Algoritmos de aprendizaje supervisado utilizando datos de monitoreo de condiciones: Un estudio para el pronóstico de fallas en máquinas

Alexander Huertas Mora

Universidad Santo Tomás
Facultad de Estadística
División de Ciencias Económicas y Administrativas
Bogotá, D.C., Colombia
2020

Algoritmos de aprendizaje supervisado utilizando datos de monitoreo de condiciones: Un estudio para el pronóstico de fallas en máquinas

Alexander Huertas Mora

Trabajo de grado presentado como requisito parcial para optar al título de:
Magister en Estadística Aplicada

Directores:
Andrés Cruz Pérez (M.Sc.)
Oscar Julián Perdomo Charry (Ph.D (c))

Línea de Investigación:
Machine Learning & Deep Learning
Grupo de Investigación:
USTAdistica

Universidad Santo Tomás
Facultad de Estadística
División de Ciencias Económicas y Administrativas
Bogotá, D.C., Colombia
2020

Dedicatoria

A mi madre y Frank, *in memoriam.*

A mi esposa y abuela, gracias.

Resumen

Este trabajo proporciona una visión general de algunos métodos de Machine Learning y Deep Learning como herramientas fundamentales en la detección de fallas potenciales de los activos físicos utilizando técnicas de monitoreo de condiciones, para esto, en la primera parte se aplican algoritmos de aprendizaje supervisado de clasificación y regresión en diferentes casos de estudio; al comparar el desempeño de los modelos se muestra la efectividad de las redes neuronales profundas LSTM, cuyas propiedades son de gran valor en el procesamiento de datos secuenciales y prometen aplicaciones más potentes en la ingeniería de mantenimiento. En la segunda parte se argumenta la efectividad al ajustar apropiadamente la arquitectura de la red neuronal e implementar algoritmos híbridos que maximizan el rendimiento del modelo. En la tercera parte se describe e implementa una aplicación Web para poner en producción un modelo de clasificación de fallas en rodamientos, el algoritmo seleccionado para la solución Web es Gradient Boosting debido al buen desempeño con el conjunto de datos y eficiencia en el uso de recursos computacionales, con este desarrollo se facilita el acceso al usuario final al modelo de clasificación. Por último, se aplica un método de análisis de supervivencia con un estimador estadístico, cuyo propósito es calcular el tiempo medio de vida de la máquina y las curvas de supervivencia, con la finalidad de comparar la probabilidad de falla durante el tiempo de operación del activo físico.

Palabras clave: monitoreo de condiciones; mantenimiento predictivo; machine learning; deep learning; modelos híbridos, confiabilidad; LSTM; industria 4.0; IoT.

Abstract

This paper provides an overview of some Machine Learning and Deep Learning methods as fundamental tools in detecting potential failures of physical assets using condition monitoring techniques, for this, in the first part supervised learning algorithms are applied for classification and regression in different case studies; comparing the performance of models demonstrates the effectiveness of deep neuronal networks LSTM, whose properties are of great value in sequential data processing and promise more powerful applications in maintenance engineering. In the second part effectiveness is argued by optimally adjusting the neural network architecture and implementing hybrid models that maximize model performance. In the third part describes and implements a Web application to put in production a model of classification of failures in bearings, the algorithm selected for the Web solution is Gradient Boosting due to the good performance with the data set and efficiency in the use of computational resources, with this development the end user access to the classification model is improved. Finally, a survival analysis method is applied with a statistical estimator, the purpose of which is to calculate the average life of the machine and the survival curves to compare the probability of failure during the time of operation of the physical asset.

Keywords: condition monitoring; predictive maintenance; machine learning; deep learning; hybrid models, reliability; LSTM; industry 4.0; Iot.

Contenido

Lista de Figuras

Lista de Tablas

1. Introducción

1.1. La ingeniería de mantenimiento en la Industria 4.0

La Industria 4.0 impone cambios de paradigmas que favorecen la productividad, flexibilidad y resiliencia de los sistemas de producción por medio de un modelo mixto entre humanos y máquinas, que aplica tecnologías de Machine Learning (ML) para minimizar la participación humana en los procesos de diagnostico y mantenimiento de activos físicos, con el objetivo de mejorar la detección de fallas potenciales en la maquinaria de manera oportuna. Según Pinto & Cerquitelli (2019), con este nuevo concepto, las empresas están generando enormes cantidades de datos de los activos físicos, que conllevan a la necesidad de aplicar métodos de aprendizaje automático, con el propósito de favorecer la toma de decisiones basada en datos. La ingeniería de mantenimiento predictivo es un pilar de la industria 4.0, en este sentido afirma Brik et al. (2019), el mantenimiento basado en la condición se está convirtiendo en un tema de investigación crucial con la finalidad de disminuir el tiempo y frecuencia de falla, mejorando el rendimiento de los activos físicos productivos.

La fabricación habilitada para internet de las cosas (IoT, por sus siglas en inglés) requiere el uso de técnicas avanzadas para el análisis de datos generados por sensores interconectados, los cuales monitorean las variables críticas con el propósito de mantener altos niveles de disponibilidad en los activos físicos, tal como afirma Zhong et al. (2017), un caso de éxito es el de la compañía General Electric (GE), la cual optimizó los procesos de producción y mantenimiento en un entorno de Big Data, por esto, en 2012 GE introdujo el concepto de internet industrial de las cosas (IIoT, por sus siglas en inglés), que sugiere que las máquinas inteligentes, los análisis avanzados y las personas conectadas son los elementos claves de la fabricación futura, con el fin de permitir una mejor toma de decisiones. Otro caso de éxito es la industria aeroespacial en Estados Unidos, que según Armes & Refern (2013), aplicó con éxito la combinación de grandes conjuntos de datos (fabricación y reparación) usando algoritmos predictivos de Machine Learning, para analizar datos en los entornos de pruebas. Con este impulso agresivo hacia las tecnologías de IIoT, los datos de sensores que se derivan de las condiciones operativas de las máquinas son cada vez más accesibles en muchas industrias, de modo que las herramientas de ML son una solución adecuada para procesar la información de una manera ágil, con la intención de aumentar la competitividad en los agresivos mercados globales. Para Suarez et al. (2017), es posible reducir el estado de indis-

ponibilidad de los activos en un 50 % y aumentar la productividad en los procesos operativos en un 20 % con el uso de herramientas avanzadas de mantenimiento predictivo.

La fabricación inteligente requiere gran demanda de especialistas, con el propósito de diseñar, operar y mantener estas industrias, tal como lo expone Killeen et al. (2019), con miles de millones de dispositivos conectados a internet, el análisis de datos con técnicas de Big Data y algoritmos analíticos se vuelve cada vez más frecuente, lo que impulsa nuevas aplicaciones de mantenimiento predictivo. Es por esto que el gobierno, los gremios industriales y la academia deben apoyar la investigación y aplicación de iniciativas que agreguen valor a los métodos tradicionales de producción, logrando así, que los sistemas informáticos inteligentes se hagan cargo de las tareas en el monitoreo de condiciones que actualmente realizan los humanos; contribuyendo a soluciones ágiles y en tiempo real de los problemas industriales, tales como: paradas de producción, exceso de mantenimiento preventivo a los activos físicos que inducen mayor probabilidad de falla funcional y altos gastos derivados por la deficiente gestión de activos físicos.

1.2. Monitoreo de condiciones

La ingeniería de mantenimiento es la columna vertebral de los procesos productivos, asegurando el mejor rendimiento posible de los activos físicos, eliminado fallos recurrentes y aumentando la vida útil de las máquinas. Según el estándar internacional ISO-14224 "Petroleum, petrochemical and natural gas industries - reliability and maintenance data for equipment" ISO14224 (2016), los procesos de mantenimiento se dividen en dos categorías las cuales se ilustran en la Figura **1-1**.

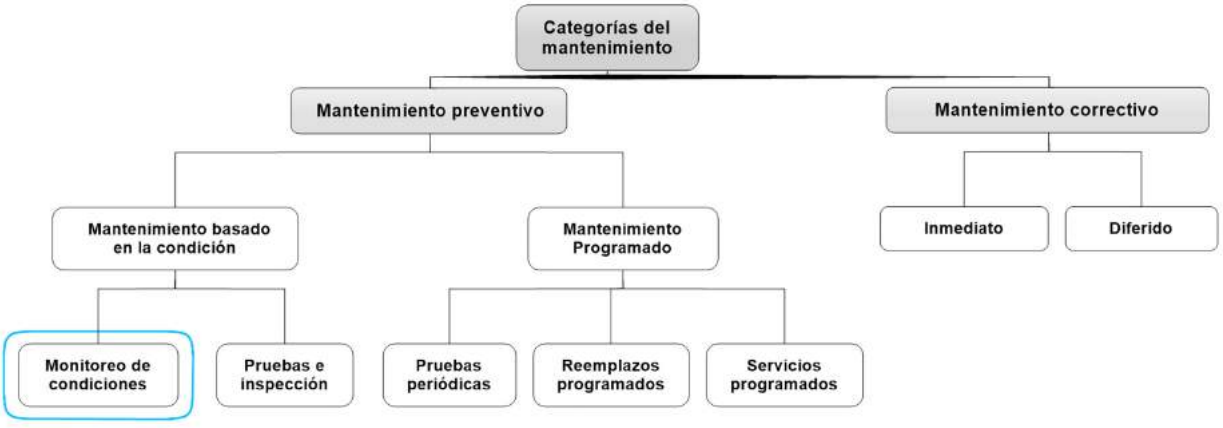


Figura 1-1.: Categorías del mantenimiento

La primera estrategia de mantenimiento es el correctivo, en otras palabras, esto implica operar hasta que el activo físico pierda su funcionalidad, en esta situación la utilización de un componente de la máquina puede incrementarse en cierta medida, pero el tiempo de inactividad por fallas es inevitable; la siguiente estrategia es el mantenimiento preventivo, el cual se clasifica en dos dominios, uno de ellos basado en el tiempo (mantenimiento programado), lo que conlleva costos altos con paradas frecuentes de producción y el segundo implica el monitoreo de condiciones que de acuerdo con Mora (2009), esta estrategia logra maximizar la vida útil del elemento y consigue reducir los costos de mantenimiento.

El propósito del monitoreo de condiciones, es clasificar proactivamente el estado de la máquina o componentes y/o predecir el tiempo hasta la falla (TTF), con el fin de lograr una alerta temprana antes que se pierda la funcionalidad del sistema, estos indicadores se pueden calcular cuando contamos con los datos de sensores y la etiqueta del estado de salud de la máquina o componente, en el momento en que estos datos están disponibles, predomina el enfoque por medio de modelos de ML y redes neuronales profundas también conocidas como Deep Learning (DNN, por sus siglas en inglés); tal como el estudio de Babu et al. (2016), en el cual desarrollan modelos de regresión basado en redes neuronales convolucionales profundas, para la estimación de la vida útil restante (RUL); además está el trabajo de Kraus & Feuerriegel (2019), donde afirman que los modelos de ML y DNN tienen alto grado de flexibilidad y favorecen la detección de patrones de fallas en relaciones no lineales, minimizando el error en el pronóstico, dando como resultado la mejora de las operaciones de mantenimiento.

Los métodos de monitoreo de condición se pueden dividir en dos grupos, el primer grupo, corresponde a las técnicas que recogen datos estructurados multivariados tales como el análisis de vibraciones, monitoreo de variables mecánicas y el análisis dinámico de máquinas eléctricas; el segundo grupo corresponde a las técnicas de procesamiento de imágenes tales como la termografía infrarroja, radiografía y ultrasonido. Actualmente un gran porcentaje de los datos obtenidos de los activos por sistemas de monitoreo están desaprovechados o en algunos casos solo se realizan análisis gráficos básicos con el propósito de encontrar una tendencia que indique una falla potencial o funcional del activo, el inconveniente de este método es que se ignoran la mayoría de las variables adquiridas por los sensores lo que disminuye la efectividad de las acciones proactivas que se deben ejecutar para mantener funcional la máquina. El objetivo de este trabajo es aplicar herramientas de Machine learning y Deep learning como una opción para facilitar la detección temprana de fallas en máquinas y/o componentes utilizando datos estructurados multivariados, en estos conjuntos de datos cada observación se etiqueta con una clase o se calcula el TTF y se aplican algoritmos de aprendizaje supervisado de clasificación y regresión con el fin de posibilitar análisis predictivos en los activos físicos, disminuyendo así los costos asociados a la baja disponibilidad y excesiva ejecución de mantenimientos preventivos en las máquinas.

La clave del éxito de un programa de mantenimiento fundamentado en el monitoreo de condiciones es que las variables que se adquieran estén asociadas a los modos y efectos de las fallas en los componentes del sistema, con la finalidad de encontrar patrones que potencialmente indiquen si el activo se encuentra en estado normal, falla potencial o falla funcional. En términos de Amendola (2014): *"se ha comprobado que un proceso de mantenimiento proactivo bien implementado y gestionado es el mejor método para controlar el riesgo, aumentar la confiabilidad y asegurar la mayor tasa de retorno del activo industrial"* (p. 77). Cada día disminuyen los costos asociados al desarrollo de sensores, sistemas de procesamiento de señales y capacidad de Hardware, esto permite que sin necesidad de grandes inversiones de capital se pueda implementar un sistema de análisis de monitoreo de condiciones aplicando técnicas de ML y DNN e integrarlo a un sistema de alertas en tiempo real que mejore la toma de decisiones de los ingenieros de mantenimiento.

2. Marco teórico y revisión de literatura

En este capitulo se estudia la teoría, antecedentes y estado del arte de las técnicas de Machine Learning y Deep Learning aplicadas en problemas de predicción de fallas en activos físicos, se analizan las principales categorías y algoritmos de aprendizaje supervisado, describiendo algunos estudios previos de investigación que son soporte para este trabajo; la finalidad de esta parte del trabajo es describir los métodos de clasificación y regresión empleados con algoritmos de ML y DNN en series de tiempo, como herramienta esencial para generar una alerta temprana y así programar una intervención correctiva o preventiva antes que suceda una falla funcional en un activo físico, en este sentido, se agrupan técnicas en tres clases como herramientas esenciales en el análisis de los datos derivados del monitoreo de condiciones en mantenimiento. A continuación las clases propuestas:

1. **Análisis exploratorio gráfico:** El análisis gráfico de variables para la detección de fallas potenciales emplea diagramas de dispersión, los cuales ayudan a detectar cambios en la tendencia de la curva o puntos de intersección con limites relacionados a los síntomas de degradación de las partes de una máquina, como precedente en la aplicación de esta técnica se documenta el trabajo de Bahtiar et al. (2018), donde realizan mediciones periódicas las cuales revelan en un análisis gráfico que los niveles de vibración del activo físico aumentan significativamente, superando el nivel de advertencia definido en el estándar internacional ISO 10816-3 “Mechanical vibration - Evaluation of machine vibration by measurements on non-rotating parts”.

2. **Métodos estadísticos:** Los métodos estadísticos parten del supuesto que el ciclo de vida del activo y su tasa de fallas siguen una distribución específica, en términos de Scheu et al. (2017): *“la distribución Weibull se aplica ampliamente en la ingeniería de confiabilidad, ya que ofrece un gran potencial para representar varias características al ajustar sus parámetros”* (p. 28). Otro método es el análisis de supervivencia que estudia la duración de tiempo hasta que ocurran uno o más eventos, en nuestro caso de estudio estos eventos hacen referencia a fallas en las máquinas, tal como lo expone el trabajo de Hernández (2010).

3. **Técnicas de Machine Learning y Deep Learning:** Con el desarrollo de múltiples algoritmos de ML y DNN es posible clasificar el estado operacional de una máquina en un rango de tiempo, con el objeto de generar una alerta temprana en caso de que se

detecte una posible falla potencial o funcional, con estas técnicas se obtienen resultados con alta precisión gracias a la flexibilidad para la configuración de hiperparámetros de estos modelos. Según Chen et al. (2019), debido a la complejidad y volumen de los datos obtenidos en mantenimiento, las técnicas de Machine Learning cada día ganan mayor atención por sus múltiples ventajas en la minería de datos. En el trabajo de Ellefsen et al. (2019), desarrollan una técnica de aprendizaje profundo no supervisado en una etapa inicial de preentrenamiento para extraer la degradación de una turbina aeronáutica, con la combinación de aprendizaje no supervisado y supervisado obtienen una tasa baja de error al calcular el tiempo hasta la falla, igualmente lo comparan con otros métodos de regresión con algoritmos de aprendizaje profundo.

Una mayor disponibilidad de datos en mantenimiento y un aumento en el poder computacional nos llevan a desarrollar los métodos de Machine Learning y las redes neuronales profundas, según Aggarwal (2018), las redes neuronales son teóricamente capaces de aprender cualquier función matemática siempre y cuando se cuente con el suficiente volumen de datos de entrenamiento, esta nueva área se conoce como aprendizaje profundo.

2.1. Machine Learning

Como señala VanderPlas (2016), en la práctica Machine Learning implica construir modelos matemáticos para descubrir el comportamiento de los datos por medio de potentes algoritmos, el "aprendizaje" se logra cuando se ajustan automáticamente los parámetros de los modelos hasta que se adaptan a los datos de entrenamiento observados, con el modelo ajustado y evaluado se pueden realizar predicciones o clasificaciones de las nuevas observaciones. En la figura **2-1** se presenta la estructura jerárquica del Machine Learning.

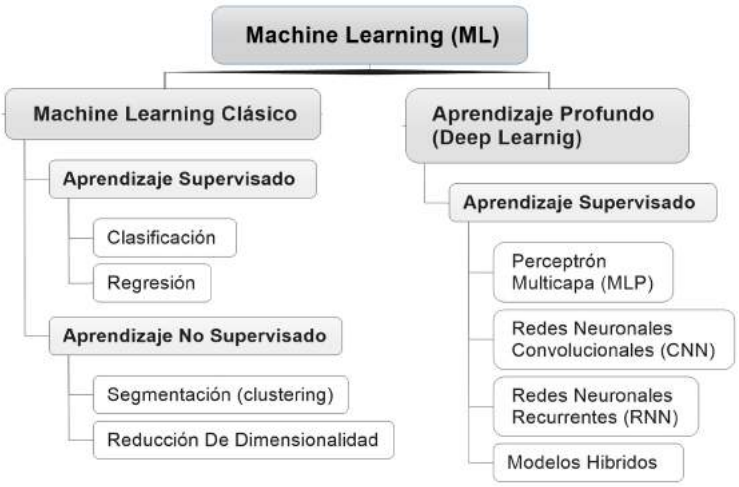


Figura 2-1.: Jerarquia del Machine Learning

Para Kassambara (2018), las dos categorías de los métodos de Machine Learning son:

- **Aprendizaje no supervisado (Unsupervised Learning):** En esta categoría no se conocen las "etiquetas" de los datos, es decir que no nos guiamos por ideas previas de los grupos a los cuales pertenecen las muestras, por lo que el algoritmo debe aprender como describir la estructura de los datos, estos métodos incluyen principalmente la agrupación (clustering) y los métodos de análisis de componentes principales (PCA, por sus siglas en inglés).

  Una de las técnicas no supervisadas aplicadas en la ingeniería de mantenimiento es la segmentación, cuya intensión es identificar patrones o grupos de un conjunto de datos multidimensionales obtenidos de sensores que monitorean variables operativas de alta importancia; como precedente de esta metodología aplicada a mantenimiento está el trabajo de Amruthnath & Gupta (2018), donde proponen e implementan una metodología para detección y clasificación de fallas de un ventilador utilizando aprendizaje no supervisado, en este estudio desarrollan un modelo de rápida implementación con una mínima dependencia de datos históricos de vibraciones de la máquina, utilizando algoritmos de agrupación de modelos de mezcla Gaussiana (GMM, por sus siglas en inglés) y K–Means con un resultado final en términos de precisión del 82.96 %.

- **Aprendizaje supervisado (Supervised Learning):** En el aprendizaje supervisado los predictores y las variables de respuesta son conocidos para construir modelos matemáticos con la intensión de predecir o clasificar observaciones que se obtengan posteriormente, estos métodos se consideran supervisados debido a que el modelo se construye con los valores conocidos de las observaciones, es decir, la máquina "aprende"de los datos conocidos con el propósito de predecir resultados futuros.

  Los algoritmos de aprendizaje supervisado se categorizan mediante la diferenciación con respecto al tipo (cuantitativo o cualitativo) de la variable de salida involucrada en el problema, la regresión se utiliza cuando el resultado es cuantitativo y la clasificación cuando el resultado es cualitativo. En la figura **2-2** se muestran algunos algoritmos clásicos de regresión y clasificación usados comúnmente en aplicaciones de ingeniería.

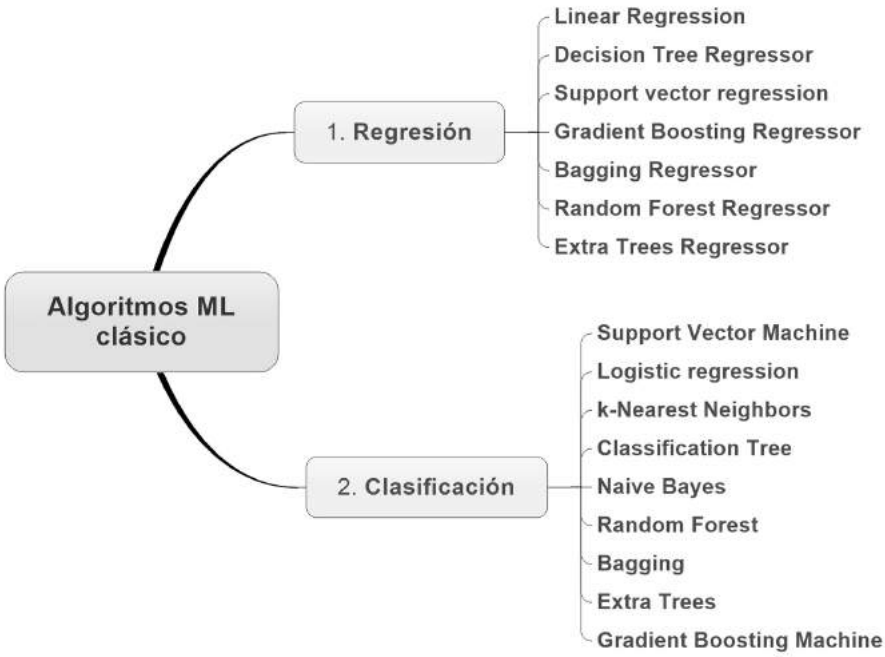


Figura 2-2.: Algoritmos clásicos de regresión y clasificación

2.1.1. Descripción de los algoritmos de Machine Learning

En esta sección, se efectúa un recorrido por los algoritmos de aprendizaje supervisado relevantes que se aplican en este trabajo, sin embargo, es importante aclarar que existen una gran variedad de algoritmos de ML que se emplean en diferentes áreas del conocimiento. En la tabla **2-1** se observa una breve descripción de los diversos algoritmos de ML aplicados en los casos objeto de este estudio, con sus principales fortalezas y limitaciones. Con el fin de comparar el rendimiento de los modelos aplicados, se deben considerar diferentes algoritmos de ML en el desarrollo de experimentos con nuevos conjuntos de datos, de igual modo, es crucial el ajuste de hiperparámetros para mejorar las métricas de desempeño, según Olson et al. (2018), la sintonización de hiperparámetros a través de la búsqueda de cuadrícula, mejora la exactitud del algoritmo en un 3 % a 5 % en comparación con su configuración de referencia, en general, los resultados muestran que seleccionar el mejor algoritmo y ajustarlo conduce a aproximadamente un aumento del 20 % en la exactitud del modelo, por esta razón en la tabla **2-1** se incorpora una columna con los hiperparámetros principales que se ajustan en cada uno de los algoritmos para conducir a una mejora en la precisión predictiva.

