Mantenimiento predictivo de motores de buques mediante aprendizaje automático

David Novoa Paradela
CITIC
Universidad de A Coruña
A Coruña, España
david.novoa@udc.es

Carlos Eiras Franco
CITIC
Universidad de A Coruña
A Coruña, España
carlos.eiras.franco@udc.es

Óscar Fontenla Romero
CITIC
Universidad de A Coruña
A Coruña, España
oscar.fontenla@udc.es

Francisco Lamas López
CESADAR
Armada Española
Cartagena, España
flamlop@mde.es

***Resumen*—En los dominios industriales, la aparición de anomalías (fallos) durante el funcionamiento de los sistemas puede provocar la ocurrencia de comportamientos indeseados como paradas inesperadas y sus consecuentes pérdidas económicas. Las técnicas de mantenimiento predictivo se encargan de monitorizar el estado de los sistemas para llevar a cabo la detección de estas anomalías en fases incipientes, evitando situaciones de emergencia y grave impacto. Además, permiten asegurar el correcto funcionamiento de los sistemas y programar las actividades de mantenimiento de forma óptima, reduciendo costes.**

Los sistemas informáticos que incorporan los buques almacenan en tiempo real la información recogida por los sensores de los motores. Estos sensores miden aspectos físicos tales como temperaturas, presiones o vibraciones. El funcionamiento normal de estos motores se encuentra descrito por sus fabricantes mediante los valores de diseño (temperaturas máximas en los cilindros, presiones mínimas de los tanques, etc.), así como los diferentes modos de fallo que se pueden producir.

En este trabajo se analizarán los métodos de mantenimiento predictivo actuales para obtener una solución basada en técnicas de inteligencia artificial que lleve a cabo este proceso de forma automática para motores de buques. El sistema desarrollado será capaz de predecir la aparición de modos de fallos a partir de un histórico de datos de un motor. Además, para que su uso sea veloz y escalable a grandes flotas de embarcaciones, se ha implementado sobre el entorno distribuido Spark.

***Palabras clave*—mantenimiento predictivo, aprendizaje automático, Spark**

I. Introducción

Los costes de mantenimiento representan una parte importante de los costes operativos totales en entornos industriales. En algunos casos, como en la industria metalúrgica, estos costes pueden llegar a encontrarse entorno al 15 %-60 % de los costes totales de producción. Algún estudio sugiere que una tercera parte del dinero invertido en la gestión del mantenimiento se desperdicia como resultado de actividades innecesarias o incorrectas [1]. Además, la caída de un sistema puede llegar a suponer enormes costes económicos para las organizaciones implicadas. Por ejemplo, una caída de tan solo 49 minutos en el servicio de ventas de Amazon en 2013 se tradujo en una pérdida de cerca de cuatro millones de dólares.

La imposibilidad en el pasado de tratar eficientemente con los grandes y continuos flujos de datos que los sistemas industriales pueden llegar a producir, ha llevado a emplear, en muchos casos, técnicas basadas en tendencias estadísticas o en la aparición directa de fallos en los sistemas. El mantenimiento predictivo actual, sin embargo, sigue una filosofía más avanzada. En lugar de depender de estas estadísticas industriales (p. ej., tiempo medio entre fallos) para programar actividades de mantenimiento, lleva a cabo una monitorización en tiempo real del estado mecánico del sistema y de otros indicadores para determinar los tiempos medios de fallo reales. La capacidad de cómputo actual permite tanto procesar cantidades de datos mayores (ejecución distribuida), como la utilización de técnicas de aprendizaje automático más avanzadas para llevar a cabo las predicciones (regresión), la detección a tiempo de estados anormales en el sistema (detección de anomalías) y su diagnóstico (clasificación).

El mantenimiento predictivo se puede entender, por tanto, como un mantenimiento preventivo [2] condicionado al estado actual del sistema y a las prediciones del estado futuro realizadas a partir de un histórico de operación.

Los buques marítimos se caracterizan por ser embarcaciones de gran tamaño y potencia. Para poder desplazarse a altas velocidades utilizan motores de gran calibre, siendo estos el núcleo principal de las naves. Estos motores, formados por una numerosa cantidad de componentes, pueden llegar a ser verdaderamente complejos. En consecuencia, dado que el correcto funcionamiento de un buque depende directamente de los motores, es habitual llevar a cabo una rigurosa sensorización de sus componentes. Las naves incorporan sistemas diseñados para leer y registrar periódicamente el estado en el que se encuentra cada elemento del motor. Estos valores se corresponden mayoritariamente con aspectos físicos como temperaturas, presiones o vibraciones de elementos como cilindros o tanques de combustible.

Como sucede con los sistemas mecánicos de entornos industriales, estos motores precisan un mantenimiento específico y su disponibilidad y correcto funcionamiento es fundamental para las organizaciones que los utilizan. En este trabajo de investigación se presenta un sistema de mantenimiento predictivo para motores de buques basado en técnicas de aprendizaje automático y diseñado para su ejecución en entornos distribuidos.

II. Trabajo relacionado

En esta sección se analizan las técnicas de mantenimiento predictivo empleadas en la actualidad en base a Ran *et al.* [3].

II-A. Categorías en función del propósito

En base al criterio de optimización seguido, podemos distinguir entre:

- Minimización de costes: las acciones de mantenimiento se llevan a cabo de acuerdo a los resultados de la predicción de fallos, por lo que la métrica de coste empleada es habitualmente el tiempo de vida útil (RUL, Remaining Useful Life) de dicho sistema o equipamiento. También es posible definir un modelo de coste *ad hoc* para el sistema [4].
- Maximización de la disponibilidad y fiabilidad: las métricas que emplean son la fiabilidad (probabilidad de un sistema o equipamiento de encontrarse en un estado de funcionamiento normal dado un intervalo de tiempo) [5] y la disponibilidad (probabilidad de que el sistema se encuentre operativo) [6].
- Optimización multiobjetivo: enfoque que busca optimizar múltiples métricas de manera simultánea para lograr un mejor equilibrio entre los objetivos. Además de las antes mencionadas, emplean métricas como el riesgo, la seguridad o la viabilidad. Generalmente, es imposible obtener los valores óptimos para todos los objetivos al mismo tiempo, por lo que se han desarrolado una gran variedad de modelos multiobjetivo [7] [8] [9].

