Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Desarrollo y validación experimental de un gemelo digital para un aerogenerador

Autor: Manuel Gómez Moreno

Tutor: Carlos Bordons Alba

Cotutor: Adolfo Juan Sánchez del Pozo Fernández

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Desarrollo y validación experimental de un gemelo digital para un aerogenerador

Autor:
Manuel Gómez Moreno

Tutor:
Carlos Bordons Alba

Cotutor:
Adolfo Juan Sánchez del Pozo Fernández

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Grado: Desarrollo y validación experimental de un gemelo digital para un aerogenerador

Autor: Manuel Gómez Moreno
Tutor: Carlos Bordons Alba
Cotutor: Adolfo Juan Sánchez del Pozo Fernández

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

Este trabajo no habría sido posible sin el apoyo de muchas personas, a las que me gustaría mostrar mi agradecimiento.

En primer lugar, a mis tutores, Carlos y Adolfo, por brindarme la oportunidad de hacer este trabajo y ayudarme en todo momento. Ha sido un orgullo recibir su guía y sus consejos, que espero queden justamente plasmados.

A mi familia por su cariño, que me arrastró cada día a dar lo mejor de mí, sin su esfuerzo, dedicación y apoyo incondicional nunca habría tenido la oportunidad de estar aquí hoy, ni de ser la persona que soy.

A Beatriz, por no dudar de mí, no dejar que me rindiera y acompañarme en todo momento.

A Eduardo, por facilitar este proceso y permitirme hacer un trabajo que me motivara y completara mi formación.

Al grupo DENiM de la Universidad de Sevilla, por dedicarme su atención y compartir desinteresadamente sus conocimientos.

A Juanma, por enseñarme el camino y ayudarme a superar las dificultades que encontraba.

A mis compañeros y profesores, por las vivencias y enseñanzas de todos estos años. Sin su compañía nada de esto habría tenido sentido.

A mis amigos y todas las personas que durante estos años me dieron sus ánimos más sinceros y me ayudaron siempre que lo necesité.

Gracias a todos, de corazón.

Manuel Gómez Moreno
Sevilla, 2021

Resumen

En este trabajo se presenta el modelado y la validación de un gemelo digital para un aerogenerador, basado en redes neuronales. Concretamente, se trata de un aerogenerador de eje horizontal, tripala y de potencia nominal 2MW. La plataforma se encuentra situada en España y se ha tenido acceso a los datos históricos durante un año de diferentes mediciones sobre uno de los aerogeneradores reales de la planta.

Las redes neuronales implementadas han sido elaboradas en un entorno software del programa de computación MATLAB. Distintas arquitecturas de redes se prueban a lo largo del trabajo, así como diferentes cambios en los datos disponibles para facilitar el entrenamiento de las redes y optimizar los resultados. El objetivo es obtener un modelo capaz de predecir con precisión la potencia generada por la máquina, dadas unas condiciones externas (ambientales) e internas (configuración de los elementos del aerogenerador) determinadas. Asimismo, se realiza una validación posterior de los diferentes modelos desarrollados, gracias a los registros disponibles, en los que se lleven a cabo numerosas predicciones de datos pasados y se comparen los resultados con la potencia real de dichos instantes; de este modo se valorará la validez y la precisión de dichos modelos. Tras esto, se decidirá cuál es la mejor de las soluciones disponibles y qué trabajos futuros podrían continuar este camino.

Abstract

In this project it will be covered the modelling and validation of a digital twin for a wind turbine, based in neural networks. In particular, it is the case of a three-bladed, horizontal-axis wind turbine of 2MW of nominal power. The plant is located in Spain and we were allowed to access to the data for a whole year, including different measures of one of the turbines of the plant.

The neural networks implemented have been developed in the enviroment of MATLAB. Different network architectures are tried throughout the work, as different changes in the available data in order to ease the training of the network and the optimization of the results. The aim is to obtain a model able to predict precisely the power generated by the turbine, given some particular external (enviromental) and internal (configuration of the elements of the wind turbine) conditions. Likewise, a subsequent validation of the different models is accomplished, thanks to the avalaible records, in which numerous predictions are carried out from past values of the data, and a comparaison beetween thes results and the actual power values is set. This way, the validity and accuracy of the models is valued. Afterwards, we will make a decision about which one is the best of the possible solutions, and what kind of work can follow in the future.

Índice

Índice de Figuras

Índice de Tablas

Notación

Notación

e	Número e
$\leqslant$	Menor o igual
$\geqslant$	Mayor o igual
$b \cdot 10^a$	Formato científico
kW	Kilovatio
MW	Megavatio
kVAr	Kilovoltioamperio reactivo
$^\circ$	Grado sexagesimal
$^\circ C$	Grado Celsius
m	Metro
$\frac{m}{s}$	Metros por segundo
C_p	Coeficiente de potencia
rpm	Revoluciones por minuto
RMSE	Raíz del error cuadrático medio (*root-mean-square error*)
MAE	Error absoluto medio (*mean absolute error*)
MAPE	Error porcentual absoluto medio (*mean absolute percentage error*)

Acrónimos

ETSI	Escuela Técnica Superior de Ingeniería
DT	Gemelo Digital (*Digital Twin*)
DTP	Prototipo de gemelo digital (*Digital Twin Prototype*)
DTI	Instancia de gemelo digital (*Digital Twin Instance*)
DTE	Entorno de gemelo digital (*Digital Twin Environment*)
PLM	Gestión del ciclo de vida de un producto (*Product Lifecycle Management*)
AI	Inteligencia artificial (*Artificial Intelligence*)
ML	Aprendizaje automático (*Machine Learning*)
DL	Aprendizaje profundo (*Deep Learning*)
NN	Redes neurales (*Neural Networks*)

1 Introducción

If I have seen further it is by standing on the shoulders of giants.

Isaac Newton, 1676

Las energías renovables están en continuo crecimiento y no se les anticipa final. Las demandas de reducción de uso de combustibles fósiles para cumplir con los acuerdos para el desarrollo sostenible y el cambio climático hacen de este tipo de tecnologías su principal instrumento. La transición hacia un modelo energético que no se base en la producción de gases de efecto invernadero, tales como el CO_2, ha motivado la expansión de plantas generadoras basadas en extraer energía de elementos naturales, sin que aparezcan elementos contaminantes.

Dentro de este marco, la energía eólica es gran protagonista, siendo en España la tecnología con más potencia instalada, 27942 MW, según se recoge en [10]. Sin que ello suponga un agravio, son constantes las inversiones destinadas a potenciar la capacidad de generación eólica en nuestro país: no solo a incrementar el número de parques sino también a mejorar los ya disponibles.

En este aspecto, los avances digitales de los últimos años han permitido la aparición de tecnologías que ayudan a la optimización de los procesos. Concretamente, los gemelos digitales brindan muchas posibilidades a la hora de gestionar y optimizar las máquinas. En este contexto, la obtención de un gemelo digital para un aerogenerador capacita, entre otras cosas, a predecir la potencia generada si se conoce una predicción de las condiciones ambientales, o potenciales fallos en la máquina dadas unas características comunes en fallos previos. En general, no solo se habilita la predicción sino también la simulación, para poder obtener información de casos hipotéticos futuros que interesen.

Podría decirse que el uso de gemelos digitales en la industria permite un mejor aprovechamiento de los recursos disponibles. Y este es, en mayor medida, el paradigma actual de la industria. Conocidas las limitaciones de disponibilidad de los recursos naturales del planeta, es necesario optimizar los procesos actuales al máximo posible, reduciendo materias, costes y residuos. Son numerosos los proyectos surgidos en torno al concepto de Industria 4.0, una nueva etapa en el desarrollo industrial caracterizada por el flujo de la información y el ahorro derivado de su conocimiento. Entre ellos, se encuentra DENiM, un proyecto financiado por la UE con el objetivo de desarrollar una plataforma die intelifencia artifial que permita una aproximación colaborativa a la gestión de energía en la industria [11].

El objetivo de una industria sostenible y un futuro más respetuoso con el medio ambiente motivaron mi interés en la materia, que profundicé gracias al grupo de DENiM de la Universidad de Sevilla. Este trabajo es un pequeño desarrollo de las posibilidades que brindan las tecnologías actuales para acometer las mejoras necesarias para mantener el desarrollo actual sin comprometer el futuro.

1.1 Descripción del aerogenerador

El aerogenerador del que disponemos es de eje horizontal, tripala y de potencia nominal de 2MW. Este tipo de aerogeneradores tienen la forma típica que puede verse en Figura 1.1.

Entre sus elementos más importantes, destacan la torre (3) sobre la que se apoya la carcasa o góndola (6), estructura que contiene los elementos de generación eléctrica, como el generador (7) o la multiplicadora (10). Esta, además, hace de soporte para las palas (11), elemento principal que generará el movimiento de

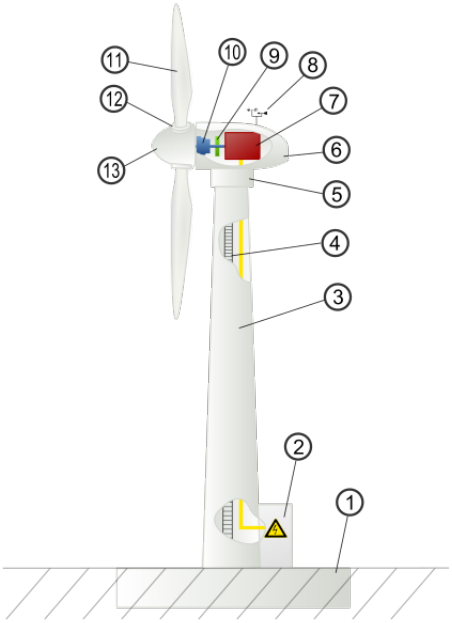


Figura 1.1 Esquema de un aerogenerador [1].

la turbina y que, gracias al sistema de giro (5) se encontrarán siempre orientadas a la dirección de la que proviene el viento.

El aerogenerador, localizado en España, se encuentra situado en una zona geográfica en la que existen fuertes corrientes de viento de manera frecuente. El funcionamiento del aerogenerador será, a grandes rasgos, aprovechar la energía cinética del viento. Esto se consigue a través de la geometría de las palas, primero, y del giro de estas, seguidamente. El proceso es el siguiente: el viento se encuentra con la oposición de las palas enfrentadas a el, con lo que se generan fuerzas en la superficie de las palas que las hacen girar. Esta transmisión de energía cinética del viento en mecánica de las palas es el principal foco de pérdidas; es imposible absorber toda la velocidad de las partículas del aire y transmitirlas al giro de una estructura. En el siguiente orden de sucesos, el eje de giro de las palas se encuentra anclado a un alternador, de modo que al girar las primeras, el alternador también girará, produciendo la corriente eléctrica.

