Universidad Autónoma de Querétaro

Facultad de Ingeniería

Maestría en Ciencias en Inteligencia Artificial

Pronóstico de Vida Útil Remanente de Sistemas Complejos basado en Meta-Aprendizaje

Tesis

Que como parte de los requisitos para obtener el grado de Maestro en Ciencias en Inteligencia Artificial

Presenta:

Ing. Mario Alberto Alberto Olivares

Dirigido por:

Dr. Arturo González Gutiérrez

SINODALES

Dr. Arturo González Gutiérrez
Presidente

Dr. Saúl Tovar Arriaga
Secretario

Dr. Efrén Gorrostieta Hurtado
Vocal

Dr. Roberto Augusto Gómez Loenzo
Suplente

Dr. Alberto Pastrana Palma
Suplente

Centro Universitario Querétaro, Qro
Febrero 2020
México.

A mi familia y en especial a mi madre, la persona que más me ha apoyado y que más ha creído en mi. Gracias por todo tu esfuerzo. Eres mi inspiración para todo lo que hago.

Reconocimientos

Se agradece al Consejo Nacional de Ciencia y Tecnología (CONACYT) por proveer los fondos para esta investigación y mi formación como maestro. Asimismo se agradece a la Universidad Autónoma de Querétaro y a su Dirección de Investigación y Posgrado de la Faculta de Ingeniería por su valiosos apoyo y proveer las facilidades para mi desarrollo como estudiante de posgrado. A los profesores que conforman a la Maestría en Ciencias en Inteligencia Artificial por su compromiso a la educación y todas sus valiosas lecciones. Por último, un especial agradecimiento al Dr. Arturo González Gutiérrez por la dirección de esta tesis.

Resumen

Pronóstico de Gestión del Estado es introducido como un campo de investigación que conecta el estudio de los modos de falla y el Mantenimiento Basado en la Condición para predecir cuándo y cómo un producto es más propenso a fallar durante su vida útil y así prevenirlo. Así, se han desarrollado muchos modelos de pronóstico utilizando técnicas de Aprendizaje Automático, ya que son preferidas debido a su capacidad para repeler el ruido y encontrar patrones ocultos. Más recientemente, los modelos de pronóstico del estado del arte se basan en Redes Neuronales Profundas debido a su capacidad para aprender de datos abstractos difíciles. Sin embargo, la mayor parte de la investigación existente aún no se ha puesto en práctica debido a algunas restricciones, particularmente en la implementación de hardware. En este proyecto se propone una red neuronal Perceptrón Milticapa, ya que es menos costosa que las arquitecturas profundas, desde el punto de vista computacional. Para mejorar los resultados de nuestra red neuronal se hace una búsqueda profunda de los hiperparámetros del modelo, utilizando una metodología tradicional de ajuste de modelos y también una técnica de Meta-aprendizaje llamada Neuroevolución. Adicionalmente, se integra un Filtro de Kalman como una técnica de posprocesamiento de bajo costo computacional. El modelo es probado usando el caso de estudio del pronóstico de vida útil remanente para turbinas de aeronaves. Los resultados indican que nuestro modelo supera a varias propuestas de la literatura basadas en Aprendizaje Profundo. (**Palabras Clave:** Redes Neuronales, Neuroevolución, Predicción, Turbofan)

Abstract

Prognosis Health Management was introduced as a field of research that connects the study of failure modes and Condition Based Maintenance to predict when and how a product is more prone to fail during its useful life. Using this concept many prognosis models have been developed using Machine Learning techniques, since they are preferred because of its ability to repeal noise and find hidden patters. More recently, the state-of-the-art prognosis models are based on Deep Neural Networks due to its capacity to learn from difficult abstract data. However, most of the existing research has not been yet put into practice due to some restrictions, particularly in the implementation of hardware, e.g., real-time embedded system. In this project a Multilayer Perceptron is proposed since it is less computationally expensive than deep architectures. In order to improve the results of this neural network, an extensive research of hyperparameters is conducted using a traditional methodology and a Meta-learning technique called Neuroevolution. Additionally, a Kalman Filter is added as a low-cost postprocessing technique to improve the predictions made by the neural network. The model is tested using the case-study of the remaining usuful life prognosis for aircraft turbines. The results indicate that our model outperforms several proposals from literature based on Deep Learning.

Índice general

Índice de Figuras

Índice de Tablas

Lista de Acrónimos

CAPÍTULO 1

Introducción

"Sólo podemos ver poco del futuro, pero lo suficiente para darnos cuenta de que hay mucho que hacer"

Alan Mathison Turing

1.1. Justificación

Con el rápido desarrollo tecnológico, los sistemas de ingeniería son cada vez más complejos, especialmente en las áreas de electrónica, energía nuclear, automotriz, y aeroespacial (Niu y cols., 2010). A medida que su complejidad aumenta, los aspectos de calidad y confiabilidad son mayormente requeridos para asegurar su funcionamiento eficiente. Aunque un buen diseño es esencial para obtener una gran confiabilidad, el deterioro debido al tiempo, uso y condiciones de trabajo, impactan directamente en su desempeño, lo que implica programas de mantenimiento más eficientes que mantengan la disponibilidad y confiabilidad de estos sistemas (Jardine y cols., 2006). No obstante, aunque la industria invierte billones de dolares cada año en mantenimiento, se presentan fallas inesperadas y catastróficas (ver Tabla 1.1) (Hu y cols., 2019).

Debido al costo en términos de vidas humanas y materiales asociado a

Tabla 1.1: Fallas catastróficas de sistemas y consecuencias (Hu y cols., 2019)

Lugar	Año	Descripción	Consecuencias
Chernobyl, Rusia	1986	Explosión nuclear ocasionado debido a una simulación de corte de suministro eléctrico	Muertes, Contaminación nuclear
Japón y USA	2013	Incendio de baterías de Litio en dos aeronaves Boeing 787	Pérdidas económicas del manufacturero

estas fallas, ha atraído la atención tanto de investigadores como desarrolladores de herramientas para la gestión de la mejora de la confiabilidad y seguridad de los sistemas de ingeniería.

Los modelos matemáticos basados en el comportamiento físico de las fallas de los sistemas han resultado ser precisos en su predicción. Sin embargo, los sistemas complejos de ingeniería resultan difíciles de modelar matemáticamente, lo cual hace imposible una predicción de fallas por este medio, ya que estos sistemas están sujetos a procesos de degradación múltiple (internos, externos, dependientes o independientes). Esto impone grandes retos debido a su complejidad matemática (Alaswad y Xiang, 2017).

1.2. Descripción del Problema

Aunque desde el 2010 la investigación y diseño de algoritmos de pronóstico han sido introducidas y varios algoritmos han sido propuestos, muchos de estos son más específicos a una aplicación más que a una herramienta genérica, y no existe hasta el momento algún algoritmo que logre desempeñarce mejor que los demás en todas las posibles aplicaciones (Peng y cols., 2010)(Khan y Yairi, 2018).

Las técnicas de Aprendizaje Automático han sido las preferidas para el desarrollo de modelos de pronóstico de vida útil remanente en sistemas complejos, esto debido a su capacidad para evadir el ruido y encontrar patrones ocultos. Recientemente, los modelos del estado del arte se basan en redes neuronales profundas, puesto que éstas han demostrado excelentes resultados para extraer información de datos abstractos multidimensionales. Por otro lado, mucha de la investigación hecha no ha sido implementada debido a la complejidad de estos modelos y a ciertas restricciones de hardware, en específico, en sistemas embebidos en tiempo real.

En este proyecto de tesis se plantea el problema consistente en determinar de manera eficiente y con la mayor precisión posible la vida útil remanente de sistemas complejos, dado un conjunto de indicadores del estado del sistema y una base datos del comportamiento del mismo en operación.

1.3. Hipótesis

Un modelo de bajo costo computacional es tan eficiente como una Red Neuronal Profunda para el pronóstico de vida útil remanente de sistemas complejos, cuando se configuran sus hiperparámetros de manera apropiada.

1.3.1. Objetivo General

Desarrollo de un modelo de bajo costo computacional para el pronóstico de vida útil remanente de sistemas complejos mediante el uso de Redes Neuronales Perceptrón Multicapa, Meta-Aprendizaje y Filtro de Kalman.

1.3.2. Objetivos Específicos

- Identificar los modelos del estado del arte para pronóstico de vida útil remanente en turbofan
- Proponer un modelo basado en una arquitectura de Red Neuronal Perceptrón Multicapa con menos parámetros que los del estado del arte
- Determinar los mejores hiperparámetros del modelo
- Desarrollar una metodología para la búsqueda de hiperparámetros de forma manual
- Implementar un algoritmo evolutivo para la búsqueda de los hiperparámetros del modelo
- Comparar los resultados de la búsqueda de hiperparámetros manual contra los resultados del algoritmo evolutivo
- Validar el modelo comparando los resultados con los del estado del arte

CAPÍTULO 2

Antecedentes

"Un sutil pensamiento erróneo puede dar lugar a una indagación fructífera que revela verdades de gran valor"

Isaac Asimov

2.1. Inicios del Mantenimiento

Junto con el nacimiento de la Revolución Industrial, el concepto de confiabilidad se convierte en un aspecto importante en la evaluación de productos y equipos industriales. El buen diseño de productos es esencial para obtener una gran confiabilidad, sin embargo, no importa que tan bueno es el diseño, estos se deterioran con el tiempo debido a las condiciones de trabajo y medio ambiente. Así, el mantenimiento surge como una tarea indispensable para lograr mantener o restaurar las condiciones de estos productos y asegurar satisfactoriamente los niveles de confianza del producto durante su vida útil (Jardine y cols., 2006). El capital y competencia de las compañías dependen de la confiabilidad, disponibilidad, seguridad, efectividad de los productos y/o equipos. Así, en el siglo XXI se establece que el mantenimiento debe ser una parte integral de la estratégica de productividad para el éxito de una organización (Dhillon, 2002). A lo largo de las décadas, las operaciones de mantenimiento han evolucionado con el crecimien-

to de la tecnología. Las estrategias de mantenimiento pueden ser generalmente clasificadas en dos categorías: Mantenimiento Correctivo (MC) y Mantenimiento Preventivo (MP) (Duffuaa y cols., 2001). El primero, es una estrategia que es usada para restaurar, reparar o remplazar equipo a su función requerida antes de que falle (Tsang, 1995). El último realiza actividades de mantenimiento previo a que la falla del equipo ocurra (Löfsten, 1999). La estrategia de MC lleva a paros de los sistemas de producción y costos altos debido a fallas inesperadas. Por otro lado, MP busca reducir la ocurrencia de fallas contribuyendo a minimizar el costo de reparación y paros de los sistemas, lo que conlleva al incremento de la calidad de los productos y servicios de las compañías

En la industria, la ejecución de MP puede ser generalmente realizada atreves de ya sea la experiencia o las recomendaciones del manufacturero original (Ahmad y Kamaruddin, 2012b). MP toma la forma de una revisión basada en un lapso, por lo cual se le refiere como Mantenimiento Basado en Tiempo (MBT), y este tiempo es típicamente determinado en modelos probabilísticos de fallo del sistema (Alaswad y Xiang, 2017). Con el rapido desarrollo tecnológico los productos son cada vez más complejos, y por consecuente una calidad y confiabilidad mayor son requeridas. MP no es usualmente aplicable cuando se trata de minimizar los gastos de operación y maximizar el desempeño de los productos (Ahmad y Kamaruddin, 2012b). De hecho, eventualmente el MP se ha convertido en los

mayores gastos de muchas compañías (Jardine y cols., 2006). Se puede determinar básicamente 3 razones de lo anterior (Labib, 2004): la primera, cada sistema trabaja en diferentes condiciones y por lo tanto requiere diferentes horarios de MP; segundo, los diseñadores usualmente no tienen experiencia de las fallas del sistema y tienen menor conocimiento de su prevención comparado

con los operadores y el personal de mantenimiento; la tercera, las compañías que se encargan de realizar los sistemas ocultan cierta información. Por lo tanto, sistemas de mantenimiento más eficientes son necesarios, especialmente en las áreas de electrónica, energía nuclear, automotriz, construcción naval y aeroespacial (Niu y cols., 2010).

2.2. Mantenimiento Basado en la Condición

El problema del mantenimiento basado en intervalos de tiempo fue presentado formalmente en (Neale y Woodley, 1975) donde se señala que el problema del MP radica en que no se conoce el intervalo de tiempo optimo para realizar el mantenimiento debido a que no todos lo sistemas se encuentran bajo las mismas condiciones y por lo tanto pensar en un tiempo constante en los intervalos de tiempo antes del mantenimiento no es la solución. De hecho, para el intervalo de 1958 a 1987 se conoce en la industria petroquímica numerosos accidentes alrededor del mundo debido al pobre mantenimiento de sus sistemas, entre los mayores incidentes figuraban dos casos del año 1977 ocurridos en México (Davies, 2012). Así entonces en 1975 es introducida una nueva estrategia de mantenimiento denominada Mantenimiento Basado en la Condición (MBC) para maximizar la eficiencia de la toma de decisiones del MP (Ahmad y Kamaruddin, 2012a). A partir de entonces MP fue dividido en dos categorías: La primera es el mantenimiento tradicional basado en intervalos de tiempo denominado como MBT, y por último el MBC. De acuerdo con (Jardine y cols., 2006) MBC es un mantenimiento que recomienda evadir mantenimiento innecesario a través de acciones únicamente cuando hay evidencia de condiciones y/o comportamiento anormal del sistema. Por ejemplo, los motores de los vehículos tienen interva-

los de cambio de aceite recomendados por el fabricante. Estos intervalos son basados en un promedio o buena aproximación en lugar de la condición actual del aceite. La idea detrás de MBC es remplazar el aceite solamente cuando se requiere y no cada intervalo de tiempo como se sugiere. Los beneficios económicos son percibidos fácilmente al evadir operaciones innecesarias. El corazón del MBC es el monitoreo del estado o condición del sistema, donde señales son continuamente monitoreadas usando ciertos tipos de sensores u otros indicadores apropiados. Así el mantenimiento es realizado “solo cuando es necesario”.

Las condiciones para el MBC se resumen a las siguientes (Niu, 2017):

- Ya sea que la prevención de la falla no es posible, o la forma en que se llega a dicha falla es desconocida de forma que pareciera que ocurre de forma aleatoria
- Existe una medida de correlación con la falla (vibración).
- Se ha identificado algún valor que indica que la falla ocurrirá si se sobrepasa.

Esta estrategia tiene buenos resultados siempre y cuando se tiene al personal entrenado para tomarse el tiempo de realizar las actividades de mantenimiento.

2.3. Mantenimiento Centrado en la Confiabilidad

El mantenimiento es una disciplina que cambia constantemente y esto debido al gran incremento en el número y variedad de activos o bienes (equipo, infraestructura, etc) en las organizaciones, los diseños complejos, nuevas

técnicas, y el cambio de punto de vista de la organización y responsabilidades de mantenimiento. Así, en 1968 se desarrolló una nueva filosofía conocida como Mantenimiento Centrado en la Confiabilidad (MCC). Esta idea llego con el Boeing 747 a los Estados Unidos, un avión de cuerpo grande. Las aerolíneas americanas al darse cuenta de que su mantenimiento costaría más de lo común declaraban al modelo 747 económicamente no rentable. United Airlines y Boeing trabajaron en desarrollar un nuevo proceso de toma de decisión para determinar qué tipo de mantenimientos era requerido para cada activo, dada la física de cada falla, buscando mantener la navegabilidad del avión siempre. Como resultado en 1968 se publica el manual de mantenimiento para esta aeronave. En los 70's el documento es generalizado para ser usado en otros aviones, apareciendo oficialmente en éste el término *reliability centered maintenance* (mantenimiento centrado en confiabilidad) (Dhillon, 2002; Sifonte y Reyes-Picknell, 2017).

"El MCC no es una estrategia de mantenimiento, sino un proceso de toma de decisiones para crear estrategias de mantenimiento optimizadas combinando otras estrategias de mantenimiento" (Sifonte y Reyes-Picknell, 2017). En otras palabras, "es una metodología científica para la formulación y optimización de estrategias de mantenimiento" (Yoshikawa y cols., 2014).

El fundamento teórico que soporta esta filosofía fue la conclusión hecha por la investigación de la Aviación Americana y la industria naval, que las fallas de los sistemas debidas a la edad de éste son solo del 8-29 % mientras que aquellas independientemente de su edad eran del 71-92 % (Yoshikawa y cols., 2014). Concluyendo que el MBT no erá la opción más adecuada debido a que no atendía a la mayor cantidad de fallas. Este increíble descubrimiento no solo era pertinente para la industria de la aviación, sino para todas aquellas industrias cuyos sistemas son demasiado complejos como para determinar que sus fallas

pueden ser debidas solo a la edad del mismo, y así es que otras industrias comenzaron a incorporar esta nueva filosofía de mantenimiento y tal es el caso de la industria de la energía nuclear. En 1998 La planta de energía nuclear de Daya Bay introdujo el MCC y hoy es considerada como la planta de energía nuclear más exitosa en implementar lo (Yoshikawa y cols., 2014).

Figura 2.1: Línea del Tiempo del Mantenimiento. Imagen propia

2.4. Futuro del Mantenimiento

Como se puede observar en la Figura 2.1 la evolución del mantenimiento ha sido constante y una consecuencia de los descubrimientos y desarrollos tecnológicos a lo largo de la historia. Y básicamente puede reducirse a tres grandes técnicas, desde el MC seguido por el MBT y finalmente el MBC. Hoy en dia el mantenimiento es idealmente una mezcla de estos tres tipos dependiendo de los objetivos y la parte del proceso que al que se le realiza el mantenimiento, a esto se le llama MCC. En práctica la elección de la estrategia optima de mantenimiento no es tan simple. Y aunque parezca que MBC es el mejor de los tres resulta que no todas las fallas pueden ser detectadas, solo aquellas relacionadas a la variable de medición y la recolección de los datos requeridos puede ser retadora. Aunque los datos de la condición del sistema están siempre disponibles, en la mayoría de los casos es costoso obtenerlos debido al equipo de monitoreo, como sensores y no todas las compañías están dispuestas a gastar en ello (Niu, 2017) (Ahmad y Kamaruddin, 2012a).Por ello en muchos casos una combinación de estas tres técnicas es usada.

Industria 4.0

Dado que el mantenimiento es una consecuencia de los desarrollos tecnológicos, observando las tendencias tecnológicas podemos predecir hacia donde va el futuro del mantenimiento.

En la Figura 2.2 se ilustra la evolución de la industria hacia la llamada Industria 4.0, termino para la cuarta revolución industrial presentado por primera vez en la feria de Hanover en 2011 (Kagermann y cols., 2011).

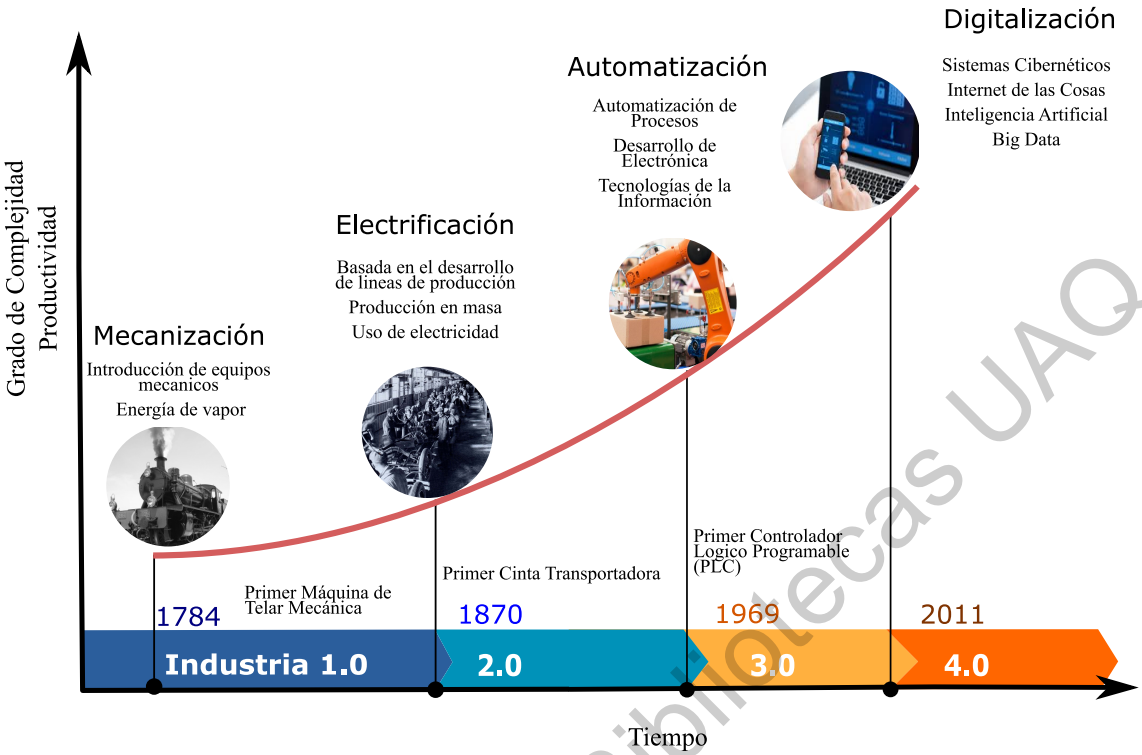


Figura 2.2: Evolución de la Industria (Imagen propia con base en DFKI 2011)

A medida que el mundo digital, físico y humano continúan integrándose, experimentamos una profunda 4ta transformación en la industria, que llega hasta nuestras vidas. La revolución industrial, Internet de las cosas (IoT, por sus siglas en inglés) y *Big Data*, están cambiando la forma en que diseñamos, fabricamos, proporcionamos productos y servicios. Está brindando oportunidades para hacer que los sistemas de producción sean más eficientes y rápidos, y más flexibles y resistentes a las complejas cadenas de suministro y redes de distribución que unen a la economía global (Zio, 2016) (Siemens, 2013).

Observando las tendencias de Tecnologías de la Información, se indica en (Siemens, 2013) que para los próximos 20 años el poder computacional, la capacidad de almacenamiento, y la velocidad de transmisión de datos por microchips se enfrentará a un incremento de miles de veces. Se podría decir el rendimiento

de una computadora portátil del año 2013 cuyo costo era de aproximadamente 500 euros será encontrado en microchips de un costo de 0.5 euros.

Estos avances en la tecnología han abierto las puertas a la practica del MBC (Alaswad y Xiang, 2017). La tecnología de adquisición, la capacidad de almacenamiento, y la capacidad de procesamiento de datos ya son una realidad y su accesibilidad a muchas compañías está cada vez más cerca. Esto ultimo es muy importante debido a que como se ve en la Figura 2.2 la complejidad de los sistemas incrementa también lo que requerirá sistemas confiables y seguros.

Las barreras económicas para realizar está practica se están mitigando con el desarrollo tecnológico y por ello hay un gran esfuerzo en la investigación del MBC al día de hoy. Los datos existen, la pregunta es ¿Cómo sacar la mayor información posible de esos datos para mejorar los sistemas de mantenimiento?.

2.5. Desarrollo de la técnica de Pronóstico y Gestión del Estado

Pronóstico y Diagnóstico

Dos grandes necesidades aparecen con el surgimiento de la filosofía de MBC. Y es cierta para sistemas mecánicos y electrónicos. Estas son el **diagnóstico** y **pronóstico**.

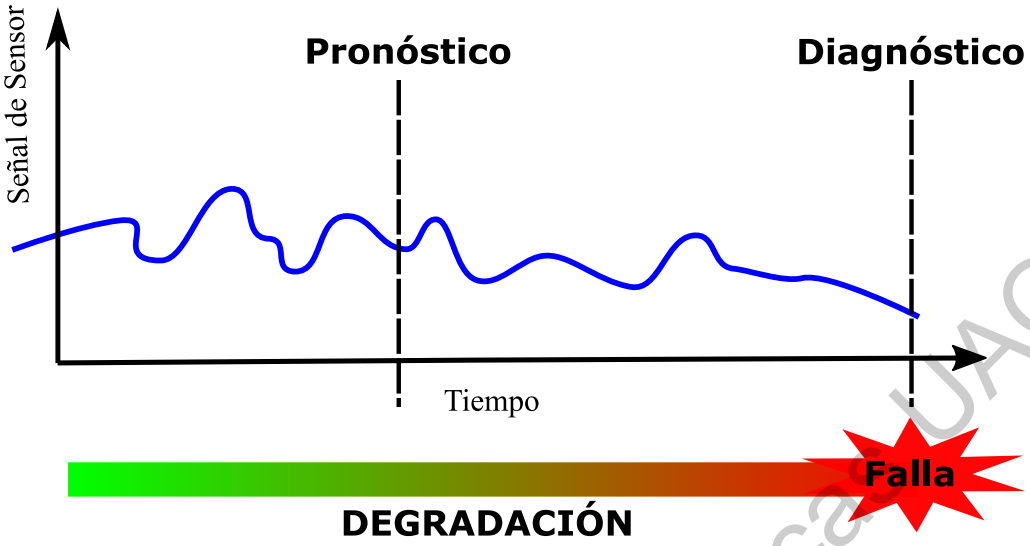


Figura 2.3: Diferencia entre diagnóstico y pronóstico. Imagen propia con información de Lee y cols. (2014)

En la Figura 2.3 se muestra el desarrollo del proceso de degradación de un sistema con respecto al tiempo hasta alcanzar una condición donde éste falla. A partir de esto se pueden conceptualizar los terminos de diagnóstico y pronóstico como (Niu, 2017):

- **Diagnóstico**: Se realiza cuando la falla ya ocurrió y consta de la exploración de relaciones de causa y efecto para aislar la causa raíz de la falla.
- **Pronóstico**: Es la evaluación de la condición actual del sistema para predecir condiciones futuras, lo cual incluye detectar una probable falla futura y el *tiempo de vida útil restante* del sistema para alcanzar dicha falla.

Desde el punto de vista de la investigación, diagnóstico es una area más madura que pronóstico, dado a lo retador que llega ser éste ultimo (Eker y cols., 2012). Sin embargo, en términos prácticos pronóstico es más eficiente que diagnostico para evitar fallas inesperadas, ahorrando costos de paros de produc-

ción y mantenimeinto no planeado (Jardine y cols., 2006) (Ahmad y Kamaruddin, 2012a).

Transformación del Mantenimiento

El termino Pronóstico de la Gestión del Estado (PGE) o *Prognosis Health Management* (por su nombre en inglés) fue introducido en el campo de la medicina como la predicción del curso futuro y resultado del proceso de una enfermedad (Abu-Hanna y Lucas, 2001). Derivado del mismo concepto, varios métodos de pronóstico se han desarrollado para mantenimiento de maquinaria en los últimos 14 años (Lee y cols., 2014), con un mayor enfacis a partir del 2010 (Peng y cols., 2010).Es un campo de investigación y aplicación que utiliza datos en la evaluación del estado de la condición actual del sistema de ingeniería para predecir cuando y como el sistema es más propenso a fallar durante su tiempo de vida útil (Hu y cols., 2019). La predicción del tiempo se convierte entonces en la predicción de la Vida Útil Remanente (VUR), el cual es un concepto importante para la toma de decisiones de mantenimiento.

Además, PGE es la disciplina que conecta el estudio de los modos de fallo a el estudio del ciclo de vida del mantenimiento de los sistemas, y también es conocido como *Systems Health Management* (SHM),o -en aplicaciones de transporte- *Vehicle Health Management* (VHM) o *Engine Health Management* (EHM) (Niu, 2017).

PGE consta de las siguientes facetas:

- Adquisición de Datos: A través de los sensores el sistema de ingeniería se adquieren señales

- Procesamiento: Se extraen las características más relevantes del sistema
- Pronóstico: Se define un umbral para cada tipo de condición anormal y se predice el tiempo de vida restante respecto a dicho umbral
- Gestión: Se habilita la toma de decisiones optima pasado en el tiempo de vida útil restante.

En la Figura 2.4 se observan diferentes estrategias de mantenimiento respecto a la complejidad e incertidumbre que presentan los sistemas. PGE se puede tratar como una forma evolucionada de MBC (Niu, 2017) pero también se ha fundamentado en las bien conocidas metodologías de MP, MCC, y por supuesto MBC (Lee y cols., 2014)

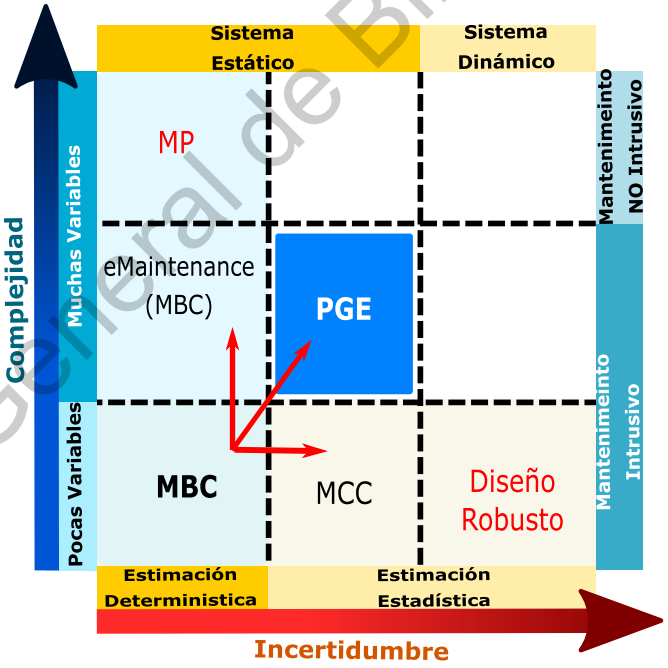


Figura 2.4: Trasformación del Mantenimiento y Futuras Tendencias. Imagen adaptada de Lee y cols. (2014)

Se pueden usar técnicas de MBC para proporcionar datos a los modelos de pronósticos de PGE y respaldar la toma de decisiones precisa y oportuna que

evite el tiempo de inactividad y maximice los beneficios (Niu, 2017). Además, hablando de tendendcias para el futuro del mantenmiento, PGE se considera la base, cuando se complementa con otra técnicas, para áreas avanzadas que incluyen Mantenimiento Autónomo, Sistema Resilientes y Sistemas de Ingeniería Inmunologicos (Lee y cols., 2014) (ver Figura 2.4). PGE es el presente y sus disciplinas necesitan ser desarrolladas más para ayudar a crecer estas áreas avanzadas.

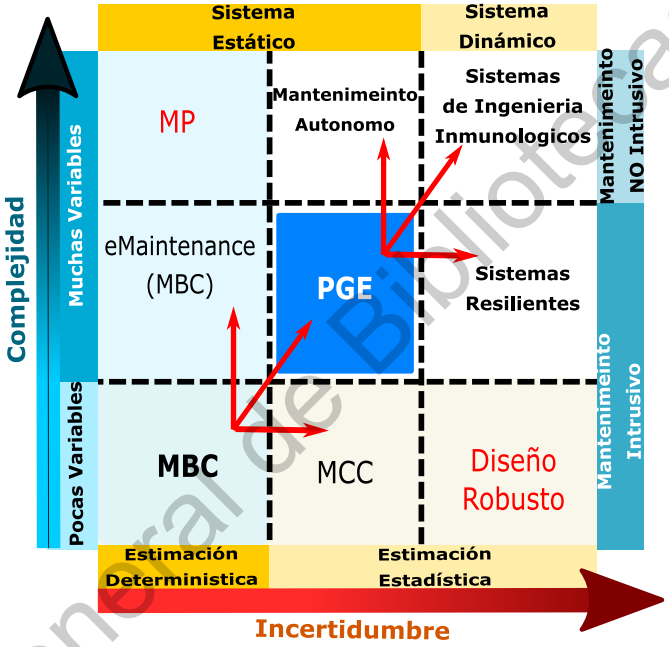


Figura 2.5: Trasformación del Mantenimiento y Futuras Tendencias. Imagen adaptada de Lee y cols. (2014)

2.5.1. Metodologías de PGE

Los enfoques técnicos para la construcción de modelos de predicción en PGE pueden ser categorizados dentro de tres clases.

Modelo Físico-Matemático

Este enfoque intenta incorporar el conocimiento de la física del sistema a la predicción de la VUR. Los modelos pueden ir desde un micro-nivel a uno macro donde se incluyen los sub-sistemas definiendo relaciones entre estos y de alguna manera simplificando su representación. La desventaja de ir de un micro a un macro nivel radica en la reducción en la precisión de predicción. Cuando el sistema es complejo (como la turbina de un avión) el desarrollo del modelo es exhaustivo (Niu, 2017).

La mayor parte de la investigación de modelos matemáticos para PGE presenta las siguientes características (Alaswad y Xiang, 2017):

- La mayoría se enfocan en un solo componente (micro nivel)
- El enfoque a múltiples componentes (macro nivel) no se ha atendido adecuadamente.
- La consideración de procesos degradación múltiple (interna, externa, dependiente e independiente) es un gran reto debido a su complejidad matemática.
- Pocos trabajos referentes a degradación múltiple se han hecho y éstos han considerando principalmente procesos de degradación independientes.

Modelo Basado en Datos

Estas técnicas son apropiadas cuando no se tiene la información necesaria para comprender el comportamiento del sistema o cuando es suficientemente complejo como para desarrollar precisos modelos matemáticos (Liu y cols.,

2009). La principal ventaja de estos métodos son que pueden dar soluciones rápidas y menos costosas, en comparación con otros enfoques, al igual que pueden proveer una cobertura amplia del sistema a comparación de los modelos matemáticos cuyo alcance es corto (Niu, 2017). La idea es utilizar las mediciones de la condición del sistema como presiones, temperaturas, velocidades, vibraciones, corrientes, etc, para crear un modelo que correlaciones estos parámetros y su comportamiento con la degradación del sistema así como la evolución de fallas con el objetivo de predecir la VUR (Elattar y cols., 2016).

La condición para llevar a cabo este enfoque en PGE es la fuente de datos lo bastante grande como para poder entrenar los algoritmos y encontrar los patrones que determinarán las correlaciones (Niu, 2017).

De igual forma las técnicas basadas en datos pueden ser categorizadas en dos grandes grupos como: Técnicas Estadísticas e Inteligencia Artificial. Técnicas de Inteligencia Artificial son preferidas sobre las otras debido a su habilidad de rechazo de ruido en datos y aprendizaje de patrones ocultos en las relaciones entre parámetros (Elattar y cols., 2016).

Modelo Híbrido

Como su nombre lo señala es una combinación del enfoque matemático y el basado en datos (Elattar y cols., 2016). El enfoque de datos puede compensar la falta de conocimiento para el modelo físico-matemático (ajuste de parámetros del modelo por medio de datos), o el modelo físico-matemático puede compensar la falta de datos en el otro enfoque. En realidad es raro encontrar trabajos puramente basados en un solo enfoque siempre se encuentra alguna contribución de una metodología con la otra(Niu, 2017). Esto métodos híbridos pueden

ser categorizados como ‘fusión pre-estimación’ y ‘fusión pos-estimación’ (Niu, 2017)(Elattar y cols., 2016). (Elattar y cols., 2016)

2.6. Inteligencia Artificial

Aunque no se ha logrado consensuar una definición universalmente aceptada para Inteligencia Artificial (IA), Nils J. Nilsson, investigador y fundador de la disciplina de IA propone que la “Inteligencia Artificial es al actividad de desarrollar maquinas inteligentes, entendiéndose por inteligencia la cualidad que permite a una entidad funcionar apropiadamente y con conocimiento de su entorno”. Así mismo en el trabajo reciente de la Universidad de Stanford (Stone y cols., 2016) se señala la IA como ’una ciencia y un conjunto de tecnologías computacionales que son inspiradas por las maneras en que los humanos usan sus sistemas nerviosos y cuerpos para percibir, aprender, razonar y tomar decisiones’.

La IA ha sido un campo de investigación creado a partir de varias disciplinas entre las que destacan: Ciencias de la Computación, Matemáticas, Neurociencia, Biología, Sociología, Psicología, Lingüística y Filosofía.

Hoy en día la IA se ha convertido en una herramienta ampliamente usada por una gran variedad de disciplinas debido a sus grandes capacidades. Sus principales aplicaciones son las siguientes:

- Aprendizaje Automático
- Sistemas Expertos
- Visión por Computadora
- Planeación

- Robótica
- Procesamiento del Lenguaje Natural

Aprendizaje Automático

La habilidad para aprender es una de las características fundamentales de un comportamiento inteligente. Desde la era de la computación, investigadores buscan el como implantar esta capacidad a maquinas. Así el estudio y modelado computacional de los procesos de aprendizaje en sus múltiples manifestaciones es conocido como Aprendizaje Automático (AA) o *Machine Learning* (por su nombre en inglés) (Aigner, 1983). Comenzó con la investigación de maquinas basadas en modelos de neuronas. En 1994 McCullon y Pills proponen el primer modelo matemático formal de una neurona el cual es base para los modelos aún utilizados (ver Figura 2.6).

