UNIVERSIDAD DE PIURA

FACULTAD DE INGENIERÍA

Metodología basada en inteligencia artificial en la gestión del mantenimiento 4.0

Trabajo de Investigación para optar el Grado de
Máster en Ingeniería Mecánico -Eléctrica con mención en Sistemas Energéticos y Mantenimiento

Henry Segundo Ramos Arévalo
Diógenes Olmedo Torres Rivera

Trabajo de Investigación para optar el Grado de
Máster en Ingeniería Mecánico -Eléctrica con mención en Sistemas Eléctricos y Automatización Industrial

John Marlo Suyón Tejada

Asesor:
Dr. Ing. César Alberto Chinguel Arrese

Piura, marzo de 2023

NOMBRE DEL TRABAJO

Metodología Basada en Inteligencia Artif icial en la Gestión del Mantenimiento

AUTOR

Grupo Dr. Chinguel Grupo Dr. Chinguel

RECUENTO DE PALABRAS

24899 Words

RECUENTO DE CARACTERES

137703 Characters

RECUENTO DE PÁGINAS

103 Pages

TAMAÑO DEL ARCHIVO

3.2MB

FECHA DE ENTREGA

Mar 12, 2023 8:43 AM GMT-5

FECHA DEL INFORME

Mar 12, 2023 8:45 AM GMT-5

24% de similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para cada base d

- 19% Base de datos de Internet
- Base de datos de Crossref
- 17% Base de datos de trabajos entregados
- 9% Base de datos de publicaciones
- Base de datos de contenido publicado de Crossr

Dedicatoria

Damos gracias a Dios y la Virgen María por su guía y protección. Al Dr. César Chinguel por su valiosa orientación y compromiso en el desarrollo de este proyecto. También reconocemos y agradecemos la colaboración incondicional de los ingenieros Pedro y Lino. Finalmente, a nuestras familias y amigos por su amor y apoyo constante.

UNIVERSITAS STUDIORUM PIURENSIS

Resumen

Este presente informe describe en la aplicación de la Inteligencia Artificial en la Gestión del Mantenimiento 4.0. El informe se divide en cuatro capítulos.

En el Capítulo 1, se presenta una visión general de la teoría matemática detrás de la Inteligencia Artificial y el Machine Learning. Se describen temas como la regresión lineal, el método del descenso del gradiente, el aprendizaje profundo y la red neuronal LSTM (Long Short Term Memory). Además, se discute la evolución de las generaciones del mantenimiento y su relación con los lenguajes de programación, en particular Python, y se presenta una descripción de las herramientas disponibles en Google Platform.

El Capítulo 2 se enfoca en el Estado del Arte, describe el uso de la Inteligencia Artificial para el Gemelo Digital, incluyendo arquitectura y metodología. El uso de la Inteligencia Artificial para la estimación del RUL (Remaining Useful Life) es un tema objetivo en este estudio, por lo que también se presenta cinco casos de estudio publicados en páginas web prestigiosas, en los que se han utilizado metodologías y arquitecturas específicas.

En el Capítulo 3 se discuten los casos de estudio presentados en el capítulo anterior y se define una metodología aplicable a nuestro caso real. Se presenta una visión general del proceso de aplicación de la Inteligencia Artificial a la Gestión del Mantenimiento 4.0, desde la identificación del problema hasta la implementación de la solución.

Finalmente, en el Capítulo 4 se analizan los resultados obtenidos de los ensayos en la nube y se ofrecen recomendaciones para futuros trabajos en este campo. Se discuten los desafíos y las limitaciones de la aplicación de la Inteligencia Artificial en la Gestión del Mantenimiento 4.0 y se presentan recomendaciones para superarlos y mejorar la eficacia de la metodología.

En suma, el presente estudio proporciona una descripción detallada de la aplicación de la Inteligencia Artificial en la Gestión del Mantenimiento 4.0, que abarca desde la teoría matemática hasta la metodología aplicable al caso real y los resultados obtenidos de los ensayos en la Google Platform, así como las recomendaciones para futuros trabajos en el tema.

UNIVERSITAS STUDIORUM PIURENSIS

Tabla de contenido

UNIVERSITAS STUDIORUM PIURENSIS

Lista de tablas

UNIVERSITAS STUDIORUM PIURENSIS

Lista de figuras

Introducción

El mantenimiento de maquinarias y equipos ha sido una actividad clave en la industria desde hace décadas. Tradicionalmente, se ha llevado a cabo de manera correctiva, preventiva, basándose en un calendario de mantenimiento programado, independientemente de si los equipos necesitaban ser reparados o no. Este enfoque ha sido costoso en términos de tiempo y recursos, ya que los equipos podrían estar fuera de servicio innecesariamente, y las piezas y componentes podrían ser reemplazados antes de que fuera realmente necesario.

Con la evolución de las tecnologías de la información y la llegada de la Industria 4.0, se ha desarrollado una nueva forma de mantenimiento: el mantenimiento 4.0. Este enfoque se basa en la integración de tecnologías digitales y físicas para el mantenimiento de maquinarias y equipos.

El objetivo de este trabajo es utilizar las redes neuronales artificiales para predecir la vida útil remanente (RUL) de los componentes o de los activos, en este caso en particular de los rodamientos del motor eléctrico de la bomba del circuito cerrado del sistema de enfriamiento de agua de proceso de una planta industrial; utilizando datos históricos de sensores de monitoreo como vibración y temperatura.

Se analiza la dinámica del modelo y la introducción de la variable temporal, considerando de esta manera las horas de operación para mejorar la precisión de las predicciones. Asimismo, se evalúa el uso de un modelo estadístico ARIMA como para generar datos sintéticos y se discute su aplicación en la predicción de datos faltantes.

En la actualidad, las empresas enfrentan una fuerte competencia en los mercados globales y, por lo tanto, la necesidad de controlar los costos y aumentar la eficiencia es cada vez más importante. El mantenimiento 4.0 puede ayudar a las empresas a reducir costos al minimizar los tiempos de inactividad y evitar el reemplazo innecesario de componentes. Asimismo, la implementación de tecnologías de inteligencia artificial en la gestión del mantenimiento puede permitir una mayor precisión en la identificación de problemas y una mayor eficiencia en la asignación de recursos, lo que puede traducirse en una mejora en la rentabilidad y la competitividad de la empresa.

UNIVERSITAS STUDIORUM PIURENSIS

Capítulo 1

La Inteligencia Artificial y el Machine Learning

La inteligencia artificial (IA) es un campo de la informática que se enfoca en la creación de sistemas y algoritmos que pueden realizar tareas que normalmente requieren inteligencia humana, como el aprendizaje, la percepción, la toma de decisiones y el procesamiento del lenguaje natural. Como explica el pionero en el campo de la inteligencia artificial, Alan Turing, "las máquinas pueden pensar".

La IA puede ser dividida en diferentes categorías, tales como la inteligencia artificial débil (o estrecha) que se enfoca en resolver tareas específicas, y la inteligencia artificial fuerte (o general) que busca replicar la inteligencia humana de manera más completa. Según el experto en IA John McCarthy, "la inteligencia artificial es la ciencia e ingeniería de hacer máquinas inteligentes".

Por otro lado, el aprendizaje automático o Machine Learning (ML) es una rama de la IA que se enfoca en enseñar a los sistemas a aprender y mejorar de manera autónoma, sin necesidad de ser programados explícitamente para realizar tareas específicas. El ML utiliza algoritmos y modelos estadísticos para analizar grandes conjuntos de datos y extraer patrones que pueden ser utilizados para tomar decisiones o hacer predicciones. Como explica el experto en ML Tom Mitchell, "un programa de computadora se dice que aprende de la experiencia E con respecto a alguna tarea T y medida de rendimiento P, si su rendimiento en la tarea T, medida por P, mejora con la experiencia E".

Existen varios tipos de aprendizaje automático, tales como el aprendizaje supervisado, el aprendizaje no supervisado y el aprendizaje por refuerzo, cada uno de los cuales tiene sus propias características y aplicaciones. Según el experto en ML Andrew Ng, "el aprendizaje automático es el campo de estudio que le da a las computadoras la capacidad de aprender sin ser explícitamente programadas".

En los últimos años, el Deep Learning, una técnica de ML basada en redes neuronales artificiales ha revolucionado el campo de la IA y ha permitido avances significativos en tareas como el reconocimiento de imágenes, el procesamiento del lenguaje natural y la conducción autónoma. Según los "padres" del Deep Learning, Yoshua Bengio, Geoffrey Hinton y Yann

LeCun, "el aprendizaje profundo ha permitido que las máquinas superen a los humanos en tareas específicas que se consideran difíciles o imposibles para los humanos".

1.1 Regresión Lineal

(Bisong, "Building Machine Learning and Deep Learnig Models on Google Cloud Platform", 2019, pág. 231..237)

Para representar un sistema utilizaremos modelos matemáticos que describan su comportamiento real en un proceso, sistema o subsistema de una planta industrial. Se puede utilizar ecuaciones diferenciales y transformada de Laplace para representar sistemas básicos y sencillos, como por ejemplo para un gas ideal en un sistema isovolumétrico, la presión podrá ser evaluada mediante la siguiente ecuación:

$$P\big(T(t)\big) = \frac{R\,T\,n}{V} \tag{1}$$

1.1.1 El modelo de regresión

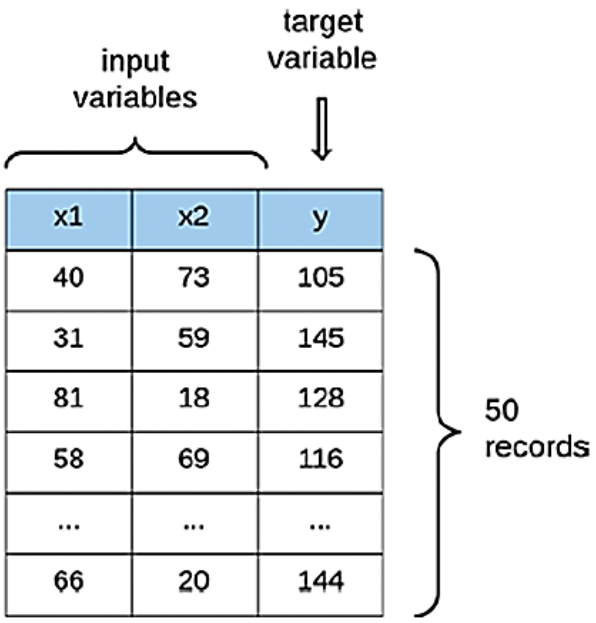
La regresión lineal es una técnica estadística que se utiliza para predecir el valor de una variable objetivo a partir de una o más variables predictoras. En este método, se asume que la relación entre la variable objetivo y las variables predictoras se puede expresar como una combinación lineal de las características.

Una combinación lineal es una suma de varios vectores, donde cada vector se multiplica por una constante o número escalar. En términos simples, esto significa que, para predecir la variable objetivo, se deben combinar las variables predictoras de tal manera que la combinación resultante pueda explicar la variabilidad en la variable objetivo.

Para ilustrar esto, Se puede considerar un conjunto de datos como el de la figura 1, que consta de dos características y una variable objetivo. En este conjunto de datos, hay 50 observaciones y se utilizan estas observaciones para encontrar una combinación lineal de las características que se ajuste mejor a la variable objetivo. Esto se hace para que cuando se tenga una nueva observación con características similares, se pueda utilizar la combinación lineal para predecir su valor de la variable objetivo.

Figura 1

Conjunto de datos de la muestra



Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag. 232.

En la regresión lineal, los pesos se asignan a cada característica en el conjunto de datos y se ajustan para capturar la relación subyacente o no tan evidente a simple vista, con la variable objetivo. El modelo de regresión lineal se escribe formalmente como:

$$\bar{y} = W_0 + W_1X_1 + W_2X_2 + \ldots + W_nX_n \quad (2)$$

Donde:

$\bar{y}$: es el valor aproximado de la salida y que queremos predecir.

W_i : donde i = {1, 2, …n}, es el peso asignado a cada característica en el conjunto de datos. La notación n es el tamaño de las características del conjunto de datos.

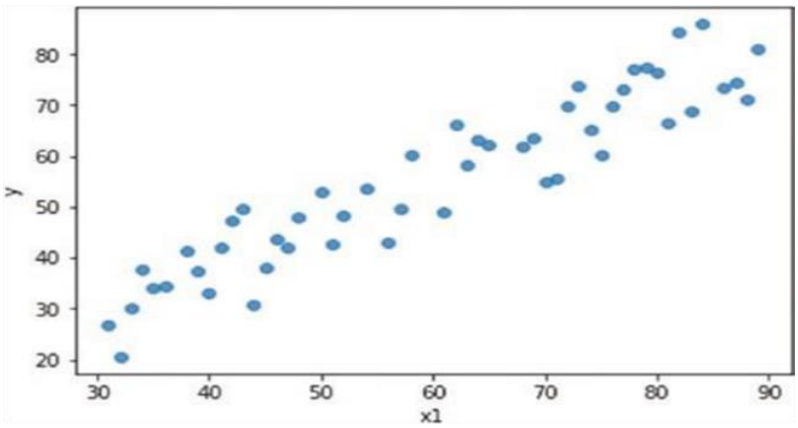
W_0: representa el término "sesgo".

1.1.2 Representación visual de la regresión lineal

En la figura 2 se ha representado gráficamente la primera característica x_1 y la variable objetivo $\bar{y}$ del conjunto de datos con los 50 registros, de esta forma es más fácil visualizar la dispersión bidimensional.

Figura 2

Diagrama de dispersión de Datos

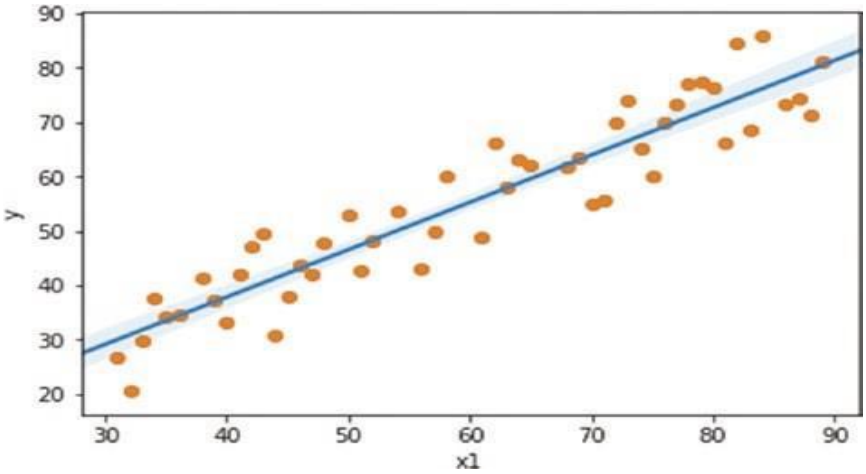


Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag. 232.

El propósito de utilizar el modelo de regresión lineal es encontrar una línea que se ajuste a los puntos de datos. Una vez encontrada, se parecerá a la línea que se muestra en la figura 3. Esta línea se conoce como la línea de regresión, la cual representa la mejor aproximación a los datos.

Figura 3

Aproximación a la dispersión de datos mediante una Recta



Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag. 234.

Para obtener la línea en el modelo de regresión lineal, es necesario establecer la función de costo, que también se conoce como función de pérdida. En el aprendizaje automático, el costo es una medida de error que se minimiza a través del algoritmo de aprendizaje. El costo también puede definirse como la penalización por una predicción incorrecta del modelo.

En el caso específico del modelo de regresión lineal, la función de costo se define como la mitad de la suma de las diferencias al cuadrado entre los valores predichos y los valores reales para todo el conjunto de datos. Esta función de costo se conoce como el error cuadrático medio y se representa como:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{i=n}(\bar{y}_i - y_i)^2 \quad (3)$$

Donde n es el número de datos de prueba.

Dicho de forma más sencilla, cuanto más se acerque el valor aproximado de la variable objetivo $\bar{y}$ a la variable real y, menor será el costo y más cercanas a la realidad serán las predicciones de nuestro modelo.

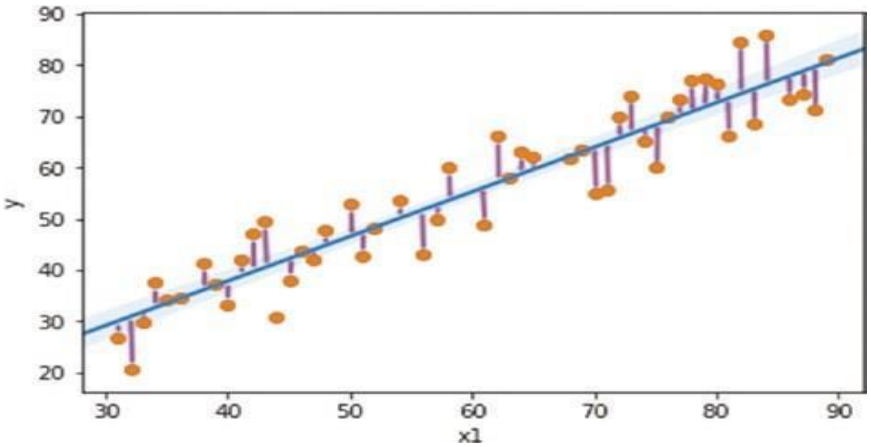
Definida la función de costo, se puede utilizar un algoritmo de optimización como el descenso de gradiente para minimizar el costo J(θ) mediante la actualización los pesos del modelo de regresión lineal.

En el aprendizaje automático, la aproximación de la regresión lineal difiere levemente de la estadística tradicional. En estadística, el objetivo principal del modelo de regresión es entender las relaciones entre las características y los objetivos al interpretar los valores promedio. En cambio, en el aprendizaje automático, el propósito del modelo de regresión lineal es hacer predicciones sobre los objetivos a partir de nuevas muestras.

La Figura 4 presenta un modelo de regresión lineal que optimiza la función de costo. Las líneas verticales moradas representan la desviación entre los resultados reales y las predicciones. En un modelo de regresión lineal, es crucial minimizar el error entre las etiquetas predichas y las etiquetas reales del conjunto de datos

Figura 4

Modelo de Regresión Lineal Mostrando los Residuos



Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag. 235.

En caso de que todos los puntos de la Figura 4 coincidan exactamente con la línea de regresión predicha, el error será igual a 0. Al interpretar el modelo de regresión, el objetivo es lograr que la medida de error sea lo más reducida posible.

Sin embargo, el enfoque principal radica en obtener un bajo error cuando evaluamos el modelo en el conjunto de datos de prueba. Es importante recordar que la prueba de aprendizaje implica que el modelo pueda generalizar y hacer predicciones precisas sobre ejemplos que no estuvieron presentes durante el proceso de entrenamiento.

1.1.3 Adaptación a la No linealidad

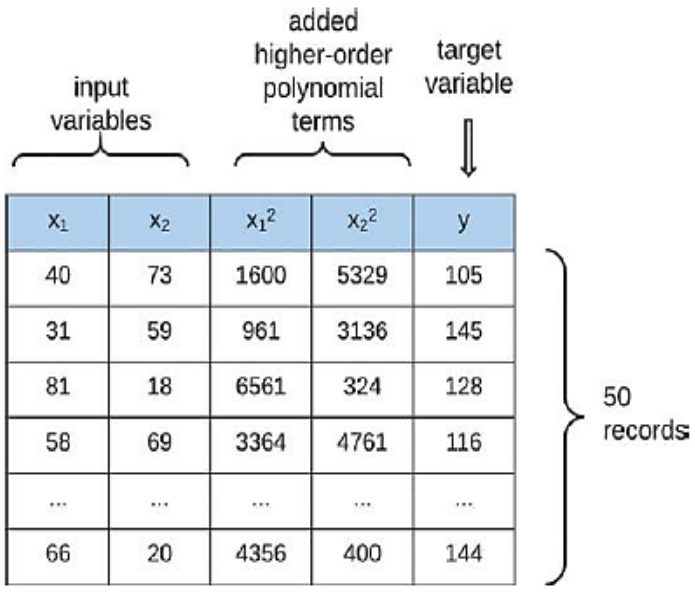
Aunque la regresión lineal supone que el conjunto de datos sigue una tendencia lineal, esta situación no se presenta en la mayoría de los casos.

Sin embargo, existe la posibilidad de adaptar la regresión lineal para ajustar o construir un modelo para conjuntos de datos no lineales. Este proceso de añadir no linealidad a los modelos lineales se conoce como regresión polinómica.

La regresión polinómica establece una relación no lineal entre los datos al incluir términos polinómicos de orden superior de las características de los datos existentes como nuevas características en el conjunto de datos. La figura 5 muestra cómo se añaden nuevas características para adaptar el modelo a la tendencia no lineal de los datos.

Figura 5

Añadir características polinómicas al conjunto de datos



x_1	x_2	x_1^2	x_2^2	y
40	73	1600	5329	105
31	59	961	3136	145
81	18	6561	324	128
58	69	3364	4761	116
...	...	...	...	...
66	20	4356	400	144

Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag. 237.

Donde:

x_1,x_2 : son las variables de entrada.

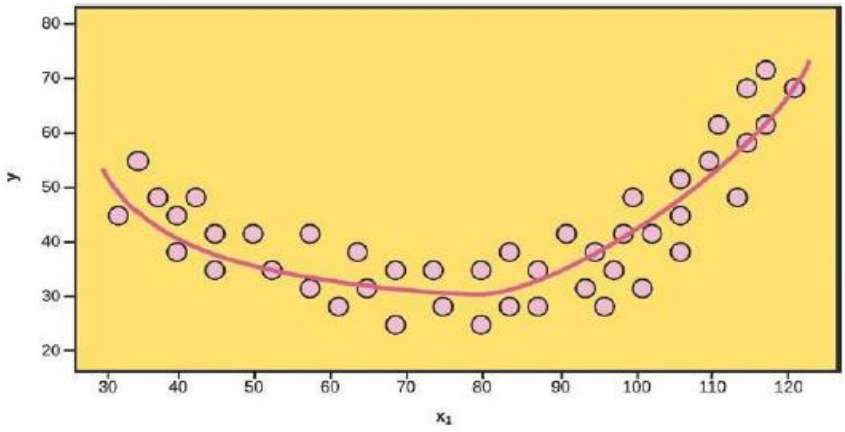
$x_1{}^2$,$x_2{}^2$: añadimos términos polinómicos de orden superior.

y : Variable objetivo

Es relevante destacar que, desde una perspectiva estadística, al buscar los valores óptimos de los pesos para minimizar el modelo, estos mantienen una relación lineal. Los modelos de regresión no lineal suelen ser propensos a sobre ajustar los datos, sin embargo, se puede solucionar este problema mediante la incorporación de técnicas de regularización al modelo. En la figura 6 se presenta una imagen que ilustra cómo funciona la regresión polinómica.

Figura 6

Funcionamiento de la regresión polinómica



Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag. 237.

1.1.4 Método de descenso del gradiente

Se utiliza para llegar a la optimización. El principio de optimización establece que el mínimo de la función de costo se encuentra cuando su derivada es igual a cero.

Función del costo:

$$J(\theta) = \frac{1}{2}\sum(\bar{y} - y)^2 \tag{4}$$

$$\theta = \theta(w) = \bar{y} \tag{5}$$

$$J(\theta) = J\big(\theta(w)\big) = \frac{1}{2}\sum(\bar{y} - wX^T)^2 \tag{6}$$

Para optimizar los valores de w, que se inicializan, se utiliza el método del descenso del gradiente.

$$wX^T = w_0 X_0 + w_1 X_1 + w_2 X_2 + w_3 X_3 + \cdots \tag{7}$$

Y, mediante el método de la cadena:

$$\frac{\partial J}{\partial w} = \frac{\partial J}{\partial \theta} * \frac{\partial \theta}{\partial w} = \sum (y - w_i . X^T) * x_i \quad (8)$$

$$\frac{J(t2) - J(t1)}{\theta(t2) - \theta(t1)} = \quad (9)$$

$$\frac{J(t2) - J(t1)}{\theta(t2) - \theta(t1)} = \sum (y - w_i . X^T) * x_i \quad (10)$$

$$\left(\sum (y - w_i . X^T) * w_i \right) * \left(\theta(t2) - \theta(t1) \right) = J(t2) - J(t1) \quad (11)$$

$$\theta(t2) - \theta(t1) = \frac{J(t2) - J(t1) * (\sum (y - w_i . X^T) * x_i)}{\left((\sum (y - w_i . X^T) * x_i) \right)^2} \quad (12)$$

$$\theta(t2) - \theta(t1) = \alpha * \sum (y - w_i . X^T) * x_i \quad (13)$$

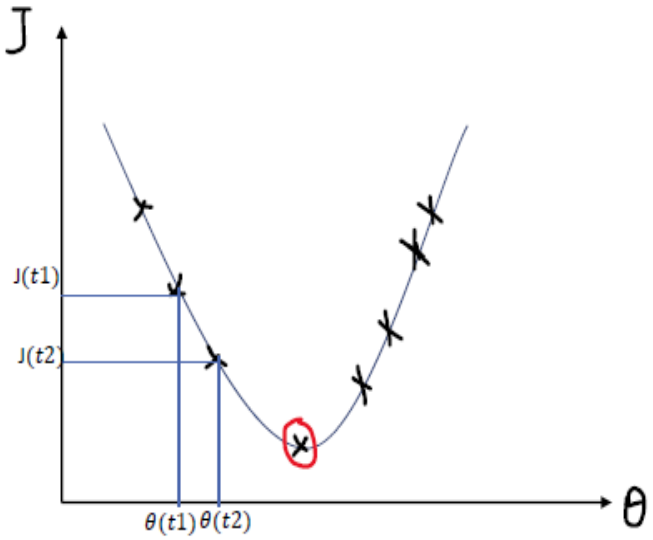
Finalmente:

$$\theta(t2) = \theta(t1) \pm \alpha * \sum (y - w_i . X^T) * x_i \quad (14)$$

Visto gráficamente en figura 7.

Figura 7

Representación Gráfica del Descenso de Gradiente



El procedimiento por seguir es:

Inicializar los pesos "W" de manera aleatoria

Para un número $\boldsymbol{i}$ de iteraciones

Hallar los $\bar{y}$ estimados

Hallar la función de costo

Optimizar los valores de "W"

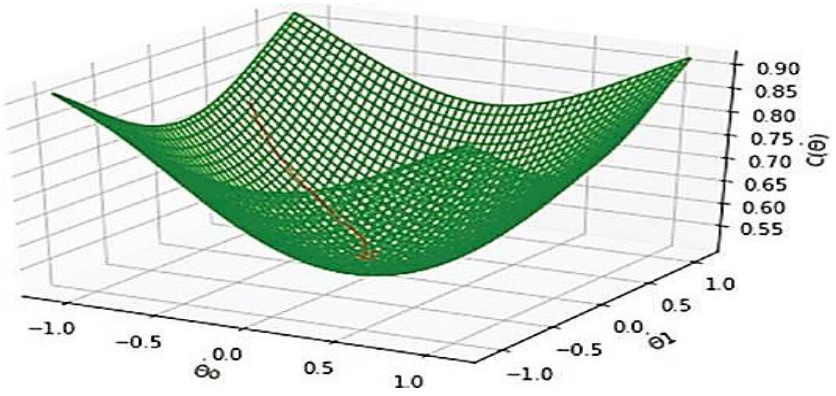
Repetir hasta que la función de costo sea muy pequeña (0.001) o un número de iteraciones adecuada como 1000 (depende del costo computacional).

El descenso del gradiente es un método de optimización utilizado para reducir la función de costo en un algoritmo de aprendizaje automático. Este algoritmo se considera iterativo porque, a través de un proceso de bucle, busca una solución aproximada mediante la actualización de los pasos siguientes en función del actual, hasta que se cumpla una condición que termine el proceso.

En la figura 8 se presenta una función convexa para ilustrar cómo el descenso del gradiente encuentra el punto mínimo en un espacio de funciones.

Figura 8

Figura de contorno - Descenso por Gradiente



Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 203.

(Bisong, Bisong, E. "Building Machine Learning and Deep Learnig Models on Google Cloud Platform", 2019, pág. 203).

Este es un espacio de funciones que se conoce como función convexa. El objetivo del descenso del gradiente en el espacio de funciones es encontrar los valores de los parámetros de la función que minimizan el coste de la función y lo llevan al mínimo global. Este hace referencia al punto más bajo del espacio de funciones. Por ejemplo, la función de coste del error cuadrático medio para la regresión lineal es muy convexa, lo que significa que el descenso del gradiente tiene una alta probabilidad de encontrar el mínimo global. Sin embargo, esto no es siempre cierto para otros tipos de espacios de funciones no convexos. Es importante recordar que el descenso del gradiente es un optimizador global para minimizar cualquier espacio de funciones. Algunas funciones pueden tener múltiples regiones mínimas, que se conocen como mínimos locales, mientras que la región más baja del espacio de funciones se llama mínimo global. Podemos citar el ejemplo de los datos de un motor de avión en varios de sus puntos de funcionamiento.

Los datos han sido extraídos de repositorio de (Kaggle).

https://www.kaggle.com/datasets/nicolascaparroz/turbofan-hpc-efficiency.

El objetivo es encontrar los coeficientes "*w*".

Tabla 1

Datos del turbo ventilador

Empuje Neto (KN)	Empuje baja presión (KN)	Empuje aire frío (KN)	Consumo combustible (g/Kn*s)	Eficiencia HPC
38.46725541	11.2175	102.952769	14.34954	0.861717
44.2178558	13.5866	108.4495	15.27196	0.789251
38.4130891	11.196753	102.89731	14.34206	0.862491
34.686831	9.7112906	99.055475	17.875012	0.911628
38.788102	11.340699	103.2807892	14.394312	0.857155

Para la primera iteración

*w*0 = 0; *w*1 =0; *w*2 = 0, *w*3 = 0, *w*4 = 0,

Se obtienen los siguientes resultados:

Tabla 2

Resultados de la Primera Iteración

y_0	y_1	y_2	y_3	y_4
-533.6	-569.2	-533.33	-511.7	-535.7
w_0	w_1	w_2	w_3	w_4
-0.042	-1.663	-0.483	-4.41	-0.65

Tabla 3

Resultados de la Segunda Iteración

y_0	y_1	y_2	y_3	y_4
-336474.3	-358931.1	-336253.5	-322630.7	-337781.2
w_0	w_1	w_2	w_3	w_4
-336474.3	-358931.1	-336253.5	-322630.7	-337781.2

Tabla 4

Resultados de la Tercera Iteración

y_0	y_1	y_2	y_3	y_4
-16947.7	-661291.0	-194107.2	-1752900.3	-258026.2
w_0	w_1	w_2	w_3	w_4
-211800886.9	-225936796.3	-211661895.9	-203086656.5	-212623512.7

Tabla 5

Resultados de la Cuarta Iteración

y_0	y_1	y_2	y_3	y_4
-10668045.2	-416262510.6	-122184588.6	-1103397199.9	-162419589.1
w_0	w_1	w_2	w_3	w_4
-133322193403	-142220316939	-133234702832	-127836851299	-133840011250

El método utilizado es correcto sin embargo para el ejemplo propuesto no hay una convergencia debido a que hemos tomado sólo algunas líneas del conjunto de datos para entrenar el modelo.

1.2 Ajustes, Procesamiento y Aprendizaje Profundo

En el campo del aprendizaje automático y la inteligencia artificial, existen varias herramientas y bibliotecas de software para el ajuste, procesamiento y aprendizaje profundo.

1.2.1 Aprendizaje Profundo

(Bisong, Bisong, E. "Building Machine Learning and Deep Learnig Models on Google Cloud Platform", 2019, pág. 327...343)

El aprendizaje profundo es capaz de manejar grandes cantidades de datos no estructurados, como imágenes, voz y texto, y extraer patrones y características importantes de ellos para realizar tareas específicas. Esto ha permitido avances significativos en aplicaciones como la visión por ordenador, el reconocimiento de voz, la traducción automática y la robótica, entre otras.

Un ejemplo popular de aprendizaje profundo es el desarrollo de coches autónomos. Los algoritmos de aprendizaje profundo son capaces de procesar y analizar grandes cantidades de datos de sensores de un coche, como cámaras y radares, para tomar decisiones de conducción en tiempo real. La traducción automática de voz también es un ejemplo de cómo el aprendizaje profundo ha mejorado significativamente la capacidad para comunicarnos en diferentes idiomas.

1.2.2 El Reto de la Representación

La tarea de aprender no es sencilla y los investigadores de diferentes áreas relacionadas con el cerebro aún no comprenden completamente cómo funciona la capacidad de aprendizaje del cerebro humano. Lo que a nosotros nos parece fácil y natural, en realidad es el resultado de procesos complejos e intrincados que nos hacen ser seres inteligentes y diferentes de otras formas de vida.

Algunos ejemplos de tareas complejas que el cerebro humano puede realizar son reconocer rostros en una fracción de segundo, aprender y comprender lenguaje profundo, así como crear y entender obras musicales. Estos son solo algunos ejemplos del asombroso potencial de la inteligencia natural.

El desafío para la investigación y la ingeniería en IA es crear máquinas capaces de entender y descomponer patrones estructurales complejos, imitando así la inteligencia natural. El aprendizaje profundo, una técnica de IA, aborda este problema aprendiendo la estructura subyacente inherente a un conjunto de datos, lo que se conoce como aprendizaje de representación.

1.2.3 Inspiración desde el cerebro

Los científicos a menudo se inspiran en la naturaleza para lograr proezas increíbles, como el avión que se inspiró en los pájaros. En este sentido, no hay mejor modelo de inteligencia para estudiar que el cerebro humano. Podemos concebir el cerebro como una sociedad de agentes inteligentes interconectados que se comunican entre sí mediante señales eléctricas. Estos agentes se denominan neuronas y son la pieza clave del sistema nervioso. Nuestro interés aquí es comprender las neuronas, sus componentes y cómo transmiten información para crear inteligencia.

Las neuronas son agentes autónomos que reciben y transmiten información a otras células del cuerpo en respuesta a estímulos externos e internos. Esto se logra mediante impulsos eléctricos generados en la fuente del estímulo, que se propagan hacia el cerebro y otras células para generar una respuesta adecuada. El complejo y coordinado funcionamiento de las neuronas es fundamental para la inteligencia humana.

A continuación, se detallan los tres componentes esenciales de las neuronas que son de mayor interés para nosotros:

- El axón, es la parte larga y delgada de una neurona que lleva las señales eléctricas de la célula hacia otras neuronas o hacia los músculos y las glándulas.

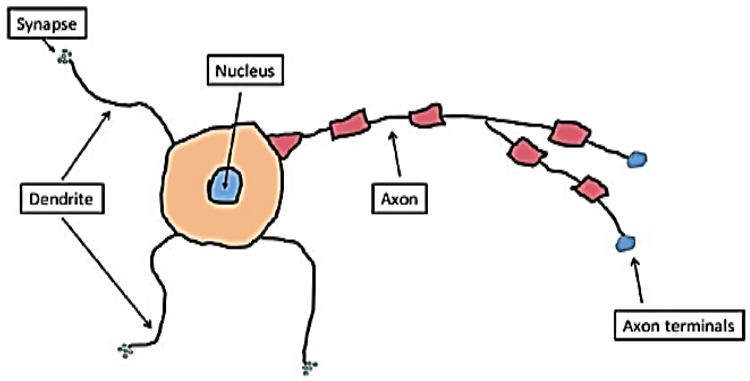
- La dendrita, es una estructura ramificada corta que se extiende desde el cuerpo celular de una neurona y recibe señales de otras neuronas o de células sensoriales.

- La sinapsis, es el punto de comunicación entre dos neuronas o entre una neurona y una célula efectora (como un músculo o una glándula).

La figura 9 muestra el axón, con su prolongación alargada conectada al núcleo de la neurona, que tiene la importante función de enviar señales eléctricas a otras células neuronales mediante los terminales axónicos. Por otro lado, la dendrita es responsable de recibir impulsos eléctricos de otras células neuronales a través de las sinapsis y enviarlos al núcleo de una célula neuronal.

Figura 9

Representación de una Neurona



Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 329.

Hasta el momento se ha logrado desarrollar una red neuronal artificial (RNA) que imita los tres componentes esenciales de una neurona biológica (dendrita, núcleo y axón), y si ha sido posible imitar la capacidad del cerebro para aprender desde una perspectiva científica e ingenieril, se podrían construir máquinas que puedan aprender características complejas y jerárquicas de diferentes ámbitos de uso.

1.2.4 Fundamento de las redes neuronales

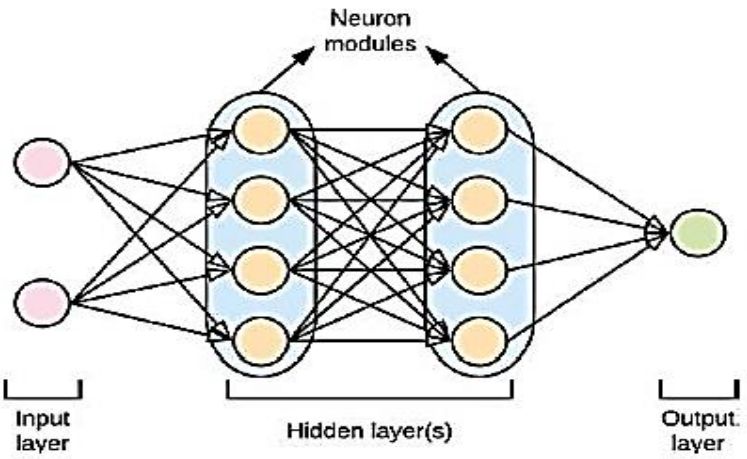
La red neuronal artificial (RNA) se basa en la estructura de la neurona biológica para crear una red de agentes conexionistas que aprenden y comparten información. Durante el proceso de transferencia de datos entre neuronas artificiales, se aprende una jerarquía de representaciones o características, lo que se conoce como aprendizaje profundo de representaciones o simplemente Aprendizaje Profundo.

Una red neuronal artificial se compone de (ver figura 10):

- Una capa de entrada
- Capa(s) oculta(s)
- Una capa de salida

Figura 10

Arquitectura de la Red Neuronal



Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 329.

La capa inicial de la red neuronal artificial (RNA) recibe la información de las características del conjunto de datos y realiza cálculos para capturar patrones de los datos. Luego, la información se transfiere a través de las capas ocultas, donde se produce el aprendizaje profundo. En las capas ocultas, hay múltiples módulos neuronales que aprenden conjuntos cada vez más sofisticados de representaciones de características. La decisión de la cantidad de neuronas en cada capa y el número de capas ocultas que forman la topología de la red es un proceso de diseño importante durante el entrenamiento de la RNA.

Las arquitecturas Perceptrón Multicapa (MLP) y Redes Neuronales Convolucionales (CNN) son dos tipos de redes neuronales utilizadas en el aprendizaje profundo.

Un MLP es una red neuronal de alimentación directa en la que las neuronas están organizadas en capas, con cada capa conectada a la siguiente capa. Cada neurona en una capa está conectada a todas las neuronas de la capa siguiente mediante un conjunto de pesos. Los datos se alimentan en la primera capa, y los resultados se obtienen en la última capa. Los MLP son utilizados en tareas de clasificación y regresión, y son altamente personalizables en términos de número de capas, número de neuronas por capa y tipo de función de activación utilizada.

Además, las CNN son una variante de las redes neuronales que se utilizan principalmente para procesamiento de imágenes y señales. Las CNN utilizan una técnica llamada convolución para extraer características importantes de las imágenes. Las capas convolucionales se encargan de detectar patrones en las imágenes, como bordes y texturas, mientras que las capas completamente conectadas al final de la red se encargan de clasificar las imágenes en diferentes categorías. La arquitectura de la CNN está diseñada para ser capaz de manejar datos de entrada de alta dimensionalidad, como imágenes.

Las MLP son redes neuronales utilizadas en tareas de clasificación y regresión, mientras que las CNN son redes neuronales diseñadas específicamente para el procesamiento de imágenes y señales.

Una ANN (Artificial Neural Network), es un modelo computacional inspirado en el funcionamiento del cerebro humano, que permite a las máquinas aprender a partir de datos y reconocer patrones complejos en ellos. Una ANN está compuesta por múltiples capas de neuronas artificiales interconectadas que procesan información y generan una salida. Cada capa procesa la información de la capa anterior y envía una salida a la siguiente capa. Las ANN se utilizan en una amplia gama de aplicaciones, incluyendo reconocimiento de voz e imagen, análisis de texto y predicción de series de tiempo, entre otros.