La potencia extraíble del viento se formula como una fracción del total de la energía cinética que poseen las partículas de aire. Esta fracción tiene un máximo, descubierto en 1926 por Betz, cifrado en un 59 %, tal y como se registra [12]. Esto puede expresarse matemáticamente como:

$$P = \frac{1}{2}\rho A v^3 C_p \tag{1.1}$$

donde P es la potencia generada por el aerogenerador, ρ la densidad del aire, A el área de barrido de las palas, v la velocidad del viento y C_p el coeficiente de potencia, esto es, la fracción antes comentada.

El coeficiente de potencia, en realidad, suele ser considerablemente menor del máximo de Betz. Este factor depende, fundamentalmente, del ángulo de ataque de la pala o *pitch*, β, y del *tip-speed ratio*, λ, o ratio entre la velocidad tangencial de las palas y la velocidad del viento. Para cada pitch, existirá una única relación entre coeficientes de potencia y tip-speed ratios. De este modo, según sea la velocidad del viento y, por tanto, de las palas, se ajustará el ángulo de pitch para poder maximizar este coeficiente en todo momento. En la Figura 1.2 podemos apreciar lo que se comenta.

Resulta evidente que, a menor angulación de la pala, mayor será la extracción de la potencia del viento pues mayor superficie de la pala estará a su exposición. La regulación y el control del coeficiente de potencia es un elemento interno de la turbina eólica en cuestión, muy complejo y del que no tendremos datos disponibles; no obstante, sabemos que en todo momento la turbina funciona con el C_p óptimo.

El funcionamiento del aerogenerador se especifica durante el tiempo según diferentes estados: la mayoría describen diferentes situaciones de fallo, en las que no se está generando energía eléctrica, a excepción del funcionamiento normal o *running*, en el que sí que se produce. La máquina escogida para llevar a cabo el gemelo digital presenta el mayor número de horas anuales produciendo de las disponibles para obtener sus

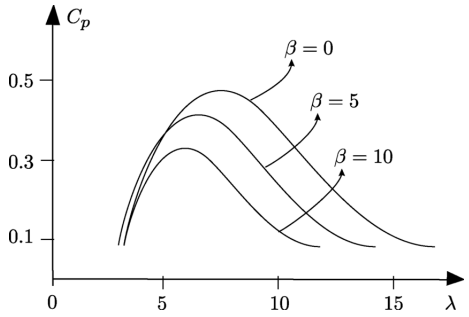


Figura 1.2 Curvas de potencia [2].

registros. Los datos, así como su obtención, son de carácter confidencial, aunque serán mencionados y se presentarán diversas gráficas a lo largo del trabajo en las que podrán apreciarse los mismos. Al respecto de los datos, tenemos a nuestra disposición un historial durante 1 año completo, concretamente del 1 de enero al 31 de diciembre de 2020, de hasta 12 magnitudes diferentes, cada una de ellas con un registro propio. En la Tabla 1.1 se presenta una relación de dichas magnitudes y las unidades en las que están registrados los valores

Tabla 1.1 Relación de datos disponibles y sus unidades.

Magnitud	**Unidad**
Dirección de viento	°
Temperatura ambiental	$°C$
Velocidad del viento	m/s
Nacelle	°
Pitch	°
Estado	(código)
Velocidad del generador	rpm
Velocidad del rótor	rpm
Potencia activa	kW
Potencia reactiva	$kVAr$
Potencia estátor	kW
Potencia rótor	kW

Con todo ello, tendremos las herramientas suficientes para poder alcanzar los objetivos de este trabajo.

1.2 Objetivos del Trabajo Fin de Grado

El objetivo principal de este trabajo es obtener el gemelo digital de una turbina eólica que permita modelar la potencia generada en el aerogenerador. Para ello, se usarán redes neuronales implementadas a través de MATLAB, con las que podremos predecir la potencia activa obtenida en la máquina, a partir de las condiciones meteorológicas y las configuraciones internas que lo condicionan. Se desarrollarán diferentes arquitecturas y configuraciones de redes neuronales para validar posteriormente todas ellas. De la discusión de los diferentes resultados extraeremos conclusiones sobre el mejor de los modelos, su precisión y posibles trabajos futuros.

Además, se tratarán el estado del arte tanto de gemelos digitales como de aerogeneradores y su relación mutua, así como la aplicabilidad a la industria y la relación con el presente trabajo.

1.3 Estructura del Trabajo Fin de Grado

La organización que sigue al trabajo es la siguiente:

Capítulo 2: Estado del arte.
Se detalla el estado del arte de los gemelos digitales y de los aerogeneradores, en secciones separadas, en las que se incluirá: definición, tipos, implementación y aplicaciones industriales que presentan.

Capítulo 3: Modelado del aerogenerador.
Se explican las razones que han llevado a tomar la decisión de usar redes neuronales para el desarrollo de los modelos. Se describen las diversas arquitecturas y configuraciones de las que se obtienen los distintos modelos. Asimismo, se detalla el tratamiento aplicado al conjunto de datos disponible para posibilitar y mejorar el aprendizaje del sistema.

Capítulo 4: Validación de los modelos.
Se analizan los resultados de cada red, tanto de los entrenamientos como de sus validaciones. Estas últimas consistirán en comparar las predicciones obtenidas en el modelo con los datos reales de los que se disponen. Se indicará la precisión de los resultados obtenidos, así como potenciales mejoras para desarrollos futuros.

Capítulo 5: Conclusiones.
Se explican las conclusiones extraídas del trabajo, las posibilidades que aportan los modelos desarrollados y posibles trabajos futuros que continúen la senda de este.

2 Estado del arte

Der philosophische Mensch hat sogar das Vorgefühl, dass auch unter dieser Wirklichkeit, in der wir leben und sind, eine zweite ganz andre verborgen liege...

Friedrich Nietzsche, 1872

Antes de introducirnos en la experimentación, parece adecuado identificar, de forma superficial, algunos aspectos básicos de las tecnologías que se van a abordar. A continuación, se explican el estado actual de los aerogeneradores y los gemelos digitales, incluyendo para estos últimos un breve repaso sobre su evolución.

2.1 Aerogeneradores

Desde los inicios del desarrollo tecnológico, el ser humano ha visto en el viento una posibilidad de aprovechamiento para sus intereses. Desde los antiguos molinos de viento usados para moler grano, la evolución técnica no se ha detenido. Actualmente, los aerogeneradores permiten la generación de energía eléctrica sin la emisión de contaminantes nocivos para el planeta, y con una tecnología versátil que abarca desde mega estructuras situadas en alta mar hasta pequeños molinos instalables en tejados.

Principalmente, existen dos tipologías de aerogeneradores, en función de la orientación del eje de rotación. En [13], el autor explica las diferencias fundamentales entre ambos:

- **Verticales:** Poseen la ventaja de que pueden operar independientemente de la dirección del viento y que la maquinaria involucrada en la generación puede encontrarse a nivel del suelo, con las consiguientes ventajas logísticas. Sin embargo, su implantación global es prácticamente inexistente desde el final de los años 1980s, siendo en esa década y la anterior en las que más se comercializó. En la Figura 2.1 podemos ver un ejemplo de este tipo de aerogeneradores.
- **Horizontales:** Existe una gran variedad en el número de palas que pueden presentar estos aerogeneradores, según sea su objetivo de construcción, indirectamente relacionadas con el *tip speed ratio*. Para la generación, se usan fundamentalmente dos tipos de turbinas:
  - Tres palas o tripala: reducen el nivel de ruido respecto de las bipala, son menos agresivas estéticamente y su momento de inercia es más sencillo de entender.
  - Dos palas o bipala: al reducir la cantidad de palas y, por tanto, la masa en el punto alto del aerogenerador, pueden construirse con estructuras más ligeras respecto de las tripalas.

  Debido a lo anterior, las turbinas tripala son las más frecuentes en las conexiones a la red eléctrica, cerca de núcleos poblacionales, mientras que las bipalas son muy usadas en el mar.

Al respecto de la localización de las turbinas, tal y como ha aparecido en el párrafo previo, existen dos tipologías de las conocidas como "granjas" de aerogeneradores: las instaladas en tierra firme, conocidas como *onshore*, y las construidas en aguas abiertas, conocidas como *offshore*. En [14], el autor hace un repaso de la evolución de este tipo de parques de generación y de cómo la viabilidad de proyectarlos no fue posible hasta asegurar suficientes beneficios que costearan la inversión. En función de las características de

Figura 2.1 Aerogenerador de eje vertical [3].

la localización, profundidad del mar, oleaje, etcétera, se deben adecuar las construcciones, que pueden ir desde cimentaciones en el lodo marino hasta plataformas flotantes atirantadas. En la Figura 2.2 podemos ver varios aerogeneradores de eje horizontal en este tipo de parques.

Figura 2.2 Aerogeneradores de eje horizontal situados en el mar [4].

España cuenta con una gran potencia instalada de energía eólica, tal y como vimos en el capítulo 1 de este trabajo. No obstante, esta tecnología sigue siendo objetivo de inversión para muchas empresas en nuestro país, y en concreto de la eólica marina, poco desarrollada en el territorio si lo comparamos con otros países del entorno.

Pero no solo a gran escala puede analizarse la energía eólica. La demanda creciente de electricidad y la necesidad de generaciones sostenibles han propiciado la aparición de aerogeneradores de pequeña potencia e incluso de autoconsumo, suponiendo una innovadora línea de desarrollo para esta tecnología. Podemos hacer una distinción de los aerogeneradores basándonos en su tamaño, indicado por el radio, tal y como se recoge en [9] y se expone en la Tabla 2.1.

Las posibilidades de la energía eólica, aún más dentro del contexto ecológico actual, son muchas. El uso de recursos naturales de manera limpia, la capacidad de adaptación de su tecnología y la innovación constante de sus elementos son grandes argumentos que garantizan su futuro.

Tabla 2.1 Clasificación de aerogeneradores basados en su tamaño y su potencia [9].