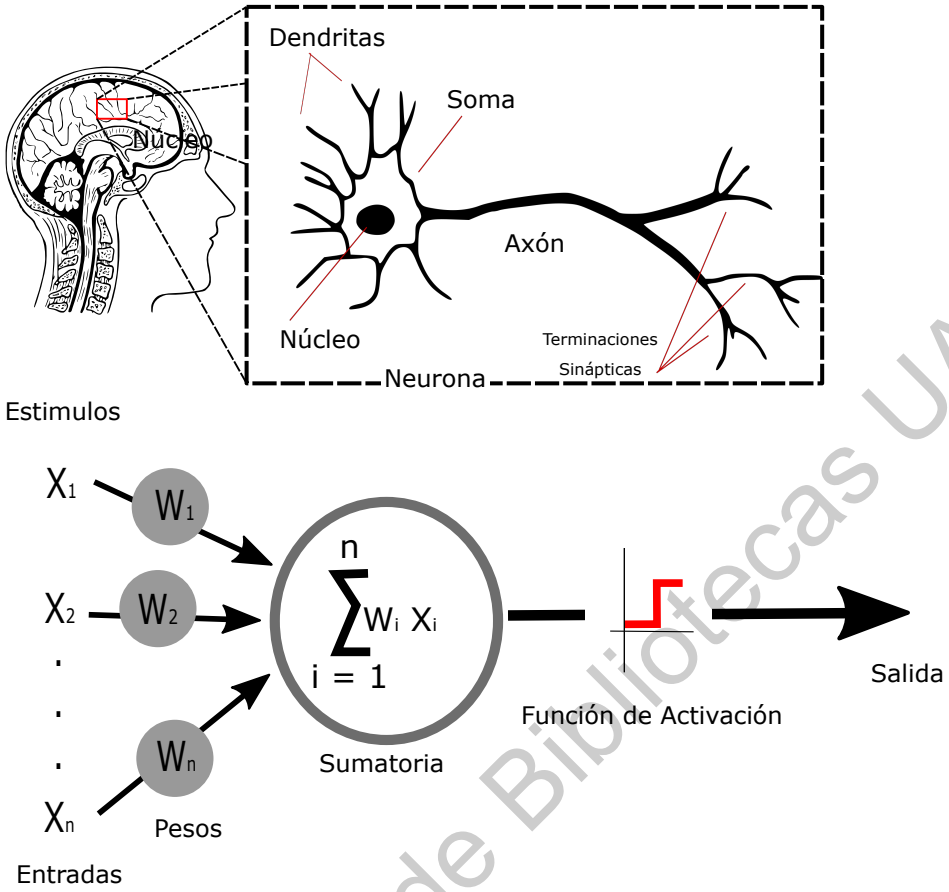


Figura 2.6: Representación de una neurona. Imagen propia

El modelo era ajustado por parámetros llamados pesos más sin embargo no se ajustaban automáticamente.

En 1957 en el Laboratorio de Aeronáutica Cornell, Frank Rosenblatt diseñó varios dispositivos de aprendizaje inspirados en el método de aprendizaje, a estos diseños los llamo “perceptrons” para enfatizar su habilidad de percepción y no lógicas (Dreyfus, 1990). El objetivo de estos dispositivos era aprender a través de ejemplos para distinguir si una entrada pertenecía a un conjunto. Hoy en día a este algoritmo es a lo que se le conoce como “Percetrón”.

Debido a la tecnología primitiva del momento la mayoría de la investigación referente al tema de las neuronas era teórica o requería de hardware es-

pecializado tal como el Perceptrón de Rosenblatt pero dio inicio a la disciplina de Reconocimiento de Patrones (Aigner, 1983). En el año 1960 en Standford se propone un nuevo modelo de neurona llamado ADELINE (de *ADAptative LINear Element*) contaba con una nueva y poderosa forma de aprendizaje que, a diferencia de Perceptrón, todavía se usa ampliamente. Widrow y sus alumnos aplicaron ADALINE con éxito a una gran cantidad de problemas. Para 1969 Minsky y Papert dieron a conocer las debilidades de las redes neuronales y es que básicamente, todas las redes neuronales sufren la misma "falla fatal"que el perceptrón; la incapacidad de ser útil de calcular ciertos predicados esenciales como XOR. Dejaron la impresión de que la investigación de redes neuronales había demostrado ser un callejón sin salida.

En los años 60s ya se hablada la reducción de errores mediante la técnica de gradiente descendiente. Kelley y Bryston desarrollan soluciones de control optimo para trayectorias de cohetes basadas en el método del gradiente (Aigner, 1983). La primer aplicación de la propagación hacia a tras del error o mejor conocido como Retropropagación (RP), utilizando el método del gradiente fue hecha en 1981 por Werbos (Werbos, 1981) , sin embargo no fue hasta 1986 que el articulo por Rumerhart (Rumelhart y cols., 1985) significativamente contribuyo a la popularización de BP para entrenamiento de Redes Neuronales con más de una capa oculta, desarrollando el conocido Perceptrón Multicapa (PMC). A partir de entonces múltiples aplicaciones fueron desarrolladas y nuevas arquitecturas fueron propuestas. En los 90's un nuevo paradigma surge como Agentes Inteligentes como sistemas capaces de reaccionar dependiendo de su entorno (Russell y Norvig, 2016) . En 2006 Hinton y Osindero desarrollan un algoritmo para rápido entrenamiento de redes con más de una capa, surgiendo así el término Aprendizaje Profundo (AP). Muchas nuevas arquitecturas referentes a AP co-

mienzan a surgir, pero no es hasta 2016 que Google DeepMind demuestran que AP es capaz de muchas cosas tras enfrentare al reto de desarrollar una IA capaz de jugar y dominar uno de los juegos más complejos, Go. Su IA llamada AlphaGo basada en DL ganó 4-1 al entonces mejor jugador del mundo de Go, Lee Sedol. La victoria de AlphaGo fue un logro histórico que los expertos acordaron una década por delante de su tiempo (Silver y cols., 2017). En 2018 OpenAI un laboratio de IA sin fines de lucro desarrollan una IA llamada OpenAI-Five, constituida de cinco redes neuronales. Esta IA derrotó a los mejores jugadores de Dota 2 el cual es un un videojuego tipo multijugador de ritmo rápido que plantea un tipo diferente de desafío con respecto con ajedrez o a Go, ya que requiere que las computadoras colaboren y administren la incertidumbre. ”La próxima gran novedad para la inteligencia artificial es la colaboración”, dice Jun Wang, científico informático del University College London que trabaja en StarCraft II, otro juego de estrategia en tiempo real (Hutson, 2018).

Los hitos y el surgimiento de los algoritmos representativos que han forjado esta área de investigación son descritos en las Figuras 2.7 y 2.8.

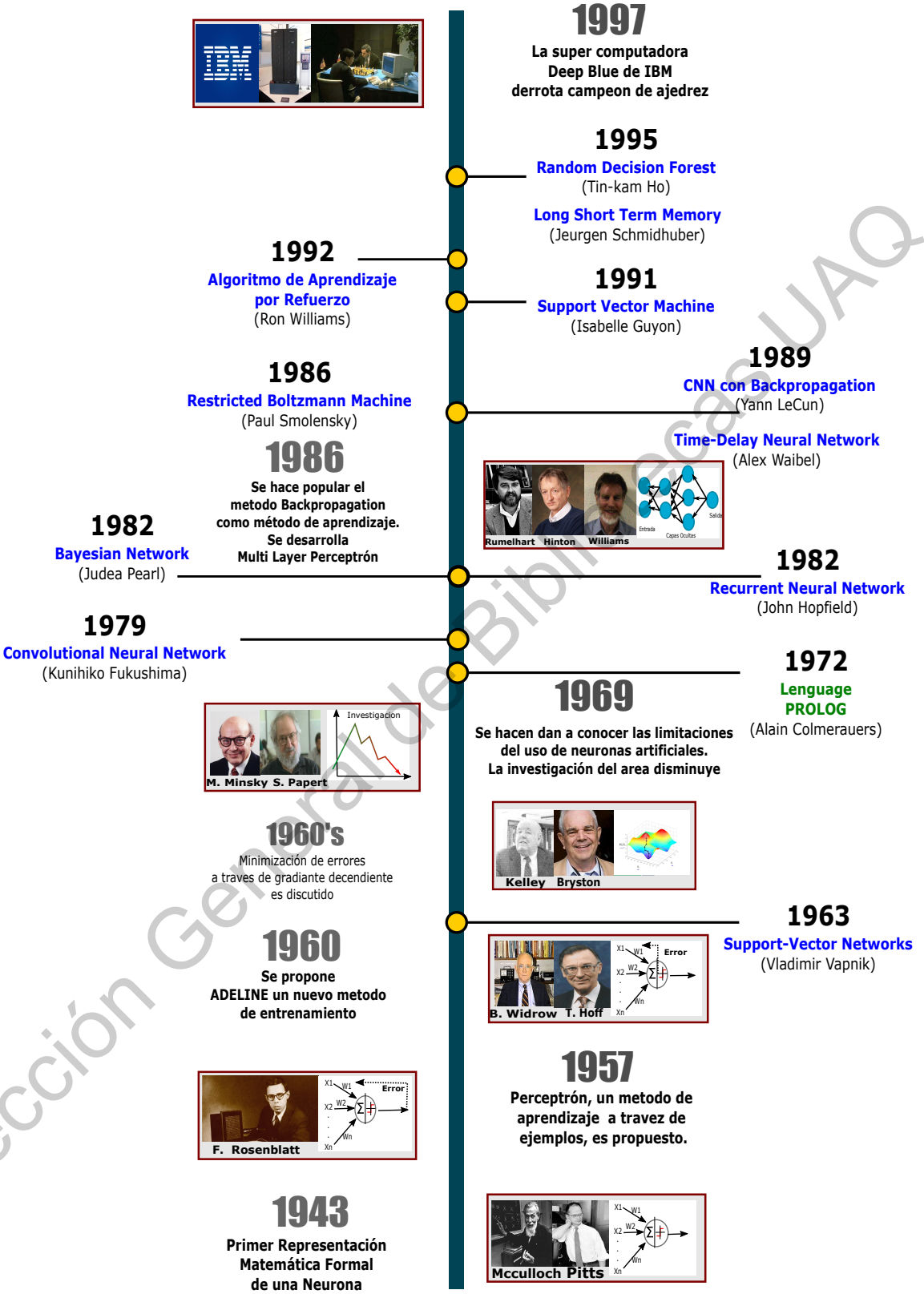


Figura 2.7: Evolución del Aprendizaje Automático (Parte 1). Imagen propia con información de Scaruffi (2016)

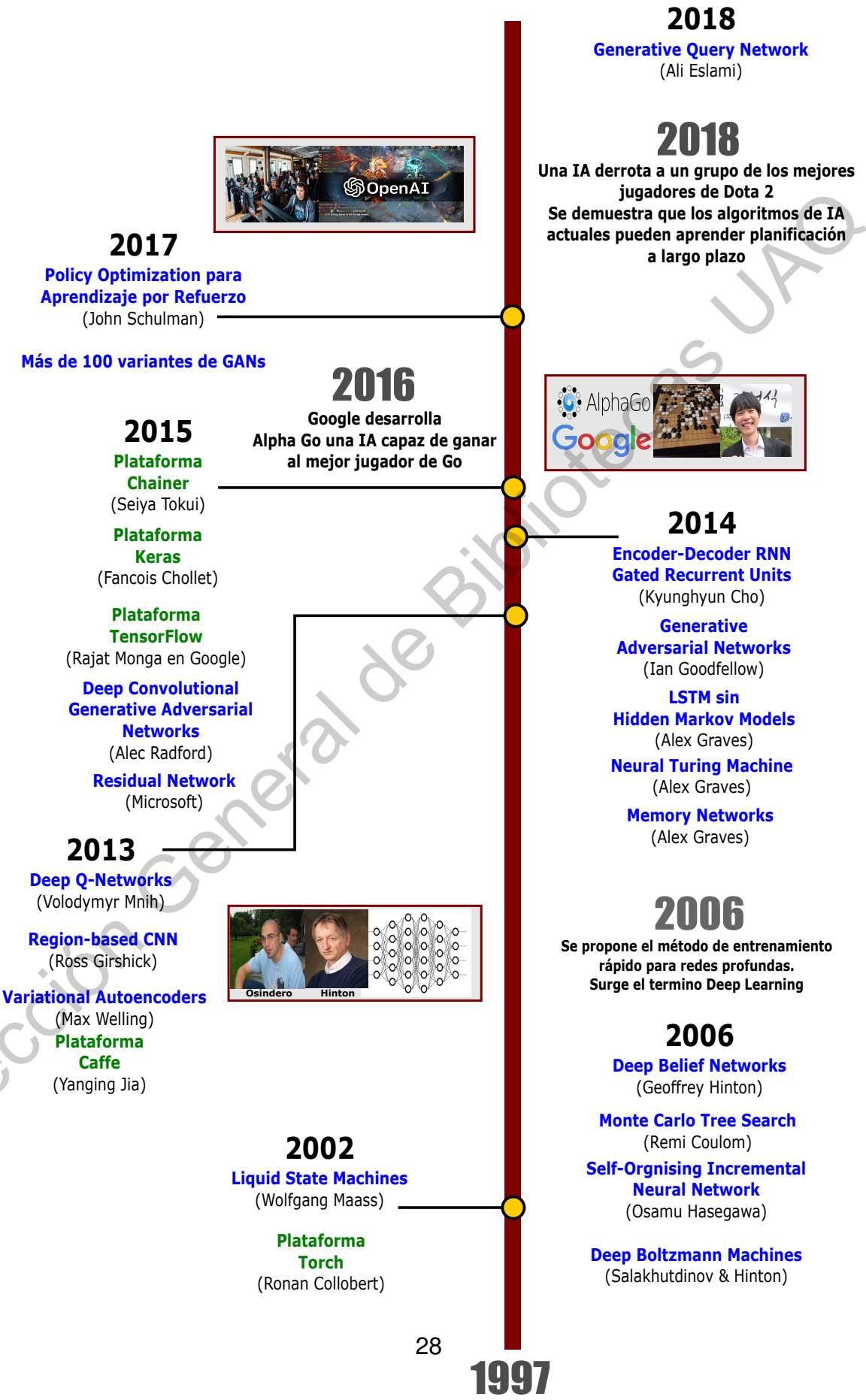


Figura 2.8: Evolución del Aprendizaje Automático (Parte 2). Imagen propia con información de Scaruffi (2016)

Los algoritmos de AA emplean métodos de cálculo para “aprender” información directamente de los datos sin depender de una ecuación predeterminada como modelo. Los algoritmos mejoran su rendimiento de forma adaptativa a medida que aumenta el número de muestras disponibles para el aprendizaje. Por esta razón es una herramienta clave hoy en día para resolver problemas de Big Data como (MathWorks, 2018):

- **Finanzas computacionales:** Calificación crediticia y algoritmos para comercio
- **Procesamiento de imágenes y visión artificial:** Reconocimiento facial, detección de movimiento y detección de objetos
- **Biología computacional:** Detección de tumores, descubrimiento de fármacos y secuenciación del ADN
- **Producción de energía**: Previsión de la carga y el precio
- **Procesamiento del lenguaje natural:** Aplicaciones de reconocimiento de voz y texto
- **Sector de transporte (Aeroespacial y Automotriz) y Manufactura:** Mantenimiento

AA es una area multidisciplinaria. Hay muchas diferentes teorías que contribuyen a su desarrollo como: Biología, Económica, Bases de Datos, Visualización, Procesamiento de Señales, Ingeniería y por supuesto Estadística. Ansaf Salleb-Aouissi, profesora de la Universidad de Columbia, indica en (Salleb-Aouissi, 2017) que de hecho la conexión entre AA y Estadística es muy fuerte, al grado de que surgen discusiones referentes a si son áreas separadas o unidas

ya que ambos proveen un conjunto de herramientas y métodos que se traslapan. Sin embargo, también indica, hay notables diferencias tales como que se habla de 'Modelos de Ajuste' en Estadística, mientras que en AA se habla de 'Modelos de aprendizaje'. En Estadística se tratan temas como pruebas de hipótesis y diseño de experimentos con métodos como análisis de varianza (ANOVA), pruebas t, regresiones lineales, análisis de la componente principal, etc. AA trata de modelado computacional en espacios multidimensionales mediante técnicas como Maquina de Soporte de Vectores, Arboles de Decisión Redes Neuronales, Aprendizaje Profundo, etc.

Los algoritmos de AA son divididos según su función y tipo de aprendizaje (Russell y Norvig, 2016).

- En aprendizaje No Supervisado, el sistema aprende incluso aunque no halla retroalimentación, por ello no es supervisado. Solo se requiere de datos de entrada. La tarea más común de este tipo de aprendizaje es el *clustering* o agrupamiento, agrupando de la forma más representativa los datos de entrada.

- El aprendizaje Supervisado el modelo aprende a partir de ejemplos, en donde se les da una entrada y salida y aprende la función que mapea las entradas a las salidas.

- En aprendizaje Semi-Supervisado, algunos ejemplos con etiquetas. Estas etiquetas incluso no tienen que ser "ciertasz son también vistas como ruido. Así el ruido y la falta de etiquetas crean una continuidad entre el aprendizaje supervisado y no supervisado.

- Otro tipo de aprendizaje es el Aprendizaje por Refuerzo. El sistema aprende

de una serie de recompensas o castigos. Es cuestión del sistema determinar que acciones dispararon las recompensas.

Aprendizaje Profundo

Como ya se menciono, Aprendizaje Profundo es un nuevo paradigma creado por uno de los pioneros de la IA, Hinton junto a Osindero. Es una de las técnicas más populares de hoy en día, y es en esencia una nueva terminología para redes neuronales, es decir, una red neuronal con más de dos capas (entrada y salida) puede ser considerada como una arquitectura profunda. Sin embargo, no es acerca del numero de capas sino de modelos que aprendan representaciones de datos con múltiples niveles de abstracción (Lecun y cols., 2015).

Meta-Aprendizaje

El meta-aprendizaje estudia cómo los sistemas de aprendizaje pueden aumentar su eficiencia a través de la experiencia; El objetivo es comprender cómo el aprendizaje en sí mismo puede volverse flexible según el dominio o la tarea en estudio (Vilalta y Drissi, 2002).

Meta-aprendizaje es otro o mejor dicho un algoritmo de aprendizaje específico, donde la fase de aprendizaje aprende a aprender de la mejor manera posible. Lo que significa que el objetivo del meta-aprendizaje es la configuración del algoritmo de AA (Jankowski y cols., 2011).

El teorema "*no free lunch*" señala que no hay un solo algoritmo que sea mejor que todos los demás en todas las tareas, pero también se indica que existe un conjunto de datos para los cuales siempre hay un algoritmo que dará mejo-

res resultados (Duch y Grudzińsk, 2018). El meta-aprendizaje puede determinar entonces cuales son las causas de que el algoritmo domine ciertas tareas determinando 1) las propiedades de las tareas que hacen al algoritmo conveniente 2) las propiedades del algoritmo que contribuyen a que domine esta tarea (Vilalta y Drissi, 2002).

Neuroevolución

Algoritmos de aprendizaje tales como las Redes Neuronales necesitan de la exploración de parámetros adecuados. Los algoritmos evolutivos proveen una interesante alternativa o complemento a los tradicionales algoritmos de aprendizaje tal como RP. Estos algoritmos evolutivos son inspirados en la teoría de la evolución de Darwing la cual señala que solo los más aptos al medio son capaces de sobrevivir y reproducirse tal que sus características genéticas son heredadas a la siguiente generación con el fin de preservar a la especie. La estructura de un algoritmo evolutivo común (Algoritmo Genético) es mostrado en la Figura 2.9. Estos algoritmos han sido utilizados a lo largo del tiempo para la optimización y búsqueda de soluciones de problemas de tipo NP, es decir, problemas cuyo espacio de soluciones es tan extenso que el tiempo necesario para explorar todas al alternativas no es viable, lo que nos lleva a los algoritmos evolutivos que realizan una búsqueda inteligente por todo el espacio de soluciones para obtener una buena solución al problema, la cual no es necesario sea la optima pero si al menos una buena solución al problema.

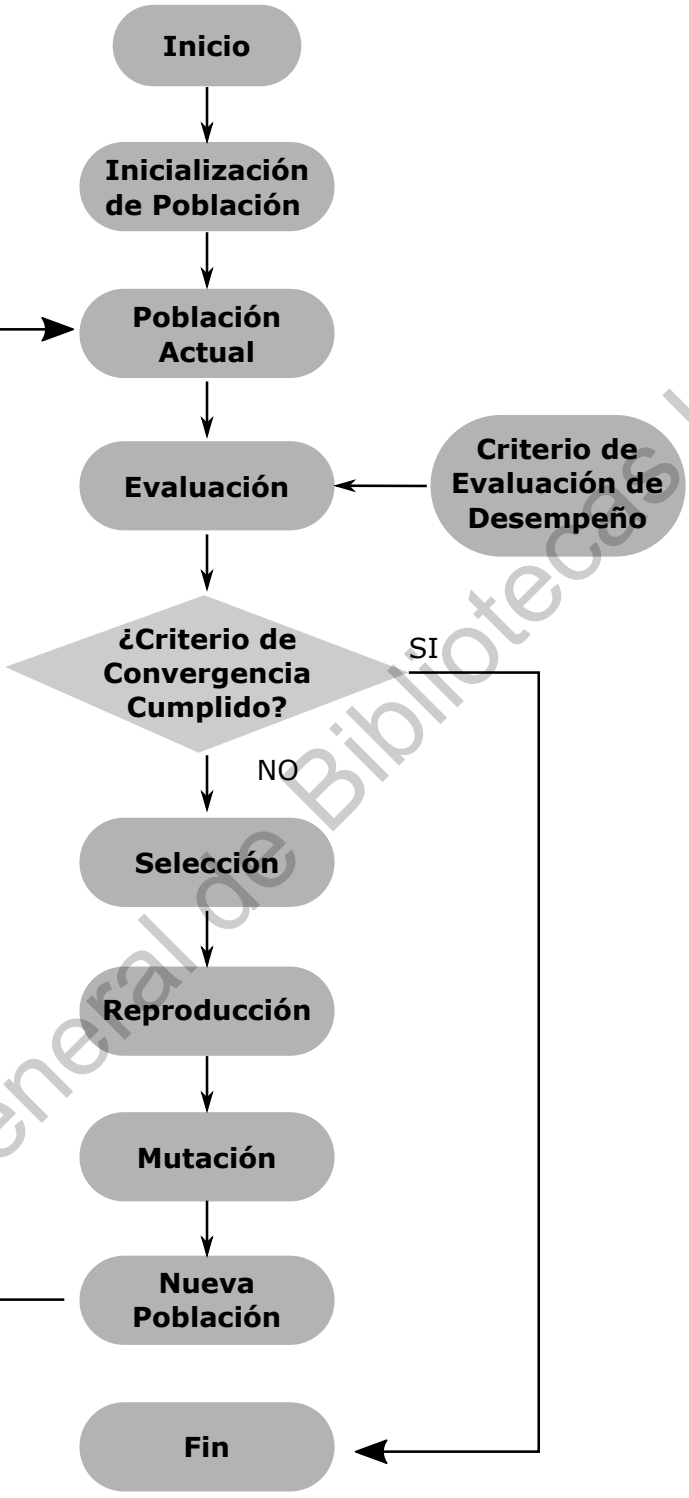


Figura 2.9: Estructura de un Algoritmo Evolutivo. Imagen propia

Neuroevolución es una técnica de meta-aprendizaje orientada específica-

mente a Redes Neuronales, donde por medio de algoritmos evolutivos, busca la configuración de la red neuronal para resolver de la mejor manera la tarea. La ventaja de utilizar algoritmos evolutivos en lugar de otro algoritmo de aprendizaje, es que varias características de la Red pueden evolucionar (modificarse) al mismo tiempo y la definición del desempeño es más flexible que la definición de una función de energía o error (Floreano y cols., 2008).

En 2018, Google, una de las compañias con mayor participación en el área de IA y AA, publicó en (Real y cols., 2018) la mejora de una red neuronal clasificadora de imágenes, mediante el uso de algoritmos evolutivos. Pese a que ya hace varios años la neuroevolución ha sido tratada para el diseño de redes neuronales, la precisión de las arquitecturas resultantes no había alcanzado aquella lograda por la propuesta generada por humanos (Real y cols., 2018). Sin embargo, en el trabajo realizado por Real se logra apreciar una gran mejora de los resultados provenientes de las arquitecturas propuestas por el algoritmo evolutivo. La diferencia con los trabajos previos realizados radica en la modificación al algoritmo evolutivo estándar. Donde el método de selección tradicional llamado "Selección por Torneo" es modificado de la siguiente manera:

- Método Tradicional: Solo los mejores individuos de la población de posibles soluciones son seleccionadas para la reproducción. Estos individuos son los que obtuvieron una mejor evaluación en la función de aptitud.
- Método Propuesto por Google: Cada individuo tiene asociado una edad. El método de la ruleta no selecciona a los mejores, sino a los más jóvenes.

Esta modificación permitió explorar más el espacio de soluciones, evadiendo caer en los óptimos locales. Se especula que algunas arquitecturas pueden obtener buenos resultados sin razón aparente, solo por suerte. Estos indi-

viduos con “suerte” sesgan la dirección de la evolución de la población en un algoritmo evolutivo tradicional, ya que los mejores son seleccionados y sus características heredadas. Con el nuevo método de selección basado en la edad, se previene el sesgo producido por esos individuos con suerte.

En los resultados de este trabajo también se muestra las comparaciones con otros métodos; Aprendizaje por Refuerzo y Búsqueda Aleatoria (ver Figura 2.10).

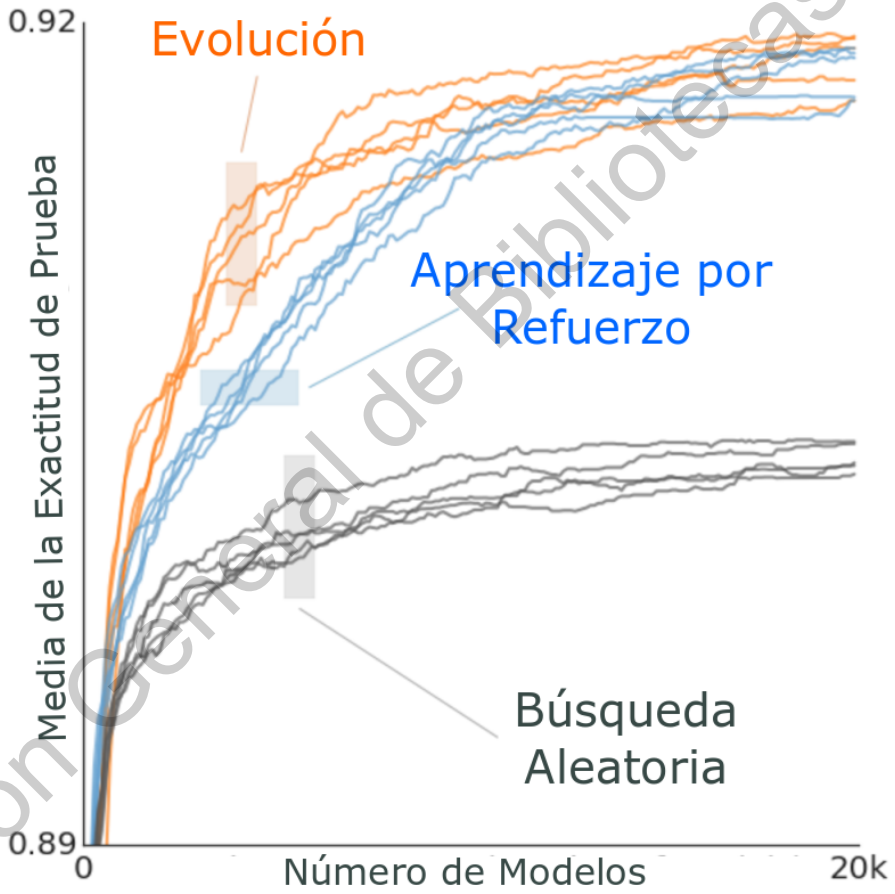


Figura 2.10: Comparación de Métodos de Aprendizaje. En el eje horizontal se describen las horas de entrenamiento y en el eje superior el desempeño de la Red. Tenga en cuenta que este gráfico no muestra el costo de cálculo de los modelos, que fue mayor para los modelos de Aprendizaje por Refuerzo (Real y cols., 2018)

La comparación con la Búsqueda Aleatoria determina la capacidad del

algoritmo evolutivo para converger mediante sus mecanismos de selección y reproducción los cuales la Búsqueda Aleatoria no los tiene. El otro algoritmo, Aprendizaje por Refuerzo, es uno de los métodos de aprendizaje populares de hoy en día dado que ha producido los mejores clasificadores de imágenes hasta el momento (Real y cols., 2018). Es importante observar que algoritmo evolutivo genera muy buenos resultados en un tiempo menor a los demás métodos.

Otro trabajo relevante y reciente, es el realizado en Uber AI labs (Such y cols., 2017), donde se demostró que un simple algoritmo genético puede entrenar redes convolucionales profundas con más de 4 millones de parámetros para jugar juegos de Atari desde píxeles, y en muchos juegos supera los algoritmos modernos de Aprendizaje por Refuerzo.

En otro trabajo también por Uber AI Labs (Lehman y cols., s.f.), se muestra cómo por medio de la combinación de gradientes con la neuroevolución se puede mejorar la capacidad de evolucionar Redes Neuronales Recurrentes y muy profundas, lo que permite la evolución de las Redes Profundas con más de cien capas, un nivel muy superior al que se demostró anteriormente a través de la neuroevolución en (Such y cols., 2017). El uso del gradiente es para lograr mutaciones “seguras”. Estas mutaciones incrementan la capacidad de un método simple de neuroevolución basado en algoritmos genéticos para encontrar soluciones en dominios de alta dimensión que requieren Redes Neuronales Profundas y / o Recurrentes.

El uso del gradiente para la mejora de algoritmos evolutivos ya ha sido probado anteriormente en (Lara y cols., 2010). En este trabajo se muestra el uso de programación matemática para subsanar las debilidades de los algoritmos evolutivos, en términos de búsqueda y convergencia de soluciones. La técnica utilizada es el gradiente junto a el NSGA-II, el cual es un algoritmo evolutivo. Los

resultados indican que el uso puntual del gradiente para mejorar algunos individuos, realmente ayuda al algoritmo NSGA-II a converger y realizar una mejor exploración, haciendo de éste algoritmo memético (híbrido) una excelente herramienta para optimización multi-objetivo.

2.7. Estado del Arte de PGE

En las Tablas 2.1, 2.2 y 2.3 se muestran algunas de las técnicas utilizadas para pronóstico.

Tabla 2.1: Técnicas de Aprendizaje No Supervisado (Lei y cols., 2018)

Técnica	**Ventajas**	**Limitaciones**
K-means	Tiempos de entrenamiento cortos	Necesita conocimiento previo. El número de conjuntos (*clusters*) debe ser definido previamente
Teoría de la Resonancia Adaptativa	Tiempos de entrenamiento cortos. Capaz de modelar conjuntos con forma no lineal	Requiere de un conveniente esquema de preprocesamiento
Mapa Auto-organizado	El mapeo de los datos es fácilmente interpretado. Capaz de organizar conjuntos complejos de datos.	Difícil de determinar que pesos de entrada usar. Requiere que los puntos vecinos se comporten de forma similar
Modelos Ocultos de Márkov	Modelos estadísticos. Escalables	Requiere una gran cantidad de datos para un modelo preciso. Puede ser complejo

AP puede encontrar características del estado de los sistemas de manera eficiente, y por ello el diagnóstico y pronóstico basado en AP se ha tornado como un prometedor campo de investigación (Zhao y cols., 2016). En la Tabla 2.3 se listan algunas de las técnicas de AP utilizadas para la predicción de VUR.

Tabla 2.2: Técnicas de Aprendizaje Supervisado (Lei y cols., 2018)

Técnica	**Ventajas**	**Limitaciones**
Arboles de Decisión	No es paramétrico y fácil de entender	Árboles demasiado complejos no generalizan bien los datos (esto se llama sobreajuste). Puede estancarse en mínimos locales
Random Forest	Desempeño mejor a *Decision Trees*. Rápido y escalable. Entrenamiento más rápido que MSV	Aumenta en bias
Naive Bayes	Simple y Requiere menos datos	Los supuestos de independencia de atributos pueden ser demasiado restrictivos. Las características deben estar correlacionadas.
Redes Bayesianas	Modelos probabilistas. Reduce el número de parámetros para aprender. Fácil de visualizar las dependencias	Requiere conocimiento previo del sistema. El aprendizaje es complejo
Maquina de Soporte Vectorial (MSV)	Alta precisión. Robusto contra el ruido. Eficiente para gran cantidad de datos.	Requiere datos de entrenamiento etiquetado. Los resultados pueden ser incomprensibles. No hay estándar para escoger la función kernel
Perceptrón Multicapa (PMC)	Pocos parámetros necesitan ser optimizados para el entrenamiento. Sistema adaptativo	No existen estándares para su estructura

Tabla 2.3: Técnicas de Aprendizaje Profundo (Lei y cols., 2018)

Técnica	**Ventajas**	**Limitaciones**
Autocodificadores Neuronales	Fácil de implementar. Reducción dimensional. Fácil de rastrear la función de costo que es minimizada por Retropropagación	Requiere gran cantidad de datos, y ajuste fino de parámetros. Aprende a capturar la mayor cantidad de información posible en lugar de solo la más relevante.
Denoising Autoencoder	Mejor para remover ruido. Implícitamente diseñado para formar un modelo generativo	Añade ruido aleatoriamente a las entradas
Autocodificadores Variacionales	Aprende que distribución de ruido utilizar. Puede generar datos usando distribuciones. Explicitamente diseñado para formar un modelo generativo.	Puede ser difícil de optimizar. Puede ser difícil de implementar
Máquina de Boltzmann Restringida	Capaces de crear patrones si hay falta de datos	Difícil de entrenar. Difícil de rastrear la función costo

Máquina de Boltzmann Profunda	Incorpora incertidumbre respecto a entradas ambiguas. Puede aprender muy buenos modelos generativos. Los parámetros de todas las capas pueden ser optimizados en conjunto	Puede que el entrenamiento sea más lento en comparación a *Deep Belief Networks*. La optimización en conjunto es no práctica para conjuntos de datos grandes.
Red de Creencia Profunda	Bueno para datos en una sola dimensión. Puede ser una herramienta más poderosa que Análisis de Componente Principal para reducir la dimensionalidad de los datos.	El entrenamiento puede ser demasiado lento e ineficiente.
Red Neuronal Convolucional	Bueno para datos multidimensionales. Bueno para extracción de características de forma local	Complicada estructura y requiere más tiempo para entrenamiento

Redes Neuronales Recurrentes, Redes con Memoria de-Corto-Largo Plazo y Unidad Recurrente Cerrada	Buenos para datos secuenciales. Pueden detectar cambios en el tiempo	Puede ser difícil de entrenar e implementar

Los modelos de Regresión de Proceso Gaussiano ha demostrado ser una herramienta superior a Maquinas de Soporte Vectorial y redes neuronales siendo útil para predicción bajo una gran dimensionalidad y cantidades de datos de tamaño pequeño, sin embargo su mayor desventaja es su gran costo computacional (Lei y cols., 2018). Por otro lado, *Neural Network Gaussian Processes*, una red profunda, es computacionalmente más eficiente (Lee y cols., 2017). Su uso en PGE no ha sido reportado en la literatura.

Referente a los algoritmos híbridos, podemos encontrar ejemplos en los trabajos de Elattar y cols. (2018) y Baptista y cols. (2018), quienes demuestran la mejora de los resultados de algoritmos de pronóstico de VUR utilizando Perceptrón Multicapa y Filtros Kalman. Estos últimos mitigan el hecho de que las predicciones de los algoritmos basados en datos tienden a estar afectados con ruido lo que reduce la precisión de la estimación. Sin embargo, aunque el uso de estos filtros es conveniente aún son sensibles al tuneo de parámetros. Por otro lado, *Deep Kalman Filter* es un filtro basado en Redes Neuronales Profundas entrenado a partir de diferentes filtros Kalman (Krishnan y cols., 2015). Su uso no

ha sido explorado de forma independiente o incluso en conjunto con otra técnica para PGE.

De forma más reciente, el uso de *variational encoders* ha demostrando buenos resultados para PGE (Yoon y cols., 2017). Dado que el modelo trabaja con modelos generativos, es eficiente para el modelado de sistemas complejos mientras que produce resultados con alta precisión, pero aun hay limitadas investigaciones para promover su potencial (Khan y Yairi, 2018).

Áreas de Oportunidad

A continuación se listan las areas de oportunidad en el campo de PGE según diferentes revisiones.

Según Elattar y cols. (2016) en su revisión de métodos de pronóstico:

- La incertidumbre no puede ser eliminada pero si manejada por medio del modelado de ruido, evasión de sobreajuste y utilizando técnicas de estimación híbridas
- El uso de métricas para la validación y verificación del modelo es indispensable
- Reconfiguración automática basada en las predicciones del modelo

Según (Lei y cols., 2018) en su revisión del campo de investigación de PGE

- Se deben establecer nuevos enfoques para desarrollar la división de estados para separar los cambios causados por el desarrollo de una falla de

los cambios causados por otros factores (condiciones de las operaciones e interferencias con el medio ambiente)

- Es significante el analizar la interacción de las fallas entre los diferentes componentes para la predicción de la VUR a un nivel de sistema.
- El manejo de incertidumbres es la base para una predicción de la VUR precisa. Más investigación al respecto es necesaria.
- La predicción de la VUR de un solo componente considerando múltiples fallas. La mayoría de la investigación simplifica los enfoques y no lo considera.

Según Zhao y cols. (2016) y Khan y Yairi (2018) en su artículo de revisión de la aplicación de Aprendizaje Profundo en PGE:

- La mayoría de los enfoques son específicos a una aplicación y no hay una manera clara para seleccionar, diseñar o implementar un algoritmo.
- Un limite debe ser definido respecto al numero de cálculos con las varias arquitecturas
- No se han explicado o documentado las razones para llevar a cabo dichas arquitecturas de los algoritmos en los diferentes artículos
- No hay mucho énfasis en el costo de la implementación de arquitecturas de Aprendizaje Profundo
- Hay un problema en que la mayoría de la investigación no ha sido implementada en práctica debido a algunas restricciones, particularmente en la implementación de hardware (sistemas embebidos en tiempo real)

2.8. Turbofan

2.8.1. Base de Datos

La base de datos para generar el modelo es proveída por el Prognostic Center of Excellence of National Aeronautics and Space Administration (NASA) (NASA Ames Research Center, s.f.), en su labor por facilitar el desarrollo de algoritmos de pronóstico.

