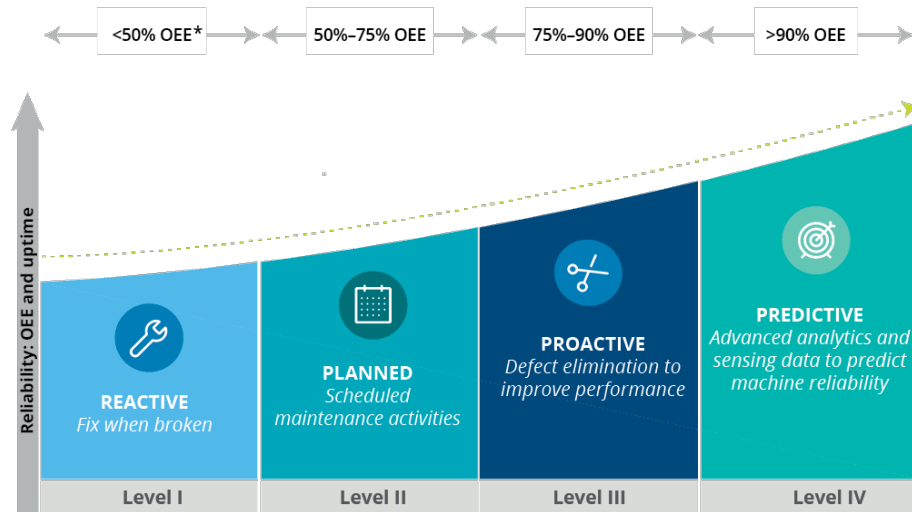
El mantenimiento predictivo (PdM) se define como una estrategia de gestión de activos que utiliza el estado real de los equipos para optimizar las intervenciones técnicas. A diferencia del mantenimiento correctivo que actúa tras la falla o el preventivo que actúa por calendario, el PdM busca intervenir en el momento exacto en que la degradación comienza a comprometer la disponibilidad operativa, pero antes de que se produzca una interrupción funcional.

3.4 Mantenimiento Predictivo y Vida Útil Remanente (RUL)

3.4.1. Efectividad de la Estrategia Predictiva

La superioridad del mantenimiento predictivo radica en su capacidad para reducir la incertidumbre sobre el estado de salud de los componentes. Como se ilustra en la Figura 3.3, la efectividad de los distintos tipos de mantenimiento varía significativamente. Mientras que los enfoques reactivos o preventivos tradicionales presentan márgenes de error considerables debido a la variabilidad del uso real, el mantenimiento predictivo logra mantener una eficacia superior al 90 % en la restauración y disponibilidad de los equipos. Esta alta tasa de éxito se debe a que las decisiones se basan en evidencia empírica proveniente de los sensores, permitiendo lo que Chao et al. (2021) describen como la capacidad de determinar, con un tiempo de antelación suficiente, el momento óptimo para la intervención.

Figura 3.3: Comparativa de eficacia entre métodos de mantenimiento.



* Original equipment effectiveness

Source: Deloitte analysis.

Deloitte University Press | dupress.deloitte.com

Nota. Se observa que el enfoque predictivo maximiza la vida útil aprovechada del equipo frente a los métodos preventivos y proactivos. Creado por Coleman et al. (2017).

3.4.2. Conceptualización de la Vida Útil Remanente (RUL)

El centro técnico del PdM es el cálculo de la Vida Útil Remanente (*Remaining Useful Life*, RUL). Conceptualmente, la RUL representa el tiempo, ciclo o distancia restante desde el momento actual de operación hasta que el equipo alcanza un umbral de falla predefinido. Para comprender su cálculo, es necesario entender el proceso de degradación como una Curva de Salud. Inicialmente, un equipo opera en condiciones nominales; sin embargo, debido a factores de carga, fricción o estrés térmico, comienza a experimentar micro-fallas imperceptibles. La RUL busca cuantificar la distancia temporal entre el punto donde se detecta el inicio de la degradación y el punto de falla funcional. La interpretación de este valor es contextual:

- **Aeronáutica:** En motores turbofán, como los del dataset N-CMAPSS, la RUL se mide en *ciclos de vuelo*. Un ciclo completo abarca desde el despegue hasta el aterrizaje.
- **Automoción:** Se cuantifica habitualmente en kilómetros (*km*) restantes.
- **Manufactura:** Puede medirse en horas de operación o unidades producidas.

3.4.3. La RUL como Problema de Modelado Estadístico

Históricamente, la RUL se estimaba mediante modelos de fatiga de materiales o leyes físicas simplificadas. No obstante, en sistemas complejos, la degradación no es lineal ni sigue un patrón único. Factores externos como la altitud, la temperatura ambiental y la velocidad de rotación de un motor generan un ruido que las ecuaciones clásicas no pueden aislar. Es aquí donde el modelado de la RUL se transforma en un problema de regresión de alta dimensionalidad. El objetivo es encontrar una función matemática capaz de mapear múltiples señales de sensores (temperaturas de gases, presiones de salida, flujos

de combustible) hacia un valor escalar de tiempo. Esta complejidad es la que justifica el uso de algoritmos avanzados de aprendizaje supervisado, los cuales pueden identificar patrones de degradación ocultos en los datos que los métodos convencionales ignorarían.

3.5. Modelos de Aprendizaje Supervisado

3.5.1. Extreme Gradient Boosting (XGBoost)

El modelo XGBoost es una implementación avanzada del algoritmo de aumento de gradiente (Gradient Boosting) que ha revolucionado el aprendizaje supervisado para datos tabulares. Su arquitectura se fundamenta en tres pilares: los árboles de decisión, el aprendizaje de conjunto (Ensemble Learning) y la técnica de Boosting.

3.5.1.1. Fundamentos Arquitectónicos

Un árbol de decisión es una estructura lógica que divide el espacio de datos en ramas secuenciales basadas en condiciones específicas, permitiendo realizar predicciones en sus nodos terminales o "hojas". XGBoost no depende de un solo árbol, sino que utiliza el aprendizaje de conjunto, combinando múltiples modelos para reducir el error que tendría un modelo individual. La característica distintiva es el Boosting, un método iterativo y secuencial. A diferencia de otros métodos que entrenan modelos en paralelo, el Boosting construye cada nuevo árbol para corregir los errores residuales de los árboles anteriores. En este proceso, se busca transformar una serie de "aprendices débiles" (modelos con capacidad predictiva limitada) en un aprendiz fuerte de alta precisión.

3.5.1.2. Ventajas Competitivas

Los beneficios de XGBoost se centran en el equilibrio entre el rendimiento predictivo y la eficiencia técnica. Según Brownlee (2016, p. 16), los dos pilares fundamentales que justifican su uso son la velocidad de ejecución y el rendimiento superior del modelo frente a otros algoritmos clásicos. Por su parte, Chen y Guestrin (2016, p. 794), creador del algoritmo,

enfatisa que XGBoost es una solución altamente escalable capaz de resolver problemas complejos del mundo real utilizando una cantidad mínima de recursos computacionales.

3.5.1.3. Formulación Matemática y Optimización

Para garantizar la precisión y evitar el sobreajuste (*overfitting*), XGBoost utiliza una función objetivo que optimiza tanto el ajuste a los datos como la simplicidad del modelo. El algoritmo construye el modelo de forma aditiva, donde en cada paso t se busca minimizar la siguiente función de pérdida:

$$L^{(t)} = \sum_{i=1}^n \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t)$$

Donde los componentes se definen de la siguiente manera:

- $L^{(t)}$: Representa la función de pérdida total para el t -ésimo árbol.
- n es el número total de puntos de datos en el conjunto de entrenamiento.
- g_i y h_i : Son la primera y segunda derivada (gradiente y hessiano), respectivamente, de la función de pérdida con respecto a la predicción anterior. Esto permite que el modelo sea genérico y soporte diversas funciones de pérdida.
- $f_t(x_i)$: Es la estructura del nuevo árbol que se está aprendiendo en la iteración actual.
- $\Omega(f_t)$: Es el término de regularización, un componente crítico que penaliza la complejidad del árbol (número de hojas y magnitud de los pesos) para asegurar que el modelo generalice correctamente ante datos nuevos.

Esta robustez matemática permite que XGBoost sea extremadamente efectivo en tareas de regresión industrial. No obstante, al enfrentarse a secuencias temporales donde la memoria a largo plazo es crucial como la degradación progresiva de un motor turbofán, es

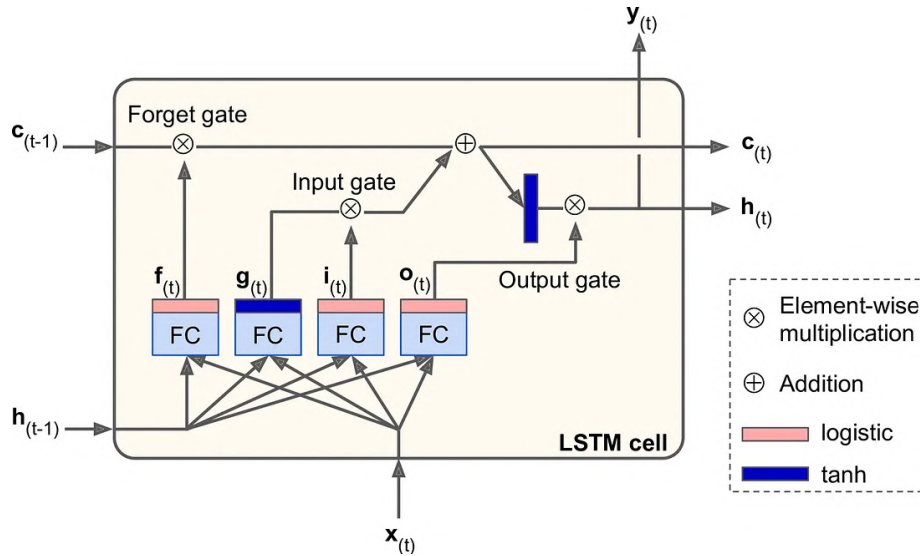
necesario explorar arquitecturas de aprendizaje profundo diseñadas específicamente para el manejo de series de tiempo.

3.5.2. Modelo Long Short-Term Memory (LSTM)

El modelo LSTM surge como una evolución crítica de las Redes Neuronales Recurrentes (RNN) tradicionales. En una RNN estándar, el sistema intenta mantener una memoria de eventos pasados mediante un estado oculto ($\mathbf{h}_{t-1}$). Sin embargo, al procesar secuencias extensas como las series temporales de sensores en un motor, estos modelos sufren del fenómeno de desvanecimiento del gradiente (*vanishing gradient*). Esto implica que la red “olvida” las relaciones de largo plazo, priorizando únicamente los datos más recientes y perdiendo la capacidad de capturar la degradación gradual del sistema.

Para solventar esta limitación, Hochreiter y Schmidhuber introdujeron la arquitectura LSTM, la cual añade una “celda de estado” ($\mathbf{C}_t$) que funciona como una cinta transportadora de información a largo plazo, protegida y regulada por mecanismos llamados compuertas (gates).

Figura 3.4: Estructura interna de una celda de memoria LSTM.



Nota. La arquitectura muestra la interacción entre el estado de la celda ($\mathbf{C}_t$) y el estado oculto ($\mathbf{h}_t$) a través de las compuertas de control. Adaptado de Géron (2018).

3.5.2.1. Mecanismos de Control: Las Compuertas

La capacidad de la LSTM para recordar u olvidar información de manera selectiva se basa en tres compuertas principales, cada una con una función matemática específica:

1. **Compuerta de Olvido (*Forget Gate*, f_t):** Determina qué proporción de la memoria anterior (C_{t-1}) ya no es relevante y debe ser descartada.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

2. **Compuerta de Entrada (*Input Gate*, i_t):** Decide qué nueva información proveniente del sensor actual (x_t) es lo suficientemente importante como para ser incorporada al estado de la celda. Se combina con un candidato a memoria ($\tilde{C}_t$) generado mediante una función tangente hiperbólica.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

3. **Compuerta de Salida (*Output Gate*, o_t):** Regula qué parte de la memoria interna de la celda se filtrará para formar el nuevo estado oculto ($\mathbf{h}_t$), que es el valor que finalmente se utiliza para la predicción de la RUL.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \cdot \tanh(C_t)$$

3.5.2.2. Funciones de Activación y su Rol Crítico

El control del flujo de información dentro de estas compuertas depende de dos funciones de activación con propiedades matemáticas distintas:

- **Función Sigmoide (σ):** Al mapear cualquier valor de entrada a un rango estricto entre **0** y **1**, actúa como un interruptor analógico. Un valor cercano a 0 representa

un “bloqueo” total de la información (olvido), mientras que un valor cercano a 1 representa un “paso libre” (recuerdo total).

- **Tangente Hiperbólica ($\tanh$):** Al oscilar entre **-1 y 1**, se utiliza para regular los valores que atraviesan la red, evitando que las magnitudes de los datos crezcan de forma incontrolada durante las operaciones matemáticas sucesivas y permitiendo modelar la dirección (aumento o disminución) de la degradación.

Esta arquitectura permite que la red mantenga un historial coherente de la salud de cualquier activo a estudiar, identificando patrones de desgaste que ocurren en periodos de tiempo prolongados.

3.6. Entorno de Simulación N-CMAPSS

Para evaluar la capacidad predictiva y la robustez de los modelos XGBoost y LSTM en condiciones cercanas a la operación real, esta investigación utiliza el conjunto de datos **N-CMAPSS** (*New Commercial Modular Aero-Propulsion System Simulation*). El modelo dinámico CMAPSS es una herramienta informática de alta fidelidad desarrollada por la NASA para la simulación de motores turbofán comerciales de gran tamaño. A diferencia de versiones anteriores, el N-CMAPSS permite modelar la degradación bajo perfiles de vuelo realistas y condiciones operativas variables.

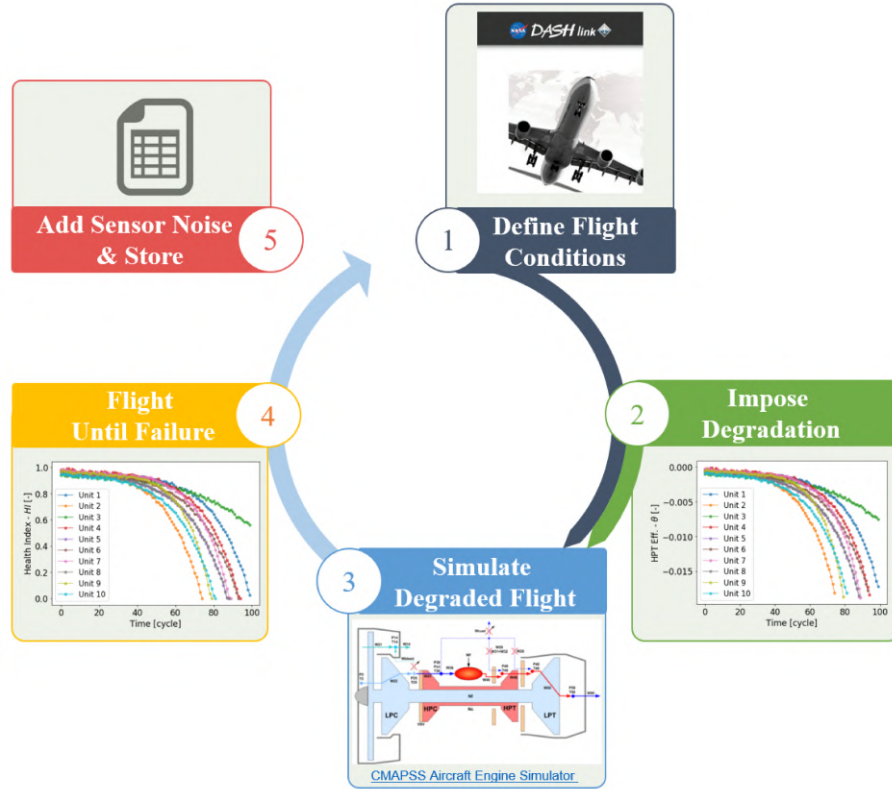
3.6.1. Proceso de Construcción de las Bases de Datos

La generación de las bases de datos (BBDD) en N-CMAPSS sigue un flujo lógico que integra datos empíricos con modelos de degradación física. Como se detalla en la Figura [3.5](#), el proceso comienza con el registro de condiciones de vuelo reales (altitud, velocidad, temperatura ambiente). Posteriormente, se induce una degradación controlada en los componentes del motor para estudiar sus efectos mediante una simulación que registra el estado

3.6 Entorno de Simulación N-CMAPSS

de salud del activo desde un estado óptimo hasta el punto de falla funcional. Este enfoque permite generar un historial completo de degradación que es vital para el entrenamiento de modelos supervisados.