1.2.5 Entrenamiento de una red neuronal

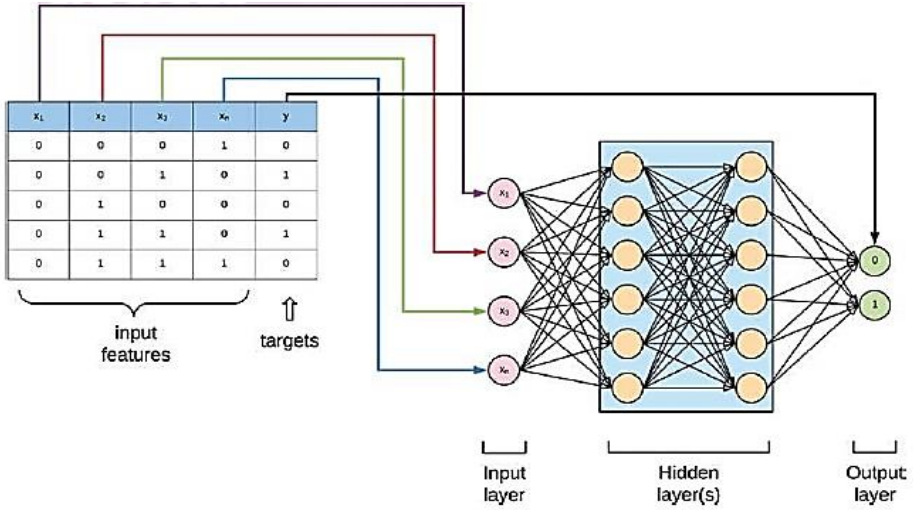
En el entrenamiento de redes neuronales profundas, se puede observar el flujo de información aprendida, la función de costo en la capa de salida y cómo se utilizan la codificación one-hot y la función de activación softmax para determinar la pertenencia a una clase en un problema de clasificación. Además, se puede entender cómo el algoritmo de retro propagación mejora los parámetros de la red y qué funciones de activación permiten a la red aprender patrones no lineales. Estas técnicas son esenciales para comprender el proceso de entrenamiento de una red neuronal y mejorar su rendimiento en tareas complejas.

Para examinar los métodos de entrenamiento de una red neuronal, tomaremos como ejemplo un problema de clasificación que tiene dos posibles salidas. Cuando se diseña una red neuronal, el número de neuronas en la capa de entrada suele ser igual al número de características presentes en el conjunto de datos, mientras que el número de neuronas en la capa de salida es igual al número de clases de la variable objetivo que se está tratando de clasificar con la red neuronal.

La figura 11 ilustra cómo las características del conjunto de datos se convierten en las entradas de la red neuronal, y cómo el número de neuronas en la capa de salida se determina por el número de clases en la variable objetivo. En este ejemplo, la red neuronal aprende dos clases, representadas por los valores 0 y 1.

Figura 11

Definición de una Red Neuronal a Partir de un Conjunto de Datos

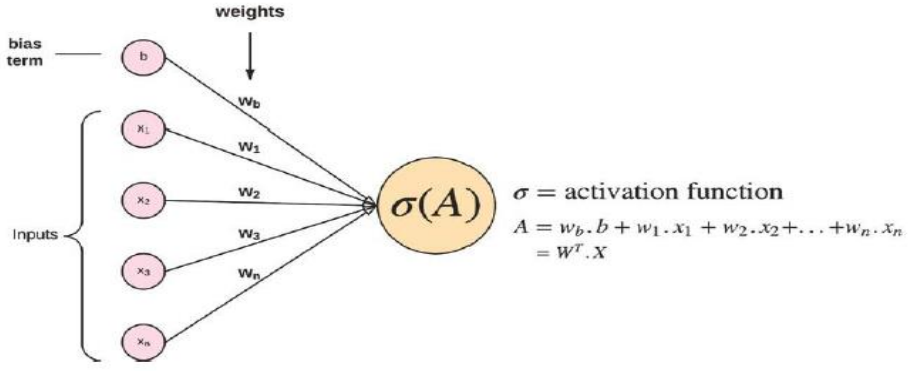


Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 334.

A cada neurona de la red neuronal se le asigna un peso, que también se conoce como parámetro. Los pesos de una capa de neuronas se multiplican por sus entradas y se procesan a través de una función de activación. Las salidas de esta función de activación se convierten en las entradas para las neuronas de la siguiente capa de la red neuronal, tal y como se ilustra en la figura 12. Este proceso se repite mientras la información se mueve de una capa de la red neuronal a la siguiente. Cada capa de neuronas también incluye una neurona de sesgo (que normalmente se establece en 1) que controla la suma ponderada.

Figura 12

La Información fluye de una capa neuronal a una neurona de la capa siguiente



Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 335.

Los pesos se inicializan como valores aleatorios y se ajustan posteriormente mediante el algoritmo de retropropagación. Las salidas de las neuronas se determinan mediante una

función de activación no lineal aplicada a la suma del peso multiplicado por las salidas más el término de sesgo de las neuronas de la capa anterior. Este proceso se llama algoritmo de aprendizaje feedforward. Para evaluar el rendimiento de la red, se utiliza una función de costo que captura la calidad de la predicción realizada por la red. El objetivo es minimizar esta función de costo, utilizando funciones como la de error cuadrático medio para problemas de regresión y la de "costo de entropía cruzada" o medida del error en problemas de clasificación softmax. Si la red produce una clasificación incorrecta, se ajustan los errores mediante el algoritmo de retropropagación.

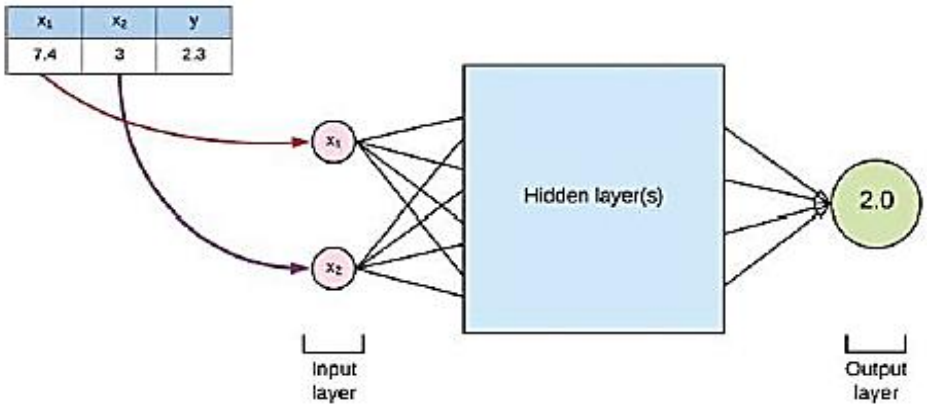
1.2.6 Función de Costo o Función de Pérdida

Cuando la red neuronal ha hecho predicciones y ha producido salidas, se evalúa la calidad de estas salidas utilizando una función de costo específica, que puede ser diferente dependiendo del problema que se esté abordando. Si el valor de la función de costo es pequeño, esto significa que la red ha hecho buenas predicciones.

En la Figura 13, se muestra un ejemplo en el que la red debe producir una salida de 2.3. Para evaluar la calidad de la salida producida por la red, se utiliza la función de error cuadrático medio, que calcula el promedio de los errores cuadrados para todas las muestras de datos en el conjunto de entrenamiento. En este ejemplo, se utiliza una sola muestra de datos para ilustrar cómo funciona la función de costo.

Figura 13

Estimación MSE de la Red Neuronal



Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 336.

$$J=\frac{1}{n}\sum_{i=1}^{n}(\bar{y}-y)^2 \tag{15}$$

$$J=(2.0-2.3)^2 \tag{16}$$

$$J=(-0.3)^2 \tag{17}$$

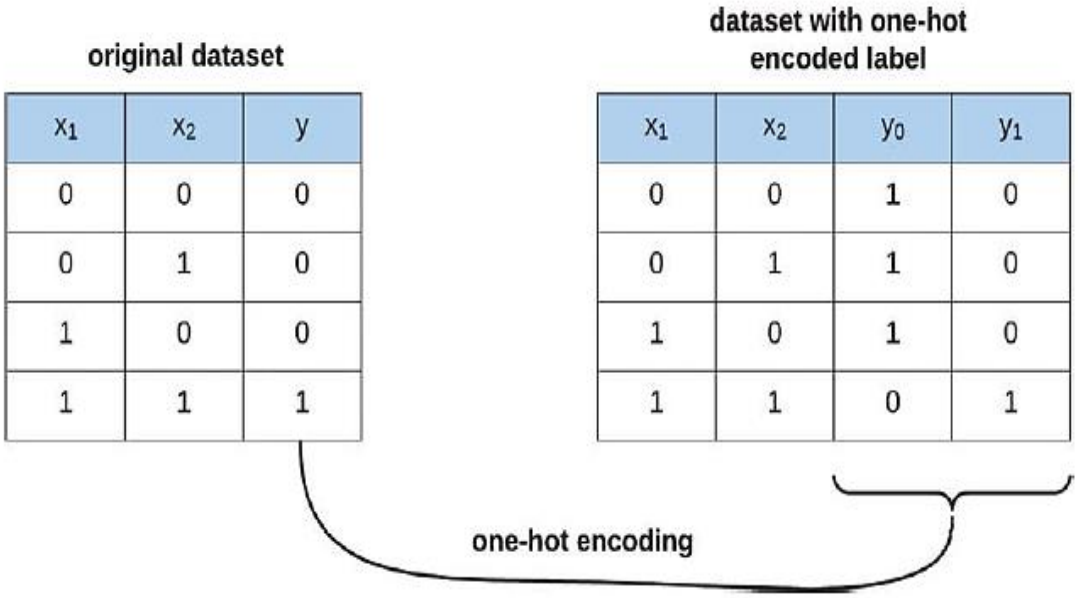
$$J=0.09 \tag{18}$$

1.2.7 Codificación One Hot

La codificación *one-hot* es un proceso de transformación de etiquetas de clase en una matriz de variables binarias utilizada en problemas de clasificación. Asigna un valor de 1 cuando la salida pertenece a una clase determinada y un valor de 0 en caso contrario (ver figura 14). Esta técnica se utiliza para convertir las etiquetas de clase en un formato que puede ser procesado por una red neuronal y es comúnmente utilizada en la capa de salida de una red neuronal junto con la función de activación softmax para determinar la pertenencia a una clase.

Figura 14

Codificación en Caliente (One Hot)



x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

x_1	x_2	y_0	y_1
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 337.

El objetivo de la codificación one-hot es transformar las etiquetas de clase de la variable objetivo en una matriz de variables binarias, de forma que se pueda representar la salida como un vector de clases distintas con la probabilidad de que un ejemplo del conjunto de datos de entrenamiento pertenezca a cualquiera de las categorías de salida. Esto facilita el proceso de entrenamiento de la red neuronal y su capacidad para clasificar correctamente los ejemplos del conjunto de datos.

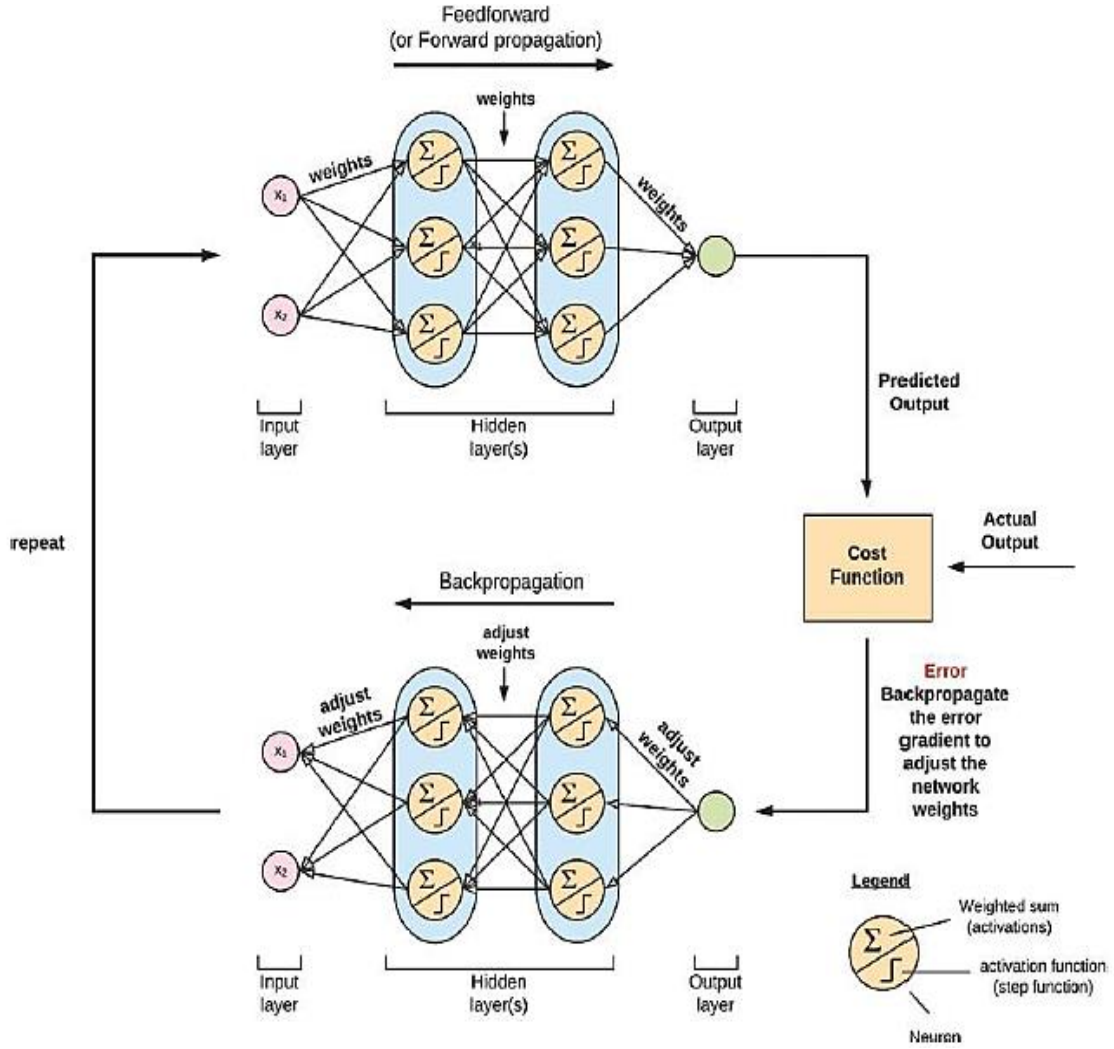
1.2.8 Algoritmo de retropropagación

En la retropropagación, se entrena la red neuronal para mejorar la precisión de sus predicciones. Para ello, se ajustan los pesos de la red, lo que afecta el valor de las activaciones de las neuronas y la capa de salida. En el proceso de retropropagación, se ajustan los pesos de

las capas previas y se vuelve a realizar el algoritmo feedforward hasta minimizar el error en la capa de salida. (véase la Figura 15).

Figura 15

Retro propagación



Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 337.

El algoritmo calcula la función de costo comparando la salida prevista de la red neuronal con las salidas reales del conjunto de datos, y utiliza el descenso del gradiente para calcular la función de costo utilizando los pesos de las neuronas en cada capa sucesiva y así actualizar los pesos que se propagan a través de la red.

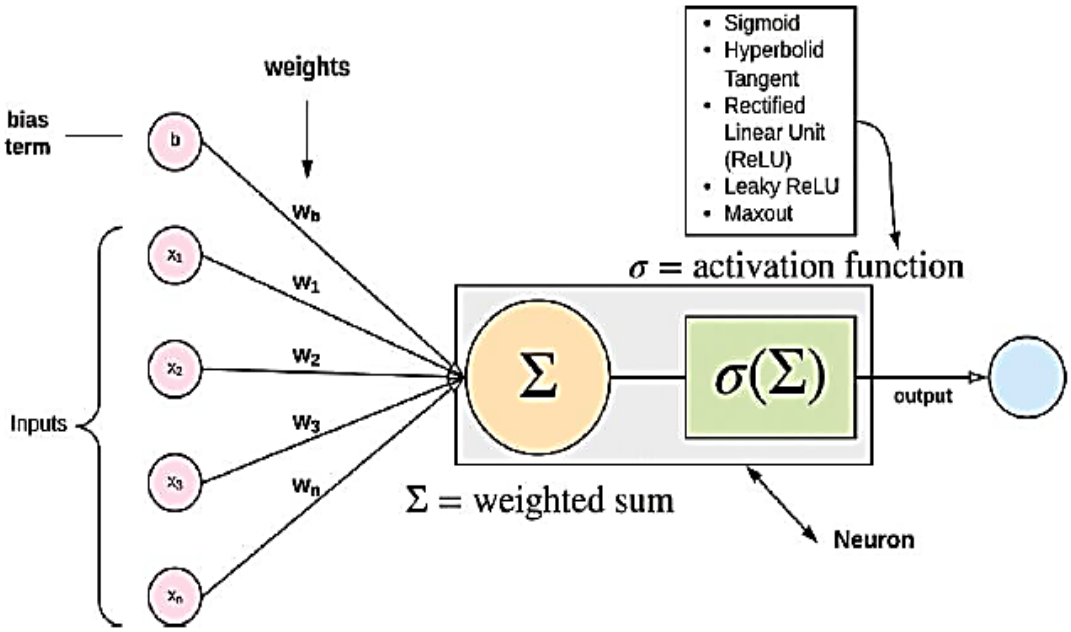
1.2.9 Funciones de activación

Las funciones de activación deciden si una neurona debe propagar su información o no a las capas neuronales siguientes, al actuar sobre la suma ponderada en la neurona a través de una función no lineal. Estas funciones son análogas a la forma en que las neuronas se comunican en el cerebro y se denominan no lineales porque inyectan capacidades no lineales

a la red. El objetivo es aprender un mapeo de entradas a salidas para un conjunto de datos cuya estructura fundamental es no lineal. La figura 16 muestra cómo se pasa la suma ponderada de pesos y sesgos a través de una función de activación.

Figura 16

Función de activación



Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 339.

El siguiente es un listado de algunas funciones de activación utilizadas por la red neuronal:

Sigmoide

Tangente Hiperbólica

Unidad lineal rectificada (ReLU)

Fugas ReLU

Maxout

1.2.9.1 Sigmoide. La figura 17 muestra una función sigmoide, que es una función matemática no lineal utilizada en redes neuronales. La función sigmoide es una función suave y curvada que toma un valor de entrada y lo transforma en un valor entre 0 y 1.

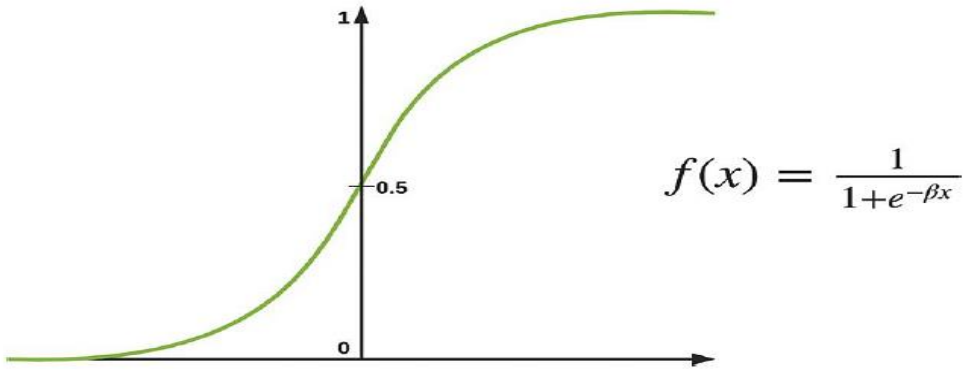
La función sigmoide se utiliza comúnmente en las capas ocultas de las redes neuronales para comprimir las entradas y producir una salida que se encuentra dentro del rango de 0 y 1. Esto es útil porque las redes neuronales a menudo necesitan tomar decisiones binarias (sí o no) basadas en la salida de una neurona, y la función sigmoide proporciona una forma de hacerlo.

La figura 17 muestra que la función sigmoide es casi lineal cerca del origen, pero se aplana a medida que la entrada se vuelve más positiva o negativa. Cuando la entrada es grande y positiva, la función se acerca a 1, y cuando la entrada es grande y negativa, la función se acerca a 0. Esto significa que los números grandes positivos se transforman en un valor cercano a 1 y los números grandes negativos se transforman en un valor cercano a 0.

Además, la figura 17 también muestra que la función sigmoide tiene un umbral en 0,5. Esto significa que, si la salida de la función sigmoide es mayor que 0,5, se considera que la neurona ha "disparado" o activado. Por lo tanto, el umbral de 0,5 se utiliza a menudo para determinar si una neurona ha producido una salida útil o no.

Figura 17

Función de Activación Sigmoidea



Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 340.

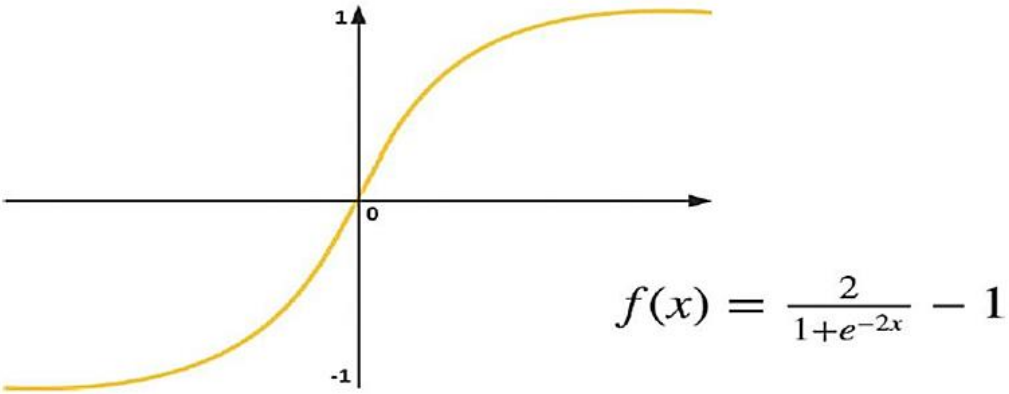
Un inconveniente de esta función es su susceptibilidad a los gradientes explosivos y evanescentes. Si los gradientes se vuelven demasiado grandes durante el proceso de entrenamiento, el modelo entregará resultados cada vez más distantes de la realidad; y si los gradientes son evanescentes, es decir, se vuelven muy pequeños, se desvanecen o disminuyen a medida que avanzan hacia las capas anteriores de la red neuronal perdiéndose la retroalimentación sin llegar a entrenar correctamente el modelo.

Otro inconveniente es que las salidas de la función no están centradas en cero y como consecuencia, los gradientes pueden llegar a ser todos positivos o todos negativos, teniendo esto un efecto contrario a la hora de minimizar el objetivo de la función costo.

1.2.9.2 Tangente Hiperbólica (Tanh). La tangente hiperbólica de la figura 18, es una función de activación que mejora la función sigmoide al llevar sus salidas dentro de un rango de -1 a 1, lo que permite que sus salidas estén centradas en cero. Aunque todavía presenta el problema de gradientes explosivos y evanescentes, su comportamiento es similar al de la función sigmoide a escala.

Figura 18

Función de Activación Tangente Hiperbólica

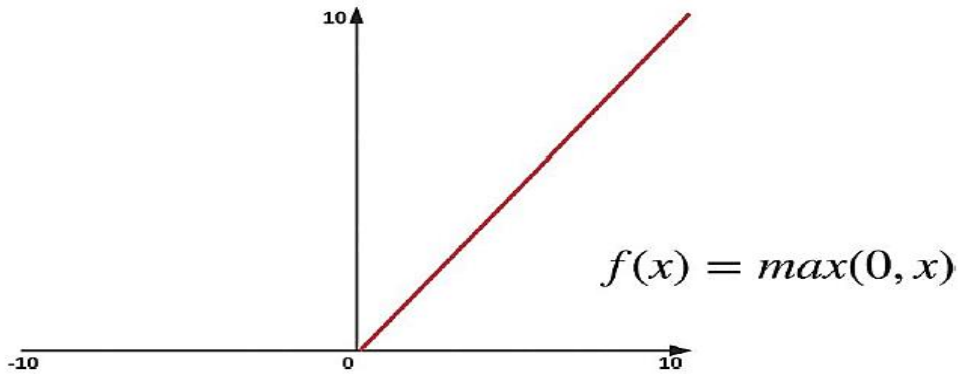


Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 341.

1.2.9.3 Unidad Lineal Rectificada (ReLU). Es una función de activación que se utiliza comúnmente en redes neuronales profundas. La función ReLU establece la activación en 0 para todos los valores de entrada x que son menores que 0, y tiene una pendiente lineal de 1 para todos los valores de entrada x que son mayores que 0. Esto significa que la función es lineal para todos los valores positivos de entrada x, lo que simplifica la derivación y el cálculo del gradiente. La figura 19 ilustra la función ReLU, donde se muestra cómo la activación se establece en 0 para valores de entrada negativos y tiene una pendiente lineal de 1 para valores de entrada positivos. Esta función de activación ha demostrado ser muy efectiva en la práctica y se utiliza en muchas aplicaciones de redes neuronales profundas.

Figura 19

Función de Activación ReLU



Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 342.

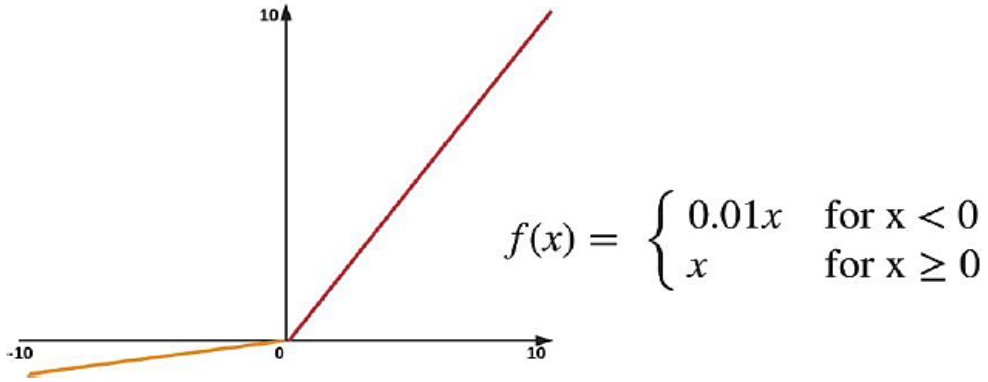
ReLU mejora su desempeño sobre las funciones de activación Tanh y sigmoide al mitigar el problema del gradiente evanescente y explosivo. Sin embargo, algunos gradientes

pueden morir durante la retropropagación con una tasa de aprendizaje alta. Sin embargo, con una tasa de aprendizaje controlada, se superaría estos problemas.

1.2.9.4 Leaky ReLU. Es otra función que propone resolver el caso de algunas neuronas que mueren completamente en ReLU. En la figura 20 se ha añadido una pequeña pendiente negativa a la función cuando x < 0.

Figura 20

Función de Activación Leaky ReLU



Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 343.

1.2.9.5 Maxout. Esta función combina las funciones ReLU y Leaky ReLU, aprovechando la eficacia de ReLU, al mismo tiempo que evita el inconveniente de que algunas neuronas mueran. No obstante, siempre hay que hacer una compensación, porque Maxout aumenta el tamaño de los parámetros de cada neurona durante el entrenamiento.

Como regla general, no se mezclan distintos tipos de funciones de activación en la misma red. Además, se suele utilizar ReLU para las capas ocultas, y la activación softmax para los problemas de clasificación en la capa de salida, ya que esta última devuelve la probabilidad de pertenencia a una clase concreta.

1.3 Análisis de Series Temporales

Hasta ahora, hemos aprendido modelos matemáticos para cualquier conjunto de datos linealmente independientes usando modelos de máquinas de aprendizaje tradicionales. Sin embargo, es importante tener en cuenta que todos estos experimentos ocurren en diferentes instantes de tiempo, y el tiempo es un espacio que contiene cualquier variable analizada. Los seres humanos también se encuentran dentro del tiempo, lo que significa que no se puede usar una red neuronal tradicional para incluir el tiempo como si fuera un parámetro más. Para resolver este problema, surge el análisis de datos de series temporales. Este enfoque implica la aplicación de técnicas de aprendizaje automático específicas para

datos que evolucionan en el tiempo, como los datos financieros, las señales de audio y las señales biomédicas, entre otros.

1.3.1 Redes Neuronales Recurrentes (RNN)

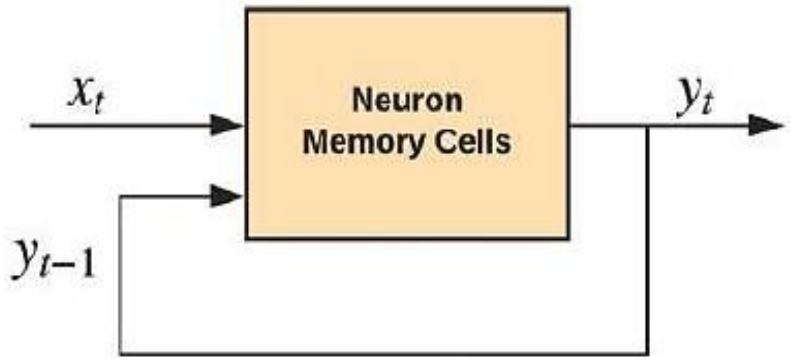
(Bisong, Bisong, E. "Building Machine Learning and Deep Learnig Models on Google Cloud Platform", 2019, pág. 443...468)

Las redes neuronales recurrentes (RNN) son una arquitectura especializada en resolver problemas de aprendizaje en los que la información del pasado está directamente relacionada con la predicción del futuro. Estos problemas secuenciales son comunes en tareas del mundo real como el modelado del lenguaje o la predicción bursátil. Las RNN son ideales para procesar datos secuenciales y su marco de retroalimentación permite incorporar información del pasado para hacer predicciones. A diferencia de las redes neuronales convolucionales, que procesan datos en forma de cuadrícula, las RNN trabajan con datos 1-D. La variante más avanzada de las RNN es la memoria a corto plazo Long Short Term - Memory (LSTM), que se utiliza para resolver problemas secuenciales como la subtitulación de imágenes, la predicción bursátil, la traducción automática y la clasificación de textos, sin embargo, también son de utilidad en el presente estudio.

1.3.1.1 La Neurona Recurrente. El primer bloque de construcción de la RNN es la neurona recurrente indicada en la figura 21. A diferencia de otras, la neurona recurrente mantiene una memoria o un estado de cálculos anteriores y lo hace tomando como entrada la salida del instante anterior y_{t-1} además de su entrada actual en un instante particular x_t .

Figura 21

Neurona Recurrente



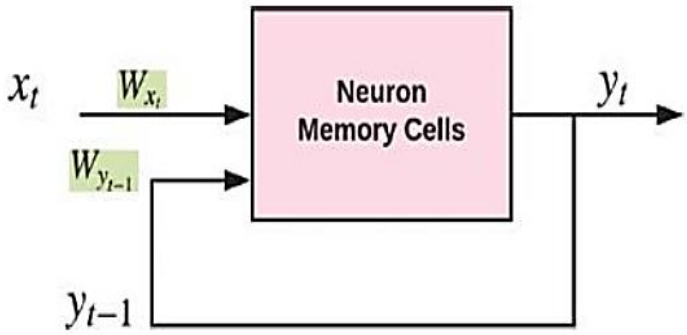
Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 444.

En la Figura 20, la neurona recurrente contrasta con las neuronas de las arquitecturas Perceptrón Multicapa (MLP) y Redes Neuronales Convolucionales (CNN) porque, en lugar de transferir una jerarquía de información a través de la red de una neurona a otra, los datos se retroalimentan a la misma neurona en cada instante de tiempo.

Por lo tanto, la neurona recurrente tiene dos pesos de entrada, w_x y w_y , para la entrada en tiempo xt y para la entrada en el instante de tiempo y_{t-1}. Véase la figura 22.

Figura 22

Neurona Recurrente con Pesos de entrada.



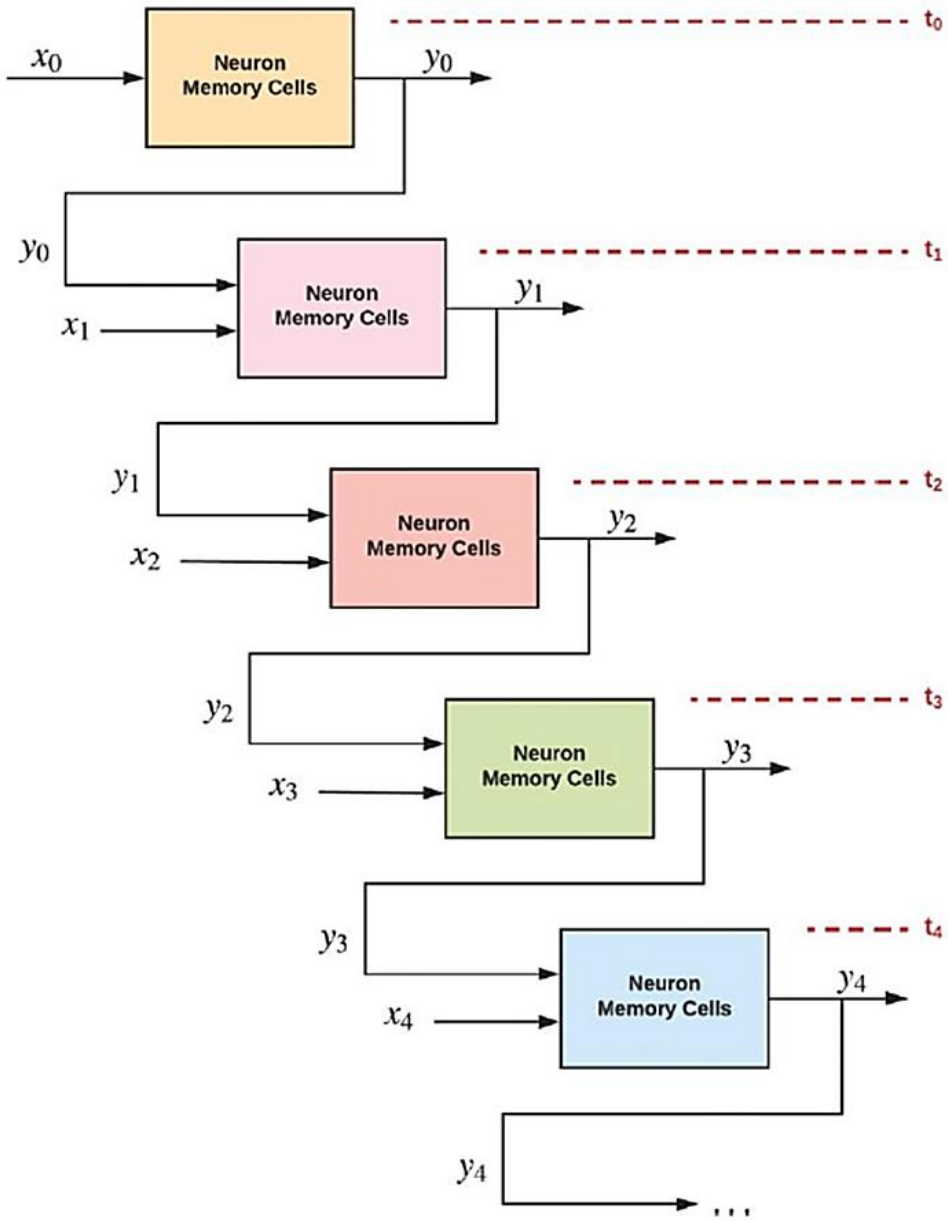
Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 444.

Al igual que otras neuronas, la RNN también inyecta no linealidad en la red haciendo pasar sus sumas ponderadas o transformaciones afines a través de una función de activación no lineal.

1.3.1.2 Despliegue del gráfico computacional recurrente. El flujo de información a través de la capa recurrente en cada instante se muestra en la figura 23. Imaginemos que disponemos de una serie de datos temporal compuesta por cinco pasos. En ese caso, replicamos la neurona recurrente cinco veces en cada uno de los instantes temporales. Cada una de estas réplicas se considera como una capa de la arquitectura de la red neuronal recurren.

Figura 23

Despliegue de la Neurona Recurrente en una Red Neuronal Recurrente



Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 446.

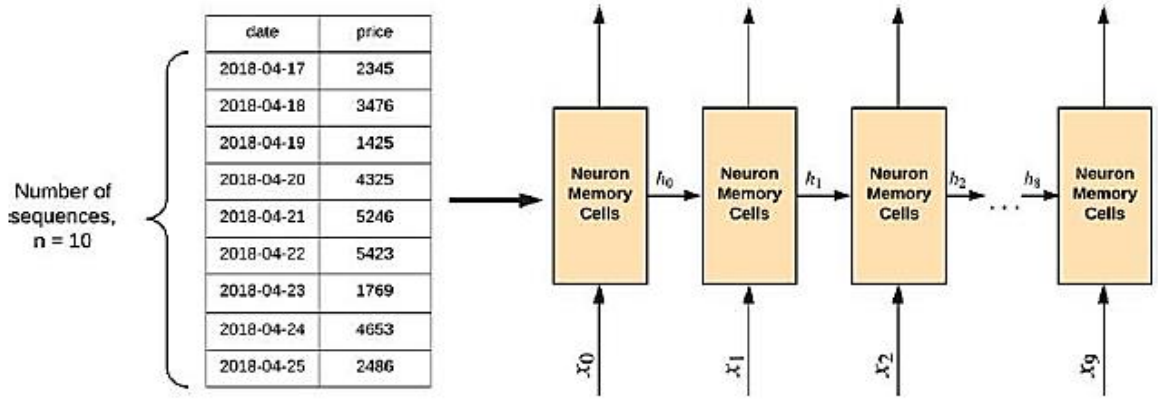
La arquitectura de la neurona recurrente permite que la red neuronal aprenda de eventos o secuencias pasadas mediante la entrada en la capa recurrente de la salida del paso de tiempo anterior y la entrada actual en el paso de tiempo actual. La neurona recurrente almacena memoria o estado en su celda de memoria para capturar información del pasado. La célula de memoria puede ser más complicada en variantes como GRU o LSTM, donde la salida en el instante de tiempo anterior contiene la memoria.

1.3.1.3 Red neuronal recurrente básica. Habíamos visto como al desplegar una red neuronal recurrente podemos observar cómo la información fluye de una capa recurrente a

otra, y que el número de capas recurrentes está determinado por la longitud de la secuencia en el conjunto de datos. La figura 24 ilustra este punto de manera breve. Si tenemos un conjunto de datos de series temporales con diez pasos, cada fila en el conjunto de datos se traducirá en diez capas en el sistema de red recurrente.

Figura 24

Conjunto de Datos a Capas

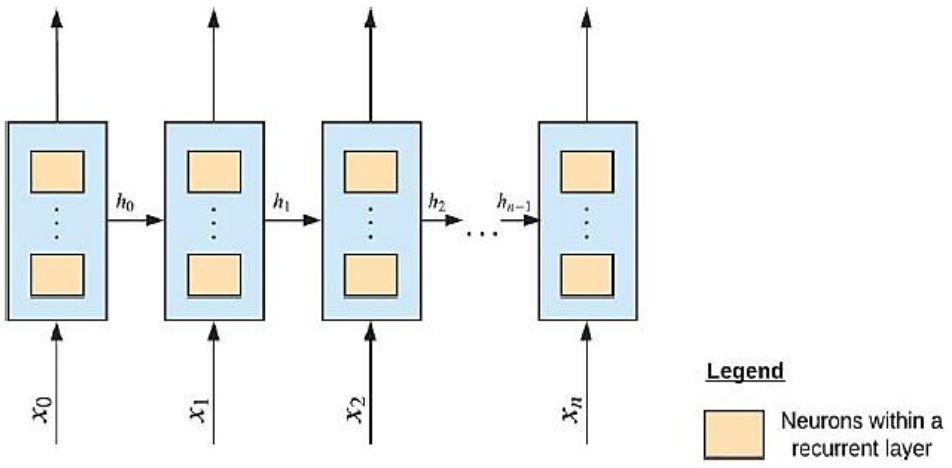


Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 447.

Es importante destacar que la capa recurrente está formada por múltiples células neuronales y no solo una, como se muestra en la figura 25. La cantidad de neuronas que se eligen para la capa recurrente es una decisión que se debe tomar al diseñar la arquitectura de la red.