Considerando que el alcance de este trabajo es la aplicación de modelos de ML y DNN en mantenimiento, se incluye en tabla **2-1** una referencia para el lector que pretenda profundizar en el desarrollo teórico de cada algoritmo.

Tabla 2-1.: Síntesis de los algoritmos de Machine Learning

Algoritmo	Descripción	Fortalezas	Limitaciones	Hiperparámetros	Referencias
Gradient Boosting	Algoritmo de conjunto con optimización numérica donde el objetivo es minimizar la pérdida del modelo agregando secuencialmente árboles de decisión.	Excelente precisión predictiva, flexibilidad para ajustarse a diferentes clases de datos, las predicciones se hacen por mayoría de votos de los alumnos débiles.	El aumento de gradiente continuará mejorando con el propósito de minimizar todos los errores, lo que puede causar un excesivo sobreajuste.	Número de arboles, Profundidad del árbol, tasa de aprendizaje y muestreo de filas.	Kurama (2020)
XGBoost	Algoritmo que recientemente ha dominado el aprendizaje automático aplicado, es una implementación de Gradient Boosting para maximizar velocidad de entrenamiento y el rendimiento de modelo.	Utilización de todos los núcleos de la CPU durante el entrenamiento, optimización de recursos de memoria y computación distribuida lo que permite manejar grandes conjuntos de datos	La alta flexibilidad da como resultado muchos hiperparámetros que interactúan fuertemente en el comportamiento del modelo.	Número de arboles, Profundidad del árbol, tasa de aprendizaje y muestreo de filas.	Brownlee (2020*b*)
Random Forest Extra Trees	Modelos compuesto por muchos árboles de decisión, al entrenar cada árbol se aprende de una muestra aleatoria de los puntos de datos y de un subconjunto de características. La diferencia entre Random Forest y Extra Trees es que muestrea sin reemplazo y los nodos se distribuyen en divisiones aleatorias, no mejores divisiones como Randon Forest	Las predicciones finales del bosque aleatorio se hacen promediando las predicciones de cada árbol individual reduciendo el problema de sobreajuste y varianza. para el caso de Extra Trees dado que las divisiones se eligen al azar para cada característica, es menos costoso desde el punto de vista computacional que un Random Forest	Genera muchos árboles lo que lo hace costoso computacionalmente, requiere más tiempo para entrenar en comparación con los árboles de decisión. Puede ajustarse en exceso a los conjuntos de datos que son particularmente ruidosos. Para el caso de algoritmos de regresión, no predice más allá del rango en los datos de entrenamiento.	Número de arboles, profundidad del árbol, muestras necesarias para una hoja y divisiones de nodo interno.	Koehrsen (2018) Bhandari (2018)
Logistic Regression	Algoritmo que se utiliza para problemas de clasificación binaria, la base de la regresión logística es la función logística (sigmoid) que toma cualquier número de valor real y le asigna a un valor entre 0 y 1.	La regresión logística es un algoritmo de clasificación simple pero muy efectivo, tiene una relación muy estrecha con las redes neuronales. El tiempo de entrenamiento es menor que otros algoritmos.	Dificultad para capturar relaciones complejas ya que tiene una superficie de decisión lineal, en los conjuntos de datos de alta dimensión puede generar sobreajuste.	Regularización (C) y penalización a la función de pérdida L1 y L2.	Chavez (2019)
Bagging	Algoritmo que genera varios subconjuntos de datos a partir de una muestra de entrenamiento elegida al azar con reemplazo.	Se centrará principalmente en obtener un modelo de conjunto con menos varianza, para producir modelos fuertes con menor sesgo.	Introduce una pérdida de interpretabilidad de un modelo, puede ser costoso computacionalmente.	número de arboles y número máximo de muestras con reemplazo.	Rocca (2019)
Naive Bayes	Algoritmo de clasificación que aplica explícitamente el teorema de Bayes bajo el supuesto que todas las variables observadas son independientes.	Baja propensión al sobreajuste, entrenamiento y predicción rápida, uso modesto de capacidad de CPU y memoria ya que no hay gradientes o actualizaciones iterativas de parámetros para calcular.	El rendimiento es sensible a los datos asimétricos, es decir, cuando los datos de entrenamiento no son representativos de las distribuciones de clase en la población general.	No se ajustaron hiperparámetros	Catanzarite (2018)
Decision tree	Descompone un conjunto de datos en subconjuntos más pequeños con un aumento en la profundidad del árbol, el objetivo aumentar la predicción por medio de nodos de decisión.	Representación visual de todos los resultados posibles, requiere menos limpieza de datos, no está influenciado por valores atípicos, maneja variables numéricas y categóricas.	El cálculo puede ser más complejo lo que implica un mayor tiempo para entrenar el modelo, un pequeño cambio en los datos puede causar un gran cambio en la estructura del árbol.	Profundidad del árbol, número mínimo de muestras del nodo, criterio.	Jain (2017)
Support Vector Machine	Los SVM encuentran una línea o hiperplano entre diferentes clases de datos, calculan un límite de margen máximo que conduce a una partición homogénea de todos los puntos de datos.	Es útil tanto para datos separables linealmente como no separables linealmente, es eficaz en los casos en que varias dimensiones son mayores que la cantidad de muestras. Eficiente uso de memoria.	No funciona muy bien cuando el conjunto de datos tiene mucho ruido, elegir el kernel y los parámetros correctos puede ser costoso computacionalmente	Parámetros del núcleo, tipo de núcleo (kernel)	Bedell (2018)
k-Nearest Neighbors	Utiliza la similitud de características para predecir el clúster en el que caerá el nuevo punto. La idea principal es que el valor o la clase de una observación está determinado por las observaciones que lo rodean.	No hace una suposición sobre el patrón de distribución de datos subyacente, se agiliza el tiempo de entrenamiento ya que almacena el conjunto de entrenamiento y aprende de él solo al momento de hacer predicciones.	Computacionalmente costoso, los datos deben pre procesarse y escalarse, las observaciones se usarán solo en el momento de la predicción por lo que es un paso costoso, sensible a datos ruidosos y atípicos.	Número de vecinos	Khandelwal (2018)

2.1.2. Regresión

En términos de Lindholm et al. (2019), la regresión se refiere al problema de aprender las relaciones entre las variables de entrada $\boldsymbol{X}$ (cualitativas o cuantitativas) y una variable de salida cuantitativa y, el objetivo es encontrar un modelo f que relacione las variables de entrada con la variable de salida, matemáticamente se describe con la ecuación 2-1:

$$y = f(\boldsymbol{X}) + \varepsilon, \tag{2-1}$$

donde y es la variable respuesta, $\boldsymbol{X} = [x_1, x_2, ..., x_p]^T$ son las variables de entrada y ε es un término de ruido o error que describe todo lo que el modelo no puede capturar, en este sentido afirma Téllez & Morales (2016): *"las propiedades de la variable aleatoria ε dependen de ciertas situaciones particulares, pero a menudo se supone que sigue una distribución normal con media cero y varianza σ^2"* (p. 2).

En el aprendizaje automático, el énfasis está en estimar algunos resultados de salida $\hat{y}_\star$ (aún no vistos) para una nueva entrada $\boldsymbol{X} = [x_{\star 1}, x_{\star 2}, ..., x_{\star p}]^T$, para hacer una predicción en los datos de prueba $X_\star$ (test), por lo tanto obtenemos la prediccíon tal como se muestra en la ecuación 2-2:

$$\hat{y}_\star = \beta_0 + \beta_1 x_{\star 1} + \beta_2 x_{\star 2} + ... + \beta_p x_{\star p}, \tag{2-2}$$

donde $\hat{y}_\star$ es la estimación, y los coeficientes $\beta_0, \beta_1, ..., \beta_p$ para los cuales nos referimos como los parámetros del modelo que se aprenden de un conjunto de datos de entrenamiento (train).

2.1.3. Clasificación

Para Kuhn & Johnson (2013), la clasificación es un problema de modelado que asigna una etiqueta de clase de tipo categórica y discreta a cada observación, en la practica la asignación de etiquetas de clase se utiliza habitualmente para la toma de decisiones, pero es importante resaltar que los modelos de clasificación tienen la capacidad de producir una predicción de valor continuo. Según Lindholm et al. (2019), en un enfoque estadístico, entendemos la clasificación como el problema de predecir las probabilidades de clase, la $Pr(y \mid \boldsymbol{X})$ describe la probabilidad para la salida (una etiqueta de clase) dado que conocemos la entrada $\boldsymbol{X}$.

Los modelos de clasificación se pueden dividir en dos tipos, cuando se asigna una de dos posibles clases se considera un problema de clasificación binaria y la clasificación multiclase aplica cuando a todas las observaciones se les asigna una de tres o más clases.

2.2. Redes neuronales

Las redes neuronales tradicionales con una capa oculta se han utilizado y analizado exitosamente en los años ochenta y principios de los noventa como el estudio que sugirió Rodins & Amin (1992), donde aplica redes neuronales artificiales para la predicción de maniobras en combate aéreo. Sin embargo, en los últimos años se ha evidenciado que las redes neuronales profundas con varias capas ocultas, o simplemente aprendizaje profundo, son aún más poderosas. Desde la perspectiva de la confiabilidad de los activos físicos, actualmente es imprescindible implementar metodologías modernas tales como las redes neuronales profundas, con la finalidad de detectar proactivamente fallas potenciales que conllevan a paradas de producción y/o aumentan significativamente los gastos de mantenimiento en las organizaciones, en este sentido señala Chen et al. (2019), que el aprendizaje profundo ha sido investigado en los últimos años como una herramienta fundamental para las tácticas de mantenimiento predictivo.

2.2.1. Redes neuronales de aprendizaje profundo

Una red neuronal de dos capas es un modelo muy útil, sin embargo, como lo expone Lindholm et al. (2019), el poder descriptivo real de una red neuronal se logra cuando apilamos múltiples capas, lo que se conoce como una red neuronal de aprendizaje profundo (DNN, por sus siglas en inglés), en la figura **2-3** se muestra un ejemplo de una red neuronal profunda, la cual consiste en una capa de entrada con cuatro unidades de memoria, dos capas ocultas con tres unidades de memoria cada una y una capa de salida con dos unidades de memoria; esta configuración de capas permite modelar relaciones complicadas, posicionándose como uno de los métodos más recientes y con mayor número de aplicaciones en el aprendizaje automático.

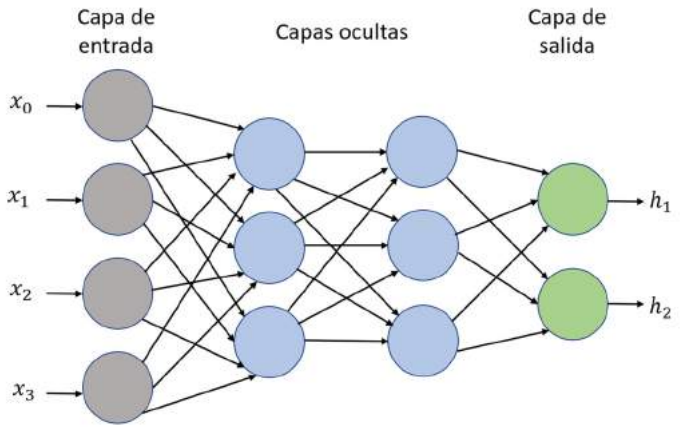


Figura 2-3.: Red neuronal profunda

Las características de las redes neuronales de aprendizaje profundo son muy útiles en el estudio de pronósticos de series de tiempo, de acuerdo con Brownlee (2019), el estudio de las redes neuronales en series de tiempo aporta significativamente a problemas con dependencias complejas no lineales, entradas con múltiples variables y pronósticos de varios pasos de tiempo, estas ventajas resultan ser muy prometedoras en la aplicación moderna para el pronóstico de fallas, utilizando datos de series de tiempo adquiridos por sensores para el monitoreo de condiciones en activos físicos.

2.2.2. Perceptrones multicapa aplicados a series de tiempo

Los perceptrones multicapa (MLP, por sus siglas en inglés) son redes neuronales simples que pueden aplicarse a problemas de pronósticos de secuencias en series de tiempo, tal como lo expone en su trabajo Dorffner (1996), en el cual realiza una descripción de las propiedades de las redes neuronales para el procesamiento de series temporales y resalta el gran valor potencial en el campo de la predicción y el reconocimiento de patrones ocultos en los datos; entre las ventajas de las redes MLP para el procesamiento de series de tiempo, sobresale su capacidad para soportar niveles altos de ruido en los datos de entrada y su facilidad para "aprender" independiente de las relaciones lineales y no lineales existentes en las variables que son objeto de estudio.

Una de las principales limitaciones de las redes MLP en el estudio de las series de tiempo se debe a que los pasos de tiempo se modelan como una variable de entrada, lo que significa que la red pierde la oportunidad de aprovechar la estructura u orden secuencial entre las observaciones, esta limitación afecta directamente la precisión del modelo. Como antecedente en una aplicación de una red MLP en mantenimiento está el estudio de Sharma et al. (2015), en el cual diagnostican fallas en máquinas rotativas utilizando un método de clasificación multiclase con un algoritmo Extreme Learning Machine (ELM), comparan su rendimiento con el perceptrón multicapa (MLP) y concluyen que el método ELM logra una mayor precisión de clasificación que la red MLP.

2.2.3. Redes neuronales convolucionales aplicadas a series de tiempo

Las redes neuronales convolucionales (CNN, por sus siglas en inglés) son un tipo de red especial que fue diseñada con el propósito de resolver problemas de identificación de imágenes, pero también se ha mostrado su utilidad en la clasificación de series de tiempo, tal como lo proponen Yang et al. (2015), al desarrollar un nuevo método que adopta una red CNN, con el propósito de automatizar la extracción de características, a fin de facilitar la tarea de clasificación de reconocimiento de movimientos en la actividad humana, utilizando datos de

series de tiempo multivariados que no fueron procesados previamente.

La capacidad de las CNN para "aprender" y extraer automáticamente características cuando los datos de entrada tienen una tipología de serie de tiempo, posibilita aplicar modelos que incluyen capas convolucionales en problemas de clasificación de fallas, empleando los datos obtenidos por técnicas de monitoreo de condiciones. Como antecedentes en la aplicación de redes CNN en mantenimiento predictivo se encuentran los siguientes trabajos: primero, el artículo de Pinto & Cerquitelli (2019), en el cual concluyen que el modelo de red neuronal CNN aplicado en la clasificación de fallas produce un incremento del 9 % en términos de precisión y una mejora de sensibilidad del 4 % al compararlo con otras técnicas de Machine Learning. Segundo, el trabajo de Hasegawa et al. (2019), en cuyo estudio aplican un clasificador con redes CNN para conjuntos de datos de vibraciones, evidenciando la efectividad del modelo para la detección de fallas para dos de las tres condiciones operativas que evalúan, con el cual obtienen una medida de sensibilidad para la clase "falla" de 77.6 %.

2.2.4. Redes neuronales recurrentes aplicadas a series de tiempo

Las redes neuronales recurrentes (RNN, por sus siglas en inglés) son actualmente una poderosa herramienta dentro del aprendizaje supervisado específicamente en las aplicaciones con series de tiempo, de acuerdo con Sak et al. (2014), los modelos RNN son muy diferentes a otras redes neuronales de aprendizaje profundo ya que contienen bucles en su arquitectura, los cuales alimentan las activaciones de red de un paso de tiempo anterior como entradas para la siguiente capa, esta característica permite mejorar las predicciones en el paso de tiempo actual, las conexiones recurrentes agregan memoria a la red y posibilitan aprovechar el orden secuencial de las observaciones, manteniendo una memoria temporal dinámica, en otros términos, el modelo puede retener información sobre el pasado facilitando descubrir correlaciones entre observaciones que están muy distanciadas en el tiempo.

En la figura **2-4** se observa una parte de la red neuronal A, que recibe una entrada X_t y genera un valor h_t con un bucle en la unidad oculta, una red neuronal recurrente estándar puede considerarse como una copia de la misma estructura, esta copia es la entrada para la siguiente capa.

El trabajo de Lee et al. (2019), presenta una aplicación en mantenimiento con el propósito de monitorear la condición de cojinetes, empleando una red RNN donde clasifican los estados del componente (falla, alerta y normal), en este estudio obtienen un resultado en términos de precisión promedio de 93 %, los autores concluyen que por medio de los datos de monitoreo de condición efectivamente es posible evaluar la degradación del activo físico.

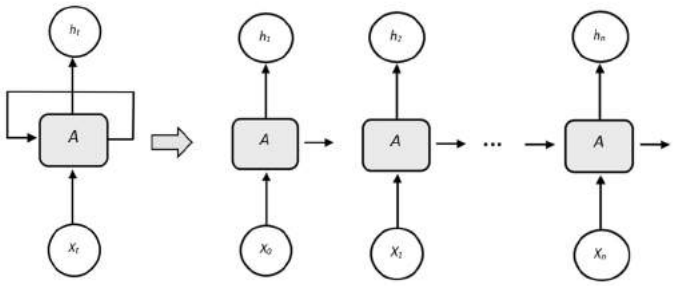


Figura 2-4.: RNN como una red neuronal profunda en el tiempo

Uno de los algoritmos derivados de las redes recurrentes que se destaca por sus aplicaciones óptimas en diferentes áreas, es la red neuronal de memoria a corto y largo plazo (LSTM, por sus siglas en inglés), en este sentido afirma Brownlee (2017): *"pero es el modelo Long Short Term Memory el que cumple la promesa de las RNN para la predicción de secuencias. por lo que hay tanto ruido y aplicaciones de LSTM en este momento"* (p. 10).

2.2.5. Redes neuronales de memoria a corto y largo plazo

Los modelos de memoria a corto y largo plazo son un tipo especial de red neuronal recurrente, ya que incorporan una serie de pasos para decidir qué información va a ser almacenada y cual borrada, la red LSTM está compuesta por capas de neuronas que para este caso se llaman unidades de memoria, las cuales tienen una formulación única que le permite evitar las dificultades inherentes de las RNN en el entrenamiento del modelo, en este sentido afirma Malhotra et al. (2015), que las redes neuronales LSTM superan el problema experimentado por las RNN mediante el uso de compuertas en las celdas, las cuales evitan que el contenido de la memoria sea perturbado por entradas y salidas irrelevantes para la predicción, esta capacidad habilita a las redes LSTM como una técnica viable para modelar el comportamiento en series de tiempo.

De acuerdo con Brownlee (2017), la clave de la celda de memoria en la red LSTM son sus tres compuertas, la compuerta que decide qué información desechar de la celda, la compuerta de entrada que determina que valores utilizar para actualizar el estado de la memoria y la compuerta de salida que define la salida de la celda partiendo de la información que contiene las compuertas de entrada y la memoria de la celda. Todas las redes neuronales LSTM tienen la forma de una cadena de módulos repetitivos de red neuronal, como se muestra en la figura **2-5** que fue tomada de Phi (2018), en la cual se observa que la estructura de repetición tiene cuatro capas de redes neuronales interconectadas.

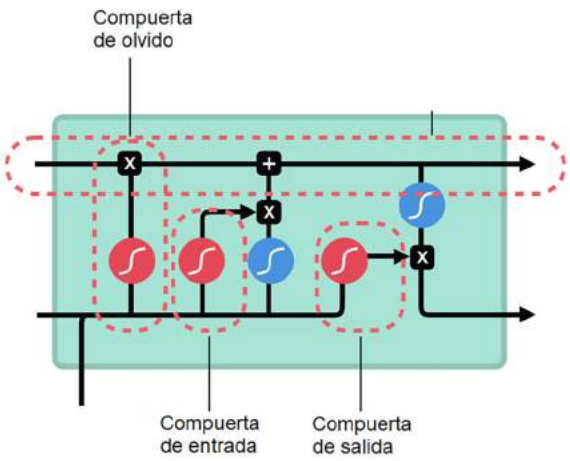


Figura 2-5.: Red LSTM, Phi (2018)

La figura **2-5** indica que cada línea transporta un vector completo, desde la salida de un nodo hasta las entradas de otros; los cuadros negros representan puntos de operaciones puntuales, como la suma de vectores, mientras que los círculos rojos son capas de redes neuronales aprendidas, los círculos azules ayudan a regular los valores que fluyen por la red, la fusión de líneas denota concatenación, mientras que una bifurcación de línea denota que su contenido se copia y las copias van a diferentes ubicaciones. En el trabajo de Phi (2018), se explica detalladamente la estructura y funcionamiento de la red LSTM.

Un problema que surge con las redes neuronales convencionales es el impacto de los datos en tiempos anteriores, estos suelen perder significancia a medida que avanza el algoritmo, con las redes LSTM garantizamos una memoria de corto y largo plazo durante un amplio periodo de procesos en el algoritmo. A continuación, se presentan algunas arquitecturas de las redes LSTM que se usan en aplicaciones de series de tiempo:

- **Vanilla LSTM:** Es una arquitectura simple con una capa de entrada, una celda oculta LSTM y una capa de salida, este tipo de red se usa en problemas de predicción con secuencias cortas, una de las ventajas para la aplicación en series de tiempo se debe a que la predicción de secuencia está en función de los pasos de tiempo anteriores.
- **Stacked LSTM:** Es una arquitectura de tipo apilada que está compuesta por múltiples capas ocultas de bloques de memoria LSTM y en algunos casos capas MLP, a este tipo de arquitectura profunda se le atribuye el excelente desempeño en la solución de problemas de alto nivel de complejidad; de acuerdo con Hermans & Schrauwen (2013), en este tipo de red cada capa resuelve gradualmente una parte de la predicción para luego pasarla a la siguiente capa hasta que obtenemos la información de salida.

- **Bidirectional LSTM:** Las redes LSTM bidireccionales (BLSTM) buscan aumentar la precisión al permitir que el modelo se entrene en la secuencia de entrada hacia adelante y también con la secuencia de entrada con una inversión de tiempo (hacia atrás) para luego enlazar el resultado final.

