UNIVERSIDADE DA CORUÑA USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA Universidade de Vigo

Trabajo Fin de Máster

Análisis estadístico de datos para el mantenimiento predictivo de maquinaria industrial

Pedro Brasa Estévez

Máster en Técnicas Estadísticas
Curso 2018-2019

Propuesta de Trabajo Fin de Máster

Título en galego: Análise estatístico de datos para o mantemento preditivo de maquinaria industrial.
Título en español: Análisis estadístico de datos para el mantenimiento predictivo de maquinaria industrial.
English title: Statistical data analysis for predictive maintenance in industrial machinery.
Modalidad: Modalidad B.
Autor/a: Pedro Brasa Estévez, Universidad de Vigo.
Director/a: Javier Roca Pardiñas, Universidad de Vigo.
Tutor/a: Bruno Fernández Castro, Gradiant.
Breve resumen del trabajo: En este trabajo se tratará de diseñar algoritmos inéditos para la predicción de valores de fallo de maquinaria con el objetivo de realizar una comparativa de varios métodos y funciones de penalización. Se llevará a cabo investigando la precisión de los modelos sobre el conjunto de datos de prueba, validando así un método para su aplicación en el análisis de datos.
Recomendaciones: Conocimientos de regresión y clasificación.
Otras observaciones: Se realizarán los estudios numéricos sobre datos simulados.

Don Javier Roca Pardiñas, profesor de la Universidad de Vigo y don Bruno Fernández Castro, Responsable Técnico de Machine Generated Data Analytics (MGDA) | Área de Sistemas Inteligentes en Red (INetS) de Gradiant informan que el Trabajo Fin de Máster titulado

Análisis estadístico de datos para el mantenimiento predictivo de maquinaria industrial

fue realizado bajo su dirección por don Pedro Brasa Estévez para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, dan su conformidad para su presentación y defensa ante un tribunal.

En Vigo, a 4 de Julio de 2019.

El director: El tutor:

Don Javier Roca Pardiñas Don Bruno Fernández Castro

El autor:

Don Pedro Brasa Estévez

Contenido

Resumen

Resumen en español

Se denomina industria 4.0 a la revolución de automatización que se vive actualmente en la industria manufacturera. Dentro de esta revolución tiene un gran peso el mantenimiento predictivo, que busca predecir los fallos de máquina antes de que tengan lugar. En este trabajo se utilizarán algoritmos de *machine learning* para crear un modelo de regresión que permita predecir estos fallos de máquina de forma que no prediga valores de vida útil (RUL) superiores a los reales.

Para ello utilizaremos un modelo de dos capas, con una primera capa de *Random Forest* para tratar los datos y así lleguen a la segunda capa de XGBoost con el que probaremos distintas penalizaciones en la función objetivo. También se probará con una penalización en la función de evaluación de XGBoost buscando minimizar el RMSE de la regresión evitando predecir RUL por encima de su valor real.

También se compararán los dos algoritmos (*Random Forest* y XGBoost) para un modelo de clasificación que arrojará valores de 1 cuando se predigan menos ciclos para el fallo de máquina de los considerados. Además se compararán todos estos resultados y se elegirán los modelos con mejor precisión.

English abstract

Recently the industry 4.0 has been growing developing in a automatization revolution in the manufacturer industry. In this revolution one of the most important developments is the predictive maintenance, which tries to predict machine breakdown before it happens. In this project two machine learning algorithms will be used for creating a regression model that allows this breakdown predictions to not be higher than the actual Remaining Useful Life (RUL) is.

We will create a two-step model. The first step runs a Random Forest algorithm for data transformation and the result starts the second step, an XGBoost algorithm in which several penalty functions will be checked. Part of them will be used for penalizing the objective function of the algorithm while others will penalize the evaluation function. The goal for this models is minimize RMSE while trying to not predict RUL values over their actual value.

Both algorithms (Random Forest and XGBoost) will be evaluated in a classification model. This modelo will obtain a value of 1 when the RUL predicted are lower than a fixed value. All this results will be checked looking for the best model for RUL prediction.

Prefacio

Los avances tecnológicos han generado un aumento de productividad desde la primera revolución industrial que se basó a finales del s.XVIII en plantas de producción mecánica basadas en energía obtenida de agua y vapor. La segunda revolución industrial se produce a principios del s. XX con la producción en masa gracias a la energía eléctrica. La tercera se produjo en 1970 con la automatización en la producción gracias a los dispositivos electrónicos y a internet [17]. Actualmente nos encontramos en la cuarta era de avance tecnológico, el alza de la tecnología industrial digital conocida como Industria 4.0 que nace a partir de 9 avances principales mostrados en la Ilustración 1 entre los que se encuentran técnicas en desarrollo actualmente como *Big Data*, simulación e *Internet of Things*.

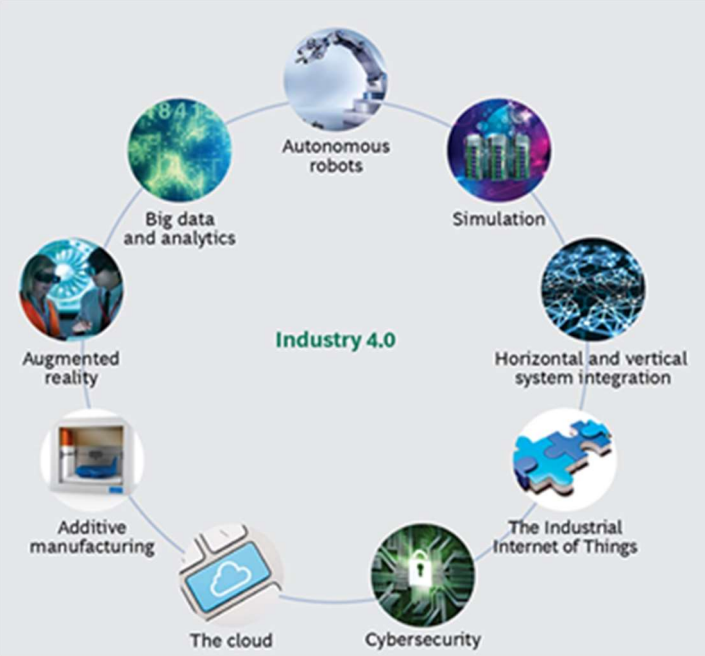

Ilustración 1. Los nueve avances que dan lugar a la Industria 4.0 [14].

Las principales ideas de la Industria 4.0 se publicaron en 2011 por Kagermann e impulsado por la Academia Nacional de la Ciencia y Tecnología de Alemania que publicó un manifiesto de industria 4.0 en 2013 [10]. A nivel europeo, la Asociación Público-Privada para Fábricas para el Futuro (FoF) desarrolla los temas de industria 4.0. En EE.UU. las cuestiones relacionadas con esta revolución son llevadas a cabo por el Consorcio de Internet Industrial (ICC).

Esta revolución se centra en mejorar la conectividad entre los diferentes sistemas de una fábrica para que cualquier elemento pueda comunicarse y compartir

información con el resto de componentes del sistema. El objetivo de la Industria 4.0 es alcanzar un mayor grado de eficiencia operacional y productividad gracias entre otras cosas a una mayor automatización [22], proporcionando mayor flexibilidad, optimización de tiempos y reducción de costes.

Internet of Things es un paradigma tecnológico que se basa en la conectividad de los sistemas donde a la maquinaria física, a través de dispositivos electrónicos como RFID, sensores y similares integrados en la red de información, proporcionan datos en tiempo real del estado de la maquinaria, proceso de producción y similares a otras máquinas o modelos dentro del sistema industrial, manteniéndolos en comunicación y facilitando la automatización de procesos.

De esta forma el IoT potencia las llamadas fábricas inteligentes (smart factories) mediante la incorporación de la tecnología a la toma de decisiones relacionadas con la producción. En este contexto, la estadística y las metodologías de aprendizaje automático (o *Machine Learning*) tienen la capacidad de convertir los datos de fabricación, obtenidos a través de los sensores que monitorizan en tiempo real los procesos productivos, en información útil para el desarrollo de una amplia diversidad de aplicaciones como, por ejemplo, el control de calidad de las manufacturas, la predicción de paradas imprevistas en máquinas, la reducción de descartes, etc.

Capítulo 1

Introducción

En este primer capítulo se trata de exponer el contexto que da lugar al trabajo desarrollado. Para ello primero se explica lo que es Gradiant, el centro tecnológico que ha planteado el proyecto. Después se expondrá brevemente el problema para, a continuación, introducir los métodos utilizados y los objetivos concretos del trabajo.

1.1. ¿Qué es Gradiant?

Gradiant es una fundación privada sin ánimo de lucro que tiene como objetivo mejorar la competitividad de las empresas mediante la transferencia de conocimiento y tecnologías en los ámbitos de la conectividad, inteligencia y seguridad. Con 101 profesionales y 11 patentes registradas, Gradiant ha desarrollado 249 proyectos diferentes de I+D+i, convirtiéndose en uno de los principales motores de la innovación en Galicia. Su facturación en 2016 alcanzó los 4,4 millones de euros y prevé superar los 5,5 millones en 2017.

El Centro se conforma a partir de un patronato que agrupa a representantes del sector público y privado. Está formado por las Universidades de A Coruña, Santiago de Compostela y Vigo; las empresas Arteixo Telecom, Egatel, Indra, R, Telefónica, Grupo Televés, y la Asociación empresarial INEO.

El compromiso del centro con la calidad es una constante desde sus inicios. Nueve meses después de comenzar su actividad logró la Certificación de Gestión de Calidad UNE-EN ISO 90001:2008 y al año siguiente obtuvo la Certificación en los Sistemas de Gestión de Proyectos de I+D+i UNE 166002. Tan solo un año después, en 2011, Gradiant consiguió ser incluido en el exigente registro estatal de Centros de Innovación Tecnológica (CIT).

Tras diez años de actividad, Gradiant se sitúa como socio tecnológico de la industria orientado a sus necesidades en el ámbito de las TIC, aportando su experiencia nacional e internacional en tecnologías para la seguridad y la privacidad; el procesado de señales multimedia; internet de las cosas; la biometría y analítica de datos y los sistemas de comunicaciones avanzadas.