Categoría	Diámetro del rótor [m]	Potencia nominal [kW]
Micro	0.5-1.25	0.004-0.25
Mini	1.25-3	0.25-1.4
Doméstico	3-10	1.4-16
Comercial pequeño	10-20	25-100
Comercial mediano	20-50	100-1000
Comercial grande	50-100	1000-3000

2.2 Gemelos digitales

El avance en el análisis de datos y la conectividad enmarcada en el Internet de las Cosas (IoT) han facilitado la aparición de la Industria 4.0, con los gemelos digitales, o *digital twins* (DT), como punta de lanza. Tal y como se describe en [15], la integración entre el gran volumen datos disponibles y el análisis de los mismos es el principal cometido de los DT, en un entorno que facilite los análisis para la toma de decisiones en tiempo real.

En el marco industrial, el término DT describe un nuevo enfoque para transformar digitalmente los procesos a partir de la informática, mediante un ecosistema digital interconectado, con el objetivo claro de optimizar los recursos [16].

Son tres los aspectos fundamentales que intervienen en el desarrollo de un DT: la adquisición de datos, favorecida por la proliferación actual de sensores de todo tipo; la simulación, para lo que ha sido indispensable elevar el nivel de la computación hasta el de hoy día, permitiendo tratar cantidades ingentes de datos; y la comunicación entre diversos procesos, mejorada con los protocolos actuales [17].

2.2.1 Concepto

Fue en 2003 la primera vez que se acuñó el término DT, por parte de Michael Grieves junto con John Vickers [18]. Durante unos años, esta materia no tuvo mucha relevancia hasta que en 2011 se publicó el primer trabajo, a lo que siguió en 2012 la definición formal que la NASA hizo de los DT [17]. A partir de entonces, esta tecnología experimentó un crecimiento exponencial en el que se mantiene a día de hoy, con numerosas empresas apostando por el desarrollo de gemelos digitales para sus procesos industriales, como General Electric o Siemens [15].

El término DT carece de una conceptualización consensuada; de las diferentes necesidades han surgido diferentes respuestas tecnológicas, cada una con un carácter propio. Es por eso que nos encontramos muchas definiciones de DT distintas en la literatura reciente.

Grieves estableció primeramente al DT como un modelo constado de 3 partes: un producto físico real, un producto virtual y una conexión bilateral entre ambos que permite el flujo de datos e información [18].

Posteriormente, el propio Grieves retomaría la tarea para contextualizar los DT dentro del ciclo de vida de un producto. Este concepto es de especial importancia en la industra, y la gestión de este ciclo, PLM por sus siglas en inglés (*product lifecycle management*) ha aportado numerosos beneficios en la producción industrial, caracterizándose fundamentalmente por cuatro fases, explicadas en [19], que se detallan a continuación:

1. **Creación:** Diseño donde se compone el sistema a partir de las especificaciones deseadas para su comportamiento.
2. **Producción:** Realización física del sistema, en la que pueden aparecer fallos no tenidos en cuenta en la etapa anterior.
3. **Operación:** Tiempo en el que el sistema ejecuta las funciones para las que se ideó.
4. **Eliminación:** Retirada del producto, con los tratamientos adicionales que se requieran para llevarlo a cabo.

En este marco, el autor propone en [19] una renovación de su concepto de DT, al que añade los siguientes términos:

- **Prototipo de gemelo digital (DTP):** descripción del prototipo físico del elemento en cuestión. Contiene toda la información necesaria para replicar una versión virtual, entre las que se incluyen: requisitos, modelo tridimensional totalmente descrito, listas de especificaciones materiales y lista de servicios.
- **Instancia gemela digital (DTI):** descripción del elemento físico concreto al que el gemelo digital permanece enlazado durante la vida del mismo. Puede incluir, entre otras informaciones, las siguientes: descripción de la geometría mediante un modelo tridimensional, lista de componentes actuales y pasados, lista de procesos llevados a cabo para su creación y lista de servicios prestados. Una agregación de numerosas instancias podría ampliar el conocimiento acerca del producto genérico.

En cualquiera de los casos, existiría un espacio para la operación de los DT llamado entorno de gemelo digital (DTE), para diversos propósitos. Tal y como se esclarece en [19], entre ellos se pueden incluir:

- **Predecir:** el cometido del DT sería predecir valores futuros relacionados con el funcionamiento del producto físico. Según si fuera prototipo o instancia, se predecirían a partir de los componentes del producto, según unas tolerancias o según valores concretos, respectivamente.
- **Inquirir:** un DTI debería responder acerca de valores presentes y pasados de las características consideradas en su creación. Esto permitiría establecer correlaciones entre fallos del producto y dichos valores, pudiendo detectar patrones en la agregación de instancias y corregirlos en otros productos similares.

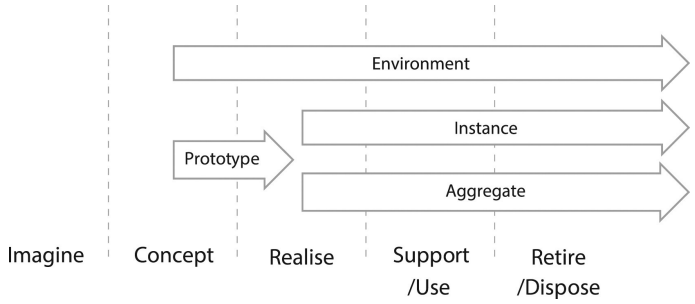


Figura 2.3 Relación entre DT y el ciclo de vida de un producto [5].

Otra herramienta útil a la hora de conceptualizar puede ser aportar definiciones concretas de aquellos conceptos clave relacionados con los DT y el ciclo de vida del producto. En [5], se aportan los siguientes conceptos identificados en una revisión sistemática de la literatura relacionada con DT:

- **Ente físico:** Entidad existente físicamente en la realidad. Una vez replicado, podría considerarse como "gemelo físico".
- **Ente virtual:** Existen múltiples dentro de un DT, cada uno con un propósito diferente. Una vez replicado, podría considerarse como "gemelo virtual".
- **Entorno físico:** Espacio físico real en el que se encuentra el ente físico y que influye en su estado.
- **Entorno virtual:** Espacio existente en el dominio digital como réplica del físico a través de metrología física.
- **Parámetros:** Tipología de información transferida entre el gemelo físico y el virtual.
- **Fidelidad:** Medida de la exactitud del gemelo, condicionada por la cantidad de parámetros, su precisión y su nivel de abstracción.
- **Estado:** Condición actual de ambos gemelos, pudiendo cuantificarse con los valores de los parámetros en cada uno.
- **Conexión de física a virtual:** Medios por los que se comunica el estado de la entidad física al entorno virtual. Esto se traduce en un cambio de los parámetros que actualizan el estado del gemelo virtual.
- **Conexión de virtual a física:** Medios que permiten un cambio en el estado del gemelo físico a partir de la información enviada por el gemelo virtual. No tiene por qué existir siempre esta posibilidad.
- **Replicación y tasa de replicación:** Es la sincronización entre los dos gemelos, entendida como evaluación de sus respectivos estados. Se lleva a cabo cada cierto tiempo determinado por la tasa de replicación o *twinning rate*.

- **Procesos físicos:** Actividades llevadas a cabo por el gemelo físico que alteran sus parámetros.
- **Procesos virtuales:**Actividades llevadas a cabo por el gemelo virtual que alteran sus parámetros.

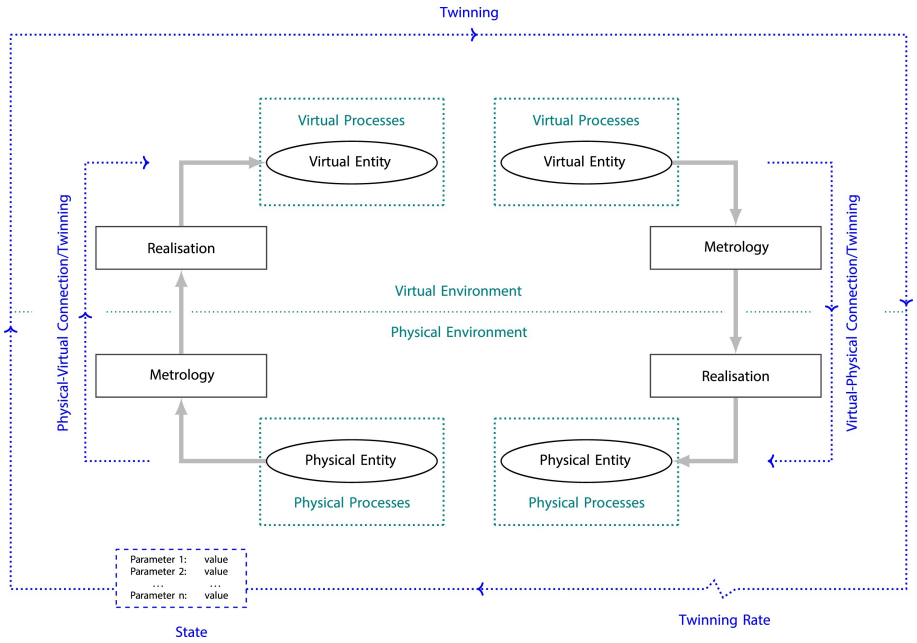


Figura 2.4 Representación esquemática de los diferentes conceptos alrededor de un DT y sus interacciones [5].

2.2.2 Aplicaciones

El uso de lo DT es amplio y puede responder a muchas demandas según como se diseñen. Una de las mejores formas de ver sus diversas aplicaciones es la propuesta en [17], en la que se distinguen varias a lo largo del ciclo de vida de un producto. Estos usos mejoran las prestaciones de los métodos tradicionales, y podemos encontrar DT orientados a los siguiente:

1. **Diseño:** permiten mejorar el diseño inicial de nuevos productos, haciéndolos más adaptables y eficientes.
2. **Producción:** habilitan la visualización en tiempo real. Además, es posible ajustar la producción de manera más sencilla y maximizar el control de la misma al disponer de simulaciones.
3. **Pronóstico y gestión de la salud:** es la aplicación más popular ya que establece una mejora evidente respecto a técnicas anteriores. Gracias a los DT se puede analizar el mantenimiento y la utilización de un producto, así como predecir su tiempo de vida útil.

El propio sofwtare que usaremos para este trabajo, MATLAB, expone en [20] algunos de los usos principales de los DT en la industria, como son:

1. **Optimización de operaciones:** Permite activar modelos que ejecuten miles de simulaciones para evaluar la aptitud de los diseños actuales y posibles mejoras que introducir.
2. **Mantenimiento predictivo:** Aplicable a la industria 4.0 para detectar cuál es el momento ideal para realizar reparaciones o reemplazar equipamiento.
3. **Detección de anomalías:** Se ejecutan modelos en paralelo que, al detectar resultados diferentes de los esperados, indican las anomalías en el comportamiento.
4. **Aislamiento de fallos:** Pueden activarse baterías de simulaciones para detectar la raíz de las anomalías y permitir aislar el fallo.