Figura 3.5: Flujo de construcción de BBDD mediante el modelo N-CMAPSS.



Nota. Representación del ciclo de integración entre condiciones reales y degradación simulada. Creado por Chao et al. (2021).

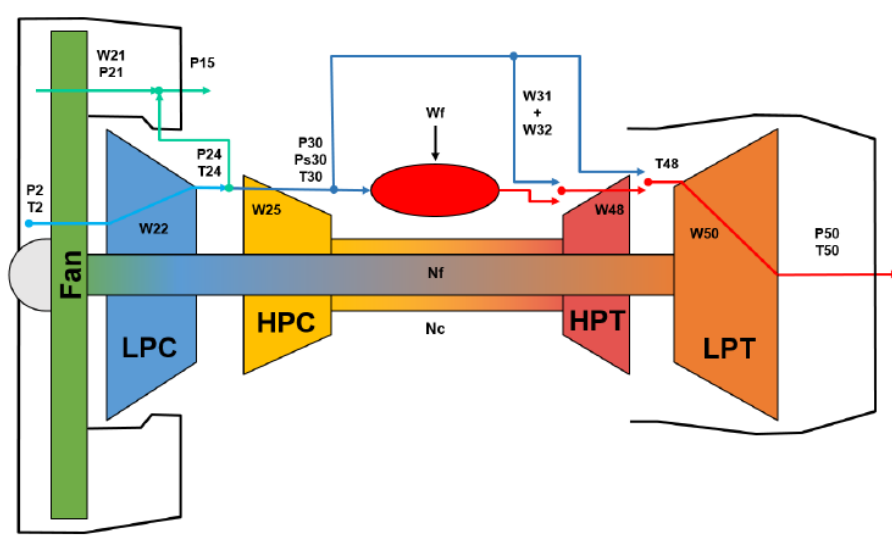
3.6.2. Arquitectura y Componentes de Interés

El sistema mecánico simulado es un motor turbofán complejo. Para los fines de esta investigación, el análisis se centra en los componentes donde la degradación es más crítica y difícil de predecir debido a las altas presiones y temperaturas:

- **HPT (*High-Pressure Turbine*):** Turbina de alta presión.
- **LPT (*Low-Pressure Turbine*):** Turbina de baja presión.

La Figura 3.6 ilustra la ubicación de estos componentes y la disposición de los sensores virtuales y físicos que capturan el comportamiento dinámico del sistema durante el proceso de desgaste.

Figura 3.6: Esquema de componentes y sensores en la turbina simulada.



Nota. Localización de los puntos de medición de presión y temperatura en los módulos HPT y LPT. Adaptado de Chao et al. (2021).

3.6.3. Selección de Subconjuntos de Datos

El ecosistema N-CMAPSS ofrece múltiples subconjuntos con variaciones en tamaño y complejidad (ver Figura 3.7). Para esta tesis, se han seleccionado tres escenarios específicos que permiten poner a prueba la robustez de los modelos ante diferentes niveles de dificultad:

- **DS01 (Punto de Referencia):** Constituye el subconjunto de menor complejidad operativa, con un único modo de falla y condiciones de vuelo homogéneas. Su inclusión en el análisis permite establecer una línea base de desempeño bajo las condiciones más controladas del ecosistema N-CMAPSS, facilitando la interpretación del impacto que la complejidad creciente de DS02 y DS03 ejerce sobre la robustez de los modelos.

3.6 Entorno de Simulación N-CMAPSS

- **DS02 (Análisis de Robustez):** Este subconjunto modela dos modos de degradación independientes. Se utilizará para evaluar la capacidad inherente de XGBoost y LSTM para diferenciar patrones de falla implícitos sin necesidad de modificar la arquitectura de los modelos.
- **DS03 (Volumen y Estabilidad ante Ruido):** Este subconjunto presenta una configuración de componentes equivalente a DS02 (HPT y LPT), pero se limita a modelar un único modo de falla. Si bien esto simplifica la naturaleza del defecto, DS03 contiene una cantidad de datos significativamente mayor. Este escenario permite evaluar si el incremento en el volumen de información robustece la predicción o si, por el contrario, induce a una complejidad innecesaria en los modelos.

Figura 3.7: Matriz comparativa de subconjuntos en el entorno N-CMAPSS.

Name	# Units	Flight Classes	Failure Modes	Fan		LPC		HPC		HPT		LPT		Size
				E	F	E	F	E	F	E	F	E	F	
DS01	10	1, 2, 3	1							✓				7.6 M
DS02	9	1, 2, 3	2							✓		✓	✓	6.5 M
DS03	15	1, 2, 3	1							✓		✓	✓	9.8 M
DS04	10	2, 3	1	✓	✓									10.0 M
DS05	10	1, 2, 3	1					✓	✓					6.9 M
DS06	10	1, 2, 3	1			✓	✓	✓	✓					6.8 M
DS07	10	1, 2, 3	1									✓	✓	7.2 M
DS08	54	1, 2, 3	1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	35.6 M

Nota. Detalle de motores, ciclos y modos de falla por cada unidad DS. Fuente: Chao et al. (2021).

3.6.4. Taxonomía de Variables y Sensores

Las señales capturadas en N-CMAPSS se categorizan según su origen y función dentro del modelo predictivo. Es crucial distinguir entre aquellas que aportan conocimiento del sistema y aquellas que podrían sesgar el aprendizaje.

1. **Sensores Físicos (x_s):** Mediciones directas de velocidad, temperatura y presión (ver Figura 3.8). Representan los datos que estarían disponibles en un entorno operativo real.

2. **Sensores Virtuales (x_v):** Estimaciones derivadas de la simulación que no son accesibles mediante instrumentación estándar en motores reales, pero que aportan información crítica para caracterizar el comportamiento interno (ver Figura 3.9).
3. **Parámetros de Salud (T):** Describen el estado interno (eficiencia y flujo) de los componentes (ver Figura 3.10). Son la base para modelar el desgaste temporal.
4. **Datos Auxiliares (A):** Metadatos como el ID de unidad y tipo de vuelo. Se destaca la variable h_s (*Health State*); sin embargo, para evitar el sesgo de información privilegiada (*data leakage*), esta variable será excluida. Este fenómeno ocurre cuando el modelo tiene acceso durante el entrenamiento a información que no estaría disponible en un entorno de predicción real. En términos prácticos, si el modelo aprende a predecir la RUL usando directamente el estado de salud del motor, una variable que en operación real requeriría desmontaje físico para medirse, el sistema aprendería a copiar la respuesta en lugar de inferirla desde los sensores. Por esta razón, la variable h_s será excluida del entrenamiento de los modelos, obligando a los algoritmos a aprender la degradación únicamente desde las señales observables de los sensores (ver Figura 3.11).

A continuación se especifican las variables de cada categoría, junto con una descripción de cada una de ellas:

3.6 Entorno de Simulación N-CMAPSS

Figura 3.8: Sensores físicos (x_s).

#	Nomenclatura	Descripción	Unidad
1	Wf	Flujo de combustible	pps
2	Nf	Velocidad física del ventilador	rpm
3	Nc	Velocidad del núcleo físico	rpm
4	T24	Temperatura total a la salida LPC	°R
5	T30	Temperatura total en la salida HPC	°R
6	T48	Temperatura total a la salida HPT	°R
7	T50	Temperatura total a la salida INT	°R
8	P15	Presión total en el conducto de derivación	psia
9	P2	Presión total en la entrada del ventilador	psia
10	P21	Presión total a la salida del ventilador	psia
11	P24	Presión total en la salida LPC	psia
12	Ps30	Presión estática en la salida HPC	psia
13	P40	Presión total a la salida del quemador	psia
14	P50	Presión total en la salida LPT	psia

Nota. Detalle de los sensores físicos.

Figura 3.9: Sensores virtuales (x_v).

#	Nomenclatura	Descripción	Unidad
1	T40	Temperatura total a la salida del quemador	°R
2	P30	Presión total en la salida HPC	psia
3	P45	Presión total en la salida HPT	psia
4	W21	Flujo de ventiladores	pps
5	W22	Flujo de salida de LPC	lbm/s
6	W25	Flujo hacia HPC	lbm/s
7	W31	Purga de refrigerante HPT	lbm/s
8	W32	Purga de refrigerante HPT	lbm/s
9	W48	Flujo de salida de HPT	lbm/s
10	W50	Flujo de salida de LPT	lbm/s
11	SmFan	Margen de pérdida del ventilador	-
12	SmLPC	Margen de pérdida de LPC	-
13	SmHPC	Margen de pérdida de HPC	-
14	phi	Relación entre el flujo de combustible y Ps30	pps/psi

Nota. Detalle de los sensores virtuales.

3.7 Método de Evaluación de Modelos

Figura 3.10: Parámetros de salud del motor (T).

#	Nomenclatura	Descripción	Unidad
1	fan_eff_mod	Modificador de eficiencia del ventilador	-
2	fan—flow—mod	Modificador de flujo de ventilador	-
3	LPC_eff_mod	Modificador de eficiencia LPC	-
4	LPC flow mod	Modificador de flujo LPC	-
5	HPC eff mod	Modificador de eficiencia HPC	-
6	HPC_flow_mod	Modificador de flujo HPC	-
7	HPT_eff_mod	Modificador de eficiencia HPT	-
8	HPT_flow_mod	Modificador de flujo HPT	-
9	LPT_eff_mod	Modificador de eficiencia LPT	-
10	LPT flow mod	Modificador de flujo HPT	-

Nota. Detalle de los parámetros de salud de los motores.

Figura 3.11: Datos auxiliares (A).

#	Nomenclatura	Descripción	Unidad
1	unit	Número de unidad	-
2	cycle	Número de ciclo de vuelo	-
3	Fc	Clase de vuelo	-
4	lis	Estado de salud	-

Nota. Detalle de los datos auxiliares.

3.7. Método de Evaluación de Modelos

En el ámbito del *Machine Learning*, la elección de métricas de evaluación es fundamental para medir el desempeño de los modelos predictivos. En el caso de problemas de regresión, como la predicción de la *Remaining Useful Life* (RUL) de un activo, dos de las métricas más utilizadas son el Error Absoluto Medio (MAE, por sus siglas en inglés: *Mean Absolute Error*) y el Error Cuadrático Medio (RMSE, *Root Mean Squared Error*).

El MAE mide el error promedio en términos absolutos, es decir, la magnitud promedio de las diferencias entre los valores predichos y los valores reales. Su expresión matemática se define como:

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

donde y_i corresponde al valor real, $\hat{y}_i$ al valor predicho por el modelo y n al número total

3.7 Método de Evaluación de Modelos

de observaciones.

Por su parte, el RMSE otorga un mayor peso a los errores de mayor magnitud, al elevar al cuadrado las diferencias antes de promediar y luego calcular la raíz cuadrada. Esto lo convierte en una métrica más sensible a valores atípicos. Su formulación es:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

Ambas métricas se utilizan de manera complementaria. Mientras el MAE proporciona una medida intuitiva de los errores promedio, el RMSE enfatiza la penalización de grandes desviaciones, lo que permite evaluar la robustez del modelo frente a errores significativos.

4

Metodología

La presente investigación se fundamenta en un diseño experimental de carácter cuantitativo y comparativo. El objetivo es determinar la resiliencia de dos arquitecturas de aprendizaje automático, XGBoost y LSTM, ante la degradación controlada de la disponibilidad de datos, simulando restricciones de información.

4.1. Selección y Caracterización de los Datos

Para asegurar la validez de los resultados en escenarios de alta complejidad, se utilizaron los subconjuntos DS01, DS02 y DS03 del *dataset* N-CMAPSS de la NASA. Estos datos representan señales multivariadas de sensores en motores turbofán, donde la relación entre las variables y la degradación es no lineal y ruidosa.

4.2. Preprocesamiento y Preparación de Datos

4.2.1. Generación de Ventanas Temporales y RUL Rectificada

Dado que los datos originales consisten en series de tiempo, se aplicó una técnica de ventana deslizante (Sliding Window) de longitud fija. Este proceso transforma las señales continuas en muestras discretas que capturan la dinámica temporal del motor. Para la variable objetivo (target), se implementó una RUL Rectificada: la RUL se mantiene constante en su valor máximo durante el 5 % inicial de la vida útil de cada motor y comienza

4.2 Preprocesamiento y Preparación de Datos

a descender linealmente a partir de ese punto hasta llegar a cero en el fallo. Si bien el comportamiento real del RUL no es estrictamente lineal, las variaciones son despreciables a nivel global y esta aproximación es suficiente para capturar la trayectoria de degradación sin sesgar el aprendizaje. Este ajuste previene que el modelo converja prematuramente a una solución trivial basada en el conteo de ciclos, forzándolo a identificar los cambios en las señales de los sensores que preceden a la degradación física.

Adicionalmente, la variable de conteo de ciclos fue excluida del conjunto de entrenamiento. Al crecer monótonicamente con el tiempo, su inclusión introduciría una correlación directa entre el índice temporal y la RUL predicha, enmascarando las relaciones entre sensores y desgaste que los modelos deben aprender.

4.2.2. Estrategia de Reducción de Datos (Simulación de Escasez)

Para cumplir con el objetivo de evaluar la robustez ante la escasez de información, se implementó un protocolo de submuestreo aleatorio a nivel de ventanas. Una vez generadas las ventanas temporales para cada motor, estas se consolidan en un conjunto global del cual se extraen muestras aleatorias para reducir el volumen de entrenamiento en cuatro niveles: 20 %, 40 %, 60 % y 80 %. No obstante, se reconoce que dada la gran magnitud del conjunto de datos N-CMAPSS, es posible que estas reducciones iniciales no sean suficientes para evidenciar una divergencia crítica en el desempeño de los modelos o para alcanzar un estado de escasez real. Por consiguiente, en las etapas de análisis profundo de esta investigación, se contempla la necesidad de reducir la base de datos a niveles aún más drásticos. Esta consideración adicional permitirá identificar con precisión el punto de quiebre de cada arquitectura y estudiar las diferencias de rendimiento bajo condiciones de extrema limitación de información.

4.3. Implementación y Entrenamiento de los Modelos

4.3.1. Entrenamiento Incremental de XGBoost

Debido a la magnitud del *dataset* N-CMAPSS, el modelo XGBoost se implementó mediante un esquema de entrenamiento incremental. Este proceso se realiza a través de un generador que entrega lotes (batches) de datos, permitiendo que el modelo se actualice secuencialmente mediante el parámetro *xgb_model*. La lógica de actualización se define como:

$$M_t = \mathcal{T}(M_{t-1}, \mathcal{B}_t)$$

Donde los componentes de la expresión se describen a continuación:

- M_t : Representa el **estado del modelo** en la iteración actual t . Es el resultado final tras haber procesado el nuevo lote de información y haberlo integrado con el conocimiento previo.
- M_{t-1} : Corresponde al **estado del modelo en la iteración anterior**. En términos de Machine Learning, representa el “conocimiento acumulado” o la estructura de árboles ya construida hasta ese punto.
- $\mathcal{B}_t$: Es el **lote de datos** (*batch*) ingresado en el paso t . Contiene las ventanas temporales de los sensores que el modelo debe procesar para identificar nuevos patrones de degradación.
- $\mathcal{T}$: Representa el **operador de entrenamiento** o transformación. Es la función que toma el modelo base y los nuevos datos para generar una actualización del gradiente, permitiendo que XGBoost aprenda de forma aditiva.

- t : Es el **índice de iteración**, que marca la progresión del tiempo o la secuencia en la que los motores son procesados por el algoritmo.

Este enfoque permite manejar eficientemente la memoria RAM disponible para el entrenamiento de modelos complejos y optimizar los hiperparámetros de forma escalable.