II-B. Categorías en función de la aproximación

En base al tipo de apoximación empleada, podemos distinguir entre:

- Aproximaciones basadas en conocimiento: métodos que emplean conocimiento experto a priori y procesos de razonamiento deductivo, p. ej., sistemas expertos y modelos basados en razonamiento. Estas aproximaciones se pueden clasificar a su vez en tres categorías:
    1. Basadas en ontologías: una ontología es una definición formal de tipos, propiedades, y relaciones entre conceptos de un dominio específico. Los modelos basados en ontologías permite integrar, compartir y reutilizar conocimientos entre entidades computacionales por medio de dichas ontologías. Son empleadas para establecer las bases de conocimiento de un sistema o dispositivo, habitualmente en combinación con otros algoritmos de razonamiento (p. ej., razonamiento basado en reglas). Se han construido diferentes ontologías para mantenimiento predictivo, como por ejemplo la propuesta por Konys [10], enfocada en el dominio de la evaluación de la sostenibilidad.
    2. Basadas en reglas: llevan a cabo la toma de decisiones en base a un conjunto de reglas preestablecido por un experto humano. Este conocimiento experto del dominio es empleado para la construcción de reglas habitualmente de tipo condicional [11]. El antecedente (condición) de la regla es un hecho o estado concreto que puede darse en el sistema monitorizado (p. ej., superar un umbral de temperatura), mientras que el consecuente suele ser una acción o salida que afecta al mundo real (p. ej., ordenar la reparación de un componente del sistema).
    3. Basadas en modelos: los enfoques basados en modelos tratan de vincular los procesos físicos de un sistema con modelos matemáticos, como modelos Gaussianos [12], modelos de sistemas lineales [13] o modelos de Markov [14].
- Aproximaciones basadas en aprendizaje automático tradicional: métodos de ML (*machine learning*) clásicos que han sido probados con éxito en el dominio del mantenimiento predictivo, precediento al enfoque *deep learning*. A continuación se mencionan las más habituales:
    1. Redes neuronales artificiales (ANN): paradigma clásico en aprendizaje automático que permite llevar a cabo predicciones y diagnóstico de fallos [15] generalmente con un alto rendimiento. Son capaces de resolver problemas no lineales, sin embargo, es habitual caer en problemas de sobreajuste o altos costes computacionales (elevado número de parámetros a entrenar).
    2. Árboles de decisión (DT): método probabilístico no paramétrico utilizado de forma no supervisada para resolver problemas clasificación y predicción. Existe una gran variedad de métodos basados en árboles de decisión [16], entre los que destaca el algoritmo Random Forest [17] por su bajo coste computacional y su estabilidad ante grandes conjuntos de datos.
    3. Máquinas de soporte vectorial (SVM): empleadas tanto de forma supervisada para resolver problemas de clasificación de fallos [18] como para detección de anomalías en su versión no supervisada (One-Class SVM) [19].
    4. Vecinos más cercanos (k-NN): método no supervisada utilizado habitualmente para la clasificación de fallos [20], aunque también existen métodos para la predicción de tiempo de vida útil (RUL) [21] o detección temprana [22] basados en los mismos principios.
- Aproximaciones basadas en *deep learning*: estas técnicas han demostrado ser en muchos casos superiores a las técnicas tradicionales resolviendo problemas de clasificación y predicción de fallos. A continuación se mencionan las más habituales:
    1. Autoencoder (AE): Los autoencoders son un tipo de red neuronal autoasociativa cuya capa de salida busca reproducir los datos presentados a la capa de entrada después de haber pasado por una fase de compresión dimensional. De esta forma, logran obtener una representación de los datos de entrada

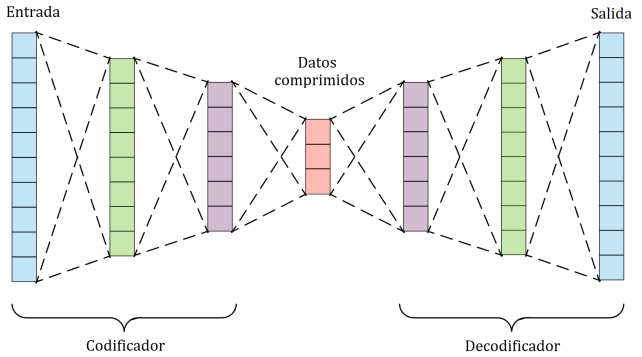

Figura 1: Ejemplo de arquitectura de red neuronal autoencoder.

en un espacio de dimensión menor al original, aprendiendo una representación compacta de los datos, reteniendo la información importante y comprimiendo la redundante. Por esta razón, son muy utilizados para la elaboración de modelos robustos ante el ruido, cualidad importante en detección de anomalías, y problemas de regresión [23]. En la Figura 1 se puede observar la arquitectura de una red autoencoder.

2. Long Short Term Memory Networks (LSTM): las LSTM son un tipo especial de redes neuronales recurrentes (RNN) [24]. Mientras que las RNN estándar pueden modelar dependencias a corto plazo, es decir, relaciones cercanas en la serie temporal de datos, las LSTM pueden aprender dependencias a más largo plazo, por lo que se podría decir que poseen una memoria mayor. Las RNN convencionales presentan problemas en su entrenamiento debido a que los gradientes que retropropagan a través de la red tienden a crecer enormemente, o a desvanecerse con el tiempo, debido a que el gradiente depende no solo del error presente sino también los errores pasados. La acumulación de errores provoca dificultades para memorizar dependencias a largo plazo. Estos problemas son solventados por las redes LSTM. Para ello incorporan una serie de mecanismos para decidir qué información va a ser almacenada y cuál borrada. Este tipo de redes son muy potentes para tareas de análisis de secuencias, por lo que su uso es habitual en tareas de mantenimiento predictivo [25].