Pero lo anterior es solo un pequeño ejemplo de los usos posibles que los DT pueden tener en la industria, algo que numerosas empresas ya han podido experimentar. Podemos encontrar numerosos ejemplos de aplicaciones reales en [21], en diversos sectores industriales. En el ámbito manufacturero por ejemplo, SAP y Dassault confiaron en DT para mejorar la funcionalidad de la producción dados los requerimientos

funcionales, u otras como Siemens y PTC lo hicieron para mejorar la eficiencia y el control de calidad. Por otro lado, otras empresas como Tesla, trataron de desarrollar un DT para cada coche fabricado en un intento de permitir la transferencia simultáneas de datos entre coches y plantas; así como GE usó la plataforma Predix para construir un parque eólico digital, con un DT para cada uno de los aerogeneradores, con perspectiva a optimizar el mantenimiento, la fiabilidad y la producción energética.

Para que la implementación de los DT en la industria sea completa, es necesario hacer uso del *big data*, esto es, el conocimiento de un conjunto de datos extenso acerca del proceso en cuestión. Ambas tecnologías se complementan y resultan fundamentales para la digitalización de los entornos industriales. Tal y como se explica en [22], la creciente complejidad en los procesos solo puede ser absorbida por los datos masivos, que a su vez pueden adquirir una forma útil e interactiva en la aplicación de los DT.

El nuevo paradigma industrial ha llegado para quedarse, y tal como se indica en [23], la producción inteligente, basada en la digitalización, debe combinarse con el big data para mejorar en eficiencia, beneficios y sostenibilidad.

3 Modelado de un aerogenerador

La construction d' une machine propre à exprimer tous les sons de nos paroles , avec toutes les articulations , seroit sans-doute une découverte bien importante. . . . La chose ne me paroît pas impossible.

Leonhard Euler, 1761

Tras el repaso realizado acerca de los DT, parece evidente la necesidad de generar un modelo de aerogenerador con la máxima fidelidad posible. Entre las opciones que nos permiten desarrollar un sistema virtual que describa el comportamiento de un aerogenerador físico, encontramos dos grandes grupos: métodos deterministas y métodos estadísticos.

Los métodos causales o deterministas han marcado el rumbo de la ingeniería durante muchas décadas: el conocimiento amplio de los fenómenos físicos que intervienen en nuestra realidad han permitido explicar a la perfección el comportamiento de todo tipo de sistemas. De esta forma, los modelos basados en ecuaciones derivados de estos métodos han posibilitado predecir comportamientos futuros de los mismos, aunque con limitaciones y condicionantes estrictos, esto es, nunca fuera de unos estándares (intervalos de valores aceptables) en las variables usadas para la predicción para que el resultado fuera correcto.

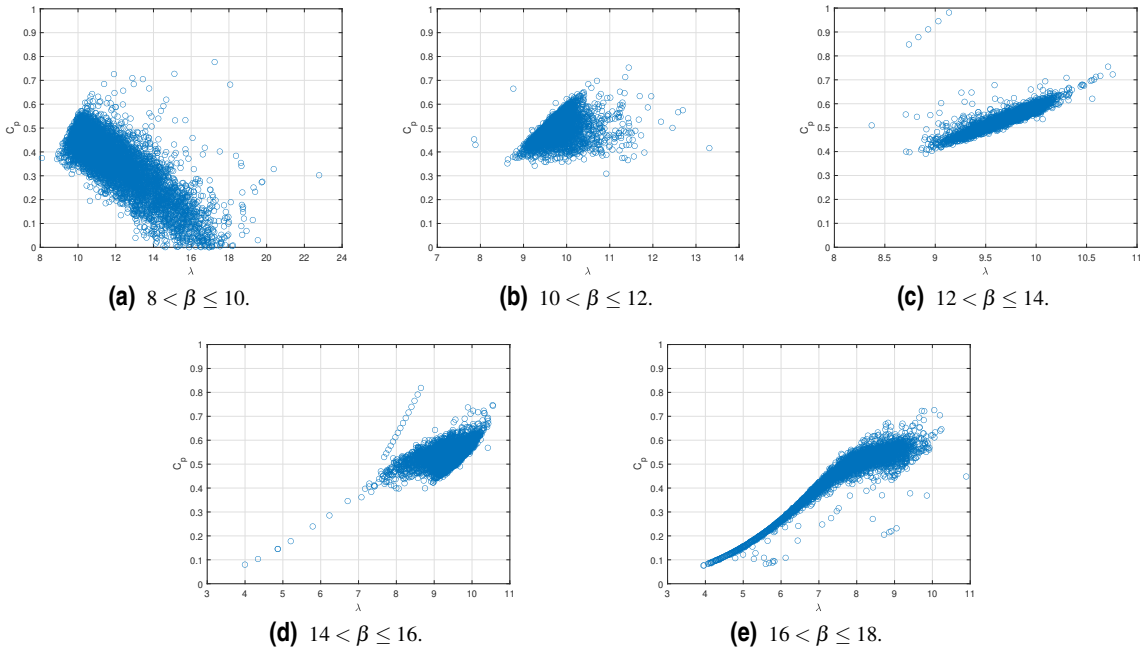
En contraposición, los modelos estadísticos no lineales recientes, introducidos en el contexto del aprendizaje automático o ML (*machine learning* en inglés), han sido capaces de mejorar la predicción en sistemas complejos respecto a métodos tradicionales. Usando como punto de partida una gran cantidad de datos históricos de las variables del sistema, sin relación preestablecida entre ellos, en lugar de una relación determinista entre las mismas tal y como hacen métodos causales, se consiguen desarrollar modelos muy precisos.

El motivo que hizo decantar este trabajo por la inteligencia artificial fue doble: por un lado, el conjunto de datos disponibles, y, por otro, y el conocimiento previo de la dificultad que entraña la física detrás de un aerogenerador.

La posibilidad que se ha tenido en este trabajo de disponer de los datos mencionados ha jugado un papel fundamental. El tener tal cantidad de valores reales de un aerogenerador facilita enormemente la tarea del entrenamiento de redes neuronales. En caso de haber optado por un método determinista, habrían sido útiles a la hora de validar el modelo comparándolo con datos reales, pero en ningún caso habría supuesto una diferencia el tener mucho menos volumen.

Si, puestos en situación, deseáramos desarrollar un modelo físico, deberíamos partir de la ecuación (1.1), completando las diferentes variables que aparecen. En principio, la velocidad del viento a cada instante no sería problema pues la conocemos. El área de barrido podría conocerse, así como la densidad del aire, que pese a no tener registro propio podría adaptarse según la temperatura ambiental. Sin embargo, el coeficiente de potencia no lo conocemos ni su deducción es trivial. La única manera de la que disponemos es, dado que tenemos datos de sobra para este tipo de análisis, despejar de esta misma ecuación el coeficiente de potencia y de ahí distinguirlo, según sea el pitch de cada instante. Esto nos permitiría representar las diferentes curvas de potencia, tal y como aparecían en Figura 1.2. Sin embargo, al plasmar esto en la realidad, nos encontramos con varios problemas. En primer lugar, esta solución propuesta solo tiene sentido en los instantes que la máquina está funcionando, con lo que cribamos los datos al conjunto que cumpla eso. Al hacerlo, nos damos cuenta de que solo existen datos significativos del coeficiente de potencia para valores de pitch de entre 8° y

18°. Esto elimina cualquier tipo de precisión al intentar obtener el C_p, pues estaríamos obviando la mayor parte de angulaciones de las palas, pero, aún solo con esos ángulos, los datos obtenidos no terminan de dibujar las curvas esperables, tal y como puede verse en la Figura 3.1.



(a) $8 < \beta \leq 10$. **(b)** $10 < \beta \leq 12$. **(c)** $12 < \beta \leq 14$.

(d) $14 < \beta \leq 16$. **(e)** $16 < \beta \leq 18$.

Figura 3.1 Diferentes curvas de potencia para diferentes ángulos de pitch.

Pese a que existen figuras como la Figura 3.1e en las que se intuye la forma de la curva de potencia, el rango de valores es tan amplio que resulta imposible poder descifrar qué valor de C_p es el adecuado para cada pitch. Estos cálculos, cuyo resultado final es obtener el *optipitch*, es decir, el pitch óptimo dado un λ, son muy complejos y las empresas dedican grupos de trabajo específicos para ello, pues cada máquina tiene sus propias curvas de potencia debido a sus especificaciones, con lo que es de esperar que no podamos sacar buenas conclusiones.

Debido a la capacidad para absorber esta incertidumbre y buscar relaciones abstractas, el ML se erige como la mejor alternativa para nuestro modelado.

3.1 Inteligencia Artificial

La digitalización en la que vivimos constituye una nueva realidad, producto de la combinación de la IA con la datificación de todos los ámbitos de la vida [24].

No es de extrañar, por tanto, la implementación de esta tecnología en ciencias e ingeniería. Buena parte de la contribución para desarrollar nuevos aspectos de la IA provienen de los conocimientos neurológicos, que permiten servir de inspiración, por un lado, y también de validación a las hipótesis artificiales si estas encuentran una relación biológica posterior, tal y como se indica en [25].

Dentro de nuestros intereses, nos enfocaremos en una de las ramas de la IA conocida como aprendizaje automático o *machine learning*, en el que se pretende que las máquinas desarrolladas tengan capacidad propia de aprendizaje. Este aprendizaje, desarrollado a través de algoritmos, puede ser de diferentes tipos, entre los que distinguiremos dos principalmente:

- **Supervisado:** El sistema tiene acceso a datos completamente definidos, de modo que debe ser capaz de hallar una correspondencia entre las entradas y la salida.
- **No supervisado:** El sistema solo conoce los datos de entrada y debe ser capaz de detectar por sí mismo las diferencias para establecer valores de salida.

En nuestro caso, dado que queremos hacer una predicción cuantificada por un valor conocido, usaremos el aprendizaje supervisado de todos los datos históricos disponibles. Además, de las diferentes técnicas

existentes para desarrollar el ML, nosotros haremos uso de las redes neuronales artificiales, que constituyen el argumento central del aprendizaje profundo o *deep learning*, DL.