SECTORES

SECTORES	LÍNEAS DE TRABAJO
INDUSTRIA 4.0	Conectividad mediante IoT industrial (IoT) y sistemas empotrados
	Sistemas de comunicaciones ad-hoc para entornos industriales
	Data Analytics y Big Data para análisis, optimización de procesos, predicción de eventos, etc
	Despliegue y gestión de infraestructuras TI y Cloud
	Seguridad de la información (confidencialidad, autenticación); controles biométricos de acceso
MARKETING, RETAIL Y AUDIOVISUAL	Identificación biométrica de cara, voz y firma manuscrita en dispositivos móviles
	Clasificación demográfica de audiencias y públicos para marketing y retail
	Procesado de lenguaje natural (PLN) para análisis de opinión e interacción con clientes

MARKETING, RETAIL Y AUDIOVISUAL	Data Analytics para marketing digital, perfilado de clientes y públicos objetivo
	Affective Computing
RECURSOS NATURALES Y SECTOR PRIMARIO	Despliegue escalable de sensores IoT para monitorizar procesos en explotaciones agroganaderas y acuícolas
	Plataformas para sistemas de control y monitorización de agro-ganaderas
	Integración de redes heterogéneas y APIs de alto nivel para provisión de servicios IoT y sistemas embebidos para Smart Farming.
	Comunicaciones avanzadas y arquitecturas de red para comunicación en tiempo real
	Data Analytics para detección de patrones, predicción de eventos optimización de procesos
	Análisis multimedia para monitorización y vigilancia
AEROESPACIAL	Comunicaciones avanzadas, subsistemas de comunicaciones y procesado de señales
	Navegación y Posicionamiento

AEROESPACIAL	Sistemas Complementarios: Detección y seguimiento de drones, simuladores, sistemas de tierra para el apoyo a la navegación precisa
	Sistemas embarcados y sensórica
	Data Analytics: explotación de datos y simulación de eventos discretos
	Procesado de señales multimedia: análisis inteligente de vídeo interpretación de escenas, vigilancia, monitorización y navegación
	Middleware
SEGURIDAD Y DEFENSA	Sistemas de comunicación seguros
	Seguridad biométrica
	Seguridad de infraestructuras y fronteras mediante el procesado de multimedia.
	Sistemas criptográficos avanzados para procesado seguro de información
	Privacidad: protección de información de carácter personal

SEGURIDAD Y DEFENSA	Data Analytics para detección de anomalías
SOCIEDAD DIGITAL Y EDUCACIÓN	Identificación y seguridad Biométrica
	Desarrollos basados en Data Analytics para eLearning y adaptive learning
	Aplicaciones TIC para el ocio
	Análisis semántico y Procesado de Lenguaje Natural (PNL)
	Tecnologías para la seguridad y privacidad de administraciones y ciudadanos
	Desarrollos para Smart Cities basados en conectividad, sensórica data analytics
	Sistemas de distribución multimedia
SALUD Y BIENESTAR	Algoritmos de procesado de señales biomédicas, de detección de caídas de detección de agitación en pacientes encamados
	Sistemas sin contacto (basados en tecnología UWB) para la medición de señales biomédicas

SALUD Y BIENESTAR	Bioinformática: análisis estándar de datos de secuenciación masiva (NGS)
	Desarrollo de sistemas mHealth, incluyendo la integración de dispositivos sensores y el desarrollo de pasarelas de comunicación
	Sistemas de acceso biométrico
	Anonimización de datos sensibles y medición del nivel de privacidad alcanzado
	Procesado de datos en el dominio cifrado
	Soluciones de control de acceso, cifrado, integridad, gestión de claves, etc.
	Diseño de sistemas seguros y apoyo en la definición de requisitos privacidad y seguridad
	Capacidades en sistemas de apoyo a la decisión clínica (CDSS), Big Data reconocimiento de patrones y data mining
TELECOMUNICACIONES	Procesado de señal (RF/analógica y digital)
	Subsistemas de Comunicaciones: móviles y por satélite

TELECOMUNICACIONES	Ingeniería del espectro
	Posicionamiento y navegación
	Sistemas embarcados
	Sistemas basados en sensores

TECNOLOGÍAS

TECNOLOGÍAS	LÍNEAS DE TRABAJO
SEGURIDAD Y PRIVACIDAD	Procesado y almacenamiento datos sensibles en entornos no confiables
	Videoanalytics for surveillance
	Multimedia Security and Forensics
CLOUD	Tecnologías para autoadministración y despliegue ágil de sistemas
	Gestión de Clouds (públicos/privados/híbridos)

	Virtualización de escritorios (Desktop as a Service)
DATA ANALYTICS	eLearning
	Human Generated Data Analytics
	Machine Generated Data Analytics
	Videoanalytics
	Health Data Analytics
BIOMETRIA	Face Recognition
	Handwritten Signature Recognition
	Speaker Recognition
	Liveness Detection
	Biometric Template Protection
IoT Y EMBEDDED SYSTEMS	Aplicaciones para Internet de las Cosas
	Middleware para Sistemas empotrados y embarcados

	Redes (6LoWPAN, descubrimiento de servicios, etc.)
	Seguridad IoT
COMUNICACIONES AVANZADAS Y PROCESADO DE SEÑAL	Subsistemas de Comunicaciones
	Ingeniería del espectro
	Posicionamiento y Navegación
PROCESADO DE SEÑALES MULTIMEDIA	Document Forensics
	Document traceability
	Document integrity authentication
	Image & Video Forensics
	Image & Video Authentication

Las cinco áreas técnicas (Información Multimodal, Comunicaciones Avanzadas, Sistemas Inteligentes, Seguridad y eSalud) en las que se divide la capacidad innovadora de Gradiant trabajan de manera transversal, convirtiendo el apoyo, el conocimiento y el trabajo compartido entre áreas en uno de los valores más importantes del Centro.

1.2. Explicación del problema

El objetivo del mantenimiento predictivo es conseguir anticipar los fallos en las máquinas y equipos de fabricación antes de que estos se produzcan y así optimizar las revisiones y minimizar las disminuciones paradas de producción de forma controlada.

El mantenimiento se basa en que toda maquinaria falla en algún momento y minimizar las pérdidas derivadas de estos fallos es un factor clave en la industria. Se han ido incorporando distintas estrategias de mantenimiento con este objetivo [20]. La más simple y todavía muy extendida es el llamado mantenimiento por fallo, que consiste en reparar la máquina únicamente cuando falle.

El método más extendido en la actualidad es el llamado mantenimiento preventivo, que consiste en planificar revisiones y reparaciones de los sistemas de forma periódica en función de la experiencia con el fallo de la maquinaria. Esta forma de mantenimiento permite evitar un fallo que en algunos casos podría ser muy grave, como se hace con vehículos o ascensores. En el caso de la industria permite redistribuir de forma planificada la producción, minimizando pérdidas al no tener una máquina parada de forma imprevista.

El Mantenimiento Predictivo (PdM), también llamado *Condition-Based Maintenance* (CBM) o *Prognostic and Health Management (PHM)*, busca detectar y predecir de manera automática los posibles fallos en manufacturas, equipos y sistemas, de manera que puedan anticiparse con tiempo suficiente las debidas acciones correctivas como la retirada del producto antes de que llegue al cliente, la reparación o el reemplazo de componentes. Esto puede conducir a importantes ahorros de costos, mayor previsibilidad y mayor disponibilidad de los sistemas.

El mantenimiento predictivo se puede afrontar como un problema basado en la clasificación o como un problema de regresión. El primer enfoque proporciona únicamente una respuesta en forma de alerta en caso de que el modelo estime el fallo de máquina en menos tiempo de uno estipulado de antemano. El segundo necesita más datos para ser preciso pero proporciona más información sobre el momento en el ocurrirá el fallo del equipo.

Para cualquiera de las aproximaciones (clasificación o regresión) resulta apropiado el uso de técnicas de aprendizaje automático basadas en agrupaciones de árboles de decisión, muy utilizadas con éxito durante los últimos años en el desarrollo de modelos complejos de datos para la detección y caracterización de patrones en diferentes ámbitos como la seguridad o la industria, entre otras. El uso amplio de

Mantenimiento Predictivo en EE.UU. ha hecho ahorrar a las empresas unos 35 billones de dólares anuales [12].

Numerosas publicaciones han trabajado en el mantenimiento predictivo en los últimos años. En este proyecto se pretende desarrollar métodos de ensemble learning con árboles de decisión para la predicción de fallos.

1.3. Métodos a aplicar

En este trabajo se propone la utilización de dos algoritmos de *machine learning* basados en *ensembles* de árboles de decisión: Random Forest y XGBoost. En este proyecto se utilizarán y evaluarán dos estrategias distintas: Un modelo de clasificación basado en los métodos anteriores; un modelo de regresión de varias etapas, combinando *Random Forest* y *XGBoost* para anticipar fallos de máquina.

1.4. Objetivos

Los objetivos de este trabajo:

1. Desarrollo de un modelo de regresión de mantenimiento predictivo para prevenir la rotura de máquina.
2. Desarrollo de un modelo de clasificación de mantenimiento predictivo para prevenir la rotura de máquina.
3. Evaluación y comparación de los modelos generados.

Capítulo 2

Metodología

Para este proyecto se han utilizado los algoritmos Random Forest y XGBoost tanto para la regresión como para la clasificación. Ambos son muy utilizados en el estudio de variables y regresión y clasificación por ser herramientas potentes con una base sencilla de predictores débiles como son los árboles de decisión. En este apartado se explican los métodos utilizados y su estado del arte, haciendo especial hincapié en la importancia que estos algoritmos tienen en la actualidad.

2.1. Ensemble Learning

En este royecto se trabajó a través de los métodos de *ensembles* o agrupaciones de árboles de decisión. Esta técnica es muy extendida dentro del aprendizaje supervisado. Se basa en la unión de distintos árboles predictores para generar un único predictor más robusto.

Las técnicas de aprendizaje supervisado son aquellas que deducen la función a partir de datos de entrenamiento. Con estas técnicas se trabaja a partir de datos distribuidos en pares de observaciones (x, y), siendo x un vector de observaciones con l variables estudiadas mientras que y representa un valor de etiqueta que puede ser numérica (regresión) o categórica (clasificación).

Los árboles de decisión son una técnica estadística con múltiples usos como la clasificación, la reducción de datos, discretización de variables, etc. La naturaleza de los datos que entran y salen de un árbol de decisión puede ser continua o discreta en función de la naturaleza del problema. Los árboles de decisión constan de nodos internos en los que se realiza un test sobre algún valor o condición de alguna de las variables de entrada. Los nodos internos están unidos a través de ramas por las que se movería la observación resolviendo test hasta llegar a los nodos hoja. En los nodos hoja se obtiene un resultado para la variable objetivo en función del recorrido hecho en el árbol.

Los árboles de decisión son estimadores débiles, pero los *ensembles* de estos árboles se convierten en un estimador más fuerte y fiable. En estos casos se generan numerosos árboles para cada observación a estudiar y cada uno de los árboles ofrece un resultado. A continuación cada árbol suma un voto al resultado objetivo que propone y se selecciona como válido.