4.3.2. Modelo de Aprendizaje Profundo (LSTM)

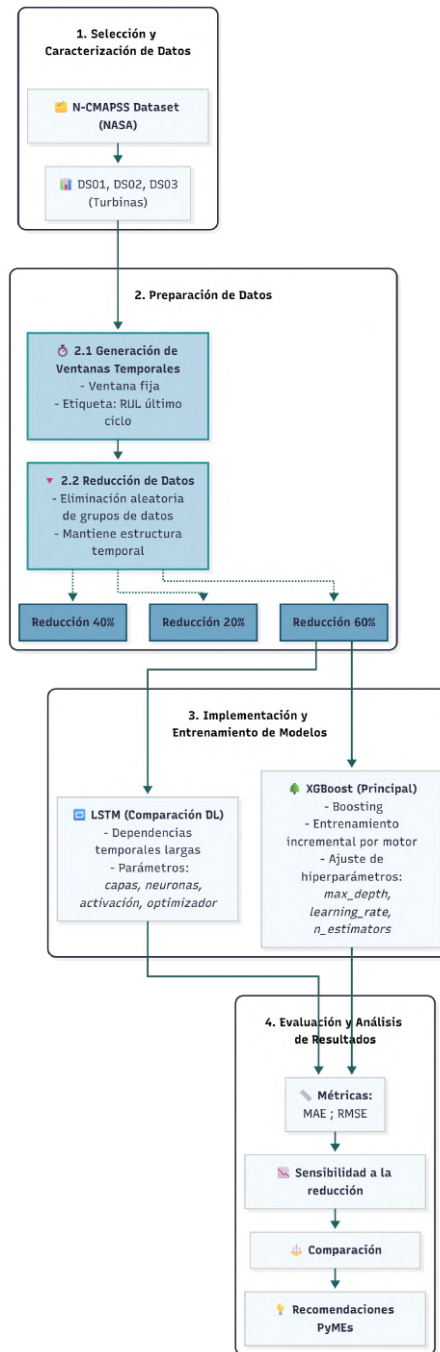
Como contraparte, se diseñó una red LSTM para capturar las dependencias secuenciales de las ventanas de datos. A diferencia de XGBoost, el modelo LSTM procesa la ventana temporal como una estructura tridimensional (*muestras, pasos_de_tiempo, caracterstias*), lo que teóricamente le otorga una ventaja en la interpretación de tendencias. El entrenamiento se realiza bajo las mismas condiciones de reducción de datos para garantizar una comparación equitativa.

4.4. Métricas de Evaluación

La precisión de los modelos se cuantifica mediante la comparación de la RUL predicha ($\hat{y}$) frente a la RUL real (y) utilizando:

- **Error Absoluto Medio (MAE)**: Indica la magnitud promedio del error en ciclos.
- **Raíz del Error Cuadrático Medio (RMSE)**: Penaliza de forma más severa las desviaciones grandes, siendo un indicador de la estabilidad del modelo.

Figura 4.1: Diagrama de Metodología.



Nota. Flujo de Metodología.

5

Implementación y Arquitectura de Modelos

El diseño y la implementación de los modelos predictivos están intrínsecamente ligados a las restricciones del entorno computacional y a las herramientas de software disponibles. En este capítulo se detallan los factores técnicos que condicionaron el entrenamiento y las arquitecturas seleccionadas para XGBoost y LSTM.

5.1. Limitaciones Computacionales y Estrategias de Mitigación

La viabilidad de entrenar modelos de *Machine Learning* con bases de datos masivas como N-CMAPSS depende críticamente de los recursos de hardware. El desarrollo de esta investigación se llevó a cabo bajo las siguientes especificaciones:

- **CPU:** AMD Ryzen 7 4800H (Procesamiento lógico y partición de lotes).
- **RAM:** 24 GB a 3200 MT/s (Cuello de botella principal para la carga de datos).
- **GPU:** NVIDIA GTX 1650 con 4 GB de VRAM (Aceleración de tensores para LSTM).

Dada la restricción de memoria RAM, fue imperativo adoptar estrategias de mitigación. La principal medida fue la **reducción de precisión numérica**, transformando los da-

tos originales de las BBDD de formato `Float64/Int64` a `Float32/Int32`. Se determinó que la pérdida de precisión decimal es marginal frente al beneficio de maximizar la cantidad de ventanas deslizantes procesables por lote, logrando cargar hasta 70.000 ventanas simultáneas sin provocar un desbordamiento de memoria.

5.2. Ecosistema de Software y Librerías

Para estandarizar el desarrollo y asegurar la reproducibilidad del experimento, se construyó un ecosistema basado en Python utilizando las siguientes librerías estándar:

- **h5py:** Fundamental para la ingesta de las BBDD, las cuales están estructuradas en formato HDF5 (*Hierarchical Data Format*), permitiendo una extracción rápida y organizada de tensores masivos.
- **Scikit-learn (Sklearn) y NumPy:** Utilizadas para el manejo matricial, escalado estandarizado de características (crucial para evitar sesgos por la magnitud de los sensores) y la partición de conjuntos.
- **XGBoost:** Librería que provee la implementación optimizada del algoritmo de *Gradient Boosting*, permitiendo el ajuste granular de hiperparámetros y el entrenamiento iterativo.
- **PyTorch (torch):** *Framework* principal utilizado para construir la arquitectura profunda del modelo LSTM, aprovechando su diferenciación automática y aceleración por GPU mediante los módulos `torch.nn` y `torch.optim`.

5.3. Configuración Arquitectónica de los Modelos

5.3.1. XGBoost: Aprendizaje Incremental y Aplanamiento de Datos

Para gestionar la restricción de 24 GB de RAM, XGBoost se configuró bajo un esquema de aprendizaje incremental. El modelo no procesa el conjunto histórico de una sola vez, sino

5.3 Configuración Arquitectónica de los Modelos

que recibe lotes iterativos. Dado que los árboles generados en lotes anteriores quedan fijos, el modelo se ve forzado a crear nuevos árboles aditivos, incrementando la complejidad del ensamble y el riesgo de sobreajuste. Por ello, y tal como se detalló anteriormente en las limitaciones del modelo, el tamaño del lote se fijó en el umbral máximo de 70.000 ventanas. Esta decisión busca otorgarle a XGBoost el mayor contexto posible por iteración, minimizando la fragmentación del aprendizaje.

Estructuralmente, XGBoost requiere datos tabulares bidimensionales. Por lo tanto, cada ventana deslizante multivariante (60 pasos de tiempo $\times$ número de sensores) debió ser transformada mediante una operación de **aplanamiento** (*Flattening* o **reshape** a 1D). Esta transformación es crítica, ya que implica que el modelo procesa la secuencia sin una noción topológica estricta del orden temporal, tratando cada lectura histórica como una característica independiente.

5.3.2. LSTM: Arquitectura Profunda y Secuencialidad

A diferencia del enfoque tabular bidimensional de XGBoost, la red LSTM fue diseñada nativamente para ingerir los datos como una estructura matricial de tres dimensiones (comúnmente denominada tensor). Esta estructura posee dimensiones $(N, 60, 42)$, donde N representa el tamaño del lote (*batch size*), 60 es la longitud de la secuencia temporal (*look-back*) y 42 son las características extraídas de los sensores.

Para procesar esta información y preservar el contexto histórico de la degradación, la arquitectura se estructuró mediante los siguientes bloques lógicos:

1. **Bloque Recurrente (*Stacked LSTM*) y Normalización:** El núcleo temporal del modelo consiste en dos capas LSTM apiladas.

- La primera capa LSTM recibe las 42 características/variables por paso de tiempo y las proyecta a un espacio oculto de 100 unidades (equivalentes a 100 neu-

5.3 Configuración Arquitectónica de los Modelos

ronas o celdas de memoria virtual). Inmediatamente después, se aplica una Normalización por Capa (*Layer Normalization*). En el entrenamiento de redes profundas, variaciones muy grandes en los datos pueden hacer que la red actualice sus pesos matemáticos de forma errática, un problema conocido como inestabilidad de gradientes. Esta normalización estandariza estadísticamente las señales dentro de la capa, asegurando que la red aprenda de manera estable y uniforme.

- La segunda capa LSTM recibe esta secuencia normalizada y refina las representaciones manteniendo la dimensionalidad de 100 neuronas. Al igual que en la etapa anterior, su salida es procesada por una segunda capa de normalización.

2. **Regularización por *Dropout*:** Una vez procesada toda la secuencia recurrente, se aplica una única capa de *Dropout* con una tasa moderada del 20 % ($p = 0,2$). En términos prácticos, esta técnica “apaga” o desactiva aleatoriamente el 20 % de las conexiones neuronales en cada ciclo de entrenamiento. Al forzar a que el modelo no pueda depender de un grupo específico de neuronas para tomar sus decisiones, se promueve que toda la red aprenda a identificar los patrones de falla, previniendo eficazmente el sobreajuste (*overfitting*) a los datos de entrenamiento.
3. **Extracción del Estado Final:** El modelo realiza una operación de compresión temporal extrayendo únicamente el estado oculto del último paso de tiempo ($t = 60$). Este vector bidimensional $(N, 100)$ condensa el aprendizaje histórico de toda la ventana deslizando.
4. **Cabeza de Decodificación Densa (*Fully Connected*):** La predicción final se construye a través de una red neuronal profunda (*feed-forward*) de tres capas lineales

5.3 Configuración Arquitectónica de los Modelos

que reducen la dimensionalidad de forma progresiva. Actuando como un “embudo” de información, la arquitectura se define así:

- La primera capa lineal comprime el vector de 100 unidades a 64 unidades.
- La segunda capa reduce de 64 unidades a 32 unidades.
- Finalmente, la tercera capa proyecta las 32 unidades hacia un único valor escalar ($output = 1$), que es la estimación de la RUL.

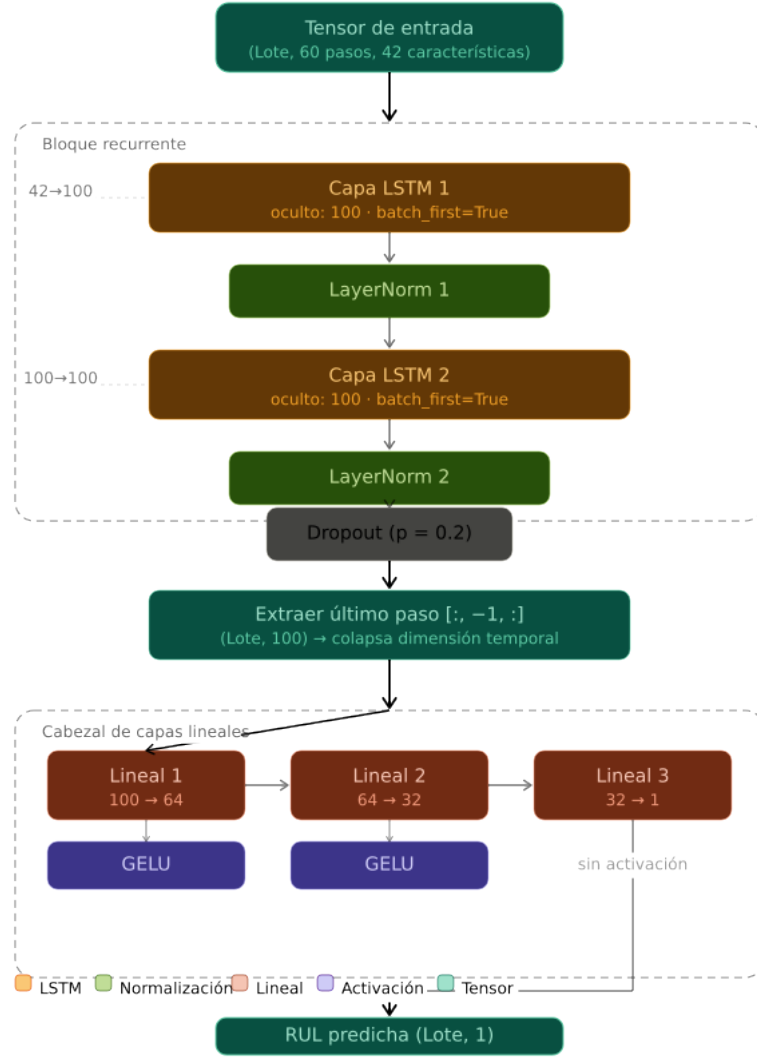
El impacto de esta reducción progresiva es crucial, ya que permite que la red consolide las características abstractas extraídas por las LSTM de forma gradual. Una reducción brusca (por ejemplo, de 100 directamente a 1) causaría una pérdida masiva de información relacional. Los pasos intermedios otorgan a la red la capacidad de ponderar con precisión qué combinaciones de sensores son los verdaderos indicadores de la falla.

5. **Activación Suavizada (GELU):** Una función de activación es el mecanismo matemático que decide si una neurona debe “activarse” (transmitir información) o permanecer inactiva. Entre las transiciones de las capas densas, se aplicó la función **GELU** (*Gaussian Error Linear Unit*). A diferencia de funciones más antiguas que simplemente bloquean los números negativos cortándolos a cero de forma tajante, GELU pondera los datos basándose en una curva de probabilidad matemática continua y suave. Esta suavidad es altamente útil, ya que permite que la red interprete fluctuaciones y desgastes muy sutiles en la degradación de los sensores sin perder información crítica por cortes abruptos, mejorando la sensibilidad predictiva del modelo.

A continuación se podrá visualizar un esquema del modelo LSTM con sus respectivas partes ya mencionadas.

5.3 Configuración Arquitectónica de los Modelos

Figura 5.1: Estructura del modelo LSTM implementado.



Nota. El diagrama muestra la estructura del modelo LSTM indicando también los cambios en cada etapa.

5.3.3. Resumen de Hiperparámetros

Las Tablas 5.1 y 5.2 consolidan la configuración definitiva utilizada para el entrenamiento de ambas arquitecturas. Estos valores se mantienen constantes en todos los escenarios de reducción de datos para garantizar que las diferencias en desempeño observadas sean atribuibles exclusivamente al volumen de datos disponible, y no a asimetrías en la confi-

5.3 Configuración Arquitectónica de los Modelos

guración de los modelos.

Cuadro 5.1: Hiperparámetros del modelo XGBoost

Hiperparámetro	Valor	Rol
<code>eta</code> (tasa de aprendizaje)	0.05	Contracción del paso de actualización
<code>max_depth</code>	6	Profundidad máxima de cada árbol
<code>subsample</code>	0.7	Fracción de muestras usadas por árbol
<code>colsample_bytree</code>	0.7	Fracción de características por árbol
λ (L2)	1	Regularización de pesos tipo ridge
α (L1)	0	Regularización de pesos tipo lasso
Función de pérdida	<code>reg:squarederror</code>	Error cuadrático medio
Árboles por lote (<i>rounds/chunk</i>)	100	Complejidad acumulada por iteración
Tamaño de lote	70000 ventanas	Máximo sin desbordamiento de RAM

Nota. Se definen todos los hiperparámetros del modelo XGBoost, junto con la función de pérdida y el optimizador utilizado para entrenar el modelo.

5.3 Configuración Arquitectónica de los Modelos

Cuadro 5.2: Hiperparámetros del modelo LSTM

Hiperparámetro	Valor	Rol
Longitud de ventana (<i>look-back</i>)	60 pasos de tiempo	Contexto histórico por muestra
Características de entrada	42	Variables de sensores seleccionadas
Unidades ocultas por capa LSTM	100	Dimensión del estado oculto
Capas LSTM apiladas	2	Profundidad del bloque recurrente
<i>Dropout</i>	0.2	Fracción de conexiones desactivadas
Arquitectura densa	$100 \rightarrow 64 \rightarrow 32 \rightarrow 1$	Decodificador de RUL
Función de activación	GELU	Entre capas densas
Función de pérdida	MSE	Error cuadrático medio
Optimizador	Adam	Descenso de gradiente adaptativo
Tasa de aprendizaje	1×10^{-4}	Factor de actualización de pesos
<i>Weight decay</i> (L2)	1×10^{-5}	Regularización del optimizador
Épocas máximas	60	Ciclos de entrenamiento completos
Lotes por época	1000	Actualizaciones de pesos por epoch

Nota. Se definen todos los hiperparámetros del modelo LSTM, junto con la función de pérdida y el optimizador utilizado para entrenar el modelo.

En síntesis, las decisiones de configuración de ambas arquitecturas responden a un equilibrio entre capacidad expresiva y control de complejidad. En XGBoost, la profundidad máxima de árbol fijada en 6 permite capturar relaciones no lineales entre los sensores sin generar particiones tan específicas que deriven en ruido o sobreajuste; en LSTM, la ventana temporal de 60 pasos de tiempo otorga al modelo el contexto histórico necesario para detectar tendencias de degradación graduales antes de que sean visibles en lecturas individuales. En ambos casos, el entrenamiento por lotes fue la estrategia que permitió operar dentro de las restricciones de memoria disponibles, aunque con consecuencias distintas: en LSTM los lotes son una técnica convencional sin efecto en la arquitectura del modelo; en XGBoost, el esquema incremental fija permanentemente los árboles de cada iteración. Bajo este diseño, el siguiente capítulo evalúa empíricamente cuál de estas dos filosofías ofrece una mayor resiliencia ante la reducción sistemática de los datos de entrenamiento.