Esta base de datos fue creada a partir de la ejecución a falla de motores turbofan usando un modelo de simulación termodinámico llamado C-MAPSS (*Commercial Modular Aero- Propulsion System Simulation*) (Saxena y cols., 2008). Los elementos del modelo están en la Figura 2.11.

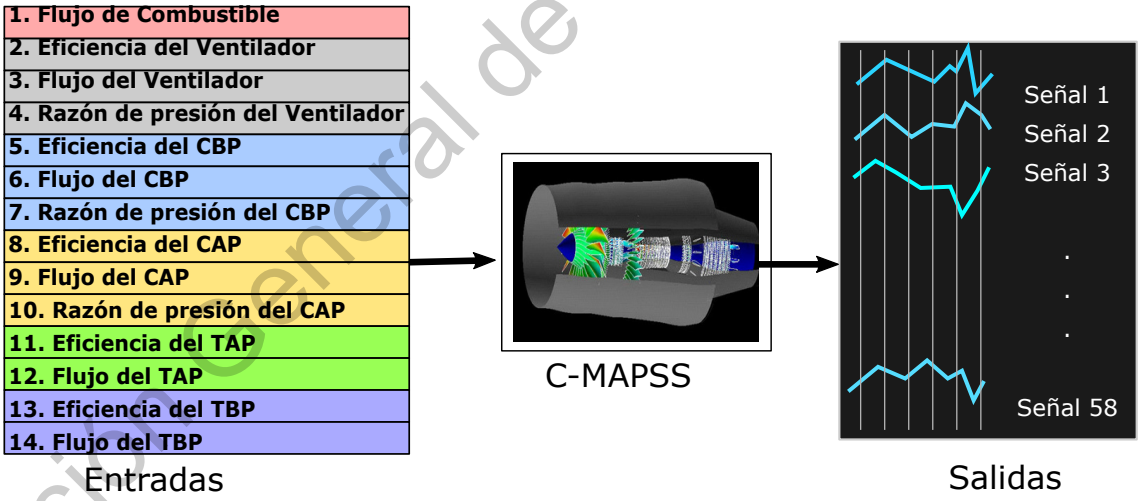


Figura 2.11: Diagrama simplificado del Modelo. Compresor de Baja Presión (CBP), Compresor de Alta Presión (CAP), Turbina de Alta Presión (TAP), Turbina de Baja Presión (TBP). Imagen Propia

El simulador consiste de 14 parámetros de entradas y 58 señales de salida, y de estas últimas solamente 21 son las que se reportan en la base de datos. Así, cada unidad turbofan provee la siguiente información:

- Número de Unidad (ID)
- Ciclos de Vuelo
- 3 Parámetros o Condiciones de Operación (altitud, número mach, ángulo del resolver del acelerador)
- 21 Señales de Sensores (ver Tabla 2.4)

Tabla 2.4: Sensores de la Base de Datos. Rankine (R), libra por pulgada cuadrada (psi), psi absoluta (psia), revoluciones por minuto (rpm), flujo (pps), libras masa (lbm), segundos (s). Tabla propia.

Número de Sensor	Descripción	Unidades
1	Temperatura en la entrada del ventilador	° R
2	Temperatura a la salida del CBP	° R
3	Temperatura a la salida del CAP	° R
4	Temperatura a la salida de la TBP	° R
5	Presión a la entrada del ventilador fan	psia
6	Presión en el conducto de desviación	psia
7	Presión a la salida del CAP	psia
8	Velocidad física del ventilador fan	rpm
9	Velocidad física del núcleo	rpm
10	Cociente de la presión del motor	
11	Presión estática a la salida del CAP	psia
12	Cociente de flujo de combustible a la presión estática en la salida del CAP	pps/psi
13	Velocidad física del ventilador fan corregida	rpm
14	Velocidad física del núcleo corregida	rpm
15	Cociente de desviación	
16	Cociente de quema de combustible-aire	
17	Entalpia de compresión	
18	Velocidad de ventilador fan demandada corregida	
19	Velocidad de ventilador fan demandada	
20	Flujo másico del refrigerante en la TBP	lbm/s
21	Flujo másico del refrigerante en la TAP	lbm/s

La base de datos contiene múltiples series de tiempo, divididas en 4 sub-

grupos de entrenamiento y prueba, ambos identificados por los nombres: FD001, FD002, FD003 y FD004 (ver Tabla 2.5). Asimismo, contiene resultados de los conjuntos de prueba para la verificación y validación del algoritmo. Cada serie de tiempo corresponde a un turbofan en particular, y en conjunto constituyen datos de una flotilla de motores, donde todos son del mismo tipo. Cada turbofan inicia con diferentes grados de desgaste inicial y variación de la manufactura, pero estos son desconocidos para el usuario. Tanto el desgaste como la variación no constituyen una condición de falla, sino características normales que describen a cada motor. Todos los datos, a excepción de los ciclos de vuelo, contienen ruido. De estos datos, hay tres configuraciones operativas (altitud, número mach, ángulo del resolver del acelerador) que dividen a las unidades en 6 condiciones de operación.

Los datos hacen referencia a un turbofan funcionando normalmente al comienzo de cada serie de tiempo, y desarrollan un modo de falla en algún punto (esta base de datos contiene dos modos de falla). En los datos de entrenamiento, la falla crece en magnitud hasta que el sistema deja de funcionar, y aquí termina el registro de los datos para la unidad. En el conjunto de pruebas, la serie de tiempo termina en algún momento antes de que el sistema deje de funcionar. El objetivo es predecir el tiempo (en ciclos de vuelo) en el que el sistema fallará completamente, o en otras palabras, predecir su VUR.

Tabla 2.5: Información de los Subconjuntos de Datos (Saxena y cols., 2008)

	FD001	FD002	FD003	FD004
Unidades de Turbofan para Entrenamiento	100	260	100	249
Muestras de Señales para Entrenamiento	20,631	53,759	24,720	61,249
Muestras de Etiquetas para Entrenamiento	0	0	0	0
Unidades de Turbofan para Prueba	100	259	100	248
Muestras de Señales para Prueba	13,096	33,991	16,596	41,214
Muestras de Etiquetas para Prueba	100	259	100	248
Número de modos de Falla	1	1	2	2
Números de Condiciones de Operación	1	6	1	6

Las propiedades de la base de datos son las siguientes (Lei y cols., 2018) (Eker y cols., 2012):

- Aunque los datos son obtenidos de una simulación de degradación es simulado con la mayor posible precisión, todavía es diferente de los casos reales.
- Es ideal para entrenamiento de algoritmos que requieren una gran cantidad de datos. Cada subconjunto de datos no incluye menos de 100 unidades de entrenamiento y prueba.
- Este es un problema típico de pronóstico basado en fusión de información de múltiples sensores.
- Es ideal para estudios de relaciones entre las tendencias de degradación y la variación de las condiciones de operación con respecto al tiempo.

- No es apropiado para generación de modelos físico-matemáticos dado que no se proveen de indices de estado de salud para la degradación del sistema completo.

2.9. Prerocesamiento

2.9.1. Etiquetando la Base de Datos

Para entrenar un algoritmo supervisado es necesario contar con pares de datos de entrada y salida, tal que el algoritmo aprenda a las relaciones entre los datos y pueda generalizar la correlación para nuevas entradas, no vistas durante el entrenamiento.

En el contexto de pronóstico de vida útil de sistemas complejos, los datos de entrada serían los valores se los sensores del sistema en monitoreo y la salida sería la vida útil remanente del sistema. Así con esta información nuestro algoritmo o modelo aprendería la correlación entre los sensores y el tiempo de vida que le queda al sistema. Sin embargo, en muchas aplicaciones industriales es prácticamente imposible tener la información de vida útil remanente, por lo que comúnmente en las bases de datos para pronóstico de fallas sólo se cuenta con los valores de los sensores.

Con respecto a la base de datos del turbofan. Hay dos métodos de etiquetado de datos utilizados comúnmente: método basado en degradación lineal y método basado en una función por partes (*piecewise*). A continuación se describen estos dos métodos y uno más basado en degradación lineal porcentual. Este último, aunque no es muy utilizado, vale la pena mencionarlo.

Degradación Lineal

El método de degradación lineal, propuesto por Peel (2008), quien fue el ganador del *Prognosis Health Management Data Challenge*, es el método más simple de etiquetar la base de datos. Peel propone utilizar como etiqueta directamente el tiempo que le queda a la turbina antes de que ésta deje de funcionar. Tal método queda descrito por la ecuación 2.1.

$$v = EoL - c \tag{2.1}$$

Donde v es al etiqueta que representa la VUR, EoL (*End of Life*) es el último ciclo de operación y c es el número de ciclos en operación.

En la tabla 2.6 tenemos un ejemplo de los datos registrados para una turbina. Esta turbina tiene 192 ciclos de operación y para cada ciclo de vuelo se tiene valores de los sensores. Utilizando el método de Peel (ecuación 2.1), EoL tiene un valor de 192 Y los valores de VUR quedarían como 191, para el primer ciclo de vuelo, seguido de 190, 189, etc.

Sin embargo, éste método sugiere que todas las turbinas se degradan linealmente a través de toda su vida, lo cual ciertamente no es el caso.

Función *Piecewise*

Propuesto por Heimes (2008), la función *piecewise* es de los métodos más utilizados. Este modelo representa mejor el comportamiento de degradación del sistema. Sugiere que la salud del sistema se comporta de forma constante hasta que ocurre una falla y entonces comienza a decrecer linealmente.

Tabla 2.6: Muestra de datos de la unidad 1 del conjunto de entrenamiento FD001. Tabla propia.

Turbofan ID	Número de Ciclos de Vuelo	Cond. de Op. 1	Cond. de Op. 2	Cond. de Op. 3	Sensor 1	...	Sensor 21	VUR
1	1	-0.0007	-0.0004	100	518.67	...	23.41	?
1	2	0.0019	-0.0003	100	518.67	...	23.42	?
1	3	-0.0043	0.0003	100	518.67	...	23.34	?
.	.	.	.	.	.	...	.	.
.	.	.	.	.	.	...	.	.
.	.	.	.	.	.	...	.	.
1	192	0.0009	0	100	518.67	...	22.96	?

Las observaciones que llevaron a Heimes al desarrollo de este método de etiquetado fueron las siguientes:

- La corrida mínima es de 127 ciclos, es decir, no hay datos con menos de 127 ciclos. La media de ciclos en la base de datos es de 209.
- En promedio el inicio de la falla debe aparecer en la muestra número 105.
- Basado en esto, un valor de VUR inicial de entre 120 y 130 ciclos parece razonable.
- La pendiente de entrenamiento para la parte de degradación es -1.

Con la información previa se determina que el modelo de degradación queda descrito por la siguiente ecuación.

$$v = \begin{cases} R_c & \text{si } 0 \leq c \leq SoF \\ EoL - c & \text{si } SoF < c \leq EoL \end{cases} \tag{2.2}$$

Donde v es al etiqueta (VUR), EoL es el último ciclo (*End of Life*), c es el número de ciclos de vuelo en operación, R_c es el valor constante inicial de VUR,

el cual varia de 120 a 130 ciclos y SoF (*Start of Failure*) es el inicio de la falla el cual es igual a $EoL - R_c$.

En la Figura 2.12, se muestra las etiquetas del ejemplo de la turbina de la Tabla 2.6 utilizando el método de Heimes con un valor de R_c = 130.

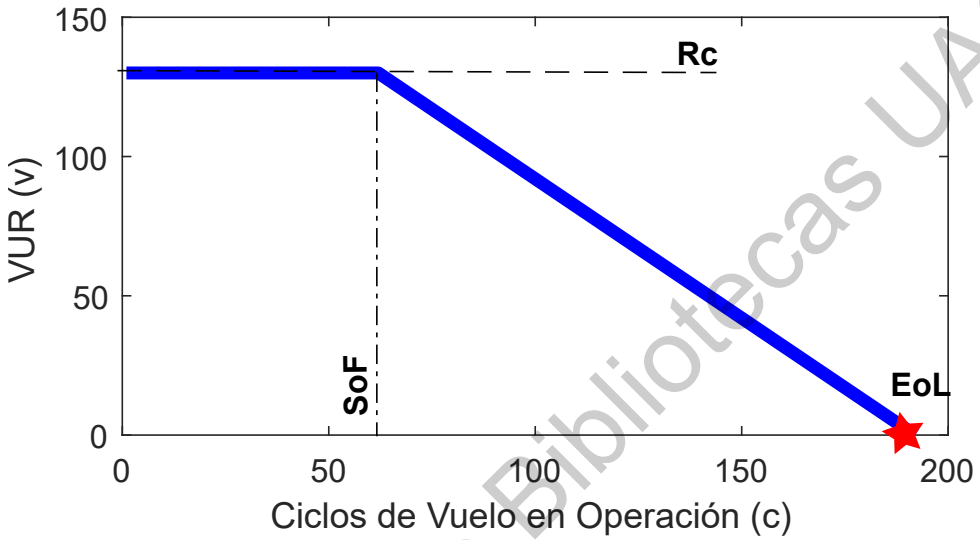


Figura 2.12: Etiquetas por medio del método de Heimes. Ejemplo utilizando la unidad 1 del conjunto de entrenamiento FD001. Imagen propia.

Cabe mencionar que el valor de Rc es un parámetro del método el cual tiene que ser ajustado durante el desarrollo del algoritmo de pronóstico.

Degradación Lineal Porcentual

El método llamado Degradación Lineal Porcentual fue propuesto por Jain y cols. (2014) y consiste en calcular el porcentaje de vida remanente del sistema de acuerdo a la siguiente ecuación:

$$v = \frac{(EoL - c)}{EoL} \tag{2.3}$$

La salida es automáticamente normalizada entre 0 y 1, lo cual provee estabilidad numérica. Y a diferencia de los otros dos métodos, el algoritmo entrenado por este método no producirá el VUR en números de ciclos, sino en porcentaje de vida. Para determinar entonces el VUR se realiza mediante la siguiente ecuación:

$$t = \frac{c}{1 - v} - c \tag{2.4}$$

donde t es el VUR estimado, c es el ciclo actual, y v es el tiempo de vida porcentual remanente.

2.9.2. Partición de Régimen

Los creadores de la base de datos indican que hay 3 variables que indican las condiciones de operación de las turbinas y estas tienen un fuerte impacto en el desempeño del sistema. El gráfico de las tres variables de operación (Figura 2.13) muestra que los datos están concentrados en seis conjuntos, indicando seis regímenes de operación. Entendiéndose por régimen de operación como una combinación en particular de condiciones de operación.

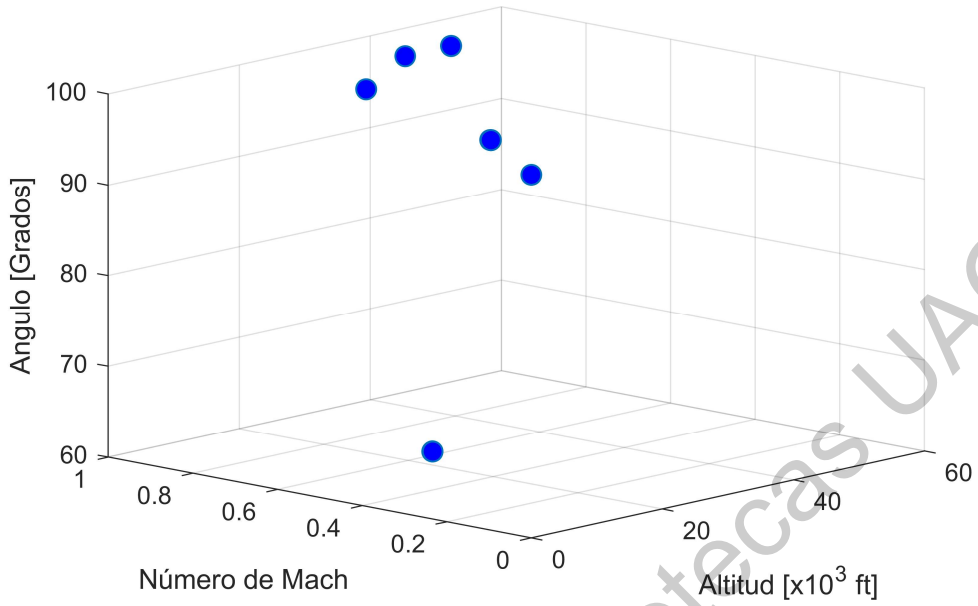


Figura 2.13: Regímenes de Operación dados por las condiciones de operación: Angulo de Resolver del Acelerador, Altitud y Número de Mach. Imagen propia.

Para reducir la dimensionalidad de los datos se ha propuesto incluso sustituir los valores de los tres parámetros de operación por un solo valor $r \in \{1, 2, 3, 4, 5, 6\}$ que indique el régimen de operación.

Por otro lado, Elattar y cols. (2018) y Peel (2008) proponen retirar los valores de los tres parámetros de operación y añadir 6 nuevos indicadores a la base de datos. Estos indicadores llamados Indicador Histórico de Funcionamiento de la Turbina (IHFT) o *Historical Engine Run Indicators* (HERI), por su nombre en inglés, representan el número de ciclos que el turbofan a permanecido en cada régimen de operación (ver Figura 2.14).

La razón de esta última propuesta es para compensar la incapacidad de las redes neuronales estáticas como PMC para retener información previa, la cual es una característica especial de las redes neuronales dinámicas como redes de Memoria de Largo y Corto Plazo (MLCP), o Red Neuronal Recurren-

te (RNR).

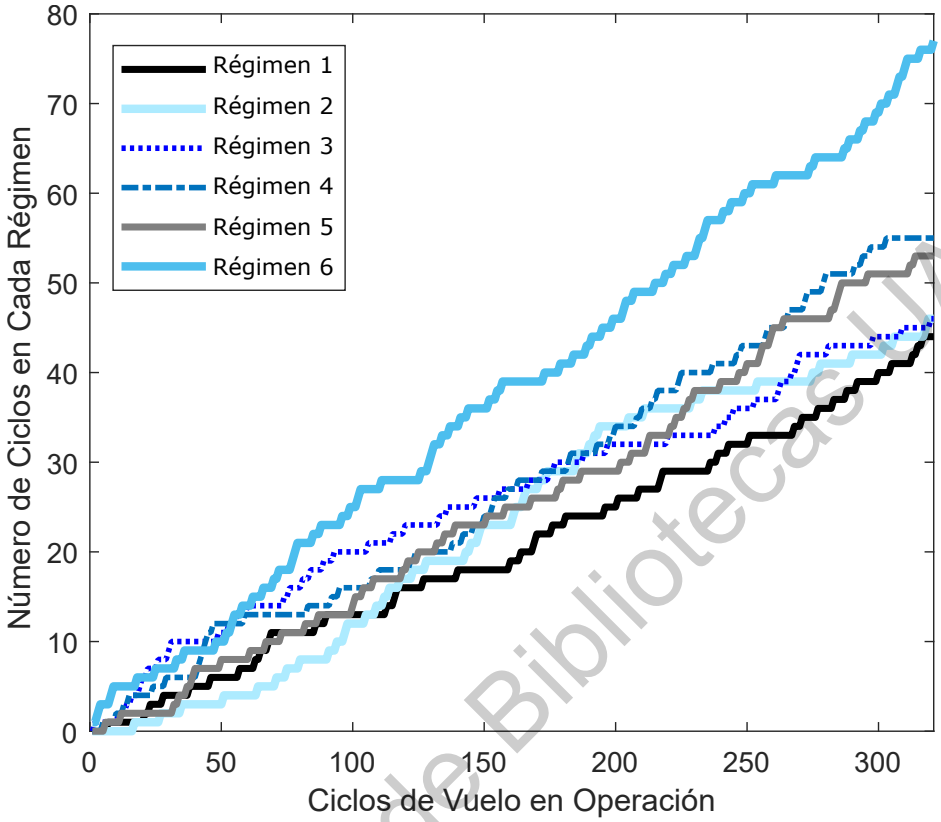


Figura 2.14: Indicadores del Funcionamiento de la Turbina. Ejemplo de la Turbina ID:1/FD004. Imagen propia.

2.9.3. Normalización de Datos

Un paso de preprocesamiento común es la normalización de componentes en un conjunto de datos antes del análisis para proporcionar una escala común en todas las características de un conjunto de datos y asegurar una contribución igualitaria por parte de cada una de las mediciones y evitar el sesgo.

Z-Score

Uno de los métodos de normalización más utilizados es el Z-Score (ecuación 2.5), el cual se basa en la media y la desviación estandar de los datos para escalarlos. La ventaja que provee este método es que reduce el efecto de los datos atípicos (*outliers*), mientras que su desventaja es que no escala los datos a un rango específico tal como la normalización Min-Max.

$$N(x^{(f)}) = \frac{x^{(f)} - \mu^{(f)}}{\sigma^{(f)}} \tag{2.5}$$

Donde $x^{(f)}$ es el dato original, f son las características/atributos/componentes, en este caso los sensores, y $\mu^{(f)}$ junto con $\sigma^{(f)}$ son la media y desviación estándar, respectivamente, de los datos para la característica f.

Tras la normalización de los datos con esta ecuación los regímenes de operación se encuentran solo escalados (ver Figura Figura 2.15).

Sin embargo mientras este método provee un escalamiento uniforme sobre los datos, Peel (2008) indica que no hay relación entre los modos de operación y los ciclos de vida remanente del sistema, por lo que maximizar la varianza de los datos sería mejor para la tarea de predicción (Peel, 2008). Esto se logra incorporando la información de cada régimen en la normalización, dando como resultado la siguiente ecuación.

$$N(x^{(r,f)}) = \frac{x^{(r,f)} - \mu^{(r,f)}}{\sigma^{(r,f)}} \tag{2.6}$$

Donde r indica el régimen de operación.

Esto además asegura una contribución igualitaria de cada sensor sin importar el regimén de operación (Sateesh Babu y cols., 2016).

En la Figura 2.15 se observa que tras la normalización por la ecuación 2.6 los 6 regímenes desaparecen.

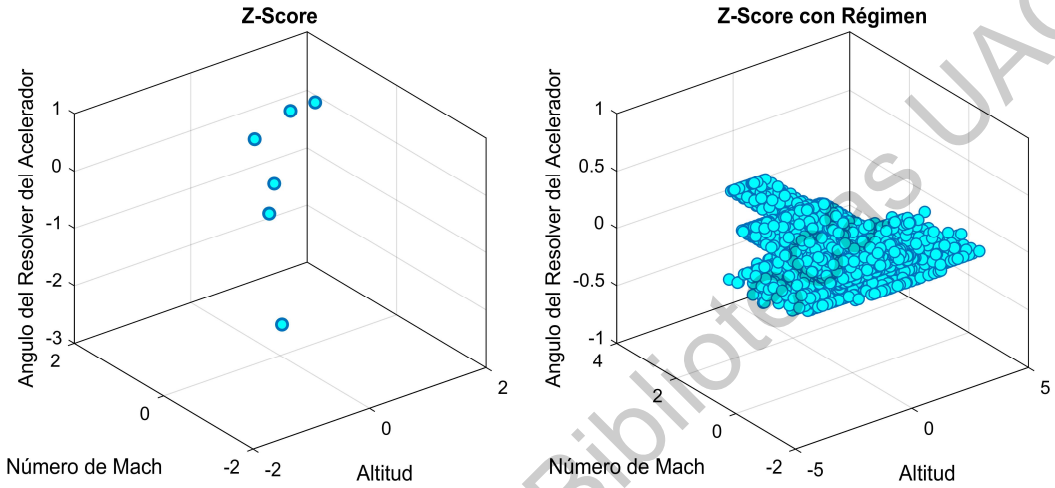


Figura 2.15: Comparación de la Distribución de los Régimenes tras la Normalización. Imagen propia.

Min-Max

Otro método utilizado ampliamente en la normalización de datos es el Min-Max. Este método ofrece la ventaja de que escala los datos a un rango definido ya sea [0,1] o [-1,1], sin embargo, a diferencia de la Z-Score no disminuye el efecto de los datos atípicos.

La ecuación para normalizar los datos en un rango de [-1,1] es la siguiente.

$$N(x^{(f)}) = 2\frac{x^{(f)} - x^{(f)}_{min}}{x^{(f)}_{max} - x^{(f)}_{min}} - 1 \tag{2.7}$$

Tomando en cuento lo anterior, respecto a dar la misma importancia a

todas las componentes (sensores) sin importar el régimen de oración, se añade la información del régimen a la normalización de Min-Max, quedando como la siguiente ecuación:

$$N(x^{(r,f)}) = 2\frac{x^{(r,f)} - x^{(r,f)}_{min}}{x^{(r,f)}_{max} - x^{(r,f)}_{min}} - 1 \tag{2.8}$$

En la Figura 2.16 se observa nuevamente que los regímenes de operación desaparecen.

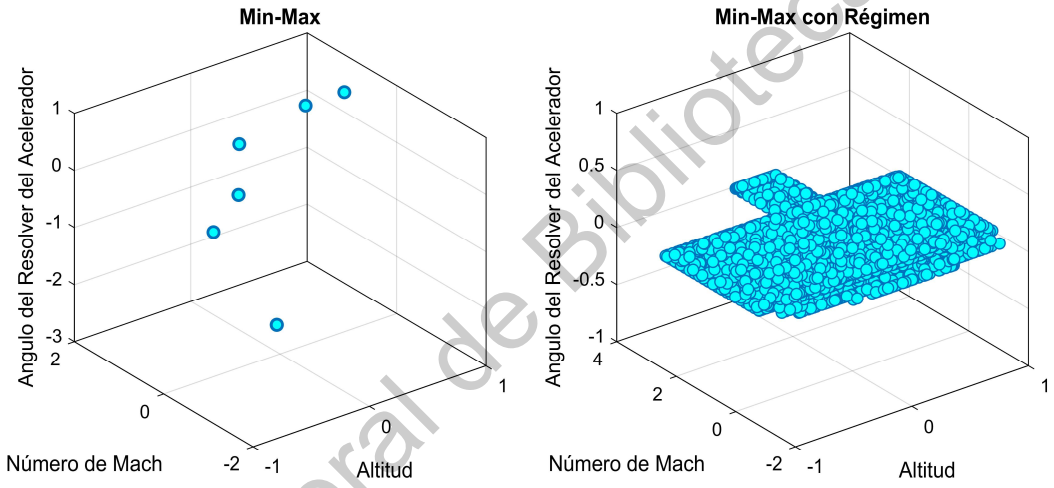


Figura 2.16: Comparación de la Distribución de los Régimenes tras la Normalización. Imagen propia.

2.9.4. Reducción de Dimencionalidad

La reducción de dimensionalidad es una técnica de preprocesamiento común que tiene como objetivo reducir el número de componentes, variables, elementos o atributos del fenómeno en estudio, manteniendo la mayor cantidad de información posible. Existen básicamente dos enfoques: Selección de Características (*Feature Selection*) y Extracción de Características (*Feature Extraction*). En la primera simplemente se selecciona un subconjunto de las carac-

terísticas, siendo éstas las más relevantes o las que mejor describen el comportamiento de nuestros datos. La segunda se refiere a la creación de nuevas características a partir de las originales, buscando que éstas sean menos y mantengan una gran cantidad de información de las originales.

2.9.5. Selección de Características

A lo largo de la literatura referente al C-MAPSS, la selección de sensores o componentes como los datos de entrada del algoritmo tiende a variar un poco. La siguiente tabla resume varios trabajos y los sensores seleccionados.

Tabla 2.7: Selección de componentes en varios trabajos de la literatura. Tabla propia.

Autor de Trabajo	Ciclos	Condiciones de Operacion	Sensores	IHFT
Elattar y cols. (2018)	No	Si	2,3,4,7,8,9,11, 12,13,14,15,20,21	Si
Lin y cols. (2018) Javed y cols. (2014)	No	No	2, 3, 4, 8, 11, 13, 15, 17	No
Sateesh Babu y cols. (2016)	No	No	Todos	Si
Heimes (2008)	No	Si	Todos	No
Peel (2008)	No	N/A	Todos	Si
Li y cols. (2018)	No	No	2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 17, 20, 21	No
Jain y cols. (2014)	Si	Si	Todos	No
Ellefsen y cols. (2019)	No	Solo para FD002 y FD004	2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 17, 20, 21 para FD001 y FD003; Todos para el FD002 y FD004	No

De este breve resumen de trabajos utilizados se puede deducir que las configuraciones de componentes seleccionadas son las siguientes.

Tabla 2.8: Combinaciones de Componentes. Tabla propia.

Configuración	ID	Ciclos de Operación	Parámetros de Operación	IHFT	Sensores
A	NO	SI	SI	NO	TODOS
B	NO	NO	SI	NO	TODOS
C	NO	NO	NO	NO	TODOS
D	NO	SI	SI	NO	2,3,4,7,8,9,11,12, 13,14,15,17,20 y 22
E	NO	NO	SI	NO	2,3,4,7,8,9,11,12, 13,14,15,17,20 y 22
F	NO	NO	NO	NO	2,3,4,7,8,9,11,12, 13,14,15,17,20 y 22
G	NO	SI	NO	NO	TODOS
H	NO	SI	NO	NO	2,3,4,7,8,9,11,12, 13,14,15,17,20 y 22
I	NO	NO	NO	SI	TODOS
J	NO	NO	NO	SI	2,3,4,7,8,9,11,12, 13,14,15,17,20 y 22

Se puede observar que respecto a lo sensores se suelen utilizar todos o sólo aquellos que tienen comportamiento monótamente ascendente o descendente (sensores 2,3,4,7,8,9,11,12,13,14,15,17,20 y 22).

La Tabla 3.1 no lista todas las combinaciones posibles, pero si las combinaciones más relevantes.

2.9.6. Extracción de Características con Análisis de Componentes Principales

Una de las técnicas más comunes para extracción de características es el Análisis de Componentes Principales (ACP).

El ACP es una técnica que reduce la dimensionalidad de los datos x a

través de buscar combinaciones lineales $a_1x, a_2x, a_3x, ..., a_qx$ (Componente Principales) que maximicen la varianza de los datos. En el proceso de maximizar esta varianza en la combinación lineal se encuentra que los vectores $a_1, a_2, a_3, ..., a_q$ son los eigenvectores de la matriz de covarianza S.

Los eigenvalores son proporcionales a la varianza de sus respectivos eigevectores o componentes principales.

Dada una matriz $\mathbf{X}$ representando un conjunto de datos de 10 observaciones con 3 variables (ver ecuación 2.9).

$$\mathbf{X} = \begin{bmatrix} 7 & 4 & 3 \\ 4 & 1 & 8 \\ 6 & 3 & 5 \\ 8 & 6 & 1 \\ 8 & 5 & 7 \\ 7 & 2 & 9 \\ 5 & 3 & 3 \\ 9 & 5 & 8 \\ 7 & 4 & 5 \\ 8 & 2 & 2 \end{bmatrix} \tag{2.9}$$

Los eigenvalores correspondientes a los datos son dados por la matriz siguiente:

$$\lambda = \begin{bmatrix} 1.769 & 0 & 0 \\ 0 & 0.927 & 0 \\ 0 & 0 & 0.304 \end{bmatrix} \tag{2.10}$$

Y los eigenvectores (componentes principales) son lo siguientes:

$$\mathbf{V} = \begin{bmatrix} 0.64 & 0.38 & -0.66 \\ 0.69 & 0.10 & 0.72 \\ -0.34 & 0.91 & 0.20 \end{bmatrix} \tag{2.11}$$

Los eigenvalores tienen la caracteristica de ser valores decrecientes. Y dado que son proporcionales a la varianza que cada componente principal contiene de los datos originales, entonces representa también la relevancia de cada componente principal, siendo así que la primer componente principal es más importante que la que sigue y la que sigue.

En la Tabla 2.9 se observa que la primer componente contiene el 59 % de la varianza de los datos y la siguiente 31 %. Por lo que al utilizar las primeras dos componentes aún podemos mantener el 90 % de la varianza total.

Tabla 2.9: Eigenvalores y su correspondiente porcentaje de la varianza total. Tabla propia.

Componente Principal	Eigenvalor	Porcentaje	Acumulado
CP 1	1.769	58.97	58.97
CP 2	0.927	30.90	89.87
CP 3	0.304	10.13	100.00

Es importante señalar que en la literatura referente al uso de la base de datos del C-MAPSS, no se ha reportado el uso de ACP o alguna otra técnica de extracción de características.

2.9.7. División de Datos

Un paso importante antes de entrenar un algoritmo de Aprendizaje Automático es la división de la base de datos en tres conjuntos: Entrenamiento,

Validación y Prueba.

- Datos de Entrenamiento: Son utilizados para ajustas los parámetros del modelo.
- Datos de Validación: Son utilizados para ajustar los hiperparámetros del modelo y monitorear el error en este conjunto, conocido tambien como Error de Generalización, el cual permite evitar el sobre entrenamiento del modelo.
- Datos de Prueba: Son utilizados para tener una evaluación final y certera del desempeño del modelo.

En varios casos de suele hablar solo de conjuntos de entrenamiento y prueba. Y si el punto es evaluar un modelo es suficiente con estos dos conjuntos de datos. Sin embargo, si lo que se busca es encontrar los mejores hiperparámetros y sólo se usa el conjunto de prueba para evaluar estos valores, muy posiblemente nos encontraremos que el modelo en producción tendra un desempeño menor al visto por el conjunto de prueba. Esto es debido a que este conjunto se evaluó multiples veces y los hiperparámetros se ajustaron para este conjunto de pruebas en específico dando como resultado un sobre ajuste del modelo. Para evitar esto se suele agregar un conjunto de Validación, que servirá para probar los hiperparámetros del modelo y el conjunto de Prueba será uno jamás visto durante el entrenamiento y ajuste de hiperparámetros, por lo que la evaluación sobre éste nos dara una idea certera del desempeño del modelo en producción.

Discrepancia de Datos de Prueba y Entrenamiento

Hoy en día es común encontrarse casos de bases de datos muy grandes para entrenamiento de modelos, sin embargo, éstas no necesariamente repre-

sentan de forma perfecta los datos vistos por el modelo en producción. En estos casos la regla más importante es (Géron, 2019):

- El conjunto de datos de validación y debe de ser tan representativo como sea posible de lo datos vistos en producción.

La forma de hacer esto es utilizar los datos que se verán por el modelo en producción y dividirlos en dos conjuntos: Validación y Pruebas. De tal forma que el conjunto de Entrenamiento será aquel por la base de datos que no es tan representativa, pero si es muy grande.

Al entrenar el modelo en esta situación y siguiendo la regla podemos encontrarnos que si el error en el conjunto de validación es grande puede ser debido a dos razones:

- El modelo ha sido sobre entrenado con los datos de entrenamiento.
- El error es debido a la discrepancia (debida a la naturaleza del problema) de los datos de validación y de entrenamiento.

Para corroborar la razón del error en el conjunto de validación es necesario incluir un cuarto conjunto de datos llamado Entrenamiento-Validación. Este conjunto debe ser una porción del conjunto de Entrenamiento. Y es para evaluar el sobre entrenamiento del modelo con el conjunto de Entrenamiento. Tal que si la diferencia de errores de Entrenamiento y Entrenamiento-Validación es muy grande, significa que hay sobre entrenamiento del modelo. De otra forma, el error visto por el modelo, se debe a la discrepancia de los datos de Validación/Prueba con los de Entrenamiento.

2.10. Métricas

Para evaluar el desempeño del algoritmo se ha utilizado el Error Cuadrático Medio (ECM) el cual queda descrito por la ecuación 2.12.

$$ECM = \frac{1}{N}\Sigma_{i=1}^{N}(\hat{t}_i - t_i)^2 \tag{2.12}$$

donde N es el número de muestras, $\hat{t}_i$ es la estimación de VUR y t_i el valor real de VUR.

Además del ECM, se utiliza una métrica llamada Penalización Exponencial del Error (PEE). Ésta fue propuesta para la competencia de algoritmos de pronóstico por (Salleb-Aouissi, 2017)Saxena2008. Tiene la característica de ser una función asimétrica, penalizando más las predicciones tardías que las predicciones tempranas (ver Figura 2.17). Un algoritmo ideal obtendría una evaluación de cero utilizando esta métrica.

$$d_i = \hat{t}_i - t_i \tag{2.13}$$

$$s_i = \begin{cases} e^{-d_i/13} - 1 & \text{si } d_i \leq 0 \\ e^{d_i/10} - 1 & \text{si } d_i > 0 \end{cases} \tag{2.14}$$

$$S = \frac{\Sigma_{i=1}^{N} s_i}{N} \tag{2.15}$$

Donde S es la evaluación del algoritmo, d_i es el error y s_i es la penalización

del error.

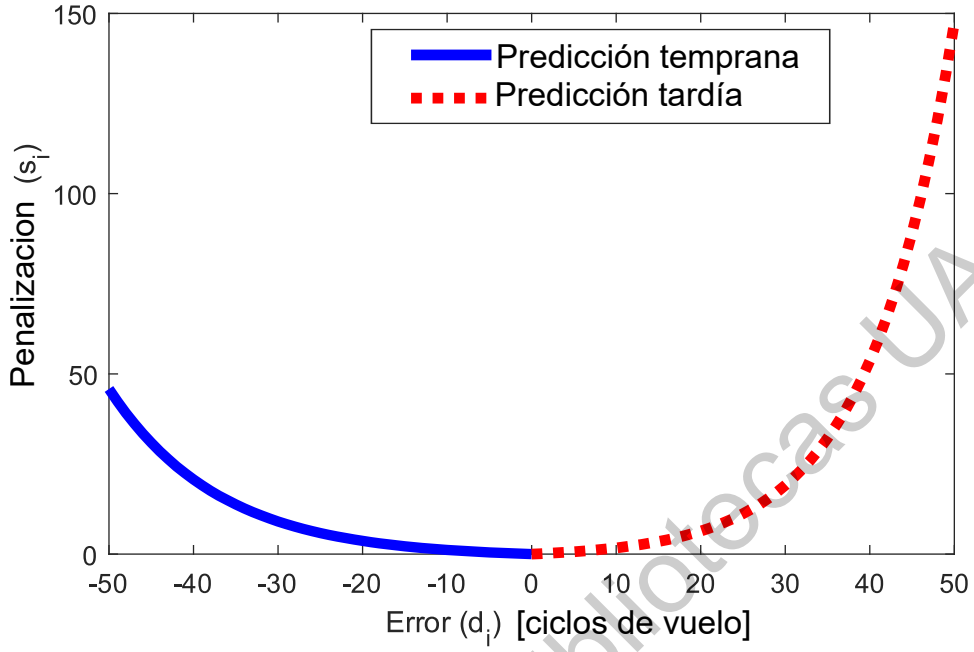


Figura 2.17: Gráfico de la Penalización Exponencial del Error. Las predicciones tardías son mayormente penalizadas. Imagen propia.