III. Contextualización del problema

Esta sección resume las características principales del escenario real sobre el que se va a desarrollar el sistema de mantenimiento predictivo presentado en este trabajo.

III-A. Buques

Los buques para los que se desea desarrollar el sistema registran de forma periódica los valores medidos por los sensores del motor en documentos CSV. Cada motor se encuentra supervisado por cerca de 300 sensores (variables), representando una amplia variedad de aspectos físicos como la temperatura del gas, la presión en los filtros, las revoluciones del motor, la humedad ambiental o la velocidad y dirección del viento. Ya que se dispone de los históricos correspondientes a aproximadamente cuatro años de operación de varios buques, nuestro conjunto (o conjuntos) de datos inicial está formado por varios millones de instancias (filas) de cientos de variables (columnas).

Un problema de estos datos es que presentan una baja calidad para poder realizar un análisis automático de los mismos. Esto se debe principalmente a dos razones:

1. Algunas de las variables presentan un valor constante durante todo el registro de datos, por lo que esas variables no aportan información relevante para realizar un análisis del estado del motor. Esto puede ser debido a diversas causas tales como, por ejemplo, un fallo en el sensor de esa variable, un error de medida o que el sensor no se encuentre conectado correctamente.
2. Las variables registradas presentan, entre ellas, frecuencias de muestreo heterogéneas, variando incluso dentro de la propia variable en diferentes instantes de tiempo del histórico de datos. Esto provoca que no sea posible disponer de información simultánea de todas las variables en cada instancia (fila) del fichero de datos.

Los sensores no registran valores cuando la embarcación se encuentra en tierra, sin embargo, existen tres modos o estados en los que el motor se puede encontrar durante sus salidas a la mar en función de las revoluciones (RPM) a las que trabaja:

- Motor apagado: RPM cercanas a cero.
- Motor a ralentí: RPM aproximadas a un umbral μ.
- Motor en operación: RPM superiores a un umbral μ.

De este modo, aunque los históricos de datos se compongan de registros correspondientes a estados del motor en plena operación o a ralentí, también incluirán registros de instantes con el motor apagado que pueden no ser de interés si no aportan información relevante para analizar el estado del motor.

En cuanto a los fallos de funcionamiento (anomalías) existentes en los históricos de datos, debido a la magnitud de los archivos, su identificación y etiquetado manual por parte de un experto es inviable. De este modo, no se dispone de un conjunto de modos de fallo localizados cuya ocurrencia haya sido confirmada por un experto. Por tanto, si se desean aplicar técnicas de detección de anomalías será de forma no supervisada.

Se dispone de un documento FMECA (Failure Mode, Effects, and Criticality Analysis) que describe de forma teórica los modos de fallo que se pueden producir durante el funcionamiento del motor, las variables o elementos que intervienen en dichos modos y sus valores nominales, máximos y mínimos. Estos valores de diseño se pueden utilizar para generar datos normales o anómalos de manera artificial, entrenar un clasificador de modos de fallo, o directamente cruzar registros reales con el fichero.

III-B. *Entorno de trabajo*

Debido a que no se trata de una única embarcación, sino de una flota de buques, la gestión y control de los mismos se lleva a cabo de manera centralizada. Cada buque envía los datos registrados por sus sensores a un nodo central donde se almacenan. Por tanto, aunque el sistema se desarrolle para un único buque, su uso debe ser extrapolable a cualquier otra embarcación de la flota.

Ya que los conjuntos de datos con los que se va a tratar pueden llegar ser de gran tamaño, y para que el sistema sea escalable a un número elevado de buques, el sistema debe desarrollarse para ser ejecutado de forma distribuida. Se empleará el sistema de ficheros distribuido HDFS [26] para almacenar los datos y el framework Apache Spark [27] para llevar a cabo operaciones como el entrenamiento de los modelos de forma distribuida. El lenguaje de programación escogido por su amplio abanico de librerías para ML ha sido Python [28].

III-C. *Objetivo*

A continuación se recogen las funcionalidades principales que el sistema debe de implementar:

- Procesar los datos leídos por los sensores para adecuarlos al flujo ML desarrollado.
- Predecir el estado futuro de un motor a partir de un histórico de datos.
- Determinar si el estado de un motor en un instante de tiempo t es normal o anómalo, es decir, si se podría corresponder con un fallo.
- Determinar si un estado anómalo del motor se corresponde con algún modo de fallo recogido en el FMECA.

IV. Solución propuesta

La solución diseñada para el problema del mantenimiento predictivo en buques se puede entender como la suma de varias soluciones a subproblemas o tareas clásicas de ML. Por ejemplo, predecir el estado del motor tras dos meses de uso es una tarea totalmente independiente de la tarea de clasificación de modos de fallo, sin embargo la salida de una puede emplearse como la entrada de la otra. Es por ello que la arquitectura diseñada para resolver el problema general se compone de cuatro bloques o tareas claramente delimitadas (Figura 2):

1. Preprocesamiento: bloque encargado de cargar los ficheros de datos del sistema de archivos HDFS, filtrarlos, limpiarlos para solventar los problemas descritos en la sección III-A y prepararlos para ser utilizados en etapas posteriores.
2. Predicción: bloque encargado de llevar a cabo de forma distribuida una predicción del estado futuro de un motor a partir de un histórico de datos.
3. Detección de anomalías: bloque encargado de llevar a cabo de forma distribuida la detección de anomalías sobre un conjunto de datos resultado de la predicción.
4. Diagnóstico: bloque encargado de, a partir de los datos de predicción y la evaluación del bloque de detección de anomalías, determinar qué modos de fallo pueden producirse, cuándo y su probabilidad.