6

Desempeño de los Modelos en las Distintas BBDD

Este capítulo constituye el núcleo experimental de la investigación, donde se evalúa de forma aislada la resiliencia de cada arquitectura ante la escasez de información. Los resultados aquí presentados proporcionan la evidencia empírica necesaria para dar respuesta a la pregunta de investigación y cumplir con los objetivos específicos de cuantificar la sensibilidad y determinar los umbrales de robustez técnica de los modelos XGBoost y LSTM.

6.1. Resultados obtenidos

El objetivo central de esta sección es evaluar la capacidad predictiva de los modelos en las tres bases de datos (DS01, DS02, y DS03). La comparación se realiza manteniendo constantes todos los hiperparámetros utilizados para cada modelo, lo que permite una evaluación técnica estandarizada y libre de sesgos por optimización diferencial.

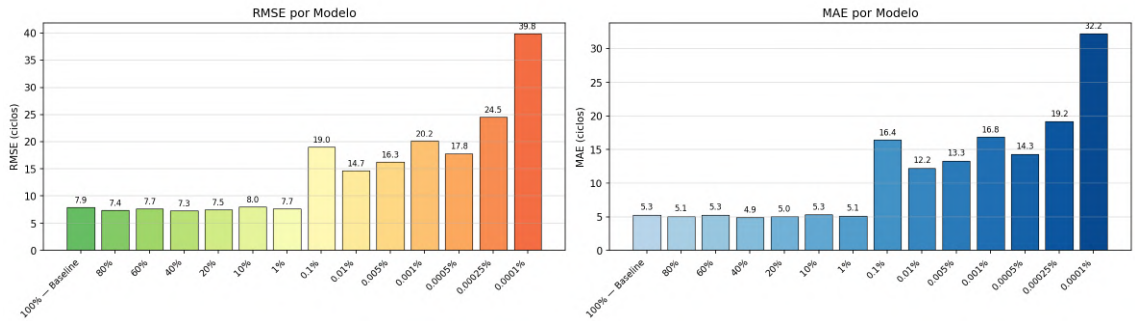
A continuación se presentarán los resultados logrados tanto por los modelos LSTM como por los modelos XGBoost elaborados, analizando su sensibilidad bajo escenarios de datos limitados.

6.1.1. Resultados LSTM

En los gráficos que se verán a continuación (Figuras 6.1, 6.2 y 6.3) se podrá observar el desempeño en RMSE y MAE de los modelos LSTM entrenados en las distintas bases de datos (BBDD) y versiones de aquellos conjuntos que corresponden a reducciones de estos mismos.

En los tres gráficos se puede apreciar cómo los modelos bajo la arquitectura LSTM presentan un comportamiento similar a pesar de operar con distintas fuentes de datos. El desempeño tanto en RMSE como en MAE se mantiene relativamente estable y exacto hasta la instancia donde los modelos solamente tienen a disposición el 1% de los datos para entrenamiento. Este umbral representa la “barrera de datos” mínima para que la red neuronal profunda mantenga una precisión operativa coherente.

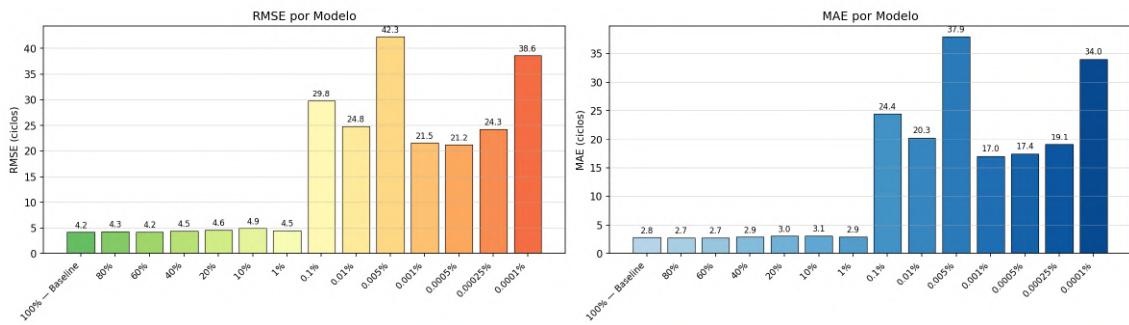
Figura 6.1: Comparación de RMSE y MAE de modelos LSTM para BBDD DS01.



Nota. Gráfico que muestra el desempeño RMSE y MAE logrado por modelos LSTM para distintos tamaños de la BBDD DS01.

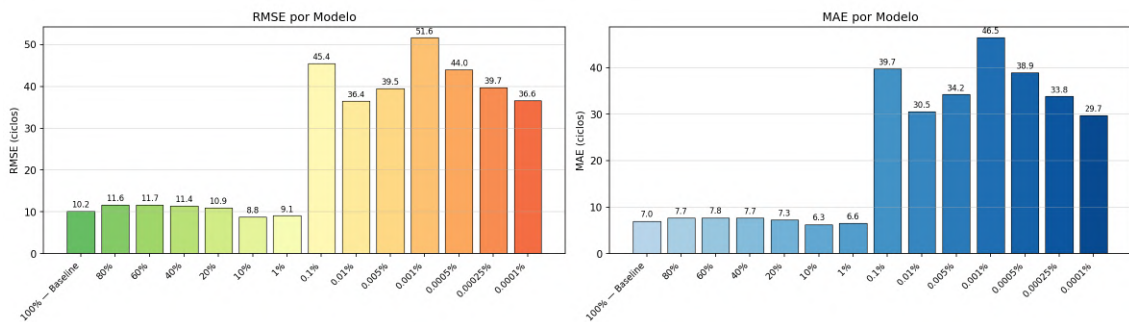
6.1 Resultados obtenidos

Figura 6.2: Comparación de RMSE y MAE de modelos LSTM para BBDD DS02.



Nota. Gráfico que muestra el desempeño RMSE y MAE logrado por modelos LSTM para distintos tamaños de la BBDD DS02.

Figura 6.3: Comparación de RMSE y MAE de modelos LSTM para BBDD DS03.



Nota. Gráfico que muestra el desempeño RMSE y MAE logrado por modelos LSTM para distintos tamaños de la BBDD DS03.

Los modelos logran un desempeño exacto, destacándose el comportamiento de la arquitectura LSTM en el conjunto DS02 (6 motores de entrenamiento y 3 de prueba), a pesar de que esta base de datos constituía un mayor desafío debido a que, a diferencia de DS01 y DS03, este contaba con dos modos de falla. También se destaca que se cumple lo mencionado en el marco teórico: aun cuando la BBDD DS03 modela un único modo de degradación (frente a los dos presentes en DS02), el hecho de que este conjunto posea una flota de entrenamiento 50 % más grande (9 motores vs 6) conllevó que los modelos bajo la arquitectura LSTM tuvieran un desempeño considerablemente inferior. Esta divergencia se explica por el balance entre la capacidad de aprendizaje y la variabilidad evaluada: al integrar una mayor cantidad de activos independientes, se produce una “interferencia de patrones” donde los matices operativos individuales de cada motor dificultan que la red consolide una firma de falla única y universal. Simultáneamente, el mayor recuento de unidades de prueba en DS03 (6 vs 3) somete al modelo a una evaluación más estricta frente a comportamientos no vistos, impactando la estabilidad de la métrica final de error.

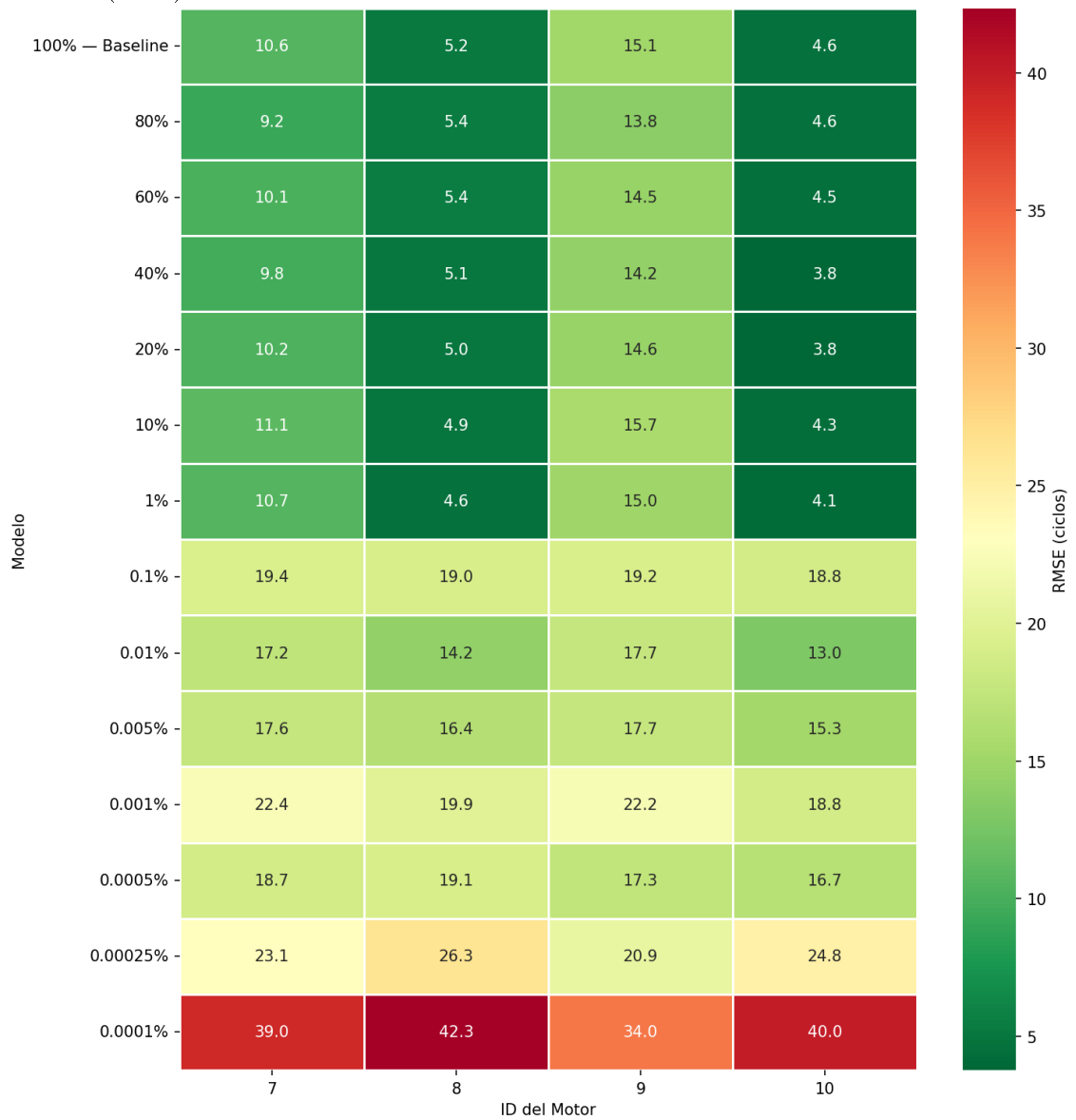
Bajo esta lógica, al contrastar DS01 frente a DS02, se observa que el modelo presenta un desempeño inferior en el primer caso a pesar de contar con recuentos de unidades totales similares (10 unidades en DS01 vs 9 en DS02). Esto se debe a que DS01, aun siendo más homogéneo en sus condiciones nominales, cuenta con un conjunto de prueba más extenso (4 motores vs 3 en DS02). Al evaluar en unidades como el N°7 y N°9 (ver Figura 6.4), el modelo registra un rendimiento inferior a la media, ya que la red LSTM no logra capturar con precisión los patrones de degradación acelerada de estos activos basándose únicamente en la homogeneidad de los 6 motores de entrenamiento. Esto confirma que la capacidad de generalización depende críticamente tanto de la representatividad de los escenarios de entrenamiento como de la variabilidad contenida en el conjunto de evaluación. Esta limitación técnica indica que el modelo LSTM no solo sufre por la escasez de datos,

6.1 Resultados obtenidos

sino por el “límite de generalización” en entornos heterogéneos, donde la interferencia de patrones entre distintos motores puede ser tan significativa como la señal de degradación misma.

Es importante mencionar que estos valores de RMSE toman el desempeño total del modelo pero existe una variación significativa en el desempeño de estas instancias dependiendo del motor sobre el cual se están realizando estas predicciones.

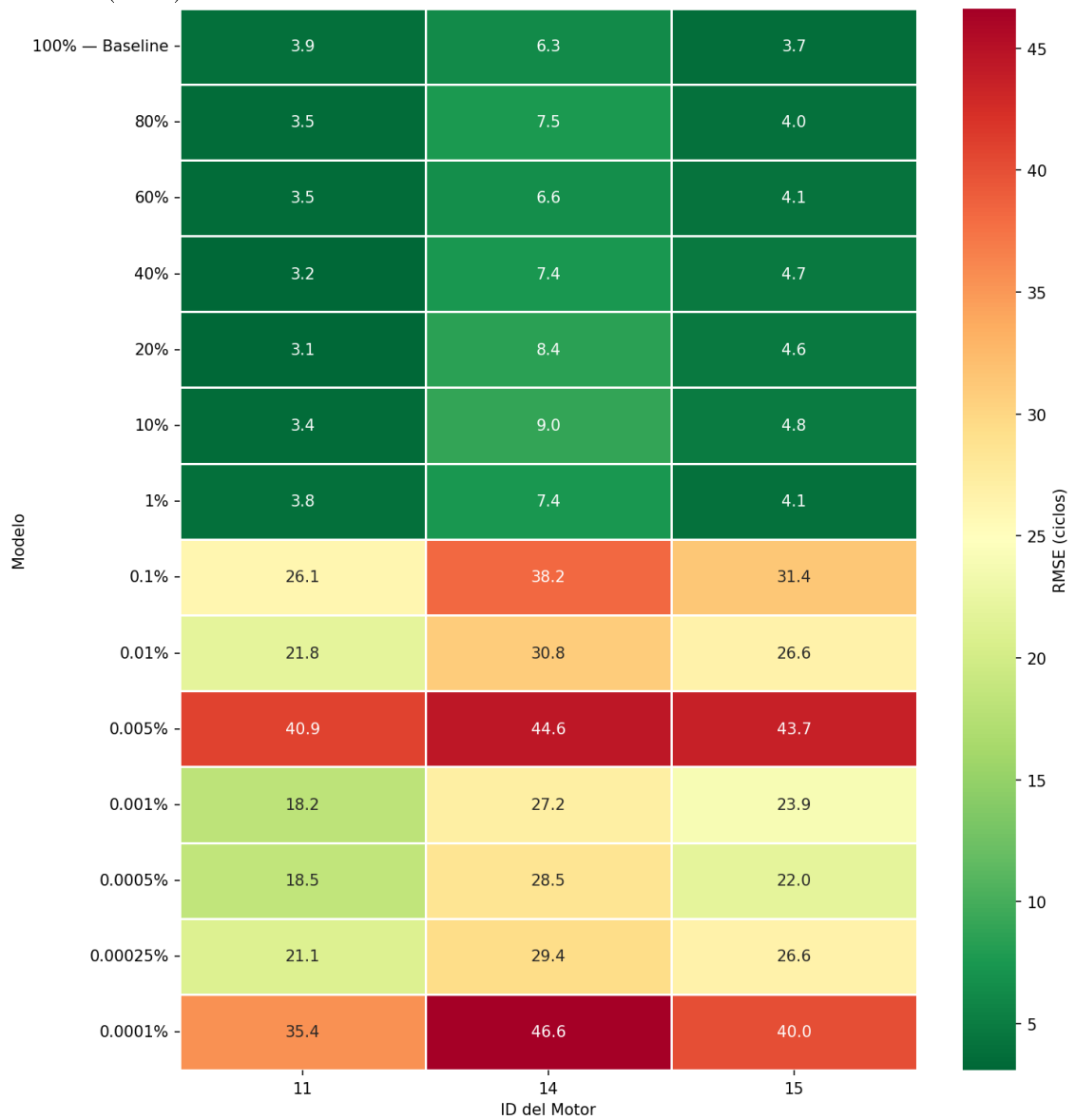
Figura 6.4: Distribución del RMSE por motor y nivel de reducción de datos en la arquitectura LSTM (DS01).



Nota. Gráfico que muestra el desempeño RMSE por motor logrado por modelos LSTM para distintos tamaños de la BBDD DS01.