Algunos artículos documentan aplicaciones en técnicas de monitoreo de condiciones utilizando redes LSTM; Zhang et al. (2018), desarrollan un enfoque basado en la arquitectura LSTM para rastrear la degradación de un sistema y predecir el tiempo de vida útil restante (RUL). De acuerdo con el trabajo de Dong et al. (2017), la predicción del RUL basada en datos se aplica con éxito aprovechando un enfoque de red LSTM, con el fin de proporcionar una predicción precisa en los procesos de mantenimiento de activos físicos. En el trabajo de Wu et al. (2018), los autores proponen la utilización de redes neuronales Vanilla LSTM en un caso de monitoreo de condiciones para motores tipo turboventilador de aviones, el rendimiento de la red neuronal Vanilla LSTM lo comparan con diferentes configuraciones de una red RNN, mostrando la mejora del rendimiento del modelo logrado por Vanilla LSTM.

2.2.6. Redes neuronales híbridas profundas

Al combinar las capacidades de las redes CNN, RNN, LSTM y MLP se obtiene una arquitectura de red neuronal híbrida, este tipo de arquitectura favorece la flexibilidad, eficiencia y maximiza las áreas de aplicación, según Brownlee (2019), los modelos híbridos están convirtiéndose en uno de los campos de estudio de mayor importancia para el desarrollo de series de tiempo con redes neuronales profundas; una red puede combinar las fortalezas de la red CNN para la extracción de características de los datos de entrada con la capacidad de la arquitectura LSTM en el medio para la predicción con datos secuenciales y un MLP de salida, este tipo de arquitectura se define como CNN-LSTM, otro tipo de red neuronal híbrida es la convolucional LSTM (ConvLSTM) que es una extensión de la CNN-LSTM, en la red ConvLSTM las unidades de memoria LSTM utilizan procesos convolucionales con el fin de leer los datos de entrada, según Trifa et al. (2017), esta arquitectura se puede usar en datos espacio-temporales y una de sus fortalezas es que reduce el número de parámetros del modelo, lo que incrementa la eficiencia computacional del hardware cuando se entrena la red neuronal.

Se presentan algunos precedentes en la aplicación de modelos híbridos en datos de tipo secuencial, en el trabajo de Swapna et al. (2018), desarrollan un sistema automatizado no invasivo basado en redes neuronales de aprendizaje profundo para realizar la clasificación con el propósito de detectar la arritmia cardiaca, comparando el rendimiento con arquitecturas híbridas de aprendizaje profundo que combinan capas convolucionales, recurrentes y LSTM; otra aplicación es el trabajo de Gunawan et al. (2018), en el cual implementan arquitecturas híbridas BLSTM-CNN, BLSTM-LSTM y BLSTM-CNN-LSTM para el procesamiento de lenguaje natural del idioma indonesio.

3. Análisis de experimentos

En este capitulo se aplican técnicas clasificación y regresión con algoritmos de Machine Learning y redes neuronales profundas, en la primera sección se describen los casos de estudio con las respectivas propiedades de estos conjuntos de datos, luego se analizan las métricas de desempeño para datos desequilibrados, se compara el rendimiento de los algoritmos en cada caso de estudio, se profundiza en el diagnostico y ajuste de arquitectura de las redes LSTM, se finaliza con un ejercicio de regresión para calcular el tiempo hasta la falla de los activos donde se compara la precisión de los modelos con métricas especializadas para tal fin.

3.1. Datos disponibles para los casos de estudio

Con el propósito de ejecutar los experimentos se seleccionaron tres conjuntos de datos de dominio público, las cuales se ajustan a las aplicaciones típicas de monitoreo de condiciones en la práctica de la ingeniería de mantenimiento, la estructura y características inherentes de cada caso de estudio soportan diferentes técnicas de análisis, permitiendo una visión general en los problemas para el pronostico de fallas de la gestión de activos físicos; a continuación, se realiza una breve descripción de los conjuntos de datos y casos seleccionados:

3.1.1. Caso de estudio 1: Turbina industria aeronáutica

El conjunto de datos publicado por Prognostics Center of Excellence NASA (2008), contiene el registro de datos de simulación en la degradación de un motor tipo turbina "Turbofan Engine Degradation Simulation Data Set", el conjunto de datos se tomó del repositorio del centro de pronósticos de la NASA y consta de datos de entrenamiento, prueba y tiempo de operación hasta la falla de 100 motores. Este caso de estudio se adapta a problemas de monitoreo de condiciones dinámicas con conjuntos de datos desequilibrados, en una secuencia de tiempo con características multivariadas, para los cuales se aplica modelos de clasificación binaria con el propósito de generar una alerta para detectar si el activo está en probable falla potencial o si se encuentra en condiciones normales de operación. En este caso estudio también se ajusta un algoritmo de regresión con el objetivo de estimar el tiempo hasta la falla del activo. En la tabla **3-1** se observan las características resumidas de este conjunto de datos.

3.1.2. Caso de estudio 2: Simulación datos mantenimiento

El conjunto de datos publicado por Patel (2018), contiene el registro de mantenimiento y monitoreo de condiciones de 100 máquinas con similares características técnicas, esta incluye información de telemetría (voltaje, rotación, presión y vibraciones), además este conjunto de datos incorpora información histórica del registro de mantenimiento, modelos de las máquinas, errores registrados y fallas por componente. Este caso de estudio aplica para modelos de clasificación multiclase en series de tiempo, con el propósito de detectar la falla funcional en uno o varios componentes del sistema con datos severamente desequilibrados. En la tabla **3-1** se observan las características resumidas de este conjunto de datos.

3.1.3. Caso de estudio 3: Análisis de vibraciones

El conjunto de datos publicado por Huang & Baddour (2019), contiene el registro de señales de vibración (aceleración y velocidad) recolectadas en rodamientos de diferentes condiciones de salud con velocidad de rotación variable en el tiempo; las condiciones de salud del rodamiento que se evalúan en este trabajo son: saludable y en falla con un defecto de la pista interna. Este caso de estudio aplica para modelos de clasificación binaria en técnicas de monitoreo de condiciones con datos equilibrados por cada clase, el objetivo es detectar la falla potencial con las señales de vibraciones antes que ocurra una avería total del rodamiento que afecte la función requerida del activo. En la tabla **3-1** se observan las características resumidas de este conjunto de datos.

Tabla 3-1.: Características de los conjuntos de datos

conjunto de datos	Total datos	Variables	Observaciones Train	Observaciones Test	Serie tiempo	Desequilibrio clases
Caso de estudio 1	944.356	28	20.631	13.096	Si	Si
Caso de estudio 2	15.770.520	18	683.388	192.752	Si	Si
Caso de estudio 3	16.000.000	2	4.000.000	4.000.000	No	No

3.2. Propiedades de los datos obtenidos por técnicas de monitoreo de condiciones

En la práctica los conjuntos de datos inherentes al mantenimiento predictivo generalmente comparten una o varias de las siguientes propiedades:

- **Multivariados:** Los datos tienen la forma de una matriz con múltiples variables y observaciones, por ejemplo, en un monitoreo de un sistema eléctrico podríamos tener múltiples observaciones de variables tales como voltaje, potencia, corriente, factor de potencia, entre otros.

- **Orden secuencial:** Las observaciones están organizadas como una serie de tiempo con una frecuencia o pasos de tiempo configurada por el sistema control que adquiere los datos, estas observaciones tienen un periodo uniforme a lo largo del tiempo y pueden registrarse en diferentes unidades tales como segundos, horas, días, ciclos, entre otras.

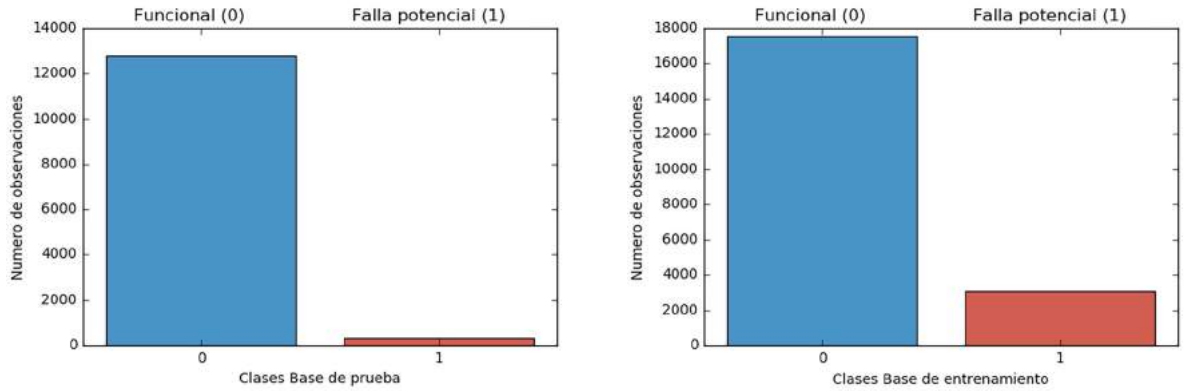
- **Diferentes escalas:** Las variables objeto del estudio pueden tener rangos, escalas o unidades de medida distintas, por ejemplo, en un análisis de vibraciones se adquieren datos de desplazamiento en unidades que equivalen a una millonésima parte de un metro ($1 \times 10^{-6}\ m$) cuya unidad en el sistema métrico se escribe como micrones (μm), mediciones de velocidad que se define como la rata de cambio del desplazamiento con respecto al tiempo y se mide en unidades de milímetros por segundo ($\frac{mm}{s}$), por ultimo valores de aceleración que se define como la rata de cambio de la velocidad con respecto al tiempo y se mide en unidades de milímetros por segundo al cuadrado ($\frac{mm}{s^2}$).

- **Datos no balanceados:** Los datos para la clasificación del estado del activo físico en la base de entrenamiento, usualmente no están equilibrados, tal como lo expone Brownlee (2020*a*), el análisis de anomalías de un sistema es un caso típico donde la distribución de clases está inherentemente desequilibrada; cuando se evalúan los datos de monitoreo de condición habitualmente el número de fallas registradas por el sistema durante un rango de tiempo, son mucho menores a las observaciones del sistema en estado normal, esto representa un desequilibrio leve o severo entre la clase mayoritaria (estado normal o funcional) y la clase minoritaria (falla funcional o potencial).

Partiendo del hecho que la clase minoritaria es la de mayor interés para los ingenieros de mantenimiento cuando se plantean problemas de predicción de fallas, es fundamental escoger las métricas apropiadas con el objeto de evaluar y comparar el desempeño de los modelos.

3.3. Métricas para evaluación de modelos de clasificación con datos desequilibrados

De acuerdo con Branco et al. (2015), la métrica más usada para evaluar el desempeño de los modelos de clasificación es la exactitud (Accuracy), por la facilidad para el usuario de interpretar el resultado del modelo, sin embargo, cuando se analizan datos de monitoreo de condiciones para la predicción de fallas en mantenimiento, la métrica de exactitud no es la más adecuada, ya que no representa significativamente el impacto de la clase minoritaria, que para este tipo de aplicación es la clase más interesante. En la Figura **3-1.a** se muestra la distribución de clases en el conjunto de datos de prueba (test) del caso de estudio 1, se observa que presenta un desequilibrio aproximado de clase con un ratio de 1:38, es decir, que por cada observación de la clase minoritaria (clase 1), tendrá 38 observaciones correspondientes para la clase mayoritaria (clase 0). En la Figura **3-1.b** se muestra la distribución desequilibrada de clases en el conjunto de datos de entrenamiento (Train), la clase mayoritaria representa el estado normal de funcionamiento de la máquina y la clase minoritaria representa un estado de falla potencial.



(a) Base de prueba (b) Base de entrenamiento

Figura 3-1.: Distribución de clases en los conjuntos de datos del caso de estudio 1

A continuación se evalúan un grupo de algoritmos de Machine Learning clásicos, con el propósito de analizar los resultados de la métrica de exactitud con los conjuntos de entrenamiento y prueba desequilibrados del caso de estudio 1, los modelos de clasificación binaria se entrenan utilizando las configuraciones predeterminadas, es decir, para este caso no se ajustan hiperparámetros para obtener las configuraciones apropiadas. En la tabla **3-2** se observa que los modelos evaluados alcanzan niveles de exactitud entre 97 % y 99 %, sin embargo, al analizar las medidas de desempeño por cada clase se observa que los modelos de ML clasifican con buenos resultados la clase mayoritaria (0-estado funcional de la máquina), pero los resultados en términos de precisión de la clase minoritaria (1-falla potencial) oscilan

entre el 49 % y 85 %, dicho de otro modo, a pesar de los altos resultados en términos de exactitud, algunos modelos tales como K-Nearest Neighbors y Decision Tree, arrojan niveles inaceptables de precisión de la clase minoritaria registrando valores por debajo de 70 %.

Tabla 3-2.: Exactitud vs. Precisión por clase

Algoritmo	Exactitud	Precisión clase 0	Precisión clase 1
Extra Trees	0.987	0.990	0.851
Random Forest	0.987	0.990	0.823
Gradient Boosting	0.987	0.991	0.812
Bagging	0.986	0.989	0.801
Support Vector Machine	0.984	0.988	0.772
Logistic Regression	0.983	0.987	0.746
k-Nearest Neighbors	0.981	0.988	0.658
Decision tree	0.976	0.988	0.535
Naive Bayes	0.974	0.997	0.494

Estos resultados en términos de precisión conducen a buscar métricas alternativas a la exactitud para comparar el desempeño de los modelos, los cuales se evalúan en problemas de clasificación con datos desequilibrados que se obtienen por técnicas de monitoreo de condiciones.

3.3.1. Aprendizaje sensible al costo en mantenimiento

De acuerdo con Brownlee (2020*a*), la mayoría de las métricas y costos asociados a los errores de clasificación, se pueden entender mejor en términos de una matriz de confusión, esta técnica nos ayuda a determinar cuántas observaciones se clasificaron correcta o incorrectamente comparando los valores observados y los pronosticados. La matriz de confusión no solo proporciona información sobre el rendimiento de un modelo predictivo sino también sobre qué clases se predicen adecuadamente, la figura **3-2** presenta la estructura de una matriz de confusión para un modelo de clasificación binaria aplicado en mantenimiento, para este caso la clase negativa típicamente asignada con la etiqueta de clase 0 corresponde al estado funcional del activo físico o componente, y la clase positiva asignada con la etiqueta de clase 1 corresponde al estado de falla funcional o potencial, las columnas de la matriz de confusión representan la clase predicha a la que pertenecen las observaciones, y las filas representan la clase real, con respecto al orden de la matriz de confusión Brownlee (2020*a*), afirma: *"el significado de las filas y columnas puede intercambiarse y, a menudo, se intercambian sin pérdida de significado"* (p. 182).

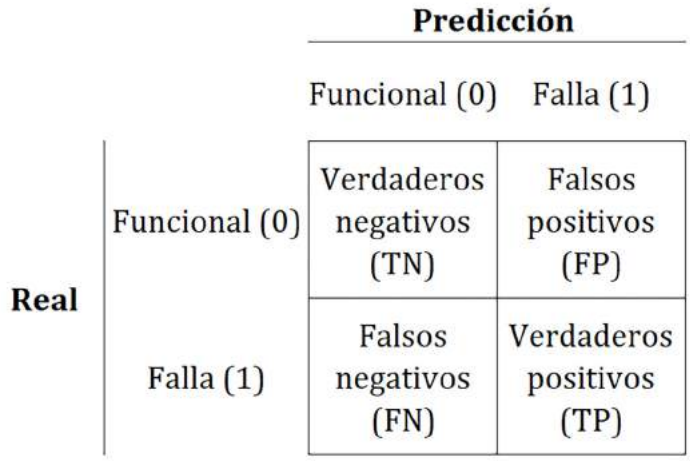


Figura 3-2.: Matriz de confusión para un modelo de clasificación binaria

El aprendizaje sensible al costo es un tipo de aprendizaje que toma en consideración los costos de clasificación errónea, en otros términos, en la práctica de la ingeniería de mantenimiento es más costoso etiquetar erróneamente observaciones clasificadas como estado normal de la máquina, cuando realmente son fallas potenciales o funcionales (falsos negativos), que etiquetar erróneamente observaciones clasificadas como falla de la máquina cuando realmente son observaciones de estado normal o funcional del activo físico (falsos positivos), los falsos negativos implican posibles pérdidas de producción por paradas no programadas de los activos y los falsos positivos conllevan a programar inspecciones proactivas o ejecutar mantenimientos preventivos que posiblemente no eran necesarios.

Dada la estrecha relación entre la clasificación desequilibrada y el aprendizaje sensible al costo, se deben definir métricas de desempeño con la finalidad de comparar los modelos de clasificación que capturen la clase minoritaria, pero que a su vez minimice los costos de las clasificaciones erróneas del modelo, dado que existen múltiples métricas y técnicas para medir el desempeño de los modelos, en este estudio nos enfocamos en las métricas que permiten comparar los modelos teniendo en cuenta los costos de clasificación errónea y la necesidad de identificar de forma óptima la clase minoritaria.

Con el propósito de cuantificar los errores de predicción de clasificación se describen cuatro medidas de desempeño, la cuales son útiles para los modelos de clasificación con datos desequilibrados, esto debido a que se centran específicamente en el rendimiento de una clase, la métricas de desempeño que se analizan en los experimentos de clasificación son precisión, Recall, ROC AUC y F-β, las cuales se describen a continuación:

- **Precisión:** Es la proporción de verdaderos positivos entre todas las observaciones que el modelo predice, tal como se muestra en la ecuación 3-1. Aunque la precisión es útil, no incluye cuántas observaciones de clases positivas reales se predijeron como pertenecientes a la clase negativa.

$$Precision = \frac{numero\ de\ verdaderos\ positivos}{(numero\ de\ verdaderos\ positivos + numero\ de\ falsos\ positivos)} \quad (3\text{-}1)$$

- **Recall:** Mide el número de predicciones positivas que acierta el modelo de todas las positivas correctas que podrían haberse acertado, tal como se muestra en la ecuación 3-2. Para la predicción de fallas en mantenimiento, el Recall o sensibilidad es muy útil ya que mide efectivamente la cobertura de la clase minoritaria (fallas).

$$Recall = \frac{numero\ de\ verdaderos\ positivos}{(numero\ de\ verdaderos\ positivos + numero\ de\ falsos\ negativos)} \quad (3\text{-}2)$$

- **ROC AUC:** Según Brownlee (2020*a*), una curva ROC es un gráfico de diagnóstico que resume el comportamiento de un modelo, el área ROC bajo la curva (ROC AUC, por sus siglas en inglés) asigna una puntuación única para un modelo de clasificación binaria, el rango de puntaje ROC AUC es un valor que oscila entre 0.0 y 1.0, donde valores iguales o menores a 0.5 señalan que el modelo no cuenta con la suficiente habilidad para clasificar y el valor de 1.0 indica un clasificador perfecto.

- **Medida F-β:** Se define como la media armónica ponderada de la precisión y el Recall tal como se muestra en la ecuación 3-3, según Brownlee (2020*a*), es la métrica que más se utiliza con el propósito de medir el desempeño de modelos en problemas de clasificación desequilibrada.

$$F\beta = \frac{(1+\beta^2) * Precision * Recall}{\beta^2 * Precision + Recall}, \quad (3\text{-}3)$$

  Donde β es el coeficiente que controla el peso o el balance entre la precisión y el Recall, los valores comunes asignados a β son los siguientes:

  * Medida F0.5 (β=0.5): En esta medida se le asigna más peso a la precisión y menos peso a el Recall, con este coeficiente se parte del supuesto que los falsos positivos son más importantes, para el caso de predicción de fallas en mantenimiento, se utiliza esta métrica cuando los costos de ejecutar múltiples mantenimientos preventivos o inspecciones son de mayor impacto que las clasificaciones erróneas que

puedan generar un paro funcional de la máquina sin previo aviso.

* Medida F1 (β=1): Equilibra el peso entre precisión y el Recall, es decir en esta medida los falsos negativos y falsos positivos son igualmente importantes. En mantenimiento esta medida significa un balance entre costos generados por inspecciones o mantenimientos preventivos y posibles fallas funcionales que se presenten.

* Medida F2 (β=2): En esta medida se le asigna más peso al Recall y menos peso a la precisión, con este coeficiente se parte del supuesto que los falsos negativos son más importantes, para el caso de predicción de fallas en mantenimiento, se utiliza esta métrica cuando los costos en que se incurren por un paro funcional de la máquina, son mucho mayores que asumir costos por inspecciones o mantenimientos preventivos cuando se generan falsas alarmas.

Estas métricas se calculan para cada clase y con el promedio de los puntajes se obtiene la medida "Macro", la cual corresponde a la media aritmética de las medidas de desempeño de las clases que se evalúan, dicho de otra forma, para valorar la medida macro-F1, primero se mide la precisión y el Recall por cada clase, luego se calcula los puntajes F1 por clase y finalmente se obtiene la media aritmética de estos puntajes, tal como se muestra en la ecuación 3-4.

$$F1\ Score\ Macro = \frac{\sum_{i=0}^{n_c} F1\ Clase_i}{n_c}, \tag{3-4}$$

Donde n_c es el numero de clases, $F1\ Clase_i$ corresponde a los valores de $F1$ por cada clase.

Por interpretabilidad, las medidas de desempeño para modelos de detección temprana de fallas se desarrollan con problemas de clasificación binaria, pero también aplican cuando se implementan en problemas de clasificación multiclase; un caso típico de clasificación multiclase en mantenimiento se da cuando se asignan etiquetas a las observaciones en rangos de tiempo antes que ocurra la falla funcional, con el propósito que el modelo genere alarmas que adviertan al operador o mantenedor del sistema, por ejemplo, a las observaciones de la serie de tiempo se le asigna la etiqueta de clase 2 (falla funcional) 5 días antes que ocurra la salida de operación del activo, a las observaciones en el rango entre 5 y 30 días antes de la salida de operación del activo se le asigna la etiqueta de clase 1 (falla potencial) y las demás observaciones se etiquetan con clase 0 que corresponde al estado normal o funcional.

Otro tipo de problema de clasificación multiclase para predicción de fallas se presenta cuando se cuenta con la información histórica de la parte o componente que causa la falla en el activo, para este caso clasificamos las observaciones que corresponden a la falla del componente 1

con la etiqueta de clase 1, las observaciones que corresponden a la falla del componente 2 con la etiqueta de clase 2 y así sucesivamente hasta finalizar con la codificación de fallas, en este caso para las observaciones en estado normal o funcional se le asigna la etiqueta de clase 0.

3.3.2. Validación cruzada estratificada

La evaluación de un modelo implica probar distintas configuraciones en la preparación de los datos, diferentes algoritmos y ajustar los hiperparámetros para mejorar el rendimiento. En este trabajo la técnica utilizada para la evaluación y ajuste de los modelos de ML es la validación cruzada (k-Fold Cross-Validation), este procedimiento conlleva a dividir el conjunto de datos en pliegues con el fin de producir una estimación mas confiable del rendimiento del modelo; un complemento a esta técnica es la validación cruzada estratificada (stratified cross-validation) la cual es muy útil cuando se analizan datos desequilibrados, con la estratificación se divide el conjunto de datos preservando la distribución de la misma clase en cada pliegue, en consecuencia, la división de los datos coincide con la distribución en el conjunto de datos de entrenamiento completo, mitigando posibles variaciones significativas en la estimación del rendimiento del modelo.

