TecnoCampus
Escola Superior
Politècnica

Centre adscrit a la
upf. Universitat Pompeu Fabra Barcelona

Grado en Ingeniería Informática de Gestión y Sistemas de Información

Abnormal Behavior Identification through Deep Learning: TensorFlow

Robert Torrell Belzach
TUTOR: Xevi Font

Curso 2021 - 2022

TecnoCampus
Mataró-Maresme

Abstract

With Predictive Maintenance, industrial factories can increase the reliability of their production. We can predict when, where and how a system will have failures on the chain of production to these factories.

The objective of this project is to implement a set of predictive models using Statistical and AI-based algorithms. We'll then put them to the test to measure what algorithms are best suited for solving this problem.

Resum

Aplicar efectivament a l´industria de la manufacturació el manteniment predictiu, es a dir, poder esbrinar quan, on i com tindrem fallades en un sistema de la cadena de producció pot resultar molt beneficiós.

Per a aplicar el manteniment predictiu, es desenvoluparan models d'intel·ligència artificial. Es posaran a prova aquests diversos models estadístics, d'aprenentatge automàtic i d'aprenentatge profund per a comprendre quines tècniques ens ofereixen els millors resultats davant aquest problema.

Resumen

Aplicar efectivamente en la industria de la manufacturación el mantenimiento predictivo, es decir, poder predecir cuándo, dónde y cómo fallará un sistema de la cadena de producción puede resultar muy beneficioso.

Para aplicar el mantenimiento predictivo, se desarrollaran modelos de Inteligencia Artificial. Se pondrán a prueba diversos modelos estadísticos, de aprendizaje automático, y de aprendizaje profundo para comprender qué técnicas nos ofrecen los mejores resultados ante este problema.

Índex

1. Introducción

El mundo en el que vivimos depende completamente de la tecnología y la industria. Una fabricación fiable, productiva y eficiente beneficia a toda la sociedad. La última gran revolución, “Industria 4.0”, que combina varias tecnologías de la Información para mejorar los procesos productivos, ha resultado ser muy exitosa.

Para asegurar la calidad de los productos y mantener altos niveles de eficiencia, el mantenimiento de la infraestructura es clave. Tener operadores cualificados, siempre preparados ante cualquier imprevisto es básico para cualquier fábrica, pero tarde o temprano siempre aparecerán problemas técnicos.

Estos problemas paran la producción en serie al completo, causando pérdidas en coste de oportunidad, y poniendo en riesgo la viabilidad de la producción.

Por tanto, sería ideal que se pudiera obtener el estado de la salud de toda la maquinaria, predecir los problemas técnicos, y solventarlos antes de que causen problemas mayores. A esta estrategia de mantenimiento se la denomina “Mantenimiento predictivo”.

Los avances en Inteligencia artificial han permitido su aplicación en una infinidad de problemas diferentes, muchos en contextos totalmente diferentes (Conducción autónoma, marketing digital, traducción e interpretación, Diagnósticos médicos, etc.) y se ha demostrado innumerables veces su utilidad.
La capacidad que poseen las inteligencias artificiales para inferir eventos antes de que estos ocurran resulta interesante para la resolución de muchos problemas reales.

Este proyecto tratará de desarrollar varios modelos capaces de analizar el estado de un motor eléctrico. Se crearan a través de un análisis exhaustivo y una metodología de trabajo científica.
Un modelo ideal seria capaz de detectar un estado anómalo e informar a los técnicos de mantenimiento de la predicción de un fallo en un sistema, idealmente con un porcentaje de fiabilidad de los resultados, o una descripción del fallo que el sistema espera que ocurra.

De conseguir crear un sistema que cumpla con estos requisitos, sería beneficioso aplicarlo en cadenas de producción reales para abaratar costes y aumentar la productividad de las cadenas de producción.

Para llevar a cabo este proyecto, se desarrollará estos modelos predictivos en base de un conjunto de datos reales. Estos datos se liberaron a Internet y el acceso a estos es gratuito y libre.
El conjunto de datos con los que se trabaja se titula "NASA Bearing Dataset" y se encuentra disponible en el siguiente enlace (https://www.kaggle.com/vinayak123tyagi/bearing-dataset).

Otros conjuntos de datos que se podrían usar para desarrollar la investigacion son "Machine Predictive Maintenance Classification" (https://www.kaggle.com/shivamb/machine-predictive-maintenance-classification) y "Microsoft Azure Predictive Maintenance" (https://www.kaggle.com/arnabbiswas1/microsoft-azure-predictive-maintenance?select=PdM_telemetry.csv).

2. Marco teórico : Contexto, antecedentes y necesidades de información

2.1. Mantenimiento en industrias

La industria está en un permanente estado de innovación tecnológica. Una constante mejora en metodologías, tecnologías y talento humano mantienen un progreso positivo en muchos sectores de nuestra sociedad. Los beneficios económicos, la eficiencia de la producción, la seguridad y bienestar de los empleados y la calidad de los productos finales son solo algunos ejemplos.

En su permanente interés de mejorar, las factorías siempre han estado muy abiertas a analizarse y mejorarse para avanzar y mantenerse más vanguardistas y competitivos ante las demás.

Uno de los problemas que cualquier fábrica siempre tendrá que intentar resolver es el de gestionar el mantenimiento y los problemas técnicos que pueden surgir.

Claramente, todas las herramientas y maquinaria que se instala y usa en las naves industriales siempre sufren desgaste en sus horas activas, y es solo cuestión de tiempo que comiencen a presentar rendimiento reducido, problemas y daños.

Al paso de los años, los técnicos, ingenieros y expertos han desarrollado varias estrategias y metodologías para evitar sufrir los problemas que surgen por el desgaste mecanico. Aún existe mucha discusión sobre la taxonomía y clasificación de estas metodologías [1], pero para los intereses de este proyecto, presentaré la clasificación de tipos de mantenimiento por el Departamento de energía del gobierno de los Estados Unidos [2]: Reactivo, Preventivo, Predictivo y RCM (reliability-centered maintenance, Mantenimiento basado en la fiabilidad).

El Mantenimiento Reactivo es dejar a toda la maquinaria hacer su trabajo hasta que se rompa. No se toma ninguna acción ni esfuerzo considerable en mantener el estado de la maquinaria en buenas condiciones, incluso manteniendo una vida útil menor que los planteados por los diseñadores de la maquinaria.

Existen ciertas ventajas de aplicar esta estrategia. Esta requiere una cantidad de inversión inicial mucho menor que en las otras técnicas, y necesita menos técnicos empleados en la planta de fabricación. Todo y así, este beneficio inicial es efímero. La maquinaria sufre un desgaste mucho mayor que con cualquier otra técnica de mantenimiento, y se deberá invertir mucho más esfuerzo y dinero cuando se presenten problemas y fallos.

Si una máquina acaba dañada y se ha de reemplazar, ya se debe gastar una inversión económica mucho mayor. Cuando se tiene que desinstalar e instalar una nueva máquina en la cadena de producción, esta tarea suele ser muy compleja de tratar . Es muy común que se requiera de conocimientos técnicos avanzados para completar la tarea. El tiempo en que la producción está parada presenta un coste de oportunidad muy elevado, así como una mayor presión y carga de trabajo a los empleados. Por tanto el malgasto no solo es económico, pero también humano.

Si tienes que mantener a un equipo de técnicos que han de estar preparados solo para cuando aparezcan estos daños, se está organizando muy ineficientemente a estos empleados. Muchos días no han de hacer ninguna tarea, y algunos días han de cumplir con una ardua tarea a contrarreloj.

Más del 55% de plantas industriales estadounidenses utilizan esta estrategia de mantenimiento. [2]

El Mantenimiento Preventivo es el tratamiento de la maquinaria, basada en tareas periódicas que ayudan a detectar, evitar o mitigar los efectos del desgaste en el componente o sistema, con el objetivo de mantener o incluso mejorar la vida útil de este.

En su forma más simple, sólo con seguir los consejos y mejores prácticas de uso y mantenimiento de la maquinaria descritas por los diseñadores de esta, ya se estaría consiguiendo una eficiencia mucho mayor que en la metodología descrita anteriormente. Estudios indican que los ahorros en inversión económica al aplicar esta técnica pueden mejorar del 12% al 18%. [2]

Tareas simples como lubricación, cambios de filtros, sustitución de piezas desgastadas, etc. pueden marcar una gran diferencia.

Las desventajas que se presentan son que los fallos catastróficos siguen apareciendo, y un componente bien mantenido puede perfectamente en horas activas fallar y tener que parar la cadena de producción al completo. Los empleados han de estar constantemente trabajando con las tareas periódicas. No seguir a la perfección la periodicidad de las tareas, pueden resultar en una peor eficiencia del mantenimiento.

El Mantenimiento Predictivo es el conjunto de tareas que miden el desgaste de los componentes de un sistema, y así conseguir analizar y obtener datos empiricos del estado de la fábrica. Estos resultados pueden mostrar las causas de probables futuros fallos y así conseguir reparar el sistema antes de que presente un problema mayor.

La mayor diferencia que presenta esta metodología respecto a la anterior, es que el mantenimiento predictivo se basa en realizar estas tareas sólo cuando es necesario, siendo así mucho más eficiente a la hora de delegar las tareas a el equipo.

Si se realiza correctamente, se puede llegar a evitar casi todos los fallos catastróficos que puedan presentarse en la cadena de producción, se organiza al equipo de mantenimiento mucho más eficientemente, se minimiza el uso de inventario y materiales necesarios, y se consigue mantener tan buena eficiencia, que supone una eficiencia energética mayor.

Se especula que supone un ahorro de entre 30% y 40% de costes de oportunidad, aunque la inversión inicial para aplicar esta metodología es mucho más superior a las otras, requiriendo herramientas que cada una puede costar más de 43,000 euros. [2]

El Mantenimiento Basado en la Fiabilidad (RCM) es similar al anterior, en que se busca actuar donde hace falta, pero intenta ser más consciente de que tan crítico resulta una parte del sistema para la producción, que recursos dispone la organización, que estrategia es más idónea para mantener cada componente del sistema, etc. Intenta tomar decisiones adaptadas a las condiciones de la realidad, manteniendo un orden de prioridad de todos los posibles riesgos que se pueden presentar.

Como claramente se puede observar, hay muchas estrategias posibles a adoptar. RCM acepta que es posible realizar un conjunto de ellas para diferentes maquinarias, y aunque unas parecen más atractivas de adoptar que otras, también la complejidad para conseguir aplicarlas aumenta considerablemente, especialmente en las dos últimas descritas.