Figura 25

Neuronas en una Capa Recurrente

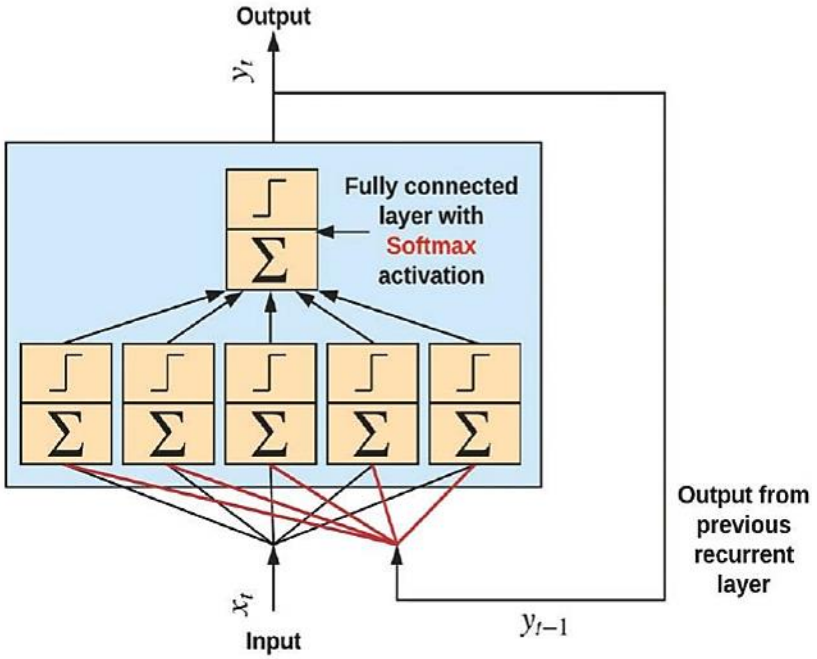


Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 449.

Cada célula neuronal en una capa recurrente recibe la salida de la capa anterior y la entrada actual como entrada y tiene dos vectores de peso. Realiza una transformación afín de las entradas y las pasa a través de una función de activación no lineal, como la tangente hiperbólica (Tanh). Luego, la salida de las neuronas se transfiere a una capa densa o totalmente conectada con una función de activación softmax para generar las probabilidades de clase. Esto se representa en la figura 26.

Figura 26

Cálculos dentro de una Capa Recurrente

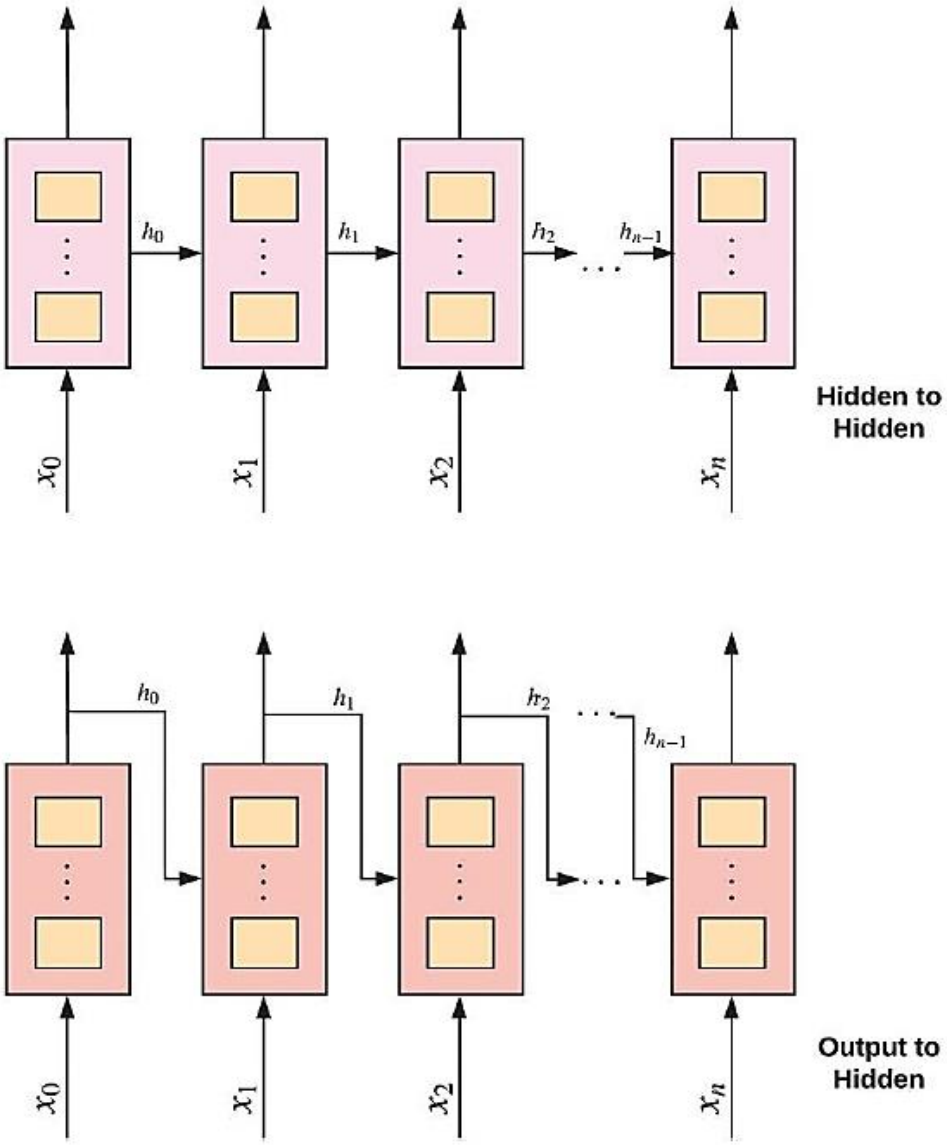


Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 450.

Hay dos formas principales de crear conexiones recurrentes entre capas recurrentes. Una de ellas es mediante la conexión recurrente entre las unidades ocultas, mientras que la otra es mediante la conexión recurrente entre la unidad oculta y la salida de la capa anterior. Estas dos opciones se pueden ver representadas visualmente en la figura 27.

Figura 27

Esquemas de Conexión Recurrentes



Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 451.

La forma en que se establecen conexiones recurrentes entre unidades ocultas es considerada superior a la forma que conecta la salida de la capa anterior con la unidad oculta, ya que la primera es capaz de capturar con mayor eficacia información sobre características complejas del pasado. Sin embargo, la configuración de salida a oculto resulta menos costosa en términos computacionales y puede ser paralelizada con mayor facilidad durante el entrenamiento.

1.3.1.4 Secuencia Asignaciones. Las redes neuronales recurrentes tienen la capacidad de representar problemas de secuencias de diferentes maneras, gracias a su flexibilidad para operar sobre las entradas y salidas de la red como secuencias. Esto las libera de las

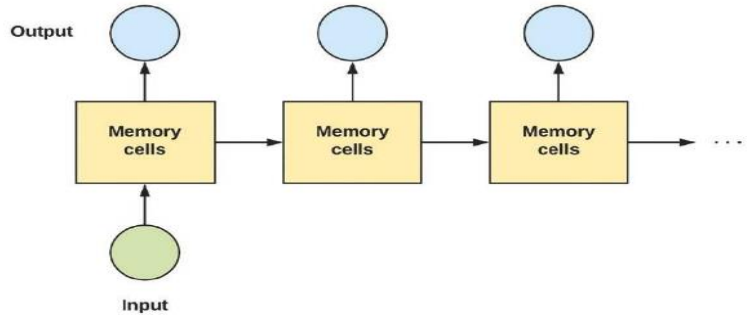
restricciones de tamaño fijo que se encuentran en otras arquitecturas de redes neuronales, como MLP y CNN.

A continuación, se presentan algunos ejemplos de problemas de secuencias resueltos mediante el uso de RNNs:

- Problema de una entrada y una secuencia de salida. Esta configuración se utiliza para problemas de subtitulado de imágenes, en los cuales se proporciona una imagen como entrada a la red y la salida es una secuencia de palabras. Un ejemplo visual de esta configuración se puede ver en la figura 28.

Figura 28

Una Entrada a una Secuencia de Salida

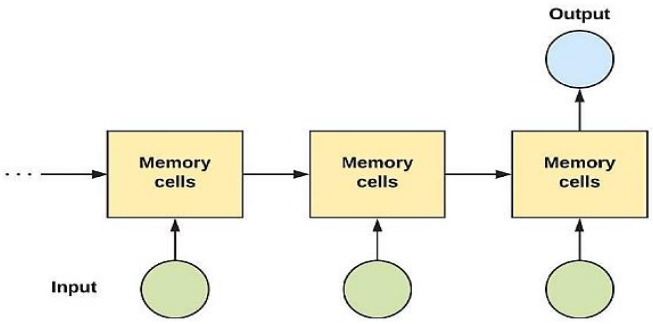


Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 452.

- Un ejemplo de problema de secuencia en el que se tiene una secuencia de entradas y una sola salida es el análisis de sentimiento. En este caso, se pasa una secuencia de palabras como entrada a la red y la salida es una clase que indica si la crítica o el sentimiento son positivos o negativos. Se puede observar un ejemplo visual de esta configuración en la figura 29.

Figura 29

Una Secuencia de Entradas a una Salida

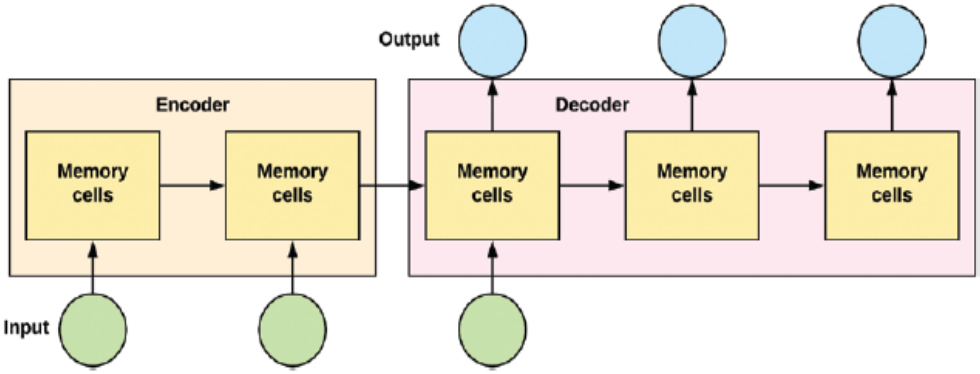


Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 453.

- Otro ejemplo de problema de secuencia es el que involucra una secuencia de entrada y una secuencia de salida. Esta operación es adecuada para aplicaciones como la traducción automática y el reconocimiento de voz, y se conoce comúnmente como codificador-decodificador o arquitectura secuencia a secuencia. En este caso, la red recibe como entrada una secuencia de palabras en un idioma específico y la salida esperada es una secuencia de palabras en otro idioma. La figura 30 ilustra visualmente esta configuración.

Figura 30

Secuencia de Entrada a Secuencia de Salida

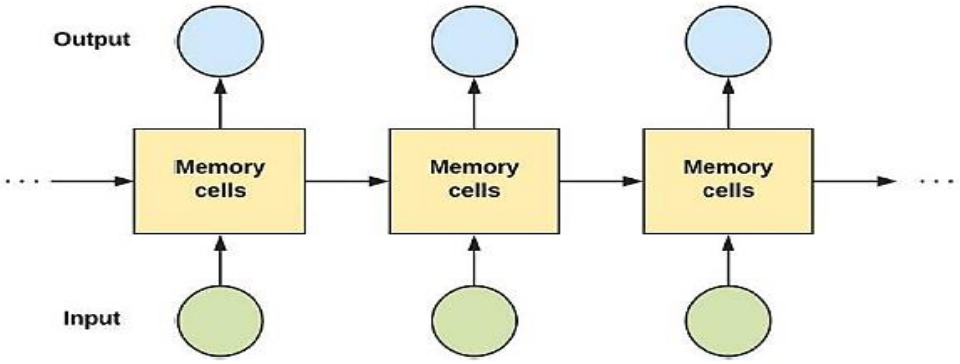


Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 453.

- Otro tipo de problema de secuencia es el que implica una secuencia sincronizada de entrada y salida. Este marco es adecuado para aplicaciones como la clasificación de vídeo, en la que se desea etiquetar cada fotograma de un video de forma individual. En este caso, la red recibe una secuencia sincronizada de entrada y salida, y la tarea consiste en asignar una etiqueta a cada fotograma. La figura 31 muestra un ejemplo visual de esta configuración.

Figura 31

Secuencia sincronizada de entrada a Salida



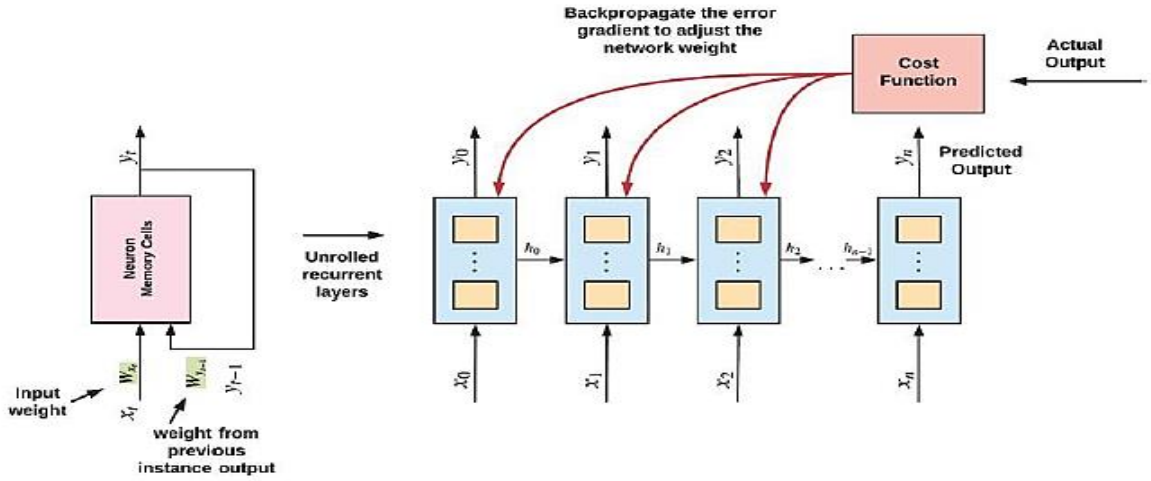
Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 454.

En los esquemas ilustrados anteriormente, la información fluye de la unidad oculta (o célula de memoria) de la capa recurrente en el instante de tiempo “t-1” a la unidad oculta en el instante de tiempo “t”. Como se ha comentado anteriormente, esto se debe a que la información transferida es más rica en características y contiene más información del pasado.

1.3.1.5 Entrenamiento de la red recurrente. Backpropagation a través del Tiempo. El entrenamiento de una red neuronal recurrente implica el algoritmo de retropropagación, que se modifica en retropropagación a través del tiempo (BPTT) debido a la estructura recurrente de la red. El BPTT se aplica desenrollando la neurona recurrente a través del tiempo y aplicando la retropropagación a las neuronas desenrolladas en cada capa de tiempo de la misma manera que una red feedforward tradicional. La figura 32 ilustra esta operación.

Figura 32

Retropropagación en el Tiempo



Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 455.

El problema del gradiente evanescente y explosivo es más común en redes neuronales recurrentes debido a las dependencias a largo plazo. Se ha propuesto la técnica de retropropagación truncada en el tiempo para mitigar este problema, pero puede ser perjudicial para problemas que dependen en gran medida de las dependencias a largo plazo. Para abordar estas deficiencias, se desarrolló una célula de memoria llamada LSTM (Long Short-Term Memory) que puede almacenar información a largo plazo en la célula de memoria de la red recurrente.

1.3.1.6 La Red de Memoria a Largo Plazo (LSTM). La LSTM es una red neuronal recurrente que se caracteriza por tener compuertas que controlan el flujo de información dentro de la célula recurrente, permitiéndole almacenar y olvidar información según sea necesario. Esto la hace muy eficaz para captar dependencias a largo plazo en un gran número de instantes temporales. La célula de la LSTM es más sofisticada que las células de las unidades

recurrentes básicas, lo que le permite mantener información relevante durante períodos de tiempo más largos. Los componentes de la LSTM son:

Célula de memoria

Puerta de entrada

Puerta del Olvido

Puerta de salida

Estos componentes permiten a la RNN recordar y almacenar acontecimientos importantes del pasado. La LSTM toma como entrada el estado anterior de la celda, c_{t-1} ; el estado oculto anterior, h_{t-1} ; y la entrada actual, x_t.

La figura 33 ilustra la célula LSTM, cuyos componentes cumplen distintas funciones para preservar las dependencias a largo plazo en los datos secuenciales.

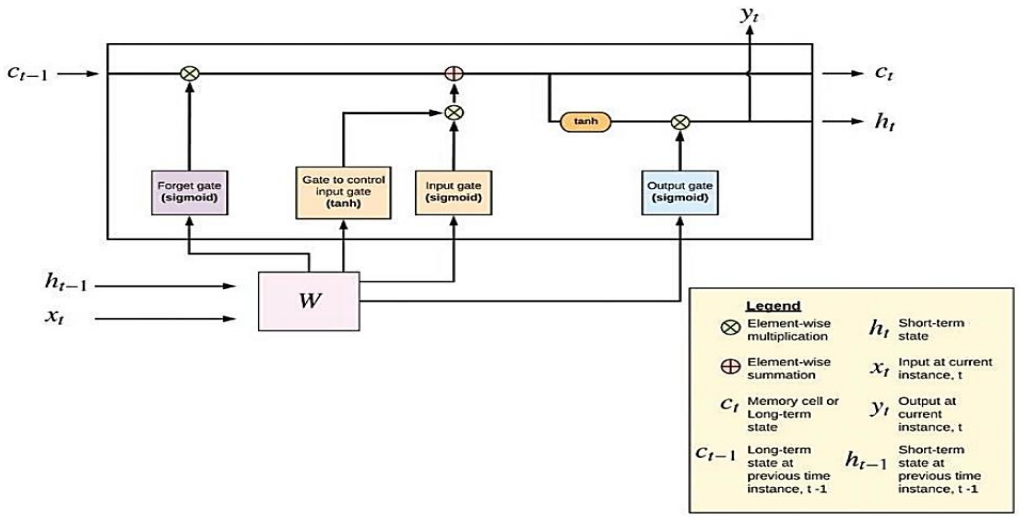
La puerta de entrada, controla información que se almacena en el estado a largo plazo o en la célula de memoria, "c". Trabajando en tándem con la puerta de entrada hay otra puerta que regula la información que fluye hacia la puerta de entrada. Esta puerta analiza la entrada actual a la célula LSTM, x_t , y el estado a corto plazo anterior, h_{t-1}.

La puerta del olvido, regula la cantidad de información en el estado a largo plazo que persiste a través de instantes de tiempo.

La puerta de salida controla cuánta información debe salir de la célula en un instante de tiempo determinado. Esta puerta controla el valor de h_t (el estado a corto plazo) y_t (la salida en el tiempo t).

Figura 33

Célula LSTM



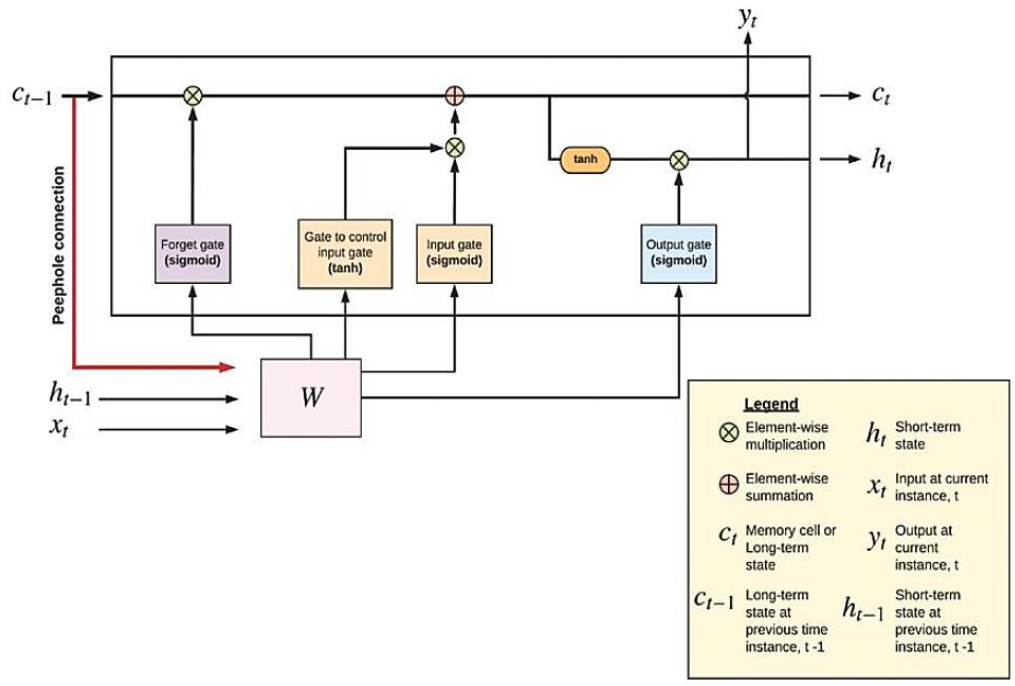
Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 459.

Es relevante destacar que los elementos que conforman las células LSTM son todas redes neuronales que están completamente conectadas. Además de las LSTM, hay otras variaciones de redes recurrentes que utilizan células de memoria, dos de las cuales son las conexiones de Agujero de Cerradura (peephole connections) y las unidades recurrentes con Compuertas Cerradas (gated recurrent units).

1.3.1.7 Peephole Conexión (Mirilla de conexión). La variante de red LSTM conocida como "conexión peephole" añade información adicional a las puertas LSTM al utilizar la célula de memoria del instante de tiempo anterior, c_{t-1}, como entrada. El propósito de esta conexión es proporcionar más información a la unidad LSTM mediante la observación de la memoria a largo plazo almacenada. La figura 34 proporciona una ilustración detallada de este proceso.

Figura 34

Conexión de la Mirilla



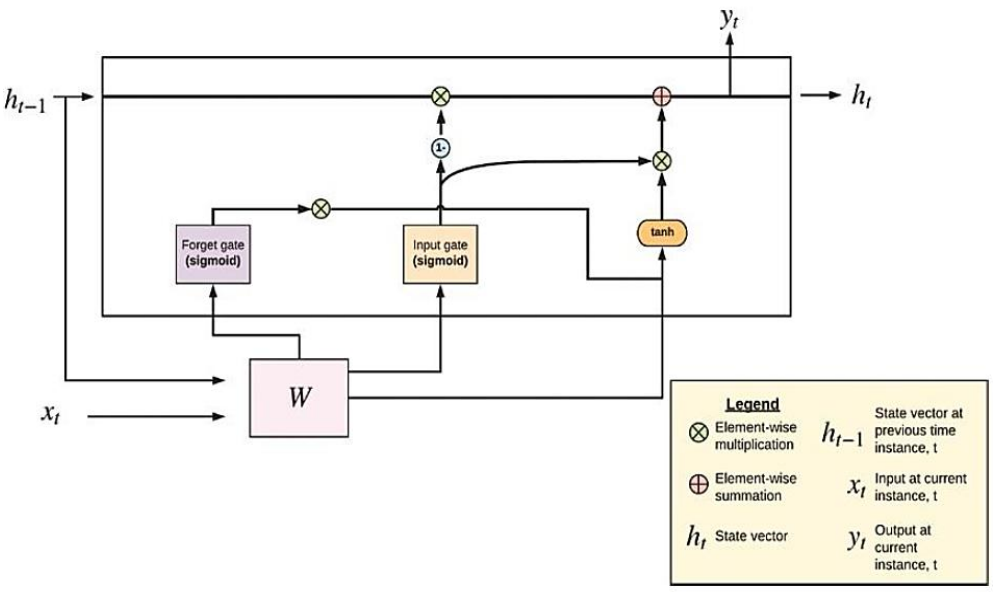
Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 459.

1.3.1.8 Unidad Recurrente Cerrada (GRU). La unidad recurrente cerrada (GRU) es una red neuronal recurrente más simple que la LSTM, pero con un rendimiento comparable y en ocasiones incluso mejor en problemas de modelado de secuencias. La GRU utiliza puertas de entrada y olvido para decidir qué información se almacena en la memoria a largo plazo y combinan la célula y las puertas de entrada.

Además, la GRU elimina la puerta de salida y devuelve el vector de estado h_t en cada instante de tiempo. Esto se ilustra con más detalle en la figura 35.

Figura 35

Unidad Recurrente Cerrada



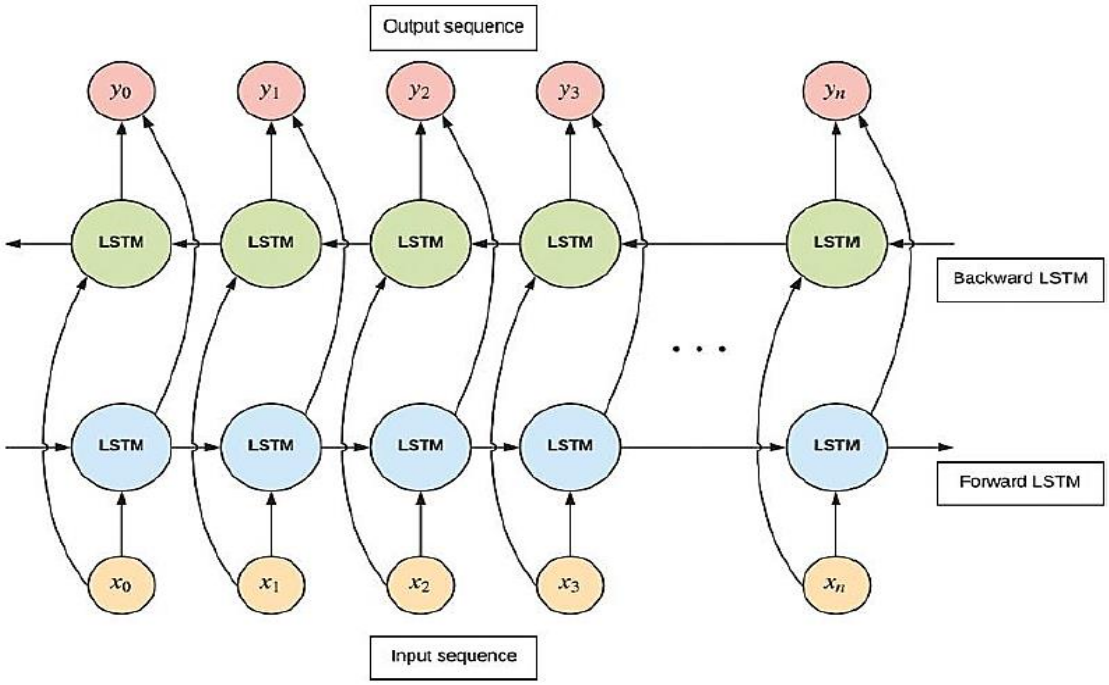
Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 461.

1.3.1.9 Redes Neuronales Recurrentes Aplicadas a Problemas de Secuencias. Las redes neuronales recurrentes LSTM son útiles en muchas aplicaciones de secuenciación, incluyendo análisis de sentimientos, traducción automática, subtitulado de imágenes y videos, y reconocimiento de voz. Estos problemas pueden abordarse con modelos de uno a muchos, muchos a uno o muchos a muchos. Para el análisis de imágenes y videos, se utiliza a menudo la Red convolucional recurrente a largo plazo (LRCN).

1.3.1.10 Redes Neuronales Bidireccionales Recurrentes. La RNN bidireccional es una arquitectura en la que se colocan dos capas recurrentes juntas, una para aprender las dependencias a largo plazo del pasado (LSTM directa) y otra para aprender las dependencias a largo plazo del futuro (LSTM hacia atrás). La entrada se invierte y se introduce en la red en la capa hacia atrás. Esta arquitectura se utiliza para tareas en las que se requiere información tanto del pasado como del futuro, como el reconocimiento de voz y el etiquetado de partes del habla. Se puede ver un ejemplo en la figura 36.

Figura 36

LSTM Bidireccional



Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 465.

La RNN bidireccional es una arquitectura que combina capas recurrentes que aprenden dependencias a largo plazo del pasado y del futuro. Al combinar los resultados de estas redes paralelas, se puede predecir más fácilmente el siguiente paso temporal de una secuencia al tener acceso a toda la información. Inicialmente diseñada para tareas de reconocimiento del habla, esta arquitectura ha demostrado un rendimiento impresionante en muchas otras tareas de predicción de secuencias. La idea subyacente es que algunos problemas de aprendizaje requieren un conjunto coherente de información, como en el caso de la interpretación humana de un idioma a otro. La RNN bidireccional se basa en el contexto de toda una frase cohesionada para una interpretación correcta.

1.4 Mantenimiento e Inteligencia Artificial

El mantenimiento y la inteligencia artificial están estrechamente relacionados ya que la IA puede ayudar a mejorar el mantenimiento en diversos aspectos. A continuación, se describe algunas de las formas en que la IA se utiliza para mejorar el mantenimiento:

- Mantenimiento predictivo: La IA puede ser utilizada para predecir fallos en equipos y maquinarias antes de que ocurran. La recopilación de datos en tiempo real de sensores instalados en los equipos puede ser analizada por algoritmos de IA que utilizan técnicas de aprendizaje automático para identificar patrones y anomalías. De esta manera, los técnicos pueden tomar medidas preventivas antes de que ocurran fallos o averías.

- Optimización de mantenimiento: La IA puede ser utilizada para analizar grandes cantidades de datos relacionados con el mantenimiento, incluyendo registros de mantenimiento, informes de fallas, informes de inspección y datos de sensores. La IA puede utilizar esta información para identificar patrones y tendencias en el rendimiento del equipo y proponer soluciones para optimizar el mantenimiento y minimizar el tiempo de inactividad.

- Soporte a la toma de decisiones: La IA también puede ser utilizada para optimizar la toma de decisiones en el mantenimiento. Los algoritmos de IA pueden analizar grandes cantidades de datos de diferentes fuentes y proporcionar recomendaciones basadas en los patrones y tendencias identificados. Esto puede ayudar a los técnicos a tomar decisiones informadas y a priorizar las tareas de mantenimiento.

En general, la inteligencia artificial puede ayudar a mejorar el mantenimiento al proporcionar una mayor eficiencia, reducir los costos y minimizar el tiempo de inactividad. La capacidad de la IA para analizar grandes cantidades de datos y aprender de ellos permite una mayor precisión en la predicción de fallos y una mejor toma de decisiones en el mantenimiento.

1.4.1 Generaciones del Mantenimiento

El mantenimiento ha evolucionado a lo largo de los años para satisfacer las necesidades cambiantes de las empresas y las industrias. A continuación, se describe una típica explicación de la evolución del mantenimiento cronológicamente:

- Mantenimiento correctivo: Hasta la década de los 60, el mantenimiento era principalmente correctivo, lo que significa que se reparaban las máquinas solo cuando se rompían. Esto era ineficiente, ya que los tiempos de inactividad imprevistos podían ser costosos para las empresas.

- Mantenimiento preventivo: Con el tiempo, las empresas comenzaron a darse cuenta de que era más rentable prevenir averías en lugar de repararlas. Así nació el mantenimiento preventivo, que implica realizar mantenimiento periódico en las máquinas para detectar y corregir problemas antes de que se conviertan en fallos.

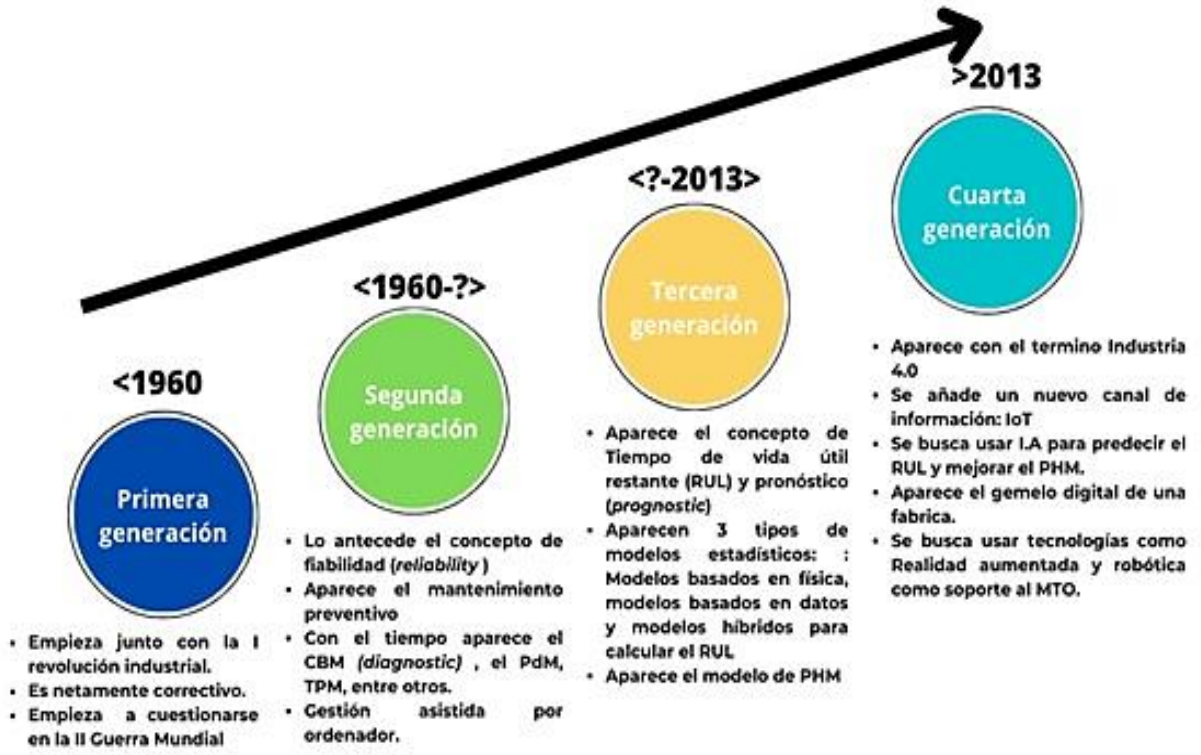
- Mantenimiento predictivo: A medida que las tecnologías avanzaban, se desarrolló el mantenimiento predictivo. Este tipo de mantenimiento utiliza tecnología de monitoreo y análisis de datos para predecir cuándo se producirá una falla y tomar medidas preventivas antes de que suceda.

- Mantenimiento proactivo: En los últimos años, se ha producido una evolución hacia el mantenimiento proactivo. Este tipo de mantenimiento implica analizar constantemente el rendimiento de las máquinas y realizar mejoras para evitar problemas antes de que ocurran.

- Mantenimiento prescriptivo: Actualmente, el mantenimiento prescriptivo es una de las últimas tendencias en la evolución del mantenimiento. Este tipo de mantenimiento utiliza la inteligencia artificial y el aprendizaje automático para recopilar y analizar grandes cantidades de datos para identificar patrones y recomendar acciones específicas para mejorar el rendimiento y prolongar la vida útil de las máquinas.

Figura 37

Generaciones del Mantenimiento



Nota. Tomado de Basado en Santiago García (2003) y Rafael Gouriveau(2016).

1.4.2 Porqué es Importante el Mantenimiento

El mantenimiento es importante por diversas razones. En primer lugar, ayuda a asegurar la disponibilidad y fiabilidad de los equipos y maquinarias en las organizaciones, lo que a su vez contribuye a la continuidad de los procesos productivos y de servicio. Esto se traduce en una reducción de los costos por pérdida de producción, ahorro de costos por reparaciones mayores y extensión de la vida útil de los activos.

Además, el mantenimiento permite mejorar la calidad de los productos y servicios que se ofrecen, ya que equipos y maquinarias en buen estado tienden a producir mejores resultados y ofrecer un desempeño más consistente. También contribuye a mejorar la seguridad laboral al garantizar que los equipos funcionen adecuadamente y minimizar el riesgo de accidentes.

Otra razón importante para el mantenimiento es que ayuda a cumplir con las normas y regulaciones vigentes en diferentes sectores. Por ejemplo, en la industria alimentaria se deben cumplir con estándares sanitarios para asegurar la inocuidad de los productos que se ofrecen, y el mantenimiento juega un papel importante en este sentido.

En resumen, el mantenimiento es esencial para el funcionamiento óptimo de las organizaciones, contribuye a mejorar la calidad de los productos y servicios, garantiza la seguridad laboral y el cumplimiento de normas y regulaciones. Por estas razones, es necesario darle la importancia y la atención que merece.

Tener el mantenimiento bajo control establece una correcta planificación presupuestal a lo largo del tiempo, optimizando la participación en los costos de producción.

La planificación y predicción de la vida útil del activo ayuda a definir cuando remplazar o continuar reparando o manteniendo un activo.

1.5 Lenguajes de Programación y Manipulación de datos más Utilizados en el Contexto del Mantenimiento 4.0

En el Mantenimiento 4.0, algunos de los lenguajes de programación y manipulación de datos más utilizados son:

a) Python: Es un lenguaje de programación interpretado de alto nivel que cuenta con una gran cantidad de bibliotecas y herramientas que lo hacen adecuado para la manipulación y procesamiento de datos. Entre las bibliotecas más populares para el procesamiento de datos en Python se encuentran NumPy, Pandas, Matplotlib, Scikit-learn y TensorFlow.

Referencias:

- Python Software Foundation. (2020). Python Language Reference.

https://docs.python.org/3/reference/index.html

- Van Rossum, G. (2009). The Python programming language. Software: Practice and Experience, 41(9), 1-19. https://doi.org/10.1002/spe.995

b) R: Es un lenguaje de programación estadística utilizado principalmente para el análisis de datos y la creación de modelos estadísticos. R cuenta con una amplia variedad de paquetes que lo hacen ideal para el procesamiento y análisis de datos.

Referencias:

- Ihaka, R., & Gentleman, R. (1996). R: A Language for Data Analysis and Graphics. Journal of Computational and Graphical Statistics, 5(3), 299-314. https://doi.org/10.2307/1390807

- R Core Team. (2020). R: A Language and Environment for Statistical Computing. https://www.r-project.org/

c) SQL: Es un lenguaje de consulta estructurado utilizado para manipular y analizar datos en bases de datos relacionales. SQL es esencial en el contexto del Mantenimiento 4.0 para el manejo de grandes cantidades de datos y la realización de consultas complejas.

Referencias:

- Date, C. J., Darwen, H., & Pascal, F. (2005). Database management systems. Addison-Wesley Professional.
- Chamberlin, D. D., & Boyce, R. F. (1974). SEQUEL: A structured English query language. ACM SIGMOD Record, 5(2), 105-106.

https://doi.org/10.1145/971697.602266

d) MATLAB: Es un lenguaje de programación y un entorno de desarrollo integrado utilizado para el análisis numérico, la modelización matemática y la visualización de datos. MATLAB es especialmente útil en el contexto del Mantenimiento 4.0 para el análisis de datos de sensores y la creación de modelos matemáticos para la predicción de fallas.

Referencias:

- MathWorks. (2020). MATLAB: The Language of Technical Computing. https://www.mathworks.com/products/matlab.html
- Higham, N. J. (2008). Functions of matrices: Theory and computation. Society for Industrial and Applied Mathematics.

e) C#: Es un lenguaje de programación de propósito general desarrollado por Microsoft que se utiliza en una amplia variedad de aplicaciones, incluyendo el desarrollo de aplicaciones para el procesamiento de datos en la ciencia de datos y el mantenimiento 4.0. C# es conocido por su eficiencia, su capacidad de integración con otras herramientas de Microsoft y su capacidad para manejar grandes cantidades de datos.

Referencias:

- Microsoft.

En el marco de esta investigación, hemos utilizado Python como lenguaje de programación para la manipulación de datos y *Google Colaboratory* o también conocido como Google Colab, como herramientas de ensayo y pruebas en la nube.

1.5.1 Python

(Bisong, Bisong, E. "Building Machine Learning and Deep Learnig Models on Google Cloud Platform", 2019, pág. 71)

Python es uno de los lenguajes preferidos en la industria para la ciencia de datos debido a su sintaxis sencilla y la gran cantidad de paquetes de aprendizaje automático y

profundo reutilizables. Estos paquetes permiten desarrollar productos de ciencia de datos sin preocuparse por los detalles internos de un algoritmo o método en particular. En esta sección se describirán los fundamentos de la programación con Python 3, que son la base para trabajar con bibliotecas de nivel superior como *NumPy, Pandas, Matplotlib, TensorFlow y Keras.*

NumPy: Es una biblioteca utilizada para el cálculo numérico y el análisis de datos. Proporciona un objeto de matriz multidimensional (llamado "*array*") que se utiliza para realizar operaciones matemáticas complejas de manera eficiente. *NumPy* también incluye funciones para la manipulación de datos, la transformación de Fourier, el álgebra lineal y la generación de números aleatorios.