2.11. Red Neuronal Perceptrón Multicapa

Recientemente, se han propuesto diferentes modelos de predicción para el pronóstico de la VUR del turbofan, entre las que se encuentran: Perceptrón Multicapa (PMC), Red Neuronal Recurrente (RNR), Red con Memoria de Larco y Corto Plazo (MLCP), Red Neuronal Convolucional (RNC), Red de Creencia Profunda (RCP), entre otras.

Las series de tiempo generalmente ofrecen una estructura de comportamiento tal que una medición en el instante k está altamente relacionada con la medición en un instante previo $k - n$, donde n es el número de retardos, pudiendo ser uno o varios. Cuando la decisión es usar redes neuronales, las mejores opciones son redes neuronales dinámicas (MLCP o RNR). Sin embargo, éstas

son computacionalmente más costosas para aplicaciones en sistemas embebidos (Elattar y cols., 2018). Por otro lado, si bien el PMC es menos costoso computacionalmente, las predicciones hechas por esta red tienden a ser ruidosas y con poca precisión, debido a que sus predicciones no estan relacionadas con las del instante previo. Por otro lado, (Elattar y cols., 2018) indica que el uso de los indicadores de tiempo, tales como IHFT, ayudan al PMC ha obtener tan buenos resultados como redes dinámicas. Y además se ha probado que el uso del Filtro Linear de Kalman puede corregir la señal, incluyendo la información de la predicción en el instante previo (Baptista y cols., 2018).

2.11.1. Optimizadores

Gradiente Descendente

En matemáticas, el gradiente $\nabla f(x)$ es una generalización multivariable de la derivada. Mientras que una derivada se puede definir solo en funciones de una sola variable, para funciones de varias variables, el gradiente toma su lugar. El gradiente es una función de valor vectorial, a diferencia de una derivada, que es una función de valor escalar. Por lo tanto, el gradiente es un vector que apunta hacia donde la función tiene un mayor incremento. La magnitud del gradiente es la pendiente de esa dirección.

Cuando se desea encontrar el máximo de una función convexa a partir de un punto p de su dominio, se toman pasos proporcionales en la dirección del gradiente de la función en el punto p. Por otro lado, si se desea encontrar el mínimo de una función convexa, entonces se toman pasos proporcionales en sentido opuesto del gradiente y en este caso se denomina algoritmo de gradiente descendente.

El algoritmo es múy genérico. Dada una función $f : \Omega \subseteq R^m \longrightarrow R$ con un mínimo en p. Para encontrar p se construye una sucesión de puntos $p_0, p_1, p_2, ...p_k, ...$ que convergen a p. El punto incial es aleatorio y el punto siguiente es dado por:

$$p_{k+1} = p_k - \alpha_k \nabla f(p_k) \tag{2.16}$$

Donde α_k es un hiperparámetro que en optimización es llamado 'tamaño de paso' y en Apredizaje Automático es llamado 'tasa de aprendizaje' (*learning rate*). Si su valor es muy pequeño, entonces el algoritmo tomará muchas iteraciones y tiempo para converger. Por otro lado, si su valor es muy alto, posiblemente nunca converja debido a que estará evadiendo el mínimo debido a los grandes pasos que toma entre cada punto.

Método de Newton

El método de Newton es un algoritmo de segundo orden porque hace uso de la matriz cuadrada de las segundas derivadas parciales, también conocida como matriz Hessiana. El objetivo de este método es encontrar mejores direcciones de entrenamiento utilizando las segundas derivadas de la función de pérdida.

$$p_{k+1} = p_k - \alpha_k (H_k^{-1} \nabla f(p_k)) \tag{2.17}$$

donde H_k^{-1} es la matriz Hessiana inversa de f evaluada en p_k.

Sin embargo, el método de Newton tiene la dificultad de que la evaluación exacta de la matriz Hessiana y su inversa son bastante costosas en términos

computacionales.

Quasi-Newton

Un enfoque alternativo, conocido como Quasi-Newton (Shanno, 1970), es usado para para resolver el inconveniente del método de Newton. En lugar de calcular la Hessiana directamente y luego evaluar su inversa, éste crea una aproximación G de la Hessiana inversa H^{-1} en cada iteración del algoritmo. Esta aproximación se calcula utilizando solo información de las primeras derivadas de la función a optimizar.

$$p_{k+1} = p_k - \alpha_k(G_k^{-1}\nabla f(p_k)) \tag{2.18}$$

Levenberg-Marquardt

El algoritmo de Levenberg-Marquardt (Moré, 1978), también conocido como método de mínimos cuadrados amortiguados, ha sido diseñado para funcionar específicamente con funciones que toman la forma de una suma de errores al cuadrado. Funciona sin calcular la matriz Hessiana exacta, sino que trabaja con el gradiente y la matriz Jacobiana J.

Considerando una función para optimizar que puede ser expresada por una suma de errores cuadrados de la forma:

$$f = \Sigma_{i=1}^{m} e_i^2 \tag{2.19}$$

donde m es el número de muestras en el conjunto de datos y e_i los errores.

Se puede definir la matriz Jacobiana de la función f como la que contiene las derivadas de los errores con respecto a cada parámetro p.

$$J_{i,j} = \frac{\partial e_i}{\partial p_j} \tag{2.20}$$

Donde $i = 1, 2, 3, ..m$ y $j = 1, 2, 3..n$, siendo n el número de parámetros. En el caso de redes neuronales artificiales, n es el número de pesos.

El gradiente puede ser calculado como:

$$\nabla f = 2J\vec{e} \tag{2.21}$$

donde $\vec{e}$ es el vector de todos los errores e_i.

Finalmente la matriz Hessiana puede ser aproximada por la siguiente expresión:

$$H \approx 2J^T \cdot J + \lambda I \tag{2.22}$$

donde λ es un factor que segura que la matriz Hessiana sea positiva y I es la matriz Identidad.

El algoritmo de Levenberg-Marquardt es un método diseñado para funciones del tipo suma de error cuadrado. Eso hace que sea muy rápido al entrenar redes neuronales medidas con dicha función. Sin embargo, este algoritmo tiene los siguientes inconvenientes.

- No se puede aplicar a funciones como el error cuadrático medio o el error

de entropía cruzada

- No es compatible con los términos de regularización
- Para conjuntos de datos y redes neuronales muy grandes, la matriz jacobiana se vuelve enorme y, por lo tanto, requiere mucha memoria

Adagrad

Adagrad (Duchi y cols., 2011) es un algoritmo para optimización basada en gradiente que hace exactamente esto: primero, le da una tasa de aprendizaje $\alpha_{k,i}$ a cada parametro $p_{k,i}$ y segundo, adapta esta tasa, realizando actualizaciones más pequeñas (es decir, valores pequeños) para los parámetros asociados con las funciones que ocurren con frecuencia, y actualizaciones con valores más grandes para los parámetros asociados con las características poco frecuentes. Por esta razón, es muy adecuado para tratar con datos escasos.

En su regla de actualización, Adagrad modifica la tasa de aprendizaje en cada paso de tiempo basado en los gradientes pasados que se han calculado.

Adadelta

Adadelta (Zeiler, 2012) es una extensión de Adagrad que busca reducir su tasa de aprendizaje agresiva y monotónicamente decreciente. En lugar de acumular todos los gradientes cuadrados pasados, Adadelta restringe la ventana de gradientes pasados acumulados a un tamaño fijo.

RMSprop

RMSprop es un método, con una tasa de aprendizaje adaptativa, no publicado pero propuesto por Geoff Hinton en un curso en línea.

RMSprop y Adadelta se han desarrollado de forma independiente al mismo tiempo debido a la necesidad de resolver las tasas de aprendizaje radicalmente decrecientes de Adagrad.

Adam

Adaptive Moment Estimation (Adam) (Kingma y Ba, 2014) es un algoritmo de optimización de primer orden (usa información de la primer derivada) basado en aprendizaje adaptativo diseñado específicamente para entrenar redes neuronales profundas, debido a su velocidad. Publicado por primera vez en 2014, Adam fue presentado en una conferencia muy prestigiosa para profesionales de aprendizaje profundo (ICLR 2015). El documento contenía algunos diagramas muy prometedores, que muestran enormes ganancias de rendimiento en términos de velocidad.

Es otro método que calcula las tasas de aprendizaje de forma adaptativa para cada parámetro. Además de almacenar un promedio exponencialmente decreciente de gradientes cuadrados pasados tal como Adadelta y RMSprop, Adam mantiene un promedio exponencialmente decreciente de gradientes pasados. Mientras que el momentum puede verse como una bola que corre por una pendiente, Adam se comporta como una bola pesada con fricción. Sin embargo, después de un tiempo, la gente comenzó a darse cuenta de que, en algunos casos, Adam realmente encuentra una solución peor que el gradiente descendente

estocástico. Se han realizado muchas investigaciones para abordar los problemas de Adam.

2.12. Filtro de Kalman

El Filtro de Kalman (FK) es una técnica de procesamiento de señal basada en un proceso iterativo y mediciones de datos con incertidumbre para generar, en general, las mejores estimaciones de la variable de interés (Kalman, 1960).

El algoritmo de FK consiste básicamente en dos etapas: predicción y actualización (Elattar y cols., 2018).

Filtro de Kalman
Entrada: $Z = \{z_0, z_1 ... z_n\}$ (Predicciónes normalizadas de VUR hechas por el PMC), $\hat{x_0}$ y P_0 (Estimaciones iniciales)
Salida: $\hat{X} = \{\hat{x}_0, \hat{x}_1 ... \hat{x}_n\}$ (Predicciones normalizadas corregidas)
1: **for** $k \in \{1...n\}$ **do**
2: $\hat{x}_k^- = A\hat{x}_{k-1}$
3: $P_k^- = AP_{k-1}A^T + Q$
4: $K_k = P_k^- H'/(HP_k^- H^T + R)$
5: $\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$
6: $P_k = (I - K_k H) * P_k^-$
7: **end for**

2.12.1. Etapa de Predicción

La etapa de predicción está definida en los pasos 2 y 3 del algoritmo. Donde $\hat{x}_k^-$ es el estimado a priori del vector de estados $\hat{x_k}$ en el tiempo k, A es la matriz de transición de estado, P_k^- es la matriz de covarianza del error asociada a la estimación a priori y Q es la matriz de covarianza del ruido en el proceso.

Elattar y cols. (2018) hacen los siguientes supuestos para adaptar el FK a la degradación del turbofan. Primeramente, se utiliza el modelo de un objeto en movimiento para el cálculo y actualización de los estados de la degradación ($\hat{x_k}$). En este modelo se asume que la degradación (el "objeto") evoluciona bajo una aceleración constante, quedando su ecuación de estados conforme la ecuación 2.23.

$$\hat{x_k} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_{k-1} \\ \dot{v}_{k-1} \end{bmatrix} \tag{2.23}$$

Donde v_k es la VUR, $\dot{v_k}$ es su razón de degradación, y Δt, el tiempo de muestreo que es igual a 1 ciclo de vuelo, ya que en la base de datos las mediciones son cada ciclo de vuelo del turbofan. Así, de ecuación 2.23 se tiene entonces que $A = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$.

La matriz de covarianza del ruido del proceso Q, correspondiente al modelo de un objeto en movimiento, se define conforme a la siguiente ecuación.

$$Q = \begin{bmatrix} \Delta t^4/4 & \Delta t^3/2 \\ \Delta t^3/2 & \Delta t^2 \end{bmatrix} \sigma_a^2 \tag{2.24}$$

Finalmente $Q = \begin{bmatrix} 1/4 & 1/2 \\ 1/2 & 1 \end{bmatrix} \sigma_a^2$,donde $\sigma_a = 0.01$, siendo este último valor obtenido por Elattar y cols. (2018) tras prueba y error.

2.12.2. Etapa de Actualización

La actualización del vector de estados a priori $\hat{x}_k$ y la matriz de covarianza del error P_k (pasos 5 y 6) depende básicamente del valor de la ganancia K_k, calculada en el paso 6 del algoritmo, donde H es la matriz que indica la relación entre mediciones y el vector de estado al momento k, en el supuesto ideal de que no hubiera ruido en las mediciones.

La salida de la red neuronal es utilizada como el estado de mediciones z_k, y éste queda descrito por la ecuación 2.25.

$$z_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} v_k \\ \dot{v}_k \end{bmatrix} \tag{2.25}$$

De la ecuación 2.25 se tiene entonces que $H = \begin{bmatrix} 1 & 0 \end{bmatrix}$. R, la covarianza del ruido en la medición, es igual a σ_z^2, donde $\sigma_z = 0.3$ (obtenido tras prueba y error).

Las condiciones iniciales se indican como $v_0 = 1$ y la razón de degradación es $\dot{v}_0 = 1/209$, ya que 209 es el promedio de ciclos de vuelo en la base de datos. Por lo que las estimaciones iniciales quedan como:

$$\hat{x_0} = \begin{bmatrix} 1 \\ 0.005 \end{bmatrix} \tag{2.26}$$

$$P_0 = \begin{bmatrix} 1/4 & 1/2 \\ 1/2 & 1 \end{bmatrix} 0.001 \tag{2.27}$$

CAPÍTULO 3

Metodología

> *"Una ciencia es tanto más útil cuanto más universalmente pueden comprenderse sus producciones; y, al contrario, lo serán menos en la medida en que éstas sean menos comunicables"*
>
> Leonardo Da Vinci

Las metodologías para el desarrollo de modelos basados en Aprendizaje Automático siguen la siguiente estructura (ver Figura 3.1). Dado un conjunto de datos, estos son preparados, en la etapa de preprocesamiento, aquí se remueve los datos atípicos, en caso de haberlos, se escalan los datos (normalizan), y se seleccionan las componentes más relevantes para reducir la dimensionalidad. Posteriormente en la etapa de modelado, se hace el ajuste de los hiperparámetros del algoritmo para que éste aprenda de los datos de la mejor manera posible. Finalmente se hace una evaluación del algoritmo y en caso de que la evaluación no sea satisfactoria se deberá modificar el modelo, ya sea probando nuevas técnicas de preprocesamineto, ajuste de hiperparámetros o el uso de un algoritmo nuevo.

Para este proyecto, seguimos la metodología para el desarrollo de algoritmos de Aprendizaje Automático. Partiremos de un primer modelo compuesto de una serie de técnicas de preprocesamiento, un algoritmo (Perceptrón Multicapa) con unos hiperparámetros iniciales y posteriormente iteramos para encontrar

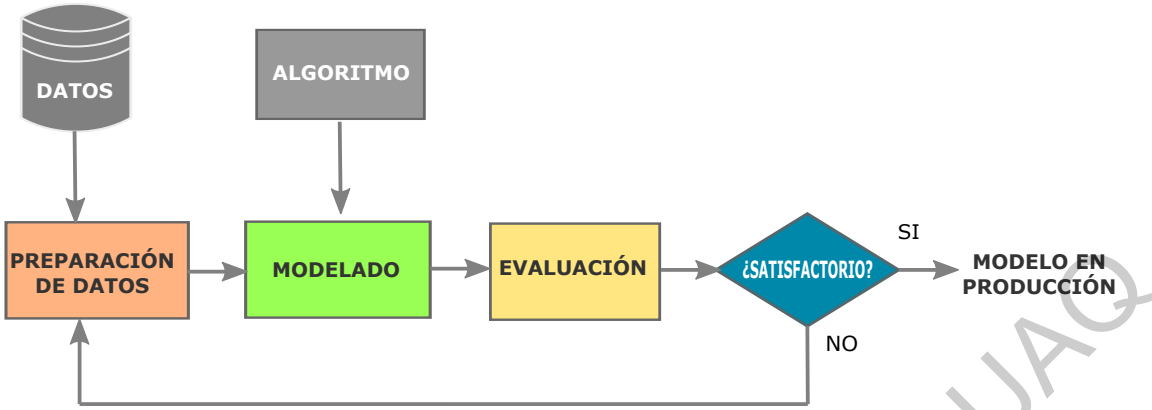


Figura 3.1: Metodologia para Aprendizaje Automatico. Imagen Propia.

las mejores técnicas de preprocesamiento e hiperparámetros que aumenten el desempeño del modelo.

3.0.1. División de Datos y Evaluación

En las aplicaciones de pronóstico no es posible contar siempre con las etiquetas de los datos para entrenar un algoritmo supervisado y por ello las bases de datos sólo contienen la información de los sensores y el tiempo que ha durado operando el sistema. La base de datos de la NASA tiene esta característica. La base de datos tiene un gran conjunto de datos para entrenamiento no etiquetados y un conjunto de prueba con su etiqueta real, de tal forma que podamos evaluar el desempeño de nuestro modelo con estos datos de prueba.

La metodología tradicional sugiere etiquetar los datos de entrenamiento utilizando alguna técnica propuesta en la literatura. Posteriormente entrenar el algoritmo con los datos de entrenamiento y las etiquetas de entrenamiento propuestas, y evaluar el modelo utilizando las métricas del Error Cuadrado Medio (ECM) y la Penalización Exponencial del Error (PEE). Para ajustar los hiperparámetros del modelo y evitar el sobre entrenamiento, la metodología sugiere

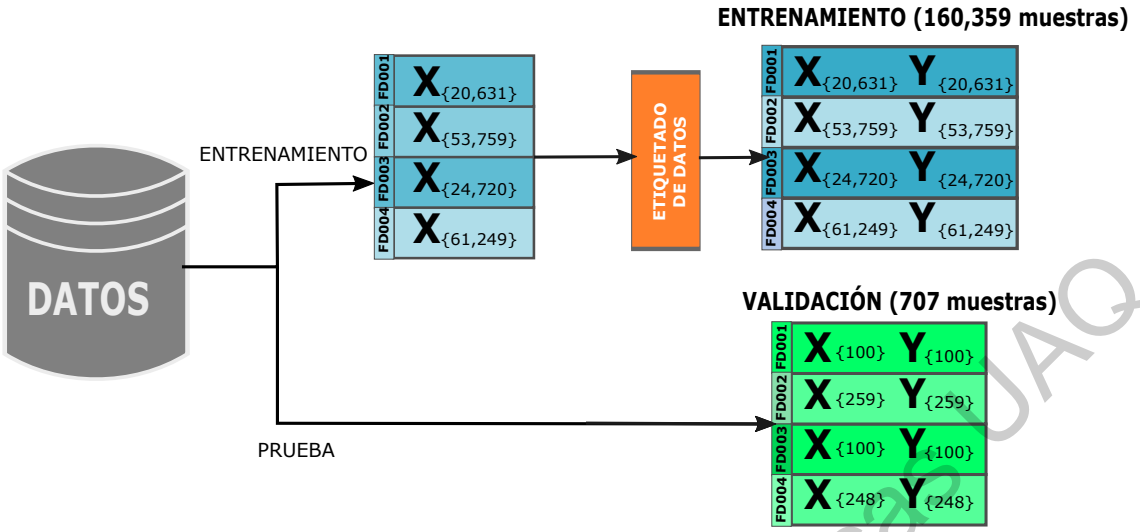


Figura 3.2: División de datos para entrenamiento y validación. **X** representa los datos de entrada del modelo y **Y** las etiquetas. Los números entre las llaves son la cantidad de muestras. Imagen propia.

también separar un porcentaje de los datos de entrenamiento y utilizarlos como validación. Dando un total de 3 conjuntos: Entrenamiento, Validación y Prueba.

Por otro lado, es claro que la distribución de los datos de entrenamiento es diferente a la de los datos de prueba. Esto surge debido a que hemos etiquetado los datos de entrenamiento con aproximaciones de las etiquetas reales, las cuales son desconocidas. Este tipo de discrepancia en distribución de datos de entrenamiento y prueba es común hoy en día en aplicaciones incluso diferentes a pronóstico de VUR. Para enfrentar este problema de discrepancia de distribución de datos de entrenamiento y prueba se recomienda fuertemente que el conjunto de datos de validación provenga de los datos de prueba (datos que vera el modelo en producción) y no de los de entrenamiento. Por lo que el conjunto de Prueba suele ser dividido en Prueba y Validación. En este caso, dado que se cuneta con una muy poca cantidad de muestras de Prueba, en comparación con las de Entrenamiento, todo el conjunto de Prueba pasa a ser también el conjunto de Validación.

3.0.2. Modelo Inicial de Prueba

La arquitectura del la red neuronal comprende tres capas (ver Figura 3.3): capa de entrada compuesta del ciclo de vuelo, 3 condiciones de operación y 21 señales de sensores, una capa oculta de 10 neuronas con una función de activación tangente hiperbólica, y una capa de salida compuesta por una neurona con función de activación lineal. La salida son las etiquetas generadas a partir de alguna de las técnicas de etiquetado. Esta última es normalizada a un valor entre 0 y 1, dividiendo por el valor máximo de las etiquetas. La red neuronal es entrenada un número máximo de 250 épocas, sin embargo, el entrenamiento se detendrá automáticamente una vez que el error de generalización (dado por el conjunto de validación) aumenta o si el error no disminuye durante 10 épocas de entrenamiento. Esta técnica de paro de entrenamiento es llamada Detención Temprana (*Early Stopping*). El algoritmo de entrenamiento es Lavenberg-Marquardt con un valor inicial de tasa de aprendizaje igual a 0.001.

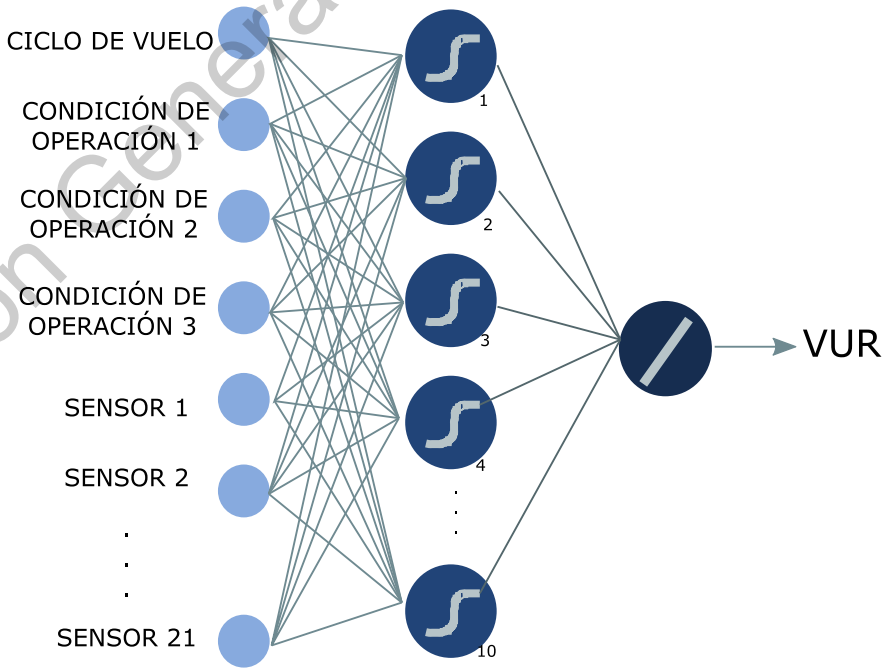


Figura 3.3: Arquitectura de la Red Neuronal. Imagen propia.

Utilizando este modelo de red neuronal, se ajustaran los parámetros del modelo, comenzando primeramente con las técnicas de preprocesamiento, y siguiendo con los hiperparámetros de la red neuronal.

Tomando en cuenta que las redes neuronales son susceptibles a sus pesos iniciales se han creado 32 conjuntos de pesos iniciales, utilizando números aleatorios dentro de un rango de ± 0.05. De esta forma cada modificación al modelo es probada 32 veces, para así tener una evaluación concreta del desempeño del modelo independiente de sus pesos iniciales.

3.0.3. Preprocesamiento

El primer conjunto de pruebas será enfocado a buscar las mejores técnicas de preprocesamiento para la base de datos.

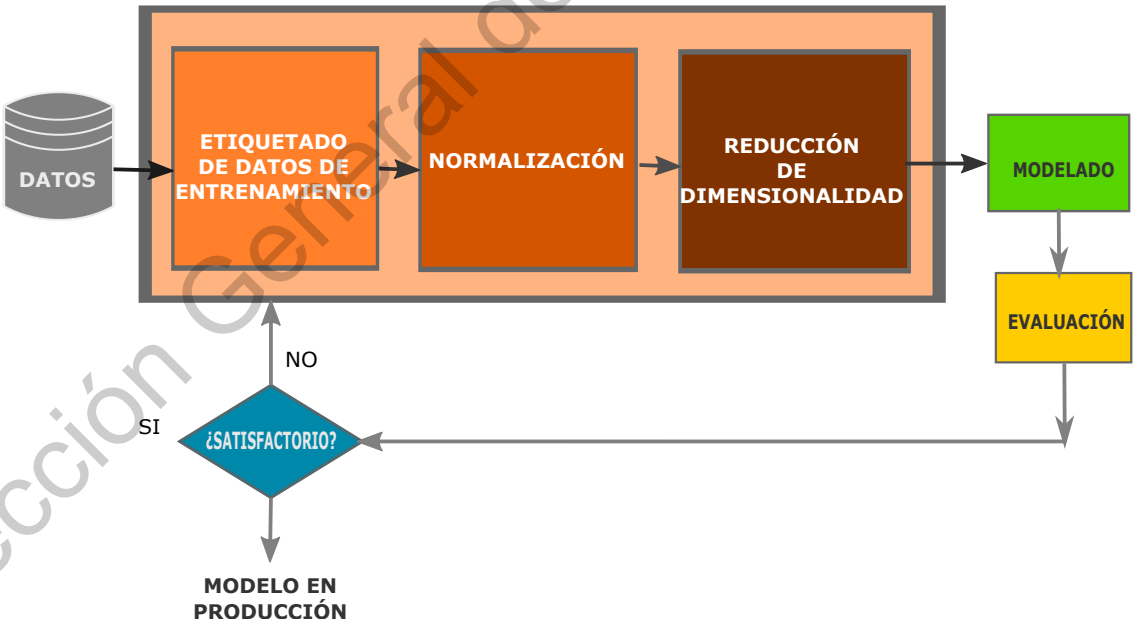


Figura 3.4: Etapas de Pre-procesamiento. Imagen propia.

Como se observa en la Figura 3.4, esta etapa buscará primeramente la

mejor técnica de etiquetado, posteriormente la mejor técnica de normalización y finalmente el mejor método de disminución de dimencionalidad.

Las técnica de etiquetado seleccionada es la propuesta por Heimes (2008). Sin embargo, esta técnica debe ser ajustada por su parámetro R_c dentro de un rango de 120 a 130. Por lo que se prueban varios valores para etiquetar los datos.

Las técnicas de normalización de datos a probar son:

1. Z-Score
2. Z-Score con Regimen de Operación
3. Min-Max
4. Min-Max con Regimen de Operación

Para la disminución de dimensionalidad de los datos se utilizará la selección de componentes mediante la prueba de diferentes combinaciones de componentes, ya sean configuraciones presentadas en la literatura y otras propuestas. Estas combinaciones están en la siguiente Tabla. Además también se evaluara el uso de la extracción de componentes con ACP.

Tabla 3.1: Combinaciones de Componentes. Tabla propia.

Configuración	ID	Ciclos de Operación	Parámetros de Operación	IHFT	Sensores
A	NO	SI	SI	NO	TODOS
B	NO	NO	SI	NO	TODOS
C	NO	NO	NO	NO	TODOS
D	NO	SI	SI	NO	2,3,4,7,8,9,11,12, 13,14,15,17,20 y 22
E	NO	NO	SI	NO	2,3,4,7,8,9,11,12, 13,14,15,17,20 y 22
F	NO	NO	NO	NO	2,3,4,7,8,9,11,12, 13,14,15,17,20 y 22
G	NO	SI	NO	NO	TODOS
H	NO	SI	NO	NO	2,3,4,7,8,9,11,12, 13,14,15,17,20 y 22
I	NO	NO	NO	SI	TODOS
J	NO	NO	NO	SI	2,3,4,7,8,9,11,12, 13,14,15,17,20 y 22

Adicionalmente, Jain y cols. (2014) ha indicado que existen ciertas unidades de turbina cuya degradación se encuentra avanzada y sólo agregan ruido al entrenamiento, por lo que al retirarlas mejora el pronóstico del algoritmo. Esta prueba también es incluida en esta etapa de preprocesamiento, ya que además de mejorar el pronóstico, la disminución de datos de entrenamiento también ayuda a reducir tiempos de procesamiento durante el entrenamiento.

Para determinar si una turbina produce ruido al entrenamiento, Jain y cols. (2014) observa los valores de ECM de cada unidad de entrenamiento, después de entrenar el modelo, y mediante una Gráfica de Control determina que las turbinas fuera de los Límites de Control son aquellas que agregan ruido. En este trabajo se utiliza la definición de dato atípico dado por la regla de 1.5xIQR, la cual sigue los siguientes pasos:

1. Calcular el rango intercuartil (IQR) de los datos, dado por la diferencia del cuartil 3 y 1.

2. Multiplicar el valor IQR por 1.5

3. Sumar 1.5 x (IQR) al tercer cuartil. Cualquier número mayor que éste es una sospecha de dato atípico

En la Figura 3.5 se ilustra las componentes del experimento para la evaluación de cada una de las técnicas de preprocesamiento. Las variables de entrada son cada una de las técnicas de preprocesamiento, y los 32 conjuntos de pesos iniciales. Los factores controlados son los datos de entrenamiento y validación, la arquitectura e hiperparámetros de la red neuronal, y el criterio de paro (Detención Temprana). El factor no controlado, es el número de épocas de entrenamiento, consecuencia del método de Detención Temprana. Las variables de salida son las métricas Error Cuadrático Medio (ECM) y Penalización Exponencial del Error (PEE) en los datos de validación.

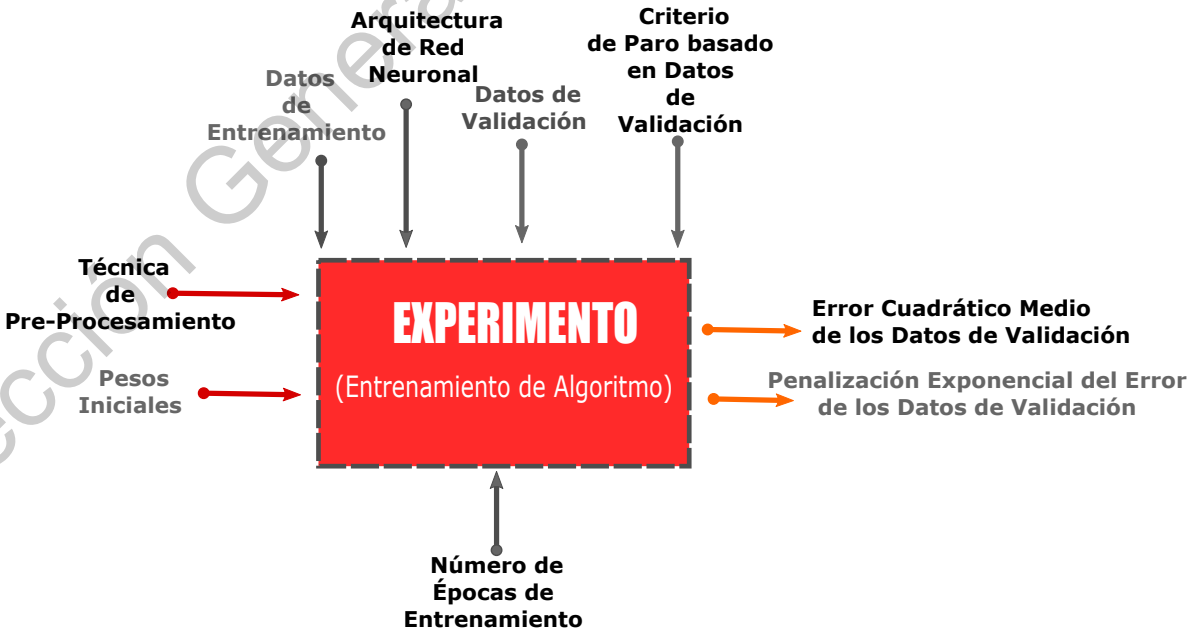


Figura 3.5: Diagrama del experimento para la selección de técnicas de preprocesamiento. Imagen propia.

3.0.4. Modelado: Selección de Hiperparámetros

Tras encontrar las mejores técnicas de preprocesamiento para el modelo inicial. El siguiente paso es el ajuste de los hiperparámetros del modelo. Así como lo ilustra la Figura 3.6. El primer paso es la búsqueda del optimizador más adecuado, después los parámetros del optimizador, la función de activación de la capa oculta ,el número de neuronas de la capa oculta, y finalmente la técnica de inicializaión de pesos.

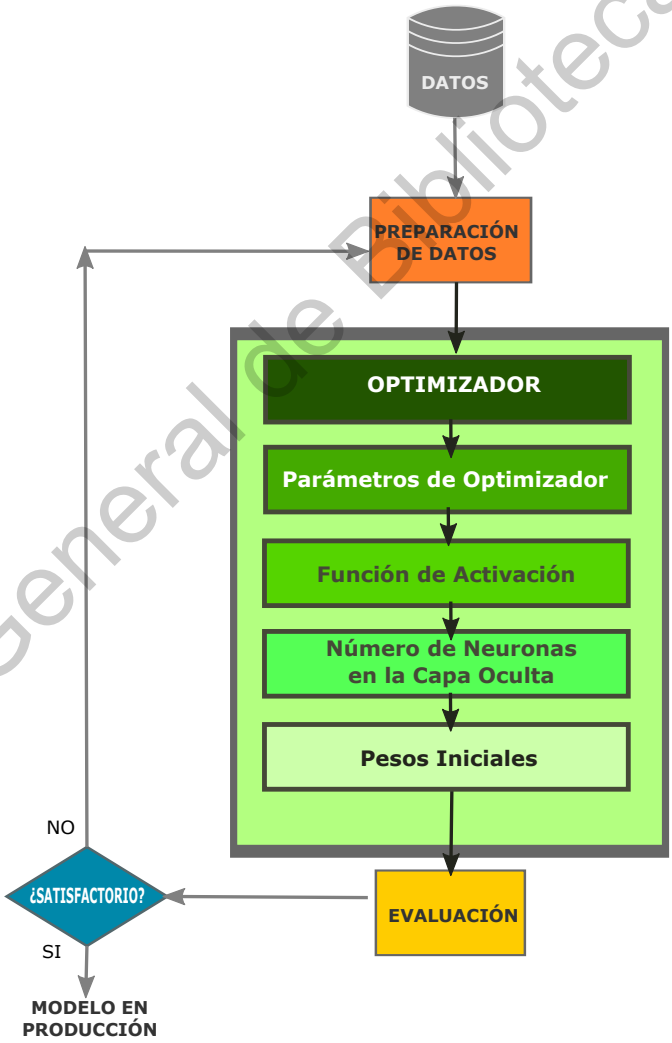


Figura 3.6: Etapas de modelado del algoritmo de Aprendizaje Automático. Imagen propia.

Los optimizadores a considerar y sus parámetros están descritos en la siguiente Tabla.

Tabla 3.2: Optimizadores. Tabla propia.

Optimizador	Parámetro	Valor Predeterminado (Recomendado)
Gradiente Descendente (GD)	Tasa de Aprendizaaje	0.01
Momentum	Tasa de Aprendizaaje	0.01
	Momentum	0.9
Nesterov	Tasa de Aprendizaaje	0.01
	Momentum	0.9
Quasi-Newton (QN)	Factor de escalameinto para determinar el desempeño	0.001
	Factor de escalamiento para determinar el tamaño de paso	0.1
Levenberg-Marquardt (LM)	Tasa de Aprendizaaje inicial	0.001
Adadelta	Tasa de Aprendizaaje inicial	1
	Fracción de gradiente para mantener en cada época	0.95
RMSprop	Tasa de Aprendizaaje inicial	0.001
	Fracción de gradiente para mantener en cada época	0.9
Adam	Tasa de Aprendizaaje inicial	0.001
	Beta 1	0.9
	Beta 2	0.999

Se prueba cada optimizador con los valores recomendados por el soft-

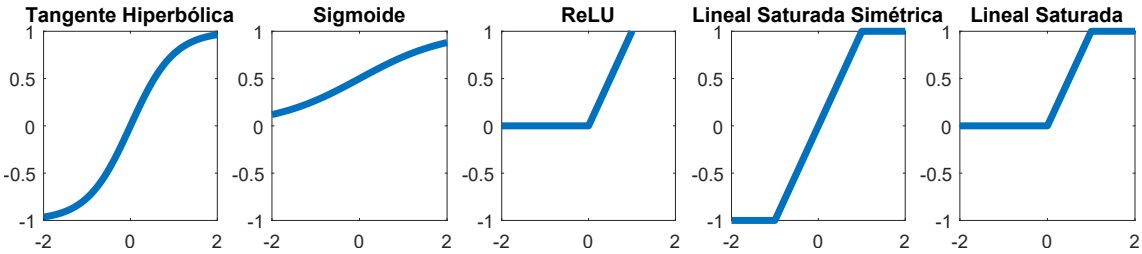


Figura 3.7: Funciones de Activación. Imagen propia.

ware y posteriormente se modifican los parámetros del mejor optimizador para encontrar un mejor resultado.