Aunque los beneficios de las técnicas de mantenimiento más avanzados son evidentes, resultaría más atractivo para las organizaciones adoptarlas si se consiguiera simplificar el proceso de implementación de estas. Para conseguirlo, se ha de reducir la complejidad de uso de las herramientas, y conseguir que las inversiones iniciales para adoptarlas sea mucho mas asequible.

2.2. Inteligencia Artificial

La Inteligencia Artificial es el campo de estudio para diseñar e implementar sistemas que pueden analizar su "entorno" (conjunto de datos con que opera) y tras analizarlo, pueden tomar decisiones que maximicen las posibilidades de conseguir un estado deseable [3]. Estas decisiones pueden ser en forma de agentes que actúan dentro del mismo entorno (como los automóviles de conducción autónoma [4], o los competidores virtuales de ajedrez [5] u otros juegos [6]) o simplemente que ofrezcan una conclusión, como la clasificación de los datos entre grupos (identificar un carácter escrito a mano en una hoja de papel [7], por ejemplo) o establecer relaciones entre elementos (recomendar una película en relación a las previas que has visto [8], por ejemplo).

Como claramente se puede apreciar, se da uso a esta herramienta casi en todas partes, y no para de crecer el interés general hacia este campo de estudio [9].

Una de las categorías más estudiadas a día de hoy es la del Aprendizaje automático, en que tiene por objetivo diseñar e implementar algoritmos y sistemas que pueden analizar un conjunto de datos previos e inferir unos resultados posteriores o identificar patrones en esos datos, teniendo la capacidad de “aprender” de estos, y seguir mejorando su fiabilidad automáticamente.

Una manera en la que se podría mejorar considerablemente la adopción de estrategias de mantenimiento mas avanzadas, sería automatizar la obtención de los datos y el diagnóstico del estado de la maquinaria, y la Inteligencia Artificial resulta idónea para aplicar tales mejoras.

El Mantenimiento Predictivo es, si se aplica efectivamente, de las mejores estrategias que existen para asegurar la calidad y minimizar el desgaste de la maquinaria. La mayoría de técnicas que existen se basan en el uso de herramientas altamente especializadas y de un coste muy elevado, que muestran datos directos del estado de los componentes de un sistema, con herramientas como sensores infrarrojos, de vibración, ultrasonidos, láser y eléctricos [10].

Estas herramientas nos proporcionarnos de estos datos sobre el estado de la maquinaria, y se pueden procesar por un modelo de IA y extraer información valiosa.

El análisis de estos datos se puede dar por expertos en comprender y analizarlos. Pero para ello, se ha de invertir dinero en contratar al personal especializado, y, si los volúmenes de los datos llegan a ser inmensos, existe un límite humano en el que ya no es posible extraer conclusiones fiables. Los sistemas de Aprendizaje Automático son mucho más aptas para poder realizar esta tarea. [11] No solo se tendría que necesitar a menos personal, pero también este no debería de realizar tareas intelectuales de gran dificultad.

Por tanto, poder aplicar los avances en el campo de la Inteligencia Artificial al mantenimiento industrial resulta de vital importancia, con claros beneficios en fiabilidad, ahorro y facilidad de uso.

2.3. Algoritmos candidatos

Como bien se ha descrito anteriormente, la Inteligencia Artificial nos aporta muchas herramientas para poder aplicar Mantenimiento Predictivo a la industria, y de hecho hay tantas, que puede resultar abrumador.

Nos resultará útil hacer una investigación previa de una lista de algoritmos diferentes que se pueden aplicar, para definir mejor los objetivos y el alcance del proyecto, y documentar qué resultados se espera encontrar.

No existen algoritmos mejores o peores, aunque claramente si de más o menos populares, o de más complejos o sencillos. Para poder cumplir los objetivos del proyecto, no debe acomodar en un solo algoritmo, cuando hay mucho potencial en una gran variedad de otros métodos. Debido a que los algoritmos candidatos son tan diferentes, ellos desempeñarán los cálculos con resultados dispares. Poder contrastarlos nos pueden ayudar a identificar bajo qué condiciones sería mejor aplicar uno o otro en una situación real.

A continuación, voy a exponer una propuesta de diferentes Indicadores clave de los resultados finales generados por el modelo:

- Matriz de confusión
- Coeficiente de determinación (R^2)
- Precision-recall curve
- F-1
- Median absolute error
- Root-mean-square deviation
- Especifilidad

Cualquiera de ellos puede aportarnos una valoración correcta de la calidad del modelo generado, depende del tipo de modelo, sera mas correcto usar uno o otro.

Esta es una propuesta de los Indicadores claves de requerimientos no-funcionales, relacionados con datos observables sobre la ejecución de los algoritmos:

- Tiempo de ejecución
- Nivel máximo uso de CPU/GPU
- Nivel medio uso de CPU/GPU
- Nivel máximo de uso de Memoria
- Nivel medio de uso de Memoria
- Consumo energético
- Gasto económico/Hora
- Tiempo de desarrollo

Y a continuación, estos son los diferentes algoritmos a implementar:

- Mediana movil (Estadistica)
- Regresion lineal (Estadistica)
- Arboles de decision (Aprendizaje supervisado)
- *Extreme Gradient Boost* (Aprendizaje supervisado)
- Naive bayes (Aprendizaje supervisado)
- *Long-Short Term Memory (Deep learning)*
- *Autoencoder (Deep learning)*
- Filtro de Kalman (Stochastic)

Esta lista es extensa, y los algoritmos que describe son muy diferentes los unos de los otros. Poder desarrollar un modelo para el mismo problema con los diferentes algoritmos nos permitirá contrastar y comprender qué clase de algoritmos resuelven mejor el problema.

2.4. Herramientas para el desarrollo

El objetivo del proyecto es mejorar las técnicas de Mantenimiento Predictivo aplicando algoritmos de Inteligencia Artificial. Por lo tanto, necesitamos el uso de herramientas profesionales, estándar y potentes para desarrollar los modelos predictivos.

La gran mayoría de estas herramientas son accesibles al público, y suelen estar protegidas bajo licencias libres.

En el desarrollo de Inteligencia Artificial, dos lenguajes de programación muy populares son Python y R, aunque existen muchos más también aptos para este ámbito de desarrollo [12].

Debido al interés del proyecto por aplicar los resultados en un entorno de ingeniería, y también por la familiaridad de uso por el autor del trabajo, se ha decidido usar Python.

Python es un lenguaje sencillo, robusto y con una comunidad de millones de programadores que le dan uso, contribuyen creando librerías y varios mantienen directamente el intérprete oficial del lenguaje.

Aunque se puede trabajar directamente desde Python sin ningún problema, existe un entorno de desarrollo nombrado “Anaconda” que, de trabajar sobre este entorno, podremos tener el intérprete de Python, varios editores de código configurados como por ejemplo Jupyter Notebook y muchas librerías de uso común en proyectos de cómputo numérico y científico ya instaladas y configuradas.

Python tiene más de trescientos mil librerías [13] libres de usar por cualquier programador. Varias de las librerías que le daremos uso serán:

- Numpy nos permitirá definir y crear matrices multidimensionales de alto rendimiento. Aunque Python ya tiene una implementación de esta estructura de datos, Numpy las vuelve a implementar bajo el lenguaje C, que al ser compilado es mucho más rápido en y ligero en requerimiento de recursos de ejecución.

- Scipy es un extenso conjunto de algoritmos y estructuras de datos implementados para una gran variedad de computaciones. Puede realizar operaciones de Optimización, Integración, Interpolación, resolver ecuaciones algebraicas, cálculos estadísticos y más.

- Matplotlib, Seaborn & Graphviz: Matplotlib es la herramienta de Python mas popular para poder generar gráficos y generar representaciones visuales de figuras sobre los datos que se procesan, generalmente para gráficos sobre planos cartesianos, pero admite muchos mas. Seaborn es una capa de abstracción de uso de Matplotlib, en que facilita mucho la creación de figuras graficas con métodos mas sencillos y configuraciones por defecto. Graphviz es otra herramienta para generar visuales sobre los datos pero especializado en estructuras de datos basadas en grafos.

- Scikit-learn es un conjunto de implementaciones de los algoritmos de Aprendizaje Automático mas comunes y populares, y aunque no ofrece ni algoritmos ni estructuras de datos para construir proyectos de Aprendizaje profundo, si contiene una extensa colección para clasificación, regresión, agrupación, reducción de dimensiones y preproceso de datos. Plantea ser la librería mas sencilla y practica para programar modelos predictivos eficaces y eficientes.

- Tensorflow, una librería desarrollada por Google que implementa varios algoritmos y cálculos de Aprendizaje automático, y permite la creación y uso de Grafos y de algoritmos de Aprendizaje Profundo.

- Keras es una librería que permite el uso de varios constructores que instancia modelos de Aprendizaje profundo bajo el uso de varios algoritmos estándar. Para este proyecto, su uso sera como capa de abstracción del uso de Tensorflow.

- XGBoost es una de las librerías más populares entre los equipos de desarrollo de competiciones de programación [14] y permite la creación y uso de estructuras basadas en el "*Gradient Boosting*".

Esta lista inicial nos permite demostrar que tenemos a nuestra disposición un gran conjunto de herramientas que nos ayudarán a realizar con éxito los objetivos del proyecto en el marco de tiempo que disponemos, y hasta la propia documentación de varias de ellas muestran proyectos de ejemplo que resuelven problemas similares al propuesto por el proyecto [15].

3. Objetivos y alcance

Debido a la naturaleza académica y teórica del proyecto, este proyecto es un trabajo de investigación aplicado a la resolución de un problema práctico, sin llegar a ser meramente un proyecto de ingeniería pura. No solo se valorará el resultado final del modelo que pueda acabar diseñando e implementado, sino también todo el recorrido que se ha tomado, la información y conocimientos que se han ido adquiriendo, las decisiones que se han tomado y, en resumen, todo el proceso por el que se habrá pasado.

La disponibilidad de las herramientas informáticas y de la documentación necesaria, que suele ser académica, pública y abierta, ayudan mucho a la viabilidad del proyecto. La mayor dificultad que supone el proyecto es el alto nivel de conocimiento especializado que se necesita para implementar el sistema, pero no hay apenas necesidades económicas.