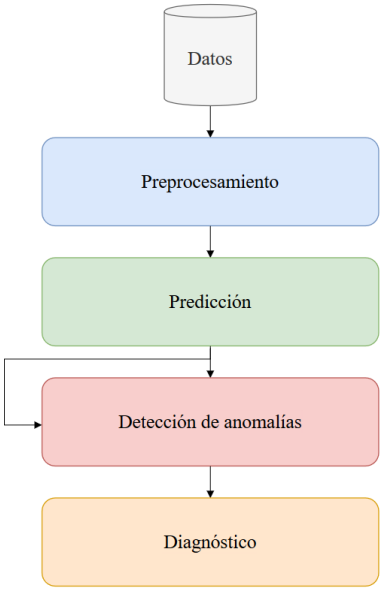

Figura 2: Tareas principales del sistema.

A continuación se van a describir en detalle cada una de las tareas que componen la solución y la arquitectura escogida con un nivel mayor de detalle (Figura 3).

IV-A. *Módulo de preprocesamiento*

Para poder entrenar los modelos de ML es necesario disponer de un conjunto de datos robusto y representativo de los diferentes estados del motor. Los valores medidos por los sensores del sistema se encuentran almacenados en ficheros CSV en HDFS, sin embargo, como se ha detallado en la sección III-A, estos datos precisan ser preprocesados. Es por ello que durante esta fase el sistema comprueba la existencia de valores nulos en los datos, unifica la frecuencia de muestreo de las variables y lleva a cabo un proceso de selección de variables.

IV-A1. Cargar datos: El flujo operacional comienza mediante la lectura del conjunto de datos que se desea procesar. Estos se encuentran almacenados en un sistema HDFS, al cual se accede mediante una dirección IP y una ruta. Los datos son leídos del CSV como un dataframe, estructura de datos tabular compuesta por filas y columnas etiquetadas.

IV-A2. Procesar datos: Para solventar el problema de la baja calidad de los datos originales (sección III-A) se llevan a cabo dos acciones:

1. Selección de variables: de todas las variables disponibles en el motor del buque se emplearán aquellas que presentan cierta variabilidad en sus valores. Esta selección de variables se lleva a cabo de dos formas, mediante un procedimiento automático y uno manual.
   - Automático: de manera totalmente automática, el sistema desecha aquellas variables que presentan

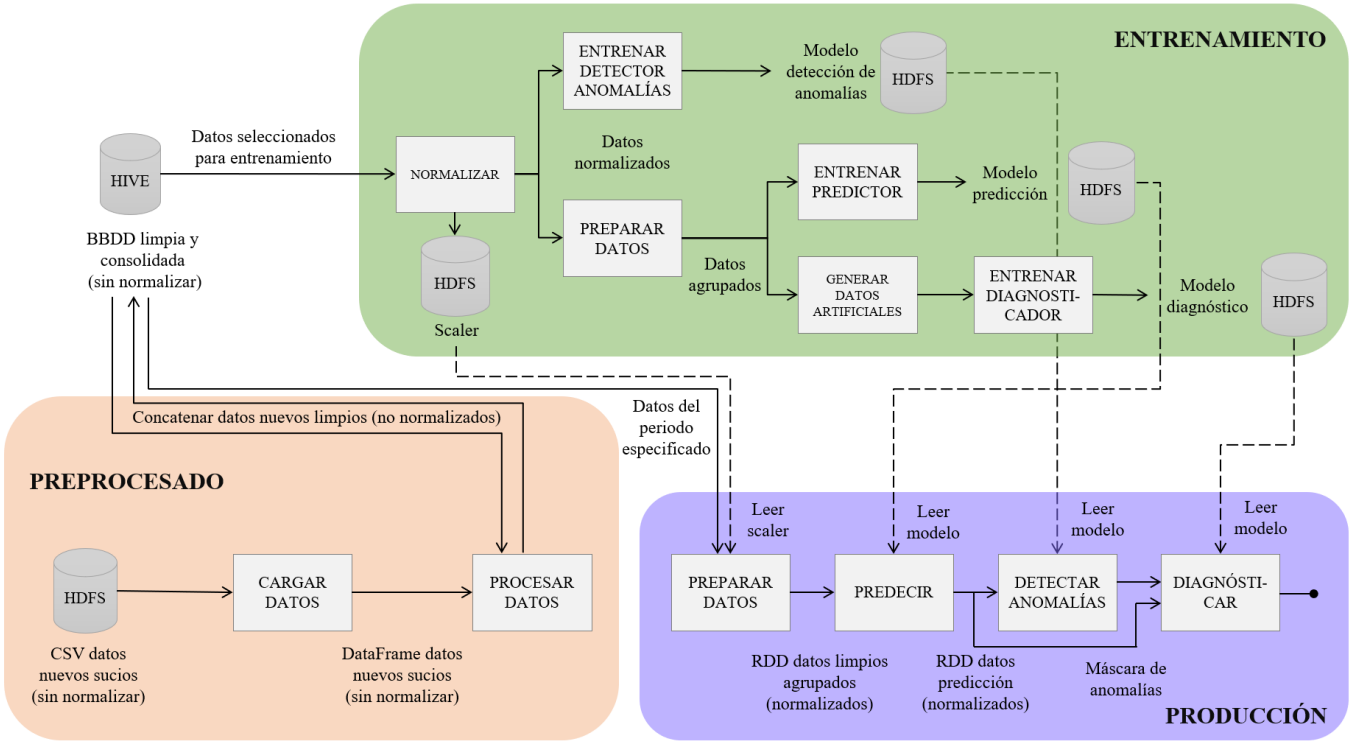

Figura 3: Arquitectura de la solución.

una desviación típica igual a cero. De esta forma, se van a eliminar aquellos sensores que presentan valores constantes y que por tanto no aportan información al sistema.

- Manual: debido a que puede que alguna variable no sea constante, pero se desee eliminarla por otras razones (se conoce que los valores que registra dicho sensor son erróneos, no aporta información en ese momento para resolver el problema, etc.), el sistema elimina aquellas variables seleccionadas por el usuario.