3.4. Comparativo del desempeño de los modelos de ML y DNN

Esta sección tiene como propósito ajustar, evaluar y comparar los modelos de Machine Learning y redes neuronales profundas en los tres casos de estudio, el interés consiste en encontrar los modelos con los mejores rendimientos utilizando técnicas de configuración de datos, configurar los hiperparámetros y explorar diferentes arquitecturas de redes.

3.4.1. Comparativo del desempeño para el caso de estudio 1

Con el propósito de mejorar el rendimiento de los modelos de ML y así permitir que sean directamente comparables con el desempeño de las redes neuronales profundas, se aplican diferentes técnicas las cuales se describen a continuación:

- **Ajuste de hiperparámetros:** El ajuste de hiperparámetros es un enfoque cuyo objetivo es explorar y evaluar la configuración idónea de un modelo, en este caso se realiza una búsqueda automática en cuadrícula con validación cruzada K-fold estratificada para evaluar metódicamente la combinación de hiperparámetros de los algoritmos de ML.

- **Sobremuestreo:** Los conjuntos de datos de entrenamiento desequilibrados pueden afectar el desempeño de muchos algoritmos de ML. Tal como lo presenta Branco et al. (2015), el muestreo de datos es una estrategia efectiva con el fin de equilibrar la distribución de clases, una de las técnicas es el sobremuestreo aleatorio que implica duplicar las observaciones de la clase minoritaria hasta equilibrar la base de entrenamiento, otra técnica es el sobremuestreo de minorías sintéticas (SMOTE) que utiliza la interpolación para sobremuestrear la clase minoritaria generando nuevos datos sintéticos como lo muestra el trabajo de Chawla et al. (2002).

- **Escalado de datos:** Según Brownlee (2017), los datos para problemas de predicción de secuencia probablemente necesiten escalarse para entrenar el algoritmo ya que es posible que las diferentes escalas afecten el tiempo de aprendizaje y el desempeño del modelo. Es factible aplicar dos tipos de escalado a las series de tiempo, la normalización, que es una técnica que escala los datos del rango original para que todos los valores estén dentro del rango de 0 y 1 y la estandarización de un conjunto de datos que implica cambiar la distribución de valores de cada variable para que la media de las observaciones sea 0 con una desviación estándar de 1.

Después de realizar diferentes experimentos de modelado con algoritmos de clasificación binaria, para los cuales se combinan técnicas de búsqueda en cuadricula de hiperparámetros con validación cruzada, sobremuestreo y escalado de datos; comparando el desempeño de los modelos en términos de F1 Macro, precisión y Recall de la clase minoritaria, se presenta en la tabla **3-3** los mejores resultados obtenidos de cada algoritmo con sus métricas de desempeño para el caso de estudio 1.

Tabla 3-3.: Comparativo del desempeño entre modelos ML y DNN: caso de estudio 1

Tipo	Algoritmo / Modelo	F1 -Score Macro	Recall Macro	Precisión Macro	Recall clase 1	Recall clase 0	Precisión clase 1	Precisión clase 0	F1 clase 1	F1 clase 0	ROC AUC	Exactitud
DNN	Vanilla LSTM	0.91	0.88	0.95	0.76	1.00	0.90	0.99	0.83	1.00	0.88	0.99
DNN	CNN	0.90	0.88	0.92	0.77	1.00	0.85	0.99	0.81	1.00	0.88	0.99
Conjunto	XGBoost	0.85	0.89	0.82	0.78	0.99	0.64	0.99	0.70	0.99	0.89	0.99
Conjunto	Extra Trees	0.85	0.86	0.84	0.74	0.99	0.68	0.99	0.71	0.99	0.86	0.98
Conjunto	Random Forest	0.85	0.87	0.82	0.76	0.99	0.65	0.99	0.70	0.99	0.87	0.98
Conjunto	Gradient Boosting	0.85	0.84	0.86	0.70	0.99	0.73	0.99	0.71	0.99	0.84	0.99
Conjunto	Bagging	0.84	0.86	0.83	0.72	0.99	0.67	0.99	0.69	0.99	0.86	0.98
Lineal	Logistic Regression	0.82	0.88	0.77	0,79	0.98	0.55	0.99	0.65	0.99	0.88	0.98
Lineal	Naive Bayes	0.81	0.93	0.75	0.88	0.98	0.49	0.99	0.63	0.99	0.93	0.97
No lineal	Decision tree	0.80	0.83	0.77	0.68	0.99	0.55	0.99	0.61	0.99	0.83	0.98
No lineal	Support Vector Machine	0.80	0.88	0.74	0.78	0.98	0.49	0.99	0.61	0.98	0.88	0.97
DNN	MLP	0.80	0.81	0.80	0.62	0.99	0.61	0.99	0.62	0.99	0.81	0.98
No lineal	k-Nearest Neighbors	0.79	0.75	0.87	0.50	1.00	0.74	0.99	0.60	0.99	0.75	0.98

Partiendo que la métrica F1-Score macro es la medida de desempeño seleccionada con el propósito de comparar los modelos evaluados en este estudio, la cual busca minimizar los

falsos negativos, sin olvidarse de los costos asociados a los falsos positivos, se observa en la tabla **3-3** que los mejores resultados se obtienen con las redes neuronales profundas (DNN), particularmente con los modelos LSTM y CNN, con los cuales se obtiene un F1-Score macro de 91 % y 90 % respectivamente, el modelo Vanilla LSTM obtiene una precisión macro de 95 %, la cual supera a los modelos de ML clásicos.

En el segundo grupo, organizados en términos de desempeño se obtienen los algoritmos de conjunto, para Smolyakov (2017), los métodos de conjunto ayudan a mejorar el rendimiento de los algoritmos mediante la combinación de varios modelos; en este sentido, se observa en la tabla **3-3** que el algoritmo de conjunto XGBoost (Extreme Gradient Boosting) con ajuste avanzado de hiperparámetros obtiene un F1-Score macro de 85 %, el Recall de la clase 1 es mayor en dos puntos porcentuales a la red Vanilla LSTM, pero la precisión de la clase 1 con un valor de 64 % en el algoritmo XGBoost indica que el número de falsos positivos es mucho mayor que los resultados obtenidos con las redes DNN. Por último, tenemos los algoritmos de ML lineales y no lineales que presentan valores de F1-Score macro menores o iguales a 82 %.

Los dos modelos de mejor rendimiento del caso de estudio 1 tienen la siguiente arquitectura:

> **Vanilla LSTM:** Se genera una instancia secuencial con una única capa oculta LSTM con 25 unidades de memoria, la capa de salida es una capa MLP completamente conectada (densa) con una única neurona, se utiliza una función de activación logística "Sigmoid" en la capa de salida con el fin de permitir que la red "aprenda", el algoritmo se compila para minimizar la pérdida de registro con "binary crossentropy" y la implementación del algoritmo de descenso de gradiente "Adam". En la figura **3-3** se presenta la arquitectura del modelo.

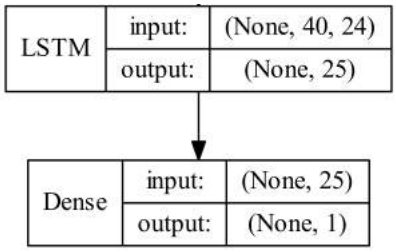


Figura 3-3.: Arquitectura modelo Vanilla LSTM

> **CNN:** Se genera una instancia secuencial con dos capas ocultas convolucionales que operan sobre secuencias unidimensionales con 64 filtros de salida por capa, seguidas de una capa de regularización del 20 % (dropout) que se conecta a una capa (MaxPooling)

que agrupa la salida de la parte convolucional, la estructura finaliza con dos capas densas, la primera con 100 neuronas y la capa de salida con 1 neurona, cuya función es interpretar las características extraídas por la parte convolucional del modelo. En la figura **3-4** se presenta la arquitectura del modelo.

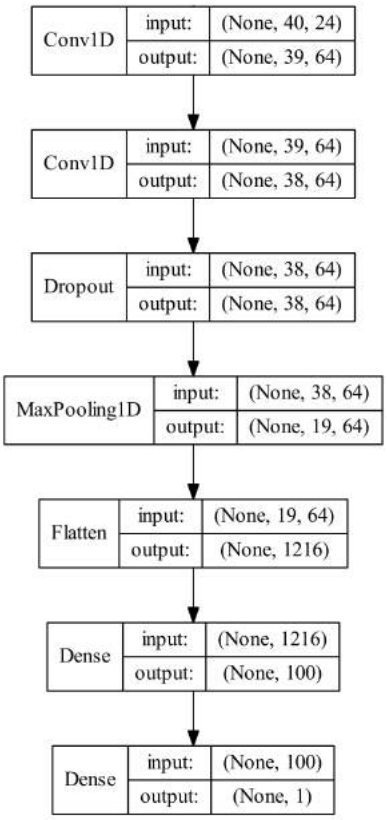


Figura 3-4.: Arquitectura modelo CNN

3.4.2. Comparativo del desempeño para el caso de estudio 2

Este caso de estudio corresponde a la aplicación de modelos de clasificación multiclase, donde la clase 0 está asociada al estado normal y las clases 1 a la 4 están asociadas a las fallas funcionales de diferentes componentes del sistema. Con el fin de evaluar y comparar los modelos de ML y DNN, en este experimento el desempeño de los algoritmos se calcula manteniendo la base de entrenamiento severamente desequilibrada, en otros términos, no se aplicaron técnicas de sobremuestreo en el ajuste de los modelos de ML y DNN, en este caso en particular la distribución está severamente sesgada para todas las clases que indican falla, por ejemplo, por cada observación de falla del componente 1 (clase1) tenemos más de 5000 observaciones del estado normal del sistema (clase 0).

Con el propósito de explorar y evaluar la configuración idónea de los algoritmos de ML se realiza una búsqueda automática en cuadrícula con validación cruzada K-fold estratificada donde se evalúa metódicamente la combinación de hiperparámetros de los algoritmos de ML incluyendo la configuración de pesos de clase del algoritmo.

Como se puede observar en la tabla **3-4**, a pesar del desequilibrio severo que presentan los datos de entrenamiento y prueba, los modelos de DNN producen un resultado satisfactorio y no se afectan significativamente por esta condición de desequilibrio de clases, el modelo Stacked LSTM (apilado de varias capas LSTM) proporciona un resultado con un F1-macro del 93 %, el cual supera por 17 puntos porcentuales el desempeño de los modelos de ML clásicos con ajuste de hiperparámetros.

Tabla 3-4.: Comparativo del desempeño entre modelos ML y DNN: caso de estudio 2

Tipo	Algoritmo / Modelo	F1 -Score Macro	Recall Macro	Precisión Macro	F1 clase 0	F1 clase 1	F1 clase 2	F1 clase 3	F1 clase 4	Exactitud
DNN	Stacked LSTM	0.93	0.92	0.94	1.00	0.86	0.98	0.88	0.92	1.00
DNN	Vanilla LSTM	0.88	0.84	0.94	1.00	0.85	0.91	0.80	0.85	1.00
Lineal	Logistic Regression	0.76	0.77	0.75	1.00	0.52	0.67	0.85	0.78	1.00
Conjunto	XGBoost	0.75	0.71	0.80	1.00	0.39	0.68	0.86	0.82	1.00
Conjunto	Extra Trees	0.72	0.74	0.74	1.00	0.46	0.66	0.73	0.74	1.00
Conjunto	Random Forest	0.72	0.66	0.81	1.00	0.44	0.56	0.88	0.71	1.00
No lineal	Support Vector Machine	0.71	0.65	0.80	1.00	0.35	0.63	0.80	0.78	1.00
No lineal	Decision tree	0.70	0.73	0.70	1.00	0.40	0.63	0.71	0.74	1.00
Conjunto	Bagging	0.70	0.64	0.79	1.00	0.41	0.64	0.71	0.77	1.00

Esta diferencia en términos de desempeño de 17 puntos porcentuales entre los modelos DNN y ML, se explica por el nivel de sesgo de las clases minoritarias debido al desequilibrio severo de los datos de entrenamiento; en este sentido Krawczyk (2016), afirma que los algoritmos de ML parten del supuesto que la distribución de clases es similar, pero en la vida real aplicaciones, como por ejemplo, la detección de fraudes y sistemas de monitoreo industrial, presentan este tipo de distribución segada y generalmente la clase minoritaria es la más importante desde la perspectiva de la minería de datos, esta situación de desequilibrio da como resultado modelos que tienen un rendimiento predictivo bajo, especialmente para la clase de mayor importancia.

La limitación por el desequilibrio de clases con los algoritmos de ML (conjunto, lineales y no lineales), se puede observar en la tabla **3-4**, este grupo de algoritmos obtienen un F1 de la clase mayoritaria (estado normal del sistema) aproximadamente del 100 %, pero cuando se observa el desempeño de la clasificación para la detección de fallas para el componente 1, el F1 de la clase 1 oscila entre 35 % y 52 % y para el componente 2 el F1 de la clase 2 oscila entre 56 % y 67 %, este nivel de error que expone la métrica F1 por clase de los algoritmos de ML es inaceptable en entornos productivos reales. Caso contrario sucede con los modelos DNN (Vanilla LSTM y Stacked LSTM) cuyos valores de F1 de las clases minoritarias que indican las fallas de componentes oscilan entre el 80 % y el 98 %.

El modelo LSTM apilado (Stacked) con el que se logró el mejor rendimiento del caso de estudio 2 tiene la siguiente arquitectura:

> **Stacked LSTM:** Se genera una instancia secuencial con dos capas ocultas LSTM con 100 y 50 unidades de memoria respectivamente, la capa de salida está completamente conectada (densa) con 5 neuronas (una neurona por valor de clase), se utiliza una función de activación "Softmax" para clasificación multiclase, el modelo se compila con el fin de minimizar la pérdida de registro con "categorical–crossentropy" implementando el algoritmo de descenso de gradiente "Adam". Para evitar el sobreajuste se incluye como método de regularización dos capas de abandono del 20 % para omitir aleatoriamente este porcentaje de neuronas en las capas LSTM. En la figura **3-5** se presenta la arquitectura del modelo Stacked LSTM.

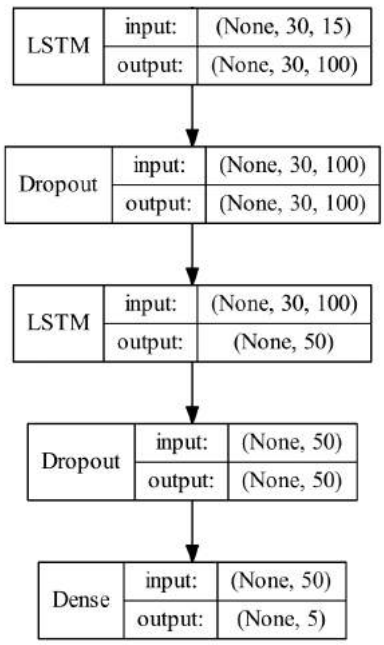


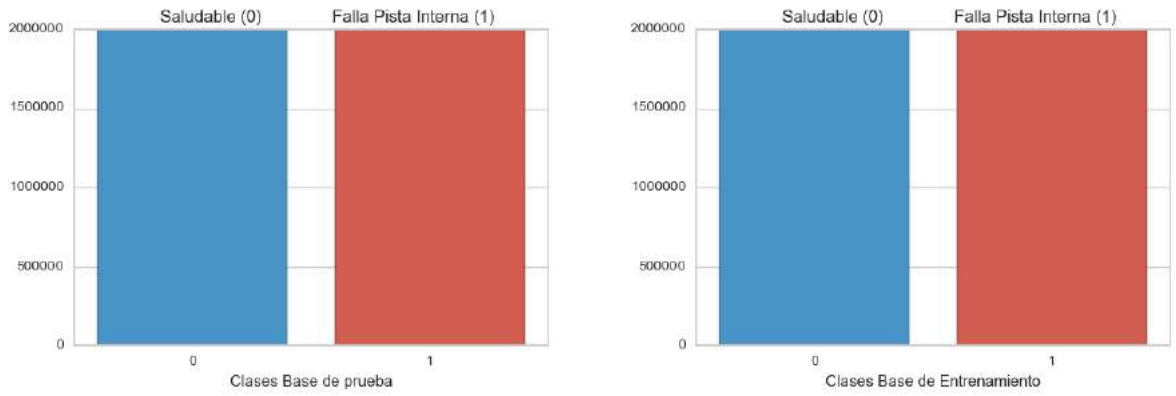
Figura 3-5.: Arquitectura modelo Stacked LSTM

3.4.3. Comparativo del desempeño para el caso de estudio 3

Este caso de estudio se ajusta a la aplicación de modelos de clasificación binaria para análisis de vibraciones en rodamientos, donde la clase 0 corresponde a los datos tomados de velocidad y aceleración de un rodamiento en estado saludable y la clase 1 corresponde en este estudio a los datos tomados de un rodamiento con un defecto en la pinta interna; en este experimento

se concatenan los datos de un rodamiento en estado saludable y un rodamiento con falla en pista interna con el fin de conformar el conjunto de entrenamiento, de igual manera se selecciona un conjunto de datos de un rodamiento en estado saludable y uno con falla en pista interna (diferentes a los utilizados en el conjunto de entrenamiento) para conformar el conjunto de prueba, con el propósito de explorar y evaluar la configuración idónea de los algoritmos de ML se realiza una búsqueda automática en cuadrícula con validación cruzada K-fold estratificada donde se evalúa metódicamente la combinación de hiperparámetros de los algoritmos de ML.

Una de las características del caso de estudio 3 es que los conjuntos de datos están equilibrados, dicho de otra manera, el número de observaciones del rodamiento en estado normal y en falla es igual tanto en la base de entrenamiento como en la base de prueba, tal como se observa la figura **3-6.a** donde se evidencia el equilibrio entre clases de la base de prueba y la figura **3-6.b** el equilibrio de clases de la base de entrenamiento.



(a) Base de prueba

(b) Base de entrenamiento

Figura 3-6.: Distribución de clases en los conjuntos de datos del caso de estudio 3

Para este caso los modelos de ML se entrenan con los hiperparametros ajustados y se comparan en términos de desempeño con arquitecturas estándar LSTM (Vanilla y Stacked), como se muestra en la tabla **3-5** los algoritmos de conjunto Extreme Gradient Boosting y Gradient Boosting entregan el mayor F1-macro con un 95 %, seguido de cerca por las redes neuronales profundas con arquitectura LSTM, con un resultado en términos de F1-macro del 94 %, lo cual indica un buen rendimiento de las redes de memoria a corto y largo plazo con solo un punto porcentual por debajo de los algoritmos basados en árboles de decisión impulsados por gradiente. En todos los modelos el Recall de la clase 1 (falla) es mayor que la clase 0 (normal), lo que indica que el número de falsos negativos es considerablemente menor al número de falsos positivos.

Tabla 3-5.: Comparativo del desempeño entre modelos ML y DNN: caso de estudio 3

Tipo	Algoritmo / Modelo	F1 -Score Macro	Recall Macro	Precisión Macro	Recall clase 1	Recall clase 0	Precisión clase 1	Precisión clase 0	F1 clase 1	F1 clase 0	ROC AUC	Exactitud
Conjunto	XGBoost	0.95	0.95	0.96	1.00	0.91	0.92	1.00	0.96	0.95	0.95	0.95
Conjunto	Gradient Boosting	0.95	0.95	0.96	1.00	0.91	0.92	1.00	0.96	0.95	0.95	0.95
DNN	Vanilla LSTM	0.94	0.94	0.95	1.00	0.88	0.89	1.00	0.94	0.93	0.94	0.94
DNN	Staked LSTM	0.92	0.92	0.93	1.00	0.85	0.87	1.00	0.93	0.92	0.92	0.92
Conjunto	Extra Trees	0.92	0.92	0.93	1.00	0.84	0.86	1.00	0.93	0.91	0.92	0.92
No lineal	Decision tree	0.92	0.92	0.93	1.00	0.84	0.86	1.00	0.93	0.91	0.92	0.92
Conjunto	Bagging	0.91	0.92	0.93	1.00	0.83	0.86	1.00	0.92	0.91	0.92	0.92
Conjunto	Random Forest	0.91	0.92	0.93	1.00	0.83	0.86	1.00	0.92	0.91	0.92	0.92
No lineal	k-Nearest Neighbors	0.90	0.90	0.92	1.00	0.81	0.84	1.00	0.91	0.89	0.90	0.90
Lineal	Logistic Regression	0.77	0.78	0.84	0.56	1.00	0.69	0.99	0.82	0.71	0.78	0.78

En este caso se observa que el desempeño de los modelos de conjunto de ML es comparable con los modelos de redes neuronales profundas LSTM, esto se debe a que los algoritmos de ML se entrenan en un conjunto de datos equilibrado con el adecuado ajuste de hiperparámetros, lo que ayuda a mejorar el rendimiento predictivo de los modelos.

Partiendo del hecho que en los tres casos de estudio que se analizan en este trabajo las redes neuronales profundas con configuraciones estándar exponen un buen desempeño para la clasificación de fallas en datos obtenidos por monitoreo de condiciones, en consecuencia este resultado conduce a desarrollar de forma robusta el ajuste avanzado de la arquitectura LSTM y los modelos híbridos DNN con el objetivo de mejorar el desempeño de los modelos predictivos aplicados en la ingeniería de mantenimiento.

3.5. Diagnóstico y ajuste de redes neuronales LSTM

Debido a la característica estocástica de las redes neuronales profundas, cada vez que se entrena el modelo con los mismos datos los resultados en términos de desempeño y predicciones varían, en términos de Brownlee (2017): *"esta aleatoriedad adicional le da al modelo más flexibilidad cuando aprende, pero puede hacer que el modelo sea menos estable"* (p. 164). Por esta razón, una buena práctica para ajustar la estructura de la red es repetir el modelo con los mismos datos de entrenamiento varias veces y medir la variación de la precisión o perdida del modelo. Como ejercicio en esta parte del trabajo se documentan los resultados de ajuste avanzado, diagnóstico y configuración de la arquitectura DNN para el caso de estudio 1.