Las funciones de activación a considerar son las siguientes (ver Figura 3.7):

- Tangente Hiperbólica
- Unidad Lineal Rectificada (ReLU)
- Sigmoide
- Lineal Saturada
- Lineal Saturada Simétrica

El número de neuronas a probar son 10, 20, 25, 30 y 35. En caso de ser necesario se aumenta el número de pruebas. Sin embargo, se espera que el desempeño no escale más allá de 30 neuronas.

En cuanto a los pesos iniciales se compara la inicialización de pesos con números aleatorios contra la inicialización utilizando la técnica Nguyen-Widrow (Nguyen y Widrow, 1990).

En la Figura 3.8 se ilustra las componentes del experimento para la búsqueda de hiperparámetros. Los factores controlados son las técnicas de preprocesa-

meinto, los datos de entrenamiento y validación, el criterio de paro basado en los datos de validación. El factor no controlado, es el número de épocas de entrenamiento, consecuencia del método de Detención Temprana. Y las salidas son los errores de ECM y PEE de los datos de validación.

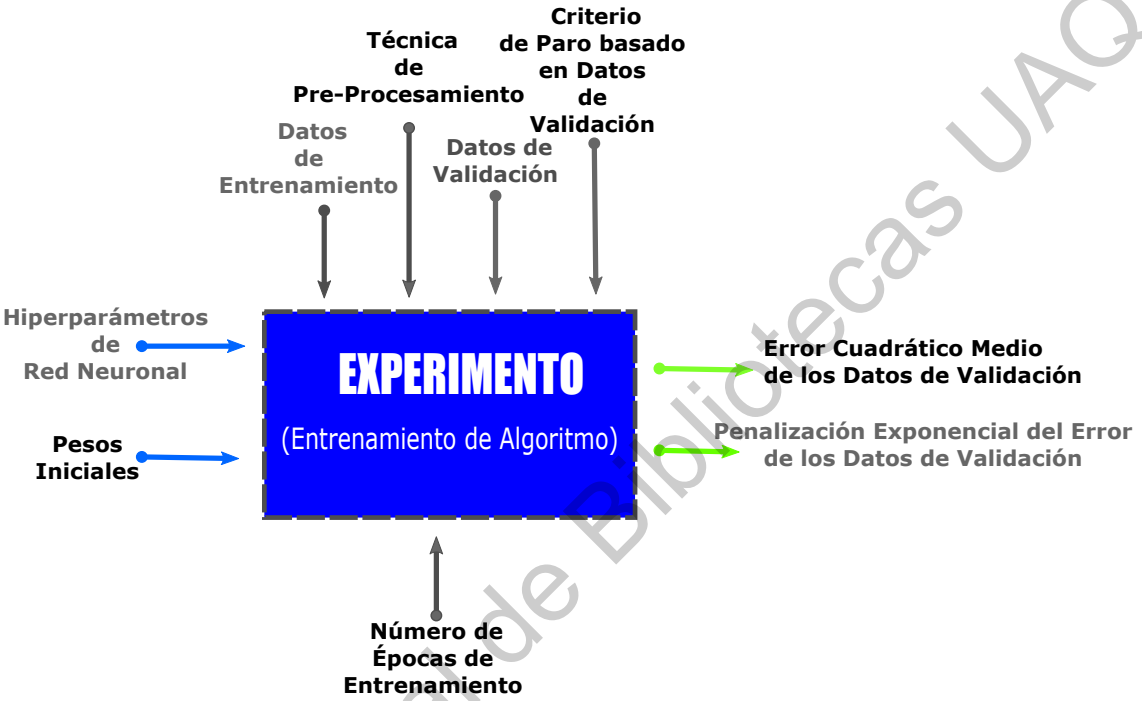


Figura 3.8: Diagrama del experimento para la selección de hiperparámetros. Imagen propia.

3.0.5. Posprocesamiento

Finalmente tras encontrar los mejores parámetros de la red neuronal y las mejores técnicas de preprocesamiento se prueba el uso del Filtro de Kalman como una técnica de posprocesamiento de la predicción hecha por la Red Neuronal. Esto con el propósito de mitigar el problema de predicciones ruidosas.

3.0.6. Neuroevolución

El algoritmo de neuroevolución a probar es el propuesto por Real y cols. (2018). Este algoritmo es simple de implementar y ha demostrado buenos resultados para la búsqueda de arquitecturas en redes neuronales tipo RNC para la clasificación de imágenes. Incluso este algoritmo ha mostrado ventajas sobre tradicionales algoritmos evolutivos como Algoritmos Genéticos y también Aprendizaje por Refuerzo para la búsqueda de arquitecturas de redes neuronales.

El algoritmo de neuroevolución es el siguiente:

Neuroevolución
1: $poblacion \leftarrow$ cola vacía
2: $historia \leftarrow \emptyset$
3: **while** $|poblacion| < P$ **do**
4: $modelo.arquitectura \longleftarrow ArquitecturaAleatoria()$
5: $modelo.error \longleftarrow EntrenarModelo(modelo.arquitectura)$
6: Agregar $modelo$ a $poblacion$
7: Agregar $modelo$ a $historia$
8: **end while**
9: **while** $|historia| < C$ **do**
10: $muestra \longleftarrow \emptyset$
11: **while** $|muestra| < S$ **do**
12: $candidato \longleftarrow$ elemento aleatorio de $poblacion$
13: Agregar $candidato$ a $muestra$
14: **end while**
15: $padre \longleftarrow$ elemento de $muestra$ con el menor error
16: $hijo.arquitectura \longleftarrow Mutacion(padre.arquitectura)$
17: $hijo.error \longleftarrow EntrenarModelo(hijo.arquitectura)$
18: Agregar $hijo$ al inicio de $poblacion$
19: Agregar $hijo$ a $historia$
20: Eliminar el elemento de al final de $poblacion$
21: **end while**
22: **return** Elemento de $historia$ con el menor error

El parámetro P determina el tamaño de la población, C es el número de iteraciones del algoritmo, S controla la agresividad de la exploración y debe estar

en un rango [2,P].

Las mutaciones deben ser simples para que el evolutivo converja. Real y cols. (2018) definen las mutaciones específicamente para redes RNC. Dado que en este trabajo se usa redes MLP. Las mutaciones quedan definidas como las siguientes:

- Agregar una neurona a la capa oculta
- Remover una neurona a la capa oculta
- Cambiar el valor de Tasa de Aprendizaje de forma aleatoria dentro de un rango [a,b].
- Cambiar la función de activación de la capa oculta
- Cambiar la función de activación de la capa de salida

Además se agrega una mutación 'identidad', la cual no modifica a el padre y simplemente es pasado a la siguiente generación. Esta mutación tiene una probabilidad del 5 %.

CAPÍTULO 4

Resultados

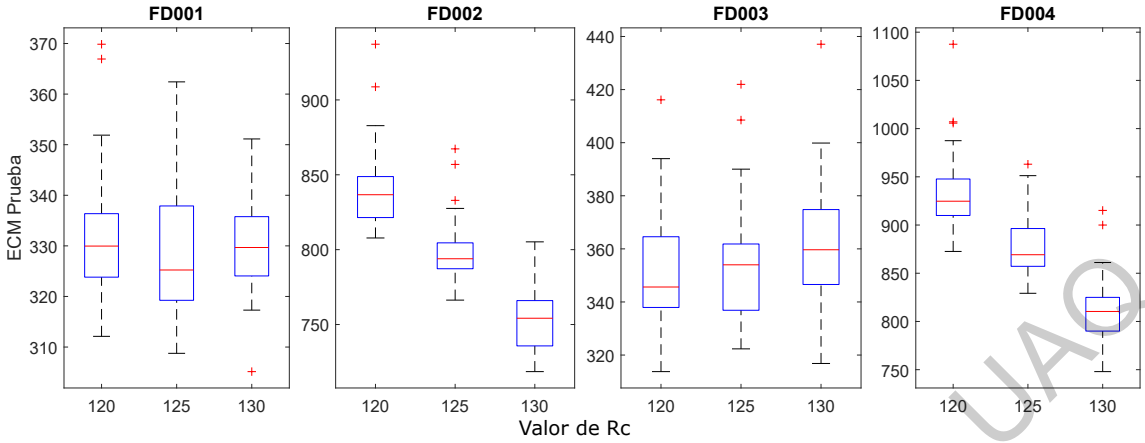
"Nada en la vida es para ser temido, es sólo para ser comprendido. Ahora es el momento de entender más, de modo que podamos temer menos"

Marie Curie

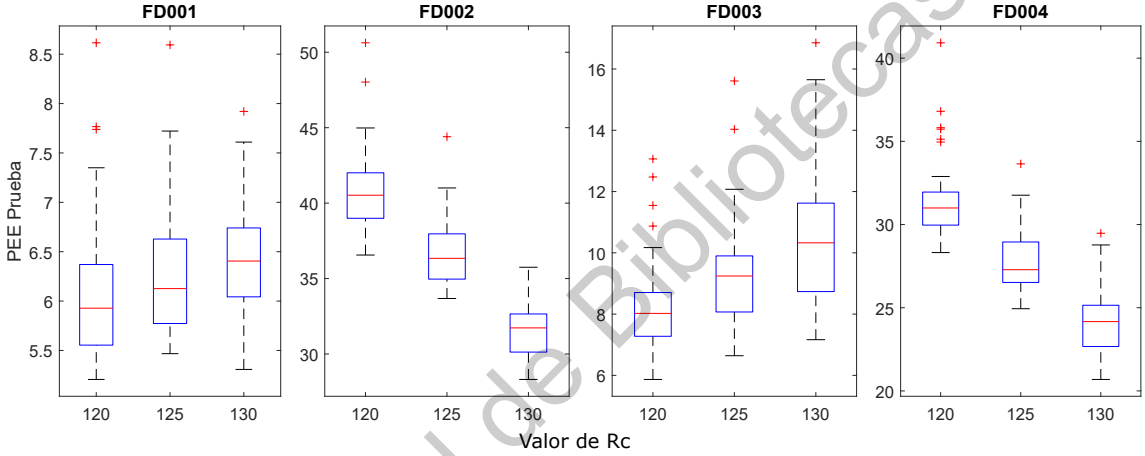
4.1. Preprocesamiento

4.1.1. Métodos de Etiquetado

En las Figuras 4.1 y 4.2 se muestran los resultados del modelo utilizando la técnica de etiquetado por Heimes (2008), usando valores de R_c iguales a: 120, 125 y 130. La Figura 4.1 ilustra los resultados para el modelo evaluado en cada uno de los conjuntos de prueba. El comportamiento de los resultados es similar utilizando ambos métricas. Es claro que para las pruebas FD002 y FD003, el mejor resultado se obtiene al etiquetar toda la base de datos utilizando $R_c = 130$. Por otro lado, para las pruebas FD001 y FD003 no es muy clara la diferencia al observar la métrica de ECM (Figura 4.1a), pero en la métrica PEE (Figura 4.1b) es apreciable que se obtiene mejores resultados con $R_c = 120$.



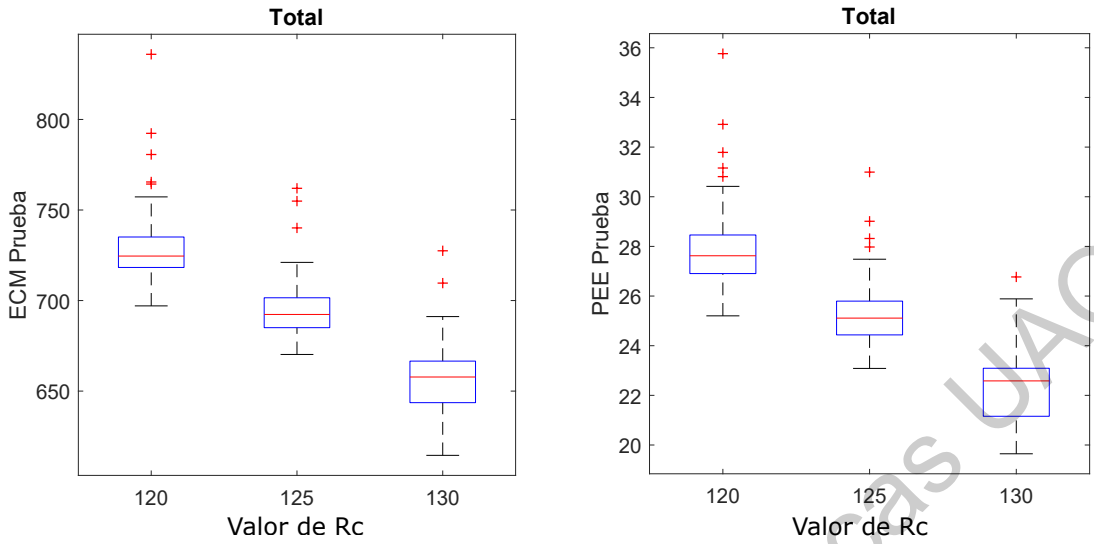
(a) Métrica ECM.



(b) Métrica PEE.

Figura 4.1: Gráfico de caja para evaluación individual del modelo usando diferentes valores de R_c. Imagen propia.

Al observar el comportamiento del modelo, combinando los 4 conjuntos de prueba, (Figura 4.2) se aprecia que el mejor resultado se obtiene mediante el uso de $R_c = 130$. Esto es debido a que la base de datos tiene una mayor cantidad de muestras de prueba FD002 y FD004 que FD001 y FD003. Por lo que al mejorar estos dos conjuntos se mejora el comportamiento general.

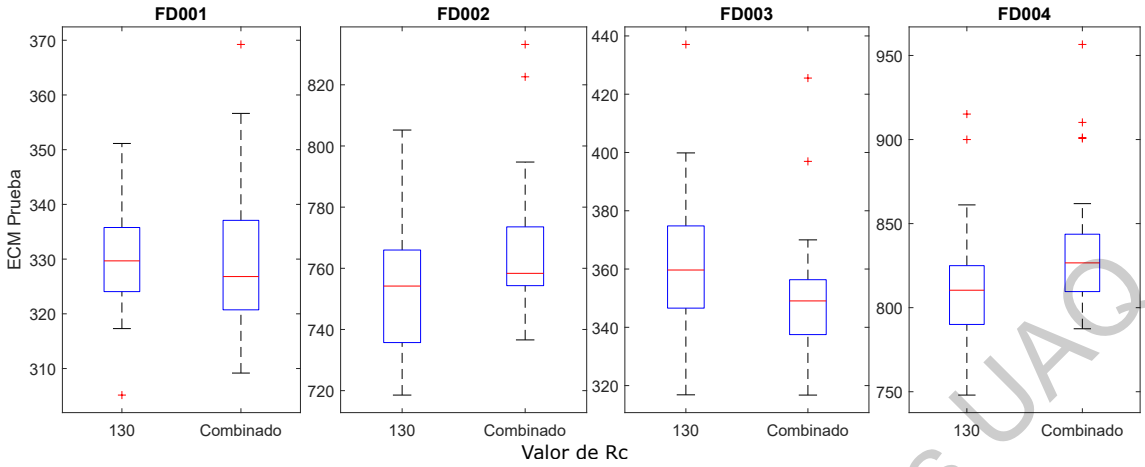


(a) Métrica ECM. (b) Métrica PEE.

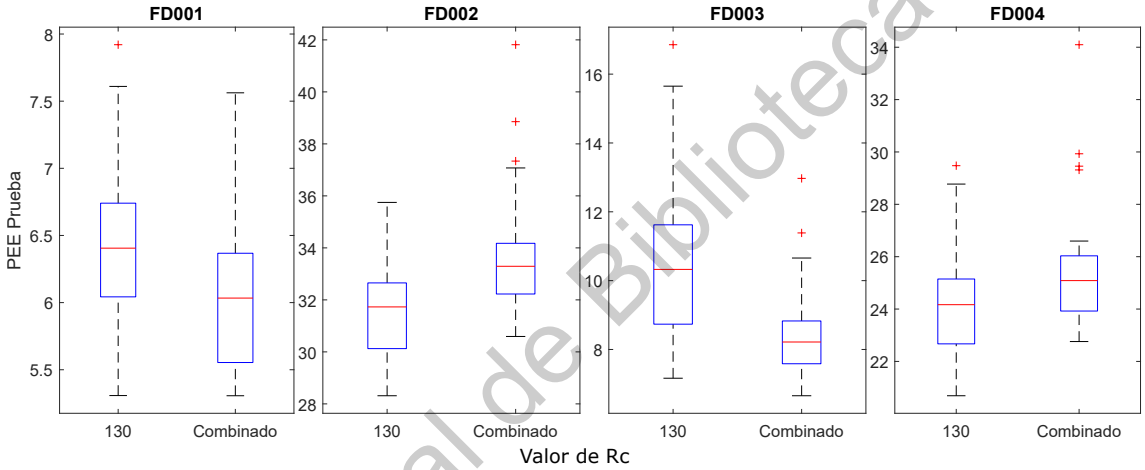
Figura 4.2: Gráfico de caja para evaluación total del modelo usando diferentes valores de R_c. Imagen propia.

Con el objetivo de explorar más el comportamiento del modelo con el parámetro R_c, se hace una prueba etiquetando los conjuntos de entrenamiento FD001 y FD003 con $R_c = 120$ y los otros dos con $R_c = 130$. En las Figuras 4.3 y 4.4 se hace la comparación de este modelo llamado 'combinado' con el modelo donde se etiqueto toda la base de datos con $R_c = 130$.

En las evaluaciones generales (Figura 4.4) se puede observar que el grupo 'combinado' tiene un desempeño un poco similar al de $R_c = 130$, aunque este último sigue pareciendo superior debido a que tiene valores de medianas y mínimos más favorables. En las evaluaciones individuales (Figura 4.3), los errores de los conjuntos FD001 y FD003 han disminuido un poco y los errores de FD002 Y FD004 han aumentado de igual forma. Sin embargo, considerando que hay más muestras de los conjuntos FD002 y FD004, el pequeño aumento de error en estos dos conjuntos se observa en la evaluación general del algoritmo, resultando el modelo 'combinado' inferior.

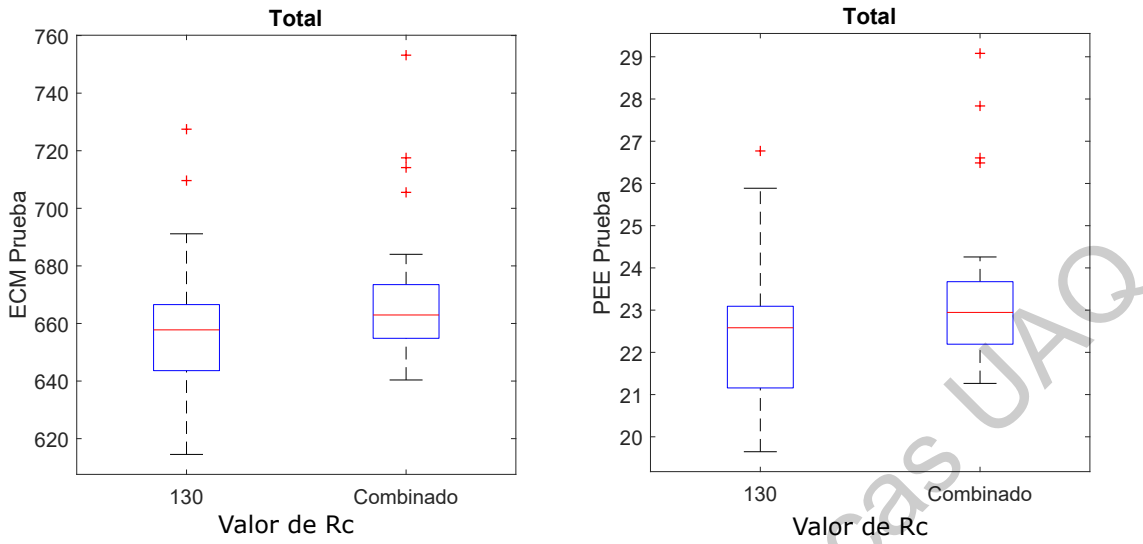


(a) Métrica ECM.



(b) Métrica PEE.

Figura 4.3: Gráfico de caja de la evaluación individual del modelo utilizando diferentes valores de R_c. Experimento 2. Imagen propia.



(a) Métrica ECM. (b) Métrica PEE.

Figura 4.4: Gráfico de caja de la evaluación total del modelo utilizando diferentes valores de R_c. Experimento 2. Imagen propia.

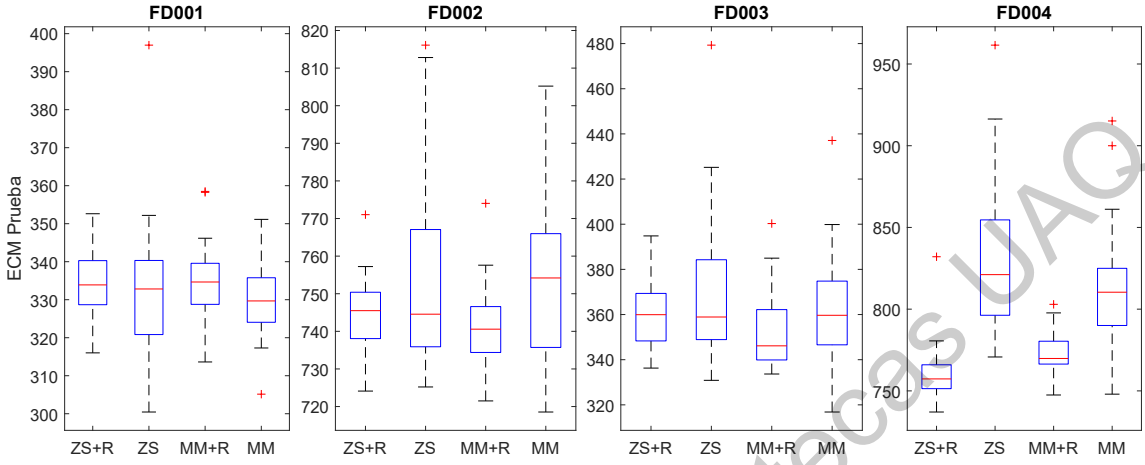
4.1.2. Técnicas de Normalización

Los siguientes resultados son considerando la base de datos etiquetada con $R_c = 130$.

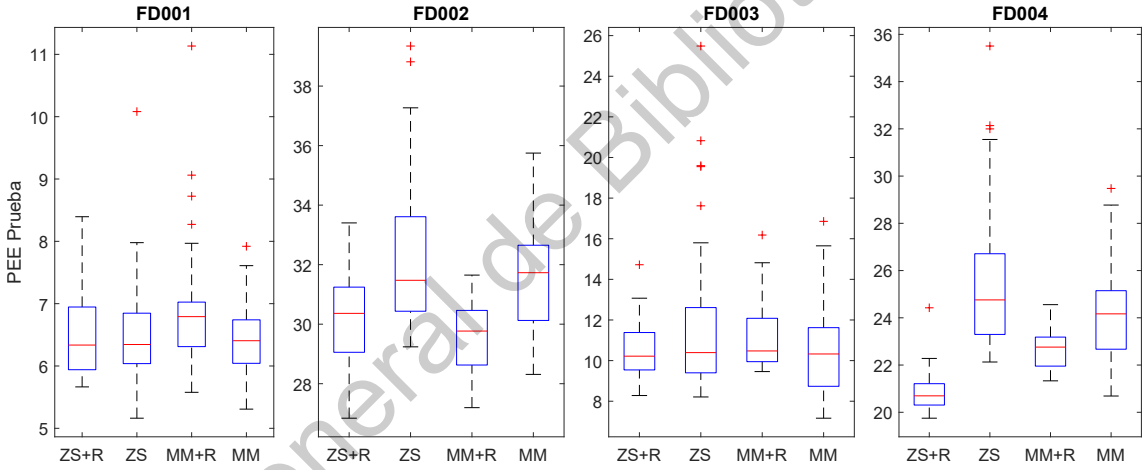
Observando la evaluación del modelo en cada conjunto de prueba (Figura 4.5) se aprecia que, utilizando ambas métricas, las pruebas FD001 y FD003 no muestran diferencia significativa con respecto a la técnica de normalización. Por otro lado, para las pruebas FD002 y FD004 se aprecia superioridad de las técnicas de normalización con régimen (ZS+R y MM+R) sobre sus versiones tradicionales (ZS y MM).

En la Figura 4.6 se observa un interesante comportamiento por parte de Z-Score con Régimen, logrando la mediana y varianza más pequeña. Por otro lado, Min-Max tiene el mínimo más bajo, mientras que tiene una de las medianas y varianzas más altas. Debido a que Z-Score tiene mejores valores en dos pro-

piedades y Min-Max sólo en una, es ésta primera la mejor opción para normalizar la base de datos.

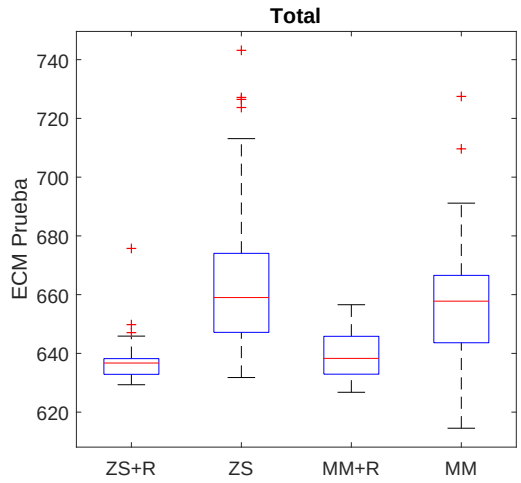


(a) Métrica ECM.

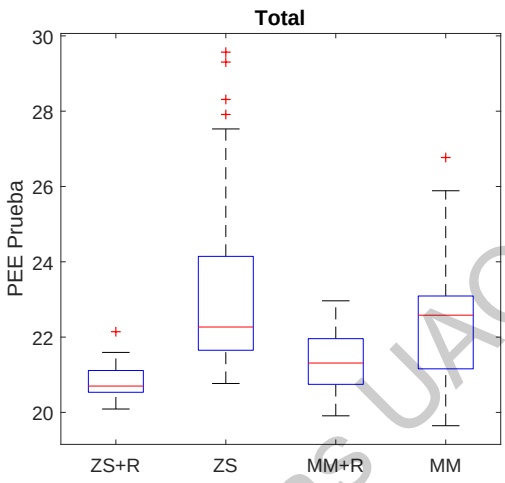


(b) Métrica PEE.

Figura 4.5: Gráfico de caja de la evaluación individual del modelo utilizando las siguientes técnicas de normalización: Z-Score con Régimen (ZS+R), Z-Score (ZS), Min-Máx con Régimen (MM+R) y Min-Max (MM). Imagen propia.



(a) Métrica ECM.

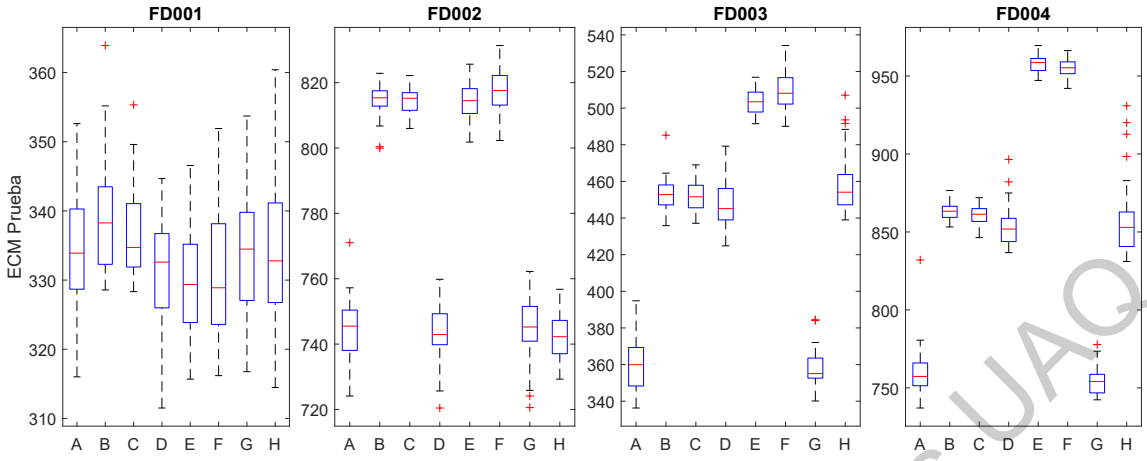


(b) Métrica PEE.

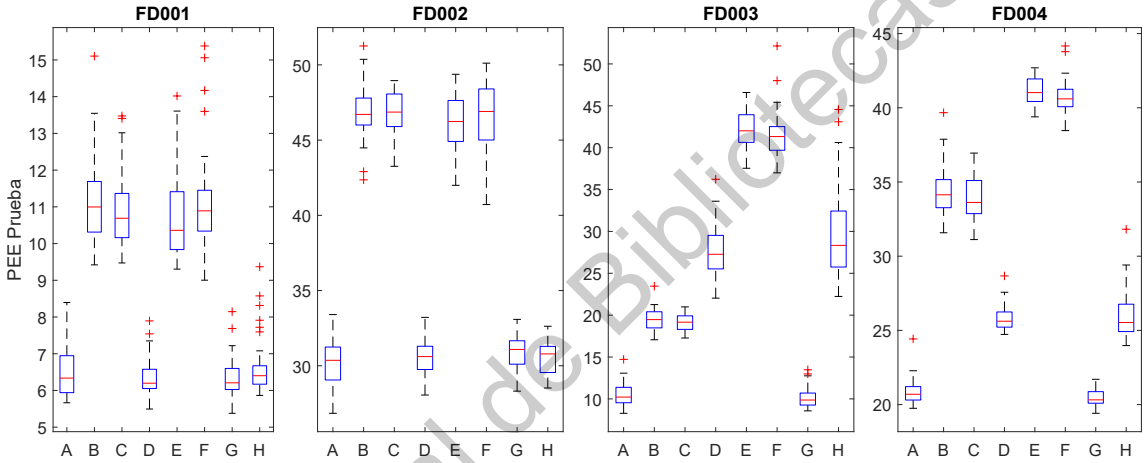
Figura 4.6: Gráfico de caja de la evaluación general del modelo utilizando las siguientes técnicas de normalización: Z-Score con Régimen (ZS+R), Z-Score (ZS), Min-Máx con Régimen (MM+R) y Min-Max (MM). Imagen propia.

4.1.3. Reducción de Dimensionalidad

Los siguientes resultados son con la base de datos etiquetada utilizando $R_c = 130$ y normalizandolos con Z-Score con Régimen.



(a) Métrica ECM.

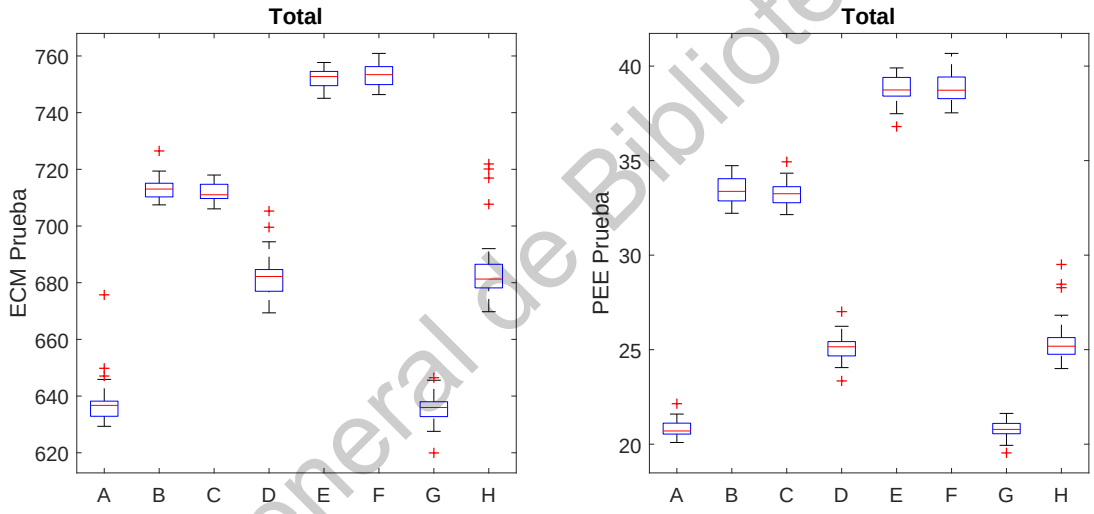


(b) Métrica PEE.

Figura 4.7: Gráfico de caja de la evaluación individual del modelo utilizando las diferentes configuraciones de componentes. Imagen propia.

La selección de componentes demuestra ser de las características más importantes en el desempeño del modelo. En las evaluaciones individuales (Figura 4.7) se aprecia que cada conjunto prueba tiene un comportamiento diferente con respecto a las configuraciones. El conjunto de prueba FD001, el más simple de todos, presenta un comportamiento independiente de la configuración, al usar la métrica ECM, sin embargo, al utilizar la métrica PEE, se observa que las configuraciones con ciclos de vuelo (A, D, G y H) producen un mejor desempeño. Este comportamiento también lo muestra el conjunto de prueba FD002, pero en

ambas métricas. Por otro lado, para las pruebas FD003 y FD004 presentan similar comportamiento. Se observa primeramente que el uso de las 3 condiciones de operación no tiene efecto, se aprecia al comparar los grupos A contra G, B contra C, D contra H y E contra F. También el uso del ciclo de vuelo vuelve a ser indispensable para obtener buenos resultados, ésto se aprecia al comparar A contra B, D contra E, C contra G y F contra H. Y por último, el uso de todos los sensores es mejor a sólo utilizar aquellos con comportamiento ascendente o descendente, y es apreciado en las comparaciones de A contra D, B contra E, C contra F y G contra H.



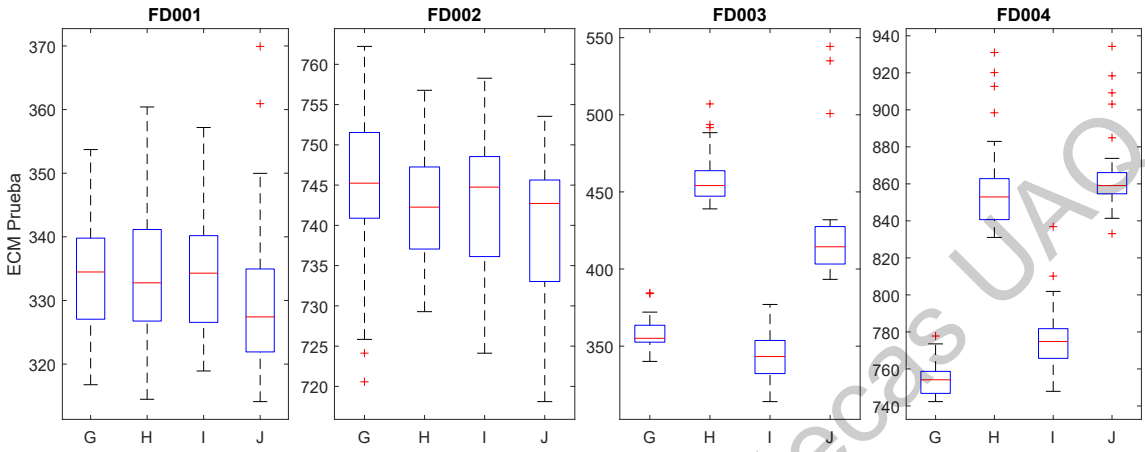
(a) Métrica ECM. (b) Métrica PEE.

Figura 4.8: Gráfico de caja de la evaluación total del modelo utilizando diferentes configuraciones de componentes. Imagen propia.

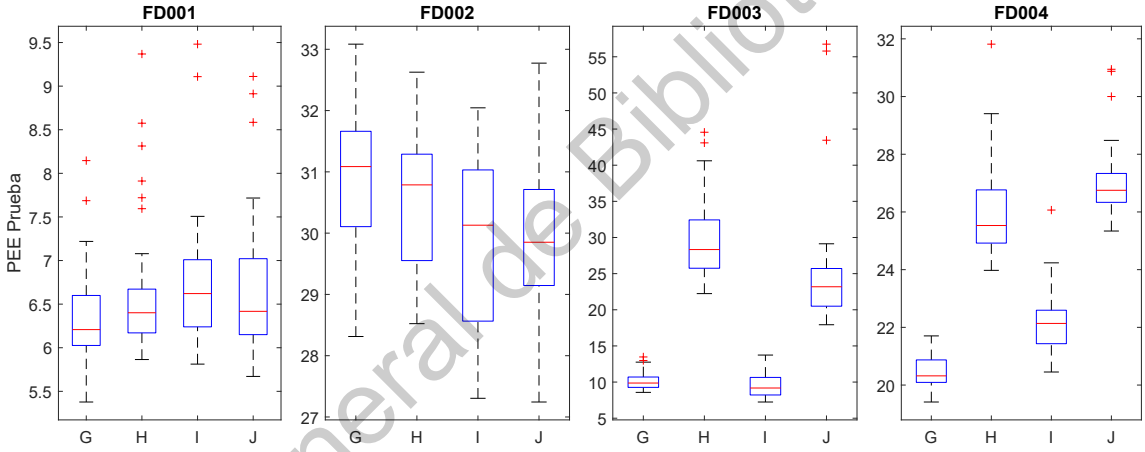
Respecto a la evaluación general (Figura 4.8), el comportamiento del modelo está gobernado por los resultados en los conjuntos de prueba FD003 y FD004, siendo así que las componentes más importantes son: el uso de ciclos de vuelo, y todos los sensores.

Una comparación individual se hace con respecto a los grupos G, H, I y J.