El objetivo de este proyecto será:

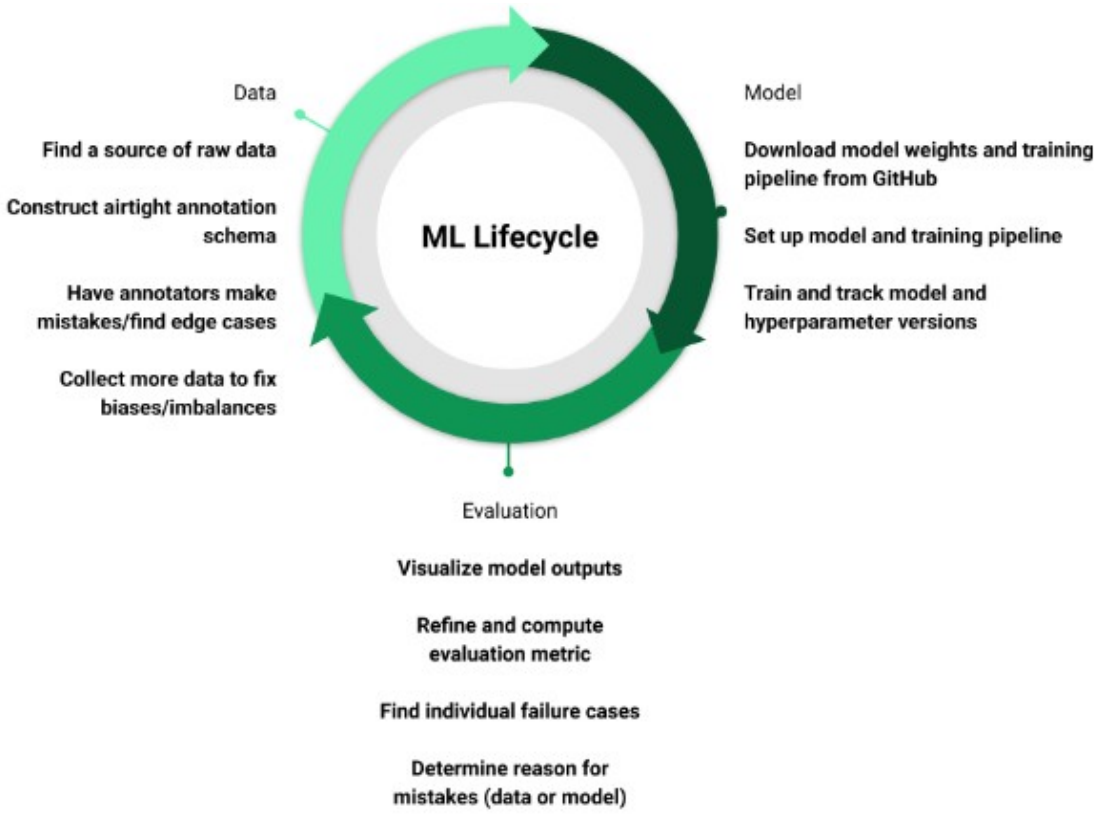
- Crear varios modelos predictivos, a través del uso de una gran variedad de algoritmos, desde los más sencillos basados en la mediana móvil, de más interesantes como técnicas de regresión hasta Redes neuronales LSTM..
- Seguir el método científico para investigar, contrastar y crear los modelos, basándose en datos y observaciones objetivas.
- Comparar el rendimiento de varios modelos y extraer conclusiones cualitativas de estos (Que modelo es más fiable, cuál es más rápido, cual requiere de menos consumo eléctrico, etc.).
- Calcular el beneficio vs. coste de desarrollar un modelo para una producción, y evaluar su efectividad.
- Evaluar los resultados en comparación con el estado de la investigación (state-of-the-art) sobre el mantenimiento predictivo.

4. Metodología

Este proyecto tiene por especial interés la aplicación de conocimientos académicos con una extensa teoría matemática que los apoyen. Además, el estudio que se realizará también tiene un uso directo en la Ingeniería, específicamente en la Ingeniería industrial.

Debido a que se busca buscar conclusiones ante hipótesis iniciales, pero bajo la búsqueda de resolución de un problema real, se sugiere el uso de una metodología inspirada en el Método científico, pero adaptada a las necesidades del proyecto, sobretodo bajo el punto de vista de la realización de un proyecto de Inteligencia Artificial.

No se puede considerar el proyecto como un trabajo meramente científico, ya que se busca generar una solución a un problema específico, y no solo meramente extraer conclusiones y adquirir conocimiento, pero tampoco tiene las características de un proyecto de ingeniería clásico en el que se podrían desarrollar con “Scrum” o “Waterfall”, ya que los modelos de Aprendizaje Automático son influidos directamente, y no diseñados, por tanto cuesta definir tareas planificadas en anterioridad y que tengan una duración de desarrollo corto y un riesgo bajo. El desarrollo del código del proyecto no se puede realizar simplemente como una lista finita de requerimientos funcionales.
Pero como también combina elementos de ambos, se sugiere el uso de una metodología adaptada a esta clase de proyectos: Machine Learning Lifecycle [18].



Esta metodología contiene tres pasos cíclicos, en los que se van iterando hasta la finalización del proyecto.

4.1. Exploración de datos

El primer paso es todo lo referente a la obtención, tratamiento, exploración/análisis y manipulación de los datos sobre los que se trabajara, antes de aplicar ningún algoritmo de Inteligencia Artificial.

Inicialmente se describe qué propiedades han de tener los conjuntos de datos sobre los que queremos operar, de donde los obtendremos, que propiedades y formato tienen, qué optimizaciones previas se les pueden aplicar (para reducir peso, facilitar el trabajo posterior, acelerar la velocidad de cómputo, limpiar los "outliers" y los datos redundantes o inútiles, etc.) y mucho más.

Este paso es muy importante debido a que todo sistema informático depende directamente de los datos que procesa. No comprender cómo usarlos, o la calidad que tienen, pueden repercutir negativamente en los resultados de las soluciones realizadas y dañar la calidad del desarrollo. Además, hacer un estudio previo a tal conjunto nos puede ayudar a tomar decisiones importantes de desarrollo e investigación, ya que dependiendo de las propiedades de estos, se puede argumentar el uso de los algoritmos, técnicas, o herramientas.

Iterativamente, sabiendo los resultados de iteraciones anteriores, y teniendo en cuenta los objetivos de la iteración actual, se pueden aplicar cambios en el conjunto de datos y así conseguir mejores resultados a partir de estas operaciones, o adaptarlas mejor a los nuevos modelos que se plantean generar.

4.2. Desarrollo del modelo

En este paso, una vez ya se ha preparado el conjunto de datos, se comienza a aplicar y diseñar los algoritmos con los que tratarán.

Este paso claramente es el más experimental, en que se ponen a prueba las diferentes hipótesis que han ido surgiendo durante el proyecto. Probar nuevas ideas, combinar tecnologías, aprovechar las herramientas ya existentes y encontrar relaciones y propiedades a los datos con los que operamos.

Este es el apartado en que se desarrolla el código con la mayoría de cálculos numéricos de alta complejidad, donde no solo se manipula la información, sino también se da un uso extenso a algoritmos, fórmulas y teoremas matemáticos.

Cada iteración se centrará en implementar un modelo predictivo en específico, comenzando por iteraciones más cortas, desarrollándose con algoritmos sencillos de estadística básica, como la mediana móvil, avanzando por el uso de los más comunes en el Aprendizaje Automático, como los Árboles de decisión, y en las últimas iteraciones, se desarrollará modelos con los algoritmos más avanzados, como el LSTM.

4.3. Evaluación del modelo

En este paso, se pone a prueba al modelo generado al hacerlo trabajar con un nuevo conjunto de datos, y se evalúa a través de diversos Indicadores clave el rendimiento de este. Una vez se haya conseguido recopilar información del modelo, se puede contrastar con los resultados por modelos generados en iteraciones anteriores, y siempre se busca argumentar la efectividad del modelo y las causas de su rendimiento.

A partir del trabajo realizado en este paso, se puede tomar decisiones importantes en las siguientes iteraciones e identificar problemas, fallos, errores y dificultades que se han encontrado al trabajar en esta iteración. Se sabe que se ha hecho hecho correctamente y que incorrectamente nos puede ayudar a corregir y optimizar el trabajo que se hará en el futuro.

También se pueden encontrar incongruencias, o datos válidos pero sorprendentes, y que nos pueden generar nuevas hipótesis e ideas a poner a prueba en el trabajo. De manera similar a los proyectos de Desarrollo Ágil, donde se busca la flexibilidad a la hora de realizar un proyecto, nosotros podemos aprovechar este formato iterativo de trabajo para adaptar el trabajo posterior a el estado presente del proyecto, y así reducir el alcance de los objetivos por cumplir, o de lo contrario, añadir más tareas e información para ser usadas en el futuro.

5. Desarrollo

5.1. Requerimientos funcionales

Los requerimientos funcionales son:

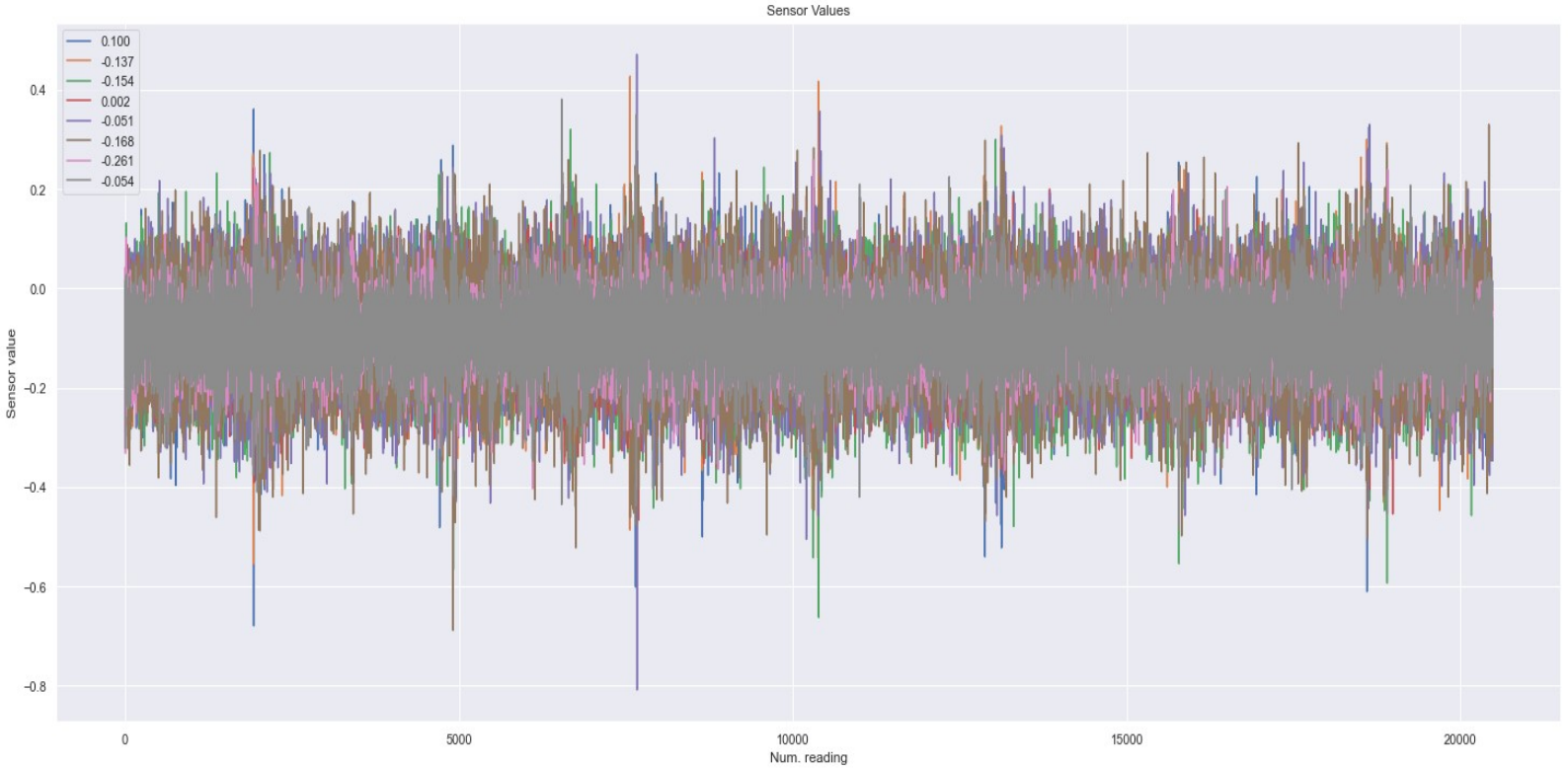
- Realizar una investigación detallada sobre el tema.
- Valorar el estado de la investigación.
- Comparar entre los diferentes productos y servicios actuales.
- Documentar el proceso de desarrollo del proyecto, siguiendo una metodología inspirada en el Método Científico.
- Documentar la base teórica por la que se apoya el trabajo práctico del proyecto.
- Implementar varios modelos predictivos a través del uso de herramientas libres, como Python y Tensorflow.
- Comparar los resultados finales entre ellos, y valorar la efectividad de cada uno de ellos en el contexto del proyecto.
- Analizar los datos y modelos predictivos resultantes para encontrar una relación causa-consecuencia de los posibles eventos registrados por el conjunto de datos.
- Calcular el coste energético y económico que tendría replicar este proyecto en un caso real, así como mantenerlo activo durante el uso en producción.