2. Estandarización de las frecuencias: para evitar el segundo problema es necesario homogeneizar, de alguna forma, la frecuencia de muestreo de las variables. Tras estudiar las distintas frecuencias presentes en las variables, se decidió crear a partir de los datos de partida, un nuevo conjunto de datos en el que exista una medida simultánea para todas las variables cada 60 segundos. Para ello se rellenarán los huecos de las variables que presenten una frecuencia de muestreo menor con el último valor disponible.

Cabe destacar que los datos una vez procesados son almacenados en una base de datos consolidada de Hive [29]. De este modo, no es necesario repetir el preprocesado de datos de forma innecesaria, agilizando las tareas de entrenamiento y producción.

IV-A3. Normalizar datos: Debido a que los valores recogidos por los sensores oscilan en rangos muy diferentes entre sí, para poder llevar a cabo correctamente los posteriores procesos de detección, predicción y clasificación mediante técnicas de aprendizaje automático se van a normalizar los datos. Esta normalización se va a llevar a cabo mediante un proceso de estandarización [30], técnica habitual en ciencia de datos que consiste en transformar los datos generando una distribución que presente media cero y desviación estándar uno. Esta estandarización se llevará a cabo para cada variable de forma individual. El normalizador entrenado (scaler) se almacenará en HDFS para ser utilizado en producción ante la llegada de nuevos datos.

IV-A4. Preparar datos: Los datos procesados y normalizados son un buen punto de partida para llevar a cabo el entrenamiento del predictor. Sin embargo, ya que el proceso de predicción se va a realizar con horizontes de horas, días, semanas e incluso meses, hemos decidido que entrenar el sistema de predicción con datos muestreados a una frecuencia de 1 dato/minuto puede dificultar el entrenamiento y la consecuente capacidad de abstracción del sistema. La solución desarrollada ha sido que el usuario pueda establecer desde un comienzo la unidad escogida para llevar a cabo la predicción (horas, días, semanas o meses), de forma que los datos normalizados se agrupen en función de dicha unidad. Por ejemplo, si vamos a realizar predicciones en la unidad días, los datos normalizados se van a agrupar (remuestrear) produciendo un único dato por

día.

Debido a que la utilización de los buques es irregular en el tiempo, es deseable llevar a cabo la predicción del estado del motor en unidades de tiempo correspondientes a horas, días, semanas o meses de uso del motor. Los datos que aportan información para modelar el funcionamiento del motor son aquellos que se corresponden con momentos en los que el motor no se encuentra apagado. Debido a esto, se ha decidido que antes de agrupar los datos, estos se filtren en función del valor de la variable RPM. Las filas del dataframe cuyo valor de RPM no supere el umbral μ serán desechadas.

IV-B. Módulo de predicción

El objetivo principal es conocer (predecir) el estado del motor en un instante de tiempo futuro. De esta manera, para llevar a cabo una predicción desde un instante de tiempo t_i, para un instante futuro que dista a $horizonte$ unidades de tiempo de uso del motor ($t_{i+horizonte}$), el sistema deberá recibir la información recogida por los sensores durante las últimas $ventana$ unidades de tiempo anteriores a t_i ($t_{i-ventana}$). En la Figura 4 se muestra una representación gráfica de estos conceptos. La unidad de tiempo se corresponderá con la escogida por el usuario durante la preparación de los datos (horas, días, semanas o meses). El usuario puede por tanto escoger el tamaño de la ventana y horizonte de predicción.

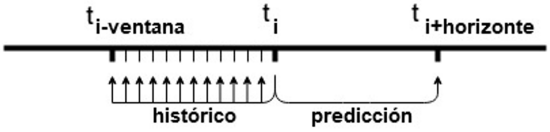

Figura 4: Utilización de una ventana de datos previos en el histórico para llevar a cabo una predicción.

El usuario podrá elegir también la técnica de predicción a emplear. Los métodos que soportan entrenamiento y ejecución distribuida son los implementados por la librería Mllib [31] de Spark. Debido a que son métodos con capacidad de predicción lineal, se da la posibilidad de emplear redes LSTM (II-B) aunque su implementación no sea distribuida. Además, ya que no se encuentra disponible en Mllib, se ha incluido la versión no distribuida de Elastic Net. La Tabla I resume los métodos mencionados.

Tabla I: Métodos de predicción disponibles en el sistema.

	Ejecución	Librería
Regresión Lineal	Distribuida	Mllib
L1 Lasso	Distribuida	Mllib
L2 Ridge	Distribuida	Mllib
Elastic Net	No distribuida	Scikit-Learn
LSTM	No distribuida	Keras

Cabe destacar que tras el entrenamiento de los modelos, estos son almacenados en el sistema de archivos HDFS para ser usados en producción del mismo modo que los normalizadores (scalers). Este comportamiento se puede observar en la Figura 3.

IV-C. Módulo de detección de anomalías

Una vez realizada la predicción del estado del motor, es necesario determinar si dicho estado se corresponde con un valor normal o un posible fallo (anomalía). Ya que no se dispone de anomalías etiquetadas, el proceso de detección de anomalías se lleva a cabo de forma no supervisada mediante el método Autoencoder (II-B). La implementación distribuida de este método se encuentra disponible en la librería Sparkling Water [32].

Mediante el error de reconstrucción emitido por el modelo Autoencoder podemos discernir entre datos normales (bajo error de reconstrucción) y datos anómalos (errores altos). Este valor puede presentarse como el error cuadrático medio de todas las variables de entrada (un único valor), o su descomposición, es decir, el error de cada una de las variables o nodos de entrada de la red.