Los modelos G y H usan los ciclos de vuelo como indicador temporal, mientras que los modelos I y J usan las 6 variables IHFT como indicador temporal.



(a) Métrica ECM.

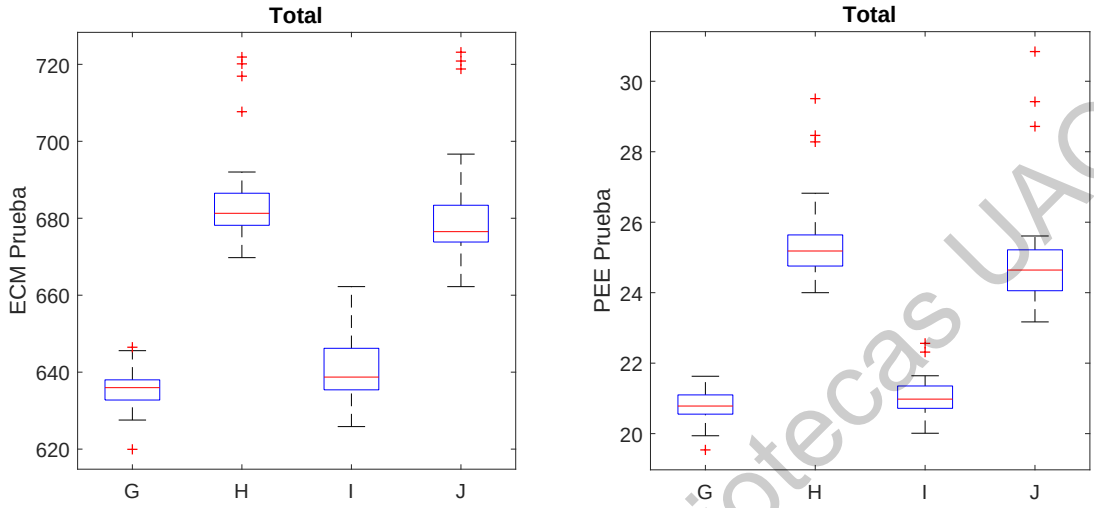


(b) Métrica PEE.

Figura 4.9: Gráfico de caja de la evaluación individual del modelo utilizando las configuraciones G, H, I y J. Imagen propia.

En la evaluación individual (Figura 4.9) se observa nuevamente que las el modelo muestra una diferencia notable en las pruebas FD003 y FD004. De estas se aprecia, al comparar G contra I y H contra J, que el uso del ciclo de vuelo como indicador temporal es equivalente a el uso de las 6 variables IHFT. Es aun más notable esta afirmación al observar el comportamiento general del modelo (Figura 4.10). De esta forma se concluye que es conveniente utilizar los

ciclos de vuelo en lugar de los indicadores IHFT ya que estos agregar 6 variables , mientras que los ciclos de vuelo sólo una.



(a) Métrica ECM.

(b) Métrica PEE.

Figura 4.10: Gráfico de caja de la evaluación total del modelo utilizando las configuraciones G, H, I y J. Imagen propia.

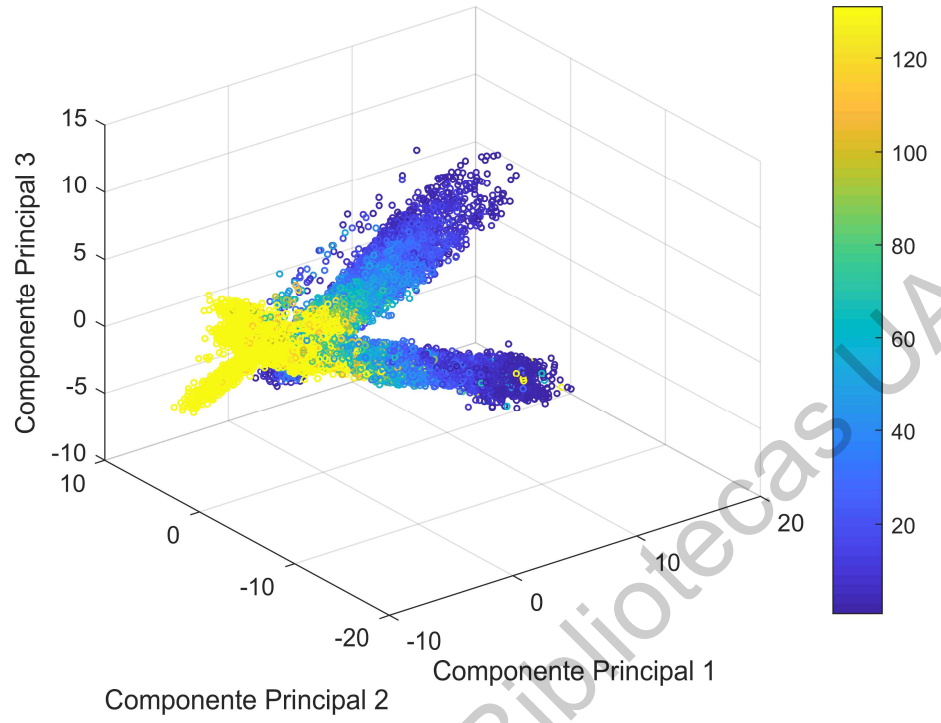
Análisis de Componentes Principales

Al aplicar ACP a los datos normalizador por medio de Z-Score con Regimen obtenemos los resultados listados en la Tabla 4.1.

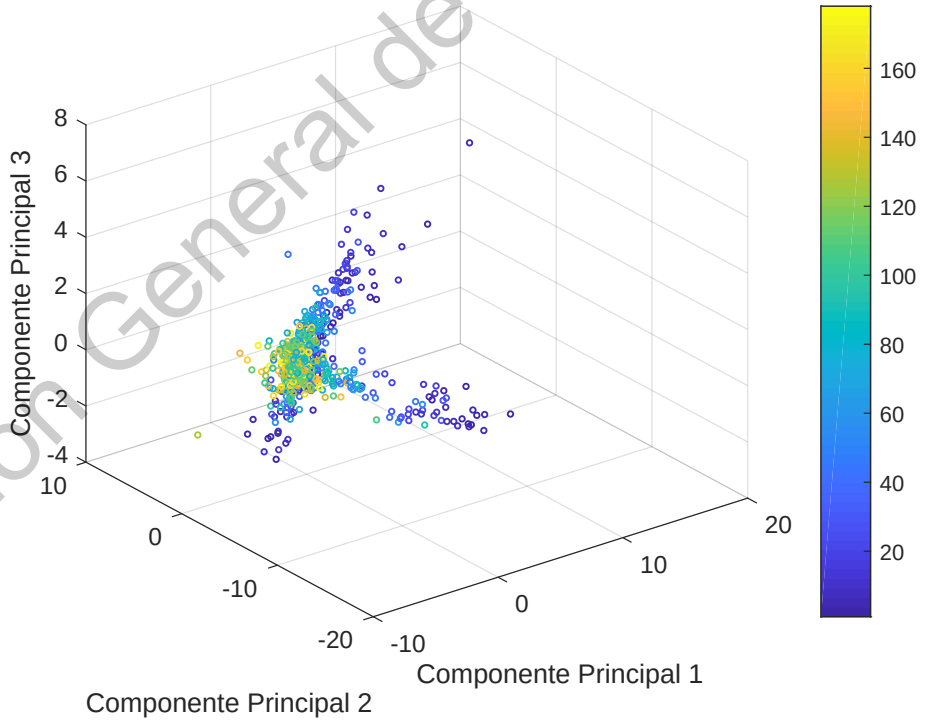
Además de reducir la dimensión de los datos para acelerar el entrenamiento y mejorar los resultados, ACP también es utilizado para la visualización de éstos. En la Figura 4.11 se observa la dispersión de los datos al utilizar tres componentes principales que mantienen el 66.265 % de la varianza de los datos originales.

Tabla 4.1: Eigenvalores de los Datos. Tabla propia.

Componente Principal	Eigenvalor	Porcentaje
1	5.8639	32.181
2	5.0226	27.564
3	1.1716	6.430
4	1.0682	5.862
5	0.9315	5.112
6	0.7329	4.022
7	0.6176	3.389
8	0.4195	2.302
9	0.4135	2.269
10	0.3985	2.187
11	0.3700	2.031
12	0.2864	1.572
13	0.2721	1.493
14	0.2048	1.124
15	0.1684	0.924
16	0.1163	0.638
17	0.0828	0.454
18	0.0431	0.237
19	0.0350	0.192
20	0.0031	0.017
21	1.54E-19	8.4322E-19
22	3.04E-24	1.6705E-23
23	3.34E-25	1.8344E-24
24	3.38E-34	1.8534E-33
25	5.94E-57	3.2607E-56
Total	18.2218	100



(a) Datos de Entrenamiento.



(b) Datos de Prueba.

Figura 4.11: Gráfico de dispersión de los datos utilizando 3 Componentes Principales. Imagen propia.

En la Figura 4.12 se muestra el porcentaje acumulado de las componentes. En la literatura se suele utilizar las componentes que guarden el 95 % de la varianza de los datos originales. De esta forma vemos que con 13 componentes podemos conservar 96.41 % de la varianza original.

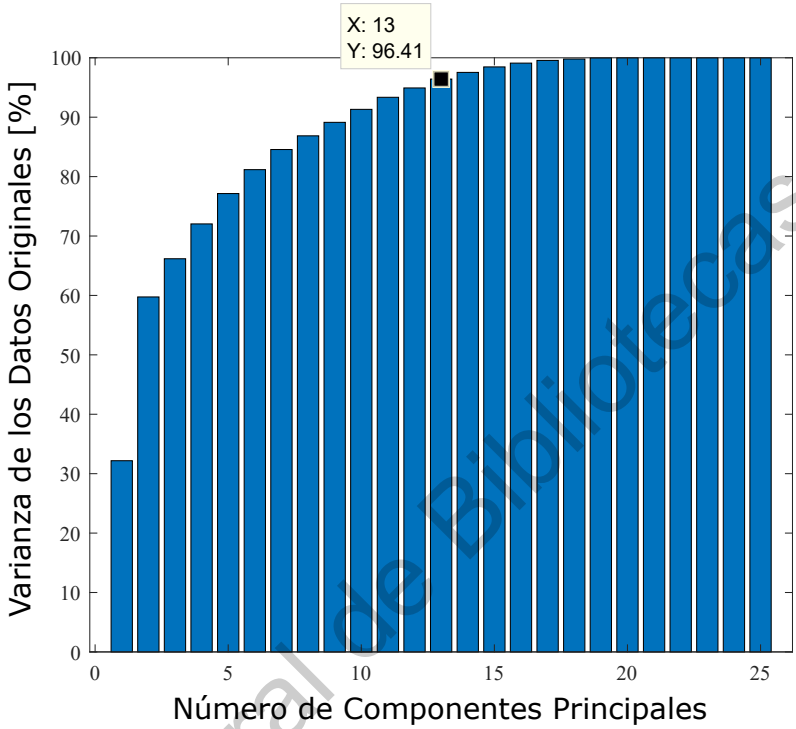
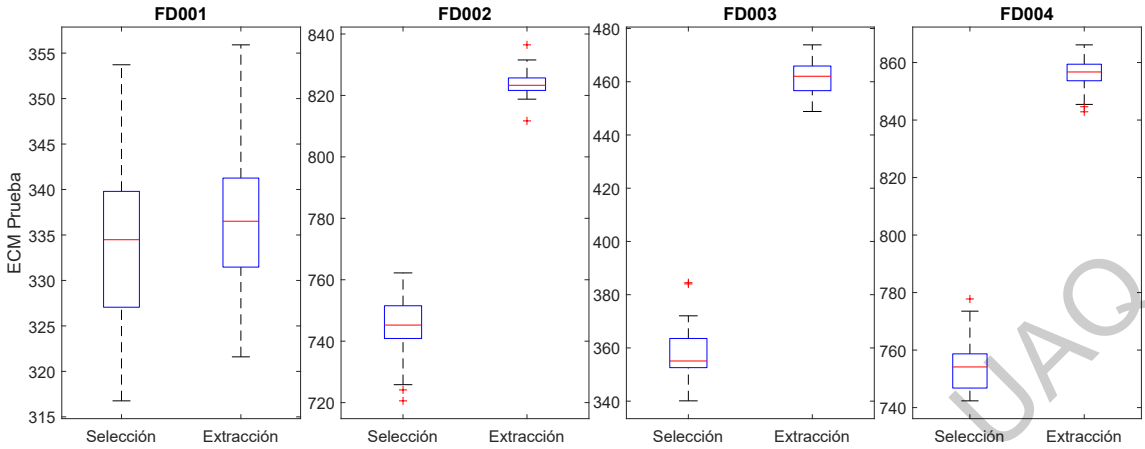
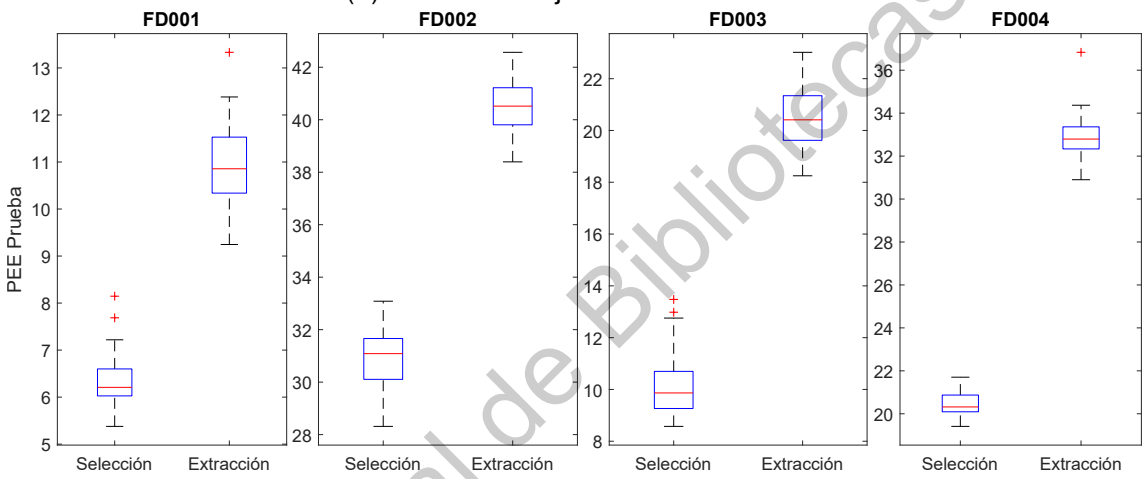


Figura 4.12: Gráfico de porcentajes acumulados de las Componentes Principales. Imagen propia.

En las Figura 4.13 y 4.14 se ilustra la comparación de entrenar el modelo de la red neuronal con estas 13 componentes principales y con la selección de características (configuración G).



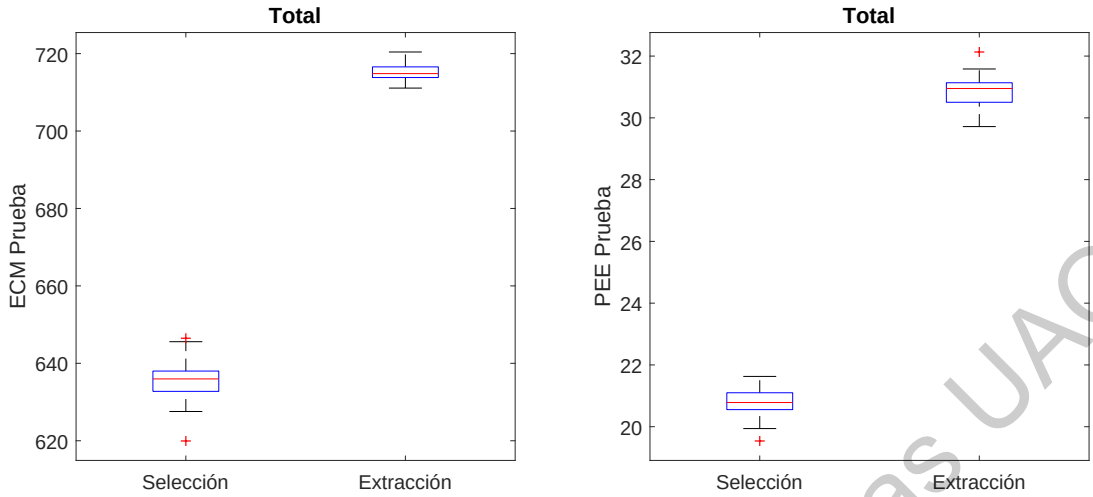
(a) ECM del conjunto de validación.



(b) PEE del conjunto de validación.

Figura 4.13: Gráfico de caja de la evaluación individual del modelo utilizando la Selección de Características (configuración G) contra la Extracción de Características (ACP). Imagen propia.

Tanto en la evaluación individual como en la total, la selección de características es por mucho una mejor opción para la reducción de dimensionalidad de los datos. Cabe mencionar que esto es debido a que el ACP es un algoritmo no supervisado, y solo busca incrementar la varianza de los datos sin tomar en cuenta la separabilidad de las diferentes clases dentro de los datos.



(a) ECM del conjunto de validación. (b) PEE del conjunto de validación.

Figura 4.14: Gráfico de caja de la evaluación total del modelo utilizando la Selección de Características (configuración G) contra la Extracción de Características (ACP). Imagen propia.

Unidades con Desgaste Avanzado

Los resultados siguientes son tilizando el modelo con $R_c = 130$, normalización Z-Score con Régimen y con las componentes de ciclo de vuelo y los 21 sensores.

Como se ha mencionado anteriormente el modelo es entrenado 32 veces debido a que se prueba con diferentes valores iniciales en sus parámetros (pesos de la red neuronal). En cada una de las 32 muestras de analiza las unidades de entrenamiento que parecen tener un valor de ECM atípico. La Figura 4.15 muestra la frecuencia con la que cada unidad fue detectada como atípica, siguiendo la regla 1.5xIQR. Para determinar finalmente si la unidad representa una turbina atípica, ésta debe de tener una frecuencia de aparición mayor o igual al 50 % de las repeticiones totales, en otras palabras, debe de haber aparecido, como valor atípico, un número de 16 veces o más.

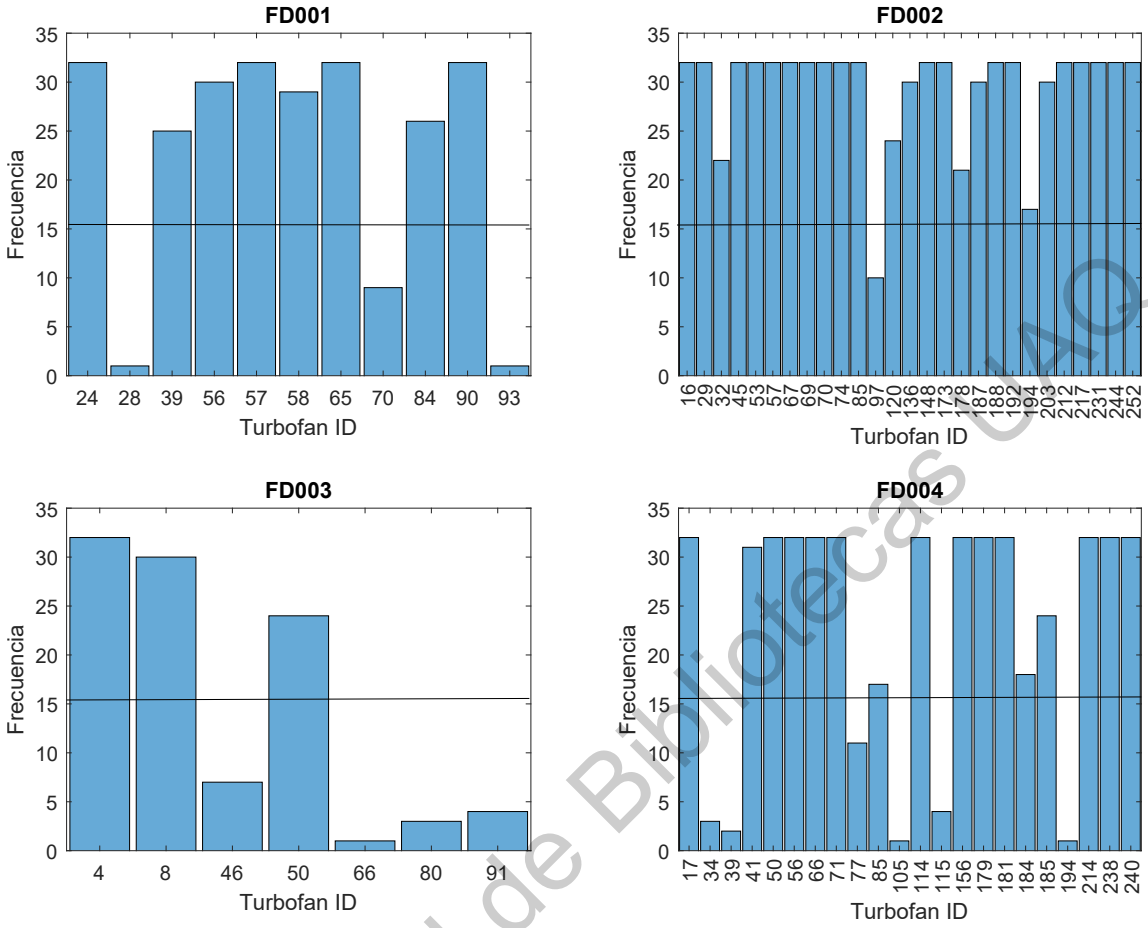
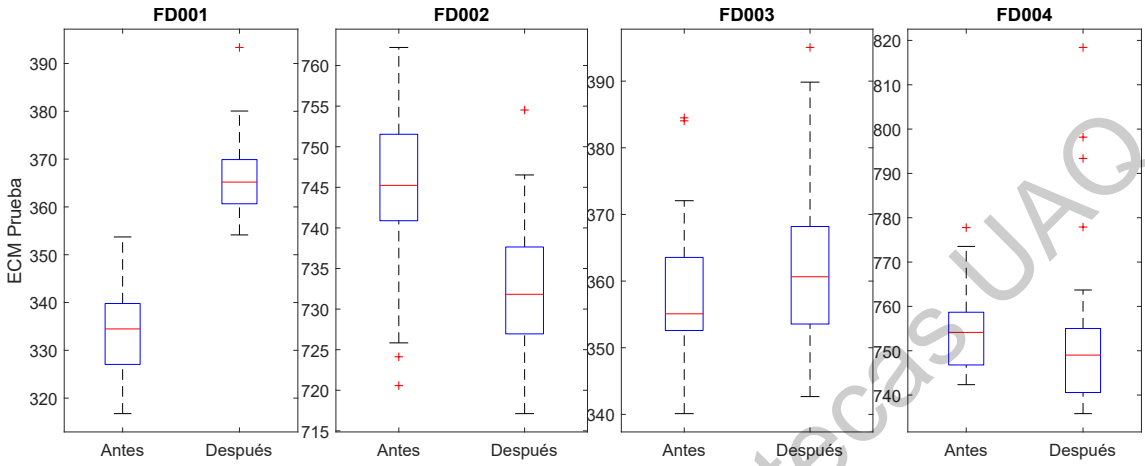


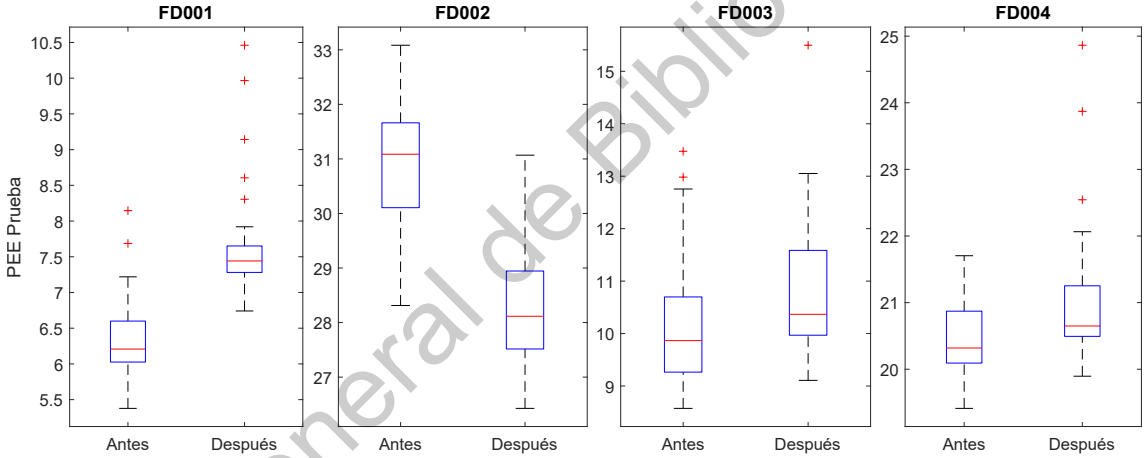
Figura 4.15: Frecuencia de unidades atípicas de entrenamiento. Un umbral es dibujado a la frecuencia de 16. Las unidades con frecuencias arriba del umbral son consideradas como unidades atípicas. Imagen propia.

A continuación se hace la comparación del modelo antes y después de remover las unidades atípicas. La Figura 4.17 muestra la evaluación del modelo de forma total y se aprecia una mejora considerablemente buena después de remover las unidades atípicas. Al observar a las evaluaciones individuales (Figura 4.16), se aprecia que la prueba más beneficiada es la FD002, mientras que la más perjudicada es la FD001. Para las pruebas FD003 y FD004 se observa también un aumento en sus errores, pero menor. En conclusión, la evaluación total del modelo se ve gobernada por los resultados en la prueba FD002 y poco afectada por el incremento en los errores de las otras 3 pruebas. Y ya que se

busca un buen desempeño en general del modelo, se remueven dichas unidades atípicas en las posteriores pruebas.

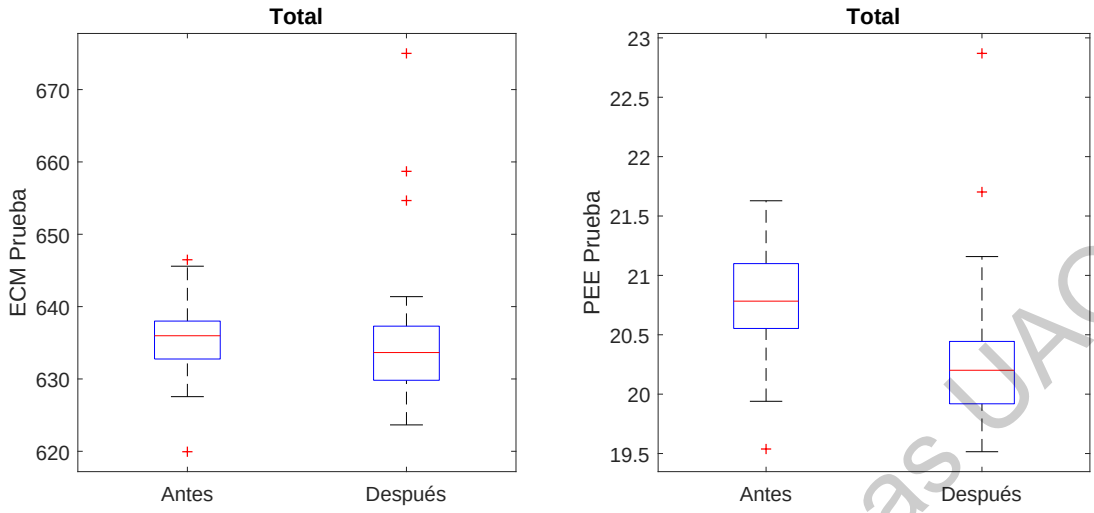


(a) ECM del conjunto de validación.



(b) PEE del conjunto de validación.

Figura 4.16: Comparación de grupos antes y después de eliminar unidades con desgaste atípico. Imagen propia.



(a) ECM del conjunto de validación. (b) PEE del conjunto de validación.

Figura 4.17: Comparación de grupos antes y después de eliminar unidades con desgaste atípico. Imagen propia.

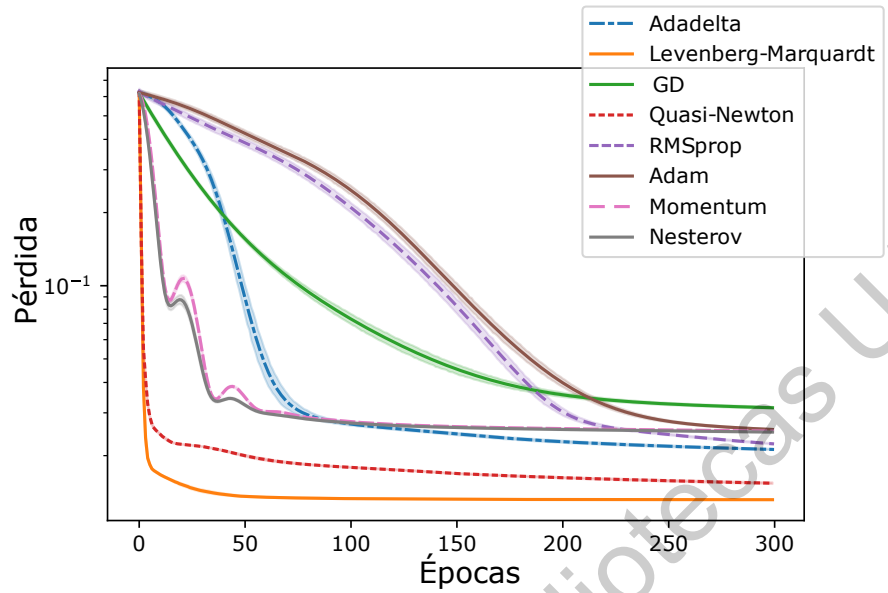
4.2. Modelado: Selección de Hiperparámetros

Las pruebas a continuación toman en cuenta la base de datos etiquetada con $R_c = 130$, normalizada con Z-Score con Régimen, las componentes de ciclo de vuelo y 21 sensores y además se han removido de la base de datos las unidades de turbofan (de entrenamiento) que tienen valores de ECM atípicos.

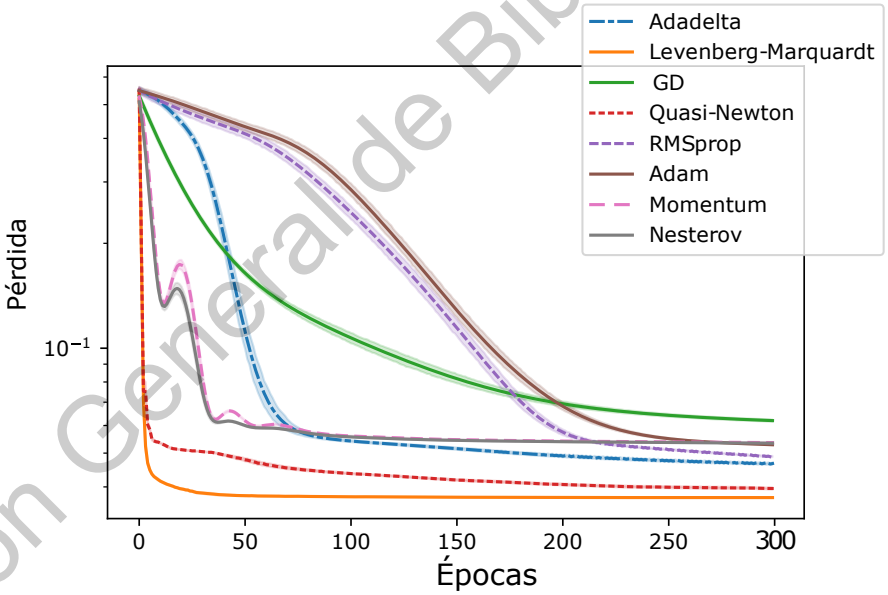
4.2.1. Optimizadores

De la Figura 4.18 se puede asegurar que los optimizadores de segundo orden (Quasi-Netwon y Lavenberg-Marquardt) convergen de forma más rápida y a mejores valores, en comparación con los optimizadores de primer orden. De entre estos dos optimizadores, Lavenberg-Marquardt es superior en cuanto a velocidad de convergencia, tanto para los datos de entrenamiento como para los

de validación.



(a) Datos de Entrenamiento.

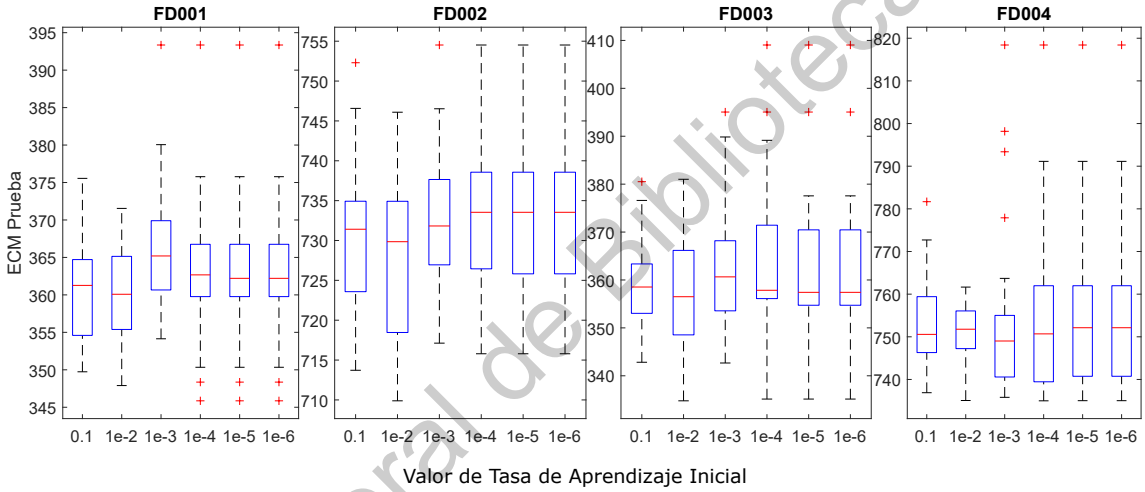


(b) Datos de Validación.

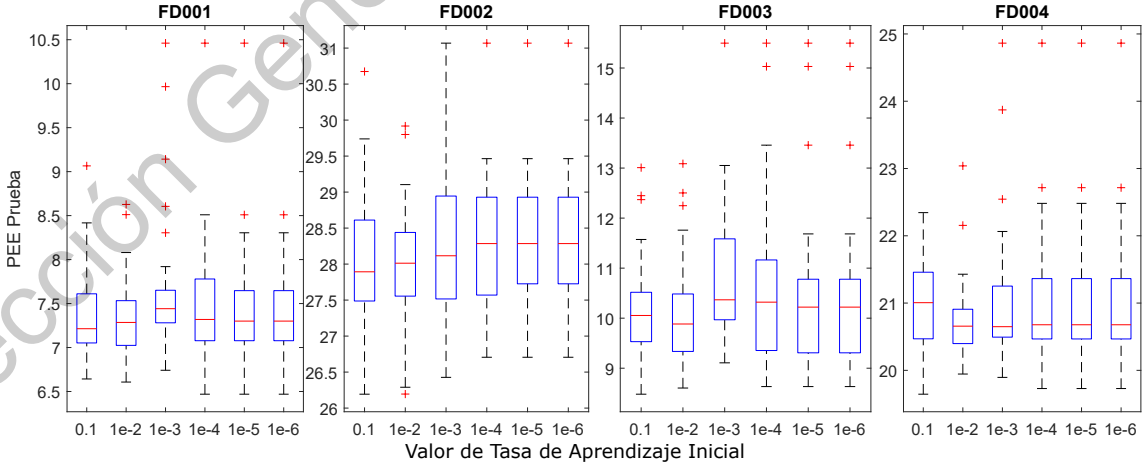
Figura 4.18: Comparación de los diferentes optimizadores durante 300 épocas con una función de pérdida ECM normalizada. El eje vertical tiene una escala logarítmica. Imagen propia.

4.2.2. Parámetros de Optimizador

El optimizador cuyos mejores resultados produce tanto en el conjunto de entrenamiento como en el conjunto de validación es Lavenberg-Marquardt. El parámetro de éste algoritmo es el valor inicial de la tasa de aprendizaje. Para encontrar el valor adecuado de este parámetro se busca entre valores de 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 0.01 y 0.1, en otras palabras, se busca en varios órdenes de 10 para encontrar el mejor valor.

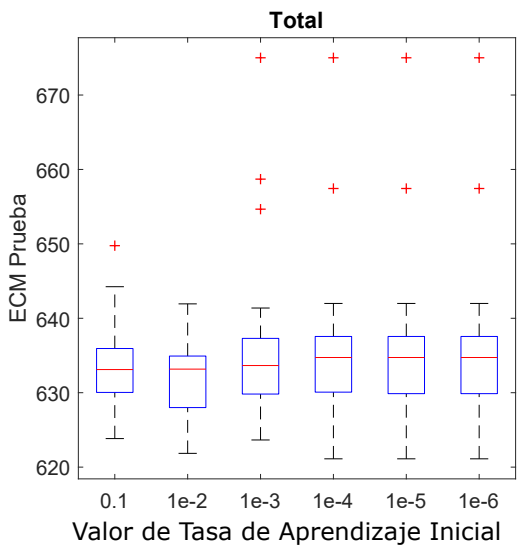


(a) ECM del conjunto de validación.

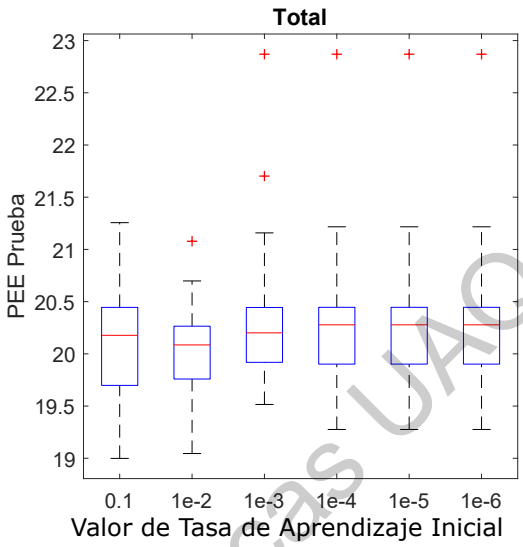


(b) PEE del conjunto de validación.