3.5.1. Ajuste de la arquitectura para la red LSTM

Partiendo de la flexibilidad en la configuración de la arquitectura LSTM se realiza una exploración de diferentes alternativas con el propósito de mejorar el desempeño del modelo:

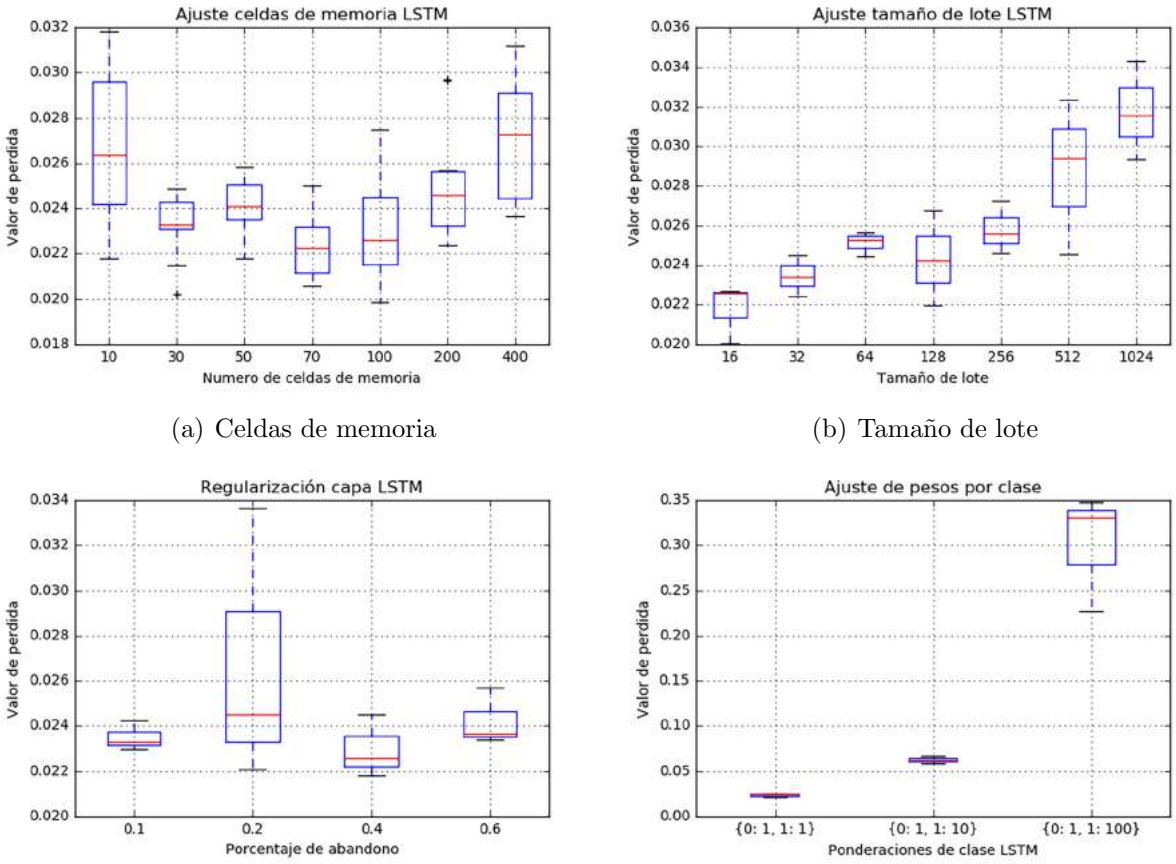
- **Celdas de memoria:** Para hallar el número de celdas de memoria de las capas LSTM se ejecuta el algoritmo 10 veces con 7 configuraciones definidas entre 10 y 400 celdas de memoria, en la figura **3-7.a** se presenta la gráfica de cajas y bigotes con los resultados finales para comparar el desempeño del modelo para cada una de las diferentes configuraciones, teniendo en cuenta que se busca minimizar la pérdida del modelo, en la figura **3-7.a** se observa que la estructura de la capa oculta LSTM con 70 celdas de memoria, registra la menor perdida media con 2.23 % y la configuración con 50 celdas de memoria indica la menor desviación estándar con 0.14 %, considerando que, el objetivo es seleccionar una configuración con un nivel de perdida bajo (mayor precisión) y la menor varianza posible (mayor estabilidad), para este caso es viable seleccionar configuraciones entre 30, 50 y 70 celdas de memoria para la capa LSTM de la red neuronal profunda.

- **Tamaño del lote:** El tamaño de lote es el número de muestras de la base de entrenamiento que se analizan antes que se actualicen los parámetros de la red, en vista que, el valor seleccionado afecta la eficiencia y velocidad de aprendizaje se exploran diferentes configuraciones, en la figura **3-7.b** se presenta la gráfica de cajas y bigotes con los resultados finales con el propósito de medir el desempeño y variación del modelo, el tamaño de lote con valor de 16 proporciona un resultado que conlleva a la menor perdida media de la red con 2.18 % y una desviación estándar de 0.15 %. Con el propósito de encontrar un balance entre velocidad de aprendizaje y menor perdida, se selecciona como tamaño de lote óptimo para el modelo el valor de 32 con una perdida media de 2.34 % y una desviación estándar de 0.1 %.

- **Regularización:** Para minimizar el sobreajuste de la red neuronal es posible incluir capas de abandono (Dropout) en la arquitectura de la red, tal como lo muestra el trabajo de Hinton et al. (2012), al incluir estas capas se omiten aleatoriamente las neuronas en la etapa de entrenamiento, lo que mejora a nivel general el desempeño de los modelos. En la figura **3-7.c** se presenta la gráfica de cajas y bigotes con los resultados finales que incluyen las mediciones de desempeño y variación, después de repetir el entrenamiento de la red incluyendo una capa de regularización, donde se evalúan diferentes porcentajes de abandono, en este caso la capa de abandono con valor de 40 % proporciona un resultado que implica la menor perdida media con 2.30 % y una desviación estándar de 0.14 %.

- **Ajuste de peso:** Partiendo del desequilibrio entre clases de la base de entrenamiento es posible asignar diferentes pesos a cada una de las clases en la red neuronal profunda, tal como lo describe Brownlee (2020*a*): *"esta modificación del algoritmo de*

entrenamiento de la red neuronal se conoce como red neuronal ponderada o red neuronal sensible al costo" (p. 229). En la figura **3-7.d** se presenta la gráfica de cajas y bigotes con las diferentes ponderaciones de clase, que se asignan para penalizar los errores de clasificación de la clase minoritaria, cuando se establece un peso de 100 a la clase 1 el error de precisión aumenta, ya que se genera un aumento de los falsos positivos, por lo tanto, el rango de valor de configuración de peso para la clase minoritaria puede oscilar entre 1 y 10 para una precisión equilibrada entre las dos clases del modelo.



(a) Celdas de memoria

(b) Tamaño de lote

(c) Porcentaje de abandono

(d) Pesos por clase

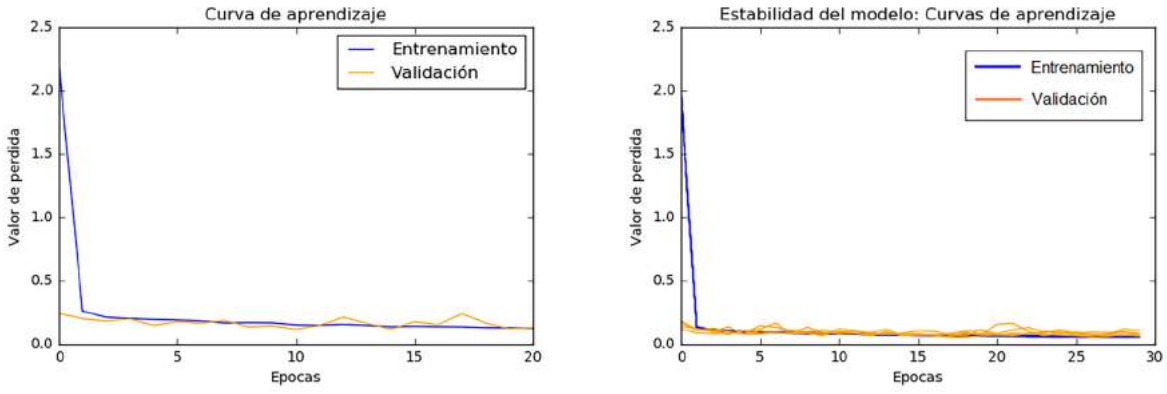
Figura 3-7.: Ajuste arquitectura red neuronal LSTM

3.5.2. Diagnóstico del comportamiento del modelo

El número de capas ocultas que se adicionan a la estructura de la red tienen un impacto significativo en el desempeño y tiempo de entrenamiento del modelo, por consiguiente, una buena practica es evaluar el historial de entrenamiento de la red neuronal profunda, para lo

cual, se debe trazar la curva de perdida de los datos de entrenamiento frente a la pérdida de los datos de validación durante las épocas de entrenamiento de la DNN, esta gráfica se conoce como curva de aprendizaje y se usa para diagnosticar el ajuste (good fit), sobreajuste (overfit) o sub-ajuste (underfit) del modelo entrenado, según James et al. (2017): *"estos modelos más complejos pueden conducir a un fenómeno conocido como sobreajustar los datos, lo que esencialmente significa que siguen los errores o el ruido, demasiado de cerca"* (p. 22). Igualmente, con las curvas de aprendizaje se puede detectar la falta de ajuste del modelo tal como afirma Goodfellow et al. (2016): *"el ajuste insuficiente ocurre cuando el modelo no puede obtener un valor de error suficientemente bajo en el conjunto de entrenamiento"* (p. 111).

Después de analizar diferentes configuraciones de capas ocultas LSTM y ajustar la arquitectura con valores óptimos, que dan como resultado menor perdida media y menor variación del modelo LSTM apilado, se observa en la figura **3-8.a** la curva de aprendizaje del modelo Stacked LSTM, el resultado final es un ajuste adecuado, esto se puede diagnosticar ya que la pérdida del conjunto de entrenamiento y la pérdida del conjunto de validación disminuyen y se estabilizan alrededor del mismo punto. Debido a la naturaleza estocástica de los modelos DNN, en la figura **3-8.b** se muestran múltiples ejecuciones de diagnóstico para los mismos datos de entrenamiento y validación, con esta gráfica se deduce que el modelo tiene un buen ajuste, es estable y muestra un comportamiento sólido durante las épocas de entrenamiento.



(a) Curva de aprendizaje

(b) Estabilidad del modelo

Figura 3-8.: Curvas de aprendizaje

El modelo Stacked LSTM configurado con la estructura óptima del caso de estudio 1 tiene la siguiente arquitectura:

> **Stacked LSTM:** Se genera una instancia secuencial con dos capas ocultas LSTM con 70 y 30 unidades de memoria respectivamente, se adiciona en la capa LSTM una función de inicialización de peso normal; para evitar el sobreajuste se incluyen dos métodos de regularización, como primer método, se adicionan dos capas de abandono del 40 % y el segundo se incluye la regularización de peso en una de las capas ocultas LSTM. Para compensar el desequilibrio de clases en la arquitectura de la red neuronal profunda se incluye el hiperparámetro de ajuste de pesos de las clases, para lo cual se asignó un peso de 1 para la clase mayoritaria (clase 0 - estado normal) y un peso de 5 a la clase minoritaria (clase 1 - falla potencial), en este caso el mayor peso de clase se utiliza para asignar una ponderación mas grande a la clase minoritaria. La capa de salida está completamente conectada (densa) con 1 neurona, se utiliza una función de activación "sigmoid", el modelo se compila con el fin de minimizar la pérdida de registro con "binary–crossentropy" implementando el algoritmo de descenso de gradiente "Adam". En la figura **3-9** se presenta la arquitectura del modelo óptimo Stacked LSTM.

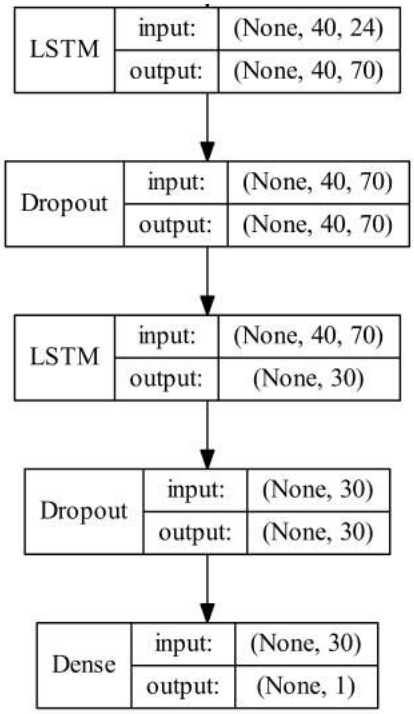


Figura 3-9.: Arquitectura modelo óptimo Stacked LSTM

Si se compara el desempeño del modelo apilado LSTM con ajustes óptimos, en relación con el modelo estándar de una capa Vanilla LSTM, se logra un aumento del ROC AUC en 6 puntos porcentuales y el Recall de la clase minoritaria aumenta de 76 % a 88 %, lo que conlleva a una disminución relevante de los falsos negativos al implementar el modelo de varias capas LSTM con ajuste óptimo de la arquitectura.

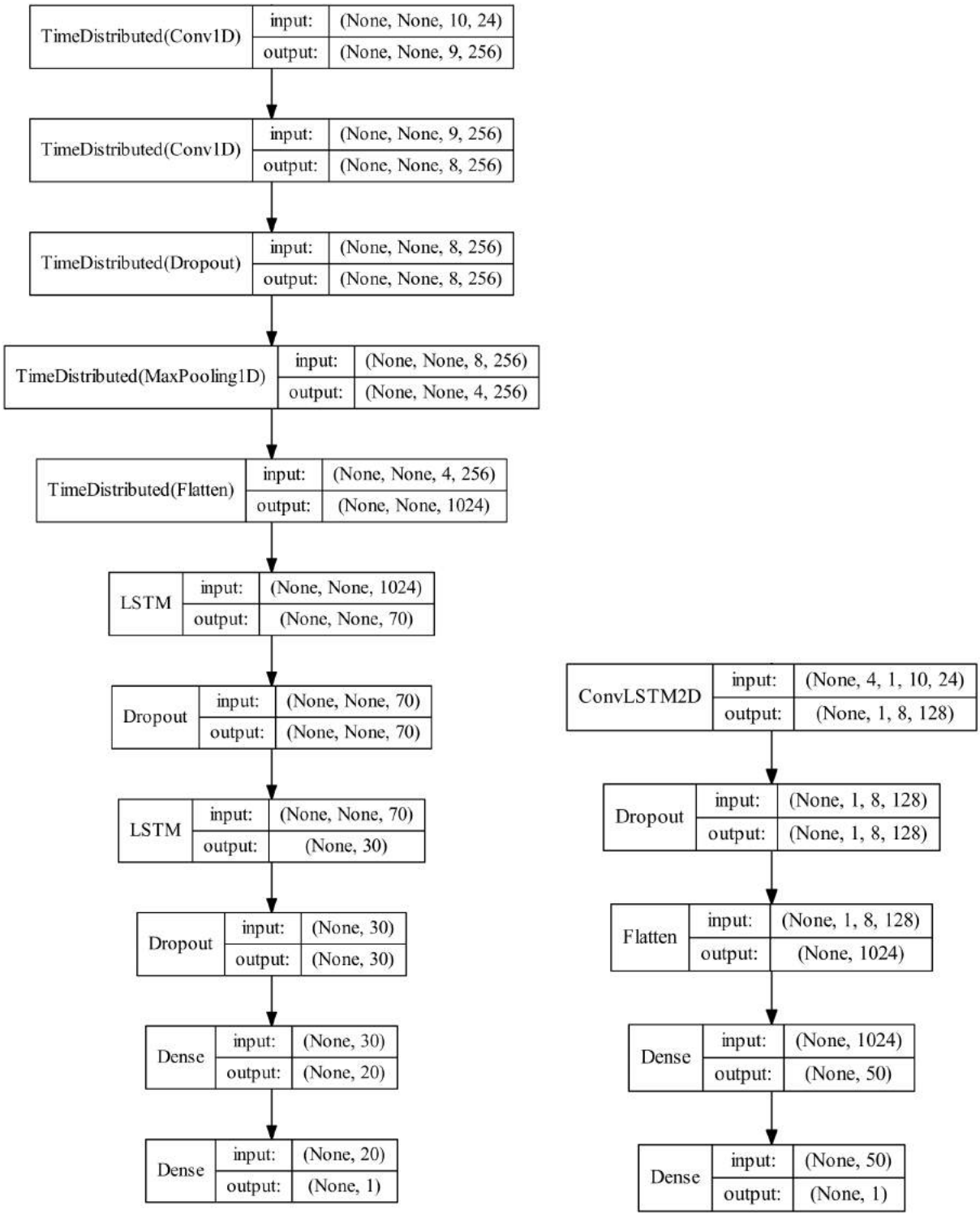
3.5.3. Arquitecturas híbridas

Tal como lo describe Brownlee (2019), las redes LSTM pueden funcionar eficientemente en datos con dependencias temporales, pero se puede mejorar el desempeño cuando se usan en modelos híbridos con CNN u otras variaciones. Es por esto, que se desarrollan y aplican dos arquitecturas híbridas para el caso de estudio 1 y se confrontan con los resultados obtenidos por los modelos de redes neuronales profundas, los dos modelos híbridos que se evalúan comparten la característica de incluir una red neuronal convolucional al extremo frontal de la estructura y se describen a continuación:

> **CNN-LSTM:** Se define una instancia secuencial agregando capas CNN en el extremo frontal seguido de capas LSTM y en la salida una capa Densa MLP. Para esto, se ajusta a las capas convolucionales la cantidad de filtros y el tamaño de núcleo a 256 y 2 respectivamente, con el fin de maximizar la exactitud del modelo. El número de filtros corresponde al número de lecturas de la secuencia de entrada y el tamaño del núcleo es el número de pasos de tiempo incluidos de cada operación de lectura de la secuencia de entrada, todo el modelo CNN está envuelto en un contenedor con una capa "TimeDistributed"; la capa de convolución es seguida por una capa de agrupación máxima (MaxPooling1D), luego estas estructuras se aplanan a un solo vector unidimensional, para usarse como un solo paso de tiempo de entrada en las dos capas LSTM con 70 y 30 unidades de memoria regularizadas, la capa LSTM se conecta a una capa MLP con 20 neuronas y con una capa de salida densa de 1 neurona para la clasificación binaria. En la figura **3-10.a** se presenta la arquitectura del modelo CNN-LSTM.

> **ConvLSTM:** Para este caso de estudio se define el ConvLSTM con una sola capa convolucional, como método de regularización se adiciona una capa de abandono del 20 % para omitir aleatoriamente este porcentaje de neuronas, la cual está conectada a una capa MLP con 50 neuronas y para la salida una capa densa de 1 neurona. Con respecto a este tipo de arquitectura, afirma Brownlee (2019): *"el ConvLSTM fue desarrollado para leer datos espacio-temporales bidimensionales, pero puede adaptarse para su uso con pronósticos de series de tiempo"* (p. 133). En la figura **3-10.b** se presenta la arquitectura del modelo ConvLSTM.

En la tabla **3-6** se presentan los resultados obtenidos con sus respectivas métricas de desempeño de los modelos híbridos y DNN aplicados al caso de estudio 1.



(a) CNN-LSTM (b) ConvLSTM

Figura 3-10.: Arquitecturas modelos híbridos

Tabla 3-6.: Comparativo del desempeño entre modelos híbridos, LSTM y CNN

Tipo	Algoritmo / Modelo	F1 -Score Macro	Recall Macro	Precisión Macro	Recall clase 1	Recall clase 0	Precisión clase 1	Precisión clase 0	F1 clase 1	F1 clase 0	ROC AUC	Exactitud
DNN híbrida	CNN-LSTM	0.93	0.95	0.90	0.92	0.99	0.81	1.00	0.86	1.00	0.95	0.99
DNN	Staked LSTM	0.89	0.94	0.86	0.88	0.99	0.72	1.00	0.80	0.99	0.94	0.99
DNN	Vanilla LSTM	0.91	0.88	0.95	0.76	1.00	0.90	0.99	0.83	1.00	0.88	0.99
DNN	CNN	0.90	0.88	0.92	0.77	1.00	0.85	0.99	0.81	1.00	0.88	0.99
DNN híbrida	ConvLSTM	0.90	0.89	0.91	0.79	0.99	0.83	0.99	0.81	1.00	0.89	0.99

El ajuste de la arquitectura de las redes neuronales profundas se efectúa mediante la técnica de búsqueda en cuadricula con repeticiones, la evaluación del rendimiento del modelo se realiza sobre el conjunto de datos de prueba, en este trabajo la técnica de validación cruzada k-Fold no se aplica para la evaluación y ajuste de las redes neuronales profundas, en este sentido Brownlee (2017), afirma: *"si tenemos los recursos, usaríamos la validación cruzada k-fold. Pero esto generalmente no es posible dado el uso de grandes conjuntos de datos en el aprendizaje profundo y la lenta velocidad de entrenamiento del modelo"* (p. 164).

Cuando se comparan las medidas de desempeño de la tabla **3-6**, se aprecia que con la red neuronal híbrida CNN-LSTM se obtiene el mejor desempeño, seguido de las estructuras LSTM y las redes neuronales con características convolucionales. La selección del modelo híbrido CNN-LSTM para este caso de estudio se fundamenta en las siguientes premisas:

1. La red CNN-LSTM obtiene el mejor F1- Score macro con 93 %, es decir 4 puntos porcentuales por encima de la red neuronal LSTM de varias capas y 8 puntos porcentuales por encima de los algoritmos de conjunto XGBoost y Extra Trees que se analizan en la tabla **3-3**. Este resultado se logra debido al balance entre las medidas macro del Recall y la precisión, las cuales obtuvieron valores de 95 % y 90 % respectivamente.

2. Partiendo del hecho que la clase 1 (falla potencial) es la de mayor importancia, con la red CNN-LSTM se obtiene la medida de Recall con un valor del 92 % para la clase 1, esta medida es de alta relevancia ya que uno de los objetivos en la clasificación de fallas es minimizar el número de falsos negativos, dado que es más costoso para los ingenieros de mantenimiento etiquetar erróneamente observaciones clasificadas como estado normal de la máquina, cuando realmente son fallas potenciales o funcionales; en la figura **3-11** se presenta la matriz de confusión del modelo CNN-LSTM, donde se observa que el número de observaciones clasificadas como falsos negativos es de 28 frente a 304 observaciones clasificadas como verdaderos positivos que corresponde a fallas funcionales clasificadas correctamente por la red neuronal híbrida.

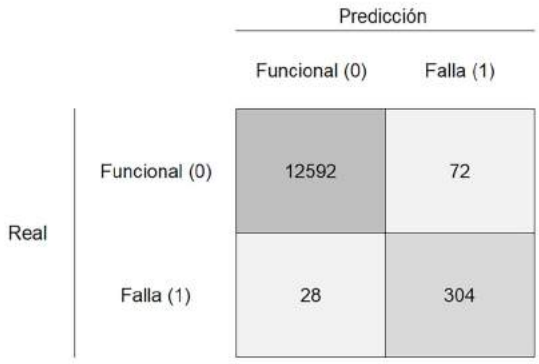


Figura 3-11.: Matriz de confusión modelo CNN-LSTM

3. El nivel de precisión de la clase mayoritaria etiquetada como clase 0 es del 99.78 %, es decir que el modelo detecta con una excelente precisión las observaciones etiquetadas como estado funcional o normal de la máquina. Tal como se observa en la figura **3-11** se obtienen de la base de prueba un total de 12.592 observaciones clasificadas como verdaderos negativos, además 72 falsos positivos que para este caso hacen referencia a observaciones clasificadas como falla de la máquina, cuando realmente son observaciones de estado funcional del activo físico.