Hay dos formas más habituales de construir *ensembles* de árboles de decisión: *Bagging* y *Boosting*.

2.2. Bagging

El *Bagging* (*Bootstrap AGGregatING*) [3] es un algoritmo de votación que se basa en remuestreos bootstrap [7]. Una muestra bootstrap se genera a partir de un remuestro de m datos aleatoriamente de forma equiprobable y con reemplazamiento de entre las n observaciones de las que se parte. Se generan así la cantidad que se considere de submuestras con n datos que constan de $m < n$ observaciones diferentes.

Se genera un árbol para cada submuestra y se hacen predicciones con todas ellas, cada una con un predictor débil distinto y se genera un predictor que para cada observación predicha le asignará el valor más votado por cada uno de los árboles.

La probabilidad de que una observación concreta sea seleccionada de una muestra bootstrap es de $1 - \left(1 - \frac{1}{m}\right)^m$. Para valores grandes de m, este valor es aproximadamente 63.2%. Debido a esto, cada submuestra bootstrap es diferente de las demás por lo que no hay ningún tipo de dependencia entre los árboles, lo que evita el sobreajuste.

2.2.1. Random Forest

El Random Forest [4] [17] fue propuesto por Breiman (2001). En este algoritmo, se construyen numerosos árboles mediante *bagging* de la muestra original, por lo que se seleccionan aleatoriamente los datos a estudiar en cada árbol de entre todos los de la muestra con repetición hasta tener M submuestras de tamaño n al igual que los datos originales.

En este procedimiento también se seleccionan aleatoriamente las variables que se estudian en cada árbol de entre todas las variables que hay en los datos. De este

modo los árboles son diferentes unos de otros tanto en los datos a estudiar como en las variables utilizadas.

De esta forma este algoritmo genera M árboles incorrelados. Esta elevada aleatoriedad hace a los modelos *Random Forest* muy robustos ante el sobreajuste. Una vez se construyen los árboles del bosque, cada nueva entrada se realiza a través del *ensemble* generado por el algoritmo mediante el voto mayoritario del conjunto de árboles (clasificación) o el valor promedio de las salidas del *ensemble* (regresión).

2.3. Boosting

Propuesto por Freund y Schapire (1996) [9] es una técnica perteneciente al campo del aprendizaje automático o *Machine Learning*. Los algoritmos de *Boosting* cambian la distribución del conjunto de entrenamiento en función de la precisión de los predictores anteriores. Es un modelo aditivo, es decir, predice el nuevo dato con un voto ponderado de los árboles individuales. Estos algoritmos seleccionan el modelo y los pesos de la votación de modo que el resultado final se ajuste bien a los datos.

Se basa en la premisa de generar una regla de predicción altamente precisa combinando un conjunto de reglas débiles e imprecisas, como los árboles de decisión.

Estos clasificadores débiles se añaden de manera iterativa, siguiendo un proceso de aprendizaje a partir de los datos de entrada, de modo que cada una de sus salidas tenga diferente peso en función de la exactitud de sus predicciones. Cada vez que se añade un nuevo clasificador débil al conjunto, los datos de entrada cambian su estructura de pesos: los casos que han sido mal clasificados ganan peso estadístico, mientras que los que fueron clasificados correctamente lo pierden.

De esta manera cada predictor incorporado más recientemente trabaja para corregir los errores cometidos anteriormente. Tras suficientes repeticiones el *ensemble* converge hacia un modelo capaz de mejorar la precisión de cada predictor por separado.

En la ilustración 2 se muestra un ejemplo de clasificación a través del *boosting*. En diferentes etapas, el algoritmo reduce el peso de los valores bien clasificados y aumenta el de los mal estimados. Al cabo de suficientes repeticiones el modelo habrá dado los pesos de forma que clasifique de la mejor forma posible.

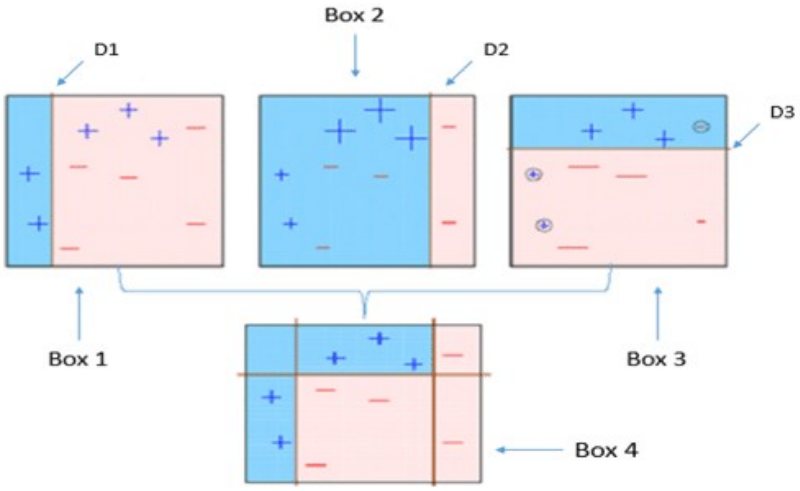

Ilustración 2. Ejemplo de clasificación a través del método boosting [1].

2.3.1. XGBoost

Uno de los algoritmos de boosting más recientes es el XGBoost (*EXtreme Gradient BOOSTing*) [5]. Es un método de código abierto muy utilizado en retos y trabajos de *data mining* como las competiciones de *machine learning* de Kaggle donde en 2015, 17 de los 25 métodos presentados utilizaron XGBoost para su resolución. Además en KDDCup 2015 los 10 equipos con mejor puntuación utilizaron este algoritmo.

A través del *boosting*, el XGBoost ejecuta K iteraciones donde en cada una se añade un nuevo árbol de decisión que intentará corregir el error cometido en las anteriores. En cada iteración del algoritmo XGBoost se añade un nuevo árbol de decisión, que se desarrolla con el objetivo de minimizar el error acumulado por los árboles anteriores.

El objetivo del XGBoost es minimizar el error de predicción optimizando la siguiente función:

$$obj(\theta) = \sum_{i}^{N} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Donde el primer término es una función que representa el error cometido entre las predicciones estimadas por XGBoost y los valores reales. Esta función por defecto suele ser el error cuadrático medio (MSE) en el caso de regresión o la precisión (ACC) para clasificación. Este primer término es personalizable por el usuario en función del objetivo del modelo concreto, se explica con más detalle esta personalización en el Capítulo 3.

El segundo término de la fórmula es una función que penaliza la complejidad del modelo para reducir el sobreajuste del modelo, ya que es un problema recurrente al trabajar con XGBoost. Este sobreajuste se debe a la relación de dependencia estadística entre los árboles construidos. Además, el tiempo de construcción del modelo aumenta al aumentar el número de iteraciones.

Los árboles de XGBoost siguen una estructura que ejecuta en cada nodo interno la bifurcación de una variable de la forma $variable < umbral$. Clasifica de esta forma las variables de entrada repetidamente hasta el nodo hoja correspondiente La variable comprobada en cada nodo y sus umbrales se llevan a cabo de forma que minimice lo máximo posible la función objetivo. Una vez en cada nodo hoja se obtiene un valor para la variable en cuestión, el resultado final es el promedio ponderado en función de los pesos de cada árbol del *ensemble* como se puede ver en la Ilustración 3.

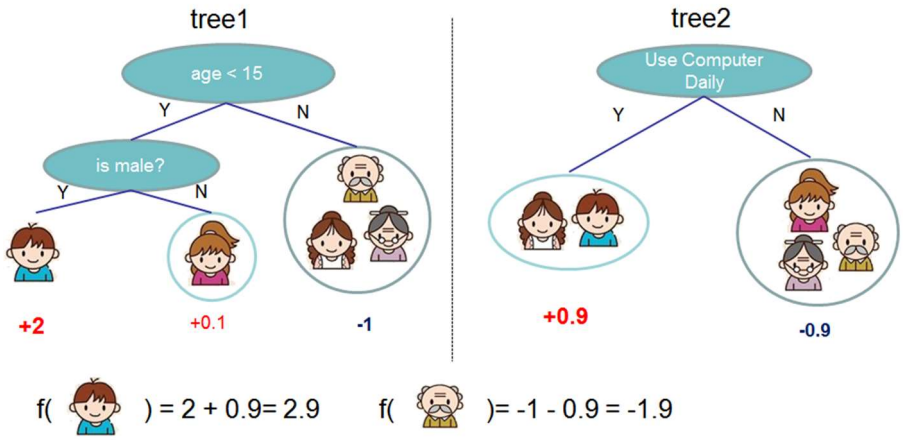

Ilustración 3. Ejemplo sencillo de XGBoost. Comprueba variables en cada nodo interno hasta llegar al nodo hoja obteniendo así un valor para las variables de entrada [5]

Por lo tanto el XGBoost es un algoritmo potente en comparación con otros métodos de *ensemble learning* pero tiene una alta tendencia al sobreajuste y es muy sensible al ruido en los datos.

Tiene una gran ventaja sobre otros modelos y es que permite añadir o modificar, además de los hiperparámetros del algoritmo para ajustar mejor el modelo, funciones objetivo o de evaluación que mejor se ajusten al estudio a realizar.

2.4. Métricas de evaluación

Se emplearán las siguientes métricas para evaluar las prestaciones de los algoritmos de regresión y clasificación: la raíz cuadrada del error cuadrático medio (RMSE) en regresión y el Área Bajo la Curva (AUC) [8] para regresión.

2.5. Recursos informáticos

Para el diseño, ejecución y diagnóstico de los algoritmos a trabajar en este proyecto se utiliza la herramienta estadística R [16], muy extendida a nivel mundial para el tratamiento estadístico en el campo del *data mining* y *machine learning*. R proporciona una gama amplia de paquetes estadísticos.Los principales paquetes utilizados en este proyecto fueron: mlr [2] [19], orientado especialmente al machine learning, selección de características y construcción de *ensembles*; Random Forest [13]; y XGBoost [6].

Capítulo 3

Descripción del algoritmo

La vida útil restante (*Remaining Useful Life, RUL*) de una máquina se define como el tiempo (o número de ciclos de fabricación) que resta hasta que se produzca un fallo inesperado en el equipo debido a, por ejemplo, la rotura o degradación de alguno de sus componentes. Este concepto de RUL es muy utilizado en investigación operativa [21].

La estimación del RUL es uno de los factores más importantes del mantenimiento predictivo ya que permite predecir fallos antes de que surjan y actuar de forma planificada.