5.2 Exploración de Datos

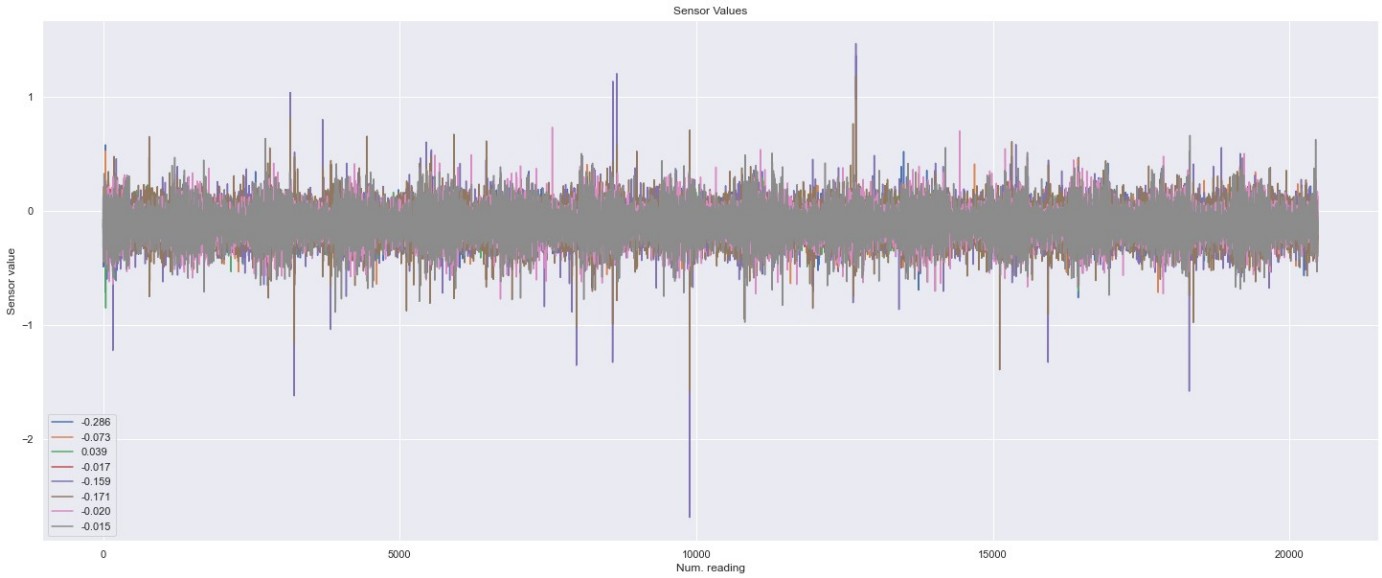
Todo proyecto de calculo estadístico, Aprendizaje automático o de “Big Data” siempre ha de comenzar por un análisis inicial de el estado y las propiedades de los datos con los que se trabajaran.
Se define las propiedades de los valores, la dimensionalidad de las observaciones, se visualizan las distribuciones normales de estos valores y se generan observaciones sobre que transformaciones parecen mas adecuadas para que resulte mas efectivo trabajar con estos datos. Sin esta valoración inicial, no se puede justificar ninguna toma de decisión del proyecto, y adquirir una comprensión efectiva de con que trabajaremos nos ayudara a generar ideas, soluciones y optimizaciones ingeniosas y eficaces.

El conjunto de datos con los que trabajaremos sera el “NASA Bearing Dataset” (https://www.kaggle.com/vinayak123tyagi/bearing-dataset). Este conjunto es una colección de registros de tres experimentos realizados por la NASA. Estos experimentos coleccionaron los valores que registran unos sensores de vibración instalados en diferentes rodamientos internos de un motor mecánico mientras este mantenía una velocidad constante de 2000 RPM.
Los archivos del conjunto de datos tiene tres carpetas, una para cada experimento realizado, y cada carpeta contiene archivos “.csv” con una captura de un segundo con 20.480 registros en total de los cuatro sensores (la velocidad de lectura de los sensores se configuró a 20kHZ ≈ 20.000 lecturas en un segundo). El intervalo de espera entre capturas es de 10 minutos.

Primero, se comprueba que valores tienen las observaciones de cada sensor en la lectura realizada el día 2003/10/22 a las 17:19:38 (que a partir de ahora la nombraremos “Captura temprana”).



y ahora el del día 2003/11/24 a las 06:21:24 (que a partir de ahora la nombraremos “Captura tardía”).

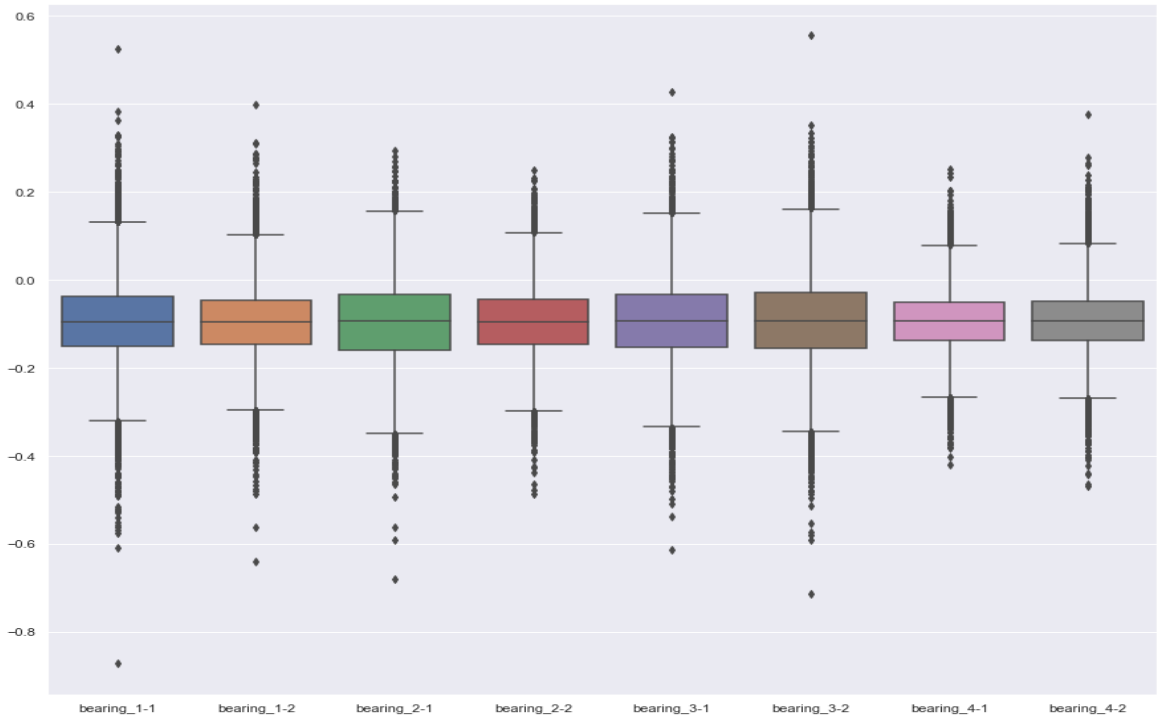


Una primera observación que podemos realizar es que las capturas en los primeros días de la experimentación tienen unas notables diferencias a las capturas realizadas unos pocos días antes de la finalización del experimento. En el inicio, el motor no ha sufrido apenas desgaste, y su rendimiento resulta bastante optimo. El rango en el que las vibraciones oscilan son entre 0.2 y -0.3, con varios picos que sobresalen del 0.4 y el -0.6 y alguno hasta del -0.8 como máximo.