El objetivo, dado un conjunto de registros del motor (procedentes o no de la predicción), es determinar cuáles son anómalos y qué variables provocan dichas anomalías. Hemos decidido dividir el proceso en tres subfases secuenciales:

1. Detectar anomalías: para determinar qué datos son normales o anómalos se realiza un primer filtrado mediante el error cuadrático medio. A partir de un error umbral precalculado, se clasifican los datos que superen dicho error como anómalos y el resto como normales. El usuario puede escoger si el cálculo de este error umbral se lleva a cabo mediante la técnica del rango intercuartílico [33] o estableciendo un porcentaje de datos anómalos en el conjunto de datos.
2. Independizar contribuciones: para determinar qué variables concretas han sido las causantes de la aparición de la anomalía en los datos clasificados en la fase anterior como anómalos, se emplea la descomposición del error de reconstrucción. De esta forma, pasamos de un único error global a tantos como variables conformen el conjunto de datos. Teniendo en cuenta que los datos se encuentran normalizados, esto nos permite ordenar las variables por su error de reconstrucción. Para determinar qué variables han contribuido más a la formación de la anomalía, el sistema utiliza un método que selecciona de manera automática aquellas variables más anómalas. El método recibe el nombre de Elbow Method [34], y permite a partir de un conjunto de variables y sus errores de reconstrucción, seleccionar automáticamente aquellas que más se desvían.
3. Construir máscara de anomalías: a partir de la selección de la subfase anterior, se construye una matriz o máscara de salida de dimensiones mxn (siendo m el número de filas o registros y n el número de columnas o variables) en la que las variables anómalas se marcan con un uno y las normales con un cero. Esta información será empleada por el posterior módulo de diagnóstico.

IV-D. Módulo de diagnóstico

El bloque de diagnóstico es el encargado de, a partir de los datos de predicción y la evaluación del bloque de detección de anomalías, determinar qué modos de fallo pueden producirse, en qué momento y su probabilidad. Para determinar la probabilidad de cada modo de fallo se ha decidido utilizar un Perceptrón Multicapa [35], modelo de clasificación supervisado basado en redes neuronales.

Debido a que no se dispone de conjuntos de datos etiquetados correspondientes a todos los posibles modos de fallo para entrenar el perceptrón, se ha decidido implementar un generador de datos artificiales que los produzca. Estudiando las características teóricas de los modos de fallo recogidas en el documento FMECA (variables implicadas, valores nominales, valores topes de rango, etc.), el generador permite construir conjuntos de datos normales y anómalos.

El modelo es entrenado considerando a cada uno de los modos de fallo disponibles en el FMECA como una posible clase de salida, de manera que ante la llegada de nuevos datos emita la probabilidad de ocurrencia de cada modo de fallo. De este modo, la arquitectura de la red tendrá en su capa de entrada tantos nodos como variables o sensores del motor, y en su capa de salida tantos nodos como modos de fallo. Esta capa de salida emplea una función softmax para emitir una probabilidad en el intervalo [0, 1].

El clasificador de modos de fallo se emplea en combinación con la máscara de salida del bloque de detección de anomalías para para llevar a cabo el diagnóstico del motor. La utilidad de la máscara de anomalías es acotar el número de modos de fallo posibles. Dado un registro de datos de un instante de tiempo, cuya máscara de anomalías contiene m variables marcadas como anómalas y n marcadas como normales, únicamente se van a considerar como posibles los modos de fallo en los que interviene alguna de las m variables marcadas en la máscara. De esta forma, los modos de fallo que todas sus variables implicadas han sido consideradas normales por el detector de anomalías serán omitidos.

IV-D1. Generador de modos de fallo artificiales: El documento FMECA recoge los umbrales nominales y topes de rango de las variables que conforman cada modo de fallo. Analizando de forma automática este documento, el sistema es capaz de obtener dicha información. El valor nominal de una variable o sensor es el valor habitual sobre el que oscila durante el funcionamiento correcto del motor. Los valores topes de rango, sin embargo, son aquellos umbrales mínimos y máximos a partir de los cuales un sensor se encuentra en un estado anormal, pudiendo dar lugar a fallos en el motor.

En base a estos valores, el sistema es capaz de generar datos asociados a un modo de fallo. Las variables que no intervienen en el modo de fallo presentarán valores que oscilan sobre sus correspondientes valores nominales, mientras que los valores de las variables que intervienen el modo de fallo serán desviadas más allá de los valores tope de rango. Para no generar conjuntos de datos demasiado homogéneos, el sistema emplea distribuciones normales y uniformes para introducir algo de variabilidad en los sensores. Por ejemplo, dado un conjunto de datos generado artificialmente, los valores que toma una variable que no interviene en el modo de fallo generado seguirán una distribución normal centrada en el valor nominal sin llegar a alcanzar los valores tope de rango.

id	date	certainty
5	2015-03-01 13:23:14	0.6352
27	2015-02-01 15:23:14	0.6723

Figura 5: Ejemplo de salida del sistema en la que el modo de fallo 5 obtiene una probabilidad de ocurrencia del 63.52 % y el modo de fallo 27 del 67.23 %.

IV-D2. Salida del sistema: La apariencia de la salida del sistema es la mostrada en la Figura 5, correspondiéndose la columa *id* con el identificador del modo de fallo en el FMECA, la columna *date* con la fecha prevista en la que se produzca el modo de fallo y la columna *certainty* con la probabilidad en formato decimal asociada a cada modo.