En este proyecto se ha trabajado a partir de la propuesta de Zhao (2017) que empleaba redes neuronales para la caracterización y predicción del fallo de máquina en datos simulados de la NASA [23]. En este trabajo se ha continuado esa línea de investigación empleando técnicas de *ensemble learning* basadas en árboles de decisión, para construir dos tipos de aplicaciones diferentes que estimen el valor de vida útil que le quede a las máquinas a partir de la información obtenida de un conjunto de sensores que monitoricen su estado.

Para ello se han desarrollado dos tipos de modelos de mantenimiento predictivo para la predicción de RUL basados en técnicas de regresión y clasificación, respectivamente. En el presente capítulo se explican ambos enfoques, empleando los algoritmos XGBoost y *Random Forest*.

3.1. Personalización de XGBoost

XGBoost tiene una particularidad de la que carece el algoritmo *Random Forest* y es la posibilidad de configurar algunos de los hiperparámetros del algoritmo con el objetivo de optimizar su rendimiento y prestaciones.

En este proyecto nos hemos centrado en estudiar dos factores concretos que se pueden modificar en XGBoost. La función objetivo (*objective*) y la función de evaluación (*feval*).

Definición de la función objetivo

El algoritmo XGBoost trabaja iterativamente generando un conjunto de árboles de decisión que buscan minimizar la función objetivo explicada en el Capítulo 2. Para ello, la reestructuración de los pesos en cada nodo hoja (de los que dependerán las predicciones realizadas) se hará en función del resultado de minimizar esta función en cada paso.

A continuación se muestra de nuevo la función objetivo de XGBoost ya citada en el Capítulo 2. El primer término es una función que representa el error cometido entre las predicciones estimadas por XGBoost y los valores reales, que por defecto es el MSE. Este es el término de la función que podemos particularizar.

$$obj(\theta) = \sum_{i}^{N} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

El segundo término es una función de regularización para minimizar el sobreajuste. Esta función es importante porque en cada iteración del algoritmo XGBoost se añade un nuevo árbol de decisión, configurando su "estructura" de manera tal que se minimice la función objetivo: es decir, que corrige el error de predicción cometido hasta el momento. Es importante destacar que este proceso iterativo (*Boosting*) introduce una relación de dependencia estadística entre los árboles construidos, lo que generalmente conlleva la aparición de sobreajuste u *overffiting*, de ahí la necesidad de incluir (entre otros mecanismos) la función de regularización

Las funciones de error por defecto en el algoritmo XGBoost son el MSE al hacer regresión y la precisión (ACC) en clasificación. Estas funciones de error penalizan todos los errores cometidos de la misma forma. Este comportamiento, sin embargo, no es deseable en ciertos escenarios, como por ejemplo en el desarrollo de aplicaciones de mantenimiento predictivo, ya que es imprescindible garantizar que el modelo sea capaz de anticipar todas las averías de la máquina. En este contexto, es de gran utilidad emplear XGBoost particularizando la función objetivo de manera que, durante el entrenamiento de los modelos, se penalice el efecto de predecir valores de RUL por debajo encima de su valor real. Utilizando el paquete de XGBOOST en R se puede definir una función que se ajuste mejor al objetivo del estudio (definiendo su primera y segunda derivada).

Definición de la función de evaluación

La función de evaluación (o *feval* en el paquete de xgboost de R) permite definir la métrica con la que se miden las prestaciones del algoritmo durante la etapa de entrenamiento. Para definir una función de evaluación solo hay que definir en este caso la función que se pretende evaluar en las iteraciones.

La función *feval* es útil combinándola con el parámetro *early_stopping_rounds*. Este comando no tiene valor por defecto. Al definirlo se le da un valor k entero y detendrá el proceso de entrenamiento del algoritmo cuando se produzcan k iteraciones consecutivas en las que no mejore el resultado obtenido con la función de evaluación. El proceso es como sigue:

1. XGBoost calcula la estructura (pesos) del árbol minimizando la función objetivo (MSE).

2. XGBoost evalúa el nuevo *ensemble* con la función definida en *feval*.

3. Si las prestaciones del *ensemble* (evaluadas mediante la función definida en *feval*) no mejoran tras k (valor de *early_stopping_rounds*) veces, el proceso de entrenamiento se detiene. En caso contrario, el proceso de construcción del ensemble continúa.

Teniendo esto presente, una técnica interesante para penalizar el efecto de predecir valores de RUL por encima de su valor real es construir una función *feval* convexa dependiente de RMSE y del número de predicciones que no anticipan los diferentes fallos de máquina. Al principio el valor de esta *feval* decrecerá con cada iteración (porque el RMSE tendrá más peso en las primeras iteraciones), pero llegará un momento en que el RMSE será más bajo a costa de que no anticipar un mayor número de errores, y por tanto la función *feval* empezará a aumentar. a partir de entonces, y al no haber mejora durante k (valor de *early_stopping_rounds*) iteraciones, el proceso de entrenamiento se detiene. El algoritmo habrá ajustado su estructura de manera que se minimice el RMSE cometido teniendo en cuenta que debe anticipar los errores.

3.4. Algoritmo de Mantenimiento predictivo basado en regresión.

El algoritmo de regresión para la estimación de RUL consta de 2 capas. Consta de una primera capa de regresión de *Random Forest*, a partir de sus resultados se realiza una segunda capa de varios modelos de regresión mediante XGBoost con una función objetivo especial para penalizar la predicción de valores de RUL más

grandes que los reales. Por último se realiza la media de las distintas predicciones de esta última capa para obtener el valor final de las observaciones.

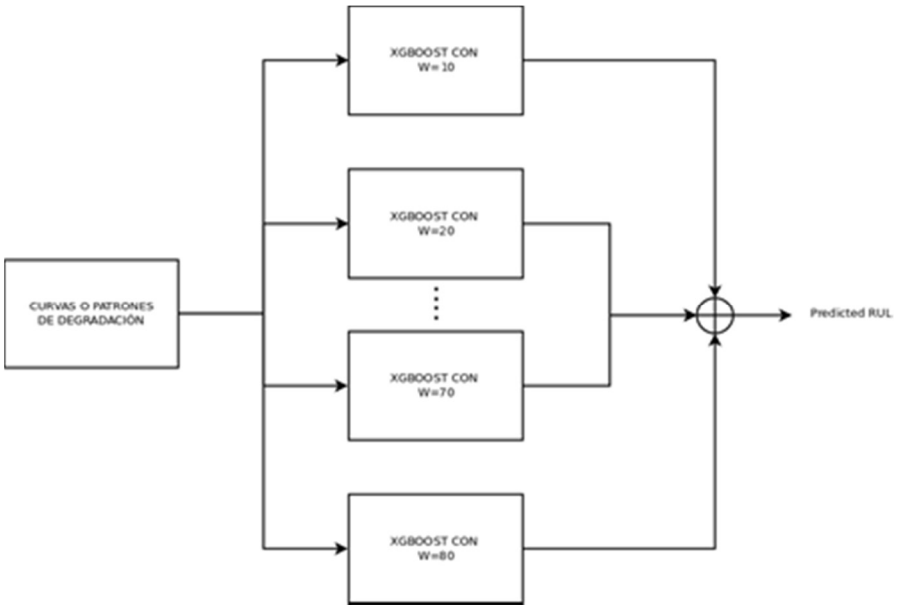

Ilustración 4. Esquema del modelo de dos capas de regresión

Descripción capa 1

El objetivo de la primera capa de la aplicación consiste en construir un modelo de regresión para obtener las curvas de degradación de los diferentes patrones de fallo observados en los datos de entrenamiento.

El primer nivel está compuesto por un único modelo de regresión (*Random Forest*), encargado de reducir la dimensión del conjunto de datos, convirtiendo los datos de entrada (variable multivariante con los sensores que monitorizan el estado de la máquinas) asociados a cada patrón de fallo en una curva normalizada unidimensional que represente la evolución del estado de la máquina desde su funcionamiento normal hasta el instante de tiempo en el que se produce cada avería.

Como el método propuesto es supervisado, es necesario construir las etiquetas de los datos con los que se entrenará el modelo. Para ello se definen dos estados: estado OK (nivel 1), que representa que la máquina funciona correctamente; y estado NOK (nivel 0), que representa que la máquina está próxima al fallo.

Para el entrenamiento de esta capa, y siguiendo la recomendación de Malinowski (2015) [25], se seleccionarán solo un conjunto de observaciones iniciales (aquellas más próximas al estado OK) y finales de los datos de entrada, con el único objetivo de evitar el ruido en el modelo de regresión. El número de observaciones iniciales y

finales a tener en cuenta se define, en ambos casos, como 5% de la longitud total de los datos de entrada.

La salida de esta capa serán las curvas de degradación de cada patrón de fallo observado. Estas curvas muestran el patrón de degradación de la máquina a lo largo de los ciclos con valores entre 1 y 0 donde 1 es que funciona perfectamente y 0 que la máquina falla o está a punto. Un ejemplo se muestra en la Ilustración 4.

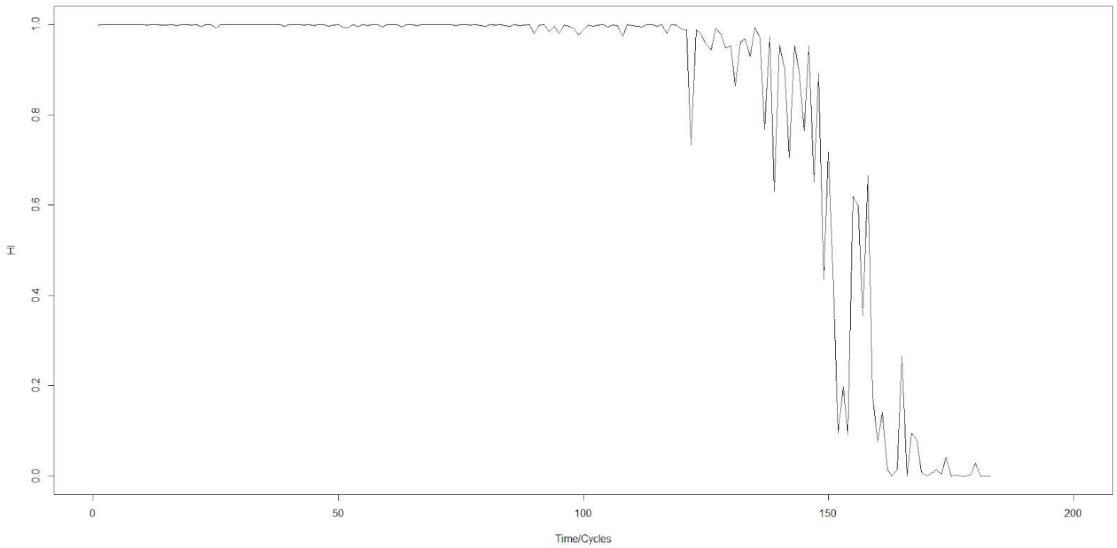

Ilustración 5. Curva de degradación del primer patrón de fallo observado. 1 indica que en ese punto la máquina funciona correctamente y la proximidad al 0 indica la proximidad al fallo.