Pandas: Es una biblioteca utilizada para el análisis de datos y la manipulación de datos estructurados. Proporciona una serie de estructuras de datos, como Series y *DataFrames*, que permiten la manipulación de datos tabulares y de series de tiempo. También incluye herramientas para la lectura y escritura de datos en una variedad de formatos, como CSV, Excel y SQL.

- *Matplotlib*: Es una biblioteca para la visualización de datos. Permite crear gráficos de línea, barras, dispersión, histogramas, entre otros tipos de gráficos. También permite la personalización de la apariencia de los gráficos y la creación de visualizaciones interactivas.

- T*ensorFlow*: Esta biblioteca es utilizada para el aprendizaje automático y la inteligencia artificial. *TensorFlow* proporciona una estructura para construir y entrenar modelos de aprendizaje automático, incluyendo redes neuronales profundas. Esta biblioteca es especialmente útil para tareas como el reconocimiento de imágenes, el procesamiento del lenguaje natural y la predicción de series de tiempo.

- Keras: Es una biblioteca utilizada para el aprendizaje profundo y la creación de modelos de aprendizaje automático. Proporciona una interfaz fácil de usar para la construcción y entrenamiento de modelos de redes neuronales profundas. Esta biblioteca se integra con *TensorFlow* y se utiliza comúnmente en la industria para tareas de aprendizaje automático como la clasificación de imágenes y la predicción de series de tiempo.

El paradigma de programación que se revisa aquí se puede adaptar o aplicar fácilmente a lenguajes similares como R, que también se utiliza comúnmente en la industria de la ciencia de datos. La mejor manera de trabajar a través de este capítulo y los siguientes en esta sección es ejecutar el código en *Google Colab* o *GCP Deep Learning VMs*.

Google Colab, es un servicio de Google que permite ejecutar código de Python en la nube de forma gratuita. Proporciona acceso a recursos computacionales como la Unidad Central de Procesamiento (CPU), Unidad de Procesamiento Gráfico (GPU) y la Unidad de Procesamiento de Tensores (TPU), lo que permite ejecutar código de aprendizaje automático

y *deep learning* de manera más eficiente. Además, permite compartir y colaborar en tiempo real con otros usuarios.

GCP Deep Learning VMs, es un servicio de *Google Cloud Platform* que ofrece máquinas virtuales preconfiguradas y optimizadas para ejecutar tareas de aprendizaje automático y *deep learning*. Proporciona acceso a GPU y TPU, así como a una variedad de herramientas y bibliotecas para el desarrollo de modelos de aprendizaje automático. También es escalable y se integra fácilmente con otros servicios de *Google Cloud Platform*.

1.5.2 Introducción a Google Cloud Platform

(Bisong, Bisong, E. "Building Machine Learning and Deep Learnig Models on Google Cloud Platform", 2019, pág. 463).

La computación en nube consiste en ofrecer servicios informáticos a través de Internet, como almacenamiento, procesamiento y red, a usuarios que pueden acceder a ellos de forma gratuita o de pago por uso. La idea principal es proporcionar una potencia de cálculo agregada a gran escala para minimizar el coste por unidad de producción. Los usuarios pueden aprovechar la velocidad y potencia de los servicios informáticos y pagar solo por lo que utilizan. La computación en nube surgió como sistema de tiempo compartido, pero ahora se ha convertido en un modelo informático que ofrece soluciones de productos como bases de datos, inteligencia artificial y análisis de datos e infraestructuras sin servidor. Grandes empresas de Tecnología de la Información (TI) como Google, Microsoft, Amazon, IBM y Oracle están gestionando centros de datos empresariales que ofrecen servicios de computación en la nube para cambiar la forma en que trabajamos y utilizamos los sistemas y servicios de software.

Google Compute es un conjunto de productos que proporciona soluciones informáticas a la medida para satisfacer una amplia gama de necesidades. Los productos incluyen: *Compute Engine, App Engine, Kubernetes Engine, Container Registry, Serverless Cloud Functions y Cloud Run.*

Compute Engine, ofrece instancias de computación virtual personalizadas para el procesamiento de datos, mientras que *App Engine* es una plataforma gestionada en la nube para el desarrollo de aplicaciones web, móviles e IoT. *Kubernetes Engine* es un gestor de orquestación para contenedores personalizados basados en *Kubernete*s, y *Container Registry* es un almacenamiento privado de contenedores.

Serverless Cloud Functions, proporciona funciones basadas en la nube para conectar y ampliar servicios en la nube, mientras que Cloud Run es una plataforma de computación gestionada que se escala automáticamente en función de las necesidades de la aplicación.

En resumen, Google Compute proporciona una amplia gama de soluciones informáticas para satisfacer las necesidades de las empresas y organizaciones que buscan soluciones personalizadas y escalables en la nube.

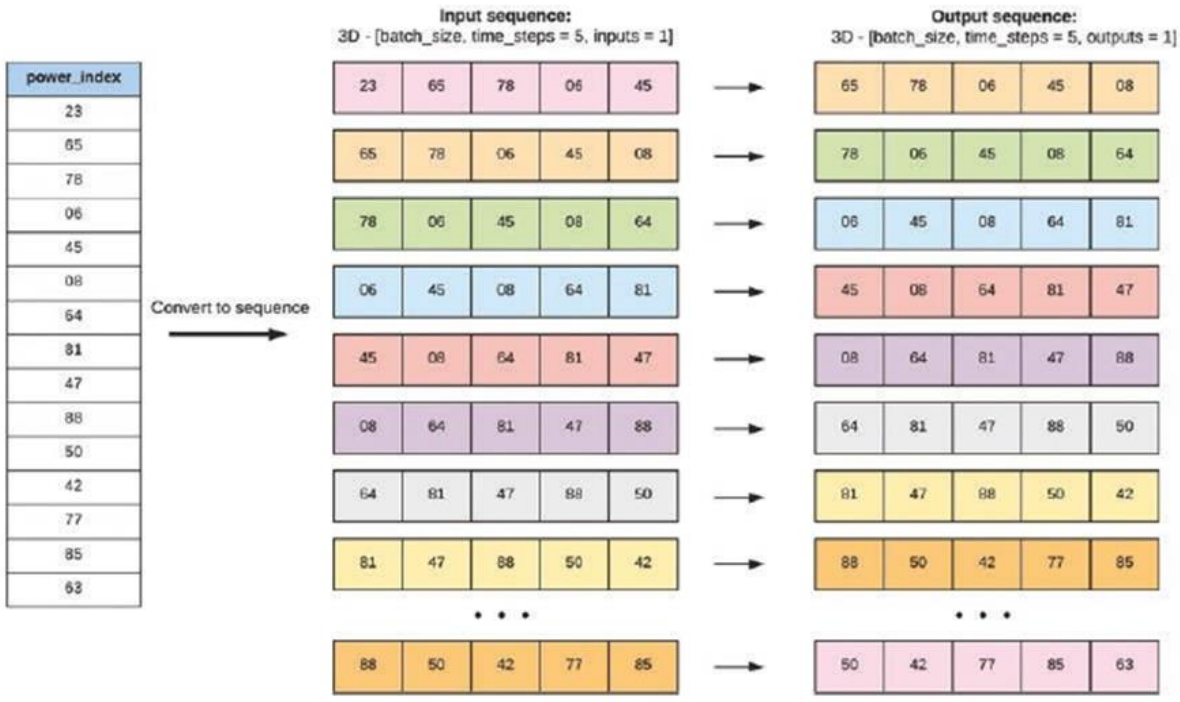
La plataforma Google Colab proporciona un entorno de cuaderno *Jupyter* gratuito para escribir y ejecutar código de *Python*. Los cuadernos *Jupyter* son entornos de programación interactivos basados en la web que permiten crear y compartir documentos que contienen código, texto, visualizaciones y otros elementos. Los cuadernos de Google Colab se ejecutan en servidores de Google, lo que permite a los usuarios aprovechar el poder de procesamiento de Google sin tener que preocuparse por la configuración del entorno de desarrollo en su propia máquina. Además, los cuadernos se guardan automáticamente en la cuenta de Google Drive del usuario y se pueden utilizar para crear presentaciones, documentos y tutoriales interactivos, además de compartirlos con otros y colaborar en tiempo real.

1.5.3 RNN con TensorFlow 2.0: Series Temporales Univariantes

Podemos citar un ejemplo de la misma fuente, que utiliza el conjunto de datos de consumo eléctrico de Nigeria para aplicar un modelo de series temporales univariantes con redes neuronales recurrentes LSTM. El conjunto de datos se procesa para el modelado de series temporales con RNN convirtiendo los datos de entrada y salida en secuencias mediante el método *'convert_to_sequences'*, como se muestra en la figura 38. Cada secuencia contiene cinco filas y una columna, donde la secuencia de salida está un paso por delante de la secuencia de entrada. Es importante escalar el conjunto de datos antes de modelar con RNNs.

Figura 38

Conversión de una Serie Univariante en Secuencias para Predicción con RNNs: Izquierda: muestra, Centro Secuencia de entrada, Derecha: Secuencia de salida.



Nota. Tomado de Ekaba B. Building Machine Learning and Deep Learning Models on Google Cloud Platform Pag 463.

El modelo se entrena en una GPU para aumentar la velocidad de entrenamiento. Se muestran gráficos de las predicciones del modelo y las series originales tanto en los valores escalados como en los normales. Si se ejecuta el código en Google Colab, se debe cambiar el tipo de tiempo de ejecución a GPU e instalar el paquete TensorFlow 2.0 con GPU.

En el Anexo 1, se muestra el código Python para la ejecución de lo descrito.

1.5.4 Procesamiento de datos con software estadístico (Gretl)

Una opción adicional para el tratamiento de datos es utilizar software de análisis estadístico como Gretel. Este software es gratuito, pero su capacidad de procesamiento de datos está limitado a la capacidad del hardware (computadora) donde corre el programa.

La falta de datos o parte de ellos es uno de los problemas que se enfrentan los investigadores y una de las alternativas planteadas es encontrar el modelo del comportamiento de la variable en estudio utilizando la metodología ARIMA (*Autoregressive Integrated Moving Average*), utilizando el software Gretl.

El análisis ARIMA implica varios pasos entre los que se incluyen identificar la estacionalidad de la serie de tiempo, seleccionar los parámetros definidos por ARIMA y ajustar el modelo a los datos observados.

Una vez que se ha ajustado el modelo ARIMA a los datos, se puede utilizar la matriz de correlación para evaluar la calidad del modelo. La matriz de correlación es una matriz cuadrada que muestra las correlaciones entre todas las variables incluidas en el modelo.

Capítulo 2

Estado del arte

En el contexto del presente estudio, el estado del arte podría definirse como un análisis en profundidad de las tecnologías, tendencias y desarrollos actuales en el campo de la industria 4.0.

En la práctica, esto implicaría una revisión sistemática de la literatura existente sobre la Industria 4.0, incluyendo estudios de casos, publicaciones científicas, informes técnicos y otras fuentes relevantes. La revisión del estado del arte debería cubrir temas como la automatización de procesos, el análisis de datos, la inteligencia artificial, la robótica, la ciberseguridad, la realidad aumentada y virtual, y otras tecnologías clave de la Industria 4.0.

El objetivo del estado del arte es proporcionar una comprensión completa del panorama actual del campo, identificando las tendencias emergentes, los desafíos clave y las áreas de investigación en las que se necesita más trabajo. Esta revisión también puede servir como base para el desarrollo de un marco conceptual o teórico para la tesis, y como base para el diseño de experimentos y metodologías de investigación.

2.1 Mantenimiento 4.0

2.1.1 Definiciones

Gemelo Digital: Según lo indicado por Liu et al., (2019a), el concepto de Gemelo Digital contiene principalmente tecnología de adquisición de datos en tiempo real, tecnología de mapeo de datos y tecnología de predicción basada en datos, puede hacer realidad la convergencia entre el producto físico y el espacio virtual.

2.1.2 Uso de la Inteligencia Artificial para Gemelo Digital

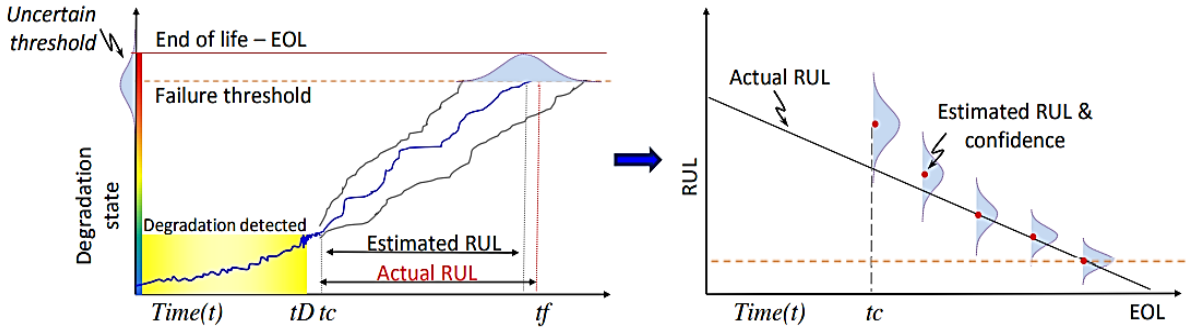
El Gemelo Digital se refiere a la creación de una réplica virtual de un activo físico, proceso o sistema, que se utiliza para el control y la toma de decisiones. Esta réplica incluye tanto información física como funcional, y puede proporcionar información sobre el producto en todas las fases de su ciclo de vida. Los autores, como Grieves y Vickers (2017), Glaessgen y Stargel (2012), Rosen et al. (2015), Boschert y Rosen (2016), Schleich et al. (2017), Tharma et al. (2018) y Liu et al. (2019a), han definido el concepto de Gemelo Digital como una simulación integrada, multifásica, multiescalar y probabilística, que utiliza los mejores modelos físicos

disponibles, actualizaciones de sensores, historial de flotas, entre otros para reflejar la vida de su gemelo. El concepto también se ha relacionado con tecnologías como adquisición de datos en tiempo real, mapeo de datos y predicción basada en datos para lograr la convergencia entre el equipamiento físico y el espacio virtual.

El mantenimiento ha evolucionado junto con la tecnología, y hoy en día, además del diagnóstico de fallos, se busca el pronóstico de los mismos para poder anticiparse a ellos. Esto implica tener en cuenta factores como el tiempo de vida residual útil (RUL) y el tiempo de recambio de piezas, que son importantes para la programación de MTO. (Khanh T.P. Nguyen, 2019).

Figura 39

Vida útil remanente (RUL)



Nota. Tomado de Rafael Gouriveau (2016)

2.1.3 Componentes de los gemelos digitales

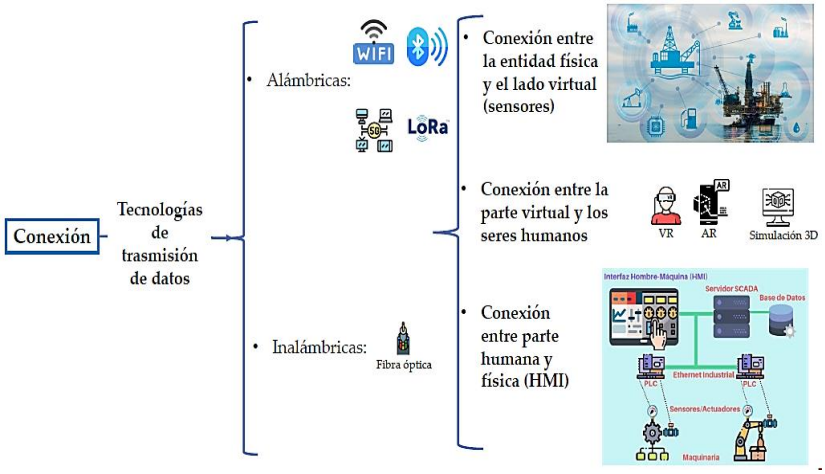
Los gemelos digitales constan de los siguientes componentes principalmente: las tecnologías de adquisición de datos, la transmisión de estos datos, la conexión entre el entorno físico y el entorno virtual y la interfaz de interacción hombre-máquina para la toma de decisiones, aprendizaje y mejoras aplicables al sistema en estudio.

Las figuras 40 y 41 representan dos ejemplos de arquitectura que se pueden configurar en una planta o proceso.

En la figura 40 hay una representación de la toma de datos a través de sensores, que bien podrían estar recolectando datos como vibraciones, temperatura, presiones, etc. esta información es concentrada para luego ser transmitida a través de redes inalámbricas o cableadas hacia el almacenamiento temporal. Con estos datos se entrena los modelos de aprendizaje automático para predecir comportamientos, tendencias y proporcionar recomendaciones para mejorar la eficiencia y fiabilidad de los sistemas, o la toma de decisiones a través de la interacción Hombre – Máquina (HMI). El gemelo digital también puede simular diferentes escenarios de funcionamiento para evaluar el rendimiento de los sistemas en diferentes condiciones y para predecir posibles fallos antes de que ocurran.

Figura 40

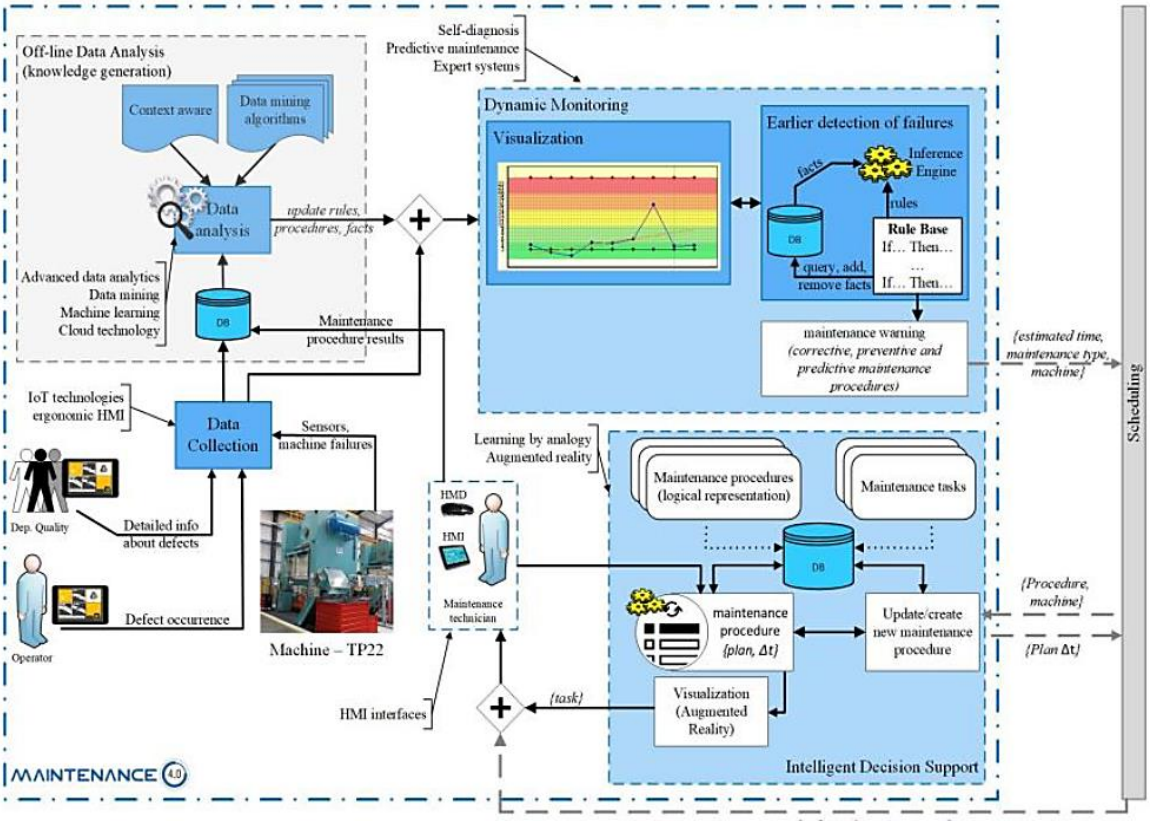
Arquitectura de un Gemelo Digital - HMI



Nota. Tomado de Revista "Gerencia de Riesgos y Seguros": Gemelos Digitales, el camino hacia la eficiencia industrial. https://www.mapfreglobalrisks.com/gerencia-riesgos-seguros/articulos/gemelos-digitales-el-camino-hacia-la-eficiencia-industrial/

Figura 41

Arquitectura de un Gemelo Digital y Gestión de Mantenimiento



Nota. Tomado de Cachada, A et al. Maintenance4.0: Intelligentand Predictive Maintenance System Architecture.

El segundo ejemplo de arquitectura (figura 41) considera la plataforma de la base de datos de gestión de activos, (Oracle, SAP, Maximo, etc.), sistemas de interacción hombre – máquina (HMI), realidad aumentada, monitoreo de parámetros en operación, sistemas de adquisición y supervisión de datos (SCADA).

Toda esta arquitectura permite la toma de decisiones en operación, aprendizaje en línea, así como la mejora en las estrategias de mantenimiento que se aplican a los componentes del sistema.

(Cachada, A. "Maintenance 4.0: Arquitectura de sistema de mantenimiento inteligente y predictivo", 2018)

2.1.4 Sistema Ciberfísico

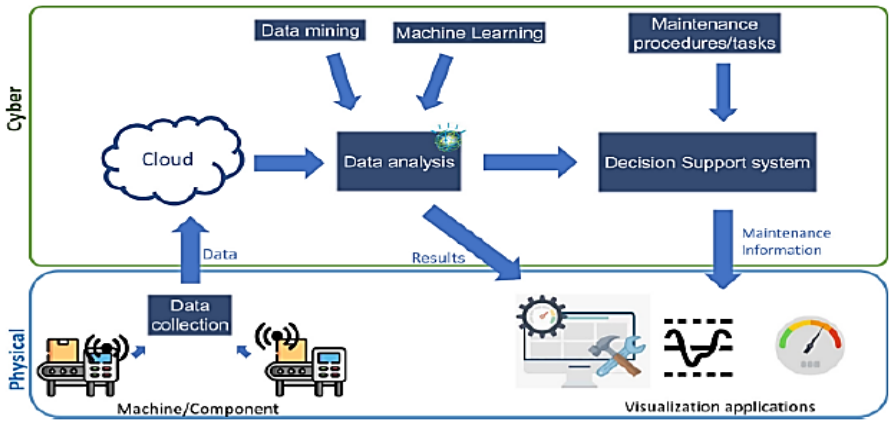
En un sistema ciberfísico los componentes físicos interactúan con los sistemas informáticos y de comunicaciones. Estos sistemas se utilizan comúnmente en la fabricación avanzada, la robótica, la ingeniería, la gestión de infraestructuras críticas, la agricultura de precisión, la salud y otros campos.

Los sistemas ciberfísicos consisten en sensores, actuadores, dispositivos de cómputo y sistemas de comunicación que están integrados en los sistemas físicos. Estos sistemas utilizan la información proporcionada por los sensores para ajustar los actuadores, lo que permite que los sistemas físicos se controlen de manera más efectiva y eficiente.

Los sistemas ciberfísicos pueden mejorar la eficiencia y la seguridad de una amplia variedad de sistemas físicos. Por ejemplo, en la fabricación avanzada, los sistemas ciberfísicos pueden optimizar la producción mediante el uso de sensores y algoritmos de control para ajustar la velocidad de la línea de producción y garantizar que los productos cumplan con las especificaciones de calidad. (Dalzochio, 2020)

Figura 42

Sistema ciberfísico



Nota. Tomado de Dalzochio, et al. Machine learning and reasoning for predictive maintenance in Industry 4.0: Current status and challenges.

2.1.4.1 Adquisición de datos. Algunas de las tecnologías de adquisición de datos disponibles, y la elección de la tecnología adecuada dependerá de las necesidades y requisitos específicos de cada aplicación.

Sensores: Los sensores son dispositivos que pueden medir y detectar cambios en el ambiente físico, como temperatura, presión, humedad, luz y movimiento. Los sensores se utilizan a menudo en aplicaciones de Internet de las cosas (IOT) para recopilar datos en tiempo real sobre el mundo físico.

Registro de datos: El registro de datos implica el almacenamiento de información en una base de datos o en un archivo. Los sistemas de registro de datos pueden recopilar información de sensores, dispositivos de red y otras fuentes de datos, y se utilizan a menudo en aplicaciones industriales y de control de procesos.

Adquisición de señales: La adquisición de señales implica la conversión de señales analógicas en datos digitales. Se utiliza a menudo en aplicaciones de medición y control, como en la industria de la automatización y en la ingeniería biomédica.

Adquisición de imágenes: La adquisición de imágenes implica la captura de imágenes digitales a partir de dispositivos como cámaras digitales y escáneres. Se utiliza a menudo en aplicaciones de reconocimiento de patrones y visión artificial.

Adquisición de sonido: La adquisición de sonido implica la captura de señales de sonido y su conversión en datos digitales. Se utiliza a menudo en aplicaciones de análisis acústico, como el procesamiento de señales de audio en la industria musical y la detección de fallas en maquinaria.

Adquisición de datos GPS: La adquisición de datos GPS implica la recepción de señales de satélite para determinar la ubicación geográfica de un objeto. Se utiliza a menudo en aplicaciones de navegación y seguimiento, como en sistemas de navegación de vehículos y en aplicaciones militares.

2.1.4.2 Transmisión de datos. Existen varias tecnologías de transmisión de datos, sin embargo, podemos considerarlas en dos grandes grupos:

2.1.4.2.1 Inalámbrica. Permiten el intercambio de información sin la necesidad de cables o conexiones físicas. Algunos ejemplos incluyen Wi-Fi, Bluetooth, Zigbee, NFC, entre otros. Estas tecnologías utilizan diferentes bandas de frecuencia y estándares de transmisión para brindar conectividad inalámbrica a dispositivos electrónicos como smartphones, computadoras, sensores, etc.

WiFi (Wireless Fidelity): Utiliza ondas de radio para transmitir datos a través de una red inalámbrica.

Zigbee (Wifi optimizado) es un estándar de comunicación inalámbrica basado en IEEE 802.15.4. Se utiliza para conectar dispositivos inteligentes en hogares y edificios inteligentes.

Ofrece una gran eficiencia energética, confiabilidad y seguridad integrada. Permite la creación de redes de dispositivos interconectados controlables remotamente.

Li-Fi (Light Fidelity): Utiliza luz visible para transmitir datos a través de una red inalámbrica. Es una tecnología similar al Wi-Fi, pero se utiliza la luz en lugar de ondas de radio.

Bluetooth: Utiliza ondas de radio de corto alcance para conectar dispositivos inalámbricos, como auriculares y teléfonos móviles.

NFC (Near Field Communication): Utiliza ondas de radio de corto alcance para conectar dispositivos inalámbricos en una distancia muy cercana.

Enlace Satelital: Utiliza satélites para transmitir datos a través de una red de área amplia (WAN).

4G/5G: Utiliza ondas de radio para transmitir datos a través de una red móvil de área amplia (WAN).

2.1.4.2.2 Cableadas o Alámbricas. Las tecnologías cableadas requieren conexión física de cables para transmitir información y ofrecen mayor velocidad y menor probabilidad de interrupciones en la señal. Sin embargo, requieren una infraestructura física y pueden limitar la movilidad y flexibilidad.

Ethernet: Utiliza cables para conectar dispositivos a una red de área local (LAN).

Fibra óptica: Utiliza fibras de vidrio o plástico para transmitir datos a altas velocidades a través de una red de área local (LAN) o de área amplia (WAN).

2.1.4.3 2.Interfaz Hombre – Máquina. El uso de gemelos digitales puede mejorar la eficiencia y la seguridad en una amplia variedad de aplicaciones, como la planificación de la producción, la optimización de la energía, la mejora de la calidad de productos, entre otras. Además, los gemelos digitales pueden ayudar a predecir y solucionar problemas antes de que ocurran en el mundo real, lo que puede resultar en un ahorro significativo de tiempo y recursos.

Se lista a continuación algunos ejemplos sobre las formas en que se puede interactuar con los gemelos digitales y el mundo físico. La elección de la forma de interacción depende de los requisitos específicos de la aplicación y de la tecnología disponible.

Interfaz gráfica de usuario (GUI): se puede utilizar una interfaz gráfica de usuario para interactuar con el gemelo digital mediante la manipulación de objetos virtuales en un entorno visual.

Sensores y dispositivos de realidad virtual: los sensores y dispositivos de realidad virtual, como las gafas de realidad virtual, pueden utilizarse para interactuar con el gemelo digital de manera más inmersiva.

Comunicación de máquinas a máquinas (M2M): se pueden utilizar dispositivos electrónicos y protocolos de comunicación para conectar el gemelo digital con el mundo físico y manipular actuadores.

Interfaces de programación de aplicaciones (API): se pueden utilizar APIs para interactuar con el gemelo digital y obtener información para la toma de decisiones.

Interfaz de comandos de línea: se puede utilizar una interfaz de comandos de línea para interactuar con el gemelo digital mediante la introducción de comandos y la visualización de resultados en una terminal.

2.2 Uso de la Inteligencia artificial para cálculo de RUL

El RUL (Remaining Useful Life) es un indicador que refleja la cantidad de tiempo que un activo puede funcionar antes de su desgaste o falla. La Inteligencia Artificial (IA) puede ayudar a calcular el RUL de manera más precisa y eficiente que los métodos manuales o tradicionales. La IA utiliza técnicas avanzadas de aprendizaje automático y análisis de datos para analizar grandes cantidades de información y detectar patrones y tendencias en el desempeño de los activos. Esto permite a las empresas predecir con mayor precisión el tiempo de vida restante de sus activos, lo que les permite planificar con anticipación su mantenimiento y reemplazo, reduciendo los costos y mejorando la eficiencia. Además, el uso de IA en el cálculo del RUL también puede ayudar a las empresas a identificar problemas potenciales antes de que se conviertan en fallas costosas, aumentando la seguridad y la confiabilidad de sus activos.

2.2.1 Modelos de Estimación del RUL

La elección del modelo de estimación del RUL dependerá de varios factores, como el tipo de activo, la cantidad y calidad de los datos disponibles, y los requisitos de precisión de la empresa. Es importante tener en cuenta que los modelos de estimación del RUL son solo una herramienta de ayuda y deben ser validados y verificados antes de su uso.

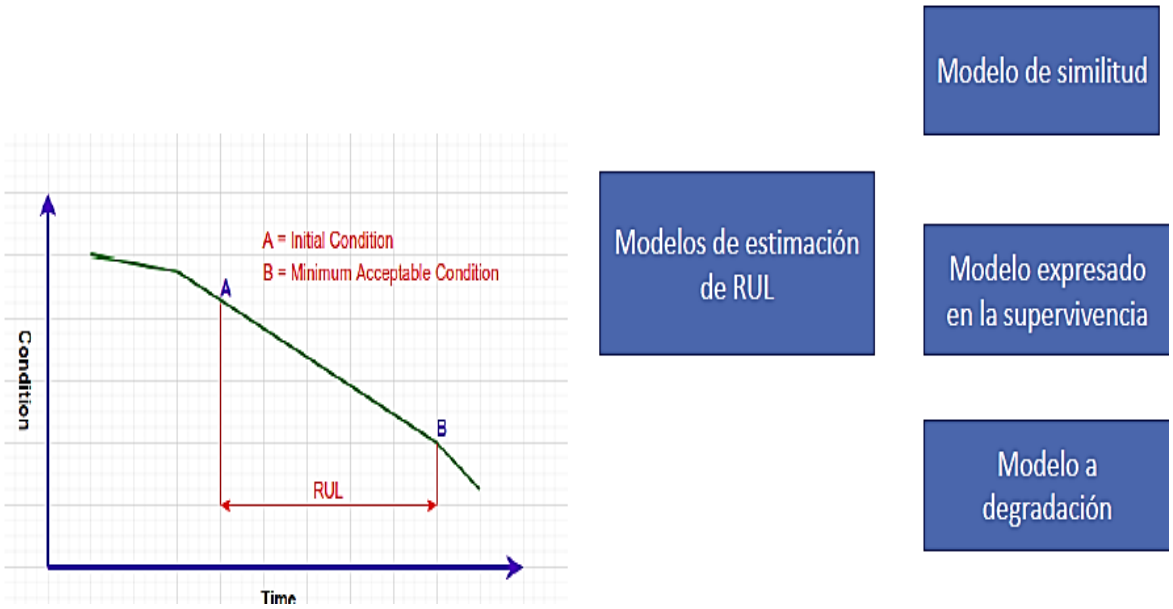
Asimismo, si el modelo sobre ajusta, la estimación perfecta en los datos de entrenamiento se pierde cuando intentamos predecir nuevos datos.

Por otro lado, si el modelo sub ajusta es robusto ante diferentes datos de entrada, pero es un mal predictor y comete errores sistemáticos.

En conclusión, Un buen modelo será aquel que describe bien los datos distintos a los que fue entrenado.

Figura 43

Tiempo de Vida Remanente (RUL)



Nota. Tomado de Predictive Maintenance, Part 3: Remaining Useful Life Estimation https://www.youtube.com/watch?v=Dd_4rbWYgI4&t=183s

2.2.1.1 Modelo de Similitud. Este modelo se basa en la comparación de un activo con su contraparte similar, utilizando información histórica y observaciones del desempeño. Se busca identificar patrones similares en la vida útil y la degradación de los activos para estimar su RUL.

2.2.1.2 Modelo de Supervivencia. Este modelo se basa en la teoría de la supervivencia y se aplica a activos que tienen un tiempo de vida finito. Se utiliza para estimar la probabilidad de que un activo siga funcionando hasta un tiempo determinado y, por lo tanto, estimar su RUL.

2.2.1.3 Modelo de Degradación. Este modelo se basa en la medición y el monitoreo de la degradación de un activo a lo largo del tiempo. La información recopilada se utiliza para estimar su tasa de degradación y, por lo tanto, estimar su RUL. Estos modelos son particularmente útiles para activos que tienen un desgaste físico y técnico predecible.

2.3 Casos de Estudio Analizados en el Presente Trabajo

2.3.1 Caso 1: Estimación de la vida útil restante de una bomba de lodos industrial

(Khan, M.M.; Tse, P.W.; Trappey, A.J.C. "Development of a Novel Methodology for Remaining Useful Life Prediction of Industrial Slurry Pumps in the Absence of Run to Failure Data", 2021)

El artículo describe un método para predecir la vida útil restante (RUL) de una bomba de lodos industrial en funcionamiento, que no tiene datos históricos de fallas. La predicción de la RUL es importante en un sistema de mantenimiento basado en condiciones, ya que ayuda a predecir cuándo una máquina fallará y cuándo se debe llevar a cabo el mantenimiento para evitar una falla costosa.

Existen tres tipos de métodos de predicción de RUL: basados en modelo, basados en datos e híbridos. El método híbrido es el más popular entre los investigadores porque combina las fortalezas de los otros dos métodos.

Obtener datos de ejecución hasta el fallo para predecir el RUL puede ser costoso y llevar mucho tiempo. En el caso de las bombas de lodos utilizadas en la industria petrolera, se puede obtener información de los sensores de vibración, pero estos datos pueden estar ruidosos y puede ser difícil distinguir entre señales válidas e inválidas. Para abordar este problema, los investigadores utilizaron cuatro indicadores estadísticos (RMS, kurtosis, PMD en el dominio del tiempo y kurtosis en el dominio de la frecuencia) para filtrar los datos de vibración y obtener conjuntos de datos válidos para el análisis.

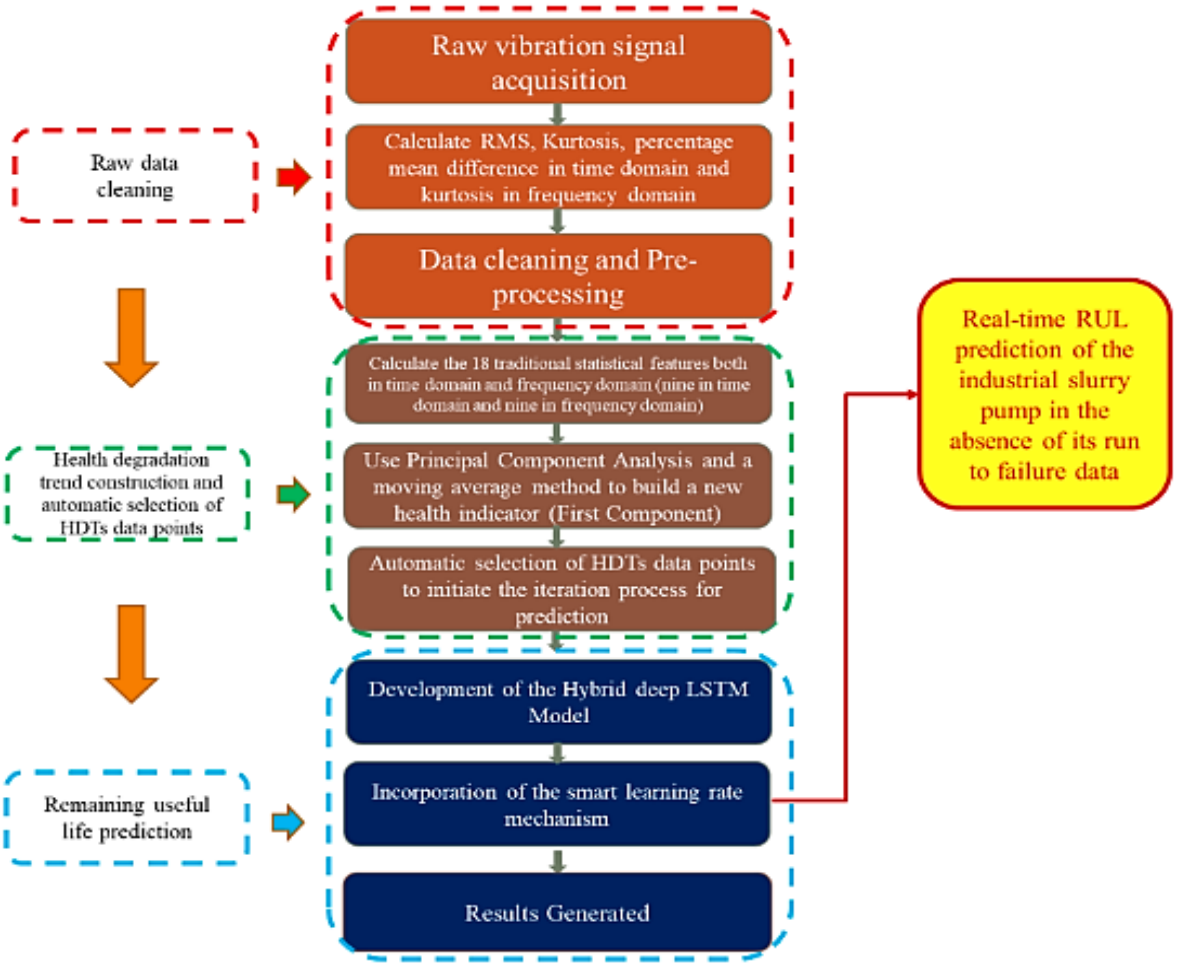
Una vez que se obtuvieron los conjuntos de datos válidos, se construyó una tendencia de degradación de la salud (HDT) utilizando el análisis de componentes principales (PCA) y un método de media móvil. Luego, se desarrolló un modelo híbrido de red neuronal de memoria a largo plazo (LSTM) que utilizó la HDT como entrada para predecir la RUL en tiempo real.

Los resultados de la predicción de la RUL en línea producidos por el modelo desarrollado fueron satisfactorios en comparación con otros métodos de predicción de RUL en línea. El artículo proporciona un enfoque novedoso y eficaz para predecir la RUL en línea de una máquina en funcionamiento en ausencia de datos históricos de fallas.

Se esquematiza la metodología utilizada en la figura 44.