V. Evaluación

Nos encontramos ante un escenario novedoso sobre el que hemos llevado a cabo procesos de aprendizaje no supervisado en varios momentos. No disponemos de un conjunto de *targets* o valores de referencia, por lo que no es posible realizar una evaluación totalmente cuantitativa del sistema mediante métricas tradicionales. El desarrollo de las distintas soluciones parciales que conforman la solución final ha sido supeditado a una evaluación mayormente cualitativa de los propios desarrolladores y la organización implicada. En esta sección se recogen algunas de estas ideas:

- Predicción: las grandes agrupaciones de datos durante la etapa de preprocesado permiten representar el estado de los buques a alto nivel correctamente. Sin embargo, debido al filtrado de los datos en función de las RPM del motor, se ha visto que una gran parte de los datos disponibles se correspondían con instantes en los que el motor se encontraba apagado. Debido a ello, cuando se realiza un agrupamiento grande (p. ej., 1 dato/semana o 1 dato/mes) resultan muy pocos datos, lo que incapacita el correcto entrenamiento de modelos de regresión que precisan un conjuntos de entrenamiento con un gran número de instancias, como es el caso de las redes LSTM. En estos casos en los que no se dispone de un gran conjunto de entrenamiento, se ha visto que los modelos más sencillos reportan mejores resultados. En la Figura 6 se muestra un ejemplo de la predicción llevada a cabo por un modelo de regresión lineal (rojo) frente a los valores a predecir registrados por un sensor (verde). El sistema de predicción es muy sensible tanto a la agrupación de los datos como a los tamaños de la ventana y el horizonte. Además, se ha comprobado que, en general, existe una fuerte correlacción entre las variables que superan el proceso de selección.
- Detección de anomalías: para esta etapa se ha podido contar con un conjunto de datos anotado por expertos

Figura 6: Predicción mediante un modelo de regresión lineal (rojo) frente al valor real a predecir (verde).

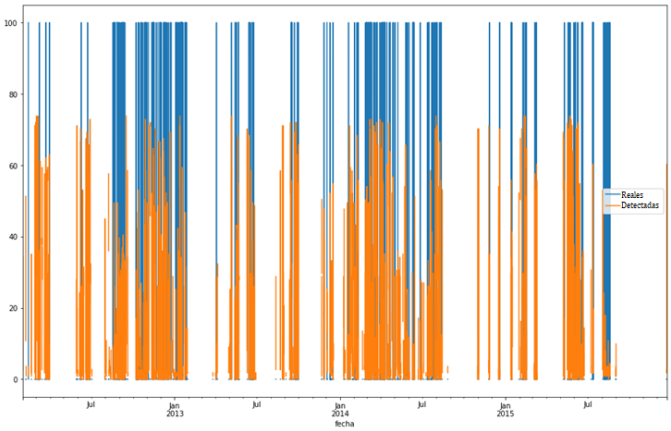

Figura 7: Estados anómalos etiquetados manualmente (azul) frente a los detectados por el método Autoencoder (naranja).

en el que se recogían los estados anómalos del motor durante un intervalo de cuatro años. Se ha probado el modelo Autoencoder entrenado de forma no supervisada y los resultados han sido satisfactorios. El modelo fue capaz de detectar la mayoría de estas anomalías, como se puede ver en la Figura 7. El valor de esta gráfica se encuentra es las coincidencias entre señales a lo largo del eje X, siendo el eje Y irrelevante.

Destacar además que esta fase es muy sensible a la parametrización, pudiendo configurarse para permitir el paso de mayor o menor cantidad de anomalías modificando el umbral de reconstrucción, y dentro de estas, seleccionar un mayor o menor número de variables implicadas mediante Elbow Method.

- Diagnóstico: la utilización de conjuntos de datos artificiales basados en los valores de diseño del motor (FMECA) nos ha permitido construir modelos de clasificación que determinen qué modos de fallo se están produciendo. Sin embargo, este comportamiento teórico del motor no tiene por qué corresponderse siempre con la realidad, ya que su funcionamiento puede variar con el uso, reemplazamiento o arreglo de piezas, etc. El entrenamiento de los modelos de diagnóstico depende directamente de este generador, por lo que es necesario construir un conjunto de datos suficientemente grande y variado.

VI. Conclusión

La solución desarrollada permite predecir la aparición de los diferentes modos de fallo descritos en el FMECA del motor de combustión de un buque. Las tareas de predicción y detección de anomalías son totalmente independientes, por lo que esta última se puede llevar a cabo tanto para momentos futuros (datos provenientes de la predicción), presentes (tiempo real) o pasados (análisis a posteriori).

Trabajando con la versión completa del flujo (Figura 3), la mayor responsabilidad recae en la tarea de predicción ya que las operaciones posteriores parten de la salida de este módulo. Para dotarlo de flexibilidad se proporciona un amplio abanico de métodos de predicción, fácilmente extendible si es preciso, a la vez que se permite al usuario configurar los parámetros de predicción para obtener los resultados más adecuados en cada situación.

El sistema es altamente configurable y su uso se puede extrapolar a otros buques que presenten características similares. Su ejecución es distribuida, por lo que los tiempos de predicción, detección y diagnóstico son bajos, siendo el mayor cuello de botella la tarea de prepocesado, operación que no se lleva a cabo de manera distribuida.

Agradecimientos

Queremos dar nuestro agradecimiento a la Armada Española por permitir la utilización de los datos del proyecto SOPRENE para la elaboración de este trabajo de investigación.

Referencias

[1] R. K. Mobley, *An Introduction to Predictive Maintenance*, 2nd ed., Elsevier, Ed. Elsevier, 1990.

[2] O. Motaghare, A. S. Pillai, and K. I. Ramachandran, "Predictive maintenance architecture," in *2018 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, Dec 2018, pp. 1–4.

[3] Y. Ran, X. Zhou, P. Lin, Y. Wen, and R. Deng, "A survey of predictive maintenance: Systems, purposes and approaches," 2019.

[4] Y. He, X. Han, C. Gu, and Z. Chen, "Cost-oriented predictive maintenance based on mission reliability state for cyber manufacturing systems," *Advances in Mechanical Engineering*, vol. 10, no. 1, p. 1687814017751467, 2018. [Online]. Available: https://doi.org/10.1177/1687814017751467

[5] S. Song, D. W. Coit, and Q. Feng, "Reliability analysis of multiple-component series systems subject to hard and soft failures with dependent shock effects," *IIE Transactions*, vol. 48, no. 8, pp. 720–735, 2016. [Online]. Available: https://doi.org/10.1080/0740817X.2016.1140922

[6] M. A. Gravette and K. Barker, "Achieved availability importance measure for enhancing reliability-centered maintenance decisions," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 229, no. 1, pp. 62–72, 2015. [Online]. Available: https://doi.org/10.1177/1748006X14550849

[7] L. Lin, B. Luo, and S. Zhong, "Multi-objective decision-making model based on cbm for an aircraft fleet with reliability constraint," *International Journal of Production Research*, vol. 56, no. 14, pp. 4831–4848, 2018. [Online]. Available: https://doi.org/10.1080/00207543.2018.1467574

[8] J. Zhao and L. Yang, "A bi-objective model for vessel emergency maintenance under a condition-based maintenance strategy," *Simulation*, vol. 94, no. 7, p. 609–624, 2018.