Descripción capa 2

El bjetivo de la segunda capa de la aplicación consiste en obtener un modelo de regresión que, a partir de las curvas de degradación obtenidas en la capa anterior, aprenda a identificar el patrón de fallo y el estado actual de la maquinaria.

Esto es, un modelo que sepa identificar no solo la curva de degradación, sino la posición (ciclo o instante de tiempo) dentro de ella en la que se encuentra la máquina.

Esta segunda etapa del algoritmo la componen un conjunto N de modelos de regresión (XGBoost) que, a partir de las curvas caracterizadas en la etapa anterior, son entrenados para predecir los ciclos de fabricación restantes hasta que se produzca el siguiente fallo de máquina.

Durante el entrenamiento de cada modelo, se define una ventana temporal de tamaño W que se desliza en pasos de una unidad temporal a lo largo de cada curva, desde el inicio (estado normal de funcionamiento de la máquina) hasta el final

(instante en el que se produce la avería del equipo). Las variables de entrada a cada modelo serán los W valores de la curva contenidos en la ventana deslizante en cada instante de tiempo, mientras que la variable objetivo a predecir será el valor de RUL en cada caso, esto es, el número de ciclos desde la última muestra contenida en la ventana en cada instante de tiempo hasta el final de la curva correspondiente.

La siguiente imagen ilustra el proceso de entrenamiento de los modelos de regresión de esta etapa.

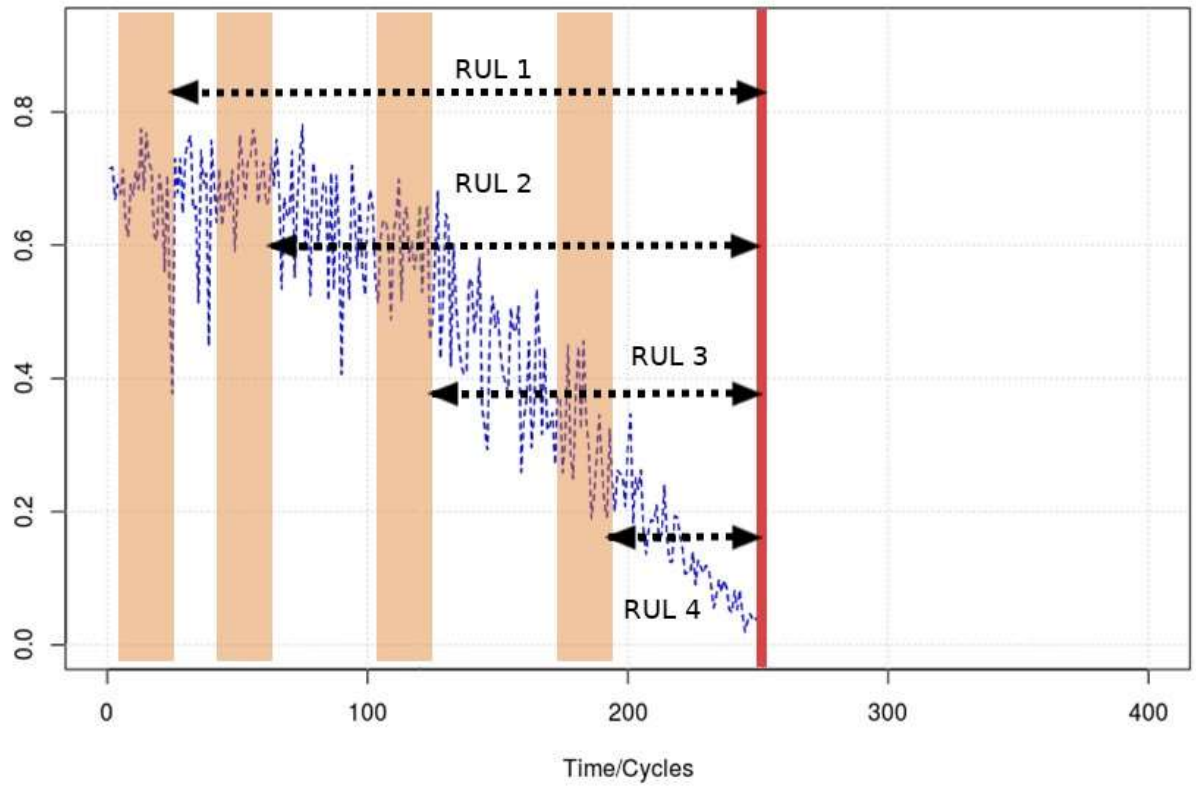

Ilustración 6. Ejemplo de distintas ventanas en distintos momentos de la curva de degradación (franjas naranjas). Los valores de RUL se calculan como los ciclos restantes desde el último dato de la ventana hasta el fallo representado por la línea roja.

Cada modelo de esta etapa será entrenado con una ventana temporal de tamaño W diferente (por ejemplo, W∈ {10,50,80,100,}).

Durante el entrenamiento de estos modelos se utiliza una función de coste particular que pondera el error de regresión cometido (RMSE) y el número de predicciones que exceden el valor real de RUL.

A la hora de elegir el máximo de tamaño de la ventana se hizo una serie de comprobaciones previas que mostraban que para ventanas mayores de 80 datos el conjunto de observaciones es demasiado grande y arroja resultados generales debido a que casi todos los datos estarían en el mismo conjunto. Lo inverso ocurre para ventanas por debajo de W=10, los conjuntos de datos son demasiado pequeños y

los resultados son muy próximos a los obtenidos si se utiliza cada observación de forma individual.

Por ello la prueba se realiza con ventanas de amplitudes desde 10 hasta 80, con las que se obtienen 8 grupos de predicciones distintas a partir del conjunto de comprobación. Una vez se tienen los 8 conjuntos de predicciones se hará la media de éstos para obtener la predicción final del valor de RUL.

Este tipo de predicción con el promedio de diferentes anchos de ventana no se ha realizado hasta ahora en la literatura y es un enfoque novedoso con el que se ha trabajado en este proyecto.

Para la evaluación de este sistema se utilizará el RMSE y una métrica que representa el número de predicciones realizadas por el algoritmo que no han sido capaces de anticipar la avería.

3.5. Descripción del algoritmo de Mantenimiento predictivo: Clasificación

Se ha implementado un algoritmo de clasificación para el mantenimiento predictivo. La idea fundamental de esta aproximación es, a partir de las series multivariantes de datos de entrada que representan los valores de los sensores instalados en una máquina, estimar la probabilidad de que la máquina falle en los próximos ciclos de fabricación.

Se ha definido para este algoritmo una variable objetivo que tendrá valor de 1 cuando la máquina esté aproximándose al fallo y 0 cuando no esté suficientemente próxima a dicho fallo.

La variable objetivo a predecir será, por tanto, un vector que representa la clase de cada muestra: clase 1 si se predice un fallo de máquina en los próximos A ciclos; clase 0 si, por el contrario, su estado no prevé una parada imprevista en los próximos A ciclos. Este vector, representa la probabilidad de fallo de la máquina durante la próxima ventana temporal de tamaño A.

En nuestro experimento este valor A se ha fijado entre 10 y 100 ciclos restantes. El mínimo seleccionado es 10 porque se ha considerado que un aviso de fallo con menos de 10 ciclos de trabajo restantes es poco tiempo y no tiene utilidad práctica. No se han elegido más de 100 ciclos restantes porque los datos con los que se ha trabajado en este trabajo tienen pocas observaciones cuyo RUL sea superior a 100.

Capítulo 4

Pruebas

En los modelos de mantenimiento predictivo, como en cualquier algoritmo de aprendizaje automático, es importante evaluar los métodos desarrollados y comprobar sus prestaciones y funcionamiento. En esta sección se explican con detalle el conjunto de datos de entrada que se ha utilizado durante la implementación del algoritmo, así como las distintas pruebas realizadas y los resultados obtenidos.

4.1. Descripción de los datos de prueba

En nuestro caso de estudio se utilizan los datos de *Turbofan Engine Degradation Simulation Data Set* publicados por la NASA [18] (no disponibles en la fecha de entrega de este trabajo) muy utilizados para entrenar y evaluar algoritmos de mantenimiento predictivo [11,15]. Estos datos están formados por varias series de tiempo multivariantes obtenida a través de una simulación de distintos momentos del mismo sistema de motores de reacción *turbofan*. Los datos de simulación de la degradación se han obtenido con C-MAPSS, que se puede utilizar para simular condiciones realistas de trabajo de este tipo de motores. La NASA ha simulado cuatro conjuntos de datos distintos según diferentes condiciones operacionales y modos de fallo. En nuestro caso, y por acotar el alcance de este trabajo y la exposición de sus resultados, se ha trabajado con el conjunto de datos número 1, que solo considera un tipo de fallo y un modo operacional con 100 fallos de máquina distintos y 100 observaciones de test.

Tabla 1. Resumen de los distintos conjuntos de datos publicados por la NASA y sus principales características.

Número de experimento	#1	#2	#3	#4
Número de modos de fallo	1	1	2	2
Número de condiciones operativas	1	6	1	6
Número de unidades de entrenamiento	100	260	100	249
Número de unidades de test	100	259	100	248

Los datos con los que se trabajó constaron en un principio de 28 variables, entre las que están el número de fallo de máquina al que pertenece la observación, que va de 1 hasta 100, y el número de ciclo de cada observación dentro de ese fallo de máquina.

El dataset está compuesto por 100 eventos de fallo, o averías, diferentes. Cada uno de estos eventos está formado, a su vez, por un conjunto de variables (C_1,..., C_{27}) que representan la información capturada de los 27 distintos sensores que monitorizan el estado la máquina a lo largo del tiempo. La variable Ciclo representa el número de ciclos de trabajo realizados con éxito hasta la ocurrencia de la avería (instante K_1,..K_{100} en cada uno de los eventos de fallo correspondientes).

Tabla 2. Ejemplo de matriz de datos de fallos de máquina con los datos obtenidos de n sensores.