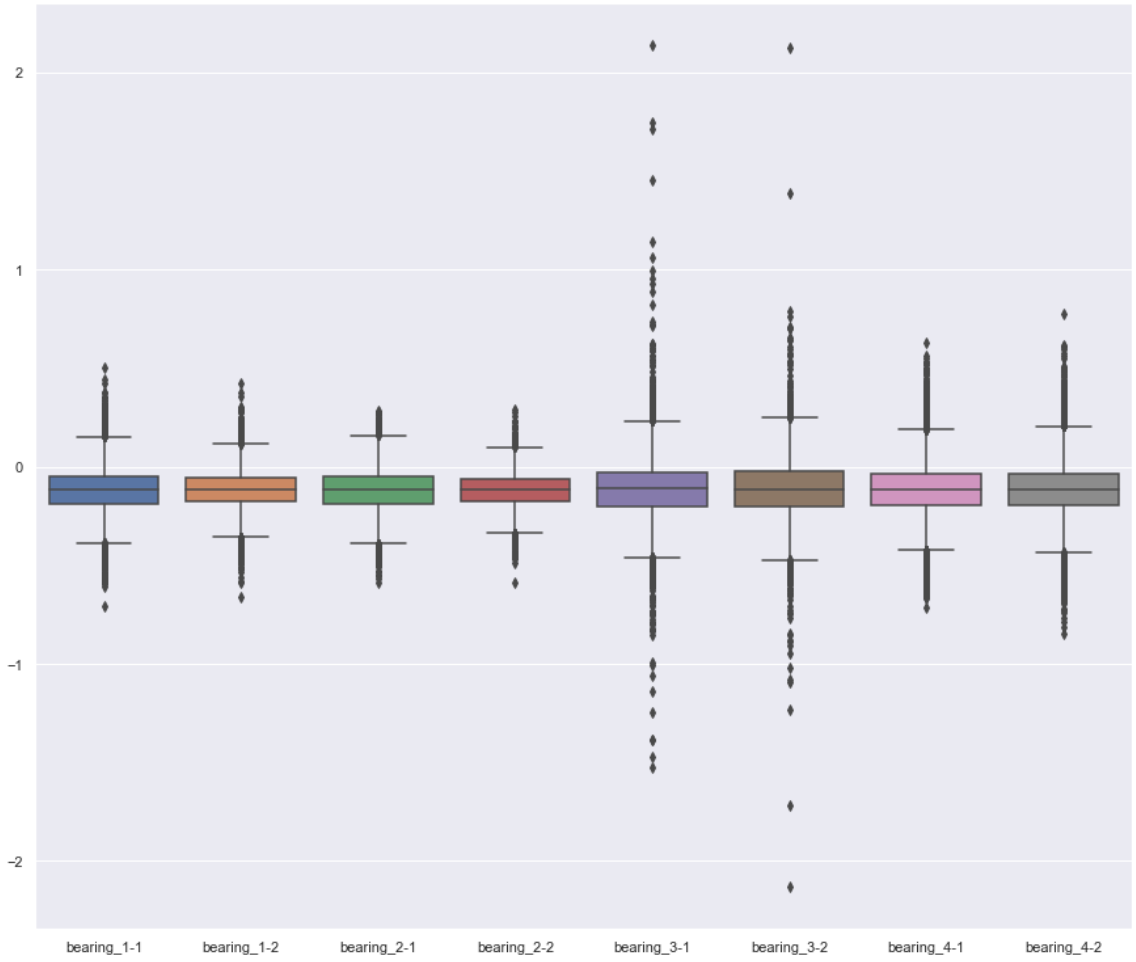
Pero, a poco antes de que el motor sufra una avería, el valor de las observaciones oscilan entre 0.3 y -0.4, con bastantes mas picos mucho mas pronunciados, que llevan a valores de 1, -1 y un máximo de menos de -2.5.

Cabe decir que, una interpretación de los valores de los sensores es que 0 seria ausencia de vibración, y cuando mas alejado este el valor absoluto de la captura, mayor vibración se ha registrado. El signo de la captura simplemente define hacia que dirección se orienta la vibración detectada.

Si generamos un grafico Boxplot de la captura temprana obtendremos lo siguiente:



Y el Boxplot de la captura tardía es:



Estos nos indican de que, inicialmente, la diferencia del rango de las observaciones entre los diferentes sensores es mucho menor que cuando el motor ya ha sufrido desgaste. La diferencia de la distribución de los cuantiles en la captura temprana es muchísimo menor que en la captura tardía, ya que en esta ultima, el sensor 3 mantienen un rango mucho mas amplio que en los demás sensores, ademas de un numero mucho mas mayor de picos de valores dramáticamente dispares.

Por lo tanto, tanto la separación de los cuantiles como el numero de valores atípicos indican cuando un eje del motor tiene un comportamiento mas errático de lo normal.

Entonces, existen dos posibles indicadores del estado de salud del motor: uno seria la evolución del rango de valores capturados en una observación, y el otro la cantidad y valores de los valores atípicos.

Para poder implementar los algoritmos que deseemos, podría resultar mas optimo, en vez de pasar simplemente los valores crudos de las lecturas, generar unas estadísticas sobre estos valores que resuman las observaciones que hemos descrito anteriormente, y que operen sobre aquella información que consideramos clave para clasificar el estado de la salud del motor.

Estos datos adicionales son los siguientes:

- max_vibr: A partir del valor absoluto de las lecturas de los sensores, cual es el mayor valor de vibración.
- diff_minmax: La distancia entre el valor de vibración mayor y menor
- avg_sensor_val: Mediana de la lectura de todos los sensores.
- perc_life: Porcentualmente, que tan cerca de “romperse” (ser la ultima lectura) esta el motor. 0 es el inicio del experimento y 100 el momento en que el motor se averiá.
- last_quart: Valor booleano. Define si esta lectura se ha de considerar como “altamente posible de averiá”. Útil para entrenar modelos clasificadores.
- is_peak: Valor booleano. Define si esta lectura en especifico es un valor atípico.

Una vez generados estos valores, podemos generar varios gráficos interesantes, que nos ayudan a comprender un poco mas las propiedades de los datos sobre los que trabajaremos.

Si generamos la matriz de correlación de los campos en la Captura tardía, veremos:

[56]:

	bearing_1-1	bearing_1-2	bearing_2-1	bearing_2-2	bearing_3-1	bearing_3-2	bearing_4-1	bearing_4-2	max_vibr	diff_minmax	avg_sensor_val	perc_life	last_quart	is_peak
bearing_1-1	1.000000	-0.346478	0.073917	0.197724	-0.037883	-0.026515	-0.002597	-0.050262	-0.091588	-0.009534	-0.233777	NaN	NaN	0.005341
bearing_1-2	-0.346478	1.000000	0.035111	-0.091564	0.021326	-0.003488	-0.016213	0.024370	-0.049253	0.003703	-0.155042	NaN	NaN	0.004941
bearing_2-1	0.073917	0.035111	1.000000	0.100466	-0.093827	-0.026630	-0.150756	0.055665	-0.086888	-0.014678	-0.284170	NaN	NaN	-0.003681
bearing_2-2	0.197724	-0.091564	0.100466	1.000000	-0.071667	-0.071364	-0.035297	0.019601	-0.067889	-0.022145	-0.283953	NaN	NaN	0.005565
bearing_3-1	-0.037883	0.021326	-0.093827	-0.071667	1.000000	-0.061928	0.115239	-0.039454	-0.168122	0.040856	-0.214843	NaN	NaN	0.040038
bearing_3-2	-0.026515	-0.003488	-0.026630	-0.071364	-0.061928	1.000000	0.056437	-0.039398	-0.192286	0.002962	-0.252594	NaN	NaN	-0.056624
bearing_4-1	-0.002597	-0.016213	-0.150756	-0.035297	0.115239	0.056437	1.000000	-0.221357	-0.157559	0.018072	-0.211538	NaN	NaN	-0.000821
bearing_4-2	-0.050262	0.024370	0.055665	0.019601	-0.039454	-0.039398	-0.221357	1.000000	-0.161297	0.018186	-0.195618	NaN	NaN	-0.002576
max_vibr	-0.091588	-0.049253	-0.086888	-0.067889	-0.168122	-0.192286	-0.157559	-0.161297	1.000000	0.809563	0.702607	NaN	NaN	0.355788
diff_minmax	-0.009534	0.003703	-0.014678	-0.022145	0.040856	0.002962	0.018072	0.018186	0.809563	1.000000	0.578490	NaN	NaN	0.308351
avg_sensor_val	-0.233777	-0.155042	-0.284170	-0.283953	-0.214843	-0.252594	-0.211538	-0.195618	0.702607	0.578490	1.000000	NaN	NaN	0.189206
perc_life	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
last_quart	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
is_peak	0.005341	0.004941	-0.003681	0.005565	0.040038	-0.056624	-0.000821	-0.002576	0.355788	0.308351	0.189206	NaN	NaN	1.000000

Que nos indica que todos los campos de lecturas de los sensores presentan una correlación muy baja mientras que entre los nuevos campos generados, hay casos donde la correlación es elevada, y casos donde es baja. ‘diff_minmax’, ‘max_vibr’ y ‘avg_sensor_val’ presentan unos niveles altos de correlación, sobretodo ‘max_vibr’. Entonces, si en algún momento hemos de implementar un algoritmo que requiera de una muy baja correlación, podría resultar optimo prescindir de ‘max_vibr’.

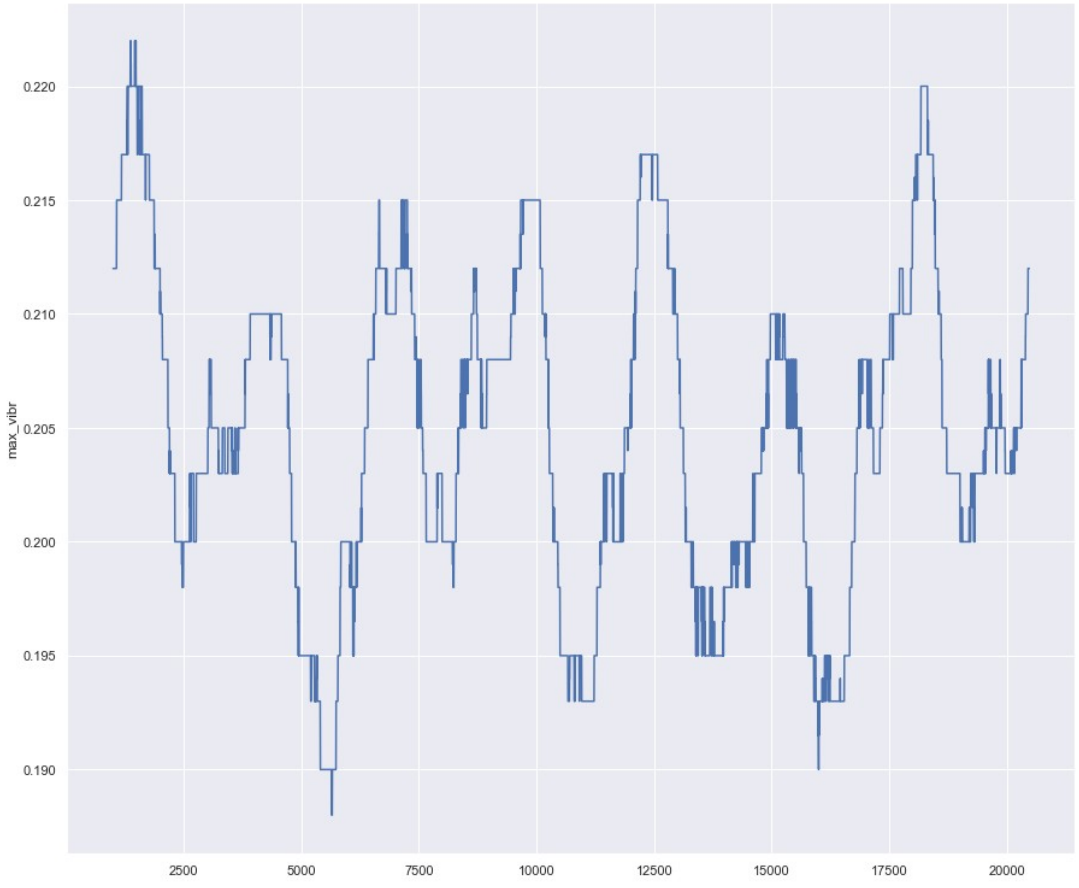
Pero, esta alta correlación también nos permite observar gráficamente unos ciertos patrones. Si imprimimos los valores de 'avg_sensor_val' con 'diff_minmax', como ejes de un plano cartesiano, obtenemos lo siguiente:

Podemos observar que, cuanto mayores sean los valores de lectura de los sensores, también son mayores la diferencia entre el valor mínimo y máximo, lo cual resulta algo lógico, si pensamos que el fenómeno de la vibración es una oscilación entre valores, y cuanta mayor la vibración, mayor la diferencia entre los picos oscilantes.