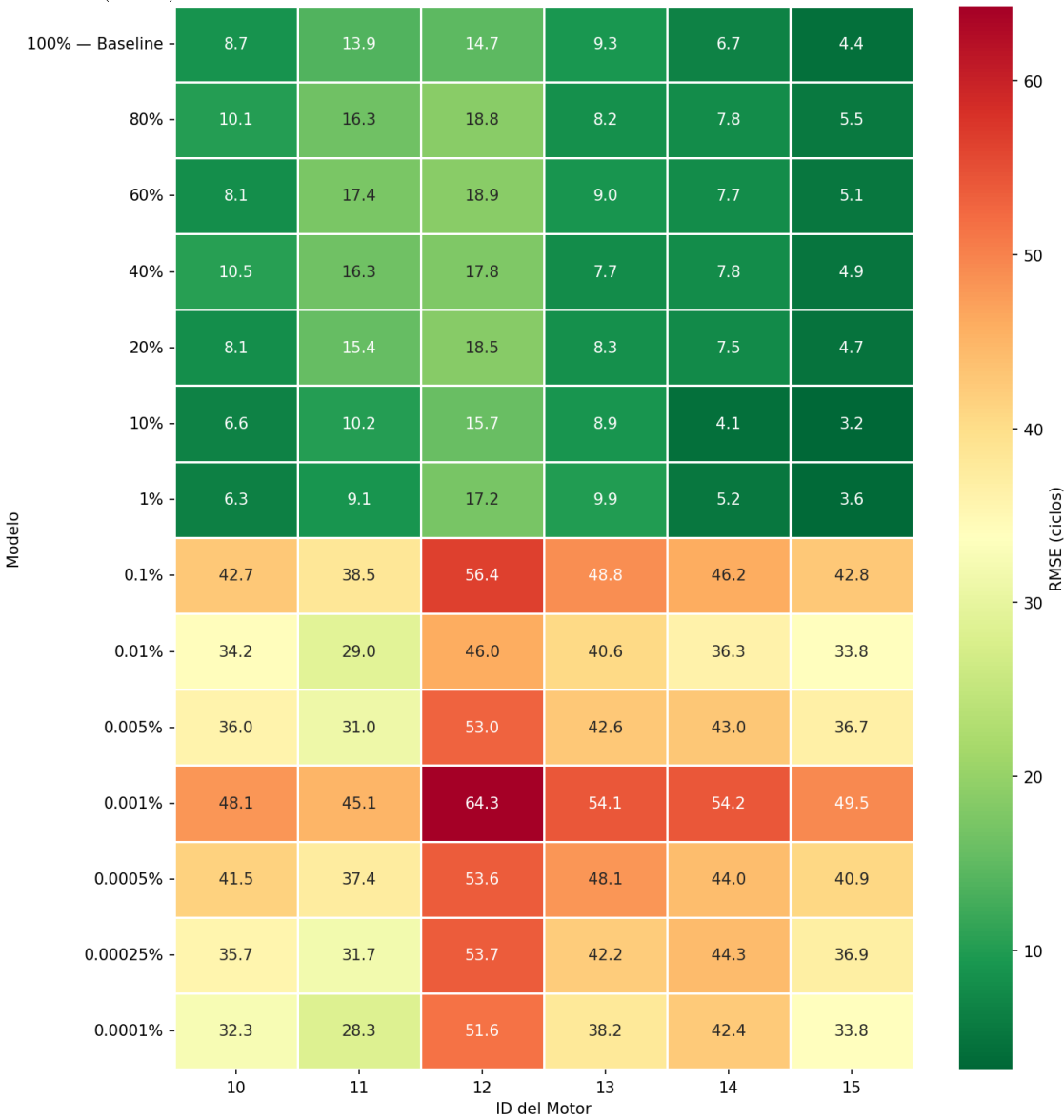
6.1 Resultados obtenidos

Figura 6.5: Distribución del RMSE por motor y nivel de reducción de datos en la arquitectura LSTM (DS02).



Nota. Gráfico que muestra el desempeño RMSE por motor logrado por modelos LSTM para distintos tamaños de la BBDD DS02.

Figura 6.6: Distribución del RMSE por motor y nivel de reducción de datos en la arquitectura LSTM (DS03).



Nota. Gráfico que muestra el desempeño RMSE por motor logrado por modelos LSTM para distintos tamaños de la BBDD DS03.

6.1 Resultados obtenidos

De las figuras se puede rescatar que el desempeño de las predicciones varía significativamente dependiendo del motor, pudiendo haber predicciones que duplican el valor RMSE del mejor motor en una misma instancia; esto se debe a que, al llegar al caso de 0,1 % de los datos, los modelos pierden gran parte de su capacidad adaptativa. Al no poder establecer relaciones robustas entre las variables para realizar predicciones exactas, se observan fluctuaciones donde el error puede incluso mejorar aleatoriamente, pero sin alcanzar un punto donde las predicciones resulten operativamente razonables.

6.1.2. Resultados XGBoost

Para contrastar la alta demanda de datos del modelo LSTM, los resultados de la arquitectura XGBoost revelan una dinámica de eficiencia del dato significativamente distinta (Figuras 6.7, 6.8 y 6.9). En los tres conjuntos analizados, existe una clara tendencia que indica que los modelos con mejor desempeño no son necesariamente aquellos con más información disponible. De manera generalizada, al reducir significativamente el tamaño de los conjuntos DS01, DS02 y DS03, las instancias logran una mejora en el rendimiento de un 27.2 %, 23.7 % y 53.5 % respectivamente, destacándose el caso de los *datasets* DS02 y DS03, donde el desempeño óptimo en RMSE se logra con tan solo el 0.001 % de los datos.

Luego de este umbral del 0.001 %, se observa un incremento acelerado en los valores de RMSE, demostrando que los modelos pierden rápidamente sus capacidades predictivas. Este punto crítico se sitúa en el 0.0001 % de la base de datos, donde el aumento del error es brusco; no obstante, esta pérdida de precisión es significativamente más gradual que la observada en la estructura LSTM. Esto indica que, en términos de resiliencia ante la escasez extrema, la arquitectura XGBoost es mucho más robusta para operar con información limitada, permitiendo una transición digital efectiva incluso en etapas incipientes de recolección de datos.

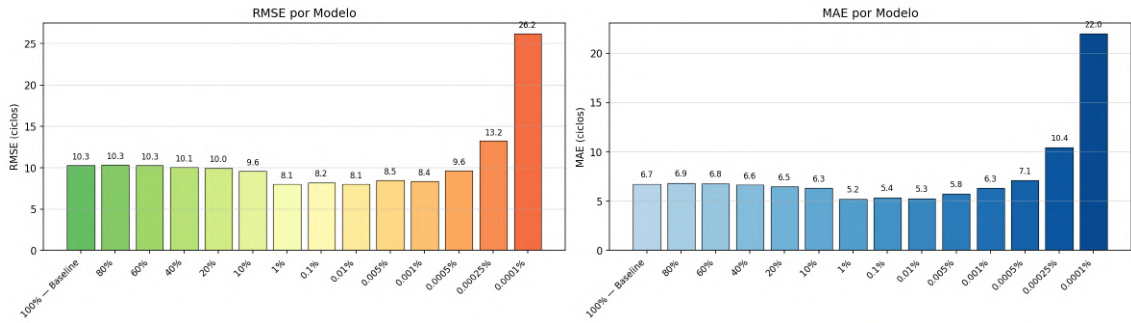
Tal como se describió en el Capítulo 5, el esquema incremental impide que los árboles generados en lotes anteriores sean modificados. Esta restricción tiene una consecuencia directa sobre la calidad del aprendizaje a medida que el volumen de datos crece: los residuales que cada nuevo árbol debe corregir no reflejan únicamente el error real entre la degradación observada y la predicha, sino también los errores acumulados de todas las correcciones anteriores. En consecuencia, el ensamble crece en tamaño pero no en calidad: los árboles generados en etapas avanzadas del entrenamiento corrigen ruido de corrección más que señal real de degradación. Al reducir el volumen de datos disponibles, se redu-

6.1 Resultados obtenidos

cen los lotes procesados y el ensamble resultante es más compacto, donde cada árbol fue construido sobre una señal de corrección menos fragmentada. Esto permite que los patrones de degradación genuinos dominen sobre el ruido acumulado, explicando la mejora en desempeño observada al reducir la cantidad de datos disponibles.

También cabe destacar que los modelos bajo esta arquitectura se ven impactados de la misma manera por las características de las BBDD, como es el caso de aquellos modelos vistos anteriormente construidos bajo la arquitectura LSTM. Esto se ve demostrado porque aquellos modelos entrenados con la BBDD DS02 tuvieron el mejor desempeño a pesar de contar con relaciones más complejas y que aquellas instancias entrenadas con la BBDD DS03, la cual contiene 15 motores, tuvieron el peor desempeño. Esto revela que ambas arquitecturas se ven fuertemente impactadas al tener que abarcar mayores casos, siendo un factor determinante en su desempeño, superando el efecto que podría tener un aumento en la complejidad de las relaciones entre las variables.

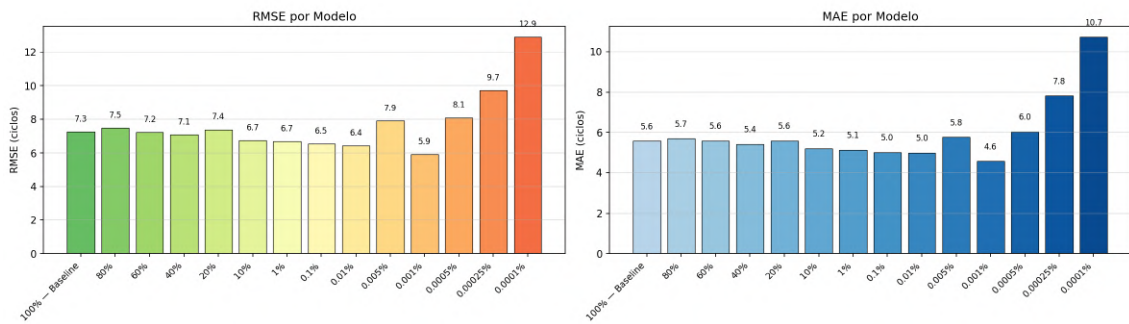
Figura 6.7: Comparación de RMSE y MAE de modelos XGBoost para BBDD DS01.



Nota. Gráfico que muestra el desempeño RMSE y MAE logrado por modelos XGBoost para distintos tamaños de la BBDD DS01.

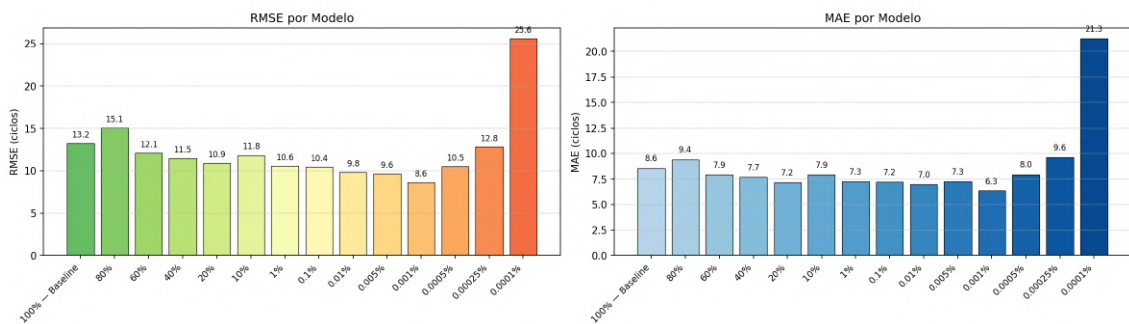
6.1 Resultados obtenidos

Figura 6.8: Comparación de RMSE y MAE de modelos XGBoost para BBDD DS02.



Nota. Gráfico que muestra el desempeño RMSE y MAE logrado por modelos XGBoost para distintos tamaños de la BBDD DS02.

Figura 6.9: Comparación de RMSE y MAE de modelos XGBoost para BBDD DS03.

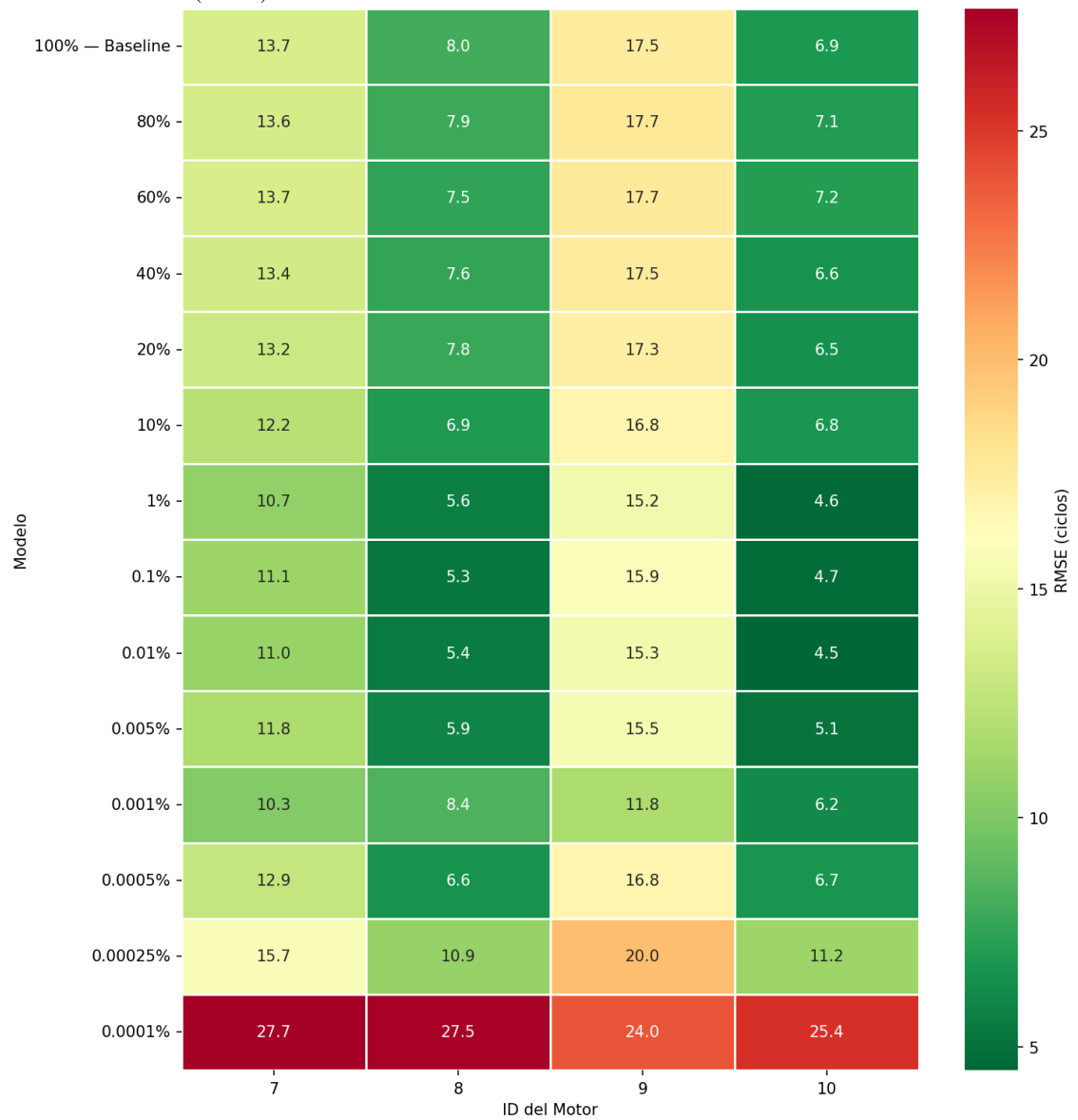


Nota. Gráfico que muestra el desempeño RMSE y MAE logrado por modelos XGBoost para distintos tamaños de la BBDD DS03.

De manera análoga a lo observado en la arquitectura LSTM, los modelos XGBoost presentan una dispersión importante en los resultados individuales de los motores que componen cada BBDD (ver Figuras [6.10](#), [6.11](#) y [6.12](#)). No obstante, el análisis de los mapas de calor permite identificar fenómenos propios de esta arquitectura en relación con la eficiencia del dato y la generalización de patrones complejos.