Figura 44

Metodología

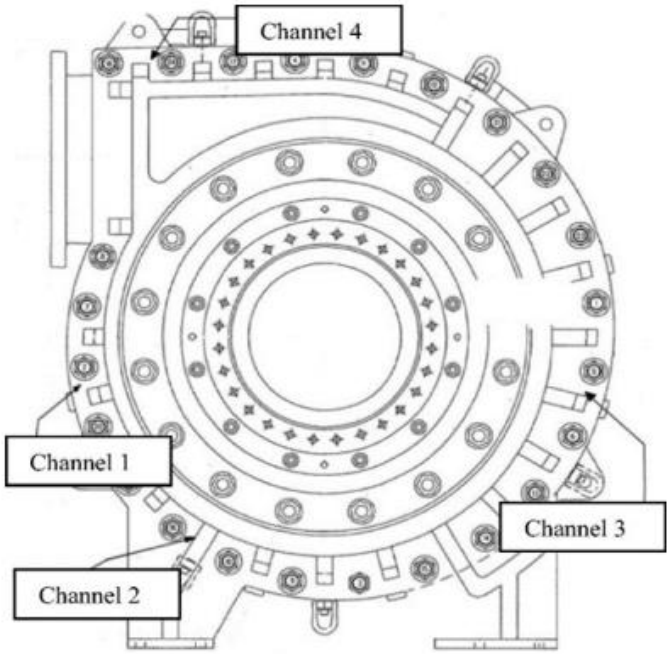


Nota. Tomado de The proposed framework for data filtering, health degradation assessment, automatic selection of HDTs data points, and online RUL prediction of slurry pump. Muhammad Mohsin Khan, Peter W. Tse * and Jinzhao Yang

En la figura 45 se indica la posición de los sensores

Figura 45

Ubicación de los Sensores de Vibración



Nota. Tomado de The proposed framework for data filtering, health degradation assessment, automatic selection of HDTs data points, and online RUL prediction of slurry pump. Muhammad Mohsin Khan, Peter W. Tse * and Jinzhao Yang

En la literatura existen muchos estudios sobre la predicción de la vida restante (RUL) de equipos utilizando redes neuronales de aprendizaje profundo. Sin embargo, en el mundo real, solo se dispone de datos previos a la falla (datos de prueba), ya que las máquinas no se permiten funcionar hasta su falla. Los métodos de ajuste de curvas no son confiables para las curvas de degradación que tienen estructuras profundas, por lo que se desarrolló un modelo híbrido de LSTM profundo con un mecanismo de tasa de aprendizaje inteligente para la estimación del RUL de una bomba de lodo industrial en operación. Se recomienda que el proceso de iteración para la predicción comience en puntos particulares de la curva de degradación, que tienen pendientes crecientes consecutivas. El mecanismo de tasa de aprendizaje inteligente incorporado en el modelo LSTM profundo híbrido ha trabajado como un "catalizador" para obtener puntos de predicción aceptables. La estrategia desarrollada de producir un punto de predicción aceptable y luego agregarlo al vector de entrada para otra predicción ha demostrado ser una alternativa exitosa al método de ajuste de curvas. Los autores están trabajando para mejorar aún más los resultados de la predicción del RUL. La metodología desarrollada también puede utilizarse para predecir la RUL de diferentes tipos de máquinas rotativas con cambios significativos en las señales observadas, como la vibración, las fuerzas y la presión.

2.3.2 Caso 2: Estimación de la Vida Útil de un Disco de Álabes de Turbina de Gas

(Shahnawaz Ahmad, A Suman, T Sidharth, Ganesh Pawar, Vikas Kumar, N.S. Vyas. "Structural Integrity Analysis and Life Estimation of a Gas Turbine Bladed-Disc", 2019)

El estudio realizado por Ahmad (2019), se justifica en la criticidad del componente de la turbina, debido a que está sometido a grandes esfuerzos mecánicos y térmicos.

El análisis de integridad estructural comienza con la evaluación de las condiciones de carga y las propiedades del material del disco de álabes. Se utilizan técnicas de modelado por elementos finitos para simular el comportamiento del disco bajo diferentes cargas y temperaturas. Se realizan pruebas no destructivas para detectar posibles defectos o grietas en el material.

La estimación de la vida útil del disco se basa en la fatiga del material, que se produce cuando el disco está sometido a ciclos repetidos de carga y descarga. Se utilizan modelos de fatiga para predecir la vida útil del disco y se comparan con los requisitos de vida útil del fabricante.

Si se detectan defectos o grietas en el disco durante el análisis de integridad estructural, se realizan reparaciones o se reemplaza el disco. También se pueden tomar medidas para reducir las cargas en el disco y prolongar su vida útil, como la mejora del diseño de la turbina o la optimización del proceso de operación.

La metodología utilizada sigue los siguientes pasos:

- Evaluación de las condiciones de carga y las propiedades materiales: Se recopila información sobre las condiciones de carga, incluyendo las cargas estáticas y dinámicas, las temperaturas y las velocidades de rotación. Se determinan las propiedades materiales del disco de álabes, como la resistencia a la tracción, la resistencia a la fatiga y el módulo de elasticidad.

- Modelado por elementos finitos: Se utiliza software de modelado por elementos finitos para simular el comportamiento del disco de álabes bajo diferentes cargas y temperaturas. Se pueden utilizar diferentes modelos para representar la geometría del disco y las condiciones de carga, y se aplican diferentes tipos de elementos finitos para discretizar la geometría del disco.

- Pruebas no destructivas: Se realizan pruebas no destructivas, como inspecciones visuales, inspecciones por líquidos penetrantes, inspecciones por ultrasonidos y pruebas de radiografía, para detectar posibles defectos o grietas en el material del disco. Estas pruebas pueden realizarse en diferentes etapas del proceso, desde la fabricación hasta el mantenimiento y la reparación.

- Análisis de fatiga: Se utilizan modelos de fatiga para predecir la vida útil del disco bajo diferentes condiciones de carga y temperatura. Estos modelos pueden basarse en

diferentes enfoques, como la teoría de la fractura mecánica o la fatiga multiaxial. Se pueden realizar pruebas de fatiga en muestras del material del disco para validar los modelos y ajustar los parámetros.

- Comparación con los requisitos de vida útil del fabricante: Se comparan los resultados del análisis de fatiga con los requisitos de vida útil especificados por el fabricante del disco. Si la vida útil estimada es menor que los requisitos, se pueden tomar medidas para reducir las cargas en el disco o mejorar la calidad del material.

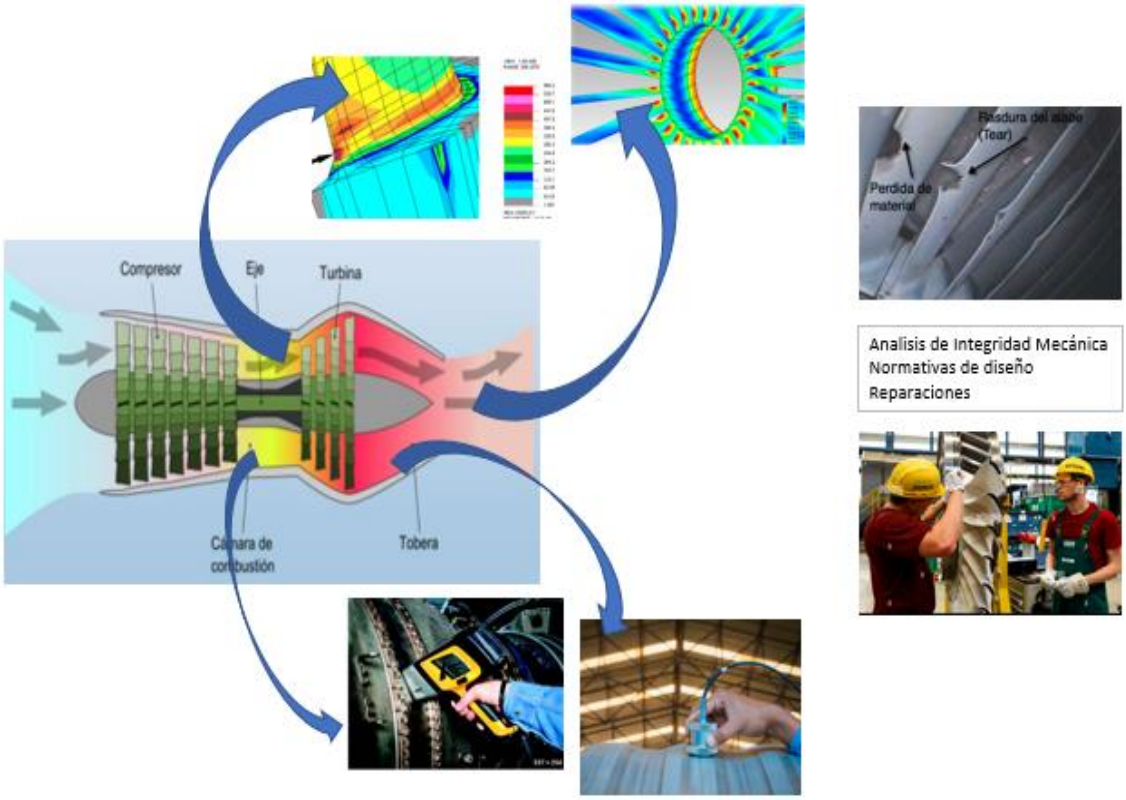
- Reparación o reemplazo del disco: Si se detectan defectos o grietas en el disco durante el análisis de integridad estructural, se pueden realizar reparaciones o se puede reemplazar el disco. Las reparaciones pueden incluir el relleno de grietas con soldadura o la eliminación de secciones defectuosas del disco. En algunos casos, se puede optar por reemplazar el disco por uno nuevo.

- Mejora del diseño o la operación: Si se identifican problemas de integridad estructural o vida útil durante el análisis, se pueden tomar medidas para mejorar el diseño de la turbina o la operación del proceso. Esto puede incluir la optimización de la geometría del disco o la reducción de las cargas en la turbina a través del control de la velocidad de rotación o la temperatura.

No se menciona ninguna arquitectura específica en el proceso de modelado o análisis utilizado en el estudio, sin embargo, podemos representar esquemáticamente el proceso como en la Figura 46.

Figura 46

Monitoreo y Mejora en el RUL de la Turbina de Gas



2.3.3 Caso 3: Diagnostico de Fallos y Estimación de la Vida Útil Restante de un Motor Aeronáutico Mediante una Red Neuronal LSTM

(M. Yuan, Y. Wu y L. Lin, "Diagnóstico de fallas y estimación de la vida útil restante del motor aeronáutico mediante la red neuronal LSTM", 2016, pág. 135...140)

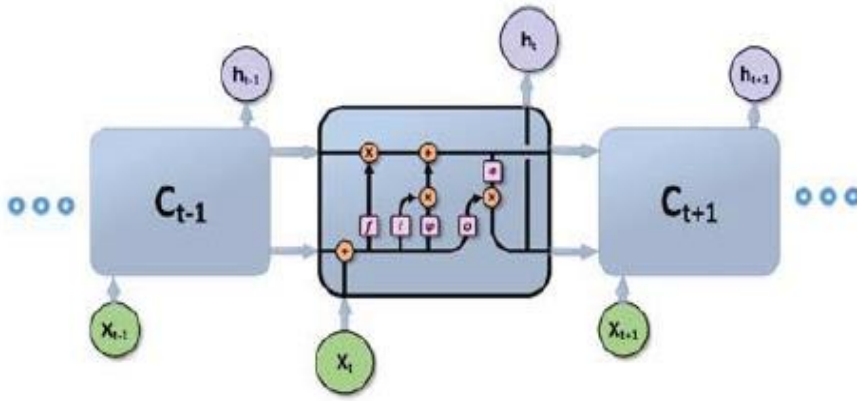
La localización de fallos y la predicción de la vida útil restante de los motores aeronáuticos es un reto complejo. Las tecnologías de pronóstico y gestión de la salud (PHM) basadas en datos se han convertido en la solución principal para el diagnóstico de fallos y la estimación de la vida útil restante. Entre las tecnologías basadas en datos, la red neuronal es uno de los modelos más avanzados. Se han utilizado diferentes tipos de redes neuronales para evaluar la vida útil restante de los motores de aviación, pero estos sólo se enfocan en un tipo de falla, lo que complica el sistema de predicción. Peel et al. intentó utilizar redes de perceptrón multicapa (MLP) y de función de base radial (RBF) para evaluar el RUL de los motores de aviación, mientras que Heimes et al. utilizaron la RNN clásica. La red de Hermes mejoró el rendimiento utilizando neuronas ocultas en la RNN, las mismas que contienen información sobre la historia de todos los elementos pasados de la secuencia.

Para superar estas limitaciones, se utilizó redes neuronales LSTM para construir un modelo común para múltiples fallos y fallos híbridos. Además, este modelo proporciona información sobre el valor estimado de la vida útil restante y la probabilidad de cada fallo al

mismo tiempo, lo que simplifica la complejidad del sistema de Pronóstico y Salud del Activo (PHM) y proporciona información más completa para el plan de mantenimiento.

Figura 47

LSTM



Nota. Tomado de Versión de la LSTM presentada por Hotchreiter y Schmidhuber en 1977

Donde:

$$f_t = \sigma(W_f . X_t + R_f . h_{t-1} + b_f) \quad (1)$$

$$i_t = \sigma(W_i . X_t + R_i . h_{t-1} + b_i) \quad (2)$$

$$o_t = \sigma(W_o . X_t + R_o . h_{t-1} + b_o) \quad (3)$$

$$\tilde{C}_t = \varphi(W_c . X_t + R_c . h_{t-1} + b_c) \quad (4)$$

$$C_t = f_{t-1} * C_{t-1} + i_t * \tilde{C}_t \quad (5)$$

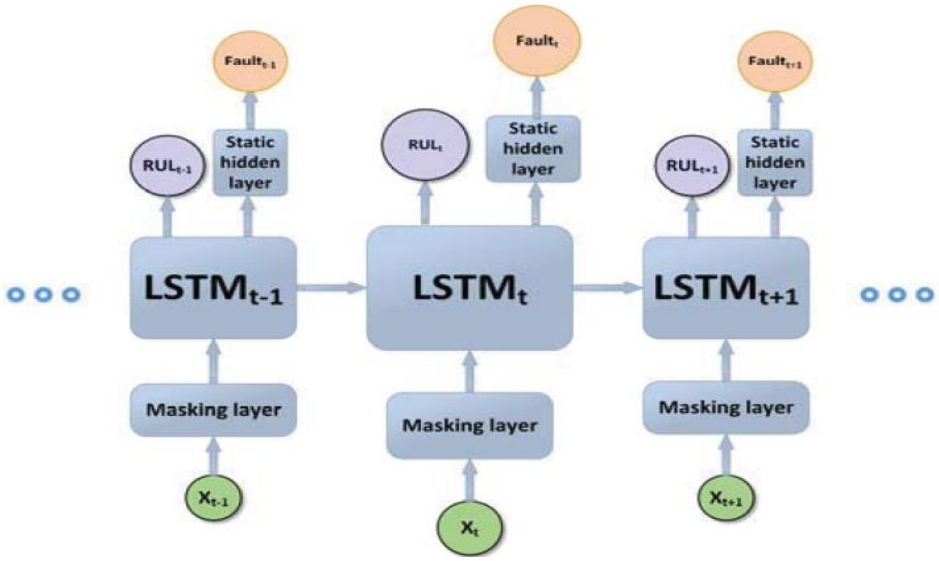
$$h_t = o_t * \emptyset(C_t) \quad (6)$$

El experimento utiliza datos proporcionados por el *Commercial Modular Aero-Propulsion System Simulation* (C-MAPSS) de NASA, que simula la propagación de daños en motores de turbina de gas de aeronaves. Los datos incluyen desde el funcionamiento normal hasta el fallo del sistema y se utilizan cuatro subconjuntos de diferentes emisiones que incluyen 21 de 58 variables de salida y tres indicadores de condiciones de funcionamiento. Para cada motor simulado se asignó un nivel inicial de desgaste y se iniciaron fallos en momentos aleatorios durante la simulación. El etiquetado RUL se realiza utilizando una máquina de vectores soporte (SVM) como detector de anomalías para encontrar el momento en que se producen los fallos. El etiquetado de fallos se realiza creando un vector binario para cada punto temporal, cuya dimensión es igual al número de categorías de fallos conocidas.

La arquitectura de la red neuronal utilizada fue:

Figura 48

Arquitectura de la Red Neuronal Utilizada en el Experimento



Nota. Tomado de 2016 IEEE/CSAA International Conference on Aircraft Utility Systems (AUS)

Se realizaron varias pruebas para estimar el RUL utilizando la configuración mencionada anteriormente. Los resultados muestran que el modelo propuesto supera a los enfoques tradicionales y otros modelos de aprendizaje automático. Además, se demostró que la inclusión de una capa oculta estática adicional mejora significativamente la capacidad de clasificación.

Para evaluar el rendimiento del modelo en la tarea de diagnóstico de fallos, se calcularon la precisión, la sensibilidad y la especificidad. Los resultados indican que el modelo propuesto tiene un mejor rendimiento que otros modelos de referencia y los resultados de las pruebas muestran que el modelo propuesto es altamente efectivo en la estimación del RUL y el diagnóstico de fallos. Además, se discuten varias configuraciones del modelo, como la partición del conjunto de datos, la función de costo, la regularización y el optimizador utilizados para mejorar el rendimiento del modelo.

En las figuras 49 y 50, se muestra gráficamente los resultados de predicción de la red neuronal LSTM estándar para evaluar la vida útil restante de motores. Las curvas azules sólidas representan el objetivo de aprendizaje de la red neuronal (RUL reales), las curvas verdes sólidas son las estimaciones de la red neuronal y las curvas rojas sólidas representan los errores. También se muestran los resultados del diagnóstico de fallos en forma de declaración de probabilidad, donde la probabilidad de fallo en el compresor de alta presión se muestra con líneas discontinuas magenta y la probabilidad de fallo en el ventilador con líneas discontinuas cian.

Figura 49

Resultado de las Pruebas de Muestras Aleatorias por Época

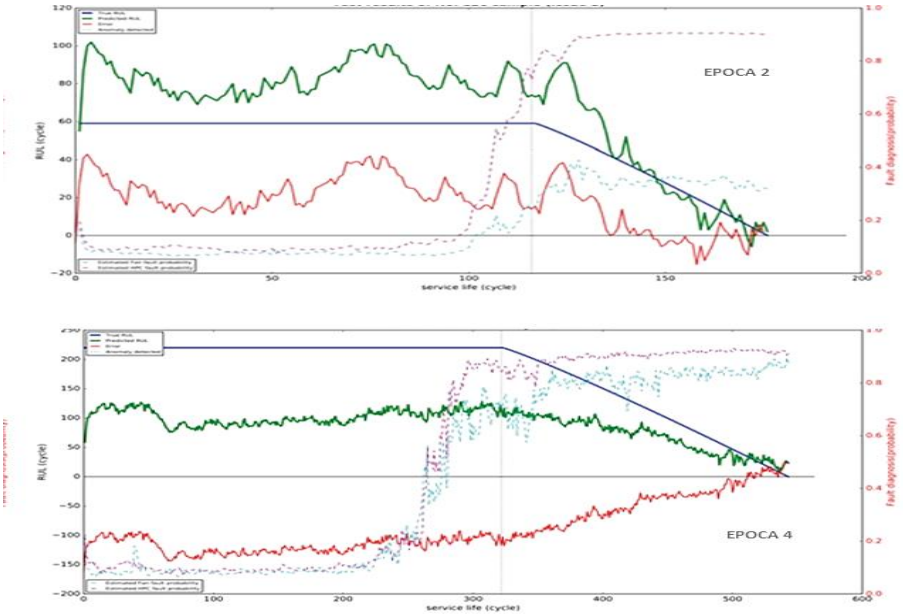
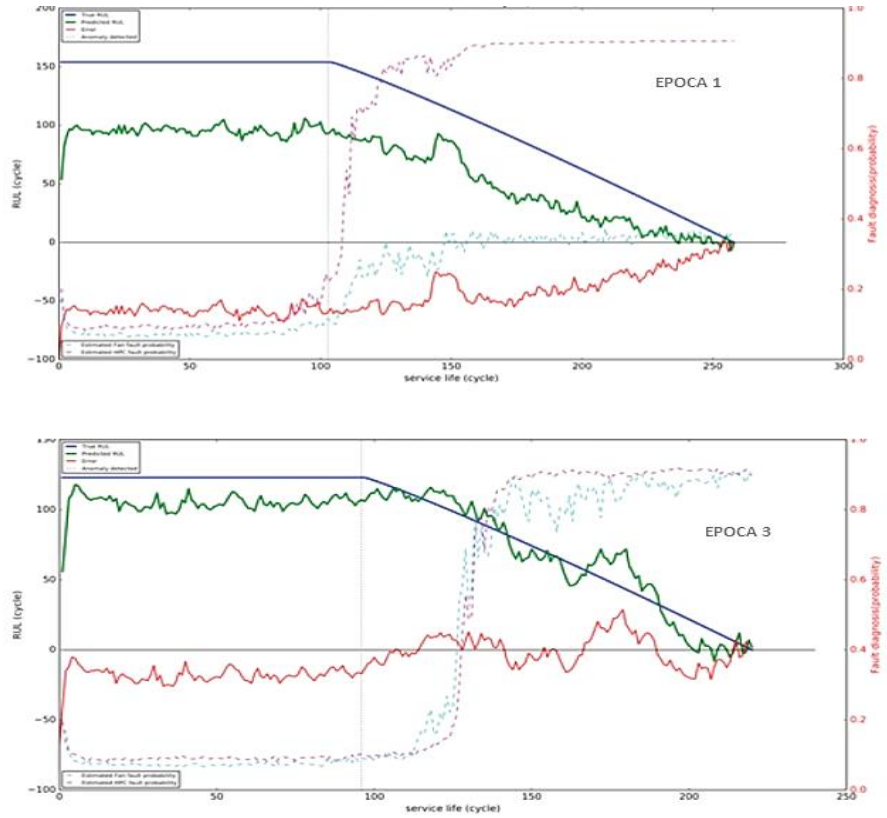


Figura 50

Resultado de las Pruebas de Muestras Aleatorias por Época



Nota. Tomado de Fault Diagnosis and Remaining Usful Life Estimation of Aero Engine Using LSTM Neural Network. October 10-12,2016 Beijing, China. Pag.139

En conclusión, este artículo se presenta el uso de redes neuronales LSTM para abordar el desafío del pronóstico preciso y la gestión de la salud de los motores de aviación, lo que se logra mediante la predicción precisa de la vida residual y la probabilidad de fallos bajo modos de funcionamiento complejos y degradaciones híbridas. Se compara el rendimiento de la LSTM con otras redes neuronales estándar y se demuestra que la LSTM es la más eficaz en los cuatro tipos de problemas de motores turbofán de aviones analizados. El estudio utiliza etiquetas de fallo creadas por un detector de anomalías y se sugiere que el rendimiento se podría mejorar aún más mediante la adición de más muestras de entrenamiento etiquetadas y técnicas de remuestreo para las muestras escasas. La metodología utilizada se podría describir como sigue:

- En primer lugar, los autores recopilaron datos de sensores de un motor aeronáutico de C-MAPSS, y utilizan estos datos para entrenar una red neuronal LSTM. Los datos incluyen mediciones de temperatura, vibraciones, presión, velocidad y otras variables relevantes del motor. Los datos se dividieron en secuencias de tiempo y se utilizaron para entrenar la red neuronal LSTM para que aprenda patrones y correlaciones en los datos.

- Una vez entrenada la red neuronal, se utilizaron conjuntos de datos de prueba para evaluar su capacidad para diagnosticar fallos y estimar la vida útil restante del motor. Los autores compararon su enfoque con otros métodos de diagnóstico y estimación de vida útil, como los métodos de regresión y los métodos basados en redes neuronales convolucionales, y demostraron que su enfoque basado en LSTM ofrece un mejor rendimiento.

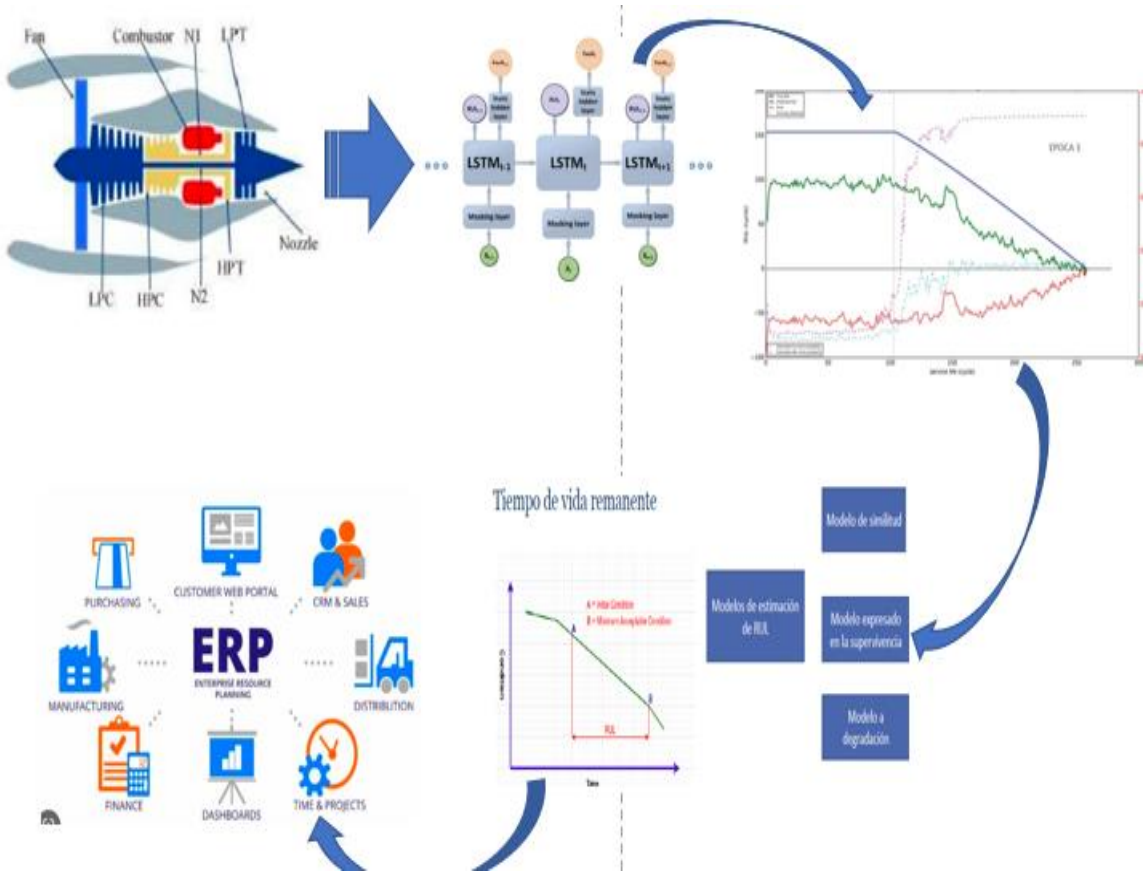
- Para identificar fallos en el motor, los autores utilizaron una técnica llamada "análisis de componentes principales" para reducir la dimensionalidad de los datos de entrada y extraer las características más relevantes. Luego, utilizaron la red neuronal LSTM para clasificar las secuencias de tiempo de entrada y detectar anomalías o patrones que indicaran fallos en el motor.

- Para estimar la vida útil restante del motor, los autores utilizaron la red neuronal LSTM para predecir la tasa de degradación del motor y, a partir de esa predicción, estimar la vida útil restante. Este enfoque permitió a los autores predecir con precisión la vida útil restante del motor y detectar cuándo el motor necesitaba ser reparado o reemplazado.

Se podría representar la arquitectura como se indica en la Figura 51.

Figura 51

Esquema de toma de datos de C-MAPSS y uso de LSTM



2.3.4 Caso 4. Streaming de series temporales sintéticas para la Monitorización Simulada de Condiciones

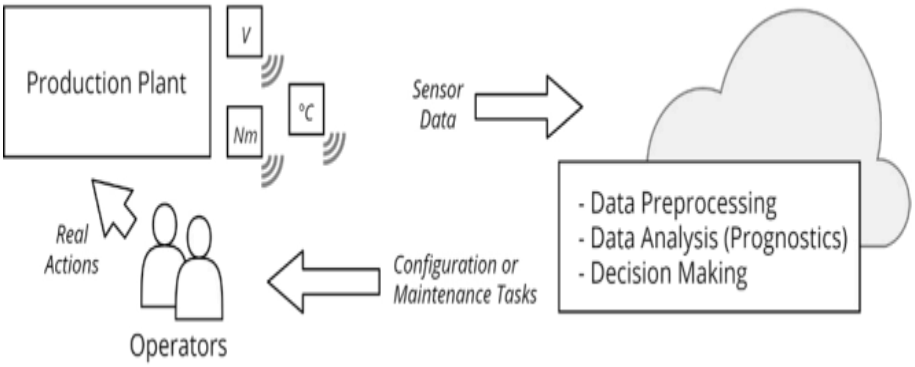
(Jan Zenisek, Josef Wolfartsberger, Christoph Sievi, Michael Affenzeller. "Streaming Synthetic Time Series for Simulated Condition Monitoring", 2018, pág. 643...648)

En este artículo se presenta una herramienta novedosa para la generación y publicación de series temporales de datos de sensores, con el objetivo de simular la monitorización del estado de plantas de producción industrial. Dado que el acceso a datos del mundo real disponibles públicamente es limitado, esta herramienta proporciona una forma de sintetizar series temporales para obtener experiencias en este campo. La herramienta se basa en archivos de configuración y puede actuar como una planta equipada con sensores, generando puntos de datos a partir de diversos modelos basados en Procesos Gaussianos y expresiones matemáticas, o reproduciendo conjuntos de datos. Además, se puede conectar sin problemas con los sistemas circundantes existentes mediante el protocolo de mensajería comúnmente usado en internet de las cosas (Message Queuing Telemetry Transport o MQTT). La herramienta pretende sentar las bases de las aplicaciones del mundo real y mejorarlas proporcionando una herramienta de simulación con la que se pueda crear prototipos y realizar pruebas.

Con la creciente importancia de la tecnología de la información en la industria de producción, se ha generado interés en la transformación de las fábricas en sistemas ciberfísicos para recopilar y analizar datos de producción de manera más eficiente. El mantenimiento predictivo es una estrategia importante en este contexto, ya que permite prevenir averías al monitorear y analizar el estado actual del objeto en tiempo real, en lugar de simplemente seguir intervalos fijos de mantenimiento. Esto puede conducir a ahorros de tiempo y costos, así como a mejoras en la calidad del producto.

Figura 52

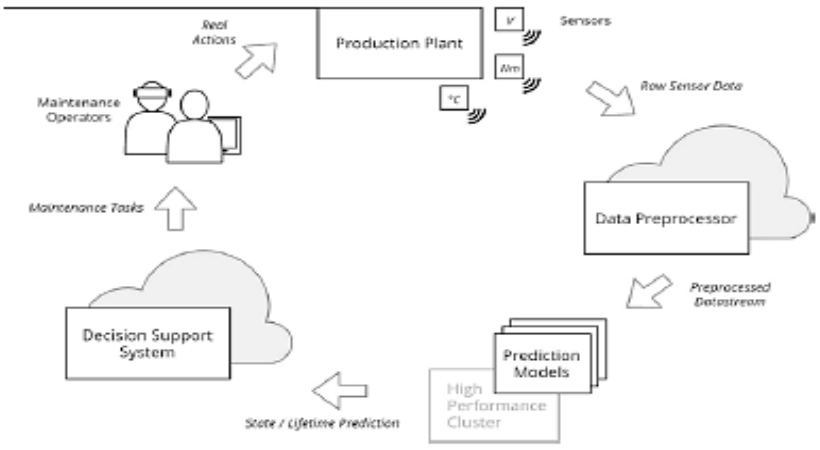
Visión Inteligente de la Producción



Nota. Tomado de Jan Zenisek, Josef Wolfartsberger, Christoph Sievi, Michael Affenzeller, Streaming Synthetic Time Series for Simulated Condition Monitoring, IFAC-Papers On Line, Volume 51, Issue 11, 2018, Pages 643-648

Figura 53

Ciclo inteligente del mantenimiento



Nota. Tomado de Jan Zenisek, Josef Wolfartsberger, Christoph Sievi, Michael Affenzeller, Streaming Synthetic Time Series for Simulated Condition Monitoring, IFAC-Papers On Line, Volume 51, Issue 11, 2018, Pages 643-648

La importancia de la recopilación de datos en la producción y cómo la transformación de la fábrica en un sistema ciberfísico es necesaria para difuminar las fronteras entre el mundo físico y virtual, de modo que los datos puedan colectarse, analizarse y reintegrarse sin fisuras. El potencial de interconectar estrechamente las plantas físicas y la tecnología de la información va desde el ahorro de tiempo, material y costes hasta la mejora de la calidad de los productos, o incluso modelos de negocio innovadores. El mantenimiento predictivo (PdM), que tiene como objetivo prevenir las averías mediante el seguimiento y el análisis del estado actual del objeto, en contraposición a los enfoques habituales de mantenimiento correctivo y preventivo. La fusión de sistemas físicos y virtuales con el apoyo de las TI es una necesidad inevitable para lograr el mantenimiento predictivo, y describe cómo se recopilan y procesan los datos para detectar el desgaste y los posibles defectos. Se destaca los desafíos asociados a la implementación de estos enfoques de industria 4.0 en la práctica, incluyendo la falta de experiencia y la necesidad de datos de sensores realistas y viables para desarrollar y probar estos sistemas presentando una solución a este desafío en forma de una herramienta de software de código abierto que permite la generación de flujos de datos personalizados y realistas para probar y desarrollar soluciones de mantenimiento predictivo.

Diseño. la dificultad de obtener conjuntos de datos de sensores del mundo real disponibles públicamente para simular una planta de producción nos obliga a generar datos sintéticos que sean realistas y se ajusten a las características de las plantas de producción del mundo real. Se establecen las principales características que deben tenerse en cuenta al generar series temporales, incluyendo la estacionariedad, periodicidad, estacionalidad, tendencia, desgaste, daños y desviaciones en las mediciones. Además, se mencionan posibles problemas adicionales que se pueden encontrar en las series temporales, como la falta de valores o las irregularidades.

Enfoques: Los investigadores han ideado diversos enfoques para generar series temporales, que han sido analizados e implementados parcialmente en el generador de flujos de datos. Uno de los enfoques más sencillos consiste en tomar muestras de un proceso gaussiano, utilizando una matriz de covarianza Toeplitz descompuesta Cholesky para generar series de sensores independientes y ruidosos con tendencias y fluctuaciones. Esta matriz es muy utilizada para el tratamiento de análisis de series de tiempo donde se necesita evaluar correlación entre variables. También se ha utilizado la ecuación de Mackey-Glass, una ecuación diferencial no lineal que modela la periodicidad y la dinámica caótica, para generar series temporales. Además, se ha utilizado un modelo de simulación SIMULINK para generar un conjunto de datos de referencia para la predicción de la vida útil restante en una flota de unidades turbo ventilador.

WEKA MOA2, es una herramienta de software para la generación de flujos de datos, pero es limitada en cuanto a la definición de diferentes modelos de series temporales y se centra más en sus capacidades de modelado. El objetivo de este trabajo ha sido determinar

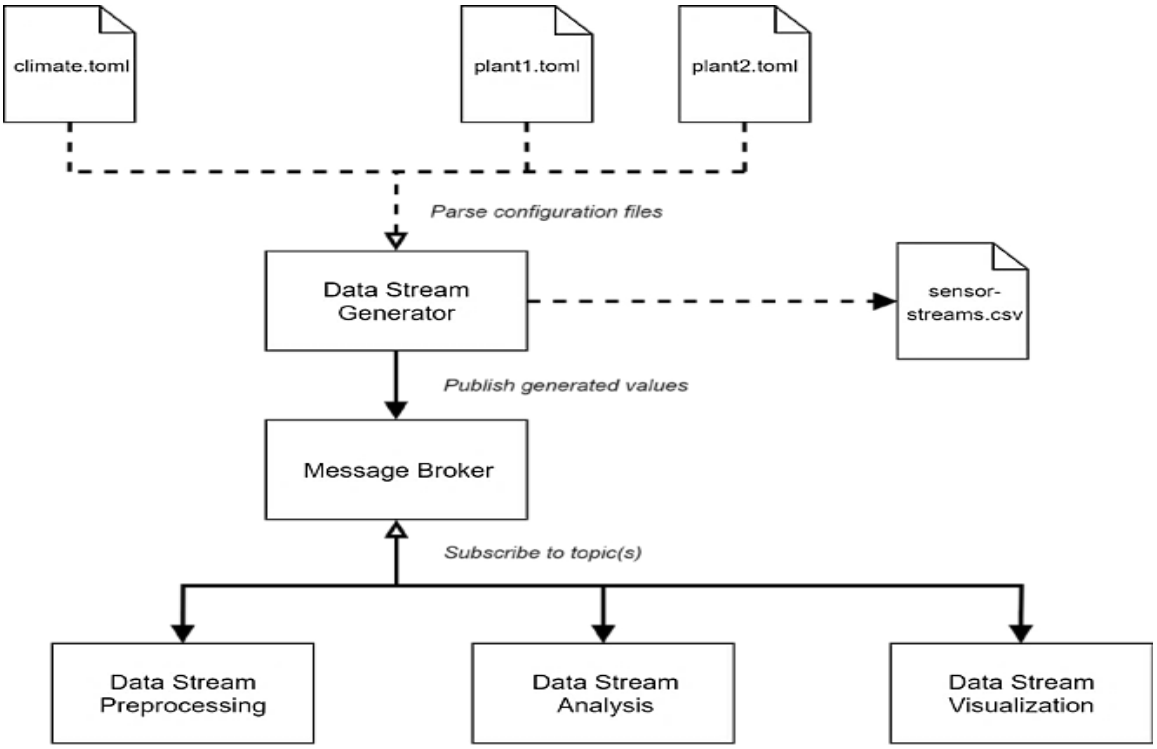
los modelos de series temporales más comunes y revertir el enfoque de modelado para que los modelos previamente entrenados puedan ser utilizados para muestrear, con opciones flexibles de configuración y exportación en tiempo real para conseguir una simulación realista.

El Generador de Flujo de Datos, se describe la implementación de una aplicación en lenguaje de programación “C Sharp” (C#) que analiza archivos de configuración en el inicio y transmite series temporales a través de MQTT a un corredor de mensajes central. La sección se enfoca en el flujo de trabajo del generador y se proporcionan ejemplos creados para producir series temporales aparentemente realistas y presentar los primeros resultados de las pruebas. Cabe destacar que estos ejemplos no provienen del modelado de datos del mundo real. La figura 54 ilustra el flujo de trabajo del generador.

En resumen, la versión actual del generador de flujos de datos admite la definición de modelos AR (Auto - Regresivos) y ARMA (Auto - Regresivos Promedio Móvil), expresiones matemáticas (condicionales) y conjuntos de datos en archivos CSV (Valores Separados por Comas) para reproducirlos. Los modelos AR y ARMA son utilizados para modelar procesos dependientes del tiempo en la economía y la naturaleza, y ambos especifican procesos estocásticos que dependen de sus propios valores pasados y de un término de error estocástico. Mientras que los modelos ARMA son débilmente estacionarios, los modelos AR tienen restricciones de parámetros adicionales. Además, se pueden utilizar expresiones matemáticas personalizadas para modelar el comportamiento periódico o las tendencias, y las dependencias entre las especificaciones de las series individuales pueden modelarse como partes aditivas, basadas en un vector de ponderación de retardo temporal.

Figura 54

Modelo de generación de datos



Nota. Tomado de Jan Zenisek, Josef Wolfartsberger, Christoph Sievi, Michael Affenzeller, Streaming Synthetic Time Series for Simulated Condition Monitoring, IFAC-PapersOnLine, Volume 51, Issue 11, 2018, Pag. 648

El generador de series de tiempo permite agregar ruido en forma de un valor aditivo distribuido normalmente partes aditivas, basadas en un vector de ponderación de retardo temporal.

Para las series ME, MEC y DSR, mientras que los modelos AR y ARMA dependen de un elemento estocástico por definición. También se pueden agregar valores atípicos, pero existen opciones más sofisticadas para partes aditivas, basadas en un vector de ponderación de retardo temporal.

Modelar propiedades distorsionadoras de las series de tiempo, como el trabajo de Saxena et al. (2008).

Arquitectura y flujo de trabajo: La Fig. 55 ilustra el flujo de trabajo de la aplicación y su interacción con otras aplicaciones. Después de analizar las series de tiempo especificadas en los archivos de configuración, se inicializa el generador de flujo de datos y se generan las series de tiempo según sus especificaciones. Los valores pasados y las dependencias entre las series se resuelven almacenando localmente los valores producidos. Las series se exportan valor a valor o colectivamente a un archivo CSV, dependiendo de la opción de exportación global en los ajustes generales. Se proporciona un ejemplo de exportación CSV con los datos de la Figura

55, que muestra series de tiempo que simulan un indicador de degradación y cinco sensores, donde el deterioro influye en la generación de otras series y provoca un cambio de rumbo en una de las series registradas más adelante.