5.3 Rolling Median

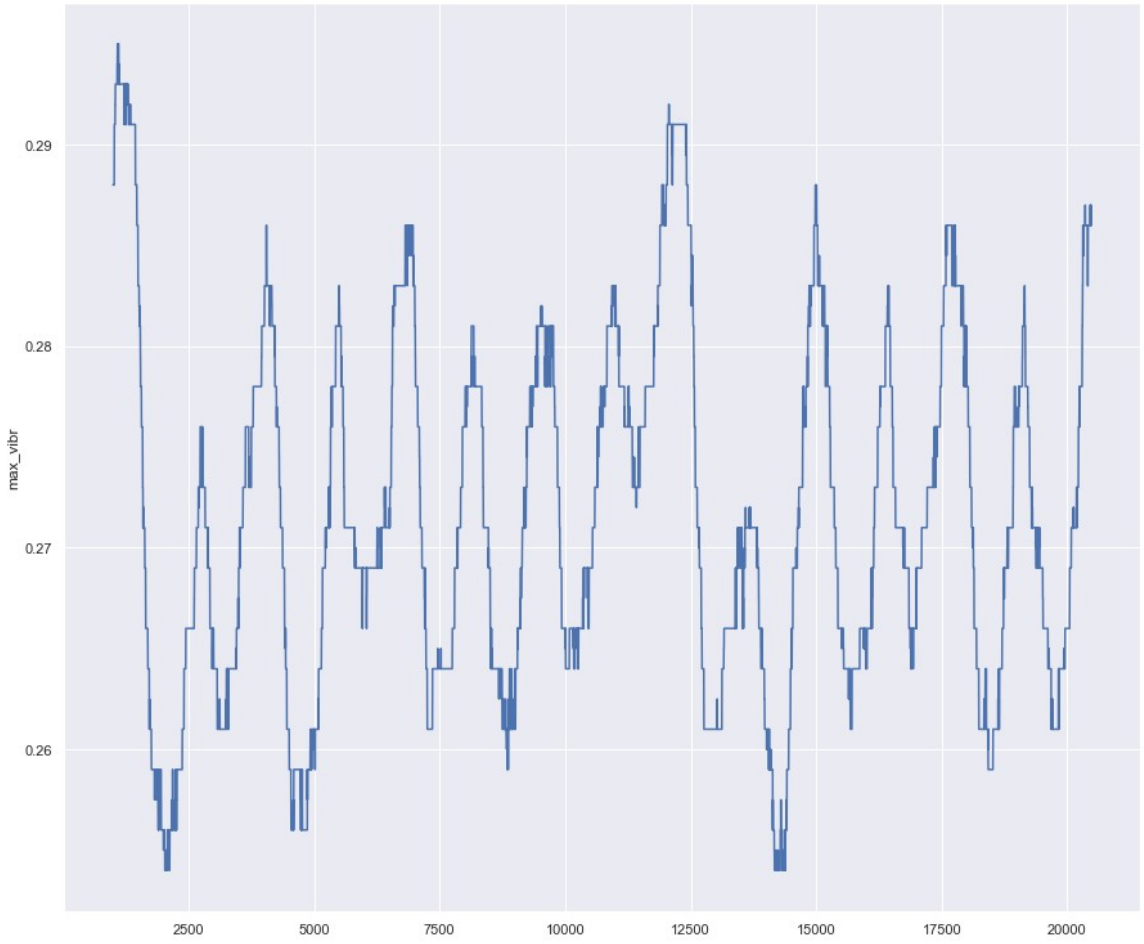
Uno de los métodos que podríamos usar para predecir la vida útil del motor, es utilizando los datos de las capturas, y pasarlos por una Media móvil, reduciendo el ruido presente en los valores, y suavizando las curvas de los gráficos. Entonces, si se detectan algunos patrones clave, se podría usar un simple heuristico que, de pasar los valores por esa simple comprobación, se clasificaría una captura como “peligro de avería” o no.

Si aplicamos una media móvil a los valores de ‘max_vibr’ de la Captura temprana, obtenemos:

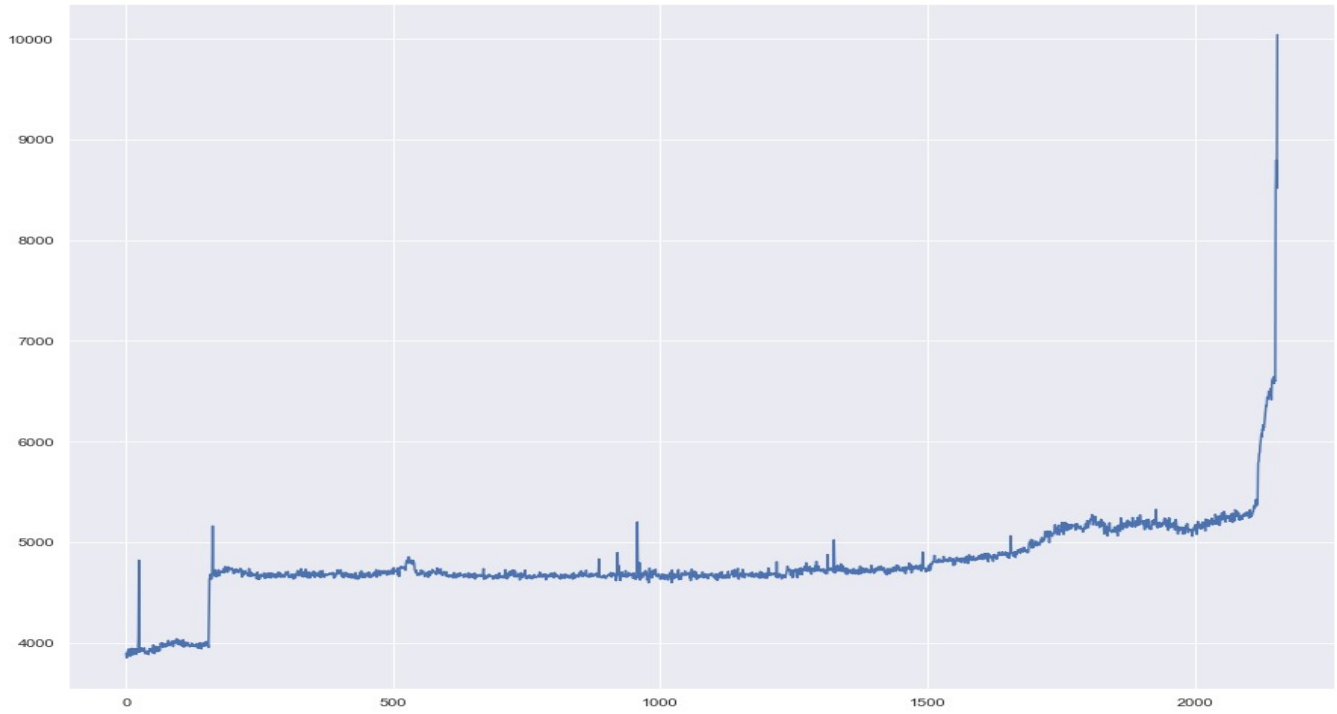


Se puede ver como este valor va oscilando algo irregularmente, pero con unas claras curvas periódicas.

Y en la Captura tardía, tenemos:



El cual vemos que el ancho de las curvas es menor al de el grafico anterior, ademas de presentar unos máximos y mínimos mayores. Entonces, si nosotros sumáramos el conjunto de valores de estas oscilaciones, y las dibujáramos sobre una linea temporal, veríamos este resultado:



En que, los primeros valores iniciales presentan unas oscilaciones muy bajas, que al cabo de doscientos registros, estos se normalizan a algo menos de cinco mil, y se mantienen casi constantemente a ese valor, hasta que comienza a incrementar. Un heurístico idóneo seria comprobar si esta suma de Medianas móviles comienzan a subir de valor, lanzando una alerta a los técnicos el momento en que se presenta esta situación.

5.4 Regresión lineal simple

La regresión lineal es un modelo en que trata de inferir cual serán los valores futuros de un conjunto de datos, definiendo una función en el espacio de coordenadas para un conjunto de observaciones.

Cuando el motor esta estable, el modelo lineal iría prediciendo con un margen de error muy bajo las siguientes observaciones. En cambio, si el motor comienza a presentar un desgaste, y el valor de 'max_vibr' va aumentando (como se ha demostrado que hace al aplicar el Rolling median), entonces los puntos comenzaran a alejarse de las predicciones del modelo inicial.

Este modelo considerara cuando el motor comienza a fallar el momento que detecta que ya no puede predecir fiablemente las siguientes observaciones.