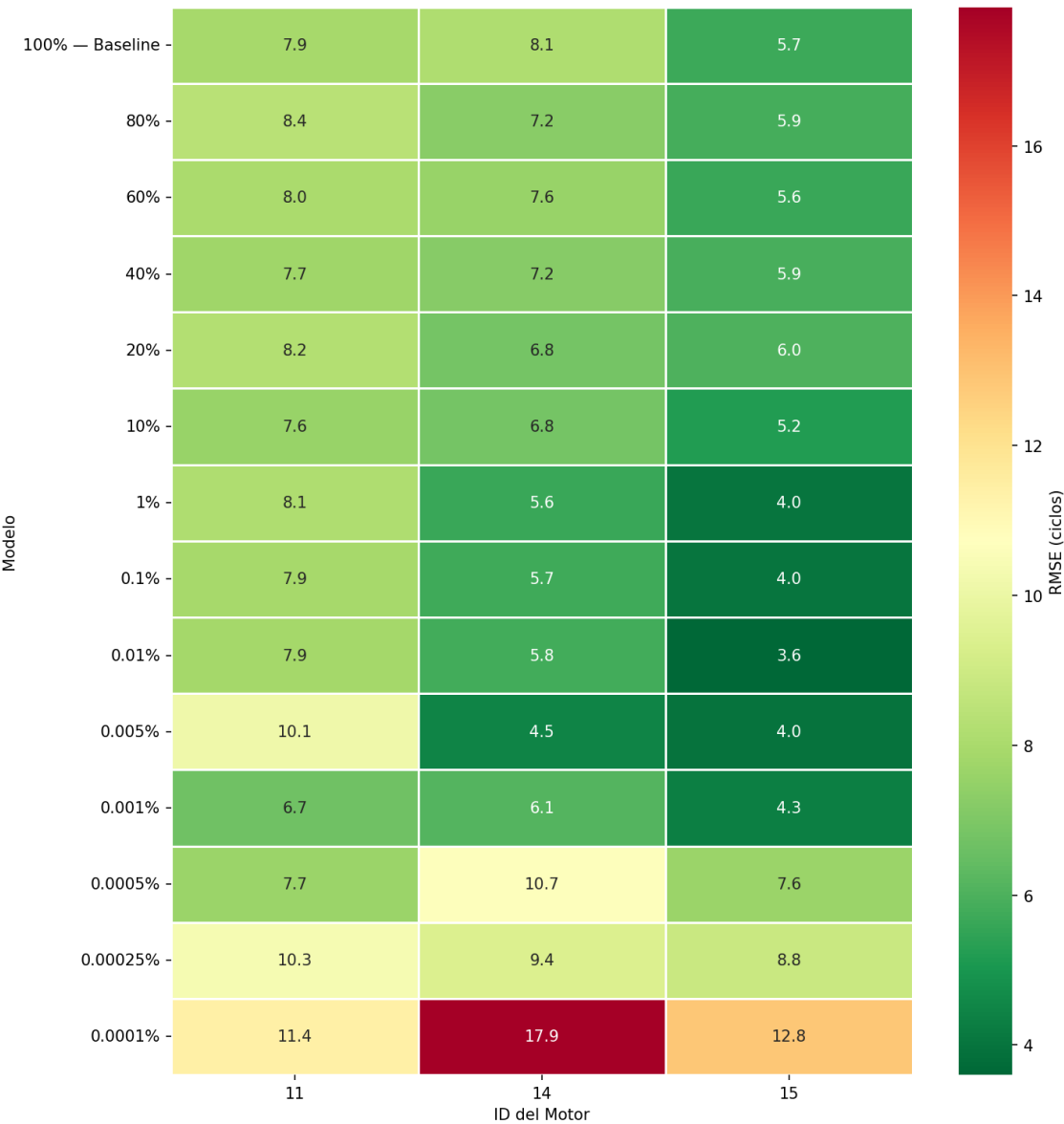
6.1 Resultados obtenidos

Figura 6.10: Distribución del RMSE por motor y nivel de reducción de datos en la arquitectura XGBoost (DS01).



Nota. Gráfico que muestra el desempeño RMSE por motor logrado por modelos XGBoost para distintos tamaños de la BBDD DS01.

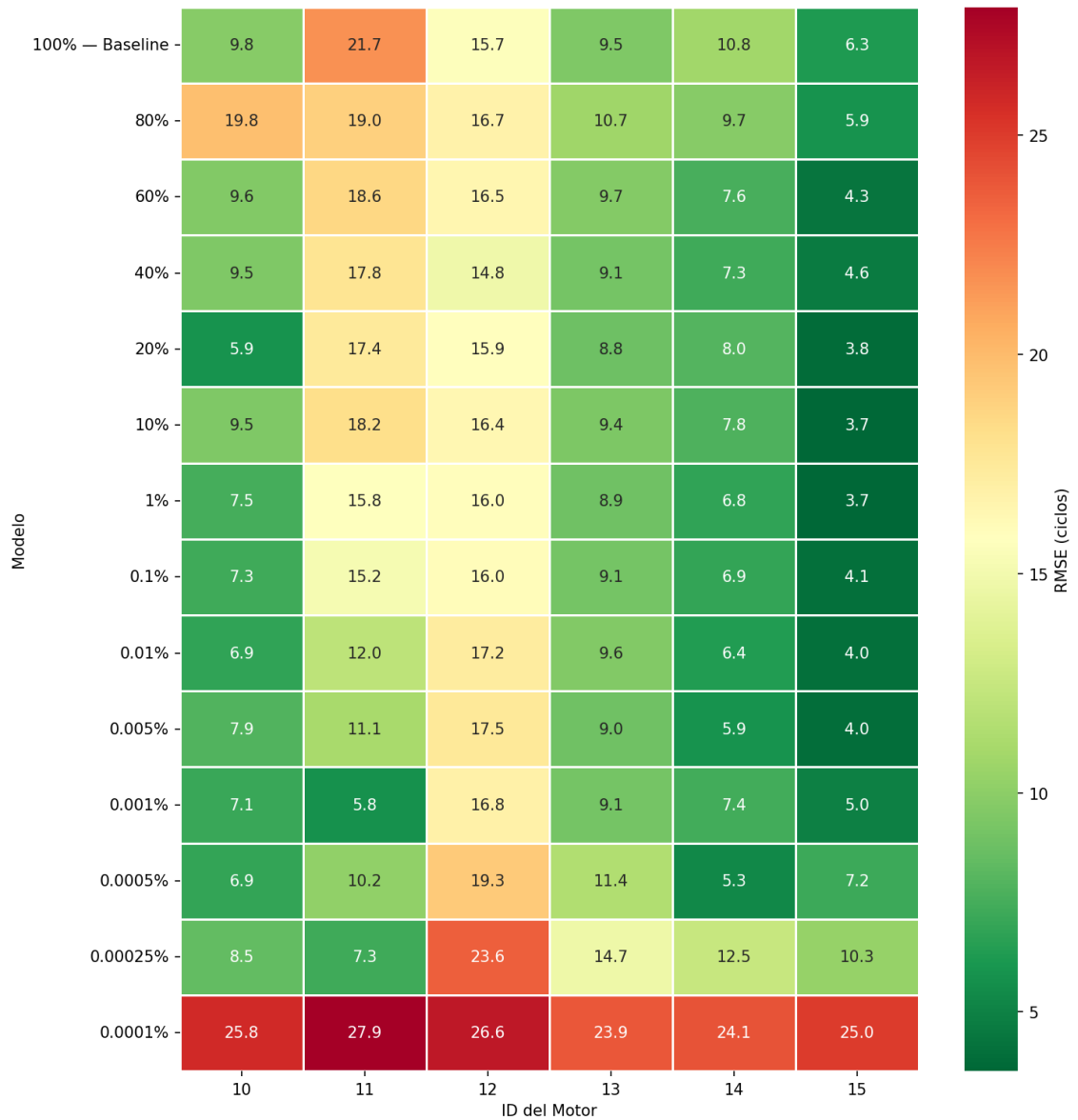
Figura 6.11: Distribución del RMSE por motor y nivel de reducción de datos en la arquitectura XGBoost (DS02).



Nota. Gráfico que muestra el desempeño RMSE por motor logrado por modelos XGBoost para distintos tamaños de la BBDD DS02.

6.1 Resultados obtenidos

Figura 6.12: Distribución del RMSE por motor y nivel de reducción de datos en la arquitectura XGBoost (DS03).



Nota. Gráfico que muestra el desempeño RMSE por motor logrado por modelos XGBoost para distintos tamaños de la BBDD DS03.

El comportamiento de XGBoost revela una sensibilidad selectiva según el perfil específico del motor evaluado. Mientras que ciertos activos, como el Motor 15 en la BBDD DS03, alcanzan errores mínimos de 3.7 ciclos, otros motores como el 12 en DS03 mantienen una resistencia persistente al aprendizaje, con errores que superan los 15 ciclos independientemente de la reducción del dataset.

Este comportamiento indica que el modelo, en su intento por generar una política de partición global, enfrenta un *trade-off* de generalización: la optimización de hiperparámetros tiende a favorecer los comportamientos promedio de la flota, dejando como “huérfanos predictivos” a aquellos motores con firmas de degradación atípicas. En términos técnicos, el modelo prioriza la estabilidad de la mayoría sacrificando la precisión en los extremos de la distribución, lo que refuerza la noción de que una arquitectura de bosque único podría estar alcanzando su límite operativo en conjuntos heterogéneos como DS03.

Finalmente, se observa un punto de ruptura crítico al utilizar solo el 0,0001% de la información, donde el RMSE aumenta de forma abrupta. A pesar de este colapso, es notable que la degradación del desempeño en XGBoost es considerablemente más resiliente que en los modelos LSTM, demostrando la superioridad de esta arquitectura ante escenarios de extrema limitación de datos y consolidándola como una herramienta técnica clave para habilitar la competitividad en organizaciones con recursos digitales restringidos.

7

Análisis Diferencial de las Capacidades Predictivas

7.1. Comparación de Exactitud Predictiva

Habiendo evaluado el comportamiento individual de cada arquitectura ante la reducción de datos en el capítulo anterior, esta sección ejecuta la comparación directa entre XGBoost y LSTM bajo condiciones idénticas de entrenamiento. El análisis se estructura en dos ejes: primero, la exactitud predictiva medida por RMSE en función del volumen de datos disponibles; segundo, la estabilidad y precisión del error analizada mediante diagramas de dispersión. El objetivo es determinar con evidencia empírica cuándo cada arquitectura representa la elección técnicamente superior.

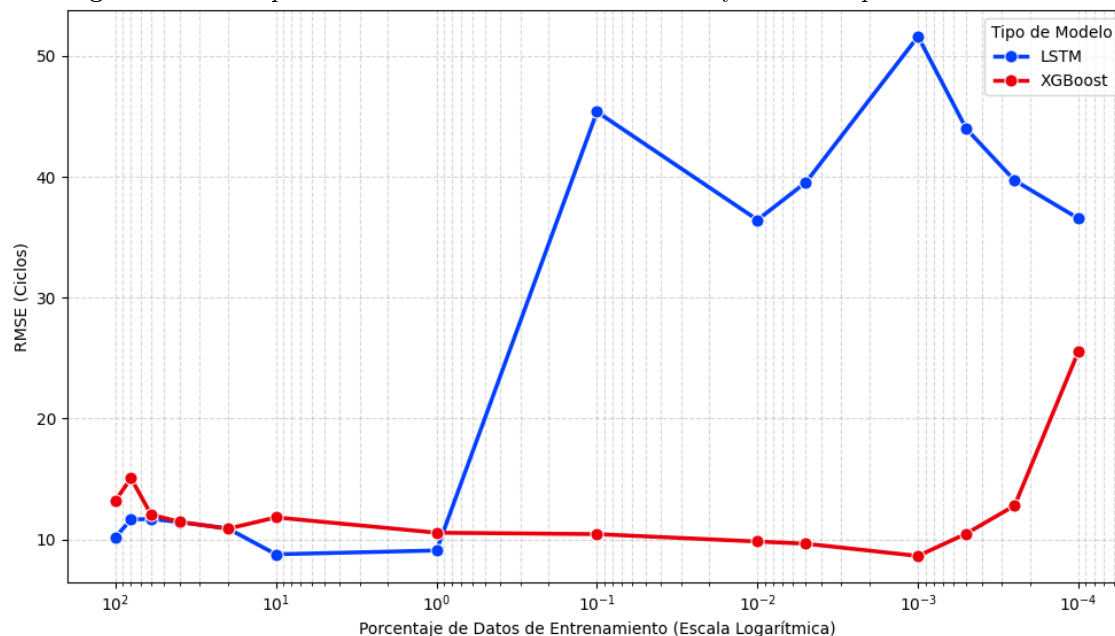
The graph illustrates the performance of two machine learning models, LSTM and XGBoost, measured by RMSE (Ciclos) against the percentage of training data (logarithmic scale). The X-axis represents the percentage of training data, ranging from 10^2 to 10^{-4} . The Y-axis represents the RMSE (Ciclos), ranging from 0 to 40. The LSTM model (blue line) shows a significant increase in RMSE as the training data percentage decreases below 10^0 , reaching approximately 40 at 10^{-4} . The XGBoost model (red line) shows a more gradual increase in RMSE as the training data percentage decreases below 10^0 , reaching approximately 26 at 10^{-4} .

Porcentaje de Datos de Entrenamiento (Escala Logarítmica)	LSTM (RMSE Ciclos)	XGBoost (RMSE Ciclos)
10^2	8.0	10.5
10^{-1}	7.5	10.5
10^0	7.5	8.0
10^{-1}	19.0	8.5
10^{-2}	15.0	8.0
10^{-3}	20.5	8.5
10^{-4}	40.0	26.0

Porcentaje de Datos de Entrenamiento (Escala Logarítmica)	LSTM (RMSE)	XGBoost (RMSE)
0.1	4.2	7.2
0.08	4.3	7.5
0.06	4.4	7.1
0.04	4.5	7.4
0.03	4.6	7.5
0.02	4.8	6.7
0.01	4.9	6.6
0.005	4.5	6.5
0.002	29.8	6.5
0.001	25.0	6.4
0.0005	42.5	7.8
0.0002	21.6	5.8
0.0001	21.3	8.0
0.00005	24.3	9.8
0.00002	38.6	12.8

7.1 Comparación de Exactitud Predictiva

Figura 7.3: Comparación de RMSE de modelos LSTM y XGBoost para BBDD DS03.



Nota. Gráfico que muestra el desempeño RMSE por motor logrado por modelos XGBoost para distintos tamaños de la BBDD DS03.

Existen dos grandes puntos a destacar que son aplicables para los tres casos presentes, el primero siendo la importante brecha que existe entre ambos modelos luego de pasar el 1 % de la BBDD. Pasado este punto la brecha logra su máxima amplitud de un 58 %, 81 % y 83 % para las BBDD DS01, DS02 y DS03, respectivamente. Destacando la resiliencia del modelo XGBoost, ya que aún cuando ambos modelos tenían a disposición tan solo 0.0001 % de los datos, está presente una brecha significativa entre ambos, posicionando al modelo XGBoost como superior en el aspecto de exactitud al momento de lidiar con datos extremadamente limitados además de la complejidad de las relaciones entre las variables, aunque sí se debe hacer una distinción clara de que esta brecha no implica que el modelo XGBoost realice predicciones de calidad, lo cual queda claro luego de ver los altos valores de RMSE logrados al lidiar con un porcentaje de datos inferior al 0.001 %.