Figura 4.19: Gráfico de caja de la evaluación individual del modelo utilizando diferentes valores de Tasa de Aprendizaje. Imagen propia.



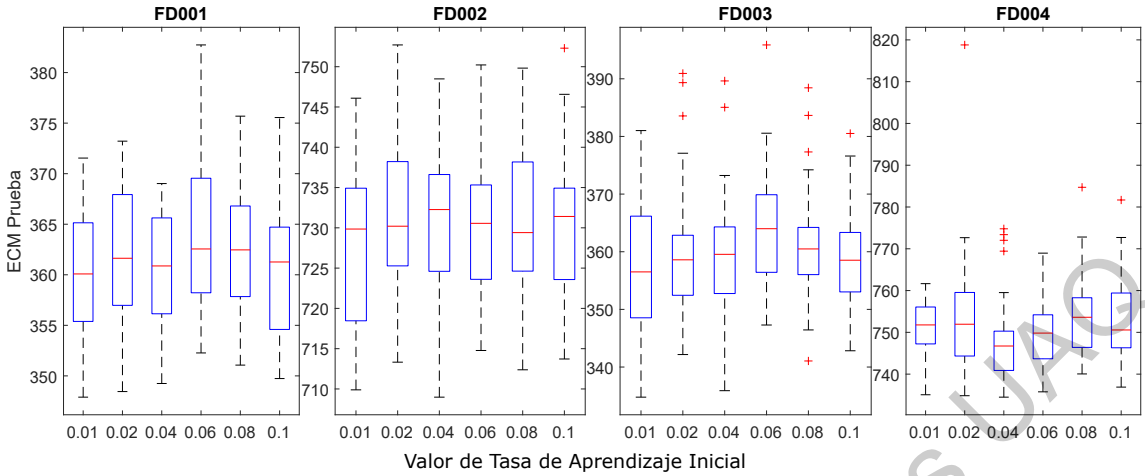
(a) Métrica ECM.



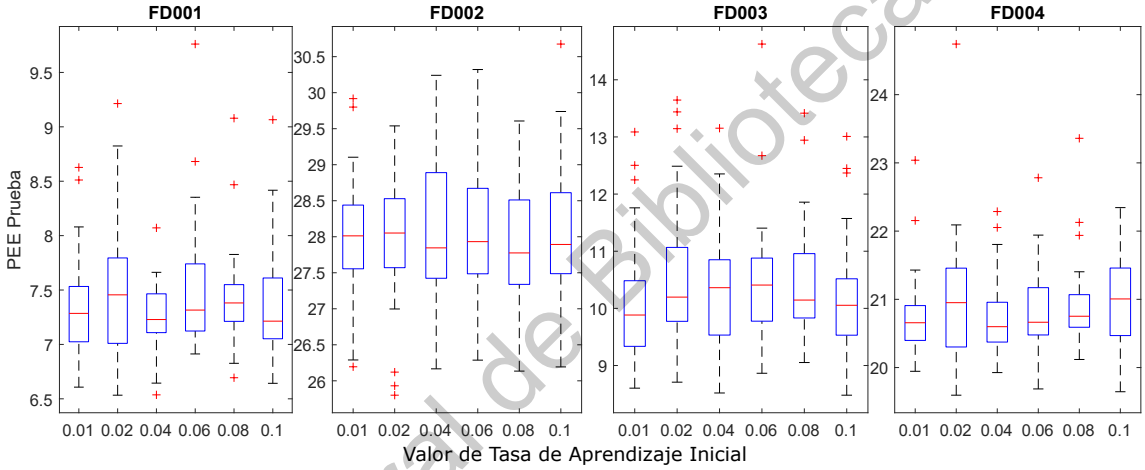
(b) Métrica PEE.

Figura 4.20: Gráfico de caja de la evaluación total del modelo utilizando diferentes valores de Tasa de Aprendizaje. Imagen propia.

Observando las evaluaciones individuales y totales del modelo (Figuras 4.19 y 4.20), el modelo parece ser poco sensible a los diferentes valores iniciales de la Tasa de Aprendizaje. Aún asi, al enfocarse en la evaluación total del modelo (Figura 4.20a), se aprecia una ventaja en cuanto a la medianas y mínimos para la métrica PEE, utilizando los valores 0.1 y 0.01.



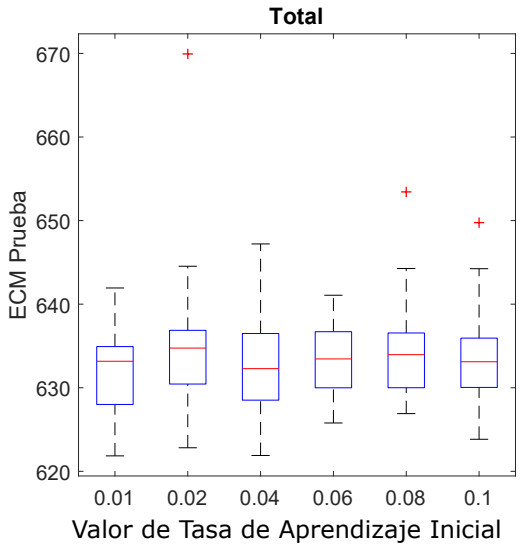
(a) ECM del conjunto de validación.



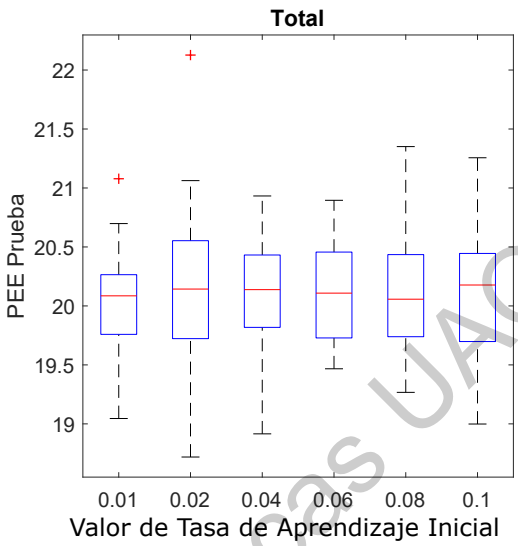
(b) PEE del conjunto de validación.

Figura 4.21: Comparación de grupos antes y después de eliminar unidades con desgaste atípico. Imagen propia.

Continuando con las pruebas de la Tasa de Aprendizaje, se busca un valor con mejores resultados dentro del rango de 0.1 y 0.01. Los resultados de ésta búsqueda se observan en las Figuras 4.21 y 4.22. Respecto a las pruebas individuales, nuevamente, no se observa una clara diferencia con respecto a los valores de prueba. De igual manera para las evaluaciones totales. Por otro lado, si bien todas las medianas son similares, se puede decir que el valor de 0.02 presenta los mínimos y máximos más favorables en ambas métricas. Por lo que se podría concluir que la mejor opción es 0.02.



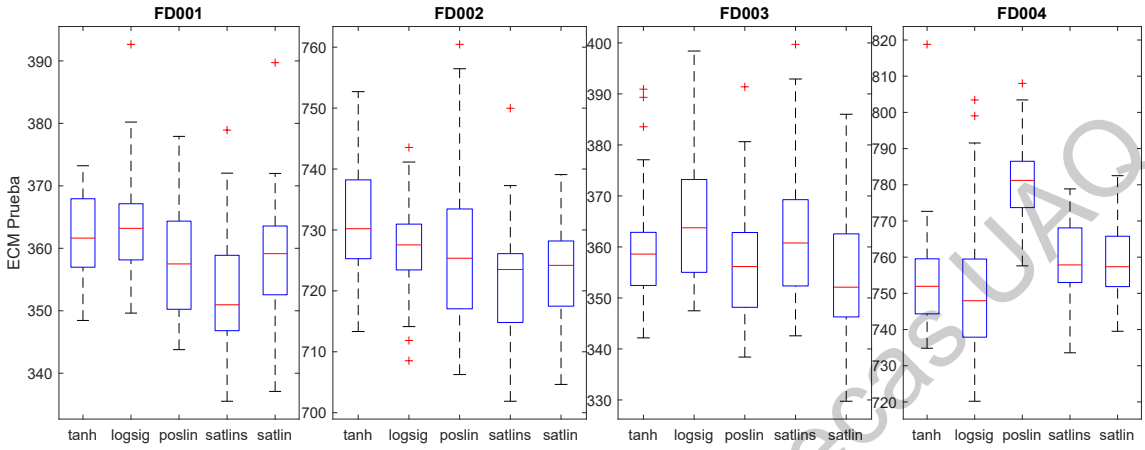
(a) ECM del conjunto de validación.



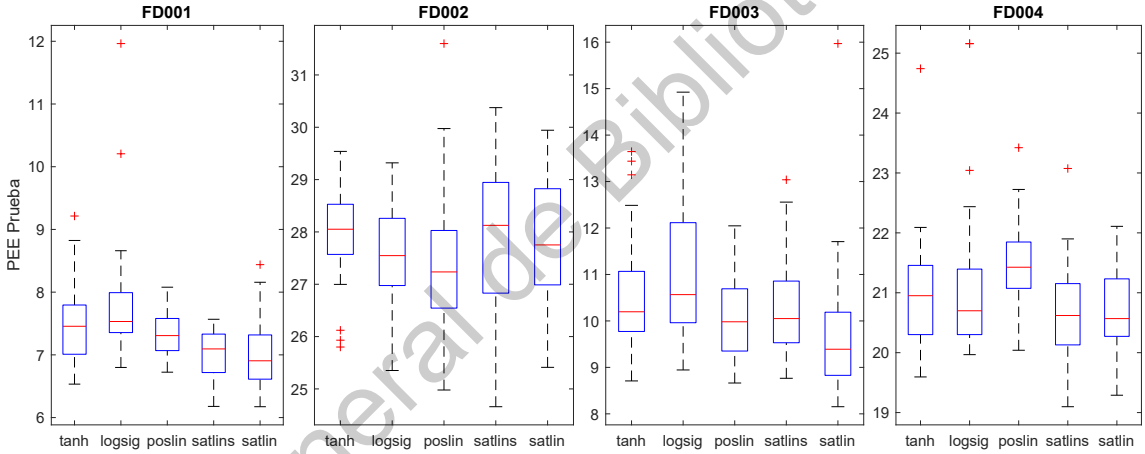
(b) PEE del conjunto de validación.

Figura 4.22: Gráfico de caja de la evaluación total del modelo utilizando diferentes valores de Tasa de Aprendizaje. Imagen propia.

4.2.3. Función de Activación



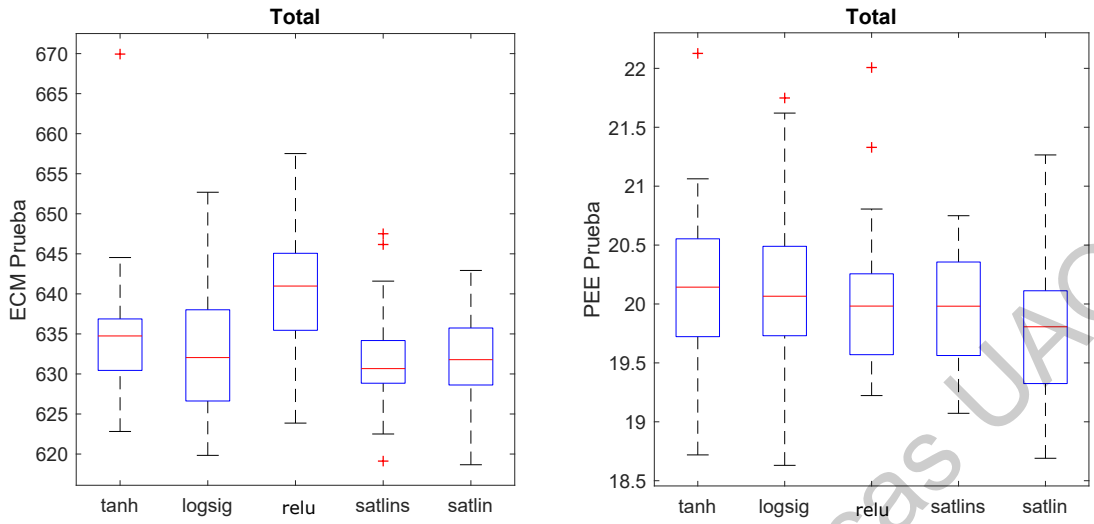
(a) ECM del conjunto de validación.



(b) PEE del conjunto de validación.

Figura 4.23: Gráfico de caja de la evaluación individual del modelo utilizando diferentes funciones de activación. Imagen propia.

En la Figuras 4.26 y 4.25, no se observa una clara diferencia entre modelos, sin embargo, la función de activación Lineal Saturada presenta una de las medianas y mínimos más bajos, por lo general, en ambas métricas. Por esta razón, se escoge ésta función de activación como una buena opción para el modelo. Además, siendo ésta una función lineal es menos costosa computacionalmente.

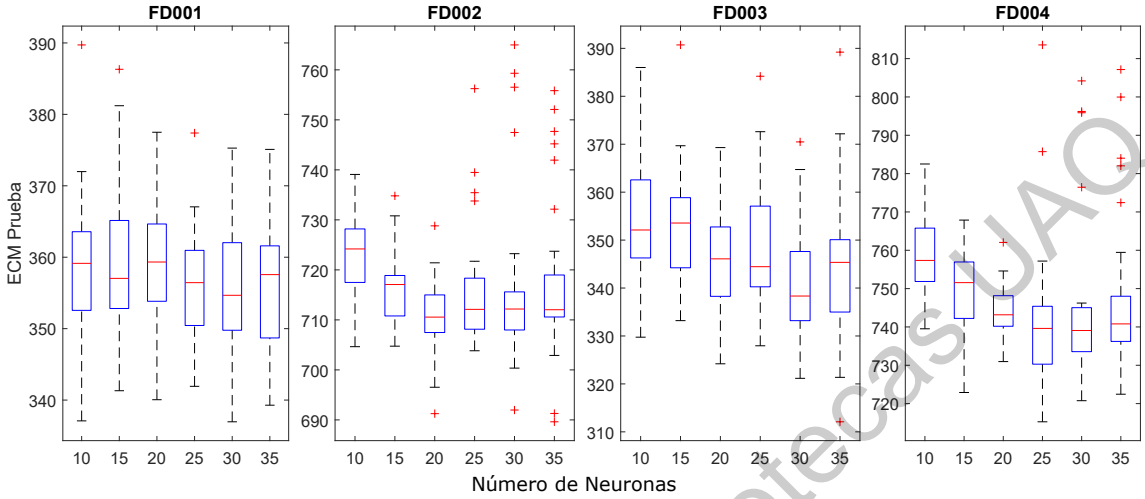


(a) ECM del conjunto de validación.

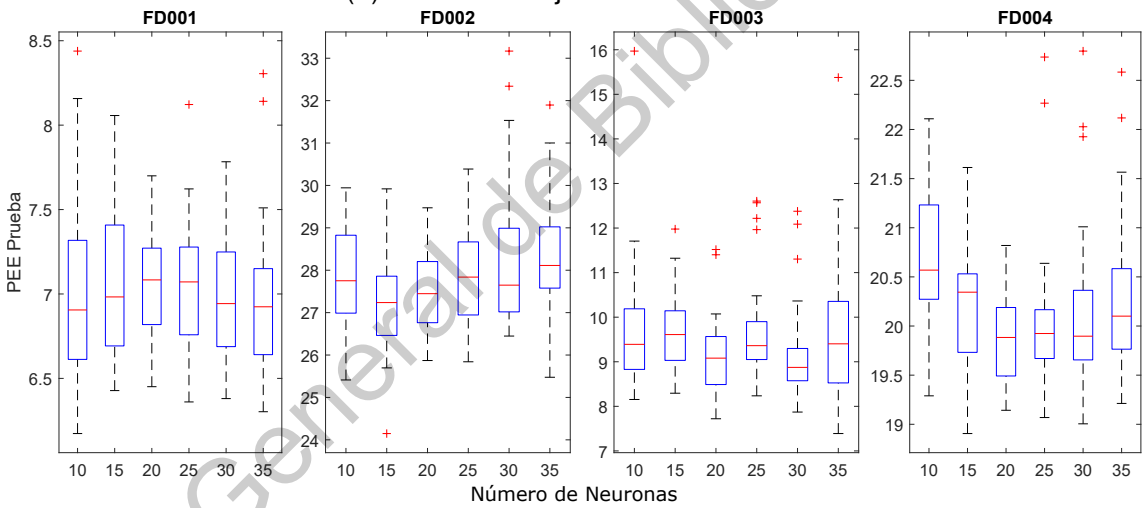
(b) PEE del conjunto de validación.

Figura 4.24: Gráfico de caja de la evaluación total del modelo utilizando diferentes funciones de activación. Imagen propia.

4.2.4. Número de Neuronas



(a) ECM del conjunto de validación.

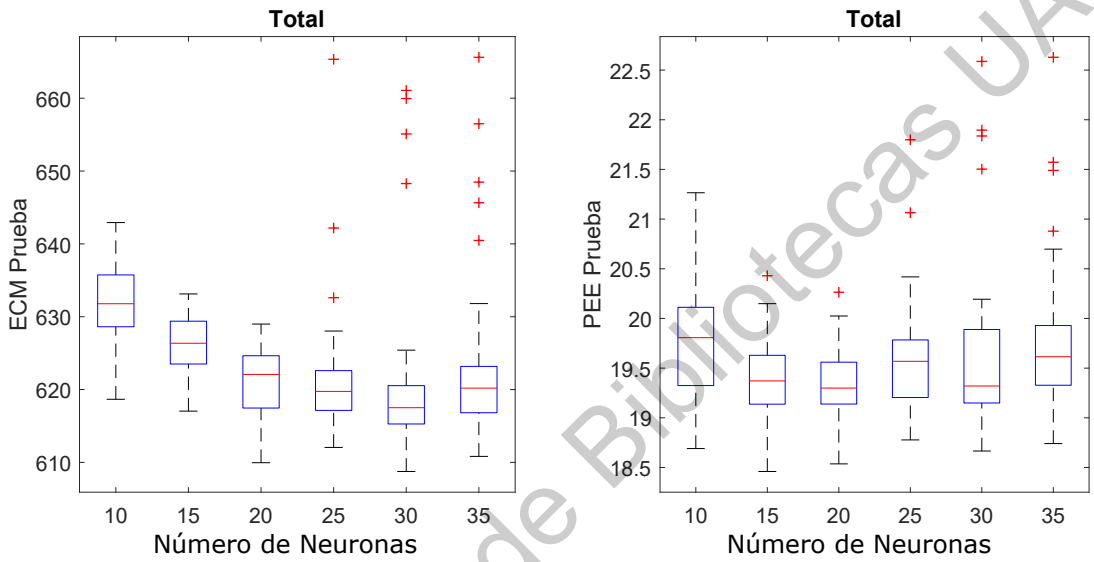


(b) PEE del conjunto de validación.

Figura 4.25: Gráfico de caja de la evaluación individual del modelo utilizando diferentes número de neuronas. Imagen propia.

En las evaluaciones individuales (Figura 4.25), el modelo parece ser insensible al número de neuronas al ser probado en el conjunto de prueba FD001, caso contrario para los demás conjuntos. Observando las medias de la métrica ECM (Figura 4.25a) se aprecia que para la prueba FD002 el desempeño del modelo no escala más alla de 20 neuronas, mientras que para FD003 y FD004

parece no escalar a mas de 30. Respecto a la métrica de PEE (Figura4.25b), el valor de neuronas para el cual el desempeño ya no escala es menor, en comparación con los resultados de ECM. Por ejemplo, para el conjunto de prueba FD002 el error no decrece más allá de 15 neuronas, mientras que para FD002 y FD004 parece ser 20 neuronas.



(a) ECM del conjunto de validación. (b) PEE del conjunto de validación.

Figura 4.26: Gráfico de caja de la evaluación total del modelo utilizando diferentes número de neuronas. Imagen propia.

En la evaluación general del modelo (Figura 4.26), el comportamiento del error respecto a el número de neuronas es más claro. Observando las medianas, el ECM no baja para más de 30 neuronas, mientras que para la PEE, no baja para más de 20 neuronas, aunque la diferencia de desempeño entre el modelo de 20 y el de 15 neuronas es relativamente insignificante, por lo que podría decirse que los valores de la PEE no bajan para más de 15 neuronas.

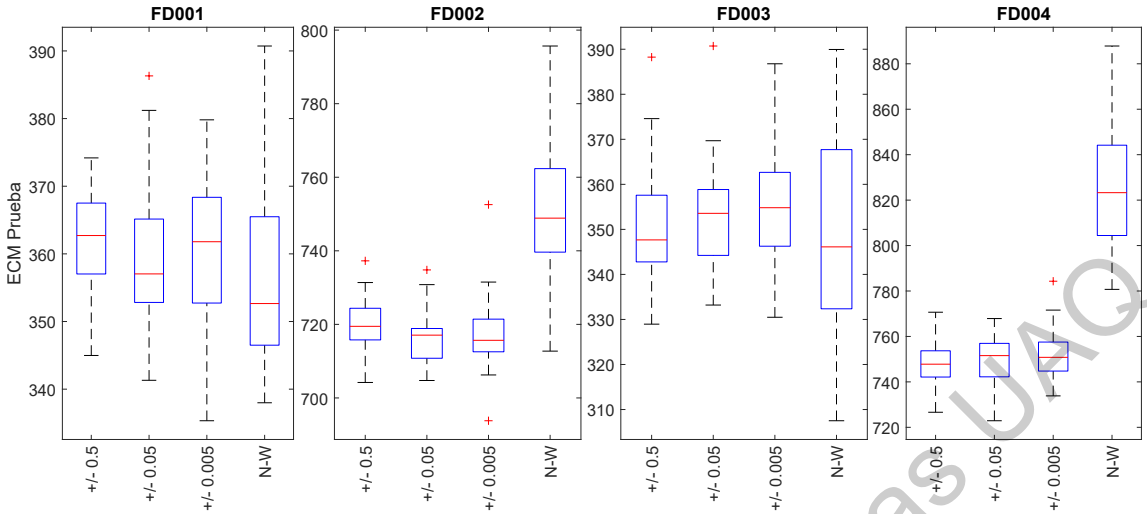
En este punto es importante señalar que se puede tomar diferentes decisiones respecto al modelo. Ya que si se busca que el modelo tenga un bajo costo computacional y un buen desempeño respecto a la métrica PEE, la opción

es 15 neuronas. Pero este modelo tendrá valores de ECM más grandes. Esta discrepancia de métricas indica que el modelo esta haciendo más predicciones tempranas que tardías.

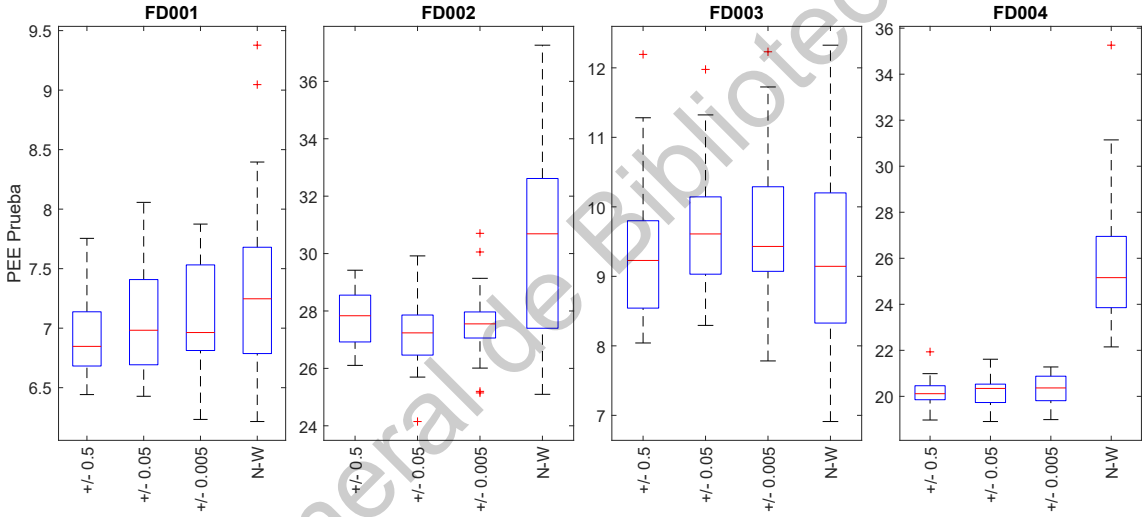
Por otro lado, para tener un valor de ECM bajo es necesario un valor de 30 neuronas, aunque esto aumenta un poco los errores para la métrica PEE. Esto indica que el modelo empieza a dejar de beneficiar un poco a las predicciones tempranas.

4.3. Pesos Iniciales

Las Figuras 4.27 y 4.28 muestran los resultados del modelo de 15 neuronas al utilizar diferentes métodos de inicialización de pesos. Cada método fue repetido 32 veces, y cada vez los valores iniciales eran diferentes. Los métodos de inicialización a comparar son el Aleatorio dentro de tres rangos diferentes ± 0.5, ± 0.05 y ± 0.005 y la técnica 'Nguyen-Widrow'.



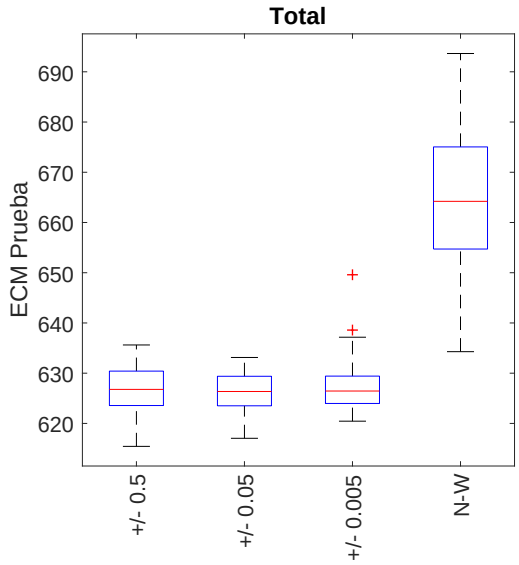
(a) Métrica ECM.



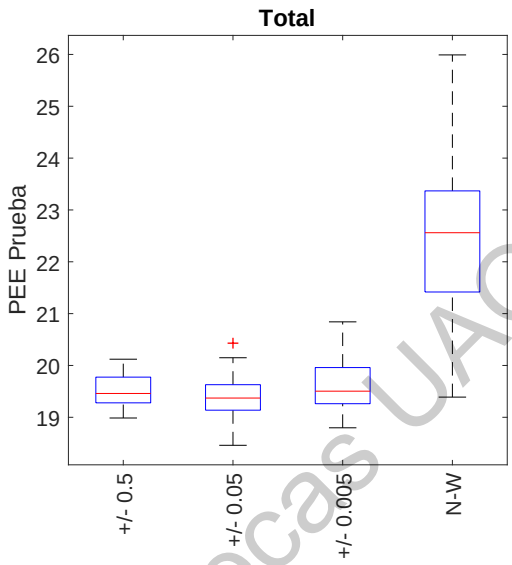
(b) Métrica PEE.

Figura 4.27: Gráfico de caja de la evaluación individual del modelo utilizando diferentes técnicas de inicialización. Imagen propia.

Respecto a las evaluaciones individuales (Figura 4.27), para las pruebas FD002 y FD004, existe un comportamiento superior del modelo al inicializarlo de manera aleatoria, en cualquier rango, en comparación con la inicialización 'Nguyen-Widrow'. Sin embargo, para las pruebas FD001 y FD003 el método aleatorio, en cualquier rango, tiene menos varianza que el 'Nguyen-Widrow'. Sin embargo, éste ultimo presenta los mejores mínimos.



(a) ECM del conjunto de validación.



(b) PEE del conjunto de validación.

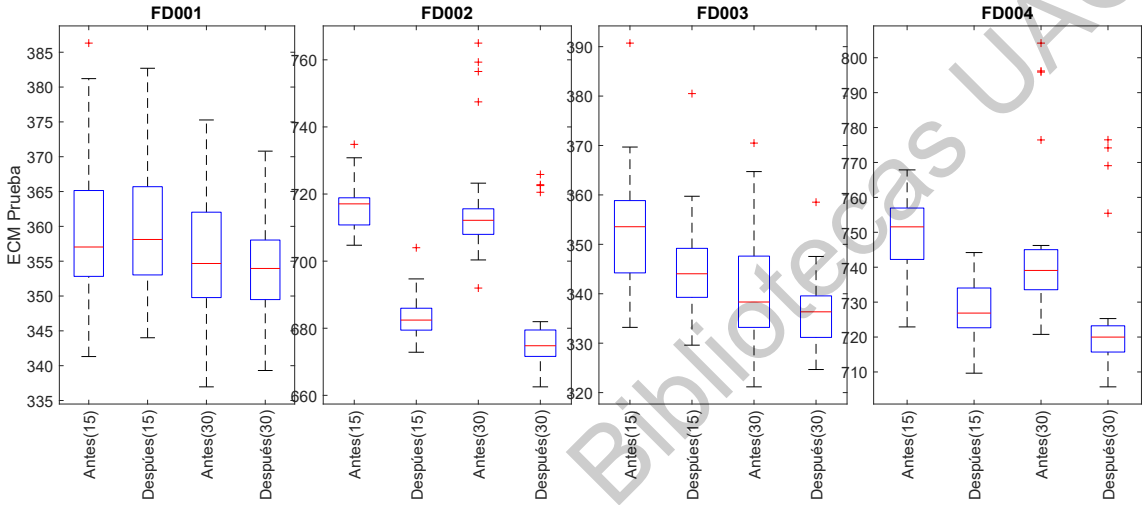
Figura 4.28: Gráfico de caja de la evaluación total del modelo utilizando diferentes técnicas de inicialización. Imagen propia.

En la Figura 4.28, se observa el comportamiento general del modelo respecto a las técnicas de inicialización. Y debido a que en las pruebas FD002 y FD004 (las que más muestras tienen) la técnica aleatoria es superior, en la evaluación general resulta lo mismo. Respecto a los rangos del método aleatorio, se observa que los resultados son muy similares. Sin embargo, los resultados utilizando un rango ± 0.05 muestran una mediana de PEE ligeramente menor, y un valor mínimo de PEE superior.

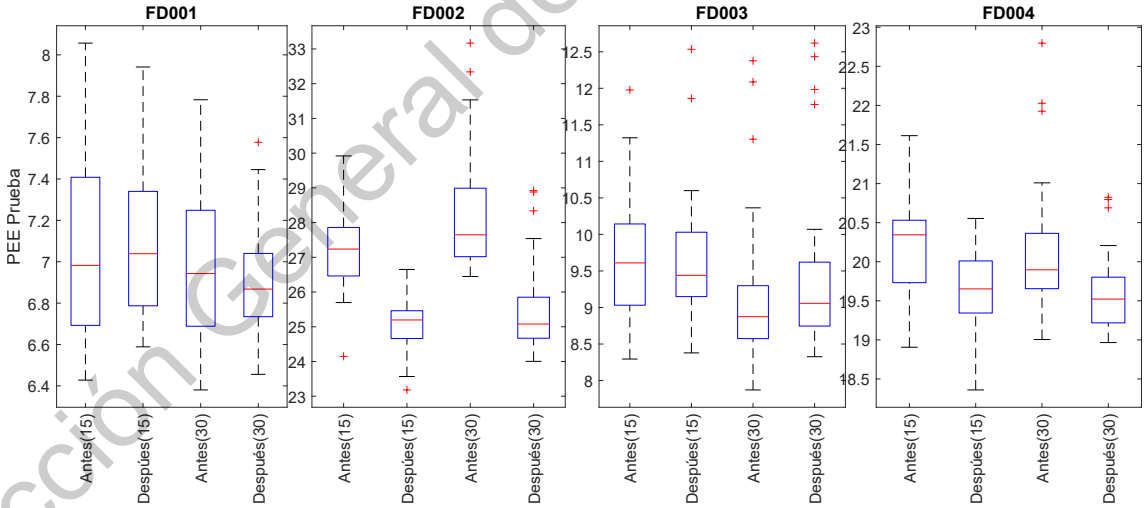
4.4. Filtro de Kalman

En la Figura 4.31 se muestra los resultados del FK aplicado a dos turbinas de prueba. En ambos ejemplos se muestra la predicción de la degradación, en términos de VUR. La linea verde punteada es la predicción hecha por el modelo de Perceptrón Multicapa y la linea gris continua representa las predicciones filtra-

das. Además se muestra el objetivo, en una estrella roja, que indica el valor real de VUR del sistema al final de la serie de tiempo. Es notorio que las predicciones hechas por el Perceptrón Multicapa son ruidosas. Al reducir el ruido con el FK observamos en ambos ejemplos, que la ultima predicción está más cercana a el valor real o *'Ground Truth'*.



(a) ECM del conjunto de validación.

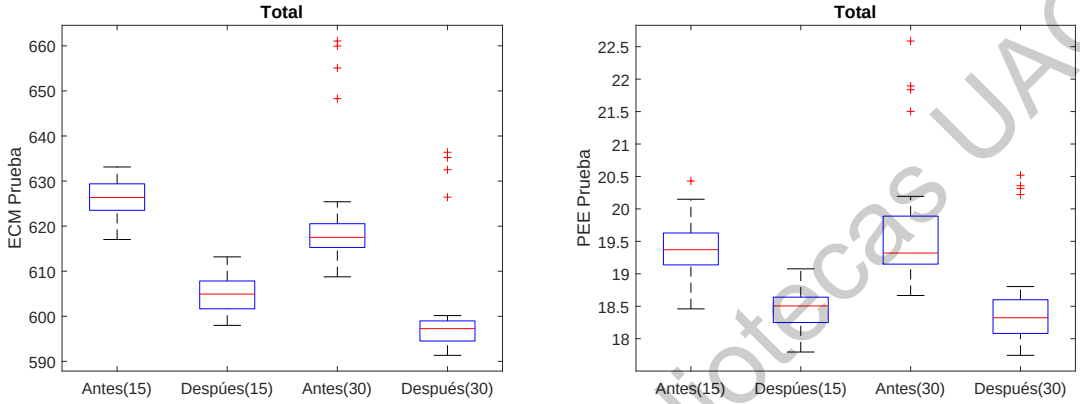


(b) PEE del conjunto de validación.

Figura 4.29: Comparación de grupos con 30 neuronas. Imagen propia.

En las Figuras 4.29 y 4.30 se ilustran los resultados de los modelos de 30 y 15 neuronas antes y después de integrar el FK a las predicciones.

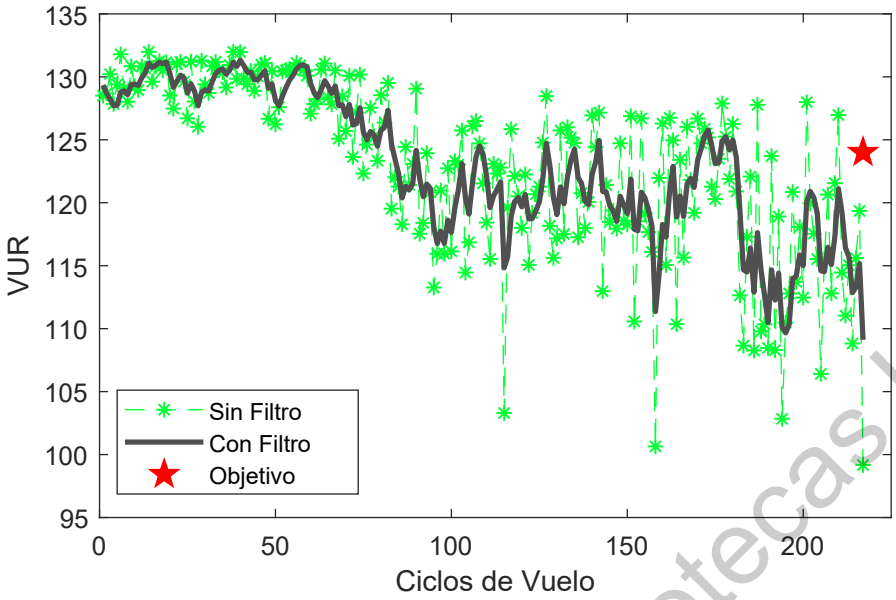
Para las evaluaciones individuales (Figura 4.29) FD002, FD003 y FD004. Tanto para el modelo de 15 como el de 30 neuronas, la mejoría es clara al utilizar el FK. Para prueba FD001 no es completamente claro el efecto del FK, sin embargo si se observa una disminución de la varianza.



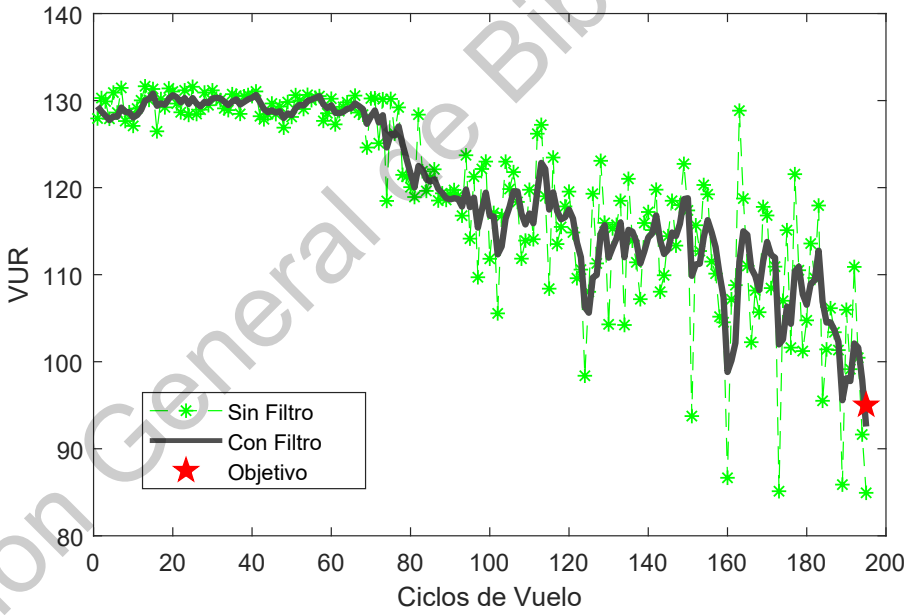
(a) ECM del conjunto de validación. (b) PEE del conjunto de validación.