El DL supone un grado más de abstracción dentro del ML, a través de transformaciones no lineales, con posibilidad de obtener buenos resultados en problemas más complejos.

Todos estos conceptos se construyen unos a partir de los otros, como ramificaciones de la materia de estudio anterior, creciendo en complejidad sucesivamente, como puede apreciarse en la Figura 3.2

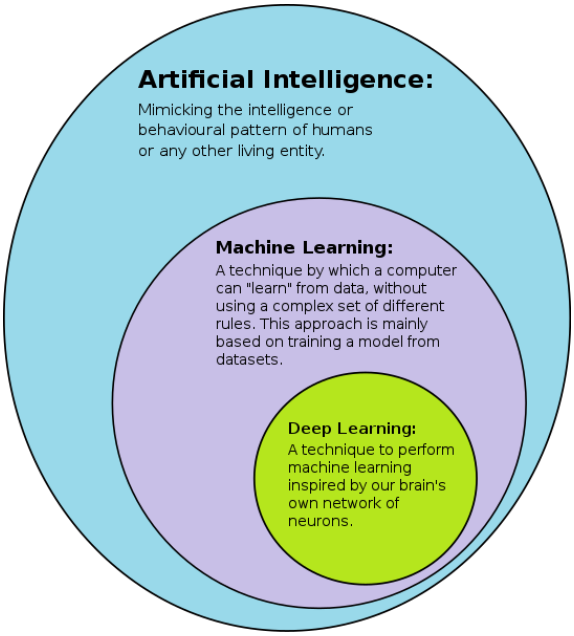


Figura 3.2 Esquematización del DL como rama del ML, y este como rama de la IA [6].

Tal y como adelantamos, nuestro algoritmo consistirá en la aplicación de redes neuronales mediante aprendizaje supervisado.

3.2 Redes Neuronales

El concepto de redes neuronales o *neural networks* (NN) proviene de una analogía directa con la biología humana. Como su propio nombre indica, son un conjunto de diferentes neuronas digitales, unidades de cálculo que procesan unos datos de entrada generando un resultado único. El esquema básico de una neurona aparece esquematizado en la Figura 3.3.

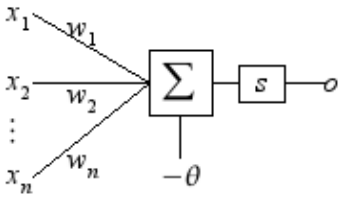


Figura 3.3 Esquema de una neurona [7].

Podemos observar los diferentes elementos que la constituyen, y que procedemos a explicar a continuación:

- **Entrada (x_i):** Son los valores de entrada a la neurona, equivalentes a las dendritas de una neurona biológica.
- **Pesos (w_i):** Son los factores multiplicadores asociados a cada una de las entradas, que cuantifican la "importancia" de cada una de ellas.
- **Sesgo o umbral (θ ó b):** Argumento adicional que se añade al conjunto de las entradas y los pesos, que componen la salida de la neurona, es decir, la entrada a la función de activación.

- **Función de activación (*s* ó *f*):** Función no lineal que recibe como argumento la salida de la neurona y la modifica según la naturaleza de esta función.

La salida de la neurona, por tanto, resulta:

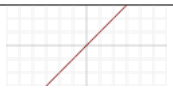
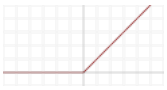
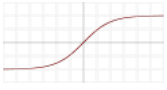
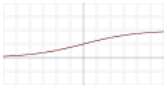
$$n = \sum_{i=1}^{i=q} (w_i \cdot p_i) + b \tag{3.1}$$

No obstante, la salida del conjunto no es esa, sino la resultante de la aplicación de la función de activación. Es decir:

$$a = f(n) = f(\sum_{i=1}^{i=q} (w_i \cdot p_i) + b) \tag{3.2}$$

Existen muchos tipos diferentes de funciones de activación, cada una con sus respectivas especificaciones. A grandes rasgos, se caracterizan por saturar los valores de la entrada en un rango de valores mucho más pequeño. No obstante, esto no es una constante en todas las funciones, ni el rango de valores de entrada que son saturados, ni el rango de salida en los que los valores se saturan. Para indicar de forma breve algunas de las funciones de activación principales y sus características, se encuentra la Tabla 3.1.

Tabla 3.1 Listado de funciones de activación y sus características principales.

Función de activación	Ecuación	Gráfica	
Lineal	$f(n) = n$		[26]
ReLu	$f(n) = \begin{cases} 0 & \text{si } n \leq 0 \\ n & \text{si } n > 0 \end{cases}$		[27]
Tangente hiperbólica	$f(n) = \frac{e^n - e^{-n}}{e^n + e^{-n}}$		[28]
Sigmoide	$f(n) = \frac{1}{1+e^{-n}}$		[29]
Escalón	$f(n) = \begin{cases} 0 & \text{si } n < 0 \\ 1 & \text{si } n \geq 0 \end{cases}$		[30]

Sin embargo, el verdadero poder de esta tecnología ocurre cuando aparecen varias neuronas en paralelo, formando una capa, y varias capas sucesivas, formando una red. En la Figura 3.4 podemos observar todas las conexiones que se crean entre neuronas al combinarlas y formar una red.

Podemos ver que, a cada neurona, le aparecen como entradas las salidas de todas las neuronas de la capa anterior. Cada una de estas conexiones tendrá su propio peso (w_{ij}^k); además, cada neurona tiene por sí misma un valor de umbral (b_i^k). Este tipo de estructuras tienen una capa de entrada, una de salida y un número indeterminado de capas intermedios u ocultas. La capa de salida tendrá tantas neuronas como parámetros simultáneos quieran predecirse. Asimismo, el tamaño de la capa de entrada dependerá del número de parámetros que sean introducidos a la red. La conjunción de todos estos valores es lo que permite en última instancia obtener un resultado concreto a partir de varios datos de entrada.

Para que los resultados sean precisos, es necesario ajustar muy bien todos los pesos y sesgos de todas las neuronas de una red. Este ajuste se realiza de forma iterativa a través de un algoritmo: se realizan los cálculos con los pesos y umbrales establecidos y se compara el resultado con el que debería ser; a partir de ahí se corrigen los diferentes valores de los pesos y sesgos. Una de las técnicas para este ajuste es el conocido como algoritmo de propagación hacia atrás o *backpropagation*. En él, para corregir los valores, se van calculando secuencialmente los errores derivados en cada una de las capas partiendo de la final a la inicial, con lo que el ajuste se desplaza "hacia atrás". Esta será, concretamente, la técnica que usemos en nuestras redes.

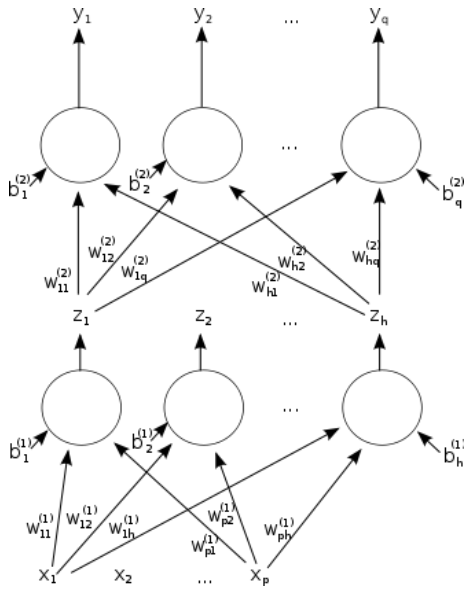


Figura 3.4 Esquema de una red neuronal [8].

3.3 Datos disponibles

Tal y como hemos comentado, tenemos a nuestra disposición un historial de datos de un aerogenerador real. Se dispone de hasta 12 registros, cada uno con sus valores de tiempo correspondiente, de modo que para cada magnitud aparece un listado con dos valores en el que se reflejan, por un lado, el dato de la magnitud y, por otro, el día y la hora de dicho registro. No obstante, cada magnitud cuenta con un número de valores diferente, pues no todas las magnitudes se han medido con la misma frecuencia a lo largo de un año. En la Tabla 3.2 encontraremos el número de valores disponibles para cada magnitud.

Tabla 3.2 Relación del número de valores por magnitud.

Magnitud	Cantidad de valores
Dirección de viento	870575
Temperatura ambiental	43254
Velocidad del viento	857741
Nacelle	60166
Pitch	333130
Estado	6695
Velocidad del generador	846408
Velocidad del rótor	481099
Potencia activa	937540
Potencia reactiva	933300
Potencia estátor	904533
Potencia rótor	305507

Como cabe imaginarse, si cada magnitud difiere en la cantidad de datos disponibles, el registro no contempla a todas en los mismos instantes. Para nuestro objetivo posterior, que es entrenar una NN, necesitamos que los datos a introducir tengan los mismos instantes de referencia para todas las magnitudes, en definitiva, que estén sincronizados. Es por ello que nuestra primera tarea con los datos será la sincronización para obtener un nuevo registro con una serie temporal completa de datos de todas las magnitudes.

3.3.1 Sincronización

Comenzamos analizando en qué forma se nos presentan los datos. Existen dos ficheros entre los que se han dividido semestralmente el conjunto de datos. En cada uno de ellos, aparecen 24 columnas distribuidas en parejas con los registros de tiempos y valores de una magnitud concreta; el número de filas de cada columna depende del número de valores disponibles de la magnitud correspondiente.

Proponemos la sincronización de los datos de la manera más sencilla posible: establecer una referencia temporal común a todos los registros de las distintas magnitudes y volcar estos en su franja temporal correspondiente. Dado que el número de valores de cada magnitud es distinto, habrá muchos casos en los que, para una determinada magnitud y un determinado instante, no haya dato disponible. Más adelante veremos como resolver esto.

Para continuar, es fundamental averiguar qué tiempo de muestreo tienen los datos; en función de este construiremos la referencia temporal. De forma general, los datos se actualizan en múltiplos de 15 segundos; por ejemplo, una secuencia encontrada en una de las magnitudes es:

'01-May-2020 09:32:30'
'01-May-2020 09:41:45'
'01-May-2020 09:43:45'

Al igual que en este caso ocurre análogamente en todas las magnitudes, con lo que parece una buena elección este tiempo de muestreo. No obstante, existen saltos temporales entre las 3 secuencias descritas, tal y como preveíamos, y que también ocurrirá en todos los registros disponibles. A lo anterior hay que reseñar que hay una magnitud que no cumple con la multiplicidad de 15 segundos: el estado de la máquina. Este registro no tiene ninguna característica respecto a sus periodos de muestreo, como podemos apreciar en la siguiente secuencia:

'13-Mar-2020 11:33:58'
'13-Mar-2020 13:55:27'
'13-Mar-2020 13:56:38'

De todos modos, siendo el tiempo de muestreo 15 segundos, el mayor error cometido al modificar estos tiempos sería de 14 segundos, algo insignificante para los tiempos necesarios en cambiar la dinámica de la máquina cuando cambia su estado, con lo que seguimos aceptando la idea propuesta.