Fallo	C_1	...	C_{27}	Ciclo (t_i)
FALLO 1	$\mathbf{X}_{1,1}$	...	$\mathbf{X}_{27,1}$	1
	$\mathbf{X}_{1,2}$	...	$\mathbf{X}_{27,2}$	2
	...	...	...	...
	$\mathbf{X}_{1,K1}$	...	$\mathbf{X}_{27,K1}$	K_1
FALLO 2	$\mathbf{X}_{1,1}$	...	$\mathbf{X}_{27,1}$	1
	...	...	...	...
	$\mathbf{X}_{1,K2}$	...	$\mathbf{X}_{27,K2}$	K_2
...	...	...	...	...
FALLO 100	$\mathbf{X}_{1,1}$	...	$\mathbf{X}_{27,1}$	1
	...	...	...	...
	$\mathbf{X}_{1,K100}$	...	$\mathbf{X}_{27,K100}$	K_{100}

Como paso previo a las pruebas se han eliminado de los datos de entrenamiento y de test las variables que no dan resultado o tienen varianza 0 entre las distintas observaciones ya que no son útiles para la predicción. También se han eliminado las observaciones que se ha comprobado que eran datos atípicos. Una vez hecho, las mismas variables suprimidas del entrenamiento también lo serán del test quedando al final de este proceso un *dataset* de entrenamiento de 19149 observaciones y 16 variables mientras que el test está formado por 13096 observaciones de las mismas 16 variables.

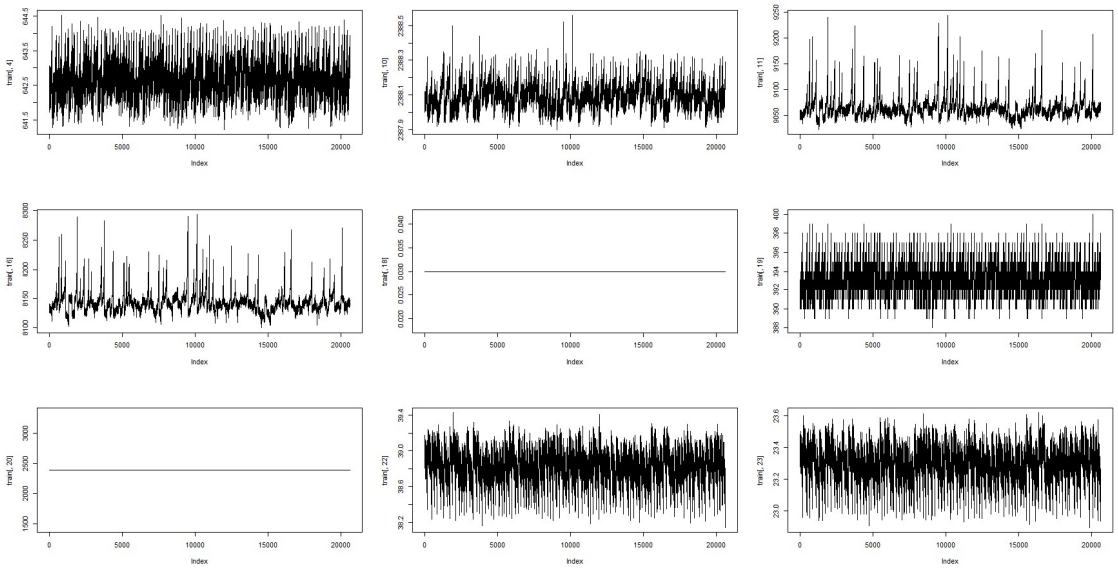

Ilustración 7. Ejemplo de lecturas de distintos sensores de los datos. Algunas no sirven para identificar degradación.

4.2. Pruebas realizadas

4.2.1. Prueba 1. Penalización cuadrática.

El algoritmo *XGBoost* aprende progresivamente de las repeticiones anteriores buscando la forma de optimizar la función objetivo, evaluando los resultados en cada etapa del procedimiento de boosting con una función de evaluación. De esta forma en la ejecución normal sin ningún tipo de restricción buscará minimizar el error cuadrático en la predicción (función objetivo) mientras evalúa utilizando el RMSE. Se utilizarán en esta prueba distintas funciones de penalización utilizadas habitualmente como función objetivo. Estas funciones se clasifican en dos grupos: los métodos de penalización exterior y los métodos barrera o de penalización interior. En el caso de la penalización exterior se ha probado con la penalización cuadrática.

El objetivo de utilizar estas funciones de penalización viene impuesto por el problema que se pretende resolver, esto es, predecir los fallos de máquina antes de

que estos se produzcan. En este contexto, es importante penalizar aquellas predicciones que, aún teniendo un error de regresión pequeño, no consigan anticipar la avería. Es por este motivo por lo que se ha decido emplear XGBoost. Este algoritmo permite particularizar la función objetivo a optimizar durante la etapa de entrenamiento de manera que se induzca a aprender aquellos patrones o reglas que permitan predecir el fallo de máquina antes de que suceda.

La penalización cuadrática añade una función de restricción $g \leq 0$ a un problema de optimización. Al tener en nuestro caso la intención de que los valores predichos de RUL no sean mayores que los RUL reales, la función de penalización sería:

$$g: \hat{y} - y \leq 0$$

Donde g es la función de error de la predicción, siendo $\hat{y}$los valores de RUL predichos por el modelo e y el valor real. Para añadir la función de penalización cuadrática al primer término de la función objetivo, se debe elevar esta función al cuadrado y añadirle un coeficiente α que determine la influencia de la penalización en la función objetivo. Cuanto más alto sea el valor de este parámetro más peso tendrá la penalización con respecto a la función objetivo. La función quedaría como sigue:

$$f_1 = (\hat{y} - y)^2 + \alpha(\hat{y} - y)^2$$

Para añadir a la programación la función objetivo también ha sido necesario añadir las funciones derivada y segunda derivada de esa fórmula en función de $\hat{y}$, como se ha explicado en el capítulo anterior.

Resultado Prueba 1

A continuación se muestran algunos de los resultados obtenidos al probar distintos valores de α con la penalización cuadrática como función objetivo. En la tabla se muestran el N y el RMSE obtenidos para cada uno de los valores.

Tabla 3. Extracto de algunos de los resultados del mantenimiento predictivo añadiendo una función objetiva de penalización cuadrática para distintos valores de α.

Función objetivo	α	RMSE	N
f_1	$\alpha_1 = 0.01$	26.77	58
	$\alpha_2 = 1$	26.24	56
	$\alpha_3 = 100$	47.43	9

Tras la realización de las pruebas se ha comprobado que la variación del valor de α afectaba demasiado a la predicción. Las distribuciones de las predicciones son similares como se puede ver en la Ilustración 8 pero la variación de α hace que las predicciones tengan un N menor y un RMSE mayor. Esta variación es casi lineal y genera que para un α suficientemente grande, algunas predicciones de ciclos son negativas lo cual no es posible. En el primer caso, para el valor $\alpha = 0.01$ el N=56 y RMSE=21.27; para el valor $\alpha = 1$, N=54 y RMSE=20.92 y para el valor $\alpha = 100$, N=1 y RMSE=52.28.

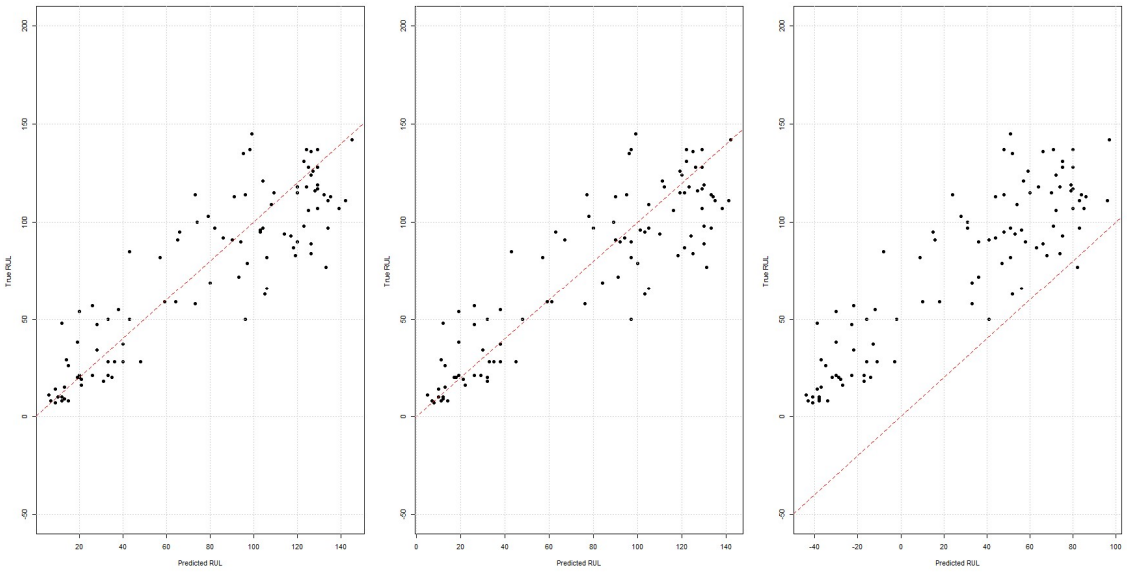

Ilustración 8. Comparación de predicciones de RUL para los valores de α 0.01, 1 y 100 respectivamente. Los puntos representan los valores predichos de RUL para cada observación y la línea discontinua roja representa el caso ideal en el que las predicciones y los valores reales de RUL coinciden.

4.2.2. Prueba 2. Penalización inversa.

La otra función que se ha probado como función objetivo con restricción pertenece al grupo de los métodos barrera. En este apartado se ha utilizado la penalización inversa.

La función inversa se utiliza aplicando una función $g_i < 0$ a la función objetivo de la forma $\frac{-\alpha}{g_i}$. Así, a medida que las predicciones son más altas que los valores reales, la fracción tendrá un valor más bajo afectando de menor manera a la función objetivo. Las funciones que sean más bajas generarán una penalización mayor a la función objetivo y por lo tanto el algoritmo trabajará para minimizar estos casos. En este caso la función objetivo es de la siguiente forma:

$$f_2 = (\hat{y} - y)^2 - \frac{\alpha}{\hat{y} - y}$$

Donde α vuelve a ser un coeficiente que mide el aporte de la penalización a la función objetivo, $\hat{y}$ vuelve a ser la predicción de RUL e y es el valor real de RUL.

Resultado Prueba 2.

Se comprobaron diferentes valores de α para evaluar la función objetivo con penalización inversa.

Tabla 4. Extracto de algunos de los resultados del mantenimiento predictivo añadiendo una función objetiva de penalización inversa para distintos valores de α.