El segundo punto a destacar es que la arquitectura LSTM logra su mayor diferenciación cuando tiene a disposición desde el 1 % de la cantidad de datos, llegando a tener

7.2 Análisis de Estabilidad del Desempeño

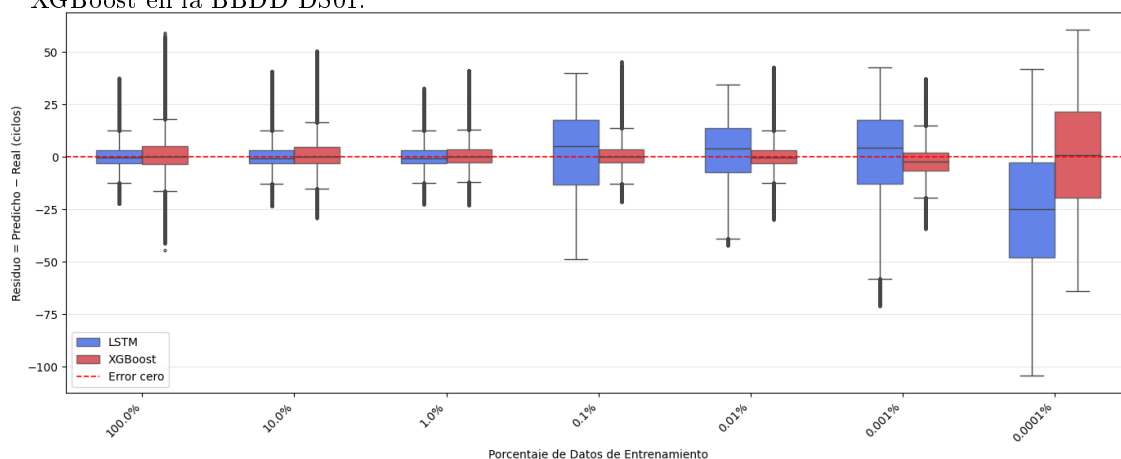
una exactitud en las tres BBDD de un 40 %, 73 % y 35 % mejor, respectivamente. Esto evidencia una de las mayores fortalezas de las redes neuronales profundas: su capacidad de extraer dependencias temporales complejas mediante sus capas recurrentes. Sin embargo, esta misma virtud se convierte en su mayor debilidad ante la falta de información masiva, generando que al reducir las BBDD a 0.1 % (equivalente a 4906, 5263 y 5570 ventanas temporales en cada dataset), los modelos pierdan toda capacidad de predicción exacta. Esta sensibilidad arquitectónica se explica por el “colapso de secuencialidad”, al fragmentarse el historial operativo, las LSTM no logran reconstruir una cadena causal coherente entre los pasos de tiempo, resultando en pesos inestables. En contraste, XGBoost se beneficia de su arquitectura basada en ensamble de árboles y aplanamiento de datos, la cual, al tratar cada ventana como una observación estadística independiente y regularizada, mantiene una firma predictiva básica mucho más resiliente que el enfoque secuencial de la LSTM cuando los datos son insuficientes para sostener una red profunda compleja.

7.2. Análisis de Estabilidad del Desempeño

Anteriormente se evaluó la exactitud con el cual ambas arquitecturas hacen sus predicciones, pero un factor importante a considerar es la precisión de las predicciones, es decir qué tan cerca o lejos se encuentra la agrupación de valores de lograr un residual (RUL verdadero - RUL predicho) igual a cero. Esto es de vital importancia porque indica la estabilidad y consistencia del modelo, información útil para establecer qué tan confiable es el modelo en alcanzar el desempeño reflejado en los gráficos anteriores. Para analizar lo anterior se desarrollaron gráficos Box-Plot que comparan los modelos de ambas arquitecturas para cada BBDD.

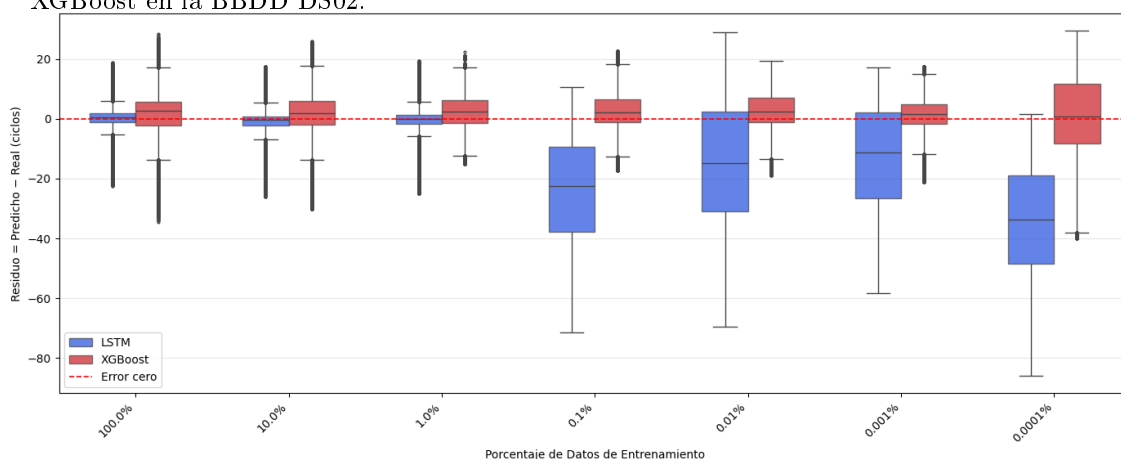
7.2 Análisis de Estabilidad del Desempeño

Figura 7.4: Distribución y variabilidad del error RMSE para las arquitecturas LSTM y XGBoost en la BBDD DS01.



Nota. Diagrama de caja que ilustra la dispersión del error; el ancho de la caja (IQR) representa la precisión de la arquitectura, mientras que la línea central indica la mediana del error o exactitud del modelo.

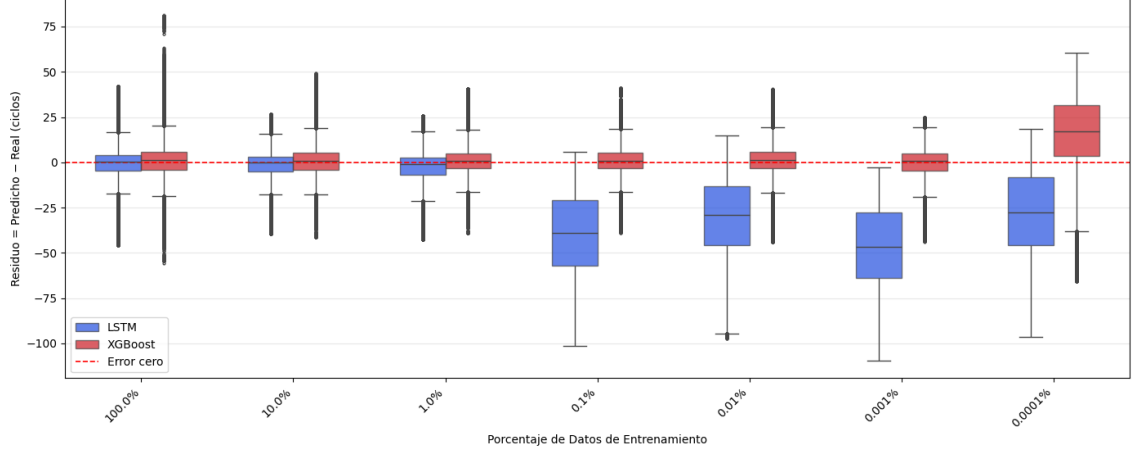
Figura 7.5: Distribución y variabilidad del error RMSE para las arquitecturas LSTM y XGBoost en la BBDD DS02.



Nota. Diagrama de caja que ilustra la dispersión del error; el ancho de la caja (IQR) representa la precisión de la arquitectura, mientras que la línea central indica la mediana del error o exactitud del modelo.

7.2 Análisis de Estabilidad del Desempeño

Figura 7.6: Distribución y variabilidad del error RMSE para las arquitecturas LSTM y XGBoost en la BBDD DS03.



Nota. Diagrama de caja que ilustra la dispersión del error; el ancho de la caja (IQR) representa la precisión de la arquitectura, mientras que la línea central indica la mediana del error o exactitud del modelo.

De manera uniforme, los modelos LSTM entrenados en las tres bases de datos logran su mejor precisión cuando disponen de al menos el 10 % de los datos. En este escenario, el rango intercuartílico ($IQR = Q_3 - Q_1$) se sitúa próximo a un residual cero, lo que indica que el 50 % de las predicciones se agrupa con errores mínimos, evidenciando una alta estabilización en los tres casos.

Este comportamiento se invierte drásticamente al reducir la disponibilidad al 0.1 %. En DS02 y DS03 (ver Figuras 7.5 y 7.6), el rango intercuartílico abandona por completo la zona del residual cero, desplazándose hacia territorio negativo. DS01 presenta un comportamiento distinto: aunque sus predicciones también se alejan del óptimo, los modelos LSTM entrenados con esta base de datos conservan el residual cero dentro del rango intercuartílico en la mayoría de los casos. Esto indica que la simplicidad y homogeneidad de DS01, un factor que es neutral cuando los datos son abundantes, se convierte en una ventaja relativa ante la escasez, volviendo a los modelos LSTM de DS01 más resilientes que sus equivalentes entrenados con DS02 y DS03.

7.2 Análisis de Estabilidad del Desempeño

Tal como se mencionó anteriormente, la ventaja en precisión y exactitud que logra el modelo LSTM termina abruptamente una vez llegado al 0.1 % de las BBDD, es desde este punto donde la arquitectura XGBoost demuestra su impresionante capacidad de resiliencia ante la falta de datos. Para las tres BBDD, aún en el caso de contar con tan solo 49, 52 y 55 ventanas temporales (equivalente al 0.001 % de los datos), los modelos pudieron retornar una mediana considerablemente cercana a un residual cero y un rango intercuartílico reducido.

Un aspecto fundamental en la evaluación de ambas arquitecturas es el comportamiento de sus valores atípicos (*outliers*), los cuales revelan sensibilidades operativas dependientes de la complejidad de la base de datos. En la BBDD DS01, se observa una clara asimetría con una concentración mayoritaria de residuos positivos. Esta aglomeración evidencia un sesgo optimista sistémico derivado de la simplicidad del dataset; el modelo converge hacia una tasa de falla nominal y tiende a sobreestimar significativamente la vida útil de aquellas unidades que presentan un desgaste más acelerado.

Sin embargo, este comportamiento difiere en las bases de datos más complejas (DS02 y DS03), donde la distribución de los valores atípicos tiende hacia una mayor simetría alrededor del error cero. La mayor riqueza de variables y condiciones operativas en estas BBDD fuerza a los modelos a realizar un ajuste predictivo más equilibrado, neutralizando el sesgo optimista predominante en DS01.

A pesar de esta simetría inicial, la magnitud y dirección de estos errores atípicos divergen drásticamente entre arquitecturas al someterlas a la escasez de datos. La arquitectura XGBoost logra mantener sus valores atípicos dentro de límites significativamente más controlados que LSTM en las tres bases de datos: los bigotes del diagrama de caja permanecen dentro de rangos operativamente razonables (≤ 25 ciclos en la mayoría de los escenarios), con puntos de error extremo que no superan los 75 ciclos, demostrando una notable resisten-

7.3 Modelos LSTM y XGBoost de mejor desempeño en las tres BBDD

cia frente al ruido estocástico. En contraste, la red LSTM exhibe una alta vulnerabilidad ante la reducción de la muestra. Particularmente en DS02 y DS03 (umbrales $\leq 0.1\%$), LSTM no solo expande la dispersión de sus *outliers*, sino que genera puntos de error en territorio negativo que superan los 75 ciclos. Cabe destacar que este colapso no es monótono, a niveles de reducción extrema, el comportamiento del modelo puede experimentar variaciones erráticas derivadas del ruido estocástico del proceso de entrenamiento, sin recuperar en ningún caso una capacidad predictiva operativamente viable. Esta dislocación hacia la subestimación severa indica que, ante la falta de datos complejos, la red neuronal colapsa hacia predicciones excesivamente conservadoras. Es relevante destacar que, en los modelos de mejor desempeño de ambas arquitecturas, la severidad del error se reduce a medida que las observaciones se aproximan al final de la vida útil del activo. Esta convergencia, evidenciada en las curvas predictivas de los modelos óptimos, indica que las arquitecturas logran capturar la dinámica de degradación acelerada en el horizonte temporal más crítico para la toma de decisiones de mantenimiento. Sin embargo, esta observación se aplica a los modelos entrenados con volúmenes de datos suficientes; bajo condiciones de colapso (LSTM a partir del 0.1%), la aparente reducción del error en las etapas finales responde en parte a la restricción natural del dominio, donde el RUL no puede ser negativo.

7.3. Modelos LSTM y XGBoost de mejor desempeño en las tres BBDD

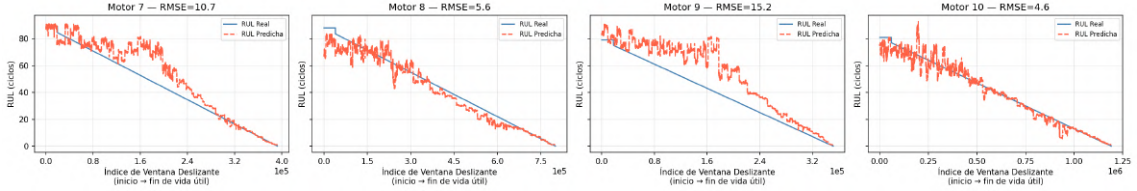
Habiendo analizado en profundidad la dispersión del error y la resiliencia paramétrica de ambas arquitecturas, a continuación se presentan los gráficos predictivos correspondientes a los mejores modelos obtenidos para cada arquitectura (LSTM y XGBoost) en las tres bases de datos evaluadas (DS01, DS02 y DS03). Es imperativo recordar que, tal como se ha demostrado empíricamente a lo largo de este capítulo, el desempeño óptimo de un modelo no está estrictamente condicionado por la máxima disponibilidad de información.

7.3 Modelos LSTM y XGBoost de mejor desempeño en las tres BBDD

Por consiguiente, las instancias seleccionadas para esta comparación final representan el volumen exacto de datos, ya sea la totalidad de la muestra o una fracción estratégica de la misma, que logró minimizar el error absoluto (RMSE) y controlar la varianza predictiva de manera más efectiva para cada caso específico.

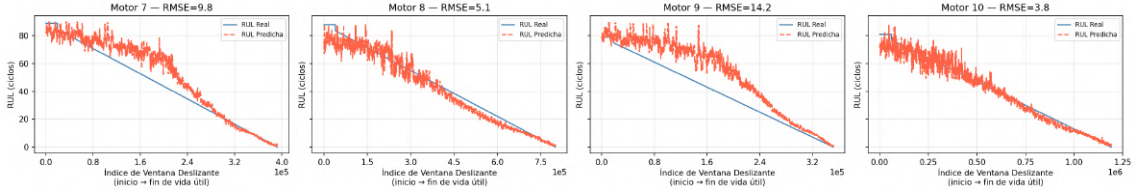
Base de Datos DS01

Figura 7.7: Evolución del RMSE por ventana temporal para el modelo óptimo de XGBoost entrenado con el 1 % de la BBDD DS01.



Nota. Gráfico que ilustra la trayectoria del error predictivo (RMSE) a lo largo de las ventanas temporales. Esta curva representa el modelo XGBoost con mejor desempeño en la BBDD DS01, logrado al utilizar una reducción al 1 % de los datos de entrenamiento para maximizar la generalización y minimizar la varianza.

Figura 7.8: Evolución del RMSE por ventana temporal para el modelo óptimo de LSTM entrenado con el 40 % de la BBDD DS01.

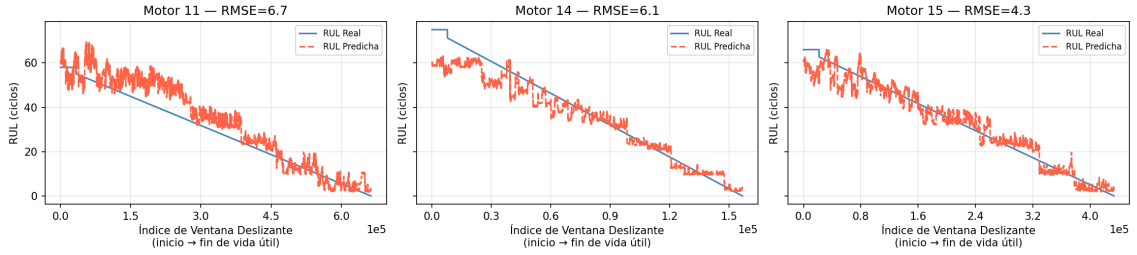


Nota. Gráfico que ilustra la trayectoria del error predictivo (RMSE) a lo largo de las ventanas temporales. Esta curva representa el modelo LSTM con mejor desempeño en la BBDD DS01, logrado al utilizar una reducción al 40 % de los datos de entrenamiento para maximizar la generalización y minimizar la varianza.

7.3 Modelos LSTM y XGBoost de mejor desempeño en las tres BBDD

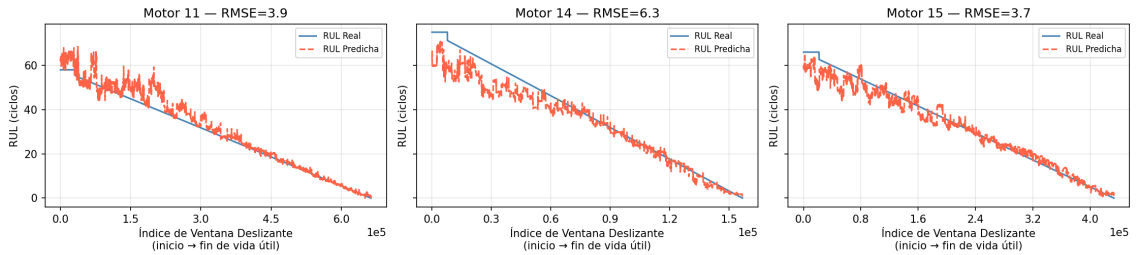
Base de Datos DS02

Figura 7.9: Evolución del RMSE por ventana temporal para el modelo óptimo de XGBoost entrenado con el 0.001 % de la BBDD DS02.