Figura 55

Ejemplo de serie temporal: Simulación de 5 sensores diferentes x1 a x5 y un indicador de deterioro d, alineados con la marca temporal global t.

t	x1	x2	x3	x4	x5	d
1000	3.51	4.22	2.11	7.53	5.23	0.000
1001	3.54	4.20	2.31	7.55	5.24	0.000
1002	3.59	4.17	2.43	7.56	5.26	0.000
...	...	...	...	...	...	...
1051	3.52	4.10	2.84	7.21	4.96	0.014
1052	3.51	4.07	2.87	7.18	4.97	0.022
1053	3.55	4.05	2.21	7.17	4.99	0.029
...	...	...	...	...	...	...

Nota. Tomado de Jan Zenisek, Josef Wolfartsberger, Christoph Sievi, Michael Affenzeller, Streaming Synthetic Time Series for Simulated Condition Monitoring, IFAC-PapersOnLine, Volume 51, Issue 11, 2018, Pag. 648

La exportación CSV es una función valiosa, pero sugiere que la transmisión de datos serie a serie tiene un mayor potencial. Para lograr esto, el generador de flujo de datos establece una conexión con un corredor de mensajes central utilizando el protocolo MQTT. El generador puede bifurcar una tarea ligera para cada serie y generar y publicar al bróker de manera independiente para aprovechar el paralelismo. El protocolo MQTT permite desacoplar las aplicaciones y admitir la mensajería de uno a muchos mediante la implementación del patrón editor-suscriptor a través de temas. Los temas en los que publican los sensores simulados se definen en los archivos de configuración y pueden seguir ciertos esquemas. El uso de RabbitMQ como intermediario para consumir los flujos de datos publicados se discute. El generador de flujo de datos puede actuar como productor y consumidor de series al mismo tiempo, y se discuten varios escenarios de aplicación en la sección siguiente.

Configuración de Condition Monitoring. Se destaca que el generador de flujos de datos es fácil de manejar y tiene un amplio campo de aplicación. Además, se discuten las capacidades de configuración y ampliación del generador, que son habilitadas gracias al uso del protocolo MQTT.

Granularidad del generador. se presentan estrategias típicas para el diseño de la configuración del generador de flujos de datos, utilizando una simulación de una planta de producción y las condiciones ambientales. Se plantean dos topologías diferentes para la configuración del generador. La primera se enfoca en la extensión de la unidad que modela el

generador, lo que implica que cuantos más sensores simule el generador, menos flexible y escalable será. Sin embargo, una configuración más detallada aumenta la complejidad general de la simulación, especialmente en lo que respecta al posterior procesamiento y evaluación del flujo de datos.

La segunda topología se refiere a la implementación de una arquitectura que permite configuraciones en referencia a las tres generaciones de desarrollo de sistemas ciberfísicos, según la clasificación de Hermann et al. (2016). En este caso, el generador podría emplearse para cada sensor individualmente, marcando las tres generaciones con una proporción cada vez mayor de sistemas inteligentes directamente en la planta y menos servicios centralizados. La inclusión de un preprocesador y un componente de razonamiento también se considera en esta topología.

Cliente de visualización. además del generador de flujos de datos, se ha desarrollado una aplicación web que utiliza la biblioteca de gráficos amCharts6 para visualizar en tiempo real las series temporales transmitidas. La aplicación se suscribe a todos los temas MQTT definidos en la cadena URL y muestra los valores de las series en un gráfico correspondiente.

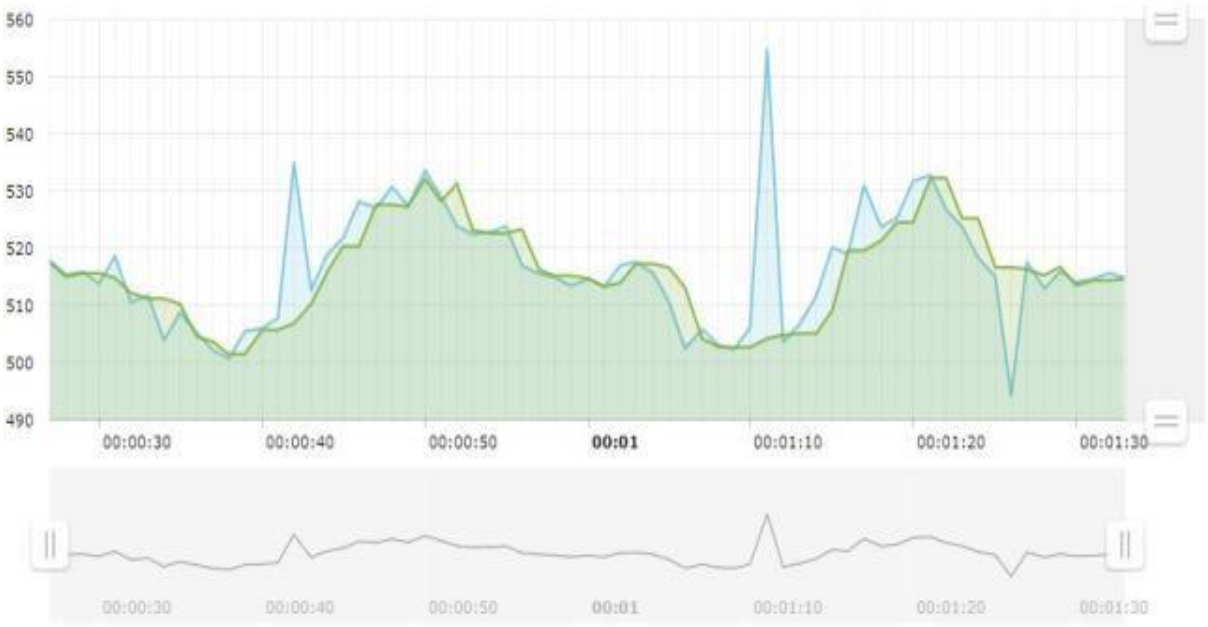
Figura 56

Supervisión del flujo de datos: Vista del cuadro de mandos con varios gráficos de series temporales actualizados constantemente

Nota. Tomado de Jan Zenisek, Josef Wolfartsberger, Christoph Sievi, Michael Affenzeller, Streaming Synthetic Time Series for Simulated Condition Monitoring, IFAC-PapersOnLine, Volume 51, Issue 11, 2018, Pag. 651.

Figura 57

Seguimiento del flujo de datos: Superposición de la versión sin procesar (azul) y preprocesada (verde) de una serie temporal.



Nota. Tomado de Jan Zenisek, Josef Wolfartsberger, Christoph Sievi, Michael Affenzeller, Streaming Synthetic Time Series for Simulated Condition Monitoring, IFAC-PapersOnLine, Volume 51, Issue 11, 2018, Pag. 651.

La visualización de los datos es especialmente útil si el generador de datos reproduce datos registrados o si se integra en un escenario real, ya que puede ayudar a comprender la configuración actual del sensor. Si se adaptan correctamente, los cuadros de mando que muestran los datos proporcionan información valiosa a los operadores y a los responsables de la toma de decisiones, según lo afirmado por Bauernhansl y otros en su trabajo de 2014.

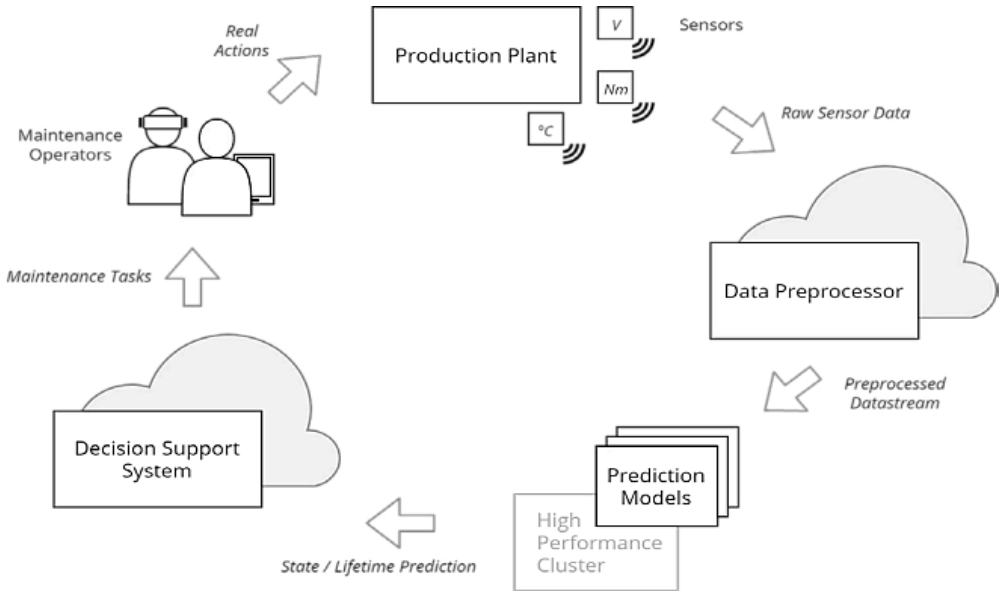
Cliente de pre - procesamiento: Se describe el desarrollo de un prototipo de aplicación que realiza un proceso de preprocesamiento y sumatoria de flujos de datos. La aplicación consolida los flujos y los alinea según fecha y hora, luego filtra los valores atípicos basándose en una ventana deslizante que utiliza la regla del rango intercuartílico (IQR). Los datos preprocesados se publican bajo otro tema, como "preprocesado / planta1 / sensor1". La capacidad de la aplicación web para superponer distintas versiones de una misma serie se ilustra en la figura 57, donde se comparan la serie sin procesar del generador y la versión preprocesada publicada por la aplicación de preprocesamiento. Este enfoque permite mejorar la calidad de los datos y obtener información valiosa para los operadores y responsables de la toma de decisiones.

La idea de mantenimiento inteligente presentada en trabajos anteriores se ilustra en la figura 59. Se menciona que el generador de flujos de datos proporciona un sustituto de la planta de producción visualizada y que se han desarrollado un servicio de preprocesamiento

y una herramienta de visualización para capturar el flujo de datos en un borde arbitrario entre los componentes mostrados. La principal contribución de este trabajo es el concepto propuesto de cómo combinar varios modelos de series temporales para simular escenarios de monitorización de condiciones, implementado como una solución de software altamente configurable para transmitir datos sintéticos. El objetivo a largo plazo es sustituir el generador de flujos de datos por flujos procedentes de una planta real, conectar los componentes de preprocesamiento y predicción con un sistema de apoyo a la toma de decisiones y cerrar el ciclo aplicando las decisiones basadas en datos en la planta con la ayuda de operadores humanos. El trabajo futuro se centrará en ampliar las funciones y evaluar las capacidades de la aplicación, así como en la implementación de un componente de predicción y en la detección de la deriva conceptual utilizando modelos.

Figura 58

Trayectoria de los datos de series temporales desde la detección en la planta de producción hasta el preprocesamiento, la predicción, la toma de decisiones y la vuelta al punto de partida a través de la tecnología de realidad aumentada utilizada por operadores humanos



Nota. Tomado de Jan Zenisek, Josef Wolfartsberger, Christoph Sievi, Michael Affenzeller, Streaming Synthetic Time Series for Simulated Condition Monitoring, IFAC-PapersOnLine, Volume 51, Issue 11, 2018, Pag. 651.

2.3.5 Caso 5. Predicción de la vida útil restante de la bomba de engranajes basado en Deep Convolutional Auto-encoder DCAE y Bidireccional Long Short-Term Memory (Bi-LSTM)

(Zhang P.; Jiang, W.; Shi, X; Zhang, S. "Remaining Useful Life Prediction of Gear Pump Based on Deep Spare Autoencoders and Multilayer Bidirectional Long – Short Term Memory Network", 2022)

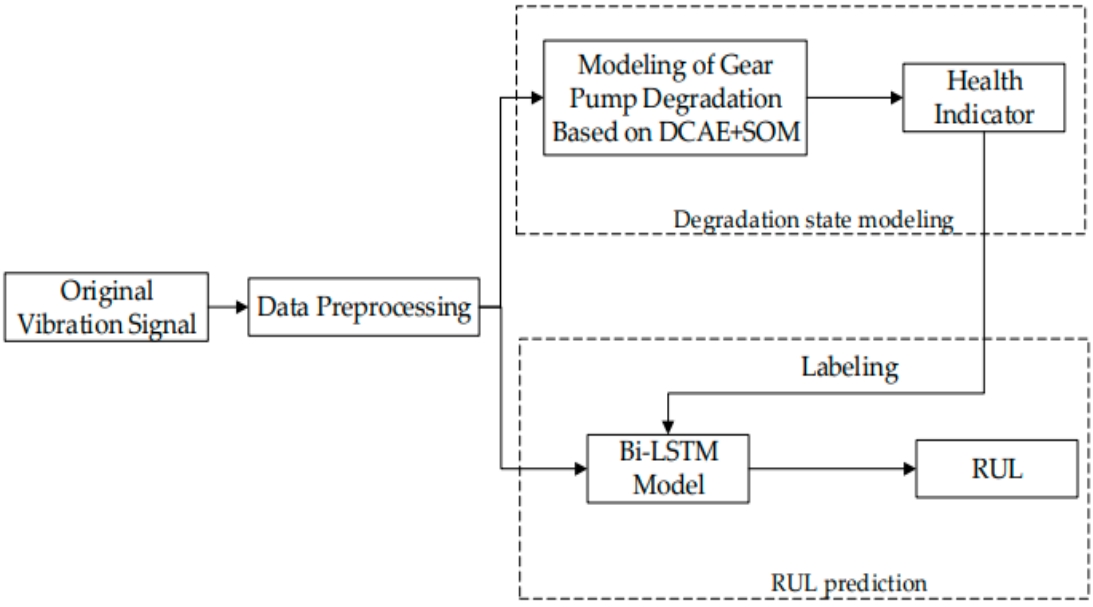
En el caso de las bombas de engranajes, se han desarrollado diversos métodos de predicción de vida útil, entre los que destaca el basado en Deep Convolutional Auto-encoder (DCAE) y Bidireccional Long Short-Term Memory (Bi-LSTM).

El modelo DCAE es una red neuronal profunda que utiliza técnicas de convolución para aprender características de los datos y reducir su dimensionalidad. Por otro lado, el modelo Bi-LSTM es una variante de las redes LSTM que permite la predicción de secuencias de datos en ambas direcciones, lo que mejora su precisión en la predicción de la vida útil restante.

La metodología utilizada. en el método de predicción de vida útil restante de la bomba de engranajes basado en DCAE y Bi-LSTM, se basa en la adquisición de datos de sensores de vibración y corriente eléctrica, los cuales son preprocesados para eliminar ruido y outliers (causados por errores de medición, errores de entrada de datos, comportamientos extremos o simplemente por casualidad). Posteriormente, se utilizan técnicas de reducción de dimensionalidad para obtener características relevantes de los datos. Ver Figura 60.

Figura 59

Esquema de Predicción RUL de la Bomba de Engranajes



Nota. Tomado de Remaining Useful Life Prediction of Gear Pump Based on Deep Spare Autoencoders and Multilayer Bidirectional Long – Short Term Memory Network Pag 4

El siguiente paso es el entrenamiento del modelo DCAE, el cual se encarga de la extracción de características de los datos y su reducción a una representación de menor dimensión. Esta representación se utiliza como entrada para el modelo Bi-LSTM, que se encarga de la predicción de la vida útil restante de la bomba de engranajes.

Un estudio reciente realizado por Zhang et al. (2020) evaluó el rendimiento del método de predicción de la vida útil restante de la bomba de engranajes basado en DCAE y Bi-LSTM. Los resultados mostraron que este método logró una precisión de predicción superior al 90%, lo que demuestra su efectividad en la predicción de la vida útil restante de la bomba de engranajes.

En conclusión, el método de predicción de la vida útil restante de la bomba de engranajes basado en DCAE y Bi-LSTM es una técnica efectiva que permitirá al departamento de mantenimiento de una Planta, gestionar la planificación del mantenimiento preventivo y la reducción de los costos de operación en la industria en particular. Además, su alto rendimiento en la precisión de predicción lo convierte en una herramienta valiosa para los expertos en mantenimiento en la toma de decisiones en la gestión de activos.

La metodología usada para la predicción de la vida útil restante de la bomba de engranajes basado en DCAE y Bi-LSTM consta de los siguientes pasos:

- Adquisición de datos: Se adquieren datos de sensores de vibración y corriente eléctrica de la bomba de engranajes en operación.
- Preprocesamiento de datos: Se realiza un preprocesamiento de los datos para eliminar ruido y valores que son muy diferentes del resto de los valores en un conjunto de datos (outliers), utilizando técnicas como filtrado, normalización y eliminación de valores atípicos.
- Reducción de dimensionalidad: Se utilizan técnicas de reducción de dimensionalidad, como Análisis de Componentes Principales (PCA), para obtener características relevantes de los datos y reducir su dimensión.
- Entrenamiento del modelo DCAE: Se entrena un modelo DCAE utilizando los datos preprocesados y reducidos de dimensionalidad. El modelo DCAE se encarga de la extracción de características de los datos y su reducción a una representación de menor dimensión.
- Entrenamiento del modelo Bi-LSTM: Se entrena un modelo Bi-LSTM utilizando la representación de menor dimensión obtenida del modelo DCAE como entrada. El modelo Bi-LSTM se encarga de la predicción de la vida útil restante de la bomba de engranajes.
- Validación del modelo: Se valida el modelo utilizando datos de prueba para evaluar su precisión en la predicción de la vida útil restante de la bomba de engranajes.
- Implementación del modelo: Una vez validado el modelo, se implementa en la operación de la bomba de engranajes para realizar la predicción de la vida útil restante en tiempo real.

En resumen, el método de predicción de la vida útil restante de la bomba de engranajes basado en DCAE y Bi-LSTM se basa en la adquisición de datos de sensores, preprocesamiento

de datos, reducción de dimensionalidad, entrenamiento de los modelos DCAE y Bi-LSTM, validación del modelo y finalmente, la implementación del modelo para la predicción en tiempo real.

2.3.6 Caso 6: Predicción de la vida útil restante de la Batería de iones de litio con la técnica del filtro de partículas sin Olor

(Qiang Miao, Lei Xie, Hengjuan Cui, Wei Liang, Michael Pecht. "Remaining useful life prediction of lithium-ion battery with unscented particle filter technique, Microelectronics Reliability", 2013, pág. 805...810)

Las baterías de iones de litio son ampliamente utilizadas en diversos sistemas, desde dispositivos portátiles hasta vehículos automotores y sistemas aeroespaciales, debido a su ligereza, alta densidad de energía y larga vida útil en comparación con las baterías tradicionales.

Sin embargo, la funcionalidad de las baterías de iones de litio puede deteriorarse con el tiempo, lo que puede causar una reducción en el rendimiento, pérdidas económicas e incluso fallas catastróficas. Por lo tanto, es importante monitorear el estado de la batería para estimar su estado de carga y su estado de salud, lo que proporciona información útil para la gestión de la batería.

La estimación del estado de carga y del estado de salud de la batería se ha abordado con diversos métodos, como mediciones de impedancia, el uso del filtro de Kalman y la tecnología de medición de espectroscopia de impedancia electroquímica. El estado de carga brinda información sobre la energía restante de la batería a corto plazo, mientras que el estado de salud describe el estado general de la batería a largo plazo y su capacidad para proporcionar el rendimiento especificado. La vida útil restante de la batería se puede predecir mediante la extrapolación de mediciones de estado de salud y un umbral de falla.

El filtro de partículas (PF) es un método eficaz para el procesamiento de señales secuenciales, utilizado en muchas áreas, como la visión artificial, el seguimiento de objetivos y la robótica. Se ha aplicado en la predicción de la vida útil restante de las baterías de iones de litio, pero su precisión no es alta. En el texto se presenta un filtro de partículas sin olor (UPF) con un algoritmo PF mejorado para la predicción de la vida útil restante de las baterías de iones de litio. El modelo de degradación se basa en la comprensión de las baterías de iones de litio y los resultados de la predicción se pueden obtener utilizando el modelo de degradación y los algoritmos UPF. Según los resultados del análisis, se puede ver que UPF puede predecir la vida útil restante real de la batería con un error inferior al 5%.

Respecto a la metodología utilizada se tiene:

- Se describe el algoritmo del filtro de partículas (PF) y el algoritmo del filtro de partículas sin olor (UPF) por separado.

- Se construye un modelo de degradación basado en la comprensión de las baterías de iones de litio.
- Se utilizan los algoritmos UPF y el modelo de degradación para predecir la vida útil restante de las baterías de iones de litio.
- Se comparan los resultados de la predicción con el valor real para evaluar la precisión del algoritmo UPF.
- Se discuten los resultados y se concluye que el algoritmo UPF puede predecir la vida útil restante real de las baterías de iones de litio con un error inferior al 5%.

El artículo no menciona específicamente una arquitectura utilizada en el desarrollo del filtro de partículas sin olor (UPF) propuesto para la predicción de la vida útil restante de la batería, sin embargo, describe cómo se ha mejorado el algoritmo del filtro de partículas (PF) original para lograr una mayor precisión en la predicción del tiempo restante de vida de la batería de iones de litio.

UNIVERSITAS STUDIORUM PIURENSIS

Capítulo 3

Análisis de Metodologías de Mantenimiento 4.0 usando IA

Se describe a continuación una comparación de las arquitecturas y metodologías mencionadas en cada uno de los casos mencionados en el capítulo 2, con el objeto de extraer lo que puede aplicar al presente trabajo de investigación.

3.1 Discusión de Metodologías

Los casos tratan aplicaciones de inteligencia artificial para estimar la vida útil restante de componentes industriales críticos. Sin embargo, los arquitecturas y metodologías utilizados son diferentes debido a las características específicas de cada activo.

En el caso 1, se trata de una bomba de lodos industrial, por lo que los principales parámetros monitoreados son: vibración, temperatura, y el flujo de la bomba. Estos datos se preprocesan y se seleccionan las características relevantes para desarrollar un modelo de predicción. Se pueden utilizar diferentes técnicas de modelado, como modelos de regresión, árboles de decisión, redes neuronales o modelos basados en el aprendizaje automático. El modelo se valida y se despliega en línea para proporcionar predicciones en tiempo real de la vida útil restante de la bomba de lodos.

En el caso de disco de álabes de turbina de gas (Caso 2), se evalúan las condiciones de carga y las propiedades materiales del disco de álabes. Se utiliza software de modelado por elementos finitos para simular el comportamiento del disco de álabes bajo diferentes cargas y temperaturas. También se realizan pruebas no destructivas para detectar posibles defectos o grietas en el material del disco. Se utilizan modelos de fatiga para predecir la vida útil del disco bajo diferentes condiciones de carga y temperatura. Los resultados del análisis de fatiga se comparan con los requisitos de vida útil especificados por el fabricante del disco. Si la vida útil estimada es menor que los requisitos, se pueden tomar medidas para reparar o reemplazar el disco. Además, si se identifican problemas de integridad estructural o vida útil durante el análisis, se pueden tomar medidas para mejorar el diseño de la turbina o la operación del proceso.

En el caso 3, es un poco más completo y además de diagnosticar fallos, se estima la vida útil remanente. Se trabaja con data real de sensores y data recopilada del modelo C-MAPSS. Se utiliza una red neuronal LSTM. El modelo se entrena con datos históricos y se utiliza

en tiempo real para monitorear el estado del motor y proporcionar alertas y notificaciones en caso de fallos o problemas potenciales.

El caso 4, más que una metodología, describe una herramienta novedosa para la generación de series temporales de datos de sensores, utilizando archivos de configuración para sintetizar series temporales para simular la monitorización del estado de plantas de producción industrial. La herramienta se basa en diversos modelos basados en procesos gaussianos y expresiones matemáticas para generar puntos de datos y puede conectarse con los sistemas circundantes existentes mediante el protocolo MQTT. El mantenimiento predictivo es una estrategia importante en este contexto, y la herramienta proporciona una solución para el desafío de obtener datos de sensores realistas y viables para desarrollar y probar estos sistemas. Los investigadores han ideado diversos enfoques para generar series temporales, como la matriz de covarianza Toeplitz, la ecuación de Mackey-Glass y el modelo de simulación SIMULINK. Se establecen las principales características que deben tenerse en cuenta al generar series temporales, incluyendo la estacionariedad, la periodicidad, la estacionalidad, la tendencia, el desgaste, los daños y las desviaciones en las mediciones.

En general, excluyendo al caso 4, cada caso presenta diferentes desafíos y requerimientos específicos, lo que conduce a diferentes enfoques y metodologías para abordar el problema de la estimación de la vida útil restante de componentes industriales críticos mediante el uso de IA.

Como recomendación, es importante tener en cuenta la validez y la representatividad de los datos sintéticos generados, y se debe verificar la calidad y la precisión de los modelos utilizados para generar los datos. Además, es recomendable comparar los resultados obtenidos utilizando datos sintéticos con los datos reales disponibles, para asegurarse de que las simulaciones se ajusten a las características de las plantas de producción del mundo real.

3.2 Comparación de Arquitecturas Utilizadas

Para el Caso 1, el proceso comienza con la recopilación de datos de los sensores instalados en la bomba de lodos para medir variables relevantes como la vibración, la temperatura y el flujo de la bomba, y la inclusión de datos de mantenimiento previo, como el historial de reparaciones y reemplazos de componentes. Luego, los datos recopilados son preprocesados y normalizados para eliminar cualquier ruido o error y asegurarse de que los datos sean comparables en el tiempo.

Luego, se desarrolla un modelo de aprendizaje automático para predecir la vida útil restante de la bomba de lodos en línea, utilizando diferentes técnicas de modelado como modelos de regresión, árboles de decisión, redes neuronales, entre otros. Este modelo es validado utilizando datos históricos y en tiempo real para evaluar su precisión y generalización, y se pueden utilizar técnicas de validación cruzada para garantizar la robustez del modelo.

En resumen, la arquitectura utilizada en el marco propuesto por Mohskin (2022) es una combinación de técnicas de recopilación de datos, preprocesamiento, selección de características, modelado de aprendizaje automático, validación del modelo y despliegue en línea para proporcionar predicciones en tiempo real. Esta arquitectura sigue un enfoque bien definido para el mantenimiento predictivo y la gestión de activos, lo que puede mejorar la eficiencia y la seguridad en las operaciones industriales.

Cuando estamos en la etapa de diseño del proceso, equipo, etc. se puede elegir la arquitectura que utilizaremos para nuestro caso en particular, sin embargo, si lo que intentamos hacer es mejoras a implementar en nuestra planta o proceso, lo más probable es que ya dispongamos de una arquitectura de toma de datos mediante sensores, transmisión de los mismos hasta un servidor o sistema Scada, de forma que nos tocaría diseñar nuestro gemelo digital desde la base de datos que disponemos o la generación de data sintética, hacia el procesamiento en la nube y el diseño de nuestro gemelo digital, dependiendo de lo que queremos obtener.

UNIVERSITAS STUDIORUM PIURENSIS

Capítulo 4

Metodología de Mantenimiento 4.0 con IA – Caso real

Para el desarrollo de este trabajo de investigación, se evaluará una de las bombas que suministran agua tratada para el proceso de una planta industrial, cuyo requerimiento es:

Flujo másico: 19,044.47 kg/h

Presión: 7.1 kg/cm2

Temperatura: 27°C

Flujo Volumétrico: 19,121 m3/h

Entalpía específica: - 3768 Kcal/kg

Gravedad específica: 0.996

Viscosidad: 0.85 cP

El agua se utiliza para el enfriamiento de los procesos dentro de la planta y por ende es vital que principalmente la temperatura, el flujo másico y la presión sean las requeridas por el proceso, sin desmerecer las otras características. Este sistema cerrado trabaja con 03 electrobombas operando simultáneamente y 01 en Stand by.

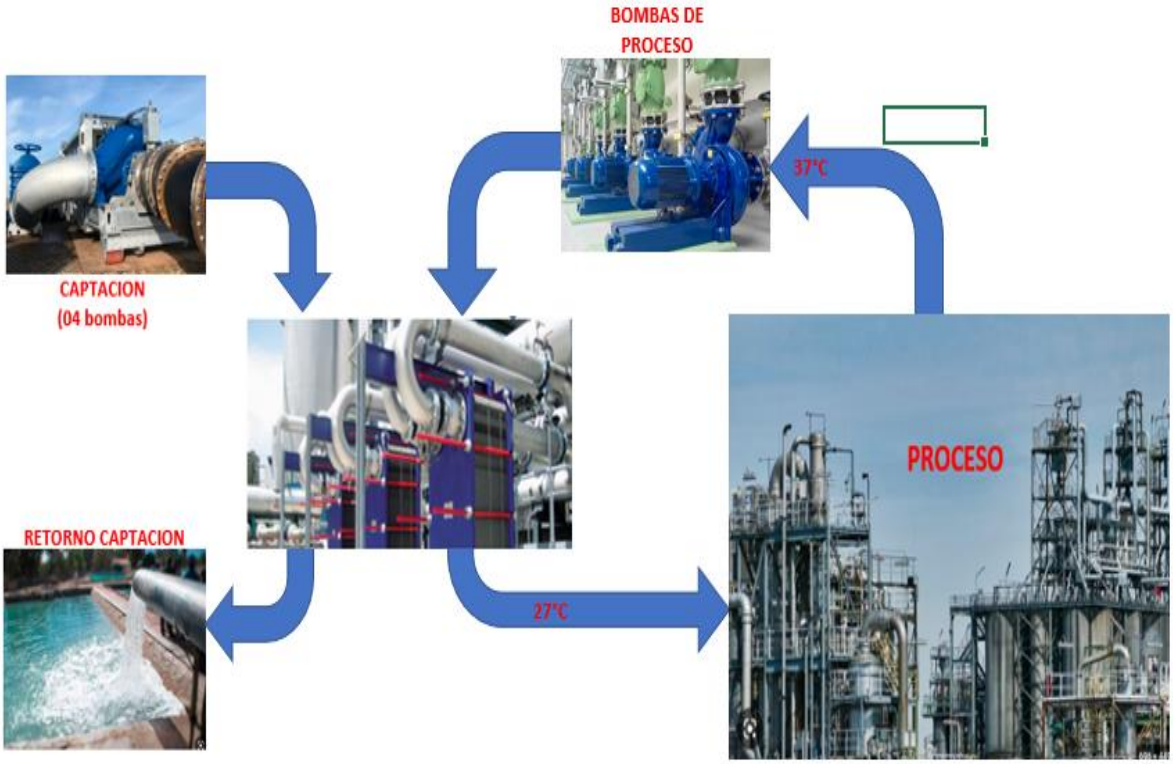
El agua retorna del proceso a 37°C, por lo que es necesario bajar dicha temperatura hasta 27°C antes de ser reinyectada nuevamente al proceso. Para disminuir los 10°C, es necesario un intercambiador de calor de placas cuyo fluido enfriador también es agua, procedente de un circuito abierto (agua de mar). Este circuito también consta de 03 electrobombas operando simultáneamente y 01 en Stand by. En la Figura 61 se esquematiza el sistema de enfriamiento de agua de proceso.

Si una de las 03 bombas que están operando falla, ya sea del circuito cerrado o del circuito abierto, el sistema se desequilibra y los subsistemas de enfriamiento en la planta no trabajarán eficientemente activando paradas de emergencia en los subprocesos. Cuando esto sucede se pone a operar la bomba en stand by, sin embargo, el tiempo aproximado para llegar nuevamente a las condiciones normales de operación toma entre una a dos horas y durante este transitorio hay pérdidas de producción por calidad de producto o reprocesos, que se traducen finalmente en gastos operativos, riesgos para la operación, multas, etc.

El objetivo de esta investigación es predecir la falla de la bomba para que la parada sea programada, de forma que el transitorio del sistema sea lo más corto posible o casi imperceptible.

Figura 60

Sistema de Enfriamiento de Agua de Planta de Proceso



4.1 Descripción del Activo por analizar (electro – bomba)

El activo por analizar corresponde a una de las bombas de agua del sistema de enfriamiento del agua de proceso.

4.1.1 Bomba

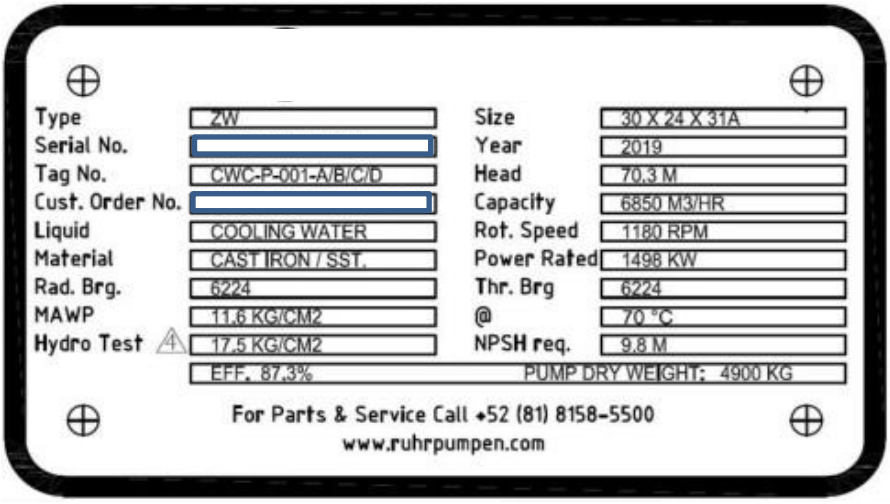
Es una bomba centrífuga horizontal, de una sola etapa, doble succión, voluta, con carcasa axialmente bipartida. Estas bombas están diseñadas para el servicio continuo.

- Tamaño: ZW 30x24x31 (A)
- Etapas: 1
- Según número de la curva: OKH 122.084.10
- Caudal nominal: 6850 (6174) m3/h
- Velocidad: 1180 RPM
- Liquido: Agua
- Diámetro máximo de sólidos; 0.00 mm

- Potencia hidráulica: 1307 (1285) kW
- Potencia nominal: 1498 (1487) kW
- Soporte del eje: Sobre rodamientos (uno en el lado libre y otro en el lado acoplamiento), ambos rodamientos son abiertos, lubricados por aceite.

Figura 61

Placa de la Bomba de Proceso



Nota. Tomado del Manual del equipo

4.1.2 Motor Eléctrico

La Bomba tiene acoplado un motor eléctrico de inducción de media tensión tipo Jaula de Ardilla.

- Potencia de salida: 1678 kW.
- Número de Polos.: 6.
- Frecuencia: 60HZ.
- Voltaje: 4000 Voltios.
- Amperaje a plena carga: 296.9 A
- Factor de Potencia: 85%.
- Clase de aislamiento: F.
- Tipo de Ventilación: TEAAC (Totalmente cerrado con enfriamiento aire - aire).
- Grado de Protección: IP55.
- Factor de Servicio: 1.
- Máxima Temperatura Ambiente: 40°C.

- Velocidad: 1188 RPM.
- Peso del Motor: 8300 Kg.
- Tipo de Rotor: Jaula de Ardilla.
- Rendimiento: 96%.
- Rating: S1.
- Instalación: Horizontal.
- Protección contra sobretensiones transitorias: Descargadores de sobretensión

Figura 62

Placa del Motor de la Bomba de Proceso

Nota. Tomado del Manual del equipo

Tanto la bomba como el motor eléctrico, cuentan con sensores de vibración, sensores de temperatura, sensor de velocidad y otros parámetros no tan importantes para el presente estudio. Pero si para el proceso.

4.1.3 Sensores de vibración

Se emplean dos tipos de sensores de vibración:

- Sensor de vibración (Vibration sensor - velometer), marca Bently Nevada, modelo 177230-01-01-05, sensibilidad del bucle principal: 0 a 25.4 mm/s ∓ 10% FS.

- Sensor de vibración (Probe, Key-Phasor, Proximity). marca Bently Nevada, modelo 31000-16-10, conexión roscada ¾”, temperatura de trabajo de -34°C a 105°C, Probeta modelo 3300XL 8 mm Probe.

El primer tipo o modelo se encuentra instalado en el motor eléctrico y mide las vibraciones en el rodamiento del lado del acoplamiento tanto en el eje horizontal como vertical. Estos dos sensores instalados se encuentran identificados como: CWCVI003X y CWCVI003Y respectivamente (ver figura 64). El lado libre del motor no cuenta con sensores de vibración.

Figura 63

Sensor de vibración (Vibration sensor - velometer)

Nota. Tomado del Manual del equipo

En la bomba, también se tiene cuatro sensores de este tipo:

En el lado acoplamiento para el registro en el eje horizontal y vertical, identificados con el TAG: CWCVI001X, CWCVI001Y, y en el lado libre de la bomba, registra vibraciones tanto horizontalmente como verticalmente. TAG: CWCVI002X, CWCVI002Y

El segundo tipo de sensores de vibración se encuentran instalados en lado del acoplamiento y lado libre desfasados 90° uno del otro tal como se ilustra en la figura 64.

Figura 64

Sensor Pxoximity

Nota. Tomado del Manual del equipo

4.1.4 Sensores de Temperatura

Los sensores de temperatura están instalados en los rodamientos de la electro - bomba y tienen la siguiente característica:

- Tipo: RTD Dual
- Resistencia: 100 ohm a 0°C
- Rango de temperatura: 0 a 150 °C
- Material Acero inoxidable 316L
- O.D. 6mm
- Rosca de montaje 1/” NPT
- Longitud del elemento: 50 mm

En el presente trabajo, estos detectores de temperaturas están identificados como:

- Para la bomba “A”
- CWCTI001A.PV: Lado libre de la bomba
- CWCTI002A.PV: Lado libre de la bomba
- CWCTI003A.PV: Lado acoplamiento de la bomba
- CWCTI004A.PV: Lado acoplamiento de la bomba

El siguiente esquema nos refleja el fujo de información recolectada en campo hasta su almacenamiento en los servidores locales. A partir de este punto tomamos los datos, para nuestro caso de la temperatura de rodamientos. Predecir la temperatura de los rodamientos a lo largo del tiempo significa predecir la vida útil de los mismos dado que cualquier modo de falla en los mismos se manifestará como un incremento en la temperatura.

4.1.5 Descripción de la Arquitectura del Sistema

En la Figura 62 se muestra la arquitectura de control e instrumentación de la planta está diseñada para permitir el monitoreo y control de los procesos en tiempo real. Para lograr esto, se utilizan sensores de campo conectados a la Sala de los gabinetes de Instrumentación y control. Estos sensores envían señales en 4 a 20 mA, que se utilizan para medir variables como la vibración, la temperatura, la velocidad, el caudal y la presión.

En la Sala de los gabinetes de Instrumentación y control, las señales que se reciben de los sensores se convierten en valores de vibración y temperatura, se envían al Sistema de Control Distribuido (DCS) en la Sala de Control. El DCS es un sistema de control avanzado que permite el monitoreo y control de los procesos en tiempo real. También tiene una alta capacidad de almacenamiento, lo que significa que puede almacenar grandes cantidades de datos de proceso.

Todas las tendencias y comportamientos que muestra la instrumentación asociada se registran en el módulo de historia (HM). Esto significa que toda la información del proceso se guarda y se puede usar para el análisis y la mejora continua del proceso.

Para permitir la comunicación entre las estaciones de operación y los gabinetes de control, se utiliza un módulo de interfaz de red que utiliza protocolos propiedad del fabricante. Esta red de comunicación permite que los datos que vienen de campo lleguen y salgan directamente del controlador.