Una vez analizada la validez de la solución propuesta, pasamos a su realización: construimos una tabla vacía por semestre con las dimensiones adecuadas cuya primera columna será un registro temporal de fecha y hora cada 15 segundos, desde las 00:00 del 1 de enero a las 23:45 del 30 de junio y desde las 00:00 del 1 de julio a las 23:45 del 31 de diciembre, respectivamente. El siguiente paso será, por cada una de las tablas, magnitud a magnitud, volcar los datos en los tiempos adecuados. El algoritmo desarrollado para implementar esto es el siguiente:

1. Leer el registro temporal del primer dato disponible de la magnitud.
2. Comparar dicho registro con el del índice (primera columna) de la tabla.
3. Si el índice es más pequeño, pasamos al siguiente dato disponible. Si no lo es, copiamos en la columna correspondiente de la nueva tabla, y en la fila de dicho índice, el valor del dato correspondiente al registro leído.
4. Desechar el registro ya leído.

Esto deberá repetirse en cada magnitud tantas veces como datos disponibles haya, algo diferente para cada uno de los registros.

Finalmente, acoplaremos las dos tablas semestrales en una única que contemple los datos de todos los registros con un índice temporal común.

3.3.2 Rellenado

Tal y como advertimos, existen muchos registros sin datos. Dado que no todas las magnitudes se registraban a la vez, hay muchos instantes de tiempo con la mayor parte de registros vacíos.

Una posible solución sería adentrarnos en el entrenamiento de las NN con estos datos "crudos". Esto implicaría un menor número de datos válidos para el desarrollo de la red, y un presumible peor comportamiento.

La otra solución posible consistiría en modificar estos datos de forma que quedaran rellenados todos los registros. Existen numerosos tipos de tratamientos disponibles en el software de MATLAB; sin embargo, con el objetivo de hacer más sencillo el proceso, optaremos por dos opciones principalmente: interpolar linealmente entre los dos valores no nulos más próximos y mantener el último valor no nulo hasta que aparezca otro nuevamente; escogiendo cada uno según la magnitud en cuestión.

La interpolación lineal consistirá en tomar la variación de valor entre estos dos datos no nulos más próximos y dividirla entre el número de datos nulos que los separa, obteniendo una tasa de variación; esta tasa se añadirá sucesivamente al valor anterior, desde el primer dato no nulo hasta el último, que será el otro dato no nulo. Este método lo usaremos para la mayoría de magnitudes cuyo valor es altamente variable y aparecen diferencias significativas entre los dos datos no nulos más próximos. Concretamente, lo aplicaremos en: dirección y velocidad del viento, temperatura, velocidad del generador y del rótor, ángulo *nacelle* y todas las magnitudes de potencia. Así, en cada una de estas magnitudes, obtendremos una progresión lineal entre los valores no nulos que, vistos en una gráfica, responderían a una recta.

Adoptando la interpolación lineal como la solución al problema, debemos hacer unas modificaciones previas a los datos. En efecto, si aplicamos esta transformación al conjunto de datos de una de las magnitudes, encontraremos resultados incoherentes en los primeros y últimos datos del registro. Esto se debe a que si los primeros (o últimos) datos del registro de una magnitud son nulos, no existen dos valores no nulos con los que aplicar la interpolación, con lo que esta se toma entre el primer (o último) valor no nulo e infinito. Para corregir esto, aplicaremos a los primeros y últimos elementos nulos una transformación distinta de la interpolación: copiaremos en todos ellos el primer y último, respectivamente, valor no nulo de la serie.

Resuelta, por una parte, la interpolación lineal, nos centramos ahora en el otro método consistente en reproducir el valor último no nulo. Esta transformación es preferible en magnitudes que tengan pocas variaciones significativas, como el ángulo *pitch* y también será fundamental para el estado de la máquina, donde los datos numéricos tienen significado propio y los resultados de interpolaciones generarían incongruencia. Es decir, la máquina mantiene el último estado registrado hasta que un nuevo dato difiera y lo cambie.

3.3.3 Valoración datos

Después de los distintos cambios efectuados a los datos, parece prudente analizar la calidad de los cambios efectuados antes de introducirlos en cualquier red.

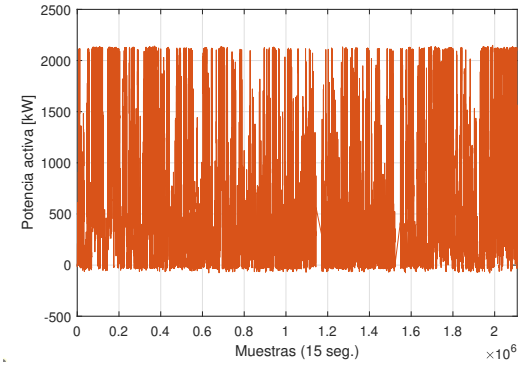
En una primera aproximación, podemos ver en dos de las magnitudes más significativas, la velocidad del viento y la potencia producida, cómo se distribuyen los datos a los largo del año. En la Figura 3.5 se aprecian dos tramos con resultados muy pobres, donde la interpolación tuvo lugar entre muchas muestras de datos, con lo que esos datos generados no tienen ninguna significación.

Es evidente como una falta de datos degenera en pobres resultados al interpolar: estos "espacios vacíos" en los datos se transforman en rectas que no añaden ningún valor a lo conocido y que pueden afectar negativamente al entrenamiento de las redes posteriores.

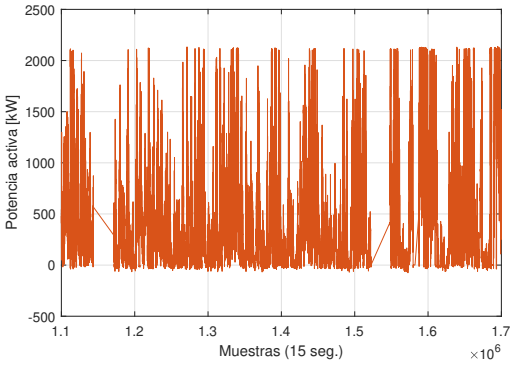
Una posibilidad para mejorar este fallo concreto puede ser, simplemente, retirar este grupo de datos del conjunto disponible. No obstante, dado que nuestro objetivo principal no es optimizar los datos para el modelo posterior sino analizar los pasos a seguir para la construcción y validación del mismo, lo dejaremos como una posibilidad a explorar.

Otro ejemplo donde apreciar la validez de los datos es usar la curva de potencia de un aerogenerador. Esta curva, que enfrenta la potencia producida por la máquina a la velocidad del viento, tiene tres tramos característicos: una pendiente, en la que la potencia es proporcional a la velocidad del viento, sin llegar nunca a su máximo; otra de saturación, en la una vez alcanzada la potencia máxima se mantiene en ese rango pese a incrementos del viento; y una última de de caída, en la que para velocidades mayores de las admisibles, el aerogenerador se desconecta y deja de producir.

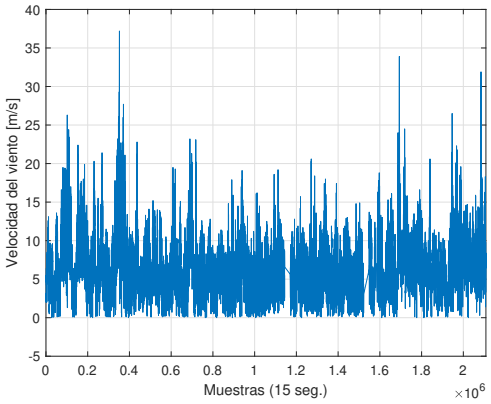
A este respecto, encontramos la curva de potencia con los datos disponibles en la Figura 3.6, tanto los reales como los rellenados. En ella, se puede apreciar como la curva describe los tramos explicados, así como una zona común de potencia nula transversal a todas las velocidades del viento, debido a desconexiones del aerogenerador. Si nos centramos en los datos interpolados, podemos ver como casi en su mayoría encajan dentro de las zonas ocupadas por datos reales, a excepción de algunas rectas muy marcadas. Parece evidente que estas excepciones son producto de la interpolación entre dos datos generando resultados poco verosímiles una vez vistos en esta representación.



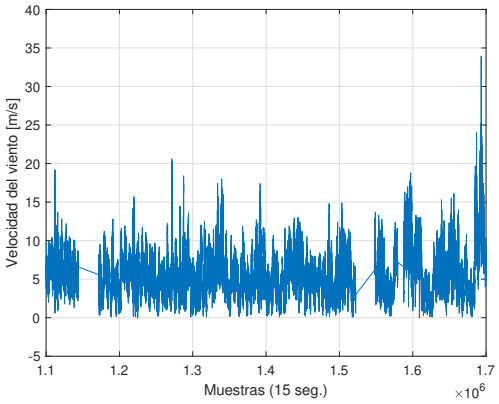
(a) Datos rellenados de potencia activa.



(b) Tramo de datos de potencia activa mal rellenados.



(c) Datos rellenados de la velocidad del viento.



(d) Tramo de datos de la velocidad del viento mal rellenados.

Figura 3.5 Análisis de los datos rellenados.

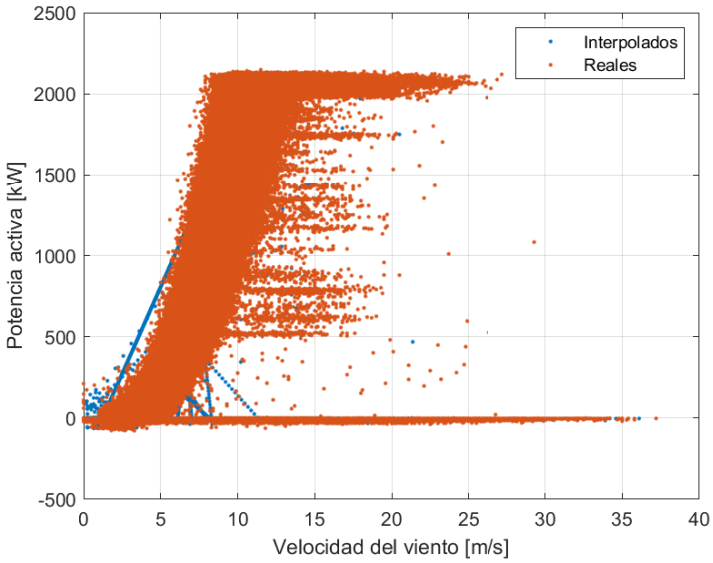


Figura 3.6 Curva de potencia para los datos disponibles.