4. Con la red CNN-LSTM se obtiene el mayor valor del ROC-AUC que registra un 95 %, esta medida de desempeño corresponde a la puntuación asignada según el área bajo la curva del gráfico que resume el rendimiento de un modelo de clasificación binaria, el análisis con la curva ROC funciona bien cuando se trata de datos desequilibrados ya que no hay sesgos que favorezcan a la clase mayoritaria. En la figura **3-12** se muestra la curva ROC de la red CNN-LSTM.

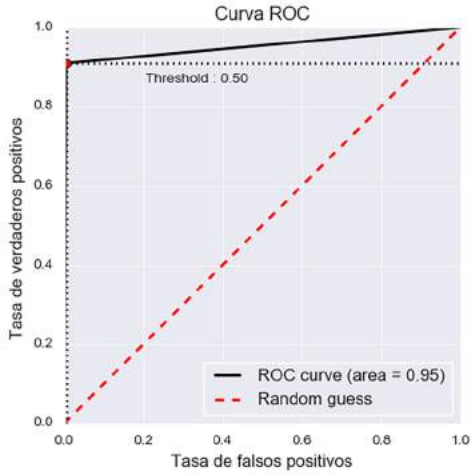


Figura 3-12.: Curva ROC modelo CNN-LSTM

5. Cuando se evalúa la curva de aprendizaje del modelo híbrido CNN-LSTM, se corrobora en la figura **3-13** que el desempeño de la red presenta un ajuste adecuado, puesto que tanto la pérdida del conjunto de entrenamiento como la de validación disminuyen y se estabilizan alrededor del mismo punto; en este caso se configura en la red neuronal una función de parada temprana para detener el entrenamiento como medida de regularización para evitar el sobreajuste.

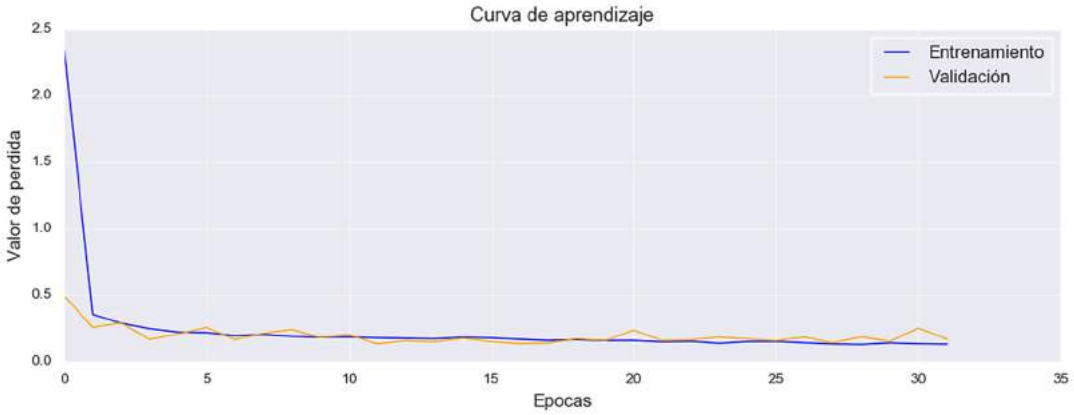


Figura 3-13.: Curva de aprendizaje modelo CNN-LSTM

De las dos redes neuronales híbridas evaluadas, la CNN-LSTM proporciona los resultados óptimos para este caso de estudio, con respecto a la red híbrida ConvLSTM a pesar de obtener un valor de F1-Score del 90 % según la tabla **3-6**, esta arquitectura presenta un problema de sobreajuste que afecta considerablemente el desempeño de la red neuronal cuando se evalúa en los datos de prueba, con el modelo ConvLSTM se obtiene el Recall de la clase minoritaria con un valor del 79 %, es decir 13 puntos porcentuales por debajo de la red CNN-LSTM, este bajo valor del Recall de la clase 1 se interpreta como un modelo de escaso desempeño con una predisposición a producir un numero de falsos negativos significativamente altos, lo que indica que es inadecuado para este caso de estudio.

3.6. Modelos de regresión aplicados en pronósticos de fallas

Los modelos de clasificación están orientados a predecir la etiqueta de una o varias observaciones, con el propósito de generar alertas tempranas de fallas potenciales o funcionales de los activos físicos, una alternativa a los modelos de clasificación son los modelos de regresión cuyo objetivo es predecir valores cuantitativos de la variable de salida involucrada

en el problema, que para el caso de los pronósticos de fallas en mantenimiento la finalidad es pronosticar el tiempo hasta la falla (TTF) o tiempo medio entre fallas (MTBF) de un componente o sistema productivo.

Con el propósito de evaluar la habilidad del algoritmo de regresión se debe analizar el error de las predicciones con respecto a los valores reales, existen diversas métricas para calcular el error de la predicción, las dos dimensiones principales para evaluar el rendimiento del modelo son el sesgo y la exactitud, la primera es la tendencia persistente del modelo a realizar estimaciones por encima o por debajo de los valores observados, la segunda mide la cercanía de los valores estimados por el modelo con los valores reales observados, afirma Caplice (2017): *"ninguna métrica individual hace un buen trabajo capturando ambas dimensiones, por lo que vale la pena tener múltiples. Las métricas más comunes utilizadas son MAPE y RMSE para mostrar la precisión y MPE para el sesgo"* (p. 16). Las definiciones y fórmulas de estas tres medidas de desempeño se muestran a continuación:

- **Error porcentual absoluto medio (MAPE):** Está definido como el promedio de los errores porcentuales absolutos entre los resultados observados y los valores estimados, tal como lo muestra la ecuación 3-5.

$$MAPE = \frac{\sum_{t=1}^{n_o} \frac{|e_t|}{y_t}}{n_o}, \tag{3-5}$$

  donde n_o es el número de observaciones, y_t es el valor real observado en el tiempo t y $e_t = y_t - \hat{y}_t$, es el error calculado como la diferencia entre el valor real observado y el valor estimado.

- **Raíz del error cuadrático medio (RMSE):** Mide el error promedio del modelo al estimar el resultado de una observación, está definido como la diferencia cuadrática promedio entre los valores reales de salida observados y los valores estimados por el modelo, tal como se muestra en la ecuación 3-6.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n_o} e_t^2}{n_o}}, \tag{3-6}$$

  donde n_o es el numero de observaciones y $e_t = y_t - \hat{y}_t$, es el error calculado como la diferencia entre el valor real observado y el valor estimado.

- **Error porcentual medio (MPE):** Está definido como la diferencia relativa promedio entre los resultados observados y los valores estimados, tal como lo muestra la ecuación 3-7.

$$MPE = \frac{\sum_{t=1}^{n_o} \frac{e_t}{y_t}}{n_o}, \tag{3-7}$$

donde n_o es el número de observaciones, y_t es el valor real observado en el tiempo t y $e_t = y_t - \hat{y}_t$, es el error calculado como la diferencia entre el valor real observado y el valor estimado.

El MAPE es una medida relativa, es decir mide el error de predicción como un porcentaje, lo cual es una ventaja puesto que provee una manera intuitiva de evaluar el error del modelo, en coherencia con Swamidass (2000): *"MAPE tiene un atractivo administrativo y es una medida comúnmente utilizada en pronósticos. Cuanto más pequeño sea el MAPE, mejor será el pronóstico"* (p. 30). Los errores como porcentaje son parte del lenguaje habitual en ingeniería, por esta razón MAPE es un concepto de fácil interpretación. El RMSE es una medida absoluta que tiene la propiedad de estar en las mismas unidades que la variable de respuesta, esta métrica informa sobre el tamaño promedio de los errores de pronóstico sin importar su signo, la principal desventaja es su interpretación ya que amplifica y penaliza con mayor fuerza aquellos errores de mayor magnitud; tanto MAPE como RMSE son métricas relevantes para comparar el desempeño en términos de precisión de los algoritmos de ML y DNN, para complementar se calcula el MPE como una medida de sesgo de la predicción indicando si los valores estimados del modelo están sobre o por debajo de los valores reales observados.

Como antecedentes de modelos predictivos de regresión en el caso de estudio 1 están los trabajos de Saxena et al. (2008), cuyo artículo describe detalladamente la operación de la turbina con las variables asociadas a su funcionamiento y modelan la propagación de daños en el sistema y el trabajo de Vardon (2018), donde aplica un algoritmo predictivo para calcular el tiempo de vida restante del activo con una red neuronal LSTM.

Con el propósito de desarrollar el modelo de regresión para el caso de estudio 1, primero se calcula el tiempo hasta la falla de cada una de las observaciones del conjunto de datos de entrenamiento, para lo cual se toma como referencia el código del trabajo de Uz (2017). Con los datos organizados se modelan 12 algoritmos de regresión de ML, manteniendo los hiperparámetros en valores por defecto de cada algoritmo; las características, ventajas y desventajas de los diferentes algoritmos se explican en el trabajo de Lanners (2019). Adicional a los algoritmos de regresión de ML se aplican 4 modelos de regresión con DNN.

Los modelos de ML y DNN se evalúan comparando los pronósticos de la base de prueba con los valores reales del tiempo hasta la falla de cada observación; en la tabla **3-7** se presentan los resultados obtenidos con las respectivas métricas de calidad del estudio de modelos de regresión para el cálculo del TTF.

Tabla 3-7.: Modelos de regresión ML y DNN

Tipo	Algoritmo / Modelo	MAPE	RMSE	MPE
DNN	Staked LSTM	22.72 %	42.81	2.01 %
DNN	Bidirectional LSTM	23.11 %	42.01	0.28 %
DNN	Vanilla LSTM	23.59 %	41.45	-2.88 %
DNN	MLP Regressor	24.00 %	41.01	-5.15 %
Conjunto	Gradient Boosting Regressor	24.38 %	40.80	-6.80 %
Conjunto	Random Forest Regressor	25.06 %	41.48	-7.51 %
Conjunto	Bagging Regressor	26.24 %	43.55	-7.20 %
Conjunto	Extra Trees Regressor	26.24 %	43.60	-7.58 %
Lineal	Logistic Regression	26.54 %	48.26	6.31 %
No lineal	Support Vector Regression	27.14 %	48.01	4.71 %
Lineal	Linear Regression	29.21 %	42.90	-9.75 %
Lineal	Bayesian Ridge Regression	29.22 %	42.91	-9.74 %
Lineal	ElasticNet	32.25 %	45.17	-7.57 %
No lineal	AdaBoost regressor	32.42 %	44.69	-20.29 %
No lineal	Decision Tree Regressor	35.26 %	59.90	-6.93 %
Lineal	Passive Aggressive Regressor	36.19 %	55.08	20.68 %

Teniendo en cuenta las tres métricas de desempeño se observa en la tabla **3-7** que los algoritmos con la mejor precisión (menor MAPE y RMSE) y con menor sesgo (MPE cercano a cero) son las redes neuronales profundas (DNN), específicamente las redes neuronales LSTM y el perceptrón multicapa; en el segundo grupo ordenados en términos de desempeño están los algoritmos de conjunto, por último, se presentan los algoritmos de ML lineales y no lineales con los mayores errores de los experimentos.

El algoritmo con menor error relativo es la red apilada LSTM con un MAPE estimado del 22.72 % y el algoritmo con menor error absoluto es Gradient Boosting con un RMSE estimado de 40.8 ciclos (unidad de medida de tiempo), pero si tomamos en cuenta las tres métricas, el algoritmo de regresión con mejor desempeño es la red LSTM bidireccional (BLSTM) con un MAPE de 23.11 %, un RMSE de 42.01 ciclos y un MPE de 0.28 % lo que indica un bajo sesgo de los valores estimados; con respecto a las ventajas de las redes neuronales BLSTM afirma Brownlee (2020*a*): *"en algunos problemas de predicción de secuencia, puede ser beneficioso permitir que el modelo LSTM aprenda la secuencia de entrada hacia adelante y hacia atrás para concatenar ambas interpretaciones"* (p. 129). La arquitectura del modelo BLSTM con el que se obtuvo el mejor rendimiento es la siguiente:

> **Bidirectional LSTM:** Se genera una instancia secuencial con dos capas ocultas LSTM las cuales contienen 200 y 100 unidades de memoria respectivamente, las capas LSTM están dentro de una envolvente bidireccional con 400 unidades de memoria, una tercera capa de salida densa MLP con 100 neuronas está completamente conectada con 1 neurona, estas tres capas están envueltas en un contenedor con una capa "TimeDistribute", el modelo se compila con el fin de minimizar la pérdida de registro con la métrica de error medio absoluto, implementando el algoritmo de descenso de gradiente "Adam". En la figura **3-14** se presenta la arquitectura del modelo Bidirectional LSTM.

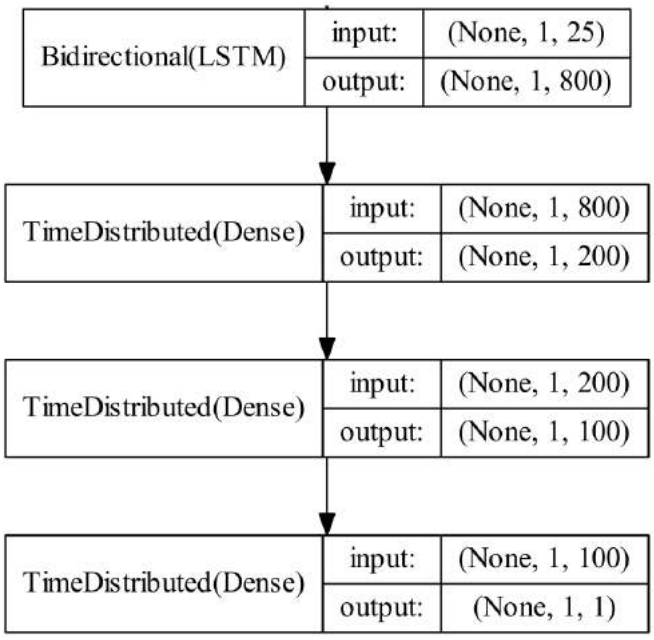


Figura 3-14.: Arquitectura modelo Bidirectional LSTM

Teniendo en cuenta los resultados de la tabla **3-7**, vale la pena resaltar que los métodos de conjunto obtienen un buen rendimiento en este experimento, ya que si comparamos el algoritmo Gradient Boosting Regressor (GBR) con la red BLSTM, la diferencia en términos del MAPE es de tan solo 1.27 %, analizando el RMSE, el algoritmo GBR alcanza el menor error del grupo de experimentos con un valor de 40.80 ciclos, con respecto al sesgo el MPE registra un -6.8 %, este valor negativo indica que los valores estimados por el modelo en general son mayores que los valores reales observadores.

Una de las ventajas significativas a la hora de estimar el modelo de regresión es que el algoritmo GBR requiere menor tiempo en la estructuración de datos, programación y entrenamiento que las redes neuronales profundas, en coherencia con este resultado está el trabajo de Olson et al. (2018), en este realizan un análisis de 13 algoritmos de ML en un conjunto de

165 bases de datos, justificando la fuerza de los algoritmos de conjunto basados en árboles de última generación y el impresionante rendimiento del algoritmo Gradient Boosting. Por esta razón a continuación se exploran diferentes ajustes de hiperparámetros con la técnica de validación cruzada en implementaciones eficientes y variaciones del Gradient Boosting Regressor. En la tabla **3-8** se presentan los resultados que se obtienen con estos modelos de regresión basados en GBR.

Tabla 3-8.: Desempeño de los algoritmos Gradient Boosting

Tipo	Algoritmo / Modelo	MAPE	RMSE	MPE
Conjunto	XGBoost Grid Search Cross-validation	23.99 %	40.51	-6.13 %
Conjunto	GBR Grid Search Cross-validation	24.33 %	40.31	-7.13 %
Conjunto	GBR hiperpárametros por defecto	24.38 %	40.80	-6.80 %
Conjunto	GBR With LightGBM	24.56 %	41.37	-6.55 %
Conjunto	Histogram GBR	24.58 %	41.33	-6.65 %

Tal como se observa en la tabla **3-8**, el algoritmo GBR con sistema de refuerzo XGBoost (Extreme Gradient Boosting) obtiene los mejores resultados del grupo de los algoritmos de aprendizaje automático del grupo de gradiente estocástico; con respecto a este algoritmo en el artículo de Chen & Guestrin (2016), describen detalladamente el "Gradient Boosting XGBoost" y concluyen que tiene la capacidad de resolver problemas del mundo real usando una cantidad mínima de recursos computacionales, es por este motivo que este algoritmo se utiliza ampliamente por la comunidad de científicos de datos para obtener resultados de vanguardia.

4. Implementación de un modelo de Machine Learning en una aplicación Web

Después de explorar la configuración apropiada de los algoritmos de ML y realizar una búsqueda automática en cuadrícula con validación cruzada K-fold estratificada para ajustar los hiperparámetros, se procede a evaluar el desempeño de los modelos en los datos de prueba (test), con el modelo seleccionado el paso subsecuente es agrupar el conjunto de datos de entrenamiento y prueba con el propósito de entrenar de nuevo el modelo y guardarlo para hacer predicciones, en este sentido afirma Brownlee (2017): *"se deben reunir todos los datos en un gran conjunto de datos de entrenamiento y ajustarlos a su modelo"* (p. 190). En otros términos, se finaliza con un modelo cuya configuración seleccionada se ajusta en todos los datos disponibles, en este paso ya no hay división de entrenamiento y prueba ni tampoco pliegues de validación cruzada, debido a que se está guardando el modelo para una aplicación operativa posterior con datos no observados; la habilidad del modelo ajustado con los datos de entrenamiento se evaluó con los datos de prueba en la etapa preliminar al proceso de almacenado del modelo final mitigando los riesgos de sobreajuste, además la principal ventaja de esta técnica es que se utilizan el 100 % de los datos disponibles para entrenar el modelo final y prepararlo apropiadamente para la predicción de nuevos datos en un entorno de producción aplicado.

Con el modelo final ajustado, entrenado, evaluado y grabado, es factible realizar predicciones desde el software en el cual se configura el algoritmo, sin embargo, otra alternativa es poner el modelo en producción, en otros términos, implementar una aplicación con la finalidad que el ingeniero de mantenimiento pueda acceder al modelo y realizar la predicción y/o clasificación de fallas de sus activos físicos productivos, esto representa una ventaja competitiva que genera valor, ya que no se requiere que el usuario final tenga instalado un software especializado y/o domine el lenguaje de programación con el que se desarrolla el algoritmo.

En el proceso de puesta en producción de los modelos de ML o DNN se puede optar por una aplicación en un dispositivo móvil. Según Tang (2018), uno los beneficios de ejecutar el modelo entrenado en una aplicación móvil (APP) es que no se requiere conectividad a internet, además para la programación de estas APP se cuenta con herramientas de código abierto en

TensorFlow (biblioteca de Python) tales como TensorFlow Mobile y TensorFlow Lite, con este conjunto de herramientas se pueden desarrollar aplicaciones listas para producción para dispositivos Android e iOS.

Otra opción para colocar a disposición el modelo para el usuario final es la implementación de una aplicación Web; en este trabajo se toma el caso de estudio 3 y se implementa una aplicación Web básica para que se ejecute en el host local, es decir que se utiliza como servidor la propia computadora. El código que se utiliza para la puesta en producción del modelo fue adaptado del trabajo de Sagar (2019). En la figura **4-1** se presenta la estructura del proyecto para la puesta en producción del modelo del caso de estudio 3, de igual manera, se describen a continuación de forma resumida los pasos principales:

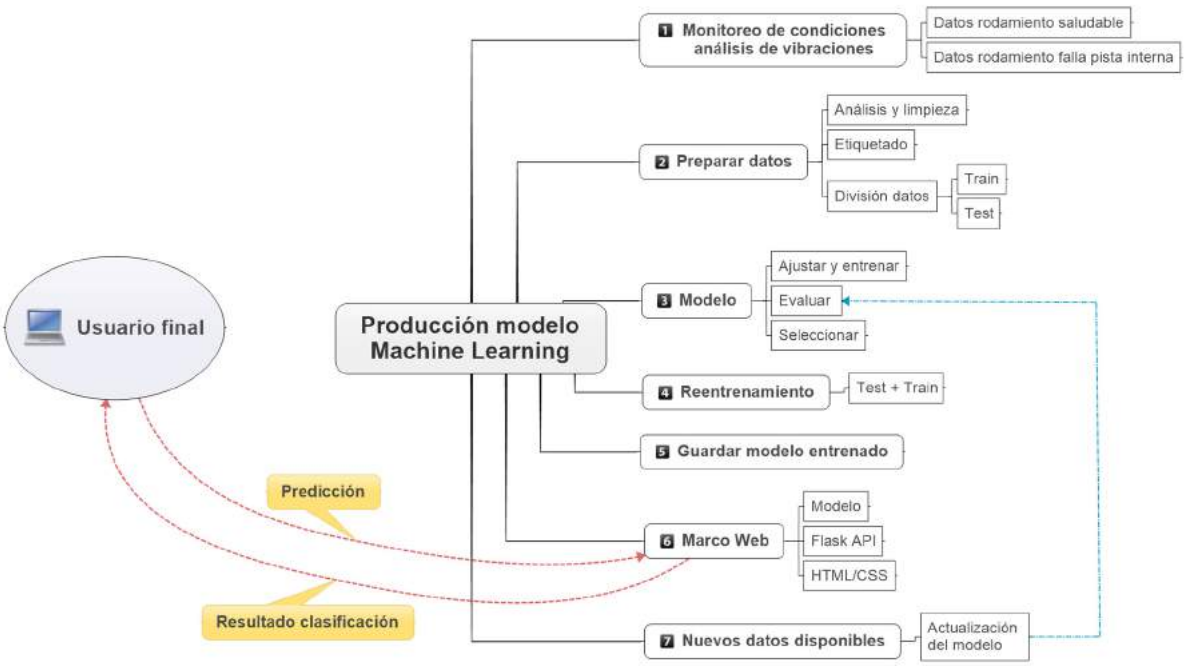


Figura 4-1.: Estructura proyecto Web: caso de estudio 3

1. **Monitoreo de condiciones:** El primer paso es adquirir los datos por técnicas de monitoreo de condiciones, específicamente para este caso se implementa un análisis de vibraciones donde se adquieren las variables de aceleración y velocidad en cuatro rodamientos con 2.000.000 de observaciones por cada rodamiento, en este experimento se toman dos rodamientos en estado saludable y dos rodamientos que presentan falla en la pista interna.

2. **Preparar datos:** En el segundo paso se analizan, organizan y se etiquetan los datos, a las observaciones del rodamiento saludable se les asigna la clase 0 y las observaciones del

rodamiento en falla se les asigna la clase 1; posteriormente se seleccionan un conjunto de datos de un rodamiento en estado saludable y un conjunto de datos de otro rodamiento con falla en pista interna para conformar los datos de entrenamiento, así mismo se seleccionan un conjunto de datos de otro rodamiento en estado saludable y otro conjunto con falla en pista interna para conformar los datos de prueba; se ajustan hiperparametros con validación cruzada estratificada y se evalúan los modelos ajustados en la base de prueba.