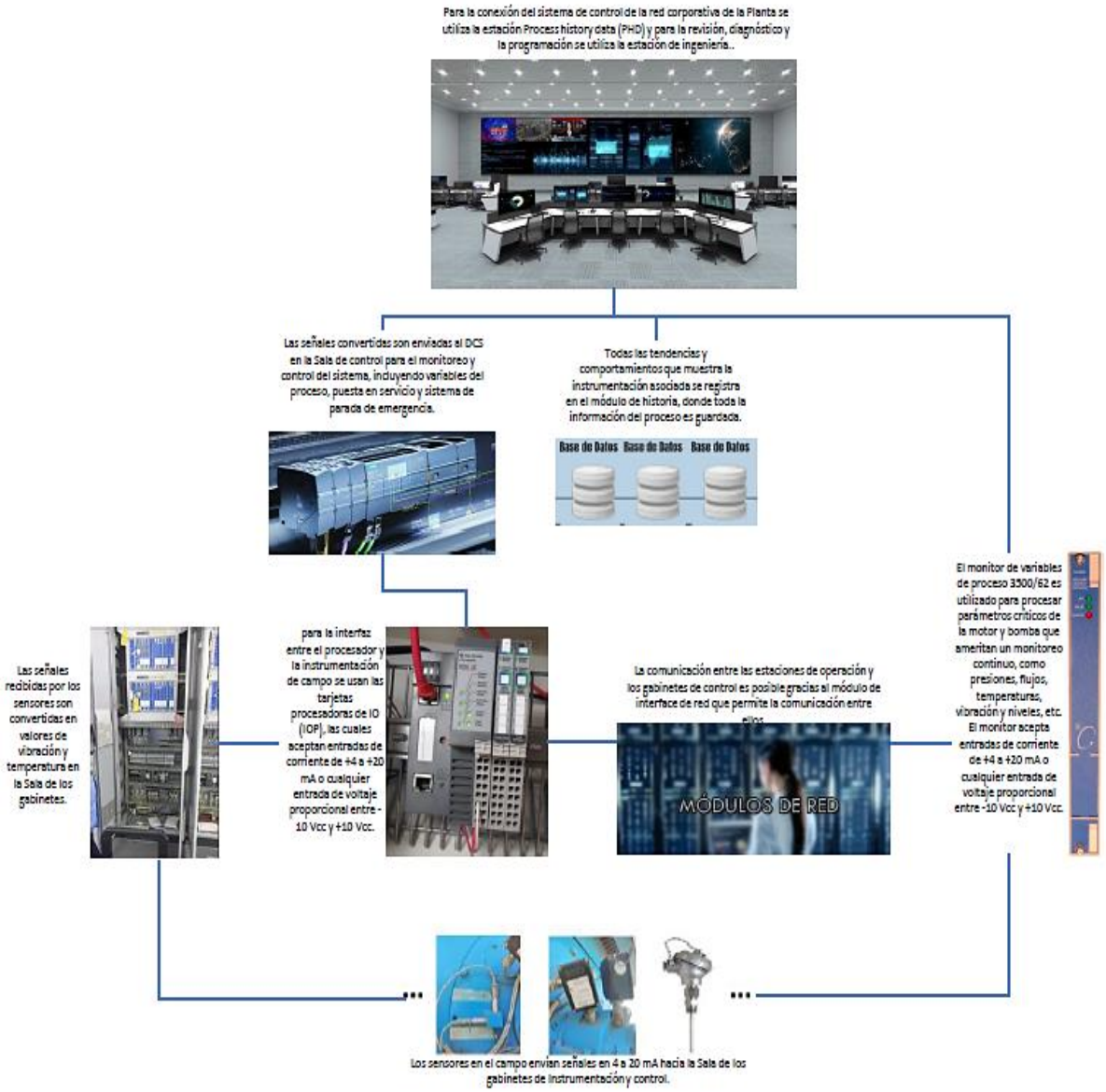
La interfaz entre el procesador y la instrumentación en campo se utilizan las tarjetas procesadoras de IO (IOP). Estas tarjetas, comúnmente usadas, están diseñadas para manejar señales análogas y digitales, lo que significa que pueden conectarse directamente a los equipos de instrumentación de campo, como transmisores, válvulas y contactos.

Entre los monitores de variables de proceso, el 3500/62, que es capaz de procesar parámetros críticos del motor y la bomba como: vibración, temperatura, presión, flujo y otros. Este monitor acepta entradas de corriente de +4 a +20 mA o cualquier entrada de voltaje proporcional entre -10 V CC y +10 V CC. También condiciona estas señales y compara las señales acondicionadas con puntos de ajuste de alarma programables por el usuario.

Para la conexión del sistema de control a la red corporativa de la planta se utiliza la estación *Process history data* (PHD) y para la revisión, diagnóstico y programación se utiliza la estación de ingeniería. Esto permite que los ingenieros y otros miembros del personal puedan acceder a los datos de proceso y realizar diagnósticos y análisis para mejorar continuamente el proceso.

Figura 65

Arquitectura del Sistema



4.2 Aplicación de la metodología técnica y obtención de datos

Como se ha descrito anteriormente la electro-bomba de agua de proceso es un equipo crítico para el funcionamiento de la planta, por lo que su monitoreo constante es fundamental con el fin de garantizar una operación eficiente y segura. Este equipo cuenta con un total de

19 sensores que monitorean diferentes parámetros en tiempo real, incluyendo la vibración, temperatura y revoluciones por minuto (RPM).

El motor eléctrico es uno de los componentes importantes de la electro-bomba y cuenta con dos sensores de vibración (ver numeral 4.2.2) en el lado del acoplamiento, en ambos ejes horizontal y vertical. Cuatro sensores de proximidad adicionales, dos en el lado del acoplamiento y dos en el lado libre, monitorean también las vibraciones del eje, anticipándose a las fallas por desgaste de rodamientos, desalineamiento, etc. y cuatro sensores de temperatura, dos en el lado libre y dos en el lado del acoplamiento con la bomba, registran la temperatura de los rodamientos para detectar cualquier incremento anormal de la temperatura de los rodamientos.

La bomba también cuenta con 4 sensores de vibración distribuidos en el lado libre y lado del acoplamiento con el motor para detectar cualquier vibración no deseada. Además, cuenta con otros 4 sensores de temperatura distribuidos de la misma forma que los sensores para detectar cualquier incremento anormal de la temperatura.

Aunque el equipo cuenta con 4 sensores para cada parámetro monitoreado, se ha notado un ligero incremento de temperatura en los rodamientos del motor eléctrico, lo que nos ha llevado a enfocar este desvío a la elaboración de una red neuronal para calcular el RUL del rodamiento en base al manifiesto de su degradación como incremento de temperatura. Esto demuestra la importancia de estos sensores y su monitoreo constante para detectar problemas tempranos y prevenir fallas catastróficas en el equipo.

En la Figura 66 se muestra un esquema con la ubicación de los sensores en la electro-bomba. En la Tabla 6. se detalla la descripción de cada uno de los 19 sensores.

Tabla 6

Descripción de los sensores

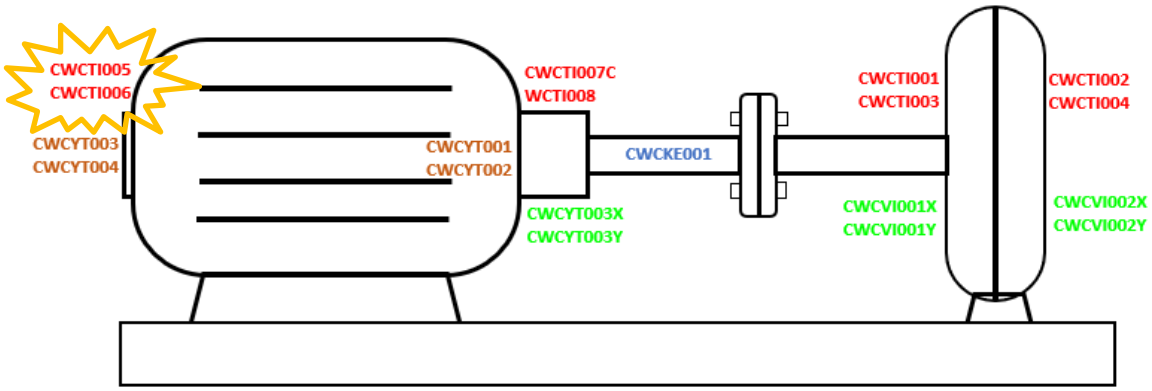
Código del Sensor	Descripción	Límite Máximos Permitido: Alarma - Parada
CWCKE001	Revoluciones por minuto (rpm)	1188 rpm
CWCVI001X	Vibración del lado acoplamiento de bomba	129um (5.1mil) - 150um(5.9mil)
CWCVI001Y	Vibración del lado acoplamiento de bomba	129um (5.1mil) - 150um(5.9mil)
CWCYT003X	Vibración del lado acoplamiento del motor	129um (5.1mil) - 150um(5.9mil)
CWCYT003Y	Vibración del lado acoplamiento del motor	129um (5.1mil) - 150um(5.9mil)
CWCYT002	Vibración lado acoplamiento del motor	4.5 – 7.1 mm/s
CWCYT001	Vibración lado acoplamiento del motor	4.5 – 7.1 mm/s

CWCVI002X	Vibración lado libre de bomba	129um (5.1mil) - 150um(5.9mil)
CWCVI002Y	Vibración lado libre de bomba	129um (5.1mil) - 150um(5.9mil)
CWCYT004	Vibración lado libre del motor	4.5 mm/s – 7.1 mm/s
CWCYT003	Vibración lado libre del motor	4.5 mm/s – 7.1 mm/s
CWCTI001	Temperatura en el lado acoplamiento de bomba	95° - 100°C
CWCTI002	Temperatura en el lado libre de bomba	95° - 100°C
CWCTI003	Temperatura en el lado acoplamiento de bomba	95° - 100°C
CWCTI004	Temperatura en el lado libre de bomba	95° - 100°C
CWCTI005	Temperatura en el lado libre del motor	95° - 100°C
CWCTI006	Temperatura en el lado libre del motor	95° - 100°C
CWCTI007	Temperatura en el lado acoplamiento del motor	95° - 100°C
CWCTI008	Temperatura en el lado acoplamiento del motor	95° - 100°C
CWCTI005	Temperatura en el lado libre del motor	95° - 100°C

Nota. Tomado del Manual del equipo

Figura 66

Distribución de Sensores Electro - Bomba



Se ha venido observado los datos registrados por los 19 Sensores y el registro indica que siempre están por debajo del límite máximo permisible establecido en la Tabla 6, sin embargo, en las últimas evaluaciones realizadas se observó un pequeño y casi imperceptible incremento de temperatura en los Sensores resaltados en la Figura 67 CWCTI005 y CWCTI006

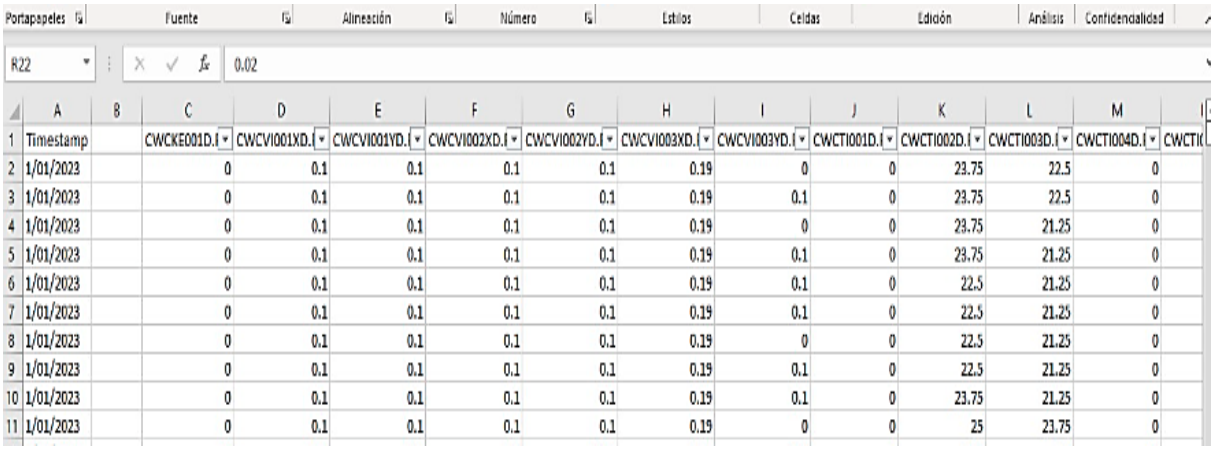
que monitorean la temperatura del rodamiento del lado libre del motor como se puede apreciar en la figura 66.

Ante este pequeño pero creciente desvío, decidimos enfocar el trabajo de investigación en la estimación de la vida útil remanente (RUL) para este rodamiento, de forma que podamos planificar y programar adecuadamente los recursos necesarios: repuestos, mano de obra, equipos, instrumentos de medición, coordinaciones, autorizaciones, etc.

Los datos se colectan en tablas como la figura 67 para un preanálisis desde las bases de datos del sistema. A partir de este punto serán manipulados y adecuados para su análisis.

Figura 67

Datos de Sensores obtenido de las bases de datos del sistema



Portapapeles | Fuente | Alineación | Número | Estilos | Celdas | Edición | Análisis | Confidencialidad

R22 | 0.02

	A	C	D	E	F	G	H	I	J	K	L	M
1	Timestamp	CWCKE001D.(	CWCVI001XD.(	CWCVI001YD.(	CWCVI002XD.(	CWCVI002YD.(	CWCVI003XD.(	CWCVI003YD.(	CWCTI001D.(	CWCTI002D.(	CWCTI003D.(	CWCTI004D.(
2	1/01/2023	0	0.1	0.1	0.1	0.1	0.19	0	0	23.75	22.5	0
3	1/01/2023	0	0.1	0.1	0.1	0.1	0.19	0.1	0	23.75	22.5	0
4	1/01/2023	0	0.1	0.1	0.1	0.1	0.19	0	0	23.75	21.25	0
5	1/01/2023	0	0.1	0.1	0.1	0.1	0.19	0.1	0	23.75	21.25	0
6	1/01/2023	0	0.1	0.1	0.1	0.1	0.19	0.1	0	22.5	21.25	0
7	1/01/2023	0	0.1	0.1	0.1	0.1	0.19	0.1	0	22.5	21.25	0
8	1/01/2023	0	0.1	0.1	0.1	0.1	0.19	0	0	22.5	21.25	0
9	1/01/2023	0	0.1	0.1	0.1	0.1	0.19	0.1	0	22.5	21.25	0
10	1/01/2023	0	0.1	0.1	0.1	0.1	0.19	0.1	0	23.75	21.25	0
11	1/01/2023	0	0.1	0.1	0.1	0.1	0.19	0	0	25	23.75	0

4.3 Revisión de Datos Obtenidos

Los datos disponibles corresponden a los Sensores de temperatura horizontal y vertical CWCTI005 y CWCTI006, para los meses de mayo a junio de 2022 (502 datos) y de enero a marzo de 2023 (805 datos).

Las pruebas fueron realizadas con tres arquitecturas distintas de la red neuronal. Se describe a continuación cada modelo utilizado y los resultados obtenidos.

4.3.1 Primer Modelo utilizado para la estimación del RUL

Se generó el modelo empleando una red neuronal LSTM La primera capa LSTM tiene 50 unidades y se especifica que debe devolver secuencias completas de salida (return_sequences=True). Además, se especifica la forma de entrada de los datos (input_shape) como un tensor de tres dimensiones con una longitud de lookback = 4 y una dimensión de una sola variable.

La segunda capa LSTM tiene 100 unidades y también devuelve secuencias completas de salida. La forma de entrada es la misma que la capa anterior.

La tercera capa LSTM tiene 200 unidades y se especifica que debe utilizar la función de activación tangente hiperbólica (activation='tanh') y devolver secuencias completas de salida.

La cuarta capa LSTM tiene 100 unidades, utiliza la función de activación tangente hiperbólica y devuelve secuencias completas de salida.

La quinta capa LSTM tiene 50 unidades, utiliza la función de activación tangente hiperbólica y devuelve secuencias completas de salida.

Finalmente, se agrega una capa densa con una sola unidad de salida.

Definimos la arquitectura de la red LSTM en Python como:

```
model= Sequential
modelo.add(LSTM(units=50, return_sequences=True, input_shape=(lookback, 1)))
modelo.add(LSTM(units=100, return_sequences=True, input_shape=(lookback, 1)))
modelo.add(LSTM(units=200, activation= 'tanh', return_sequences=True))
modelo.add(LSTM(units = 100, activation= 'tanh', return_sequences=True))
modelo.add(LSTM(units = 50, activation= 'tanh', return_sequences=True))
modelo.add(LSTM(units = 25, activation= 'tanh'))
model.add(dense(1,))
```

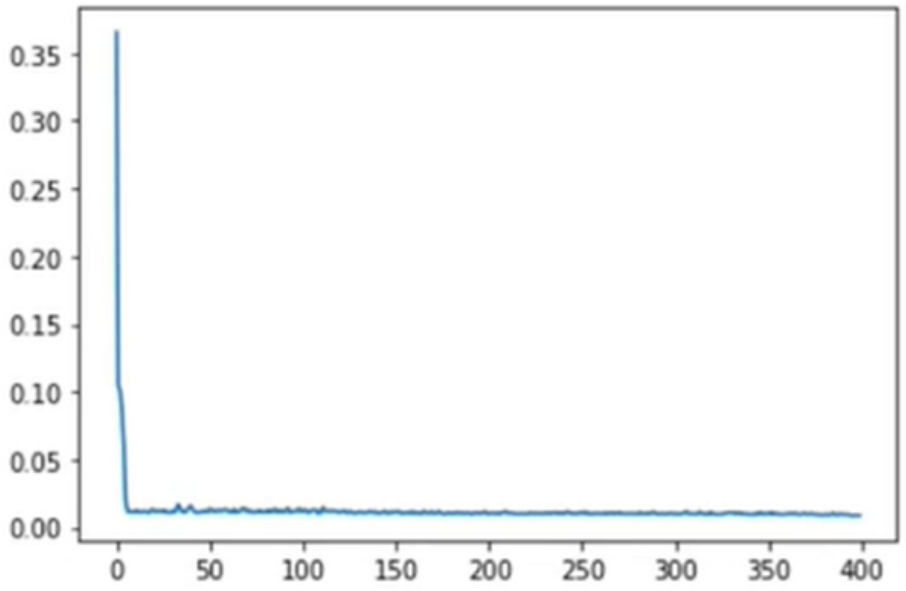
En resumen, este modelo de red neuronal LSTM utiliza varias capas para procesar datos de series de tiempo y predecir una variable de salida.

Para el entrenamiento del modelo se utiliza 400 épocas y un batch de 32 y la totalidad de los datos.

La curva de pérdida se muestra en la figura 68 y se observa un rápido descenso.

Figura 68

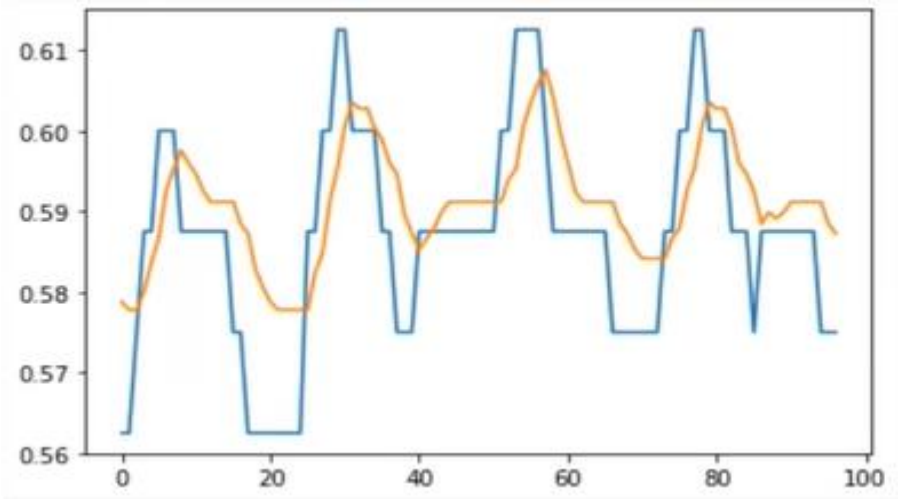
Curva de Perdida (Loss)



En la figura 69 se muestra el resultado del modelo de la red para los primeros 502 datos.

Figura 69

Resultado del primer modelo Temperatura real (azul), Temperatura predicha (rojo)



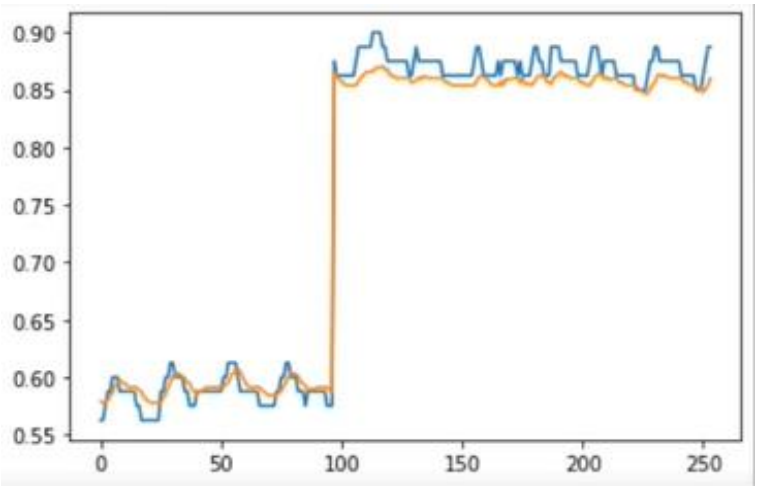
Observamos que hay una ligera diferencia entre las predicciones y los datos de reales. Esto se debe a que la cantidad de datos para entrenar el modelo no es suficiente.

Debemos señalar que en esta red no se toma en cuenta el horómetro, el modelamiento considera sólo la cantidad de datos disponibles.

Si modelamos la red con los datos de los meses mayo a junio (502) y la de enero marzo (805), los resultados de la gráfica sería la siguiente:

Figura 70

Resultado del modelado Temperatura real (azul), Temperatura predicha (rojo) para todo el conjunto de datos de mayo a junio de 2022 y enero a marzo de 2023



La gráfica de la predicción de la figura 70, con los datos de prueba, sufre un salto entre un grupo de datos y otro, lo cual no es un resultado adecuado y esto se debe a la falta de datos del mes de abril.

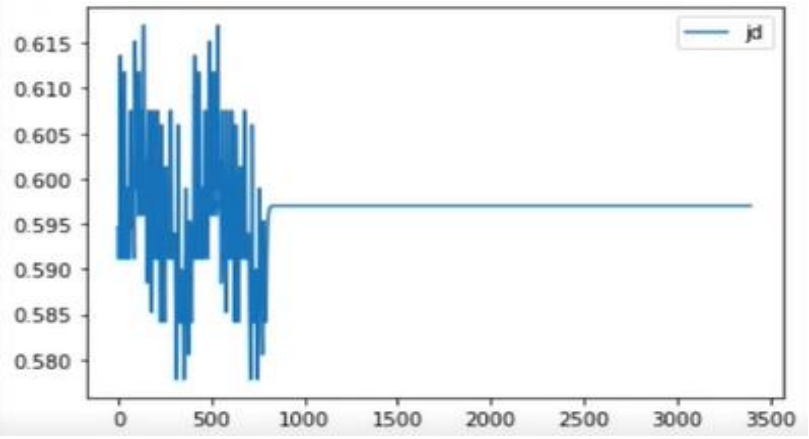
Limitación en el modelo de la LSTM. Esto podría ser el resultado de la elección de los parámetros del modelo, la complejidad de la arquitectura de la red neuronal, o la falta de datos de entrada relevantes. En estos casos es necesario ajustar el modelo o agregar más datos para mejorar la precisión de las predicciones.

Presencia de ruido en los datos. Esto podría ser el resultado de errores en la medición, datos faltantes o una mala calidad de los datos. En este caso se debe prestar atención a la calidad de los datos de entrada y considerar la posibilidad de utilizar técnicas de preprocesamiento para eliminar el ruido.

Analicemos los resultados de la predicción de la vida útil del activo (RUL).

Figura 71

Predicción del RUL



Los resultados no reflejan una buena predicción, no se puede obtener el RUL. Se observa que hay una perdida en la tendencia. El espacio se vuelve una línea recta en el futuro estabilizando su valor en 0.595 (59.5°C), lo cual no es correcto dado que el rodamiento por su degradación deberá incrementar la temperatura en el futuro.

Si el modelo de la LSTM no fue entrenado con datos relevantes o con suficientes datos que cubran toda la vida útil del activo, la predicción se ve limitada su comportamiento después de cierta cantidad de iteraciones se vuelve constante.

Estabilidad en el comportamiento del equipo: si la línea recta en la gráfica del RUL representa una estimación precisa del tiempo restante de la vida útil del activo, esto podría ser una indicación de que el comportamiento del activo es estable y predecible después de cierto punto. En este caso la LSTM podría estar proporcionando una predicción precisa de la vida útil restante del activo y la línea recta podría indicar que no se espera que el equipo falle o se deteriore significativamente durante el tiempo restante. Para este caso, dado que la degradación de los componentes o subsistemas ocurre de manera natural a lo largo de su vida útil, nos inclinaríamos a decir que para este resultado obtenido faltarían datos que cubran hasta predecir la vida útil del activo.

Limitaciones en el modelo: otra posible causa de que la gráfica del RUL de la LSTM termine en una línea recta podría ser una limitación en el modelo de la red neuronal. Esto podría deberse a la elección de los parámetros del modelo, la complejidad de la arquitectura de la red neuronal y la falta de datos de entrenamiento relevantes. En estos casos podría ser necesario ajustar el modelo o agregar más datos de entrenamiento para mejorar la precisión de las predicciones.

4.3.2 Segundo Modelo utilizado para la estimación del RUL

Para este segundo modelo, consideramos el horómetro del activo como variable tiempo. Esta consideración toma como hipótesis que los componentes del activo sufren envejecimiento (pérdida de la vida útil) con el transcurrir del tiempo.

Tenemos dos opciones: utilizar una regresión lineal o una Artificial Neural Network (ANN)

Nos decidimos por utilizar una ANN para modelar el comportamiento de la temperatura.

Empezamos a procesar los datos, pero desde el punto de vista de una recta. Nuestra hipótesis es que la temperatura es una función del tiempo, es decir que la única variable que influye en la variación de esta temperatura es el tiempo, cuyos valores son registrados con el horómetro del activo.

A continuación, describimos el código Python que define y entrena nuestra red ANN.

```
# definir arquitectura de la red de regresión lineal

Model = Sequential ()

Model.add(Dense(1,))
```

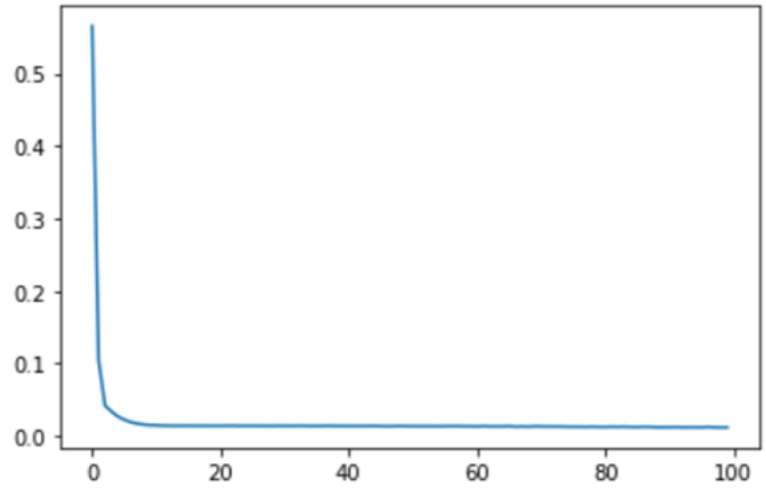
Esta arquitectura permitirá crear un modelo de red neuronal muy simple. El modelo consta de una solo capa, totalmente conectada (Dense) con una solo neurona. La función de activación de la neurona no está especificada, por lo que se utiliza la función de activación lineal por defecto.

En general esta red tiene la particularidad de ser utilizada para problemas de regresión en los que se desea predecir un valor numérico continuo a partir de un conjunto de datos de entrada. Sin embargo, esta red es muy simple, y la hemos elegido para analizar sus resultados del modelo y ver si se ajusta a nuestra necesidad para el objetivo planteado, que es predecir la vida útil del activo (RUL), a partir del análisis de las temperaturas en el lado libre del motor eléctrico.

Si graficamos la función de pérdida, el comportamiento en su disminución es más suave (ver figura 72). Este resultado ya nos indicaría que el rendimiento del modelo nos permitirá obtener mejores resultados de la predicción.

Figura 72

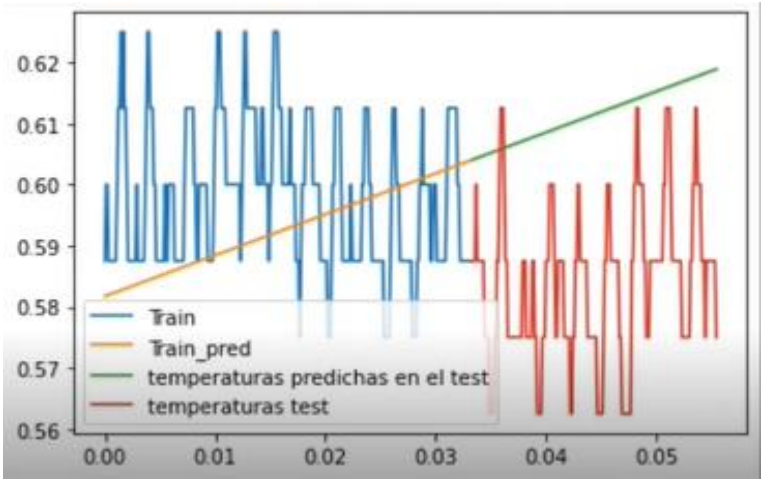
Función de pérdida



Los resultados del modelo muestran la dinámica en el rango mayo a junio del 2022, según la siguiente gráfica.

Figura 73

Entrenamiento, Pruebas y Predicción



Los resultados indican que tenemos mayor desviación estándar en ciertos puntos temporales, pero en promedio, este modelo está calculando los resultados con error en promedio de ± 2 °C.

Como lo mencionamos anteriormente, se utilizó una sola neurona sin función de activación. Es sólo una regresión lineal, es decir utiliza la función de activación lineal por defecto, suficiente para modelar la temperatura en función del tiempo (horómetro del activo).

Analizando los resultados de las métricas de validación obtenidas:

R2 train = 0.9612, es el coeficiente de determinación R2 obtenido al evaluar el modelo en los datos de entrenamiento. Indica que la calidad del ajuste del modelo en los datos utilizados para entrenar la ANN, es aceptable.

R2 test = 0.9548. Es el coeficiente de determinación R2 obtenido al evaluar el modelo en los datos de prueba que no se utilizaron durante el entrenamiento. Indica que la capacidad de generalización del modelo a nuevos datos, son favorables.

Estos valores obtenidos nos indicarían una buena capacidad de nuestro modelo, y tendría la capacidad suficiente para obtener una predicción de una forma eficiente.

4.3.3 Tercer Modelo utilizado para la estimación del RUL

Para este tercer modelo, al igual que el modelo N° 2, utilizamos una red neuronal ANN (Artificial Neural Network), pero con otra arquitectura, descrita a continuación:

```
Mode12 = Sequential()

Mode12.add(Dense(100,activation='relu'))

Mode12.add(Dense(50,activation='relu'))

Mode12.add(Dense(20,activation='relu'))

Mode12.add(Dense(1))
```

El código presentado define una red neuronal secuencial (Sequential) compuesta por cuatro capas densamente conectadas (Dense). La arquitectura de la red es la siguiente:

La primera capa tiene 100 nodos y utiliza la función de activación ReLU (Rectified Linear Unit).

La segunda capa tiene 50 nodos y también utiliza la función de activación ReLU.

La tercera capa tiene 20 nodos y también utiliza la función de activación ReLU.

La última capa es la capa de salida, que tiene un solo nodo y no tiene función de activación.

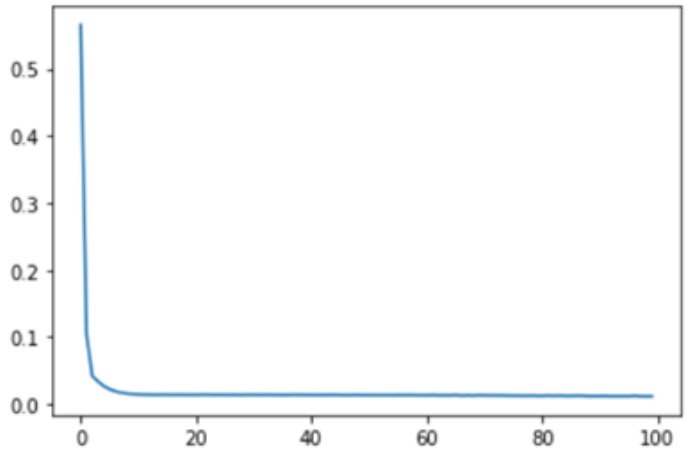
La capa de salida es la que genera la predicción de la red y no utiliza una función de activación ya que se trata de un problema de regresión y se espera una salida numérica continua.

En resumen, la red neuronal definida por este código es una red secuencial de cuatro capas densamente conectadas que utiliza la función de activación ReLU en las tres primeras capas y no utiliza función de activación en la capa de salida. Esta red se utiliza para resolver problemas de regresión.

La función de pérdida, para este tercer modelo, también desciende en forma suave (ver figura 74). Este resultado nos indica que el rendimiento del modelo nos permitirá obtener mejores resultados en la predicción.

Figura 74

Función de pérdida

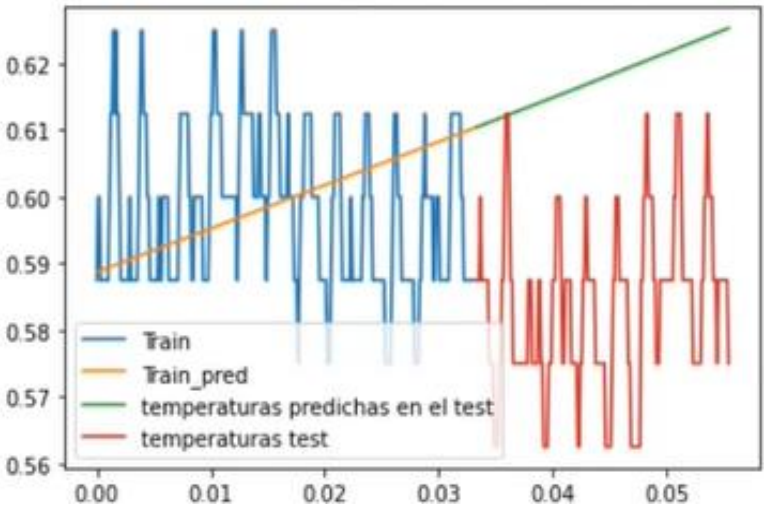


Se han definido como temperatura máxima admisible 100°C, y en valores normalizados, sería equivalente a 1.00. También hemos definido como horómetro máximo admisible de 9000 horas.

Esta gráfica corresponde al periodo mayo – junio 2022:

Figura 75

Entrenamiento, pruebas y predicción



La pendiente tiene una ligera pendiente con respecto al modelo anterior

De igual forma, este modelo también nos permitirá predecir la variable temperatura.

Analizando los resultados de las métricas de validación obtenidas:

R2 train = 0.9856, es el coeficiente de determinación R2 obtenido al evaluar el modelo en los datos de entrenamiento. Indica que la calidad del ajuste del modelo en los datos utilizados para entrenar la ANN, es aceptable.

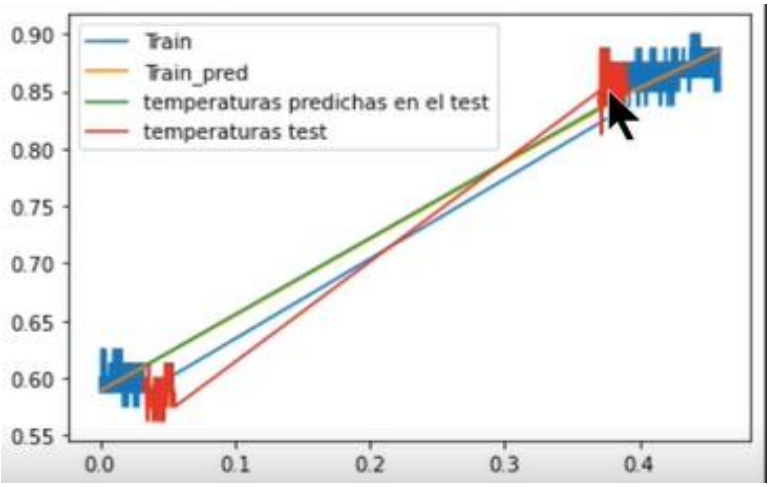
R2 test = 0.9212. Es el coeficiente de determinación R2 obtenido al evaluar el modelo en los datos de prueba que no se utilizaron durante el entrenamiento. Indica que la capacidad de generalización del modelo a nuevos datos, son favorables.

Estos valores obtenidos nos indican una buena capacidad de predicción de nuestro tercer modelo, pero será suficiente realizar el cálculo con el segundo modelo.

La figura 76 muestra los valores de temperatura vs tiempo a partir de enero 2023 hacia el futuro.

Figura 76

Entrenamiento, pruebas y predicción



El modelo predice el comportamiento de la variable temperatura de manera adecuada, y cumple con la predicción

Con estos datos o resultados del tercer modelo, se calculó el RUL = 4481 horas; es decir que la temperatura alcanzará los 100°C cuando llega a las 4481 horas de operación. Esta predicción tiene un error del 2%.

Horas	Temperatura
4481	100.005402
4482	100.012413
4483	100.019417
4484	100.026428
4485	100.033440
...	...
8995	131.610352
8996	131.618210

8997 131.626053

8998 131.633881

8999 131.641693

[4519 rows x 1 columns]

4.3.4 Modelo alternativo para generar datos sintéticos durante la estimación del RUL

En la serie de datos temporales del proceso en estudio en el trabajo de investigación, no se cuenta con información entre los meses de julio hasta diciembre de 2022. Por lo tanto, se propone un método alternativo para generar data sintética que complementará la data faltante para poder realizar el estudio y cálculo del RUL, mediante machine learning con redes neuronales.

El método alternativo propuesta es el uso de modelos estadísticos ARIMA, que nos permitirán generar la data sintética para complementar el periodo que no se cuenta con datos de la variable en estudio, Temperatura del rodamiento de motor eléctrico de la bomba de proceso.

4.3.4.1 Modelo ARIMA (Autoregresivo de Media Móvil Integrado). Los modelos ARIMA son sofisticados porque permiten crear modelos que contengan el componente de tendencia y estacional. El modelo ARIMA, solo modela series temporales de procesos estacionarias, por el cual se debe evaluar la estacionariedad de la serie temporal.

La principal característica del modelo ARIMA, es el enfoque estocástico que le da a las series de tiempo.

En sus inicios este modelo de series de tiempo estaba muy limitados, por razones de cómputo, actualmente con el desarrollo de la tecnología computacional ha hecho posible el manejo de datos para la modelización ARIMA.

4.3.4.2 Enfoque y procesos estocásticos. Un proceso estocástico podemos definirlo como una familia de variables aleatorias que se asocian a un conjunto índice de números reales, del cual a cada elemento del conjunto tiene correspondencia con una y solo una variable aleatoria que evoluciona con el tiempo.

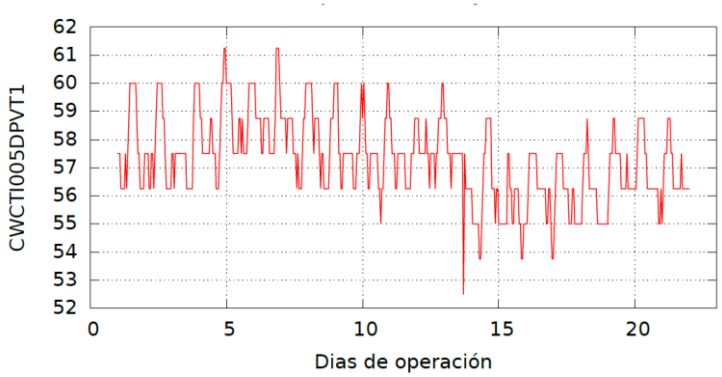
4.3.4.3 Estacionariedad. Es un proceso estocástico cuya serie temporal presenta media y varianzas constantes en todo el historial.

4.3.4.4 Modelado para generación de datos sintéticos

4.3.4.4.1 Serie temporal de la variable en estudio. Los datos tomados del sensor de temperatura del cojinete vienen definidos con el nombre CWCTI005DPVT1, cuya grafica de series temporales es como sigue.