Figura 4.30: Gráfico de caja de la evaluación total del modelo utilizando diferentes técnicas de inicialización. Imagen propia.

Para la evaluación total del modelo de 15 y 30 (Figura 4.30) se tiene que el uso de FK mejora bastante los errores, en ambas métricas, pero especialmente en la del ECM. Finalmente se observa que el modelo de 15 y 30 neuronas tienen un comportamiento similar para la PEE, sin embargo, el de 30 neuronas, sigue siendo mejor en ECM.



(a) Ejemplo 1.



(b) Ejemplo 2.

Figura 4.31: Ejemplos de los resultados del FK al pronóstico de VUR de dos turbinas de prueba diferentes. Imagen propia.

4.5. Selección de Modelo de Producción

Tras el ajuste del modelo inicial de Perceptrón Multicapa se ha encontrado que el modelo responde mejor con la siguiente configuración:

- Etiquetado con $R_c = 130$
- Normalización con Z-Score con Régimen
- Componentes: ciclos de vuelo y 21 sensores
- Removiendo las turbinas de entrenamiento con ECM atípico.
- Tasa de Aprendizaje igual a 0.02
- Función de activación Lineal Saturada en la capa oculta y función de activación Lineal en la capa de salida
- 30 neuronas en la capa oculta
- Filtrado de predicciones con FK

En las Tablas 4.2 y 4.3 se listan los 32 resultados de ECM y PEE, correspondientes a los 32 entrenamientos con pesos iniciales diferentes. Los resultados estan ordenados con respecto a el desempeño total del algoritmo.

Se observa que en ambas métricas, la muestra 31 del modelo es de los que mejores resultados obtuvo, siendo segundo lugar en ECM y primero en PEE. Por otro lado, dando prioridad a al métrica de PEE, se tiene que la mejor opción es la muestra 3 del modelo.

Tabla 4.2: Evaluación ECM en orden descendente. Tabla propia.

Muestra	FD001	FD002	FD003	FD004	Total
32	353.91	668.33	330.23	711.94	591.33
31*	357.42	665.65	337.24	710.89	591.47
9	349.20	662.53	337.32	719.01	592.03
11	349.74	672.15	337.71	709.30	592.27
15	362.04	669.93	347.55	705.73	593.34
17	351.77	675.40	333.54	711.42	593.91
1	354.79	671.24	328.41	717.07	594.07
3	362.09	667.51	332.59	717.04	594.31
29	347.60	671.63	331.90	720.01	594.71
24	370.80	671.23	331.58	712.90	595.32
26	355.66	671.59	324.67	722.17	595.57
5	350.96	679.60	342.62	708.98	595.76
8	352.88	676.89	330.76	717.58	596.37
2	362.09	672.97	327.83	719.74	596.59
6	358.81	671.64	333.30	720.53	596.69
21	363.86	674.02	337.01	715.43	597.01
16	341.17	680.31	330.60	722.03	597.51
28	358.66	677.02	330.20	719.24	597.75
19	349.53	679.36	328.01	721.37	597.75
27	352.30	681.96	332.81	715.98	597.88
23	354.00	676.52	337.11	719.96	598.13
22	362.21	673.19	338.79	720.22	598.41
10	351.26	677.35	340.74	720.53	598.76
20	355.75	672.66	339.79	724.50	598.94
14	347.93	674.18	342.62	725.11	599.00
13	347.17	680.87	338.09	721.20	599.34
12	357.12	677.05	331.78	724.27	599.53
4	356.49	676.54	335.68	725.29	600.16
7	339.31	722.78	344.06	755.46	626.44
18	345.29	725.84	339.35	769.08	632.51
30	354.26	720.53	344.87	776.48	635.21
25	349.45	722.52	358.53	774.17	636.39

Tabla 4.3: Evaluación PEE en orden descendente. Tabla propia.

Muestra	FD001	FD002	FD003	FD004	Total
3	7.26	24.00	8.85	19.02	17.74
31*	7.05	24.44	8.74	19.19	17.92
9	6.81	24.41	8.34	19.59	17.96
15	7.44	24.31	9.62	18.97	17.97
32	6.90	24.59	9.05	19.13	17.97
10	6.63	24.38	8.96	19.51	17.98
17	6.64	24.84	8.68	19.39	18.07
1	6.93	24.77	9.19	19.17	18.08
28	7.00	24.81	8.97	19.20	18.08
24	7.58	24.56	8.49	19.44	18.09
27	6.61	25.06	8.82	19.29	18.13
2	7.03	25.19	8.37	19.24	18.16
29	6.55	24.93	8.33	19.77	18.17
20	7.29	24.75	9.33	19.40	18.22
12	6.75	24.90	8.76	19.75	18.24
26	6.83	25.28	8.33	19.61	18.28
21	6.86	24.88	8.96	19.99	18.36
22	7.44	24.34	10.07	19.88	18.37
13	6.63	26.07	8.61	18.99	18.37
11	6.88	25.48	8.98	19.50	18.42
8	6.72	25.54	9.40	19.48	18.47
5	7.01	25.75	9.78	19.10	18.51
23	6.82	25.09	9.14	20.21	18.54
4	6.84	25.53	9.41	19.71	18.56
6	6.96	25.88	9.06	19.64	18.64
19	6.77	26.26	9.17	19.68	18.78
14	6.86	25.82	9.86	19.83	18.78
16	6.46	26.41	9.62	19.54	18.80
30	7.45	27.54	12.62	20.80	20.22
7	6.58	28.88	12.43	20.09	20.32
25	7.10	28.34	11.78	20.82	20.36
18	6.87	28.92	11.98	20.69	20.52

4.6. Neuroevolución

Los datos utilizados para la neuroevolución fueron preparados utilizando las mejores técnicas de preprocesamiento resultantes de los experimentos previos.

Los parámetros del algoritmo de neuroevolución son configurados de la siguiente manera:

- Población inicial: 100 individuos
- Máximo número de iteraciones: 500
- Parámetro de exploración (S): 25 % de la población
- Rango de Tasa de Aprendizaje: [0.001-1]
- Máximo número de neuronas: 25
- Mínimo número de neuronas: 10
- Aptitud de individuo: Métrica de PEE en el conjunto de validación
- Máximo número de epocas de entrenamiento de cada modelo: 250
- Número de epocas de paciencia para 'Detencion Temprana': 10

En la Figura 4.32 se ilustra los resultados de la neuroevolución. Típicamente se utiliza la media de la población para observar el comportamiento de la misma. En este caso se utiliza la mediana debido a que algunos pocos individuos resultan con errores de PEE extremadamente grandes, lo cual sesga la media de la población.

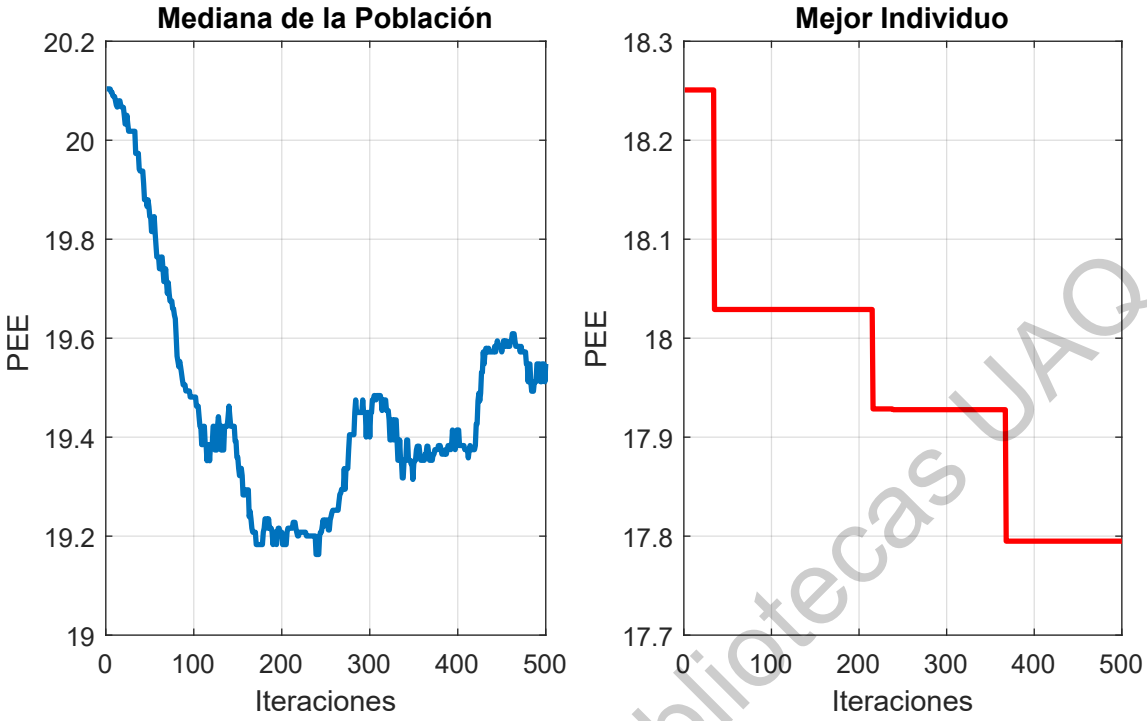


Figura 4.32: Gráfico del comportamiento de la neuroevolución utilizando la métrica PEE. Imagen propia.

Tras el proceso de neuroevolución (ver Figura 4.32) la arquitectura con el mejor desempeño tiene la siguientes características:

- Tasa de aprendizaje inicial: 0.3113
- Función de activación en la capa oculta: Lineal Saturada
- Función de activación en la capa de salida: Sigmoide
- Número de neuronas en capa oculta: 10

En las Tablas 4.4 y 4.5 se observa los resultados del mejor modelo en cada conjunto de prueba y de forma total. Además, se muestra el mismo modelo después de agregar el FK para reducir el ruido de las predicciones.

Tabla 4.4: Resultados del mejor individuo antes y después del FK, utilizando la métrica ECM. Tabla propia.

	FD001	FD002	FD003	FD004	Total
Sin FK	342.69	696.64	350.50	758.12	619.18
Con FK	350.20	667.15	332.25	734.05	598.42

Tabla 4.5: Resultados del mejor individuo antes y después del FK, utilizando la métrica PEE. Tabla propia.

	FD001	FD002	FD003	FD004	Total
Sin FK	6.37	23.97	9.14	19.44	17.79
Con FK	6.71	22.56	8.68	19.66	17.33

En las Tablas 4.6 Y 4.7 se hace la comparación de los resultados del modelo utilizando la metodología de búsqueda de parámetros manual y mediante la búsqueda con neuroevolución. Se observa que para la métrica ECM se aumentaron los errores. Esto es debido a que la neuroevolución se enfoco en la métrica de PEE. Sin embargo, en comparación con lo resultados previamente reportados por los modelos de 10 neuronas (Figura 4.26a), éste modelo los supera. Por otro lado, respecto a la métrica de PEE, el modelo supera considerablemente el propuesto previamente, respecto a el error total. Se observa que para el conjunto de prueba FD001 y FD004 aumento un poco el error, pero se ha disminuido el de FD002 y FD003, resultando en un error total superior al modelo previo.

Tabla 4.6: Comparación de modelos resultantes de diferentes metodologías utilizando la métrica ECM. Tabla propia.

	FD001	FD002	FD003	FD004	Total
Sin Neuroevolución	362.09	667.51	332.59	717.04	594.31
Con Neuroevolución	350.20	667.15	332.25	734.05	598.42

Tabla 4.7: Comparación de modelos resultantes de diferentes metodologías utilizando la métrica PEE. Tabla propia.

	FD001	FD002	FD003	FD004	Total
Sin Neuroevolución	7.26	24.00	8.85	19.02	17.74
Con Neuroevolución	6.71	22.56	8.68	19.66	17.33

4.7. Validación del Modelo

Para validar el rendimiento de nuestro modelo, éste se compara con algunos de los algoritmos principales de Aprendizaje Profundo: RNC y RCP (Bengio y cols., 2013). El primero ha captado mucha atención para trabajar datos de alta dimensionalidad, como imágenes y series de tiempo, ya que éste realiza un proceso de extracción de características automatizado Khan y Yairi (2018); el último es una técnica prometedora utilizada para obtener las características más útiles relevantes para la tarea de estimación Zhang y cols. (2017).

Los siguientes trabajos seleccionados de la literatura también se usan para el conjunto de datos de turbofan (ver Tabla 4.8): el primer intento de usar RNC para estimación de la vida útil remanente en 2016 por Sateesh Babu y cols. (2016); un trabajo más reciente en 2018 usando RNC por Li y cols. (2018); y dos modelos del trabajo de Zhang y cols. (2017) que incluyen una RCP común y un modelo más complejo llamado *Multi Objective Deep Belief Networks Ensemble* (MODBNE), que combina varias RCP diseñadas por un algoritmo evolutivo multiobjetivo.

Tabla 4.8: Trabajos en el Estado del Arte. Tabla propia.

Trabajo	Año	Modelo
Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life (Sateesh Babu y cols., 2016)	2016	RNC
Remaining useful life estimation in prognostics using deep convolution neural networks (Li y cols., 2018)	2018	RNC(2)
Multiobjective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics (Zhang y cols., 2017)	2017	MODBNE
Multiobjective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics (Zhang y cols., 2017)	2017	RCP

Mirando a las evaluaciones con ECM (Tabal 4.9), el modelo propuesto tiene una evaluación total superior a las técnicas tradicionales RNC y RCP, aunque MODBNE sigue superando a la propuesta. En cuanto a la evaluación de cada prueba, se observa que el modelo propuesto supera a todos los demás en el conjunto de prueba FD004.

Por otro lado, en la evaluación de PEE (Tabla 4.3), observamos que el modelo propuesto supera a todas las arquitecturas de aprendizaje profundo, con respecto a la evaluación total. Esto es realmente debido a que, nuevamente, en el conjunto FD004, el modelo propuesto tiene el error más bajo.

Tabla 4.9: Comparación con ECM. Tabla propia.

Prueba	RNC	RCP	RNC(2)	MODBNE	Propuesto
FD001	340.33	231.34	N/A	226.20	350.20
FD002	917.73	735.49	N/A	627.50	667.15
FD003	392.73	216.38	N/A	156.50	332.25
FD004	849.72	892.81	N/A	821.40	734.05
Total	737.95	645.95	N/A	572.13	598.42

Tabla 4.10: Comparación PEE. Tabla propia.

Prueba	RNC	RCP	RNC(2)	MODBNE	Propuesto
FD001	12.87	4.18	2.74	3.34	6.71
FD002	52.39	34.87	40.20	21.57	22.56
FD003	15.96	4.42	2.84	4.22	8.68
FD004	31.80	32.07	50.27	26.44	19.66
Total	34.43	25.24	33.15	18.24	17.33

CAPÍTULO 5

Conclusiones

“Haz las cosas lo más simple que puedas, pero no te limites a lo simple”

Albert Einstein

Se ha propuesto un modelo de bajo costo computación para el pronóstico de vida útil remanente de turbinas de avión. Este modelo comprende un red neuronal de tipo Perceptrón Multicapa de 10 neuronas junto con un Filtro de Kalman. Los resultados indican que nuestro modelo supera a otros del estado del arte basados en redes neuronales profundas. Se probaron dos metodologías para la búsqueda de hiperparámetros: una búsqueda ‘manual’ y una búsqueda mediante una técnica de meta-aprendizaje llamada neuroevolución.

Pese a que el método manual produjo resultados competitivos con el estado del arte, el método por neuroevolución produjo mejores resultados. También cabe mencionar, que el algoritmo de neuroevolución es más simple y rápido de implementar que la búsqueda manual de parámetros.

Por otro lado, es importante señalar los siguientes descubrimientos. La componente de ciclos de vuelo funciona como un indicador temporal, así como las seis variables IHFT, a diferencia de que ésta primera es sólo una variable. En la literatura se suele descartar esta componente para los modelos. De igual

forma, en la literatura se suele utilizar solo los sensores con comportamiento monótamente ascendente o descendente. Sin embargo, según los resultados, se obtiene un mejor desempeño al utilizar todos los sensores, lo que indica que hay información valiosa en esas componentes usualmente descartadas. Un factor crucial para el éxito del modelo da bajo costo es el algoritmo de entrenamiento Levenberg-Marquardt. Este optimizador de segundo orden provee una ventaja a las redes neuronales pequeñas sobre las profundas, ya que estas últimas no pueden ser entrenadas mediante este algoritmo debido a su gran cantidad de parámetros.

La redes neuronales estáticas, como el Perceptrón Multicapa, tienen predicciones ruidosas en aplicaciones con series de tiempo debido a que no correlacionan las predicciones previas con la actual, sin embargo al utilizar el Filtro de Kalman es posible hacer esta importante correlación de series de tiempo, resultando así en predicciones menos ruidosas, con la ventaja de que el Filtro de Kalman es una técnica de bajo costo computacional para procesamiento de señales.

El resultado más notable fue en el conjunto de prueba FD004, al superar a todos los modelos del estado del arte. Esta prueba, presuntamente es la más complicada de las cuatro debido a que sus turbinas son simuladas con 6 condiciones de operación y 2 modos de falla diferentes, complicando la relaciones entre los datos. Sin embargo, dicho aumento de dificultad también se traduce en una mayor varianza en los datos, lo cual es beneficioso en redes estáticas como Perceptrón Multicapa.

En conlusión, para el desarrollo de algoritmos de pronóstico se recomienda explorar a profundidad los algoritmos de bajo costo computacional, ya que como se demostró, una simple red neuronal configurada apropiadamente puede

producir tan buenos resultados como una red profunda.

Trabajos futuros incluyen una evaluación extensa de la neuroevolución, incluyendo más parámetros del modelo tal como las técnicas de preprocesamiento y haciendo una comparación entre diferentes algoritmos de nueroevolución. Además, como se ha demostrado, el método de optimización ha sido fundamental para obtener buenos resultados. Siguiendo la línea de investigación de neuroevolución, algunos algoritmos han sido desarrollados para remplazar los optimizadores tradicionales, combinando algoritmos evolutivos y técnicas de gradiente, obteniendo así la ventaja de ambos métodos.

Adicionalmente, desde el campo de investigación de Aprendizaje Profundo, se han propuesto técnicas con excelentes resultados para el procesamiento de texto. Estas técnicas conocidas como *'embeddings'* tienen la finalidad de extraer el significado de una palabra conforme a su contexto permitiendo a los sistemas digitales interpretar el idioma escrito. Gracias a estas técnicas de han logrado sistemas de para inferir el sentimiento de un texto, traducción automática de idiomas, clasificación automática de temas, etc. Sería interesante, utilizando estas técnicas de procesamiento de texto, determinar el significado del conjunto de mediciones de un sistema tal como una turbina, con la finalidad de inferir su estado de salud. En la literatura se reportan pocos trabajos del uso de esta técnicas para series de tiempo, sin embargo se considera que es una importante área de investigación en el desarrollo de algoritmos de pronóstico y diagnóstico.

Por otro lado, una etapa clave para el desarrollo de algoritmos de pronóstico de VUR es el etiquetado de datos, comúnmente hecho por expertos. En este caso se usa técnica propuesta en la literatura y evaluada por otros. Sin embargo, en el área de Aprendizaje Automático varias técnicas de Auto-etiquetado se han propuesto para tratar bases de datos con pocas etiquetas, principalmente en el

área de visión por computadora. A la fecha se reportan pocos trabajos utilizando estas técnicas para la base del turbofan y por lo tanto se considera una área de oportunidad importante de investigación. Se considera que el apropiado etiquetado de la base de datos es más beneficioso que el desarrollo de modelos más complejos utilizando las etiquetas tradicionales.

Por último, las bases de datos para pronóstico tendrán un inherente problema de discrepancia de distribuciones de datos de entrenamiento y de prueba, debido al etiquetado manual de la base de datos. En la literatura, referente a el desarrollo de algoritmos de pronóstico de VUR, no se habla al respecto, sin embargo es un problema que debe de tratarse adecuadamente durante el desarrollo de los algoritmos. Este problema se ha convertido en algo común en aplicaciones de visión por computadora, donde entrenan modelos con una gran cantidad de imágenes provenientes de Internet, sin embargo las imágenes en producción son diferentes, principalmente en la calidad de éstas. Algunas técnicas se han estado proponiendo para tratar este problema de forma más eficiente y es pertienente la investigación del uso de estas técnicas para los algoritmos de pronóstico de VUR.

Referencias

Abu-Hanna, A., y Lucas, P. J. (2001). Prognostic models in medicine. *Methods of Information in Medicine-Methodik der Information in der Medizin*, *40*(1), 1–5.

Ahmad, R., y Kamaruddin, S. (2012a). An overview of time-based and condition-based maintenance in industrial application. *Computers and Industrial Engineering*, *63*(1), 135–149. Descargado de http://dx.doi.org/10.1016/j.cie.2012.02.002 doi: 10.1016/j.cie.2012.02.002

Ahmad, R., y Kamaruddin, S. (2012b). A review of condition-based maintenance decision-making. *European J. of Industrial Engineering*, *6*(5), 519. Descargado de http://www.inderscience.com/link.php?id=48854 doi: 10.1504/EJIE.2012.048854

Aigner, K. (1983). *Machine Learning* (R. S. Michalski, J. G. Carbonell, y T. M. Mitchell, Eds.). Berlin, Heidelberg: Springer Berlin Heidelberg. Descargado de http://link.springer.com/10.1007/978-3-662-12405-5 doi: 10.1007/978-3-662-12405-5

Alaswad, S., y Xiang, Y. (2017). A review on condition-based maintenance optimization models for stochastically deteriorating system. *Reliability Engineering and System Safety*, *157*, 54–63. Descargado de http://dx.doi.org/10.1016/j.ress.2016.08.009 doi: 10.1016/j.ress.2016.08.009

Baptista, M., Henriques, E. M., de Medeiros, I. P., Malere, J. P., Nascimento, C. L., y Prendinger, H. (2018). Remaining useful life estimation in aeronautics: Com-

bining data-driven and Kalman filtering. *Reliability Engineering and System Safety*, *000*, 1–12. doi: 10.1016/j.ress.2018.01.017

Bengio, Y., Courville, A., y Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798–1828.

Davies, A. (2012). *Handbook of condition monitoring: Techniques and methodology*. Springer Netherlands. Descargado de https://books.google.com.mx/books?id=UzTyCAAAQBAJ

Dhillon, B. (2002). *Engineering Maintenance*. CRC Press. Descargado de http://www.crcnetbase.com/doi/book/10.1201/9781420031843https://www.taylorfrancis.com/books/9781420031843 doi: 10.1201/9781420031843

Dreyfus, S. E. (1990, sep). Artificial neural networks, back propagation, and the Kelley-Bryson gradient procedure. *Journal of Guidance, Control, and Dynamics*, *13*(5), 926–928. Descargado de http://arc.aiaa.org/doi/10.2514/3.25422 doi: 10.2514/3.25422

Duch, W., y Grudziński, K. (2018). Meta-learning: searching in the model space. Descargado de http://arxiv.org/abs/1806.06207

Duchi, J., Hazan, E., y Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*(Jul), 2121–2159.

Duffuaa, S., Ben-Daya, M., Al-Sultan, K., y Andijani, A. (2001, sep). A generic conceptual simulation model for maintenance systems. *Journal of Quality in Maintenance Engineering*, *7*(3), 207–219. Descargado de http://www.emeraldinsight.com/doi/10.1108/13552510110404512 doi: 10.1108/13552510110404512

Eker, Ö. F., Camci, F., y Jennions, I. K. (2012). Major challenges in prognostics: study on benchmarking prognostic datasets, Major challenges in prognostics: study on benchmarking prognostic datasets. (July), 148–155. Descargado de https://dspace.lib.cranfield.ac.uk/handle/1826/9994

Elattar, H. M., Elminir, H. K., y Riad, A. M. (2016). Prognostics: a literature review. *Complex & Intelligent Systems*, *2*(2), 125–154. Descargado de http://link.springer.com/10.1007/s40747-016-0019-3 doi: 10.1007/s40747-016-0019-3

Elattar, H. M., Elminir, H. K., y Riad, A. M. (2018). Conception and implementation of a data-driven prognostics algorithm for safety–critical systems. *Soft Computing*(2010), 1–18. doi: 10.1007/s00500-017-2995-7

Ellefsen, A. L., Bjørlykhaug, E., Æsøy, V., Ushakov, S., y Zhang, H. (2019). Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture. *Reliability Engineering and System Safety*, *183*(June 2018), 240–251. Descargado de https://doi.org/10.1016/j.ress.2018.11.027 doi: 10.1016/j.ress.2018.11.027

Floreano, D., Dürr, P., y Mattiussi, C. (2008). Neuroevolution: From architectures to learning. *Evolutionary Intelligence*, *1*(1), 47–62. doi: 10.1007/s12065-007-0002-4

Géron, A. (2019). *Hands on machine learning with scikit learn keras and tensorflow* (Early Release ed., Vol. 2nd Edition). O'Reilly Media.

Heimes, F. O. (2008, oct). Recurrent neural networks for remaining useful life estimation. En *2008 international conference on prognostics and health management* (pp. 1–6). IEEE. Descargado de http://ieeexplore.ieee.org/document/4711422/ doi: 10.1109/PHM.2008.4711422

Hinton, G. E., y Osindero, S. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*.

Hu, C., Youn, B. D., y Wang, P. (2019). *Engineering Design under Uncertainty and Health Prognostics*. Descargado de http://link.springer.com/10.1007/978-3-319-92574-5 doi: 10.1007/978-3-319-92574-5

Hutson, M. (2018, aug). AI takes on video games in quest for common sense. *Science*, *361*(6403), 632–633. Descargado de http://www.sciencemag.org/lookup/doi/10.1126/science.361.6403.632 doi: 10.1126/science.361.6403.632

Jain, A. K., Kundu, P., y Lad, B. K. (2014). Prediction of Remaining Useful Life of an Aircraft Engine under Unknown Initial Wear. (September 2015).

Jankowski, N., Gr, K., y Intelligence, C. (2011). *Meta-Learning in Computational Intelligence* (Vol. 358; N. Jankowski, W. Duch, y K. Gra bczewski, Eds.). Berlin, Heidelberg: Springer Berlin Heidelberg. Descargado de http://link.springer.com/10.1007/978-3-642-20980-2 doi: 10.1007/978-3-642-20980-2

Jardine, A. K., Lin, D., y Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, *20*(7), 1483–1510. doi: 10.1016/j.ymssp.2005.09.012

Javed, K., Gouriveau, R., y Zerhouni, N. (2014). A New Multivariate Approach for Prognostics Based on Extreme Learning Machine and Fuzzy Clustering. , 1–14.

Kagermann, H., Lukas, W.-D., y Wahlster, W. (2011). Industrie 4.0: Mit dem internet der dinge auf dem weg zur 4. industriellen revolution. *VDI nachrichten*, *13*(11).

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, *82*(1), 35. doi: 10.1115/1.3662552

Khan, S., y Yairi, T. (2018). A review on the application of deep learning in system health management. *Mechanical Systems and Signal Processing*, *107*, 241–265. Descargado de `https://doi.org/10.1016/j.ymssp.2017.11.024` doi: 10.1016/j.ymssp.2017.11.024

Kingma, D. P., y Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krishnan, R., Shalit, U., y Sontag, D. (2015, noviembre). Deep Kalman Filters. *ArXiv e-prints*.

Labib, A. W. (2004). A decision analysis model for maintenance policy selection using a CMMS. *Journal of Quality in Maintenance Engineering*, *10*(3), 191–202. doi: 10.1108/13552510410553244

Lara, A., Sanchez, G., Coello, C. A., y Schütze, O. (2010). Hcs: A new local search strategy for memetic multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, *14*(1), 112–132. doi: 10.1109/TEVC.2009.2024143

Lecun, Y., Bengio, Y., y Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. doi: 10.1038/nature14539

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., y Sohl-Dickstein, J. (2017). Deep Neural Networks as Gaussian Processes. *arXiv preprint*(Nips). Descargado de http://arxiv.org/abs/1711.00165 doi: arXiv:1711.00165v3

Lee, J., Wu, F., Zhao, W., Ghaffari, M., Liao, L., y Siegel, D. (2014). Prognostics and health management design for rotary machinery systems - Reviews, methodology and applications. *Mechanical Systems and Signal Processing*, *42*(1-2), 314–334. Descargado de http://dx.doi.org/10.1016/j.ymssp.2013.06.004 doi: 10.1016/j.ymssp.2013.06.004

Lehman, J., Chen, J., Clune, J., y Stanley, K. O. (s.f.). Safe mutations for deep and recurrent neural networks through output gradients. En *Proceedings of the genetic and evolutionary computation conference on - gecco '18* (pp. 117–124). New York, New York, USA: ACM Press. doi: 10.1145/3205455.3205473

Lei, Y., Li, N., Guo, L., Li, N., Yan, T., y Lin, J. (2018). Machinery health prognostics: A systematic review from data acquisition to RUL prediction. *Mechanical Systems and Signal Processing*, *104*(December 2017), 799–834. Descargado de https://doi.org/10.1016/j.ymssp.2017.11.016 doi: 10.1016/j.ymssp.2017.11.016

Li, X., Ding, Q., y Sun, J. Q. (2018). Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering and System Safety*, *172*(December 2017), 1–11. Descargado de https://doi.org/10.1016/j.ress.2017.11.021 doi: 10.1016/j.ress.2017.11.021

Lin, Y., Li, X., y Hu, Y. (2018). Deep diagnostics and prognostics: An integrated hierarchical learning framework in PHM applications. *Applied Soft Computing Journal*(February). Descargado de http://dx.doi.org/10.1016/j.asoc.2018.01.036 doi: 10.1016/j.asoc.2018.01.036

Liu, J., Wang, W., y Golnaraghi, F. (2009). A multi-step predictor with a variable input pattern for system state forecasting. *Mechanical Systems and Signal Processing*, *23*(5), 1586–1599.

Löfsten, H. (1999, jul). Management of industrial maintenance – economic evaluation of maintenance policies. *International Journal of Operations & Production Management*, *19*(7), 716–737. Descargado de https://www.emeraldinsight.com/doi/10.1108/01443579910271683 doi: 10.1108/01443579910271683

MathWorks. (2018). *Aprendizaje automático: Tres cosas que es necesario saber - MATLAB & Simulink.* Descargado 2018-10-14, de https://la.mathworks.com/discovery/machine-learning.html

Moré, J. J. (1978). The levenberg-marquardt algorithm: implementation and theory. En *Numerical analysis* (pp. 105–116). Springer.

NASA Ames Research Center. (s.f.). *Prognostics Center - Data Repository.* Descargado 2018-10-23, de https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/

Neale, M., y Woodley, B. (1975). Condition Monitoring Methods and Economics. *Symposium of the Society of Environmental Engineers*.

Nguyen, D., y Widrow, B. (1990). Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. En *1990 ijcnn international joint conference on neural networks* (pp. 21–26).

Niu, G. (2017). *Data-Driven Technology for Engineering Systems Health Management* (1.ª ed.; Springer, Ed.). Beijing China: Springer Singapore. doi: 10.1007/978-981-10-2032-2

Niu, G., Yang, B. S., y Pecht, M. (2010). Development of an optimized condition-based maintenance system by data fusion and reliability-centered maintenance. *Reliability Engineering and System Safety*, *95*(7), 786–796. Descargado de http://dx.doi.org/10.1016/j.ress.2010.02.016 doi: 10.1016/j.ress.2010.02.016

Peel, L. (2008, oct). Data driven prognostics using a kalman filter ensemble of neural network models. En *2008 international conference on prognostics and health management, phm 2008* (pp. 1–6). IEEE. Descargado de http://ieeexplore.ieee.org/document/4711423/ doi: 10.1109/PHM.2008.4711423

Peng, Y., Dong, M., y Zuo, M. J. (2010). Current status of machine prognostics in condition-based maintenance: A review. *International Journal of Advanced Manufacturing Technology*, *50*(1-4), 297–313. doi: 10.1007/s00170-009-2482-0

Real, E., Aggarwal, A., Huang, Y., y Le, Q. V. (2018, feb). Regularized Evolution for Image Classifier Architecture Search. Descargado de http://arxiv.org/abs/1802.01548

Rumelhart, D. E., Hinton, G. E., y Williams, R. J. (1985). *Learning internal representations by error propagation* (Inf. Téc.). California Univ San Diego La Jolla Inst for Cognitive Science.

Russell, S. J., y Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Salleb-Aouissi, A. (2017). *Introduction to Machine Learning — 5.1 Machine Learning Concepts — Material del curso CSMM.101x — edX.* Descargado 2018-11-02, de https://courses.edx.org/courses/course-v1:ColumbiaX+CSMM.101x+1T2017/courseware/

f034da4969be4eeeb0215dc177406c28/3ff0223d18734c7e942e371812ebd606/1?activate{_}block{_}id=block-v1{%}3AColumbiaX{%}2BCSMM.101x{%}2B1T2017{%}2Btype{%}40vertical{%}2Bblock{%}40989282658f67416facd

Sateesh Babu, G., Zhao, P., y Li, X.-L. (2016). Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life. En (pp. 214–228). Descargado de http://link.springer.com/10.1007/978-3-319-32049-6{_}29http://link.springer.com/10.1007/978-3-319-32025-0{_}14 doi: 10.1007/978-3-319-32025-0_14

Saxena, A., Goebel, K., Simon, D., y Eklund, N. (2008, oct). Damage propagation modeling for aircraft engine run-to-failure simulation. En *2008 international conference on prognostics and health management* (Vol. 41, pp. 1–9). IEEE. Descargado de http://www.embase.com/search/results?subaction=viewrecord{&}from=export{&}id=L71115746{%}5Cnhttp://dx.doi.org/10.1515/jpm-2013-2003{%}5Cnhttp://sfx.aub.aau.dk/sfxaub?sid=EMBASE{&}issn=03005577{&}id=doi:10.1515{%}2Fjpm-2013-2003{&}atitle=Growth+factors+in+pregnancies+complicatehttp://ieeexplore.ieee.org/document/4711414/ doi: 10.1109/PHM.2008.4711414

Scaruffi, P. (2016). *Intelligence is not Artificial*. Descargado de http://www.scaruffi.com

Shanno, D. F. (1970). Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, *24*(111), 647–656.

Siemens. (2013). Pictures of the Future. , 59. Descargado de www.eiemens.com/pof

Sifonte, J., y Reyes-Picknell, J. (2017). *Reliability Centered Maintenance-Reengineered*. 6000 Broken Sound Parkway NW, Suite 300 Boca Raton, FL 33487-2742: CRC Press. Descargado de https://www.taylorfrancis.com/books/9781498785198 doi: 10.1201/9781315207179

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, *550*(7676), 354–359. Descargado de http://dx.doi.org/10.1038/nature24270 doi: 10.1038/nature24270

Stone, P., Brooks, R., Brynjolfsson, E., Calo, R., Etzioni, O., Hager, G., ... Teller, A. (2016). Artificial Intelligence and Life in 2030. *One Hundred Year Study on Artificial Intelligence: Report of the 2015-2016 Study Panel*, 52. Descargado de https://ai100.stanford.edu/sites/default/files/ai{_}100{_}report{_}0901fnlc{_}single.pdf doi: https://ai100.stanford.edu

Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., y Clune, J. (2017). Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. Descargado de http://arxiv.org/abs/1712.06567 doi: 1712.06567

Tsang, A. H. (1995, sep). Condition-based maintenance: tools and decision making. *Journal of Quality in Maintenance Engineering*, *1*(3), 3–17. Descargado de http://www.emeraldinsight.com/doi/10.1108/13552519510096350 doi: 10.1108/13552519510096350

Vilalta, R., y Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial Intelligence Review*, *18*(2), 77–95. doi: 10.1023/A:1019956318069

Werbos, P. J. (1981). Applications of advances in nonlinear sensitivity analysis. En *Proceedings of the 10th ifip conference, 31.8 - 4.9, nyc* (pp. 762–770).

Yoon, A. S., Lee, T., Lim, Y., Jung, D., Kang, P., Kim, D., ... Choi, Y. (2017, sep). Semi-supervised Learning with Deep Generative Models for Asset Failure Prediction. Descargado de http://arxiv.org/abs/1709.00845

Yoshikawa, H., Zhang, Z., Instrumentation, A. D., y Plants, N. P. (2014). *Progress of Nuclear Safety for Symbiosis and Sustainability*. Descargado de http://link.springer.com/10.1007/978-4-431-54610-8 doi: 10.1007/978-4-431-54610-8

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhang, C., Lim, P., Qin, A. K., y Tan, K. C. (2017). Multiobjective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2306–2318. doi: 10.1109/TNNLS.2016.2582798

Zhao, G., Zhang, G., Ge, Q., y Liu, X. (2016). Research advances in fault diagnosis and prognostic based on deep learning. *Prognostics and System Health Management Conference (PHM-Chengdu), 2016*, 1–6.

Zio, E. (2016). Some Challenges and Opportunities in Reliability Engineering. *IEEE Transactions on Reliability*, *64*(29 Jun 2017), 1769–1782. doi: 10.1109/TR.2016.2591504