Nota. Gráfico que ilustra la trayectoria del error predictivo (RMSE) a lo largo de las ventanas temporales. Esta curva representa el modelo XGBoost con mejor desempeño en la BBDD DS02, logrado al utilizar una reducción al 0.001 % de los datos de entrenamiento para maximizar la generalización y minimizar la varianza.

Figura 7.10: Evolución del RMSE por ventana temporal para el modelo óptimo de LSTM entrenado con el 100 % de la BBDD DS02.

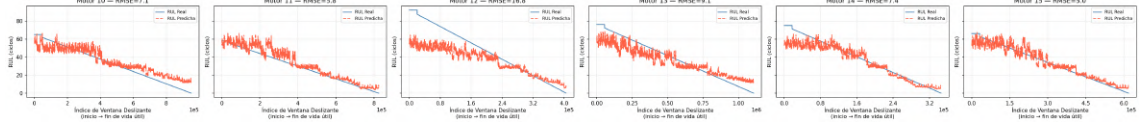


Nota. Gráfico que ilustra la trayectoria del error predictivo (RMSE) a lo largo de las ventanas temporales. Esta curva representa el modelo LSTM con mejor desempeño en la BBDD DS02, logrado al utilizar el 100 % de los datos de entrenamiento para maximizar la generalización y minimizar la varianza.

7.3 Modelos LSTM y XGBoost de mejor desempeño en las tres BBDD

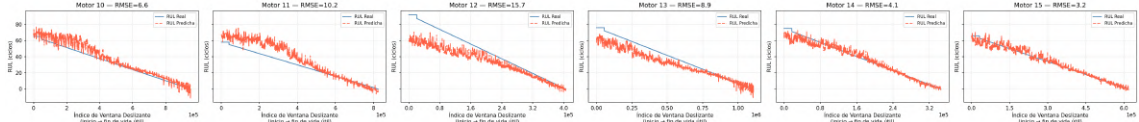
Base de Datos DS03

Figura 7.11: Evolución del RMSE por ventana temporal para el modelo óptimo de XGBoost entrenado con el 0.001 % de la BBDD DS03.



Nota. Gráfico que ilustra la trayectoria del error predictivo (RMSE) a lo largo de las ventanas temporales. Esta curva representa el modelo XGBoost con mejor desempeño en la BBDD DS03, logrado al utilizar una reducción al 0.001 % de los datos de entrenamiento para maximizar la generalización y minimizar la varianza.

Figura 7.12: Evolución del RMSE por ventana temporal para el modelo óptimo de LSTM entrenado con el 10 % de la BBDD DS03.



Nota. Gráfico que ilustra la trayectoria del error predictivo (RMSE) a lo largo de las ventanas temporales. Esta curva representa el modelo LSTM con mejor desempeño en la BBDD DS03, logrado al utilizar una reducción al 10 % de los datos de entrenamiento para maximizar la generalización y minimizar la varianza.

Aplicaciones Industriales y Hoja de Ruta de Implementación

La evidencia experimental obtenida en esta investigación no solo responde a una pregunta académica, sino que define una hoja de ruta técnica concreta para organizaciones que buscan implementar sistemas de Mantenimiento Predictivo con recursos de datos limitados. A continuación se detallan las implicancias operativas directas de los hallazgos, organizadas según el estadio de madurez digital de la organización.

Los resultados obtenidos demuestran que, para organizaciones emergentes o industrias que recién comienzan a digitalizar sus procesos, la disponibilidad de datos de falla no es necesariamente una barrera absoluta para adoptar el PdM. La evidencia experimental cuantificó que XGBoost logra predicciones operativamente viables ($RMSE \leq 10$ ciclos) con una fracción mínima de los datos que requiere LSTM, lo que redefine el umbral de entrada a la digitalización predictiva. En este contexto, los siguientes beneficios tienen implicancias directas sobre la planificación tecnológica de las organizaciones:

Los resultados obtenidos destacan los siguientes beneficios del modelo XGBoost para la industria:

- **Desempeño bajo Escasez (Arranque en Frío):** Se demostró que el modelo XGBoost exhibió una robustez superior en condiciones extremas de escasez de datos, a

pesar de que los perfiles de la NASA (N-CMAPSS) simulan relaciones operativas y de degradación altamente complejas.

- **Eficiencia en el Volumen de Datos y Umbral de Activación Operativa:**

XGBoost demostró requerir un volumen de datos varios órdenes de magnitud inferior al del modelo LSTM para alcanzar un umbral de predicción operativamente viable. Esta asimetría define la ventana de activación operativa: mientras LSTM demanda en promedio 52.466 ventanas deslizantes para alcanzar predicciones viables, XGBoost lo logra con tan solo 52 ventanas en promedio — una reducción de tres órdenes de magnitud que elimina la barrera de datos para organizaciones en etapas tempranas de instrumentación.

- **Gestión del Riesgo Financiero y Operativo:**

El análisis de la dispersión del error reveló que cuando LSTM se enfrenta a una escasez de datos, tiende hacia un colapso pesimista (subestimación severa del RUL). En la industria, esto se traduciría en el reemplazo prematuro de componentes útiles, inflando los costos de mantenimiento (OPEX). XGBoost, al mantener sus valores atípicos bajo control, evita estas fluctuaciones extremas, permitiendo una planificación logística más estable.

- **Confiabilidad en el Horizonte Crítico:**

Un hallazgo vital para la seguridad operativa es que los modelos de mejor desempeño de ambas arquitecturas logran reducir drásticamente sus errores al acercarse el final de la vida útil del activo, siempre que cuenten con el volumen mínimo de datos requerido. Esto garantiza que las alertas generadas en el período crítico (previo a la falla) son altamente confiables para ejecutar mantenimientos correctivos programados (*Just-in-Time*), mitigando el riesgo de paradas de planta no planificadas cuando se opera por encima del umbral de datos identificado.

- **Baja Barrera de Infraestructura Tecnológica:** Mientras que la arquitectura LSTM requiere de hardware especializado (GPU) para procesar tensores masivos de forma eficiente, se demostró que XGBoost ofrece una alta resiliencia operando sobre hardware estándar (CPU y memoria RAM convencional). Para una organización con recursos limitados, esto elimina una inversión inicial crítica en infraestructura de computación de alto rendimiento, permitiendo que la democratización del PdM sea tanto digital como económica.

8.1. Propuesta de Adopción Escalonada

Con base en lo anterior, no se trata de elegir definitivamente un modelo sobre el otro, sino de aplicarlos según la madurez digital de la empresa. XGBoost se posiciona como la solución óptima de entrada para organizaciones donde la recolección de datos de falla es lenta o incipiente, permitiendo habilitar capacidades predictivas operativas desde las etapas tempranas de instrumentación, aprovechando los umbrales de activación cuantificados en esta investigación. A medida que la organización acumula un volumen de datos robusto y la variabilidad operativa de la flota queda documentada, la transición hacia arquitecturas como LSTM se vuelve justificable para exprimir el máximo nivel de precisión a largo plazo.

8.1.1. Casos de Aplicación por Sector Industrial

- **Manufactura (PyMEs):** Una empresa con una sola línea de producción y menos de 6 meses de datos históricos puede implementar XGBoost como primera aproximación al PdM, obteniendo predicciones de RUL operativamente viables desde las etapas tempranas de instrumentación.
- **Minería:** Las flotas de equipos pesados que operan en ciclos de degradación prolongados y con múltiples modos de falla (análogos a DS02) se benefician de la robustez

8.1 Propuesta de Adopción Escalonada

de XGBoost durante la fase inicial de recolección de datos, para luego migrar a LSTM una vez acumulado un historial representativo de la flota.

- **Energía y Utilities:** En plantas con instrumentación avanzada y múltiples años de datos de sensores, la arquitectura LSTM representa la opción óptima para maximizar la exactitud predictiva y minimizar los costos de mantenimiento a largo plazo.

Conclusión

La presente investigación tuvo como propósito fundamental evaluar la resiliencia y viabilidad de distintas arquitecturas de Machine Learning aplicadas al Mantenimiento Predictivo (PdM), con un enfoque crítico en escenarios de disponibilidad limitada de datos. A través de la simulación de condiciones operativas reales y la degradación sistemática de las bases de datos N-CMAPSS, se logró ir más allá de la tradicional búsqueda del menor error predictivo, revelando el comportamiento intrínseco de los modelos ante la escasez de información, una barrera característica de las Pequeñas y Medianas Empresas (PyMEs).

9.1. Resiliencia vs. Complejidad Arquitectónica

El contraste entre las arquitecturas evaluadas confirmó que no existe un modelo universalmente superior, sino una dependencia crítica del contexto de los datos. Se demostró que la red profunda LSTM posee una capacidad sobresaliente para capturar las dinámicas de degradación complejas cuando dispone de volúmenes masivos de información (superando a XGBoost por márgenes de hasta un 73 % en exactitud (DS02), con ventajas que oscilan entre el 30 % y el 73 % según la complejidad del conjunto de datos). Sin embargo, esta superioridad está estrictamente condicionada a la abundancia de datos; al reducir la información disponible al 0,1 %, la arquitectura LSTM sufre un colapso predictivo derivado de la pérdida de conectividad causal en sus capas recurrentes.

9.2 El Umbral Crítico de Viabilidad Operativa

Por el contrario, XGBoost demostró una extraordinaria robustez ante el estrés por escasez. La arquitectura basada en árboles, por su parte, logró mantener capacidades predictivas estables incluso al ser entrenada con fracciones mínimas de información (0,001 %), evidenciando que los modelos más simples pueden ser significativamente más eficientes en la extracción de patrones cuando el historial operativo es naciente o fragmentado.

9.2. El Umbral Crítico de Viabilidad Operativa

El hallazgo empírico más significativo de esta investigación radica en la cuantificación del volumen mínimo de datos requerido para alcanzar una estabilidad predictiva razonable (RMSE inferior o igual a 10 ciclos). Mientras que la arquitectura LSTM requirió un promedio histórico de 52.466 ventanas deslizantes para estabilizar sus predicciones, XGBoost alcanzó un desempeño operativo competitivo requiriendo un promedio de tan solo 52 ventanas temporales.

Esta brecha abismal en el requerimiento de información redefine la viabilidad de la Industria 4.0 para organizaciones emergentes, desmintiendo la noción de que el mantenimiento predictivo es exclusivo de corporaciones con bases de datos consolidadas durante años.

9.3. Estabilidad, Sesgo y Gestión del Riesgo

El análisis de la varianza del error y los valores atípicos (outliers) reveló implicancias logísticas vitales. Se descubrió que, ante la falta de datos, la red LSTM tiende a generar predicciones severamente pesimistas, subestimando drásticamente la Vida Útil Remanente (RUL). En un entorno industrial real, este sesgo se traduciría en mantenimientos correctivos prematuros y un aumento injustificado de los costos operativos (OPEX). XGBoost, al exhibir una distribución de errores mucho más simétrica y contenida bajo escasez, minimiza el riesgo de paradas innecesarias.

9.4 Reflexión Final y Adopción en la Industria 4.0

Para efectos de seguridad operativa, los modelos de mejor desempeño de ambas arquitecturas demostraron una convergencia confiable en las etapas finales del ciclo de vida, validando la capacidad de generar alertas precisas en el horizonte crítico previo a la falla. Sin embargo, esta confiabilidad está condicionada a que el modelo cuente con un volumen mínimo de datos operativo: el umbral del 1 % para LSTM y fracciones significativamente menores para XGBoost.

9.4. Reflexión Final y Adopción en la Industria 4.0

Los resultados obtenidos permiten concluir que la implementación del Mantenimiento Predictivo no debe abordarse desde la selección estática de un algoritmo de estado del arte, sino a través de una estrategia de adopción escalonada.

XGBoost se consolida como la herramienta fundacional ideal para organizaciones y PyMEs que inician su transición digital, permitiendo habilitar capacidades predictivas operativas desde etapas tempranas de instrumentación, operando dentro del umbral mínimo identificado experimentalmente. Posteriormente, a medida que la organización madura digitalmente y acumula un volumen de datos robusto, la transición hacia arquitecturas profundas como LSTM se vuelve el camino lógico para maximizar la exactitud a largo plazo. De este modo, esta investigación aporta un marco técnico y estratégico que democratiza el acceso a la confiabilidad predictiva, adaptándose a la realidad operativa de recursos de datos y hardware limitados en la industria actual.

Referencias

- Achouch, M., Dimitrova, M., Ziane, K., Sattarpanah Karganroudi, S., Dhouib, R., Ibrahim, H., y Adda, M. (2022). On Predictive Maintenance in Industry 4.0: Overview, Models, and Challenges. *Applied Sciences*, 12(16), 3. <https://doi.org/10.3390/app12168081>
- Allen, G. (2020). *Understanding AI Technology* (inf. téc.). Joint Artificial Intelligence Center (JAIC). Consultado el 6 de agosto de 2025, desde https://scholar.google.es/citations?view_op=view_citation&hl=es&user=n8CBJFwAAAAJ&citation_for_view=n8CBJFwAAAAJ:W7OEmFMyl1HYC
- Brownlee, J. (2016). *XGBoost With Python: Gradient Boosted Trees with XGBoost and scikit-learn*. Machine Learning Mastery. Consultado el 16 de julio de 2025, desde <https://books.google.es/books?hl=es&lr=&id=HgmqDwAAQBAJ&oi=fnd&pg=PP1&dq=XGboost+basics&ots=nNeIcbL5NF&sig=nfRQEkL9meQEozZu1FneTlKai2U#v=onepage&q=XGboost%20basics&f=false>
- Chao, M., Kulkarni, C., Goebel, K., y Fink, O. (2021). Aircraft Engine Run-to-Failure Dataset under Real Flight Conditions for Prognostics and Diagnostics. *Data*, 6(1). <https://doi.org/10.3390/data6010005>
- Chen, T., y Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 794. <https://doi.org/10.1145/2939672.2939785>
- Coleman, C., Damodaran, S., Chandramouli, M., y Deuel, E. (2017). *Making Maintenance Smarter: Predictive Maintenance and the Digital Supply Network* (inf. téc.). Deloitte Insights. Consultado el 12 de marzo de 2026, desde <https://www.deloitte.com/us/en/insights/industry/manufacturing-industrial-products/industry-4-0/using-predictive-technologies-for-asset-maintenance.html>
- Das, U., Tey, K., Seyedmahmoudian, M., Mekhilef, S., Idris, M., Van Deventer, W., Horan, B., y Stojcevski, A. (2017). Forecasting of Photovoltaic Power Generation and Model Optimization: A Review [Disponible en: <http://dx.doi.org/10.1016/j.rser.2017.08.017>. Accedido: 2025-08-31]. *Renewable and Sustainable Energy Reviews*, 81, 6. <https://doi.org/10.1016/j.rser.2017.08.017>
- Géron, A. (2018). *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media. Consultado el 28 de septiembre de 2025, desde <https://www.oreilly.com/library/view/neural-networks-and/9781492037354/ch04.html>
- MaintainX. (2024). *The State of Industrial Maintenance 2024* (inf. téc.). MaintainX. Consultado el 16 de julio de 2025, desde <https://www.getmaintainx.com>

REFERENCIAS

- Mhlanga, D. (2023). Artificial Intelligence and Machine Learning for Energy Consumption and Production in Emerging Markets. *Energies*, 16(2), 3. <https://doi.org/10.3390/en16020745>
- Russell, S., y Norvig, P. (2022). *Artificial Intelligence: A Modern Approach* (4.^a ed.). Pearson.
- Smilkov, D., y Carter, S. (2016). Tensorflow Playground. Consultado el 31 de agosto de 2025, desde <https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=gauss®Dataset=reg-gauss&learningRate=0.03®ularizationRate=0&noise=0&networkShape=3,2&seed=0.63664&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=true&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false>
- Tawil, A., Mohamed, M., Schmoor, X., Vlachos, K., y Haidar, D. (2024). Trends and Challenges towards Effective Data-Driven Decision Making in UK Small and Medium-Sized Enterprises. *Big Data and Cognitive Computing*, 8(7), 10. <https://doi.org/10.3390/bdcc8070079>
- Taye, M. (2023). Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions. *Computers*, 12(5), 3. <https://doi.org/10.3390/computers12050091>