[9] Y. Xiang, D. Coit, and Z. Zhu, "A multi-objective joint burn-in and imperfect condition-based maintenance model for degradation-based heterogeneous populations," *Quality and Reliability Engineering International*, vol. 32, no. 8, pp. 2739–2750, Dec. 2016.

[10] A. Konys, "An ontology-based knowledge modelling for a sustainability assessment domain," *Sustainability*, no. 10, p. 300, 2018.

[11] M. D. Ying Peng and M. J. Zuo, "Current status of machine prognostics in condition-based maintenance: a review," *The International Journal of Advanced Manufacturing Technology*, vol. 50, no. 1, pp. 297–313, Sep. 2010.

[12] N. Chen, Z.-S. Ye, Y. Xiang, and L. Zhang, "Condition-based maintenance using the inverse gaussian degradation model assessment domain," *European Journal of Operational Research*, vol. 243, no. 1, p. 190–199, 2015.

[13] A. LUCIFREDI, C. MAZZIERI, and M. ROSSI, "Application of multiregressive linear models, dynamic kriging models and neural network models to predictive maintenance of hydroelectric power systems," *Mechanical Systems and Signal Processing*, vol. 14, no. 3, pp. 471 – 494, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0888327099912578

[14] M. Calder and M. Sevegnani, "Stochastic model checking for predicting component failures and service availability," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 1, pp. 174–187, 2019.

[15] B. Samanta and K. Al-Balushi, "Artificial neural network based fault diagnostics of rolling element bearings using time-domain features," *Mechanical Systems and Signal Processing*, vol. 17, pp. 317–328, 03 2003.

[16] J. R. Quinlan, "Improved use of continuous attributes in C4.5," *CoRR*, vol. cs.AI/9603103, 1996. [Online]. Available: https://arxiv.org/abs/cs/9603103

[17] J. Shi, N. Niu, and X. Zhu, "A fault diagnosis method for multi-condition system based on random forest," *2019 Prognostics and System Health Management Conference (PHM-Paris)*, pp. 350–355, 2019.

[18] Y. Chen, H. Yigang, Z. Li, L. Chen, and C. Zhang, "Remaining useful life prediction and state of health diagnosis of lithium-ion battery based on second-order central difference particle filter," *IEEE Access*, vol. PP, pp. 1–1, 02 2020.

[19] G. Ratsch, S. Mika, B. Scholkopf, and K. . Muller, "Constructing boosting algorithms from svms: an application to one-class classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1184–1199, 2002.

[20] G. A. Susto, A. Schirru, S. Pampuri, S. McLoone, and A. Beghi, "Machine learning for predictive maintenance: A multiple classifier approach," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 812–820, 2015.

[21] Z. Liu, W. Mei, X. Zeng, C. Yang, and X. Zhou, "Remaining useful life estimation of insulated gate biploar transistors (igbts) based on a novel volterra k-nearest neighbor optimally pruned extreme learning machine (vkopp) model using degradation data," *Sensors (Basel, Switzerland)*, 2017.

[22] X. long Chen, P. hong Wang, Y. sheng Hao, and M. Zhao, "Evidential knn-based condition monitoring and early warning method with applications in power plant," *Neurocomputing*, vol. 315, pp. 18 – 32, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231218305496

[23] M. Ma, C. Sun, and X. Chen, "Deep coupling autoencoder for fault diagnosis with multimodal sensory data," *IEEE Transactions on Industrial Informatics*, vol. 14, pp. 1137–1145, 2018.

[24] J. Yuan and Y. Tian, "An intelligent fault diagnosis method using gru neural network towards sequential data in dynamic processes," *Processes*, vol. 7, p. 152, 2019.

[25] R. Yang, M. Huang, Q. Lu, and M. Zhong, "Rotating machinery fault diagnosis using long-short-term memory recurrent neural network," *IFAC-PapersOnLine*, vol. 51, no. 24, pp. 228 – 232, 2018, 10th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896318322912

[26] D. Cutting and M. Cafarella. (2016) Apache hadoop hdfs. https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html, Revised at 11/08/2019.

[27] A. S. F. AMPLab. (2014) Apache spark. https://spark.apache.org, Revised at 11/08/2019.

[28] G. van Rossum. (2014) Python. https://www.python.org, Revised at 11/08/2019.

[29] A. Hive. (2011) Apache software foundation. https://hive.apache.org/, Revised at 11/08/2019.

[30] Standardization, or mean removal and variance scaling. https://scikit-learn.org/stable/modules/preprocessing.html#standardization-or-mean-removal-and-variance-scaling, Revised at 11/08/2019.

[31] M. A. Spark. (2014) Apache spark. https://spark.apache.org/mllib/, Revised at 11/08/2019.

[32] H2O.ai. (2015) H2o sparkling water. http://docs.h2o.ai/sparkling-water/2.2/latest-stable/doc/about.html, Revised at 11/08/2019.

[33] B. B. Frey, Ed., *The SAGE Encyclopedia of Educational Research, Measurement, and Evaluation*. Thousand Oaks, California: AGE Publications, Inc, 2018.

[34] R. L. Thorndike. (1953) Elbow method. https://en.wikipedia.org/wiki/Elbow_method_(clustering), Revised at 11/08/2019.

[35] P. Marius, V. Balas, L. Perescu-Popescu, and N. Mastorakis, "Multilayer perceptron and neural networks," *WSEAS Transactions on Circuits and Systems*, vol. 8, 07 2009.