Función objetivo	α	RMSE	N
f_2	$\alpha_1 = 0.01$	27.72	15
	$\alpha_2 = 0.1$	33.47	10
	$\alpha_3 = 0.2$	35	8
	$\alpha_4 = 0.4$	36.70	7

En este caso un α menor a 0.1 disminuye el valor de N aumentando el RMSE en menor grado que la penalización cuadrática. Al variar los valores de α se comprueba que los mejores valores para con menor N están en los valores de $\alpha = 0.2$ con N=8 y RMSE=35 y $\alpha = 0.4$ con N=7 y RMSE=36.7. Al ser el RMSE de ambas opciones próximo se considera como opción prioritaria reducir N lo máximo posible.

4.2.3. Prueba 3. Función de evaluación.

Como tercera prueba se ha empleado el valor de MSE como función objetivo de XGBoost (por defecto para este algoritmo). Definiéndose, por el contrario, una función de evaluación personalizada. Esta función aplica una penalización al RMSE proprocional a la cantidad de valores predichos por encima del valor auténtico RUL. De esa forma si la función de evaluación no mejora en consecutivas iteraciones, XGBoost dejará de ejecutarse para arrojar como mejor resultado la menor función objetivo conseguida hasta entonces. La función es como sigue:

$$f_3 = RMSE + \alpha_e \frac{n}{T}$$

Siendo n el número de predicciones de RUL cuyo valor es mayor que el valor real de RUL ($\widehat{RUL} > RUL$) y T el número total de datos. α_e es un coeficiente que cuantifica la contribución de la penalización a la función de evaluación análogo al α de las pruebas anteriores.

De esta forma, aumenta el RMSE tanto como α_e multiplicado por la proporción de predicciones de RUL que son mayores que los valores reales. Esta función no se ha utilizado como función objetivo porque el algoritmo *XGBoost* necesita una función objetivo derivable dos veces en función de $\hat{y}$ pero esta función no lo es.

Resultado Prueba 3

En esta prueba se ha aplicado la función de evaluación indicada variando el valor de α_e y se estudiaron los valores de RMSE y N resultantes:

Tabla 5. Extracto de algunos de los resultados del mantenimiento predictivo añadiendo una función objetiva de penalización inversa para distintos valores de α.

α_e	RMSE	N
0.5	20.07	20
1	32.09	4
10	46.32	2

Los valores de N obtenidos en esta prueba son menores que en las dos pruebas anteriores. En valores más altos de α_e se reduce el N hasta valores próximos al 0 pero aumentamdo el RMSE. Por ello se considera mejor modelo el que tiene $\alpha_e = 1$, al obtener un N=4 frente a los 2 conseguidos con $\alpha_e = 10$ pero su RMSE es 32.09 frente a los 46.32. Las dos predicciones por debajo del valor real de RUL de diferencia entre ambos casos se ha considerado aceptable ya que permite reducir el RMSE en 14 unidades.

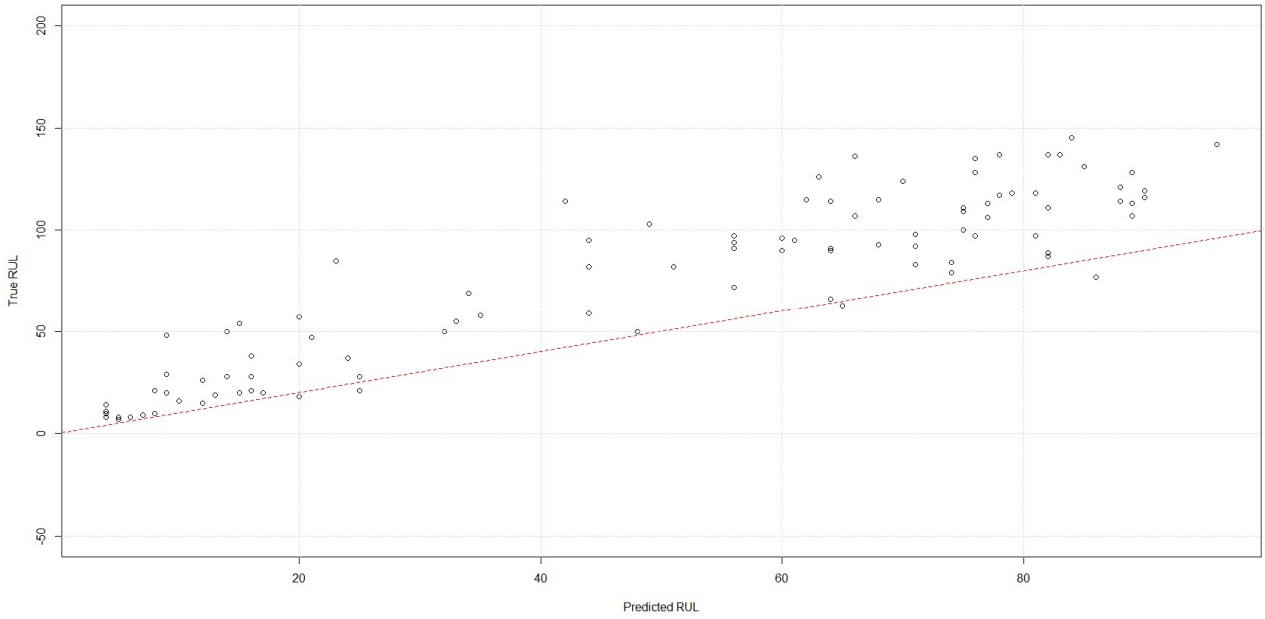

Ilustración 9. Diagrama de dispersión de la función de evaluación con $\alpha_e = 1$. Los puntos representan los valores predichos de RUL para cada observación y la línea discontinua roja representa el caso ideal en el que las predicciones y los valores reales de RUL coinciden.

4.2.4. Prueba 4. Función objetivo y evaluación penalizadas

Además de aplicar la función de evaluación directamente a un modelo con la función objetivo por defecto, se ha probado a combinar las Pruebas 2 y 3 para aplicar la función de evaluación penalizada a un modelo *XGBoost* con una función objetivo de la Prueba 2.

Resultado Prueba 4

Se han comprobado los resultados para diferentes valores de α. El valor de α_e se mantuvo constante fijado en el valor que arrojó un menor N y RMSE en la Prueba 3.

Tabla 6. Valores para distintos niveles de penalización al añadir una función objetivo con penalización inversa a la función de evaluación seleccionada anteriormente.

α	*RMSE*	*N*
10^{-5}	31.2234	7
10^{-4}	30.8873	4
0.001	29.1338	5
0.1	29.8633	15
10	28.2993	16

Se puede observar en la tabla 6 que para valores de α menores de 0.001 los resultados obtenidos a nivel de N y RMSE son muy similares. Además de ser próximos entre si, también lo son respecto al resultado de f_3 con $\alpha_e = 1$ de la prueba 3.

Comparado con dicho resultado de la prueba 3, el RMSE del $\alpha = 10^{-4}$ ha bajado de 32.09 a 30.8879 mientras que el valor N es igual en ambos casos y es el menor obtenido en las dos pruebas.

La reducción del RMSE se ha considerado insuficiente para el aumento de complejidad del modelo al añadir la función objetivo y la evaluación. Por este motivo, se considera como mejor modelo el que solo utiliza f_3 con $\alpha_e = 1$.

4.2.5. Prueba 5. Modelo de clasificación

En esta prueba se evalúa la implementación de un modelo de mantenimiento predictivo basado en clasificadores, En esta aproximación, las covariables o datos de entrada del modelo de clasificación serán los valores de los sensores que monitorizan el estado de la máquina y sus componentes, mientras que la variable objetivo a predecir será un vector que representa la clase de cada muestra: clase 1 si se predice un fallo de máquina en los próximos A ciclos; clase 0 si, por el contrario, su estado no prevé una parada imprevista en los próximos A ciclos.

En este caso se ha estudiado cómo afecta el valor de la ventana (A) a las prestaciones del algoritmo. Por medio de la métrica AUC y se comprueba si cualquier ventana es adecuada o alguna amplitud no clasifica correctamente los datos. En este caso no hay restricciones que aplicar, por lo que se prueba clasificando con *Random Forest* y con XGBoost y se comparan los resultados.

Resultado Prueba 5

Los resultados son muy similares en niveles de AUC entre Random Forest y XGBoost. Las ventanas por debajo 20 ciclos para la alerta y las que están por encima de 80 arrojan un valor menor de AUC debido a que las categorías están desbalanceadas en los datos con los que se ha llevado a cabo el proyecto.

El resto de resultados son similares independientemente del número de ciclos para la alerta y de la amplitud de ventana. Obteniéndose en esos casos un valor de AUC$\geq$0.9.

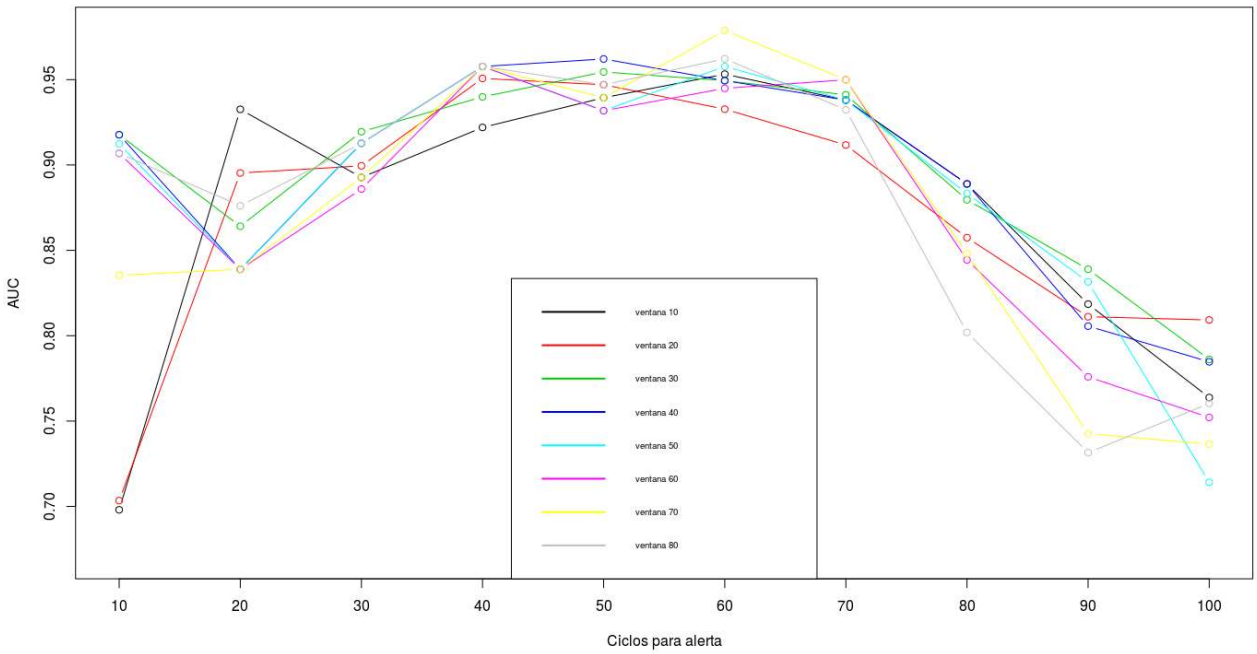

Ilustración 10. AUC del modelo de clasificación con Random Forest para distintas amplitudes de ventana y distintos ciclos para alerta.