Figura 77

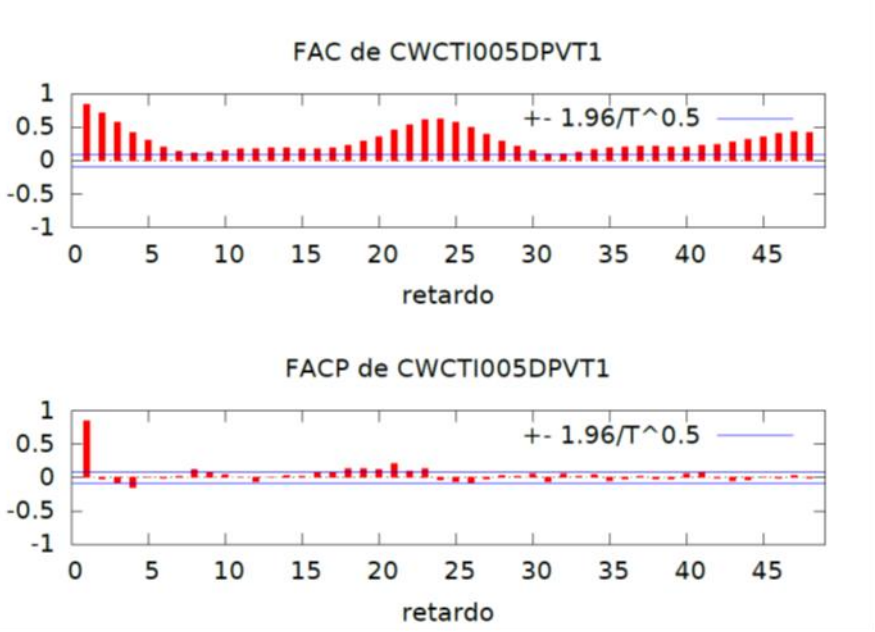
Temperatura - tiempo



4.3.4.4.2 Evaluación de la Estacionariedad de la Variable en estudio. Para evaluar la estacionariedad se evalúan los correlogramas FAC (función de autocorrelación) y FACP (función de autocorrelación parcial).

Figura 78

FAC y FACP



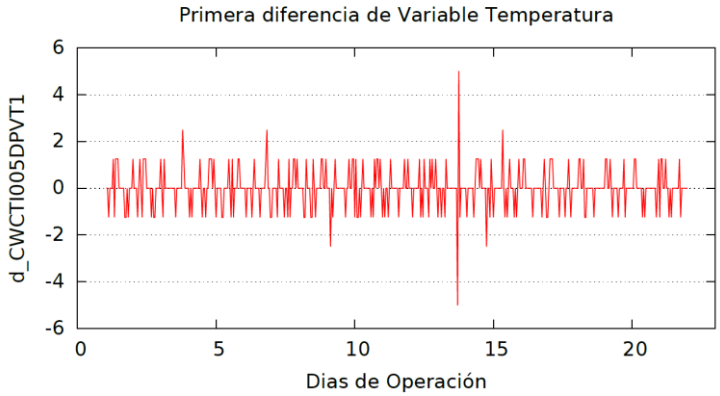
Del correlograma FAC se observa que los datos se encuentran fuera del rango, por el cual se concluye que la serie temporal es no estacionaria.

Para transformar la serie temporal estacionaria a una variable no estacionaria se tomarán las primeras diferencias de las variables.

4.3.4.4.3 Primeras Diferencias de la variable en estudio. De la variable principal Temperatura se toman las primeras diferencias, de esta manera transformamos la variable temperatura en una nueva variable de primeras diferencias de temperaturas, cuya grafica de series temporales es como sigue.

Figura 79

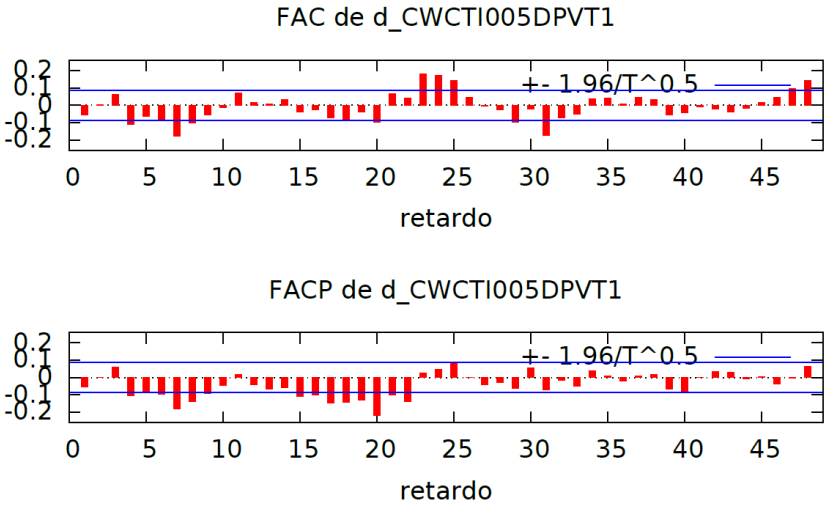
Serie temporal



4.3.4.4.4 Evaluación de la Estacionariedad de primeras diferencias de la variable en estudio. Para evaluar la estacionariedad se evalúan los correlogramas FAC y FACP.

Figura 80

Evaluación de la Estacionariedad



Del correlograma FAC se observa que la mayor parte de los datos se encuentra dentro del rango, por el cual se concluye que la serie temporal de las primeras diferencias de temperatura es estacionaria.

Conociendo que la variable transformada, primeras diferencias de la temperatura, es estacionaria se procede a modelamiento ARIMA.

4.3.4.4.5 Estacionalidad. La estacionalidad de una serie de datos se observa en la ciclicidad que se presenta en forma regular.

Conforme a la gráfica de los datos de la variable de temperatura se concluye que el proceso tiene un comportamiento estacional., el cual debe ser considerado en el modelamiento ARIMA, con ARIMA Estacional (SARIMA).

Modelamiento ARIMA. De los correlogramas se determina que los índices (p,d,q) para el modelo ARIMA.

p: corresponde al modelo autorregresivo.

d: corresponde al modelo integral de las diferencias de las variables.

q: corresponde a la modelo media móvil.

De la gráfica de los correlogramas de la primera diferencia de la variable de temperatura se concluye:

p=7, d=1, q=6 modelo ARIMA (7,1,6) para no estacional.

Para el modelo ARIMA Estacional (SARIMA) se inicia con los datos siguientes,

P=1, D=1, Q=1 modelo SARIMA (1,1,1) para estacional.

Haciendo uso de la herramienta GRETL se calcula el modelo.

ARIMA (7,1,6) x SARIMA (1,1,1).

Figura 81

Model ARIMA en Gretl

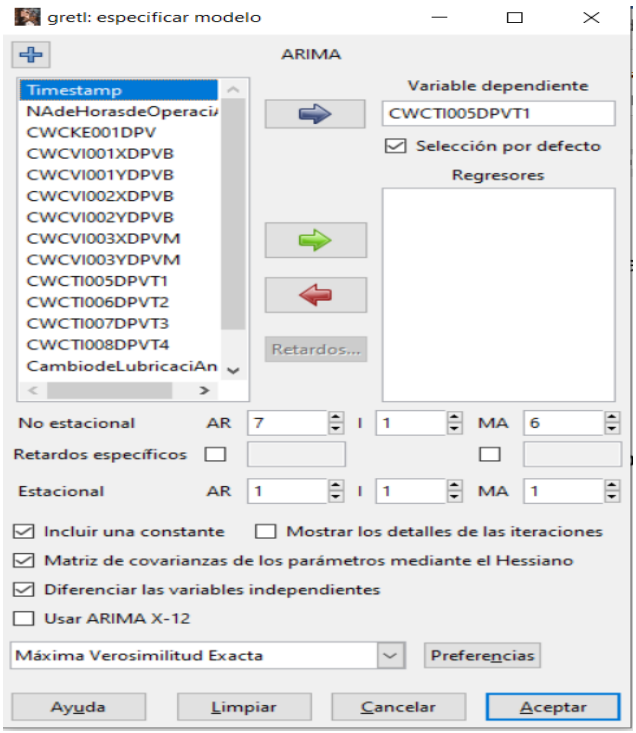


Figura 82

Predicción de la temperatura

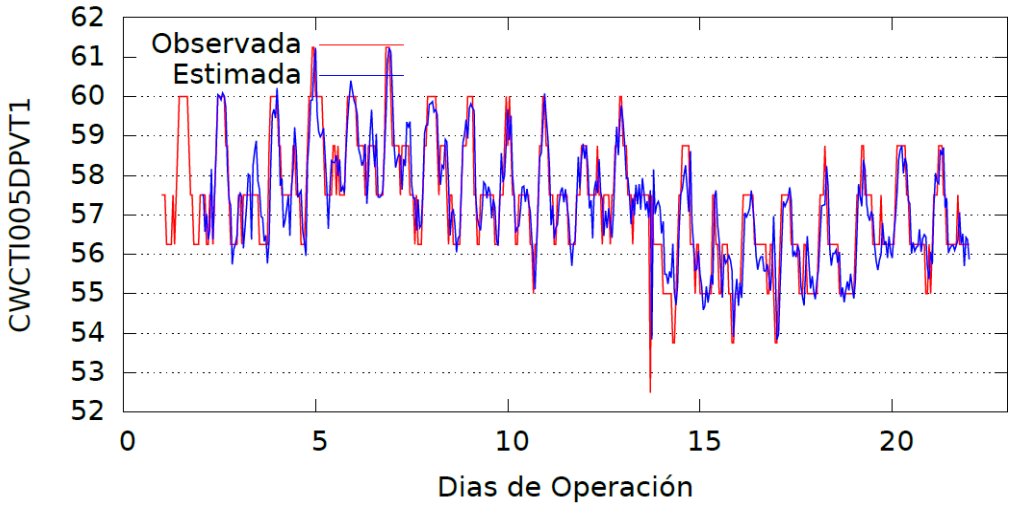


Figura 83

Residuos de la regresión

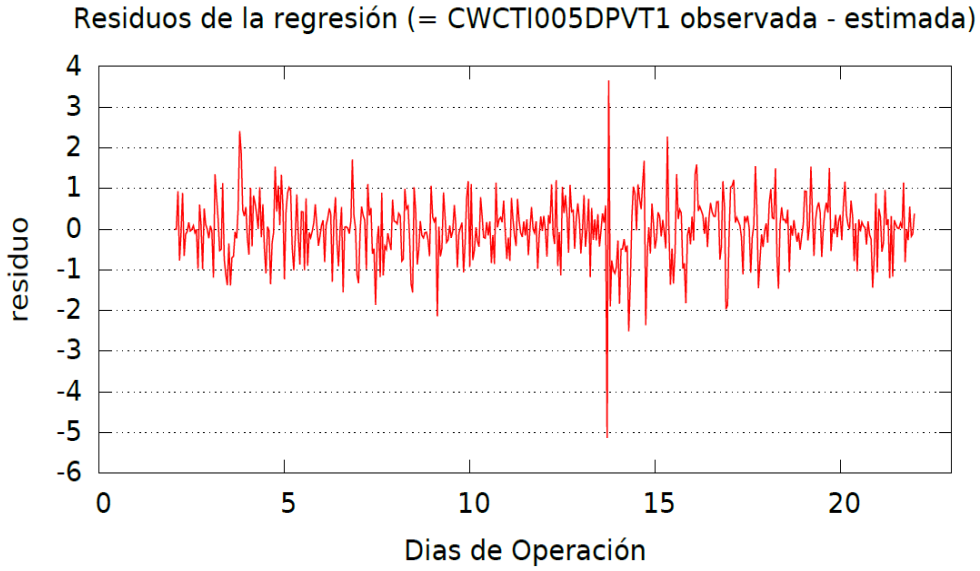


Figura 84

Predicción de la temperatura

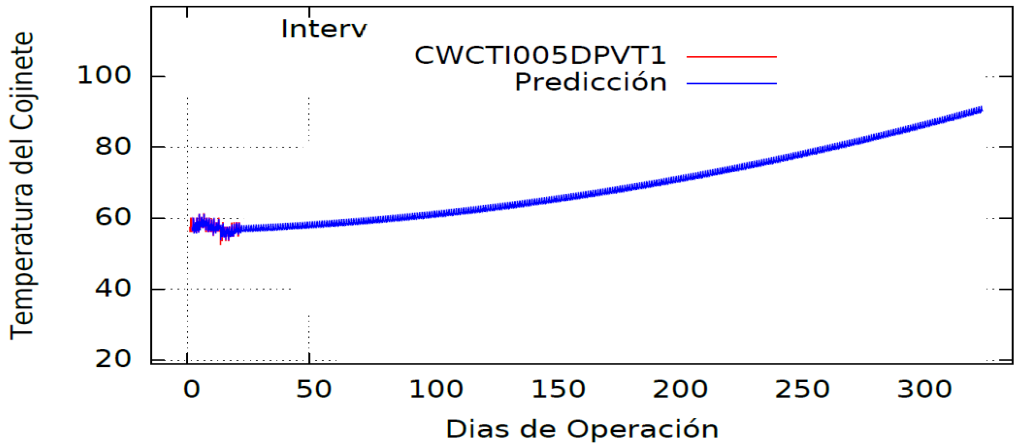


Figura 85

Predicción de la temperatura

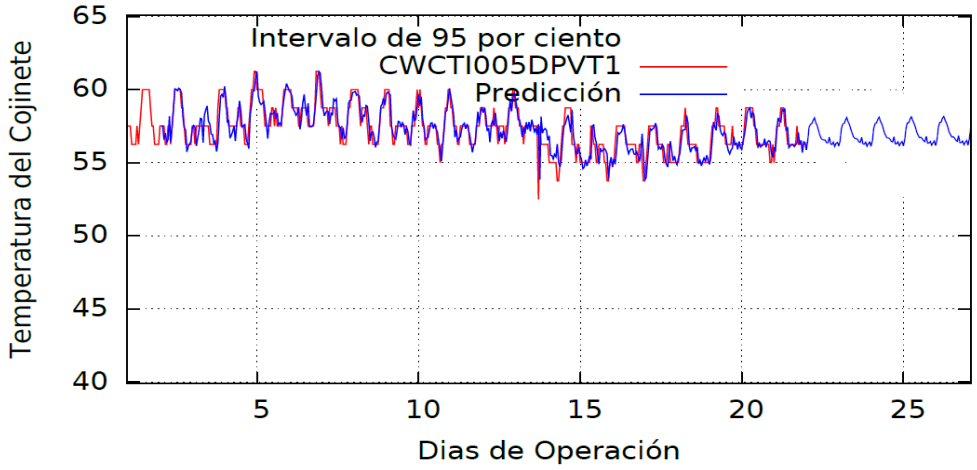
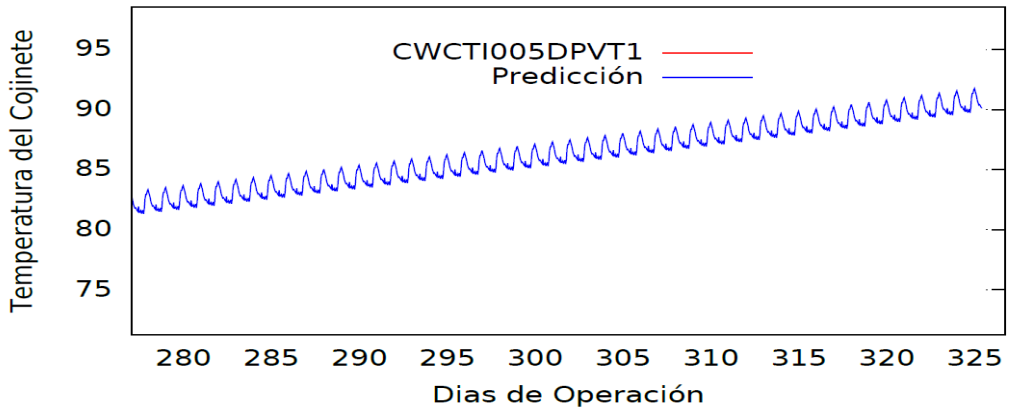


Figura 86

Predicción de la temperatura



UNIVERSITAS STUDIORUM PIURENSIS

Conclusiones

Respecto al primer modelo entrenado:

- El modelo utilizado presenta una ineficiencia en la capacidad para predecir valores futuros dado que, si bien las predicciones de corto plazo son aceptables, cuando las predicciones se alejan del conjunto de datos la predicción ya no es aceptable. Esto se debe al sobre ajuste que presenta el modelo cuyo R2 = 0.989.

Respecto al segundo modelo entrenado:

- Los resultados obtenidos, nos demuestran que el modelo utilizando la ANN se logra un buen ajuste con un R2train = 0.9612. Dicho modelo predice los datos de prueba con un ajuste R2test=0.9548, por lo tanto, es un modelo adecuado para predecir nuevos datos con mejor precisión respecto al primer modelo.

Respecto al tercer modelo entrenado:

- Los resultados obtenidos, nos demuestran que el modelo utilizando la ANN logra un buen ajuste con un R2train = 0.9856. Dicho modelo predice los datos de prueba con un ajuste R2test=0.9212, por lo tanto, este modelo nos indica una buena capacidad para predecir nuevos datos, sin embargo, como podemos apreciar, al intentar encontrar un mejor modelo para obtener un mejor ajuste, la predicción en los datos de prueba es menos eficiente respecto al segundo modelo.

- Los estudios y pruebas realizados nos indican que el tempo de vida útil restantes es del orden de 1,573 horas de operación, contadas a partir del 08/02/2023. Esto nos alerta para realizar la planificación de la intervención programada y adquisición de todos los recursos necesarios a fin de que la indisponibilidad del equipo sea corta.

- El cálculo del RUL permite al ingeniero de mantenimiento tomar las acciones inmediatas y de mediano plazo, dentro de las cuales deberá analizar si las condiciones externas actuales de la bomba, la han llevado a un envejecimiento prematuro del activo.

- Asimismo, deberá gestionar la estrategia de mantenimiento más adecuada que corrija las anomalías que actualmente se han presentado y han llevado al activo a una tendencia hacia valores de alarma en el rodamiento del motor eléctrico del lado libre.

- Respecto al uso del modelo estadístico ARIMA evaluado mediante el software Gretl, podemos concluir que aunque no fue concebido para dicho propósito, se puede aplicar también para modelar el comportamiento de variables de procesos industriales para la predicción en el tiempo.

- Se mostró el uso de este modelo estadístico para complementar datos sintéticos cuando no se tiene datos reales en ciertos espacios de tiempo.

- Queda a criterio para futuros trabajos, el uso de esta metodología estadística.

- Al igual que la LSTM, ARIMA presenta el problema de predicción a futuro cuando la cantidad de datos disponibles es mucho menor que los datos a predecir en el tiempo.

- La IA puede ser utilizada para analizar grandes cantidades de datos históricos de mantenimiento y predecir cuándo es probable que una máquina o equipo falle. Esto permite que las empresas programen el mantenimiento preventivo antes de que ocurra un problema, lo que puede reducir el tiempo de inactividad y los costos de mantenimiento. Esto permite a los técnicos de mantenimiento centrarse en las áreas problemáticas y resuelvan antes de que se conviertan en problemas mayores.

Referencias bibliográficas

Bisong, E. (2019). *Bisong, E. "Building Machine Learning and Deep Learnig Models on Google Cloud Platform".* OTTAWA, ON., Canada: Apress Berkeley, CA. Obtenido de https://doi.org/10.1007/978-1-4842-4470-8_19

Bisong, E. (2019). OTTAWA, ON., Canada: Apress, Berkeley. doi:10.1007/978-1-4842-4470-8_19

Cachada, A. (2018). *"Maintenance 4.0: Arquitectura de sistema de mantenimiento inteligente y predictivo".* Turin, Italia. doi:10.1109/ ETFA.2018.8502489

Dalzochio, J. K. (28 de Julio de 2020). Obtenido de https://www.sciencedirect.com/science/article/abs/pii/S0166361520305327

Dueñas Ramírez, L. M., Villegas López, G. A., Castiblanco Tique, S., & Castaño Restrepo, C. A. (2021). Casos de éxito en la implementación del mantenimiento predictivo mediante el uso de tecnologías de la industria 4.0 en empresas colombianas.

FAA, A. T. E. O. Malfunctions Basic Familiarization for Flight Crews, 2006, Chapter 1.

Jan Zenisek, J. W. (11 de 2018). *https://doi.org/10.1016/j.ifacol.2018.08.391.*

Kaggle. (s.f.). *https://www.kaggle.com/datasets/nicolascaparroz/turbofan-hpc-efficiency.*

Khan, M., Tse, P., & Trappey, A. ". (16 de Diciembre de 2021). *https://doi.org/10.3390/s21248420.* (Sensors, Ed.) doi:10.3390/s21248420

M. Yuan, Y. W. (10 de Octubre de 2016). *https://ieeexplore.ieee.org/abstract/document/7748035.* doi:10.1109/AUS.2016.7748035.

Medjaher, K., Zerhouni, N., & Gouriveau, R. (2016). *From prognostics and health systems management to predictive maintenance 1: Monitoring and prognostics*. John Wiley & Sons.

Moreno Muñoz, A. (2022). *Implementación del mantenimiento TPM, Técnicas Lean y 4.0 en una fábrica de automoción* (Master's thesis, Universitat Politècnica de Catalunya).

P., Z., Jiang, W., Shi, X., & Zhang, S. (Octubre de 2022). *https://doi.org/10.3390/pr10122500.*

Qiang Miao, L. X. (Junio de 2013). Obtenido de https://doi.org/10.1016/j.microrel.2012.12.004

Rafael Gouriveau; Kamal Medjaher; Noureddine Zerhouni. “From Prognostics and Health Systems Management to Predictive Maintenance 1: Monitoringand Prognostics”. (2016)

Reference., P. S. (2020). Obtenido de https://docs.python.org/3/reference/index.html

Saxena, A., Goebel, K., Simon, D. y Eklund, N. (2008, octubre). Modelado de propagación de daños para simulación de funcionamiento hasta el fallo de motores de aeronaves. En *2008 congreso internacional sobre pronóstico y gestión de la salud* (págs. 1-9). IEEE.

Shahnawaz Ahmad, A Suman, T Sidharth, Ganesh Pawar, Vikas Kumar, N.S. Vyas. (2019). *Structural Integrity Analysis and Life Estimation of a Gas Turbine Bladed-Disc*. Procedia Structural Integrity. Obtenido de https://doi.org/10.1016/j.prostr.2019.08.101.

Van Rossum, G. (2009). *The Python programming language*. Obtenido de Software: Practice and Experience: https://doi.org/10.1002/spe.995

Apéndices

UNIVERSITAS STUDIORUM PIURENSIS

Apéndice A Conjunto de datos de consumo eléctrico de Nigeria para aplicar un modelo de series temporales univariantes con redes neuronales recurrentes LSTM

Para aumentar la velocidad de entrenamiento, el modelo se entrenará en una GPU. Si ejecuta el código en Google Colab, cambie el tipo de tiempo de ejecución a GPU e instale el paquete TensorFlow 2.0 con GPU.

```
# importar TensorFlow 2.0 con GPU
pip install -q tf-nightly-gpu-2.0-preview
# import packages
import tensorflow as tf
import pandas as pd
import numpy as np
importar matplotlib.pyplot como plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
# confirma que tensorflow puede ver la
GPU device_name =
tf.test.gpu_device_name() if
device_name != '/device:GPU:0':
raise SystemError('Dispositivo GPU no
encontrado') print('GPU encontrada en:
{}'.format(nombre_dispositivo))
# ruta del archivo de datos
file_path = "nigeria-consumo-de-potencia.csv"
# cargar datos
parse_date = lambda dates: pd.datetime.strptime(dates, '%d-%m')
data = pd.read_csv(file_path, parse_dates=['Mes'], index_col='Mes',
date_parser=parse_date,
engine='python', skipfooter=2)
# imprimir nombre
de columna
data.columns
# cambiar los nombres de las columnas
data.rename(columns={'consumo energético Nigeria':'consumo
energético'}, inplace=True)
# split in training and evaluation set
datos_entrenamiento, datos_evaluación =
entrenar_prueba_split(datos, tamaño_prueba=0,2,
aleatorio=False)
# MinMaxScaler - centrar y escalar el
conjunto de datos scaler =
MinMaxScaler(feature_range=(0, 1)) data_train
= scaler.fit_transform(data_train) data_eval
= scaler.fit_transform(data_eval)
# ajustar los datos univariantes para la predicción de series temporales
def convert_to_sequences(datos, secuencia,
is_target=False): temp_df = []
for i in range(len(data) - sequence):
if is_target:
temp_df.append(datos[(i+1): (i+1) +
```

```
secuencia]) else:
temp_df.append(datos[i: i +
secuencia]) return np.array(temp_df)
# parámetros
time_steps = 20
tamaño_lote = 50
# crear datos de entrenamiento y de prueba
train_x = convert_to_sequences(data_train, time_steps, is_target=False)
train_y = convert_to_sequences(data_train, time_steps, is_target=True)
eval_x = convert_to_sequences(data_eval, time_steps, is_target=False)
eval_y = convert_to_sequences(data_eval, time_steps, is_target=True)
# construir modelo
modelo = tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(128, input_shape=train_x.shape[1:],
return_sequences=True))
model.add(tf.keras.layers.Dense(1))
# compila el modelo
model.compile(loss='mean_squared_error',
optimizador='adam
',
métrica=['mse'])
# imprimir resumen
del modelo
model.summary()
# create dataset pipeline
train_ds = tf. data.Dataset.from_tensor_slices(
(train_x, train_y)).shuffle(len(train_x)).repeat().batch(batch_size)
test_ds = tf.data.Dataset.from_tensor_slices((eval_x, eval_y)).batch(batch_
size)
# entrenar el modelo
history = model.fit(train_ds, epochs=10,
steps_per_epoch=500)
# evaluar el modelo
pérdida, mse = modelo.evaluar(prueba_ds)
print('Pérdida de prueba: {:.4f}'.format(loss))
print('Prueba mse: {:.4f}'.format(mse))
# predict
y_pred = model.predict(eval_x)
# traza la secuencia predicha
plt.title("Prueba del modelo",
fontsize=12)
plt.plot(eval_x[0,:,0], "b--", markersize=10, label="instancia de
entrenamiento") plt.plot(eval_y[0,:,0], "g--", markersize=10,
label="objetivos") plt.plot(y_pred[0,:,0], "r--", markersize=10,
label="predicción del modelo") plt.legend(loc="superior izquierda")
plt.xlabel("Tiempo
") plt.show()
```

Apéndice B Código Python para estimación del RUL del rodamiento del motor eléctrico de la bomba de proceso mediante una ANN

Este modelo es para completar algunos datos faltantes o datos de parada por casas externas al activo. Vamos a generar data sintética como si esas paradas no se hubieran realizado. Esto con el fin de tener más datos para el entrenamiento final.

```
#montamos colab
from google.colab import drive
drive.mount('/content/drive')
import numpy as np
import random
# Fijar la semilla a 42
random.seed(42)
#importamos la ruta y obtenemos el df
import os
import pandas as pd
os.chdir("/content/drive/MyDrive/mto")
df = pd.read_excel('df1.xlsx')
df
#Renombrar cada columna
columns = ['Hora','TC5', 'TC6']
df.columns = columns
df
#graficar el comportamiento temporal
import matplotlib.pyplot as plt
df['TC6'].plot()
#df['TC5'].plot()
Modelo LSTM para predecir los datos faltantes.
#librerías
import numpy as np
from keras.models import Sequential
from keras.layers import LSTM, Dense
from sklearn.preprocessing import MinMaxScaler
#colocar la fecha como índice.
#df['Fechas'] = pd.to_datetime(df['Fechas'])
#df.set_index('Hora', inplace=True)
# Imputar valores faltantes con interpolación lineal
#df = df.interpolate()
#df
# Normalizar los datos
#Para esto usaré la función anterior de normalización.
def fscale(a,bottom = 0,top = 100):
  #a es un np.array de 1D
  #devuelve un array normalizado de 0 de 1 con los valores en ese rango.
  scale = []
  for i in a:
    j = (i-bottom)/(top-bottom)
    scale.append(j)
  return np.array(scale)
TC6_scale = fscale(np.array(df['TC6']))
```

```
#print(TC6_scale)
# pasarlo al df
df['TC6'] = TC6_scale
df
df2=pd.read_excel('df2.xlsx')
print(df2)

# Crear función para preparar datos de entrenamiento
def prepare_data(data, lookback):
    X, y = [], []
    for i in range(lookback, int(len(data)*0.80)):
        X.append(data.iloc[i-lookback:i])
        y.append(data.iloc[i])
    X, y = np.array(X), np.array(y)
    X = np.reshape(X, (X.shape[0], X.shape[1], 1))
    y = np.reshape(y,(-1,1))
    return X, y
lookback = 4 # Horizonte de tiempo X = [Xt-3,Xt-2,Xt-1,Xt] => y = [Xt+1]
#Entrenamiento del primer sector de mayo a junio
X_train_1,y_train_1 = prepare_data(df_6,lookback)
#Entrenamiento del segundo sector de enero a Marzo
X_train_2,y_train_2 = prepare_data(df5,lookback)
#Unir ambos conjuntos de entrenamiento
X_train = np.vstack((X_train_1,X_train_2))
y_train = np.vstack((y_train_1,y_train_2))
print(y_train)
# Definir tamaño de ventana temporal
#lookback = 2
# Preparar datos de entrenamiento
#X_train, y_train = prepare_data(df_6, lookback)
# Definir arquitectura de la red LSTM
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(lookback, 1)))
model.add(LSTM(units=100, return_sequences=True, input_shape=(lookback, 1))
)
model.add(LSTM(units=200,activation = 'tanh', return_sequences = True))
model.add(LSTM(units = 100, activation = 'tanh', return_sequences = True))
model.add(LSTM(units = 50, activation = 'tanh', return_sequences = True))
model.add(LSTM(units = 25, activation = 'tanh'))
model.add(Dense(1,))
#model.output = (None, 1, 1)
# Compilar modelo
model.compile(optimizer='adam', loss='mean_squared_error',metrics= ['mean_a
bsolute_error'])
# Entrenar modelo
loss = model.fit(X_train, y_train, epochs=400, batch_size=32)
print(type(loss.history))
plt.plot(loss.history['mean_absolute_error'])
model.output
KerasTensor: shape=(None, 1) dtype=float32 (created by layer 'dense_9')>
v #Conjunto de test:
```

```
def prepare_data_test(data, lookback):
    X, y = [], []
    for i in range(lookback+int(len(data)*0.80),len(data)):
        X.append(data.iloc[i-lookback:i])
        y.append(data.iloc[i])
    X, y = np.array(X), np.array(y)
    X = np.reshape(X, (X.shape[0], X.shape[1], 1))
    y = np.reshape(y,(-1,1))
    return X, y
X_test,y_test = prepare_data_test(df_6,lookback)
X_test2,y_test2 =prepare_data_test(df5,lookback)
X_test3 = np.vstack((X_test,X_test2))
print(np.shape(X_test3))
y_test3 = np.vstack((y_test,y_test2))
print(np.shape(y_test3))

print(len(X_train))
print(len(y_train))
print(np.shape(X_test))
y_test
#Función de desnormalización.
def desnormalizar(array,x_min=0,x_max=100):
  return np.array(array*(x_max-x_min)+x_min)
# Hacer predicciones
#X_test = np.reshape(X_test, (1, lookback, 1))
y_pred = desnormalizar(np.array(model.predict(X_test)))
y_pred2 = desnormalizar(np.array(model.predict(X_test2)))
y_pred3 = desnormalizar(np.array(model.predict(X_test3)))
y_test_des = desnormalizar(y_test)
y_test_des2 = desnormalizar(y_test2)
y_test_des3 = desnormalizar(y_test3)
# Imprimir predicciones
#print('Predicciones:')
#print(y_pred)
len(y_pred2)
len(y_test_des2)
df2 = pd.DataFrame(data={'hora': range(0,len(y_test)),
                         'test': y_test.reshape(-1,),
                         'pred': fscale(y_pred.reshape(len(y_test),))})
df3 = pd.DataFrame(data={'hora': range(0,len(y_test2)),
                         'test': y_test2.reshape(-1,),
                         'pred': fscale(y_pred2.reshape(len(y_test2),))})
df4 = pd.DataFrame(data={'hora': range(0,len(y_test3)),
                         'test': y_test3.reshape(-1,),
                         'pred': fscale(y_pred3.reshape(len(y_test3),))})
df2
df2['test'].plot()
df2['pred'].plot()
df4['test'].plot()
df4['pred'].plot()
```

```
# Obtener r2 de prueba y test y revisar si el modelo se ajusta correctament
e
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error as ms
y_train_des = desnormalizar(y_train)
print(np.shape(y_train_des))
x1 = np.array([[[0.5875],
                [0.6],
                [0.6],
                [0.5875]]])
print(np.shape(x1))
y1 = model.predict(x1)
print(y1)
y_jd = []
X_train_1[0][1:]
print(np.vstack((X_train_1[0][1:],[[0.594]])))
for i in range(0,3000):
  if i<len(X_train_1):
    x1 = X_train_1[i]
    y = model.predict(x1.reshape(1,-1,1))
    y_jd.append(y.reshape(-1,))
  else:
    x1 = np.vstack((x1[1:],y))
    y = model.predict(x1.reshape(1,-1,1))
    y_jd.append(y.reshape(-1,))
y_jd
dfjd = pd.DataFrame(data = {'jd': np.array(y_jd).reshape(-1,)})
dfjd.plot()
r2 de prueba
y_pred_entrenamiento_des=desnormalizar(np.array(model.predict(X_train)))
y_pred_entrenamiento=np.array(model.predict(X_train))
r2_entre = r2_score(y_train,y_pred_entrenamiento)
print('el r2 en el entrenamiento fue de ', r2_entre)
r2_test = r2_score(y_test3,fscale(y_pred3))
print('el r2 en el test fue de ', r2_test)
#mse de prueba
mse_train = mse(y_train,y_pred_entrenamiento.reshape(397,))
print('el mse en el entrenamiento fue de ',mse_train)
from sklearn.metrics import accuracy_score as ac

#mse de test
mse_test = mse(y_test,fscale(y_pred))
print('el mse en el entrenamiento fue de ',mse_test)
el mse en el entrenamiento fue de 0.00010743297942835282
#Podemos deducir que el modelo ha Ajustado.
#Podemos revisarlo con los datos temporales del sensor en el canal 5.
df_original = df.read_excel('data.xlsx')
df_original = df.read_excel('data.xlsx')
df['Fechas'] = pd.to_datetime(df['Fechas'])
df.set_index('Fechas', inplace=True)
<keras.engine.sequential.Sequential at 0x7f627889a3a0>
```

```
#Parte2
#Modelos basados en regresión lineal y en ANN
#para estimar el comportamiento según horas de uso.
len(df['TC6'])
502
df2['horas'] = range(2103,2103+len(df2['CT6'])) #
df2
df1= pd.DataFrame(data = {'hora': range(0,502),'TC6':df_6.iloc[:]})
df2 = pd.DataFrame(data = {'hora':df2['horas'],'TC6': df2['CT6']})
print(df1)
print(df2)
##Modelo de Regresión lineal simple
df2['TC6'] = fscale(np.array(df2['TC6']),bottom = 0, top = 100)
df2
CT6t = np.vstack((np.array(df1['TC6']).reshape(-
1,1),np.array(df2['TC6']).reshape(-1,1)))
CT6t
#Escalar las fechas
df1['hora'] = fscale(np.array(df1['hora']),bottom= 0, top= 9000)
df2['hora'] = fscale(np.array(df2['hora']),bottom= 0, top= 9000)
print(df1)
print(df2)
horast = np.vstack((np.array(df1['hora']).reshape(-
1,1),np.array(df2['hora']).reshape(-1,1)))
dft = pd.DataFrame(data = {'hora': horast.reshape(-
1,), 'CT6t' : CT6t.reshape(-1,)})
dft
# Definir tamaño de ventana temporal
#lookback = 2
# Preparar datos de entrenamiento
#X_train, y_train = prepare_data(df_6, lookback)
X_train = np.concatenate((np.array(dft['hora'],dtype=np.float32)[0:300],np.
array(dft['hora'],dtype=np.float32)[700:]))
X_train = X_train.reshape(-1,1)
print(np.shape(X_train))
y_train = np.concatenate((np.array(dft['CT6t'],dtype=np.float32)[0:300],np.
array(dft['CT6t'],dtype=np.float32)[700:]))
y_train = y_train.reshape(-1,1)
print(np.shape(y_train))
X_test = np.array(dft['hora'],dtype=np.float32)[300:700]
X_test = X_test.reshape(-1,1)
print(np.shape(X_test))
y_test = np.array(dft['CT6t'],dtype=np.float32)[300:700]
y_test = y_test.reshape(-1,1)
print(np.shape(y_test))
# Definir arquitectura de la red de regresión lineal
model = Sequential()
model.add(Dense(1,))
#model.add(LSTM(units=50, return_sequences=True, input_shape=(lookback, 1))
)
```

```
#model.add(LSTM(units=100, return_sequences=True, input_shape=(lookback, 1)
))
#model.add(LSTM(units=200,activation = 'tanh', return_sequences = True))
#model.add(LSTM(units = 100, activation = 'tanh', return_sequences = True))
#model.add(LSTM(units = 50, activation = 'tanh', return_sequences = True))
#model.add(LSTM(units = 25, activation = 'tanh'))
#model.output = (None, 1, 1)
# Compilar modelo
model.compile(optimizer='adam', loss='mean_squared_error',metrics= ['mean_a
bsolute_error'])
# Entrenar modelo
loss = model.fit(X_train, y_train, epochs=100, batch_size=32)
from sklearn.metrics import r2_score
y_pred = model.predict(X_test)
r2_lr = r2_score(y_test,y_pred)
print('r2 con regresión lineal es: ', r2_lr)
y_pred_train = model.predict(X_train)
r2_lr_train = r2_score(y_train,y_pred_train)
print('r2 en train con linear regresion es: ', r2_lr_train)
horas_train = X_train.reshape(-1,)[0:300]
temp_train = y_train.reshape(-1,)[0:300]
horas_test = X_test.reshape(-1,)[0:200]
temp_test = y_test.reshape(-1,)[0:200]
temp_pre_train = y_pred_train.reshape(-1,)[0:300]
temp_pre_test = y_pred.reshape(-1,)[0:200]
import matplotlib.pyplot as plt
plt.plot(horas_train,temp_train)
plt.plot(horas_train,temp_pre_train)
plt.plot(horas_test,temp_pre_test)
plt.plot(horas_test,temp_test)
plt.legend(['Train','Train_pred','temperaturas predichas en el test','tempe
raturas test'])
<matplotlib.legend.Legend at 0x7f1c41671e50>
model2 = Sequential()
model2.add(Dense(100,activation= 'relu'))
model2.add(Dense(50,activation= 'relu'))
model2.add(Dense(20,activation= 'relu'))
model2.add(Dense(1))
# Compilar modelo
model2.compile(optimizer='adam', loss='mean_squared_error',metrics= ['mean_
absolute_error'])
# Entrenar modelo
loss = model2.fit(X_train, y_train, epochs=100, batch_size=32)
y_pred = model2.predict(X_test)
r2_lr = r2_score(y_test,y_pred)
print('r2 con ANN es: ', r2_lr)
y_pred_train = model2.predict(X_train)
r2_lr_train = r2_score(y_train,y_pred_train)
print('r2 en train con ANN regresion es: ', r2_lr_train)
horas_train = X_train.reshape(-1,)[0:300]
temp_train = y_train.reshape(-1,)[0:300]
```

```
horas_test = X_test.reshape(-1,)[0:200]
temp_test = y_test.reshape(-1,)[0:200]
temp_pre_train = y_pred_train.reshape(-1,)[0:300]
temp_pre_test = y_pred.reshape(-1,)[0:200]
import matplotlib.pyplot as plt
plt.plot(horas_train,temp_train)
plt.plot(horas_train,temp_pre_train)
plt.plot(horas_test,temp_pre_test)
plt.plot(horas_test,temp_test)
plt.legend(['Train','Train_pred','temperaturas predichas en el test','tempe
raturas test'])
horas_train = X_train.reshape(-1,)
temp_train = y_train.reshape(-1,)
horas_test = X_test.reshape(-1,)
temp_test = y_test.reshape(-1,)
temp_pre_train = y_pred_train.reshape(-1,)
temp_pre_test = y_pred.reshape(-1,)
import matplotlib.pyplot as plt
plt.plot(horas_train,temp_train)
plt.plot(horas_train,temp_pre_train)
plt.plot(horas_test,temp_pre_test)
plt.plot(horas_test,temp_test)
plt.legend(['Train','Train_pred','temperaturas predichas en el test','tempe
raturas test'])
yp = []
for i in range(0,9000):
  x = np.array(fscale(np.array([i]), bottom = 0, top = 9000))
  y = model2.predict(x.reshape(-1,1))
  yp.append(y)
yp = np.array(yp).reshape(-1,)
df_pred = pd.DataFrame(data = {'temp' : desnormalizar(yp)})
df_pred
print(df_pred[df_pred['temp'] >= 100])
#planets_not_2008.head()
            temp
4481 100.005402
4482 100.012413
4483 100.019417
4484 100.026428
4485 100.033440
...         ...
8995 131.610352
8996 131.618210
8997 131.626053
8998 131.633881
8999 131.641693
[4519 rows x 1 columns]
```