3. **Modelo:** Para este ejercicio se selecciona el algoritmo de Machine Learning Gradient Boosting, considerando que en la tabla **3-5**, se evidencia un excelente desempeño de las métricas macro de F1, recall y precisión con valores de 95 %, 95 % y 96 % respectivamente, adicionalmente al buen rendimiento del modelo en los datos de prueba este algoritmo fue diseñado para maximizar la eficiencia del tiempo de cómputo y mejorar el uso de los recursos de memoria de la máquina disponibles cuando se entrena el modelo, tal como lo expone Sapountzoglou et al. (2020), este algoritmo de conjunto combina varios árboles de decisión y una de las ventajas es que su costo computacional y tiempo de entrenamiento del modelo es relativamente bajo, por lo tanto este algoritmo es adecuado para aplicaciones de predicción de fallas cuando se analizan datos en tiempo real, esta propiedad también favorece la actualización continua del modelo Web con nuevos datos con el fin de maximizar la precisión en la detección de fallas del activo físico.

4. **Reentrenamiento:** Con el modelo Gradient Boosting evaluado y seleccionado, se toman las 8.000.000 de observaciones que corresponde al 100 % de los datos de entrenamiento y prueba para constituir un nuevo gran conjunto con el cual se entrena de nuevo el modelo.

5. **Guardar Modelo:** Se procede a guardar el modelo entrenado en un archivo con el propósito de cargarlo cuando se requiera hacer predicciones de una o varias observaciones nuevas, en este caso se utiliza el formato "Pickle" que es la herramienta estándar de Python para la serialización de objetos.

6. **Marco Web:** Según Sagar (2019), el objetivo principal de la puesta en producción del modelo en una aplicación Web es generar valor facilitando las predicciones para el usuario final; el marco Web utiliza el paquete Flask de Python, esta herramienta nos brinda una serie de utilidades que facilita la construcción de páginas Web y permite que el usuario final interactúe con el modelo, para esto se ingresan los datos de aceleración y velocidad en la página web y por medio del modelo de clasificación binaria guardado, el sistema entrega la etiqueta de clase que hace referencia al estado de salud del rodamiento.

7. **Nuevos datos disponibles:** Para el adecuado mantenimiento del modelo se requiere evaluar periódicamente el desempeño del modelo con los nuevos datos adquiridos, en

términos de Brownlee (2017): *"un modelo es tan bueno como los datos utilizados para entrenarlo. Si los datos utilizados para entrenar su modelo eran de hace un año, tal vez esa nueva información recopilada hoy resultaría en un modelo diferente y más hábil"* (p. 199). Esto conlleva a la actualización del modelo incorporando nuevos datos con la finalidad de mantener o mejorar la habilidad de predicción de fallas, evitando así que se degrade el desempeño del modelo con el tiempo.

En la figura **4-2** se presenta la página Web local con dos campos para ingresar los datos de aceleración y velocidad, los cuales se adquieren por medio de la técnica de análisis de vibraciones en el rodamiento, cuando el usuario final ingresa los datos de una observación y da clic en el botón "predicción de estado", el marco Web entrega de inmediato el resultado de la clasificación binaria del modelo entrenado que esta guardado, en este ejemplo los datos que se ingresan de aceleración=0.199713 y velocidad=4.323565 presentan en pantalla la etiqueta de clase 1 que efectivamente corresponden a una alerta por falla en pista interna en el rodamiento.

Figura 4-2.: Aplicación Web caso de estudio 3

El marco Web que se implementa en este trabajo tiene características básicas y está orientado a los ingenieros de datos que desean ir más allá del desarrollo de un modelo de ML, para que un sistema se considere de producción comercial se deben establecer múltiples herramientas de programación utilizando lenguajes tales como HTML y CSS, con el propósito de estructurar y definir el estilo de la página Web, adicionalmente a nivel comercial se requiere desarrollar sistemas de mayor complejidad y seguridad para el manejo de información.

5. Análisis de supervivencia

El análisis de supervivencia es un método estadístico que estudia el tiempo de supervivencia y los factores que influyen en él, como expresa Borges (2005): *"el análisis de supervivencia tiene como objeto de estudio el tiempo de seguimiento hasta la ocurrencia de un evento de interés"* (p. 244). Una de las áreas con mayor nivel de aplicación de esta metodología es la medicina, específicamente en los estudios clínicos, los cuales se expresan en términos del tiempo de supervivencia, según Fernández (1995), la medida de supervivencia no es exclusiva para estudiar el tiempo hasta la muerte de un paciente ya que también se puede estudiar el tiempo hasta la recaída de un tratamiento, progresión de una enfermedad y respuesta de una intervención médica, tal como el estudio de Borges (2005), donde se analiza el riesgo de muerte de pacientes que acudían al servicio de diálisis peritoneal del Hospital Clínico Universitario de Caracas entre los años 1980 y 1997, en este trabajo el autor explica de forma detallada la teoría del análisis de supervivencia, en el cual concluye que es una técnica muy poderosa para modelar eventos en datos temporales con variables que están asociadas al riesgo de muerte en los pacientes.

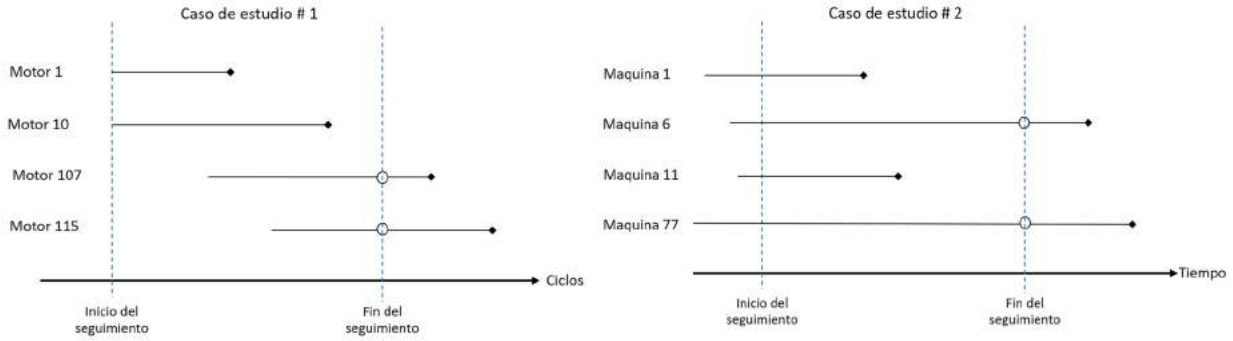
El análisis de supervivencia se puede aplicar en varias disciplinas científicas, en el ámbito de la sociología se encuentran aplicaciones con modelos de supervivencia, tal como el estudio de Fagbamigbe et al. (2020), donde se evalúa el momento de la primera incidencia de violencia domestica contra las mujeres después del matrimonio y determina los factores asociados con estos tiempos de ocurrencia de los eventos. En la ingeniería, específicamente en el área de mantenimiento los eventos de interés son las fallas que se presentan en los activos físicos, como precedente en la aplicación de esta técnica se documenta el trabajo de investigación de Montoya (2011), en el cual se comparan dos modelos de supervivencia con el propósito de estimar los tiempos hasta la falla de los tramos de tubería de una empresa de suministro de agua en España.

En este trabajo se aplican técnicas de análisis de supervivencia para los casos de estudio 1 y 2, estos dos conjuntos de datos comparten las siguientes características que son fundamentales para emplear esta metodología:

- Los conjuntos de datos poseen características de series de tiempo, por lo que es viable calcular el tiempo hasta la falla (TTF, por sus siglas en inglés) de las máquinas objeto del estudio.

- Los conjuntos de datos contienen individuos que están censurados por la derecha, es decir, hay máquinas que presentaron una falla en el tiempo de seguimiento, algunas máquinas se mantienen en estado funcional sin presentar fallas desde el inicio del seguimiento hasta el final del seguimiento y otras máquinas entraron al estudio después del inicio de seguimiento, estos dos últimos tipos de eventos son censurados y deben considerarse como tales a la hora del análisis.

En la figura **5-1.a** se presenta el esquema de los datos del caso de estudio 1, las líneas verticales punteadas indican el inicio y final del seguimiento, los rombos indican la falla del activo físico o componente y los círculos denotan los eventos censurados por la derecha. En la figura **5-1.b** se muestra el esquema de los datos del caso de estudio 2 con eventos censurados por la derecha y puesta en marcha del activo antes del inicio del seguimiento.



(a) Caso de estudio 1 (b) Caso de estudio 2

Figura 5-1.: Eventos censurados

En los análisis de supervivencia es viable implementar estimaciones no paramétricas, semi-parametricas o modelos paramétricos, el alcance de este capítulo es la implementación de métodos no paramétricos para estimar la probabilidad de supervivencia de un activo físico hasta un punto del tiempo, en este sentido afirma Fernández (1995): *"los métodos estadísticos más utilizados son los no paramétricos"* (p. 4). El uso de los estimadores no paramétricos en los análisis de supervivencia de acuerdo con Moore (2016), se debe a que brindan la flexibilidad suficiente para analizar fenómenos de los cuales se desconoce el tipo de distribución o no se adaptan una familia paramétrica especifica. La función de supervivencia está definida por la ecuación 5-1:

$$S\left(t\right) = Pr\left(T > t\right), \;\; 0 < t < \infty, \tag{5-1}$$

donde $S\left(t\right)$ es la probabilidad de supervivencia hasta un tiempo t, esta función toma el valor de 1 en $t = 0$ la cual disminuye o permanece constante con el tiempo y T es una variable aleatoria positiva.

5.1. Estimación de la curva de supervivencia

Según Moore (2016), el estimador no paramétrico de la función de supervivencia más utilizado fue propuesto por Kaplan & Meier (1958). Este estimador formalmente se define como el producto sobre los tiempos de falla de las probabilidades condicionales de sobrevivir al siguiente tiempo de falla, tal como se muestra en la ecuación 5-2.

$$\hat{S(t)} = \prod_{t_i \leq t} (1 - \hat{q}_i) = \prod_{t_i \leq t} \left(1 - \left(\frac{d_i}{n_i}\right)\right), \tag{5-2}$$

donde n_i es el número de individuos en riesgo en el tiempo t_i, d_i es el número de individuos que fallan en el tiempo t_i y $\hat{q}_i$ es la probabilidad de falla.

5.1.1. Curva de supervivencia del caso de estudio 1

Con el propósito de estimar la curva de supervivencia del caso de estudio 1, primero se calcula el número de ciclos hasta que se presenta la falla de cada uno de los 100 motores de la base de entrenamiento (máquinas que presentaron una falla en el tiempo de seguimiento), segundo se calcula el número de ciclos de operación de cada uno de los 100 motores de la base de prueba (máquinas que no presentaron falla y entraron al estudio después del inicio de seguimiento), en el tercer paso se etiqueta cada motor con el estado, el cual indica si durante el estudio se presentó una falla (estado=1), en caso contrario se considera que está censurado por la derecha (estado=0). Los valores resumidos obtenidos con el estimador de Kaplan y Meier se presentan en la tabla **5-1**.

Tabla 5-1.: Resumen del estimador de Kaplan y Meier caso de estudio 1

N	Eventos	Mediana	0.95LCL	0.95UCL
200	100	202	199	214

El número total de registros, denotado por N en este caso de estudio corresponde a 200 motores, de los cuales 100 presentan eventos de falla, con una mediana del tiempo de supervivencia de 202 ciclos, cuyo límite de confianza inferior del 95 % para la supervivencia de los motores es de 199 ciclos y un límite de confianza superior del 95 % de 214 ciclos. En la figura **5-2** se observa la curva de la función estimada de supervivencia, en el eje y de la gráfica tenemos la probabilidad de supervivencia en porcentaje, en el eje x el tiempo en ciclos y la curva azul representa la respectiva función de supervivencia con su intervalo de confianza al 95 %.

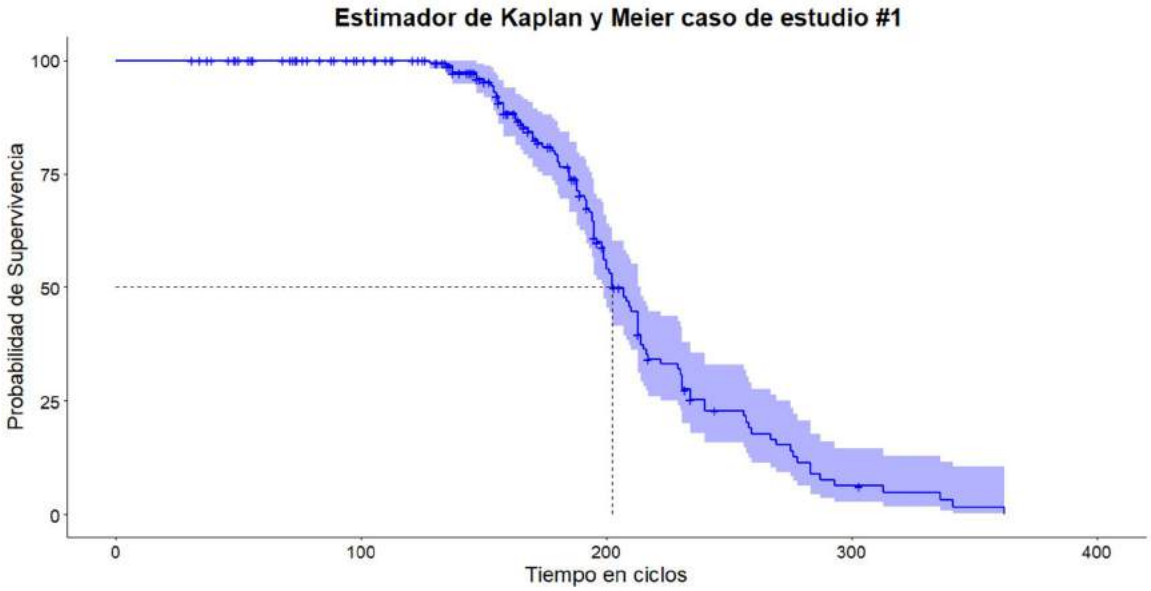
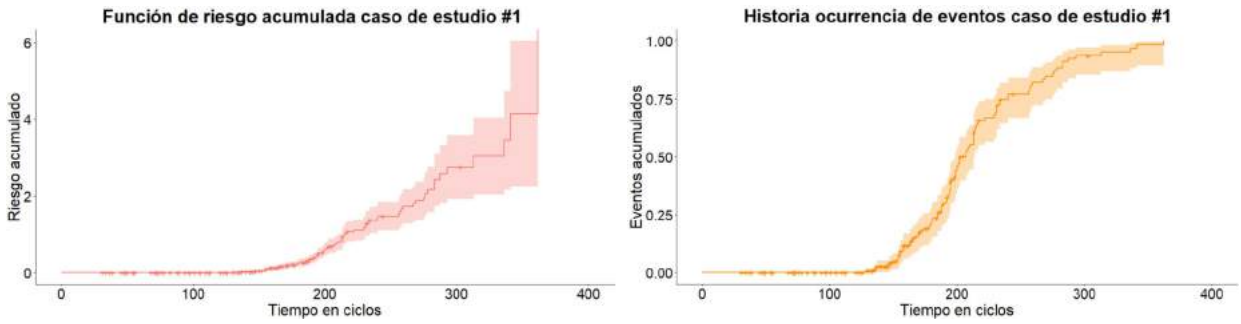


Figura 5-2.: Curva función estimada de supervivencia caso de estudio 1

Como se puede observar en la figura **5-2**, la probabilidad de supervivencia de los motores hasta el ciclo 137 es aproximadamente del 100 %, a partir de este ciclo empieza a disminuir, en líneas puntadas se señala la mediana con una probabilidad del 50 % de supervivencia, a partir de 257 ciclos la probabilidad de supervivencia del motor es menor del 20 %.

En la figura **5-3.a** se presenta la función de riesgo acumulada con la cual se representa las estimaciones del peligro de falla de los motores, como se puede observar el riesgo es mínimo al principio de la vida de los motores y va aumentando con el número de ciclos de operación, a partir de 200 ciclos el riesgo de falla funcional aumenta significativamente. En la figura **5-3.b** se muestra la historia de ocurrencia de eventos acumulados, donde se evidencia que a partir de 300 ciclos aproximadamente el 99 % de los motores presentan un evento de falla.



(a) Función de riesgo acumulada

(b) Historia eventos acumulados

Figura 5-3.: Función de riesgo e historia eventos caso de estudio 1

5.1.2. Curva de supervivencia del caso de estudio 2

Con el propósito de realizar el análisis de supervivencia del caso de estudio 2, primero se calcula el tiempo medio entre falla (MTBF, por sus siglas en inglés) de cada una de las 100 máquinas, una de las diferencias de este caso de estudio con respecto al caso de estudio 1, es que se cuentan con características de las máquinas tales como modelo, edad operacional, histórico de actividades de mantenimiento ejecutadas en el activo y reportes de alertas por errores que presenta el activo durante el tiempo de monitoreo, por esta razón, como segundo paso se organiza e incluye la información para la estimación de supervivencia del activo, por último se etiqueta cada máquina con el estado, el cual indica si durante el estudio se presentó una falla (estado=1), caso contrario las máquinas que se mantengan en estado funcional sin presentar fallas desde el inicio del seguimiento hasta el final del seguimiento, estas se consideran datos censurados por la derecha (estado=0).

En la figura **5-4** se presenta la curva de la función estimada de supervivencia obtenida con el estimador de Kaplan y Meier, en la gráfica observamos que la mediana del tiempo de supervivencia es de 48.6 días, con un intervalo de confianza del 95 % cuyo rango varía desde 42.9 días a 54 días, también se observa que después de 65 días la probabilidad que la máquina continúe en estado funcional sin presentar fallas es de tan solo el 30 %.

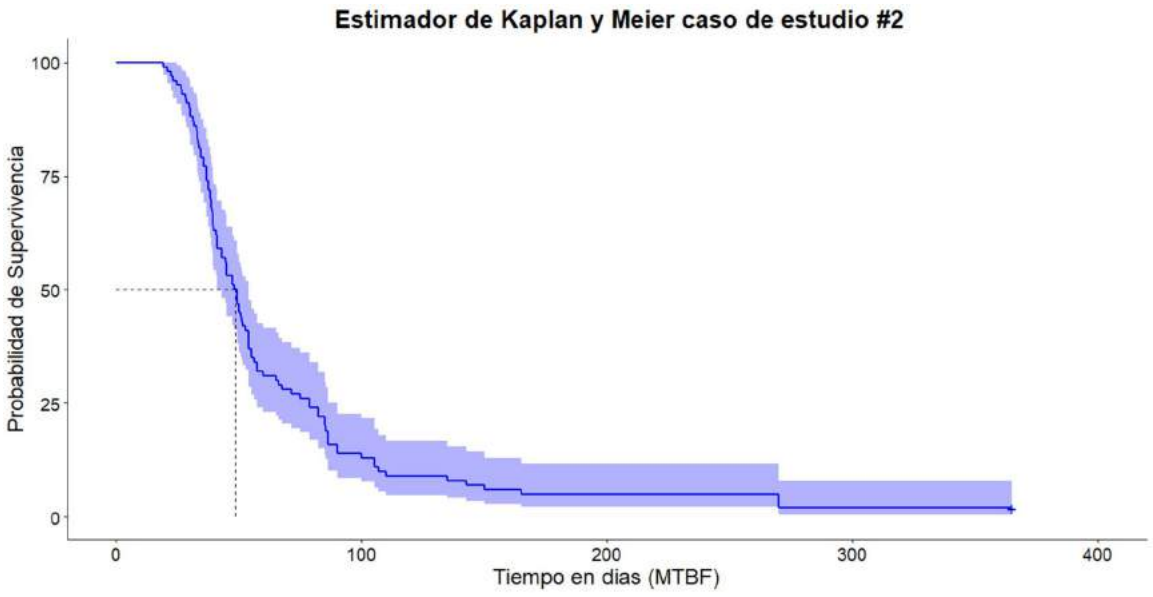


Figura 5-4.: Curva función estimada de supervivencia caso de estudio 2

Dado que en el caso de estudio 2 contamos con la variable "modelo", la cual hace referencia al modelo especificado en la placa de características de la máquina, se procede a analizar y comparar los tiempos de supervivencia de los cuatro modelos de máquinas disponibles, con el

objetivo de validar si el modelo de la máquina influye en la disponibilidad inherente del activo físico. Antes de trazar las curvas de supervivencia por cada modelo se realiza una prueba no paramétrica de equivalencia, utilizando la metodología para comparación de grupos de tiempos de supervivencia descrita en Moore (2016), en la tabla **5-2** se presentan los resultados de la prueba de equivalencia para comparación de grupos de tiempos de supervivencia.

Tabla 5-2.: Prueba de equivalencia para comparación de grupos

	N	Observado	Esperado	(O-E)2/E	(O-E)2/V
modelo=model1	16	16	4.25	32.5084	36.444
modelo=model2	17	17	9.29	6.3932	7.404
modelo=model3	35	34	35.75	0.0861	0.143
modelo=model4	32	31	48.70	6.4359	14.363

Chi-cuadrado= 51.3 con 3 grados de libertad, p-valor= 4e-11

En la tabla **5-2** el número de máquinas por cada uno de los cuatro modelos está representado en la columna N, el valor observado corresponde al número de máquinas por modelo que no están censuradas, el valor del estadístico chi-cuadrado es 51.3 con 3 grados de libertad, se obtiene un $p - valor = 4 \times 10^{-11}$ el cual es estadísticamente significativo al nivel del 5 %, en otras palabras, este resultado indica que es viable trazar y comparar las curvas de supervivencia por cada modelo ya que son estadísticamente diferentes.

Las curvas de supervivencia por modelo de máquina se presentan en la figura **5-5**, donde se observa que la mediana del tiempo de supervivencia de modelo 1 es de 32.2 días, con un intervalo de confianza del 95 % cuyo rango varía desde 28.8 días a 41.2 días, la mediana del modelo 2 es de 39 días, con un intervalo de confianza del 95 % cuyo rango varía desde 34.5 días a 50 días, la mediana del modelo 3 es de 51.4 días, con un intervalo de confianza del 95 % cuyo rango varía desde 45 días a 67.5 días y por último el modelo 4 que, de hecho, muestra una ventaja de supervivencia sobre los demás modelos con una mediana de 65.5 días, con un intervalo de confianza del 95 % cuyo rango varía desde 54 días a 100 días.

Tal como se observa en la figura **5-5** las máquinas que revelan mayor vulnerabilidad a los fallos corresponden a los modelos 1 y 2, para los cuales a partir de 30 ciclos disminuye la probabilidad de supervivencia rápidamente, por lo tanto, podemos deducir que estos dos modelos de máquinas presentan mayor frecuencia de fallas, lo que aumenta el gasto de mantenimiento y riesgos por perdidas de producción.