Al igual que el caso anterior, estos datos anómalos podrían identificarse y ser eliminados en todas las magnitudes, para evitar malos resultados posteriores.

En cualquier caso y pese a los fallos comentados, parece que la interpolación de los datos ha proporcionado resultados válidos y útiles para el entrenamiento de NN, que es nuestro fin último.

3.3.4 Medias diezminutales

Además de lo anterior, existen otras soluciones posibles para mitigar los fallos de las interpolaciones. Una de ellas sería la de tomar valores medios en tramos equiespaciados a lo largo de todo el histórico y en todas las magnitudes. En concreto, para un registro de un año con muestras cada 15 segundos, tomar la media de periodos de 10 minutos parece una propuesta razonable.

Este nuevo conjunto de datos, más reducido, podría simplificar el entrenamiento de las redes posteriores y reducir su complejidad, con lo que podríamos obtener resultados más robustos aunque menos precisos.

Sin embargo, esto supone la aparición de nuevos problemas, distintos de los previos. Por un lado, gran parte de los valores del estado de la máquina pueden perder su sentido; para que aparezca un número coherente en el dato diezminutal, el estado de la máquina no debe haber variado en esos diez minutos a los que se les calcula la media. Esto no siempre ocurre y se pierden bastantes datos aceptables por desconexiones puntuales del aerogenerador, aunque dado que el número de valores con el estado de la máquina en funcionamiento es lo suficientemente significativo, podríamos seguir adelante con ello. Por otro lado, existe un problema con la veracidad de los datos que tomemos como medios, y para eso no podemos perder la perspectiva del funcionamiento de la máquina. La potencia, en primer lugar, y el resto de variables que dependen de ella, en segundo, son fundamentalmente producto de la velocidad del viento a cada instante. Este fenómeno físico es, cuanto menos estable y homogéneo, con lo que tomar medias del valor de la velocidad del viento cada diez minutos haría más difícil la interpretación de la información: un valor diezminutal de 10 m/s podría ser que durante diez minutos la velocidad del viento fue aproximadamente de 10 m/s, con lo que la máquina produjo en todo momento; o bien que la mitad del tiempo fue de 20 m/s y la otra mitad de 0 m/s, con lo que la máquina solo produjo la mitad del tiempo.

Para ejemplificar esto, podemos observar la Figura 3.7, donde se aprecian zonas en las que se omite información importante, pues en el plazo de diez minutos la producción puede llegar a duplicarse o a reducirse la mitad.

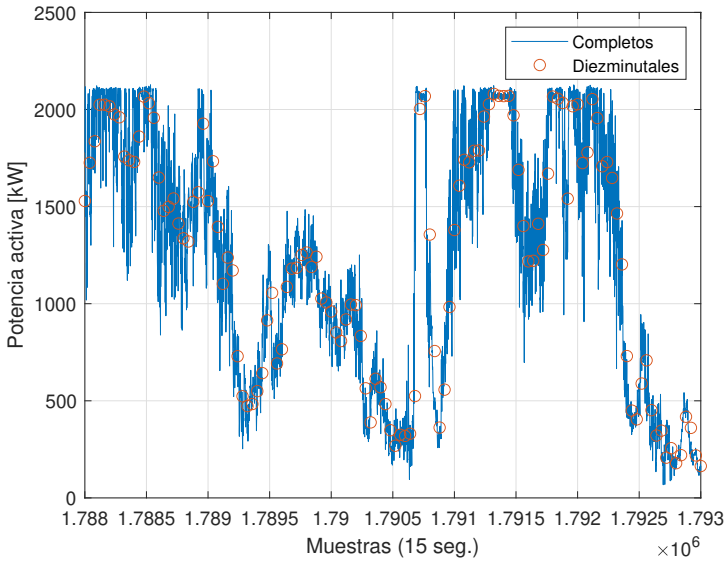


Figura 3.7 Datos de potencia activa completos y diezminutales.

Es por esto por lo que, aunque hayamos obtenido estas medias para evaluarlas y tenerlas a nuestra disposición, no haremos uso de ellas en apartados posteriores.

3.3.5 Saturación de potencias negativas

Otro de los potenciales problemas para el entrenamiento de la red es la aparición de valores negativos en la potencia activa generada; el hecho de introducir valores menores a cero complica la validación de la red ya

que necesitaría muchas más operaciones, esto es, muchas más neuronas, para ser capaz de precisar tanto valores positivos, de hasta 2 millares, como negativos, de unas pocas decenas.

La potencia negativa aparece en el aerogenerador, principalmente en los momentos de desconexión de la máquina, donde absorbe parte de la potencia de la red. Una posible opción seria desechar el conjunto de datos en los que la máquina no funciona, pero existen instantes anteriores y posteriores en los que sigue apareciendo esta potencia inusual, tal y como podemos apreciar en la Figura 3.8, donde la mayor parte de valores negativos no pertenecen a momentos en los que la máquina está funcionando ($estado = 100$).

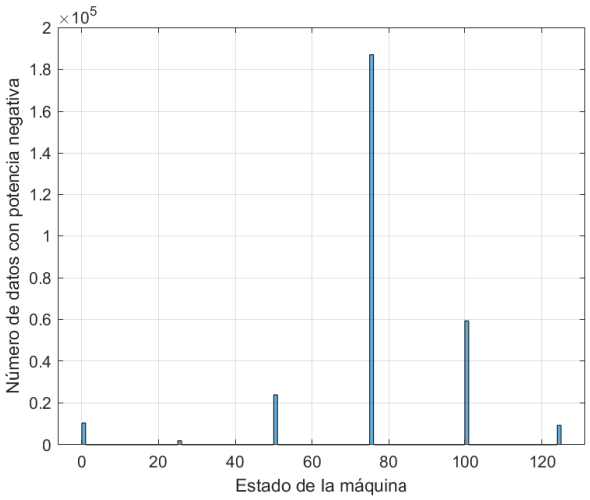


Figura 3.8 Histograma del estado de la máquina para valores de potencia activa negativos.

Una posible vía para enmendarlo será saturar los valores mínimos de la potencia a 0 en el conjunto de datos que se le entreguen a la red neuronal para su entrenamiento, siempre que los resultados obtenidos en la red con el conjunto original sean excesivamente pobres. Esto ocurrirá, principalmente, en los modelos más sencillos, de una sola capa y pocas neuronas.

No hay que olvidar que la opción anterior hará que la red sea incapaz de predecir valores negativos ante valores de entrada reales, pero mejorará en el resto de predicciones, las cuales consideramos más importantes. El objetivo último de la red es predecir con cierta precisión la potencia cuando la máquina está produciendo, no tanto cuando la máquina claramente está parada, con lo que pequeños errores en la potencia predicha en ese último caso no suponen un inconveniente suficiente como para descartar esta propuesta.

Así pues, aplicaremos está saturación de la potencia en ciertos modelos que se verán más adelante.

3.4 Modelos desarrollados

Para este trabajo, haremos uso del toolbox disponible en el software de MATLAB. En concreto, haremos uso de las conocidas como *shallow neural networks*, un tipo de NN sin excesiva complejidad, con pocas capas ocultas, con la que haremos predicciones.

Dado que el objetivo último de un aerogenerador es proporcionar energía, trataremos de predecir los valores de potencia activa generada por la máquina según las condiciones en las que funcione (externas e internas).

Con esta perspectiva, se desarrollan a continuación diferentes modelos en los que se probarán diferentes configuraciones de NN, en las que cambiarán tanto los parámetros de entrada como las arquitecturas (capas y número de neuronas).

Los primeros pasos los daremos con la guía de inicio del propio toolbox de manera que se mantendrán las configuraciones por defecto; tan solo modificaremos el número de neuronas que conforman la única capa oculta, así como los parámetros de entrada a la red. Posteriormente, pasaremos a programar manualmente las redes e introducir configuraciones más complejas y personalizadas.

Tal y como hemos comentado, para las entradas a la redes usaremos las magnitudes que condicionan el entorno de operación de la máquina, así como aquellas que caractericen el funcionamiento interno de la misma. De ambas tipologías depende la producción de energía del aerogenerador, directamente. Según la

complejidad que pretendamos asumir en cada modelo, añadiremos más o menos magnitudes con todos sus datos.

Como mínimo, serán necesaria la velocidad del viento, que es en última instancia la que determina la capacidad de energía disponible para su aprovechamiento. No en vano, añadiremos como mínimo la temperatura, que condiciona la densidad del aire y, por tanto, la cantidad de masa que impacta en las palas, influyendo en la fuerza de empuje; así como la dirección del viento y la angulación de la pala o pitch, que especifican la posición del aerogenerador respecto del viento y que no tiene por qué ser óptima en todo momento, con lo que es necesario su conocimiento. A estas magnitudes se le añadirán progresivamente otras en los sucesivos modelos, incrementando la complejidad pero también la precisión de los mismos.

Por último, antes de pasar a revisar los distintos modelos, debemos hablar de la distribución de los datos para el entrenamiento de las redes. Por sencillez y para facilitar la comparación entre los modelos a desarrollar, se usará la misma configuración en los conjuntos de datos de entrenamiento, validación y test en todos los casos. Concretamente, escogeremos un cuarto de todos los datos disponibles (un año), que usaremos como conjunto que se pone a disposición del software. El resto de datos, tres cuartas partes, los usaremos como medida de la precisión de la red, a modo de validación cruzada, testando en un conjunto diferente del proporcionado, ya que dado que son muchos datos y queremos evitar sobreajuste u *overfitting*[1] en inglés. En cuanto al conjunto de datos entregados a MATLAB, el propio programa establece una división de los mismos en los tres subconjuntos mencionados: entrenamiento, validación y test. Ya que llevaremos a cabo un testeo de manera independiente, reducimos al mínimo, un 5 %, la cantidad de valores que se asignan a este subconjunto. Por otro lado, asignaremos al entrenamiento un 60 % de los datos del conjunto, dejando un 35 % al subconjunto de validación para que, en sucesivas iteraciones, mejore la precisión de la red.