Si observamos el error que nos devuelve el modelo predictivo para la Captura temprana y sus dos observaciones siguientes, y las comparamos con las de la Captura tardía y también sus próximos vecinos, veremos que ciertamente hay una clara diferenciación en el resultado del modelo:

```
arrayResultsLinear1_1 = linearRegressionModelsList[100].predict(outliersDataFrameList[100][['max_vibr']])
arrayResultsLinear1_2 = linearRegressionModelsList[101].predict(outliersDataFrameList[101][['max_vibr']])
arrayResultsLinear1_3 = linearRegressionModelsList[102].predict(outliersDataFrameList[102][['max_vibr']])
arrayResultsLinear2_1 = linearRegressionModelsList[-100].predict(outliersDataFrameList[-100][['max_vibr']])
arrayResultsLinear2_2 = linearRegressionModelsList[-99].predict(outliersDataFrameList[-99][['max_vibr']])
arrayResultsLinear2_3 = linearRegressionModelsList[-98].predict(outliersDataFrameList[-98][['max_vibr']])

print(sum(arrayResultsLinear1_1) / len(arrayResultsLinear1_1))
print(sum(arrayResultsLinear1_2) / len(arrayResultsLinear1_2))
print(sum(arrayResultsLinear1_3) / len(arrayResultsLinear1_3))

print(sum(arrayResultsLinear2_1) / len(arrayResultsLinear2_1))
print(sum(arrayResultsLinear2_2) / len(arrayResultsLinear2_2))
print(sum(arrayResultsLinear2_3) / len(arrayResultsLinear2_3))
```

```
0.1909621563552876
0.18897270374530128
0.19130099125933753
0.25906313784852203
0.2575247814834646
0.2537561892670554
```

Entonces, un modelo predictivo eficaz para predecir el estado del motor, es avisar cuando el margen de error de la predicción del modelo sea suficientemente elevado, superando un valor heuristico.

5.5 Árbol de Decisión

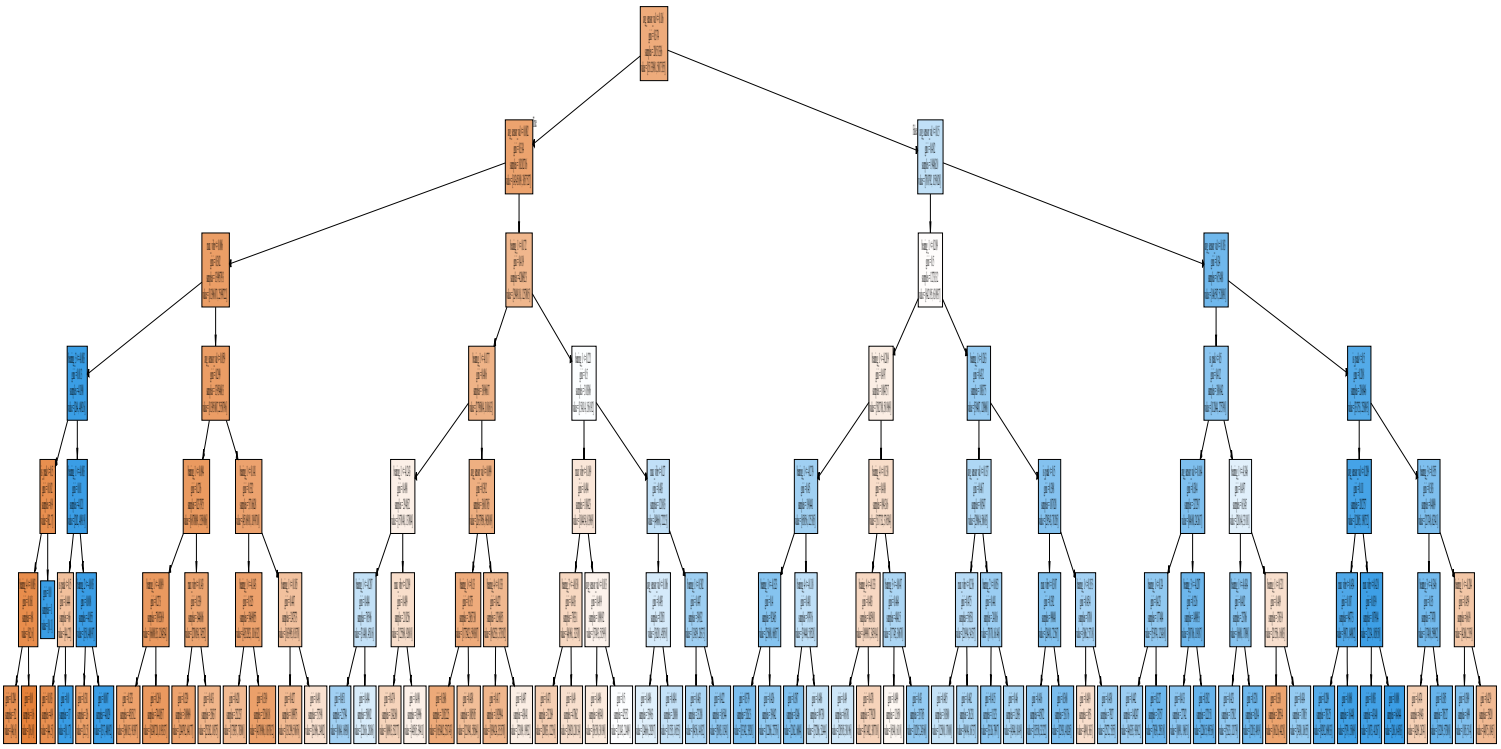
Los arboles de decisión son un algoritmo que automáticamente nos valora si una observación es de un motor saludable o si presenta un alto riesgo de avería. Los métodos anteriores asignan un valor numérico continuo. El proceso para conseguir una valoración discreta es a través de un valor heuristico. Los arboles de decisión ya puede clasificar directamente el resultado, sin necesidad de definir heuristicos ambiguos y relativamente poco fiables.

Este algoritmo trata de generar un Grafo en forma de Arbol, en el que los nodos contienen conjuntos de observaciones, el mismo nodo procesa una operación booleana a todos los datos, separa los datos entre los que han sido clasificados como “True” y los que han sido “False” y los enviá a nodos diferentes. El objetivo del algoritmo es conseguir generar un conjunto de nodos que separe lo mas posible los datos entre ellos, consiguiendo idealmente una relación de 50/50.

Un valor común para evaluar tal separación es el Coeficiente Gini, común en Macroeconomia, que describe el desbalance en una distribución desigual. Cuanto mas tiende a 0, mas igual es la distribución, y por tanto es el valor a optimizar por el algoritmo.

Una de las grandes ventajas que posee este algoritmo de Aprendizaje Automático, es que su proceso de clasificación es totalmente transparente. Eso quiere decir que el modelo no solo muestra que clasificación sugiere a un conjunto de datos, sino también el porque la sugiere.

Si generamos un árbol de decisiones para los datos, obtendremos el siguiente resultado:



Como podemos observar, es un modelo altamente complejo, con múltiples divisiones de los datos finales.

Desgraciadamente, si vemos el color de los nodos hoja, podemos observar que hay una muy baja clasificación satisfactoria de los datos. Cuanto mas “incoloro” es el nodo hoja, menos seguro esta de haber clasificado exitosamente los valores. Hace falta generar un árbol con niveles de profundidad mayores, pero el coste de procesamiento se eleva exponencialmente, hasta el punto que resulta casi inviable procesarlo.

Ademas, este algoritmo puede también resultar erróneo, debido a que la naturaleza de los datos con los que tratamos es linealmente temporal, y por tanto, las observaciones no son independientes, sino que mantienen un contexto y una dependencia entre las observaciones anteriores y posteriores. Los arboles de decisión no tienen en cuenta esa temporalidad, y como les falta ese contexto, los resultados son desfavorables.

5.6 *Gardient Boosting*

Gardient Boosting es un algoritmo que entrena todo un gran conjunto de modelos de predicción simples, a menudo de complejidad muy reducida y precisión baja, y los va entrenando paralelamente para que, el sumatorio de predicciones de los modelos clasifiquen, en conjunto, mucho mejor que cuando tan solo un modelo demasiado complejo y pesado se encarga de clasificar todos los elementos.

Poder clasificar a través del uso de muchos modelos diferentes permite una mayor flexibilidad y adaptabilidad ante los datos que ha de procesar, a cambio de una reducción en transparencia y explicación de como ha llegado a sus conclusiones.

Como se ha sugerido anteriormente, se cree que, debido a que los datos están relacionados entre ellos a través de su orden en la linea temporal, los algoritmos de clasificación que no tienen en cuenta esa relación temporal no conseguirán altos indices de éxito.

Una de las grandes complicaciones que tiene este método, es la hiperparametrizacion: Existe una cantidad abrumadora de opciones y configuraciones que se le puede definir al algoritmo, que se necesita de un muy buen conocimiento teórico y de las propiedades de los datos para obtener los resultados mas optimizados posibles.

La configuración y generación del modelo se ha hecho a través del uso de XGBoost, la librería que se ha hecho uso para desarrollar este algoritmo.

```
data_dmatrix = xgboost.DMatrix(data=Dec_tree_Dataframe,label=outliersDataFrameList_target)
xg_Model = xgboost.XGBClassifier(n_estimators=20, n_jobs=6, tree_method="approx", subsample=0.5,objective="binary:logistic")
xg_Model.fit(Dec_tree_Dataframe, outliersDataFrameList_target)
```

La segunda linea del código adjunto arriba describe los parámetros que se han configurado en el proyecto.

“n_estimators” es el numero de modelos predictivos sencillos que se usaran.

“n_jobs”es el numero de hilos de ejecución paralelos que se usan para entrenar los modelos.

“tree_method” es el algoritmo de generación de los modelos que se desea utilizar, siendo “approx” el que genera arboles de decisión cuyo calculo de clasificación de los datos es aproximado y optimizado a costa de exactitud al clasificar, creando modelos con menor tasa de éxito de clasificación pero mas rápidos de procesar.

“subsample” define que porcentaje de datos se usaran para entrenar los modelos. En este caso, por cada modelo solo se escogerá aleatoriamente la mitad de datos a entrenar a cada modelo.

“objective” es la función de optimización del modelo. Este modelo buscara entrenarse para minimizar al máximo el error que represente el valor escogido. “binary:logistics” define un

porcentaje de la Regresión logística de los resultados, en forma de porcentaje. Este objetivo es valido para problemas de clasificación binarios, como lo es el problema que intentamos solventar (Si el motor esta dañado o en buenas condiciones).

El modelo generado ha resultado tener una precisión de clasificación del 75.73%.

5.7 Long-Short Term Memory

La estructura de datos mas mediática de los últimos años es la Red Neuronal, cuyo nombre ya indica que esta herramienta esta inspirada en la estructura del cerebro del Reino animal. Esta consiste en un grafo en que sus nodos, llamados Neurona, en que se ordenan por capas de profundidad. Cada capa puede variar en su densidad de neuronas. Las neuronas reciben datos de una multitud de nodos previos a ella, multiplica cada valor por un factor de multiplicación asociada a el vértice, suma los valores ademas de un valor de peso propio del nodo y pasa el valor final a una función de activación, la cual esta decidirá si la neurona deberá "activarse" o no, y que valor extraerá. Este se propaga a los siguientes nodos sucesivamente.