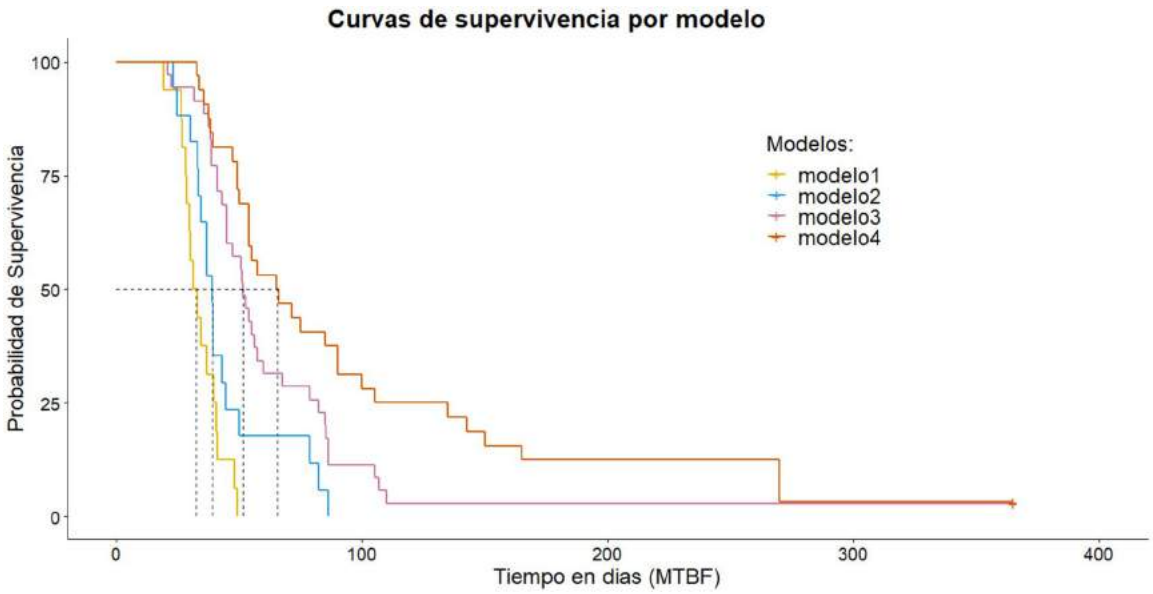


Figura 5-5.: Curvas de supervivencia por modelo caso de estudio 2

En la práctica, al obtener un tiempo medio entre fallas mayor, se maximiza la disponibilidad inherente del activo físico, de acuerdo con Mora (2009): *"la disponibilidad inherente es la probabilidad que el sistema opere satisfactoriamente cuando se requiere"* (p. 80). De manera que, si se parte del supuesto que los tiempos de reparación son iguales en todos los modelos, se concluye que la máquina del modelo número 4 brinda la mayor disponibilidad inherente minimizando los tiempos de paro de operación y gastos por reparaciones.

6. Conclusiones y recomendaciones

6.1. Conclusiones

Se planteó la aplicación de métodos de Machine Learning y Deep Learning como una opción apropiada en la detección de fallas potenciales o funcionales de los activos físicos, empleando información derivada de técnicas de monitoreo de condiciones en mantenimiento, con el propósito de maximizar la disponibilidad de la maquinaria y aumentar la productividad en los procesos operativos. Para estimar los modelos, se evaluaron tres casos de estudio, cuya estructura y características de datos difieren entre sí, lo que avala que los métodos descritos en este trabajo probablemente se ajustan a múltiples aplicaciones típicas de monitoreo de condiciones en la práctica de la ingeniería de mantenimiento.

En el proceso de evaluación de rendimiento de los modelos, se comparó el desempeño de varios algoritmos de aprendizaje supervisado en aplicaciones de clasificación y regresión, en este proceso se corroboró que las redes neuronales profundas, proporcionan un excelente desempeño para resolver problemas secuenciales con múltiples variables y observaciones debido a su alto grado de flexibilidad; en especial, las redes neuronales LSTM resultan ser muy prometedoras en la aplicación moderna para el pronóstico de fallas, estas redes recurrentes son una poderosa herramienta para el análisis de datos que están estructurados como una serie de tiempo, de la misma manera, se mostró el excelente rendimiento de los algoritmos de conjunto, particularmente los algoritmos Gradient Boosting y su versión mejorada XGBoost.

Durante el desarrollo de los experimentos se comprobó la mejora del desempeño de las redes LSTM al incluir en la arquitectura un mayor número de capas ocultas y ajustar la estructura de la red en una configuración efectiva que incrementó la precisión y el Recall del modelo, igualmente, se expuso la mejora del rendimiento de los modelos de Machine Learning con el ajuste de hiperparámetros empleando la técnica de validación cruzada estratificada.

Con la implementación de las arquitecturas híbridas, se presentó la eficiencia y fortalezas al combinar capacidades de las redes neuronales profundas para el desarrollo de datos configurados como serie de tiempo. Es importante mantener monitoreada la curva de aprendizaje, ya que un aumento de capas o combinación de redes neuronales en exceso puede acarrear un sobreajuste del modelo, lo que afectaría negativamente el rendimiento del modelo cuando se trabaja con datos no conocidos.

Se describió el problema de clasificación desequilibrada, el cual es frecuente en datos de monitoreo de condición debido a la distribución desigual de clases, por lo que se probó la técnica de ajuste de pesos en las redes neuronales profundas que resultó en la mejora de las métricas de desempeño del modelo, además, con la implementación del aprendizaje sensible al costo se tomó en consideración la clasificación errónea del modelo, por esta razón, para la evaluación de los casos de estudio se utilizó la métrica F1 que equilibró la importancia entre las consecuencias generadas por los falsos negativos y falsos positivos.

En el proceso de comparación de algoritmos de regresión para la predicción del tiempo hasta la falla del activo físico, se validó que la red LSTM bidireccional obtuvo el mejor desempeño en términos de precisión y sesgo del modelo, de igual forma, se exploraron diferentes configuraciones del algoritmo Gradient Boosting Regressor con apropiados resultados en la precisión del modelo y partiendo de su eficiencia computacional puede ser un punto de partida para la predicción de fallas si el tiempo de entrenamiento y recursos informáticos son limitados.

El enfoque presentado con el método estadístico de análisis de supervivencia permitió por medio del estimador no paramétrico, calcular la probabilidad que un activo se mantenga en estado funcional hasta un punto del tiempo y estimar el tiempo medio hasta la falla con su respectivo intervalo de confianza, adicionalmente este método permitió analizar y comparar los tiempos de supervivencia en función de las características que influyen en la disponibilidad inherente del activo físico. Cuando se comparan las técnicas de Machine Learning con el método estadístico utilizado, se comprobó la factibilidad de aplicación de ambos métodos para la detección temprana de fallas, de igual manera se observó la capacidad de procesamiento de grandes volúmenes de datos con los algoritmos de aprendizaje supervisado, sin necesidad de validar supuestos con respecto las propiedades de los datos.

Con respecto a la puesta en producción del modelo de machine Learning, se corroboró la viabilidad de implementar una aplicación Web con el propósito de facilitar las predicciones para el usuario final, el marco Web utilizó el paquete Flask de Python, esta herramienta brinda una serie de utilidades que facilita la construcción de páginas Web y permite que el usuario final interactúe con el modelo. En relación con la selección del algoritmo Gradient Boosting como modelo final ajustado con la totalidad de los datos, se consideraron dos factores, el primero su excelente desempeño con el conjunto de datos de pruebas y el segundo que su costo computacional y tiempo de entrenamiento del modelo es relativamente bajo, estas condiciones favorecen la actualización continua del modelo con los nuevos datos adquiridos por el sistema de monitoreo de condiciones.

6.2. Recomendaciones

En este estudio proporcionamos diferentes perspectivas para la clasificación y/o predicción de fallas con datos estructurados, en futuros trabajos se podría fortalecer esta investigación abordando las técnicas de monitoreo de condiciones que se basan en procesamiento de imágenes tales como la termografía infrarroja, radiografía, ultrasonido y análisis espectral de vibraciones, utilizando modelos de clasificación soportados en redes neuronales convolucionales para detectar potenciales fallas en los componentes del activo físico.

Con respecto al método estadístico, en este trabajo se logró trazar la curva de supervivencia del activo por medio del estimador no paramétrico de Kaplan y Meier, pero se recomienda un estudio adicional y completar la implementación mediante un análisis de regresión utilizando el modelo de riesgos proporcionales de Cox y los modelos paramétricos de análisis de supervivencia, midiendo la idoneidad para ajustarlos a distribuciones de tipo Weibull o gamma.

Otra posibilidad para un trabajo posterior incluirá desarrollar e integrar la implementación de la aplicación Web para que se considere de producción comercial, estructurando el estilo, contratando un servidor Web para acceso remoto y desarrollando sistemas de seguridad para mitigar riesgo con el manejo de la información.

A. Anexo: Códigos de programación

Los códigos de programación, se agrupan por carpetas que hacen referencia a los capítulos y secciones de este trabajo, estos se encuentran disponibles para el lector en el enlace https://github.com/alexanderhuertas/proyectodegrado, en este enlace se incluye:

- Cuadernos en formato Jupyter Notebook empleando lenguaje Python, en los cuales se desarrollan los modelos de Machine Learning y redes neuronales profundas del capitulo 3.

- Carpeta con archivos en lenguaje Python, Jupyter Notebook, HTML y CSS para ejecutar la aplicación Web que se desarrolla en el capitulo 4.

- Código en RStudio con el análisis de supervivencia que se desarrolla en el capitulo 5.

Referencias

Aggarwal, C. (2018), *Neural Networks and Deep Learning*, Springer International.

Amendola, L. (2014), *Gestión integral de activos físicos*, Ediciones PMM institute for learning, Valencia.

Amruthnath, N. & Gupta, T. (2018), 'Fault class prediction in unsupervised learning using model-based clustering approach', *2018 International Conference on Information and Computer Technologies (ICICT)* pp. 5–12.

Armes, T. & Refern, M. (2013), 'Using big data and predictive machine learning in aerospace test environments', *IEEE AUTOTESTCON* .

Babu, P., Zhao, P. & Li, L. (2016), 'Deep convolutional neural network based regression approach for estimation of remaining useful life', *Database Systems for Advanced Applications* **9642**, 214–228.

Bahtiar, E., Nugroho, N., Hermawan, D., Wirawan, W. & Khuschandra. (2018), 'Triangle bracing system to reduce the vibration level of cooling tower - case study in pt star energy geothermal (wayang windu) ltd – indonesia', *Case Studies in Construction Materials* **8**, 248–257.

Bedell, Z. (2018), 'Support vector machines explained'. `https://medium.com/@zachary.bedell/support-vector-machines-explained-73f4ec363f13`,Web; accedido el 02-08-2020.

Bhandari, N. (2018), 'Extratrees classifier'. `https://medium.com/@namanbhandari/extratreesclassifier-8e7fc0502c7`, Web; accedido el 02-08-2020.

Borges, P. (2005), 'Análisis de supervivencia de pacientes con diálisis peritoneal', *Revista Colombiana de Estadística* **28(2)**, 243–259.

Branco, P., Torgo, L. & Ribeiro, R. (2015), A survey of predictive modelling under imbalanced distributions, Technical report, Faculdade de Ciencias - Universidade do Porto.

Brik, B., Bettayeb, B., Sahnoun, M. & Duval, F. (2019), 'Towards predicting system disruption in industry 4.0: Machine learning-based approach', *Procedia Computer Science* **151**, 667–674.

Brownlee, J. (2017), *Long Short-Term Memory Networks With Python: Develop Sequence Prediction Models With Deep Learning*, Machine Learning Mastery.

Brownlee, J. (2019), *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*, Machine Learning Mastery.

Brownlee, J. (2020*a*), *Imbalanced Classification with Python: Choose Better Metrics, Balance Skewed Classes, and Apply Cost-Sensitive Learning*, Machine Learning Mastery.

Brownlee, J. (2020*b*), *XGBoost With Python: Gradient Boosted Trees with XGBoost and Scikit-learn*, Machine Learning Mastery.

Caplice, C. (2017), Ctl.sc1x supply chain fundamentals v5.1, Technical report, MIT Center for Transportation Logistics, MITx MicroMasters in Supply Chain Management.

Catanzarite, J. (2018), 'The naive bayes classifier'. `https://towardsdatascience.com/the-naive-bayes-classifier-e92ea9f47523`, Web; accedido el 02-08-2020.

Chavez, G. (2019), 'Understanding logistic regression step by step'. `https://towardsdatascience.com/understanding-logistic-regression-step-by-step-704a78be7e0a`, Web; accedido el 01-08-2020.

Chawla, V., Bowyer, W., Hall, O. & Kegelmeyer, P. (2002), 'Smote: Synthetic minority over-sampling technique', *JAIR* **16**, 321–357.

Chen, C., Liu, Y., Sun, X., Cairano, C. & Titmus, S. (2019), 'Automobile maintenance prediction using deep learning with gis data', *Procedia CIRP* **81**, 447–452.

Chen, T. & Guestrin, C. (2016), 'Xgboost: A scalable tree boosting system', *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* pp. 785–794.

Dong, D., Li, X. & Sun, F. (2017), 'Life prediction of jet engines based on lstm-recurrent neural networks', *IEEE Prognostics and System Health Management Conference* .

Dorffner, G. (1996), Neural network for time series processing, Technical report, University of Vienna and Autrian research institute for artificial intelligence.

Ellefsen, A., Bjorlykhaug, E., Aesoy, V., Ushakov, S. & Zhang, H. (2019), 'Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture', *Reliability Engineering and System Safety* **183**, 240–251.

Fagbamigbe, A., Akintayo, A., Oshodi, O., Makinde, F., Babalola, M., Araoye, E., Enabor, O. & Dairo, M. (2020), 'Survival analysis and prognostic factors of time to first domestic violence after marriage among nigeria, kenya, and mozambique women', *Public Health* **181**, 122–134.

Fernández, S. (1995), Análisis de supervivencia, Technical report, Unidad de Epidemiología Clínica y Bioestadística. Complexo Hospitalario-Universitario Juan Canalejo, Cad Aten Primaria, 2:130-135.

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep learning*, Massachusetts Institute of Technology, London.

Gunawan, W., Suhartono, D., Purnomo, F. & Ongko, A. (2018), 'Named-entity recognition for indonesian language using bidirectional lstm-cnn', *Procedia Computer Science* **135**, 425–432.

Hasegawa, T., Saeki, M., Ogawa, T. & Nakano, T. (2019), 'Vibration-based fault detection for flywheel condition monitoring', *Procedia Structural Integrity* **17**, 487–494.

Hermans, M. & Schrauwen, B. (2013), Training and analyzing deep recurrent neural networks, Technical report, Ghent University.

Hernández, A. (2010), Análisis estadístico de datos de tiempos de fallo en r, Master's thesis, Universidad de Granada.

Hinton, G., Srivastava, N., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2012), Improving neural networks by preventing co-adaptation of feature detectors, Technical report, Department of Computer Science, University of Toronto.

Huang, H. & Baddour, N. (2019), 'Bearing vibration data under time-varying rotational speed conditions'. `https://data.mendeley.com/datasets/v43hmbwxpm/2`, Web-database; accedido el 12-12-2019.

ISO14224 (2016), 'Petroleum, petrochemical and natural gas industries - reliability and maintenance data for equipment'.

Jain, R. (2017), 'Decision tree. it begins here'. `https://medium.com/@rishabhjain_22692/decision-trees-it-begins-here-93ff54ef134`,Web; accedido el 02-08-2020.

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2017), *An Introduction to Statistical Learning: with Applications in R*, Springer Texts in Statistics,Corr. 7th., Book 103.

Kaplan, E. & Meier, P. (1958), 'Nonparametric estimation from incomplete observations', *J. Am. Stat. Assoc.* **53(282)**, 457–481.

Kassambara, A. (2018), *Machine Learning Essentials: Guía práctica en R*, Edición Kindle.

Khandelwal, R. (2018), 'K-nearest neighbors(knn)'. `https://medium.com/datadriveninvestor/k-nearest-neighbors-knn-7b4bd0128da7`, Web; accedido el 02-08-2020.

Killeen, P., Ding, B., Kiringa, I. & Yeap, T. (2019), 'Iot-based predictive maintenance for fleet management', *Procedia Computer Science* **151**, 607–613.

Koehrsen, W. (2018), 'An implementation and explanation of the random forest in python'. `https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76`,Web; accedido el 02-08-2020.

Kraus, M. & Feuerriegel, S. (2019), 'Forecasting remaining useful life: Interpretable deep learning approach via variational bayesian inferences', *Decision Support Systems* .

Krawczyk, B. (2016), 'Learning from imbalanced data: open challenges and future directions', *Prog Artif Intell* **5**, 221–232.

Kuhn, M. & Johnson, K. (2013), *Applied Predictive Modeling*, Springer, New York.

Kurama, V. (2020), 'Gradient boosting in classification: Not a black box anymore'. `https://blog.paperspace.com/gradient-boosting-for-classification/`, Web; accedido el 01-08-2020.

Lanners, Q. (2019), 'Choosing a scikit-learn linear regression algorithm'. `https://towardsdatascience.com/choosing-a-scikit-learn-linear-regression-algorithm-dd96b48105f5`, Web; accedido el 01-08-2020.

Lee, W., Wu, H., Yun, H., Kim, H., Jun, M. & Sutherland, J. (2019), 'Predictive maintenance of machine tool systems using artificial intelligence techniques applied to machine condition data', *Procedia CIRP* **80**, 506–511.

Lindholm, A., Wahlstrom, N., Lindsten, F. & Schon, T. (2019), Supervised machine learning lecture notes for the statistical machine learning course, Technical report, Uppsala University.

Malhotra, P., Vig, L., Shroff, G. & Agarwal, P. (2015), 'Long short term memory networks for anomaly detection in time series', *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning.* pp. 89–94.

Montoya, L. (2011), Comparación de dos modelos de regresión en fiabilidad, Master's thesis, Universidad de Granada.

Moore, D. (2016), *Applied Survival Analysis Using R*, pringer International Publishing, Edición de Kindle.

Mora, A. (2009), *Mantenimiento planeación, ejecución y control*, Alfaomega, Ciudad de México.

NASA (2008), 'Turbofan engine degradation simulation data set'. https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/, Web-database; accedido el 10-11-2019.

Olson, R., La Cava, W., Mustahsan, Z., Varik, A. & Moorey, J. (2018), Data-driven advice for applying machine learning to bioinformatics problems, Technical report, Institute for Biomedical Informatics, University of Pennsylvania.

Patel, A. (2018), 'Predictive maintenance using machine learning microsoft casestudy'. https://github.com/ashishpatel26/Predictive_Maintenance_using_Machine-Learning_Microsoft_Casestudy/tree/master/data, Web-database; accedido el 30-07-2020.

Phi, M. (2018), 'Illustrated guide to lstm's and gru's: A step by step explanation'. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21, Web; accedido el 01-07-2020.

Pinto, R. & Cerquitelli, T. (2019), 'Robot fault detection and remaining life estimation for predictive maintenance', *Procedia Computer Science* **151**, 709–716.

Rocca, J. (2019), 'Ensemble methods: bagging, boosting and stacking'. https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205, Web; accedido el 02-08-2020.

Rodins, E. & Amin, M. (1992), 'Maneuver prediction in air combat via artificial neural networks', *Computers Mathematics with Applications* **24**, 95–112.

Sagar, A. (2019), 'How to easily deploy machine learning models using flask'. https://towardsdatascience.com/how-to-easily-deploy-machine-learning-models-using-flask-b95af8fe34d4, Web; accedido el 15-02-2020.

Sak, H., Senior, A. & Beaufays, F. (2014), 'Long short-term memory recurrent neural network architectures for large scale acoustic modeling', *INTERSPEECH* pp. 338–342.

Sapountzoglou, N., Lago, J. & Raison, B. (2020), 'Fault diagnosis in low voltage smart distribution grids using gradient boosting tres', *Electric Power Systems Research* **182**, 1–12.

Saxena, A., Goebel, K., Simon, D. & Eklund, N. (2008), 'Damage propagation modeling for aircraft engine run-to-failure simulation', *IEE International Conference on Prognostics and Health Management* .

Scheu, M., Kolios, A., Fischer, T. & Brennan, F. (2017), 'Influence of statistical uncertainty of component reliability estimations on offshore wind farm availability', *Reliability Engineering and System Safety* **168**, 28–39.

Sharma, S., Malik, H. & Khatri, A. (2015), 'External fault classification experienced by three-phase induction motor based on multi-class elm', *Procedia Computer Science* **70**, 814–820.

Smolyakov, V. (2017), 'Ensemble learning to improve machine learning results'. `https://blog.statsbot.co/ensemble-learning-d1dcd548e936`, Web; accedido el 16-02-2020.

Suarez, S., Marcos, M., Peralta, M. & Aguayo, F. (2017), 'The challenge of integrating industry 4.0 in the degree of mechanical engineering', *Procedia Manufacturing* **13**, 1229–1236.

Swamidass, P. (2000), 'Encyclopedia of production and manufacturing management'. `https://doi.org/10.1007/1-4020-0612-8_580`, Web; accedido el 30-07-2020.

Swapna, G., Soman, K. & Vinayakumar, R. (2018), 'Automated detection of cardiac arrhythmia using deep learning techniques', *Procedia Computer Science* **132**, 1192–1201.

Tang, J. (2018), *Intelligent Mobile Projects with TensorFlow*, Packt Publishing Ltd, Birmingham.

Trifa, A., Sbai, A. & Chaari, W. (2017), 'Enhancing assessment of personalized multi-agent system through convlstm', *Procedia Computer Science* **112**, 249–259.

Téllez, C. & Morales, M. (2016), *Modelos estadísticos lineales. Con aplicaciones en R*, Ediciones de la U, Bogotá.

Uz, F. (2017), 'Deep learning for predictive maintenance with long short term memory networks'. `https://github.com/Azure/lstms_for_predictive_maintenance`, Web; accedido el 25-03-2020.

VanderPlas, J. (2016), *Python Data Science Handbook*, O'Reilly Media, Inc, US.

Vardon, P. (2018), Prédiciton de la panne d'une turbine – nasa, Technical report, Ecole Polytechnique.

Wu, Y., Yuan, M., Dong, S., Lin, L. & Liu, Y. (2018), 'Remaining useful life estimation of engineered systems using vanilla lstm neural networks', *Neurocomputing* **275**, 167–179.

Yang, j., Nguyen, M., San, P., Li, X. & Krishnaswamy, S. (2015), 'Deep convolutional neural networks on multichannel time series for human activity recognition', *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence* pp. 3995–4001.

Zhang, J., Wang, P., Yan, R. & Gao, R. (2018), 'Deep learning for improved system remaining life prediction', *Procedia CIRP* **72**, 1033–1038.

Zhong, R., Xu, X., Klotz, E. & Newman, S. (2017), 'Intelligent manufacturing in the context of industry 4.0: A review', *Engineering* **3**, 616–630.