Podría usarse cualquiera de los dos métodos y ambos obtienen el AUC más alto entre A=40 y A=60 ciclos llegando a sobrepasar un AUC=0.95. El XGBoost parece por las gráficas ser más sensible a la amplitud de ventana, bajando su AUC más abruptamente a por encima de W=60. En cuanto a la mejor amplitud de ventana no hay grandes diferencias, se podría usar cualquiera de ellas sin que haya grandes variaciones de resultados.

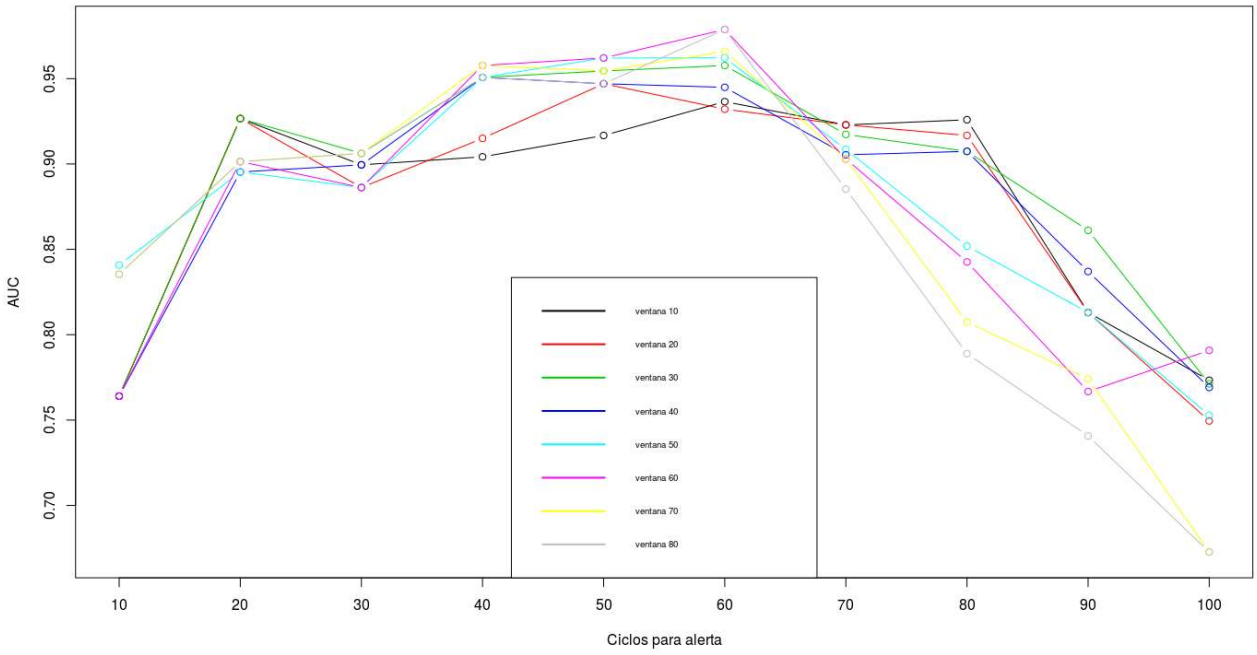

Ilustración 11. AUC del modelo de clasificación con XGBoost para distintas amplitudes de ventana y distintros ciclos para alerta.

4.3. Conclusiones

En este capítulo se han realizado diferentes pruebas con el objetivo de evaluar distintas estrategias que mejoren las predicciones de los valores de RUL mediante regresión y clasificación con *Random Forest* y XGBoost. Tras comparar la penalización cuadrática y la inversa en la función objetivo y una penalización personalizada en función de los valores mal predichos en la función objetivo se llegó a la conclusión de que el mejor modelo es el que utiliza f_3 como función de evaluación penalizada y con la función objetivo por defecto de XGBoost, que es el error cuadrático medio (MSE).

En cuanto a la clasificación no se observaron diferencias entre el uso de *Random Forest* y XGBoost ni en los tamañosde ventana utilizados para las predicciones. El único valor a tener en cuenta en la clasificación son los ciclos A por debajo de los cuales el algoritmo arroja el resultado de estar próximo al fallo. Si ese valor A crea dos clases desbalanceadas por ser un valor demasiado bajo o demasiado alto los modelos serán menos precisos. En el resto de casos su funcionamiento es muy prometedor con cerca de un AUC=0.95.

Capítulo 5

Discusión

En relación con el objetivo 1 se ha logrado desarrollar modelos de regresión fiables combinando *Random Forest* y XGBoost.

Respecto al objetivo 2 se ha desarrollado con éxito y buena efectividad un modelo de clasificación tanto con XGBoost como con *Random Forest.*

En la comparación del objetivo 3 se han probado diferentes configuraciones y parametrizaciones con el objetivo de obtener los mejores resultados anticipando el mayor número posible de averías. De todos los modelos desarrollados el que arrojó mejores resultados es el de la prueba 3 que solo utiliza la función de evaluación penalizada y el MSE como función objetivo.

5.1. Líneas futuras

Se proponen las siguientes líneas de investigación con el objetivo de profundizar y evolucionar los resultados obtenidos en este trabajo:

1. Los algoritmos han sido implementados y evaluados con datos simulados. Se propone emplear datos reales para comprobar y contrastar con un mayor grado de confianza su efectividad .
2. En el algoritmo de mantenimiento predictivo basado en un agrupaciones de modelos de regresión con XGBoost y distintos tamaños de ventana W, se propone añadir un etapa adicional con un nuevo modelo de regresión que calcule de manera automática, y a partir de las predicciones obtenidas en los modelos anteriores, la mejor ponderación posible entre los diferentes tamaños de ventana utilizados, de manera que se optimice el error cometi

Bibliografía

[1] Arora, U. (2010). Gradient Boosting Part1–Visual Conceptualization. https://dimensionless.in/gradient-boosting/. Accedido 2 julio 2019.

[2] Bischl, B., Lang, M.,Kotthoff, L., Schiffner, J., Richter, J., Studerus, E., Casali cchio, G. and Jones, Z. M. (2016) mlr: Machine Learning in R. Journal of Machin e Learning Research, 17 (170), 1-5.

[3] Breiman, L. (1996). Bagging predictors. Machine Learning 24, 123-140.

[4] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32

[5] Chen, T., and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).

[6] Chen, T, He, T., Benesty, M., Khotilovich,V., Tang, Y., Cho, H., Chen,K., Mit chell, R., Cano, I., Zhou,T., Li, M., Xie, J., Lin, M., Geng, Y. and Li,Y. (2018). x gboost: Extreme Gradient Boosting. https://CRAN.R-project.org/package=xgboo st. Accedido 10 octubre 2017.

[7] Efron, B and Tibshirani, R. (1993). An introduction to the bootstrap. Chapma n & Hall.

[8] Fawcett, T. (2006). An introduction to ROC analysis. Pattern recognition letters, 27(8), 861-874.

[9] Freund, Y. and Schapire, R. 1996. Experiments with a new boosting algorithm. Proceedings of the Thirteenth International Conference on Machine Learning. 148-156. Morgan Kaufmann.

[10] Kagermann, H., Lukas, W. and Wahlster, W. (2011) Industrie 4.0 - Mit dem Internet der Dinge auf dem Weg zur 4. industriellen Revolution. In: VDI Nachrichten, Issue 13.

[11] Le Son K, Fouladirad M, Barros A. Remaining useful life estimation on the nonhomogenous gamma with noise deterioration based on Gibbs filtering: a case study. In: Proceedings of the IEEE conference on prognostics and health management (PHM). Denver, CO: IEEE; 2012. p. 1–6.

[12] Lee J.(2003). Approaching zero downtime. The Center for Intelligent Maintenance Systems. Harbor Research Pervasive Internet Report;

[13] Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. R news, 2(3), 18-22.

[14] Lu, Y. (2017). Industry 4.0: A survey on technologies, applications and open research issues. Journal of Industrial Information Integration 6, p. 1-10.

[15] Malinowski S, Chebel-Morello B, Zerhouni N. Remaining useful life estimation based on discriminating shapelet extraction. Reliab Eng Syst Saf 2015;142:279–88.

[16] R Core Team (2018). R: A language and environment for statistical computin g. R Foundation for Statistical Computing, Vienna, Austria

[17] Rüssmann, M., Lorenz, M., Gerbert, P., Waldner, M., Justus, J., Engel, P. an d Harmisch, M. (2015). Industry 4.0. The future of productivity and growth in ma nufacturing industries. The Boston Consulting Group. April.

[18] Saxena, A. and Goebel, K. (2008). Turbofan Engine Degradation Simulation Data Set, NASA Ames Prognostics Data Repository. http://ti.arc.nasa.gov/projec t/prognostic-data-repository. Accedido 15 enero 2018.

[19] Schiffner, J., Bischl, B., Lang, M.,Kotthoff, L., Richter, J., Jones, Z. M., Prob st, P., Pfisterer, F., Gallo, M., Kirchoff, D., Kühn, T. and Thomas, J. (2016). Mlr Tutorial. Cornell University.

[20] Shin, J. H. and Jun, H. B. (2015). On condition based maintenance policy. Jo urnal of Computational Design and Engineering 2, 119-127.

[21] Si, X. S., Wang, W., Hu, C.H. y Zhou, D.H. 2011. Remaining useful life estimation. A review on the statistical data driven approaches. European Journal of Operational Research 213, 1-14.

[22] Thames, L., Schaefer, D. (2016). Software-defined cloud manufacturing for Industry 4.0. Procedia CIRP 52, p. 15-19.

[23] Zhao, Z., Liang, B., Wang, X. and Lu, W. (2017). Remaining useful life prediction of aircraft engine based on degradation pattern learning. Reliability Engineering & System Safety, 164, 74-83.