Los algoritmos de Aprendizaje Automático generan esta estructura de datos y, en la fase de entrenamiento, cada vez que la Red Neuronal ha procesado los datos y se evalúa el resultado de su proceso, este reajusta los pesos de los vértices y trata de configurar la Red para que acabe generando un modelo predictivo con altos valores de éxito.

Long-Short Term Memory (de ahora en adelante "LSTM") es un tipo de red neuronal que es capaz de retener información importante sobre los datos procesados anteriormente, y que esta información influya en tiempo real a los nodos de la red neuronal, consiguiendo así analizar los datos manteniendo el contexto temporal, y se suele decir informalmente que es un algoritmo con capacidad de "memorizar".

Para implementar este algoritmo, se ha hecho uso de Tensoflow y Keras, librerías que soportan el uso de algoritmos de Inteligencia Artificial basados en Redes Neuronales.

```
model = Sequential()
inputShape = numpy.reshape(outliersArrayList,(numpy.shape(outliersArrayList)[0], 1, numpy.shape(outliersArrayList)[1]))

#model.add(CuDNNLSTM(100,return_sequences=True, input_shape = (1, 7)))
#model.add(LSTM(125, input_shape = (1, 7), activation='relu'))
model.add(CuDNNLSTM(25, input_shape = (1, 7), return_sequences=False))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(inputShape, numpy.asarray(outliersDataFrameList), batch_size=1, verbose=1)
print(model.summary())

#model.fit(outliersArrayList, target, nb_epoch=10000, batch_size=1, verbose=2,validation_data=(x_test, y_test))
```

El código expuesto arriba define las capas de la red neuronal que deseamos crear. El modelo estará compuesto de una capa de nodos LSTM, seguido de una capa de "Dropout" del 20%, cuyo función es eliminar el inmuta de un 20% de los nodos por los que pasa, el cual ayuda a evitar el *overfitting* y generar modelos mas flexibles, seguido de una capa de Normalizacion de los valores, que ayuda a limpiar los valores normalizando el rango de estos, y estos acaban siendo los datos de entrada de una ultima capa de un solo nodo, cuya función sera clasificar la conclusión final de los datos procesados.

El resumen de datos de la generación del modelo es el siguiente:

```
1023950/1023950 [==============================] - 2857s 3ms/step - loss: 0.5742 - accuracy: 0.7400
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
cu_dnnlstm (CuDNNLSTM)       (None, 25)                3400
_________________________________________________________________
dropout (Dropout)            (None, 25)                0
_________________________________________________________________
batch_normalization (BatchNo (None, 25)                100
_________________________________________________________________
dense (Dense)                (None, 1)                 26
=================================================================
Total params: 3,526
Trainable params: 3,476
Non-trainable params: 50
_________________________________________________________________
None
```

Si nos fijamos en el valor de “accuracity”, este nos refleja que el 74% de datos han sido clasificados correctamente. Este resultado, aunque no llegue para nada a un ideal 90%, para ser la fase de entrenamiento del modelo, no es malo que tenga este resultado, ya que nos demuestra que el modelo no esta sufriendo de *overfitting*.

```
inputShape = numpy.reshape(outliersArrayList,(numpy.shape(outliersArrayList)[0], 1, numpy.shape(outliersArrayList)[1]))

results = model.evaluate(inputShape, numpy.asarray(outliersDataFrameList), batch_size=128)
results
#predict = model.predict(data)
```

```
8000/8000 [==============================] - 18s 2ms/step - loss: 0.5728 - accuracy: 0.7400

[0.5727645754814148, 0.7400000095367432]
```

Si evaluamos como el modelo clasifica nuevos datos, podemos observar que también ha clasificado los valores con un 74% de éxito. Por tanto, el modelo es capaz de valorar cuando el motor presenta defectos un 74% de comprobaciones. Este valor se ha considerado sorprendente debido a que, como los modelos generados no mantienen ninguna relación temporal entre los datos, se pensaba que los resultados finales serian mucho peores. Pero, XGBoost ha sido capaz de generar todo un conjunto de Arboles de decisión que, en conjunto, clasifica correctamente los instantes capturados.

Seguramente eso es debido a que varios arboles habrán sido desarrollados para identificar patrones de valores que solo aparecen en las fases iniciales de la vida útil del motor, y otros arboles que en conjunto capturan patrones para las fases finales de este.

6. Conclusiones

El proyecto ha conseguido generar un conjunto de modelos predictivos aplicando sobre los mismos datos diferentes algoritmos. Se ha conseguido analizar el resultado del proceso de cada uno de estos, y la eficacia que han demostrado para resolver el problema de diagnostico de salud de un motor eléctrico.

Una de las observaciones mas interesantes que podemos extraer de los resultados, es que hemos podido sacar información lógica sobre como y porque el motor eléctrico ha sufrido la avería final, con algoritmos como Rolling Media y Regresion Lineal que describen una clara evolución en los valores extraídos de los sensores de vibración, y estos algoritmos de regresión sencilla pueden servir perfectamente para implementar un modelo predictivo fácil de generar y que requiera pocos recursos.

Todo y así, el uso practico de estos modelos requerirían una configuración previa de un umbral heuristico para clasificar una lectura de datos, y este umbral puede diferenciarse mucho dependiendo de el motor que se estudia y los datos a los que se compara el umbral. Se debería de investigar mas ejemplos de motores y su diagnostico dando uso a estos algoritmos sencillos.

Otra observación que cabe destacar es que, el algoritmo que ha generado peores resultados ha sido sin duda el Arbol de decisiones. Este modelo no mantienen relación temporal y acaba abarcando una estructura tan compleja que resulta ineficiente. Para generar un solo modelo que identifique correctamente todos los posibles patrones que los datos puedan aportar a su clasificación, se debería crear un modelo de unas dimensiones inalcanzables.

El LSTM fue el algoritmo que, desde un principio, prometía ofrecer los resultados mas favorables: Las redes neuronales se suelen considerar que, de estar bien diseñadas y entrenadas, son el modelo mas flexible, potente e inteligente de los demás algoritmos, y han sido capaces de generar automáticamente muchas soluciones a problemas que parecen imposibles de diseñar a través de la lógica humana.
Todo y así, los resultados finales han resultado ser muy similares a los que han ofrecido el Gradient boosting y sin embargo, LSTM ha tardado muchísimo mas en procesar el modelo y calcular la evaluación final, y ha consumido mucha mas memoria al equipo para solventar el problema, sobretodo en comparación al modelo generado por XGBoost.
Se debería de analizar en mas profundidad mejores configuraciones y optimizaciones a los modelos generados tanto por Gradient boosting como por LSTM, debido a que ambos algoritmos aun presentan muchísimo margen de mejora aun y ofreciendo resultados positivos.

Y finalmente, se ha de reconocer que se ha deseado poder evaluar mejor los modelos: No se han aplicado la gran mayoría de algoritmos de evaluación propuestos en la sección “Algoritmos candidatos” del documento, ni se han generado modelos predictivos usando “Naive Bayes”, “Autoencoder” ni “Filtro de Kalman”. Estos algoritmos candidatos pueden perfectamente ofrecer resultados positivos si se desarrollan unas implementaciones optimizadas a el problema a solventar.

7. Bibliografia

[1] Flavio Trojan, Rui F. M. Marçal, “Proposal of Maintenance-types Classification to Clarify Maintenance Concepts in Production and Operations Management”
Academic Star Publishing Company, 2017
[2] U.S Department of Energy, “Operations & Maintenance Best Practices, Release 3.0”
Academic Star Publishing Company, August 2010
[3] Russell, Stuart J., Norvig, Peter, “Artificial Intelligence: A Modern Approach (2nd ed.)” Upper Saddle River, New Jersey: Prentice Hall, 2003
[4] Junyan Hu , Parijat Bhowmick , Farshad Arvin, Alexander Lanzon, Barry Lennox , “Cooperative Control of Heterogeneous Connected Vehicle Platoons: An Adaptive Leader-Following Approach”
IEEE ROBOTICS AND AUTOMATION LETTERS, VOL. 5, NO. 2, 2020
[5] Vijaya Sai Krishna Gottipati, Kyle Goyette, Ahmad Chamseddine, Breandan Considine, “Deep Pepper: Expert Iteration based Chess agent in the Reinforcement Learning Setting”
arXiv:1806.00683v2 [cs.AI], 2018
[6] OpenAI, “Dota 2 with Large Scale Deep Reinforcement Learning”
arXiv:1912.06680v1 [cs.LG], 2019
[7] Anith Adibah Hasseim, Rubita Sudirman, Puspa Inayat Khalid, “Handwriting Classification Based on Support Vector Machine with Cross Validation”
Scientific Research, Engineering, 5, 84-87, 2013
[8] Leidy Esperanza MOLINA FERNÁNDEZ, Prof. Dr. Sandjai BHULAI, “Recommendation System for Netflix”
VRIJE UNIVERSITEIT AMSTERDAM, 2018
[9] Cem Dilmegani, “Future of AI according to top AI experts: In-Depth Guide for 2022”
https://research.aimultiple.com/future-of-ai/, 2022
[10] Cem Dilmegani, “Predictive Maintenance Tools: Top 12 Tools & Selection Guide”
https://research.aimultiple.com/predictive-maintenance-tools/, 2021
[11] Thomas L. Griffiths, “Understanding Human Intelligence through Human Limitations”
Trends in Cognitive Sciences, 2020
[12] Cordenne Brewster, “8 Best Programming Languages for AI Development in 2022”
https://trio.dev/blog/best-languages-for-ai, 2022
[13] Python Software Foundation, “Find, install and publish Python packages with the Python Package Index”
https://pypi.org/, 2022
[14] Distributed (Deep) Machine Learning Community, “XGBoost - ML winning solutions (incomplete list)”

https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions, 2022
[15] Pavithra Vijay, “Timeseries anomaly detection using an Autoencoder” https://keras.io/examples/timeseries/timeseries_anomaly_detection/, 2020
[16] Eric Hofesmann, “The Machine Learning Lifecycle in 2021” https://towardsdatascience.com/the-machine-learning-lifecycle-in-2021-473717c633bc, 2021
[17] Talent.com, “¿Cuánto gana un Ingeniero informático en España?” https://es.talent.com/salary?job=ingeniero+inform%C3%A1tico#:~:text=El%20salario%20i ngeniero%20inform%C3%A1tico%20promedio,hasta%20%E2%82%AC%2030.000%20al%2 0a%C3%B1o. , 2022
[18] Instituto Nacional de Estadística, “PIB pm Oferta (Precios corrientes)” https://www.ine.es/jaxiT3/Datos.htm?t=30678, 2021
[19] Instituto Nacional de Estadística, “Gasto en I+D interna en relación con el PIB por años y sectores de ejecución” https://www.ine.es/jaxi/Datos.htm?tpx=50191, 2020