Con respecto al entrenamiento y la configuración de las neuronas, existe la posibilidad de inicializar toda la red con valores predeterminados. Otra de las opciones es que el propio programa los establezca de manera aleatoria. Dado que no conocemos ninguna indicación previa acerca de cómo mejoraría una inicialización concreta de los pesos o *w, weigth*, y los sesgos o *b, bias*, decidimos que sea MATLAB quien lo asigne aleatoriamente. Pese a ello, sí que definiremos la semilla de aleatoriedad antes de cada entrenamiento, esto es, que en todos los procesos el software tome los mismos números aleatorios, para que si modificamos una misma red y la simulamos en distintas ocasiones, los resultados sean comparables.

Sin más que comentar al respecto, se exponen a continuación los diversos modelos desarrollados.

3.4.1 Modelo A

Comenzamos de la manera más sencilla posible, partiendo de la guía de uso del toolbox de MATLAB.

La primera decisión es escoger los parámetros de entrada, así como la magnitud a predecir. Tal y como hemos comentado, los parámetros escogidos son los mínimos necesarios y la magnitud predicha será la potencia activa, como se muestra en la Tabla 3.3.

Tabla 3.3 Magnitudes de la red neuronal A.

Entradas	**Salida**
Dirección del viento	Potencia activa
Temperatura ambiental	
Velocidad del viento	
Pitch	

Dado que no hemos introducido en los parámetros de entrada el estado de la máquina, debemos tratar de compensar esta falta de información de la red de algún modo. Para ello, filtraremos los datos introducidos en el software, de modo que del conjunto especificado para MATLAB eliminaremos todos aquellos en los que la máquina no esté funcionando.

En cuanto a la arquitectura de la red, la propia guía nos fuerza a una red de una sola capa oculta con las funciones de activación definidas; tan solo podemos variar el número de neuronas. Siendo este el caso, escogemos 20 neuronas para dicha capa, 5 veces más del número de parámetros de entrada. Este número parece adecuado: no es excesivamente grande como para poder programar la red en un futuro, pero tampoco es pequeño, lo que permite cierta capacidad de abstracción a la red para predecir. En cuanto a a las funciones

[1] Adecuación excesiva de la red a los datos con los que entrena de forma que, al simular posteriormente con datos significativamente diferentes de los anteriores, los resultados sean deficientes.

de activación mencionadas, nos encontramos con una sigmoide a la salida de la capa oculta y con una función lineal a la salida de la red, para obtener un valor numérico en la predicción. Podemos ver la disposición de la red en la Figura 3.9, donde se aprecia que existen 20 neuronas en la capa *hidden*, cada una con su peso y su sesgo con salida a la función de activación, y la posterior capa de salida.

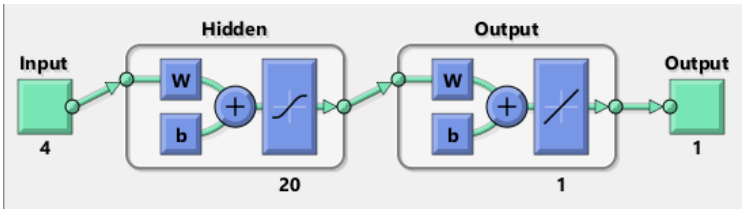


Figura 3.9 Arquitectura del modelo A.

3.4.2 Modelo B

Podemos prever que, pese a que el modelo anterior tiene neuronas suficientes, los pocos parámetros de entrada pueden resultar en unas predicciones pobres. Entre las posibles mejoras a esto, existe la posibilidad de incluir el estado de la máquina entre las magnitudes de entrada; de esta forma no se perderían muchos de los datos al filtrar los que corresponde a la máquina en funcionamiento. Además, es probable que esto mejore la predicción en los momentos en los que la máquina no esté funcionando, instantes en los que el modelo previo tiene bastantes probabilidades de fallar al no haber sido entrenado con datos de ese tipo.

De esa forma, los parámetros escogidos son los que se muestran en la Tabla 3.4.

Tabla 3.4 Magnitudes de la red neuronal B.

Entradas	**Salida**
Dirección del viento	Potencia activa
Temperatura ambiental	
Velocidad del viento	
Pitch	
Estado	

Por otro lado, mantendremos la misma estructura que en el Modelo A, para poder comparar ambos y decidir si con la arquitectura propuesta en ellos es suficiente.

3.4.3 Modelo C

Para el siguiente modelo, tomaremos una estrategia totalmente distinta de los dos anteriores. En lugar de usar la guía proporcionada por MALTAB, desarrollaremos el código que programe totalmente la red neuronal, pudiendo personalizar mucho más su configuración.

Entre los cambios, el más destacable es la introducción de una segunda capa oculta. Dado que en los Modelos A y B solo usamos una sola capa, la mejor manera de aumentar la precisión de los resultados sin influir sobre el número total de neuronas es aumentando el número de capas. Por eso, vamos a distribuir las 20 neuronas en dos capas, una primera de 16 y una segunda de 4, tal y como puede verse en la Figura 3.10. A la nueva capa introducida se le pondrá como función de activación una sigmoide, al igual que la ya existente.

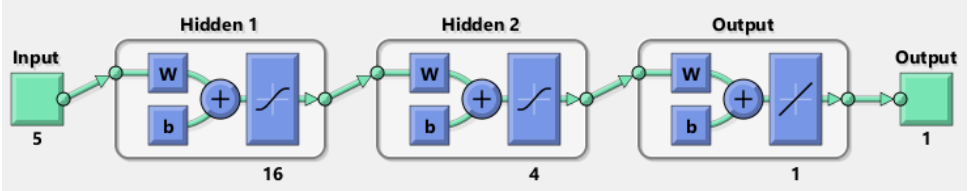


Figura 3.10 Arquitectura del modelo C.

En referencia a los parámetros, no se van a ver modificados con respecto del Modelo B. La principal razón es que solo existe un parámetro más que pueda indicar las condiciones del aerogenerador y su entorno sin que impliquen una relación directa con la potencia: la nacelle. Dado que esta magnitud es poco significativa, pues el aerogenerador se orienta siempre de cara al viento, se consideran suficientes las ya expuestas como para poder dar una buena predicción.

3.4.4 Modelo D

En este último caso se propone una configuración mucho más ambiciosa respecto a todas las anteriores. Se pretende con ello demostrar que, si quitamos la imposición de límites por complejidad, la predicción puede llegar a ser tan precisa como se desee.

Así, añadiremos los datos de la nacelle, último parámetro restante por incluir, quedando el listado completo tal y como aparecen en la Tabla 3.5.

Tabla 3.5 Magnitudes de la red neuronal D.

Entradas	Salida
Dirección del viento	Potencia activa
Temperatura ambiental	
Velocidad del viento	
Pitch	
Estado	
Nacelle	

Del mismo modo, se modificará la arquitectura del sistema para añadir más grados de abstracción: añadiremos una capa oculta más, intermedia entre las existentes en el Modelo C, con 8 neuronas y función de activación sigmoidal. El resto permanece idéntico al caso anterior, resultando como se muestra en la Figura 3.11.

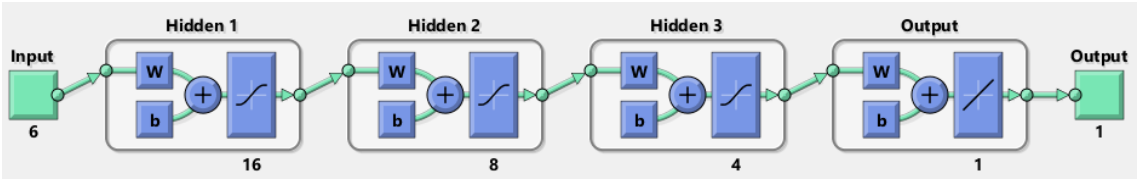


Figura 3.11 Arquitectura del modelo D.

De este modo, se establecen diferentes grados de complejidad entre los modelos y podremos apreciar la precisión respectiva de cada uno, pudiendo hacer comparaciones y extraer conclusiones de las mismas, lo que se llevará a cabo en el capítulo 4.

4 Validación de los modelos

Whether you can observe a thing or not depends on the theory which you use. It is the theory which decides what can be observed.

Albert Einstein, 1926

Tal y como adelantábamos, nos disponemos a continuación a validar los modelos presentados anteriormente y lo haremos en dos fases para cada uno de los modelos: primeramente, analizaremos el entrenamiento de los datos llevado a cabo en cada una de las redes; posteriormente, comprobaremos la precisión de las predicciones de potencia para todos los modelos.

El rendimiento del entrenamiento de una red neuronal tiene varias maneras de cuantificarse, siendo las más básicas el tiempo necesitado para el proceso así como el número de iteraciones, o *epoch*, llevadas a cabo, esto es, el número de "correcciones" que las neuronas han precisado para alcanzar la configuración óptima. Esta configuración óptima es, como podemos imaginar, la que adquiere una mayor precisión en los resultados. Sin embargo, para llevar a cabo esta tarea, el programa tiene definido un algoritmo interno que se encarga de mejorar la red a cada iteración del entrenamiento. En nuestro caso, de todas las posibilidades que MATLAB brinda, se ha optado por el algoritmo *Levenberg-Marquardt backpropagation*, que actualiza los valores de los pesos y sesgos según la optimización de Levenberg-Marquardt. El algoritmo, tal y como se describe en [31], consiste en simplificar la matriz hessiana, H, y el gradiente, g, a través de la matriz jacobiana, J, definida como las primeras derivadas de los errores de la red con respecto a los pesos y sesgos, y del vector de errores, e, de la siguiente forma:

$$H = J^T J \tag{4.1}$$

$$g = J^T e \tag{4.2}$$

De esta forma, el algoritmo implementa una solución similar a la de Newton, tal como sigue:

$$x_{k+1} = x_k - [J^T J + \mu I]^{-1} J^T e \tag{4.3}$$

Es un método recomendado para problemas de regresión en NN con tamaño medio y que destaca por su velocidad de ejecución, con lo que parece adecuado para nuestro caso de estudio.

Por su parte, la validez de los resultados será examinada de dos formas: una analítica, a través de diferentes métricas de regresión, y otra visual, a través de la gráfica con las predicciones del año completo.

Entre las distintas métricas de regresión, van a evaluarse dos: el error medio cuadrático o RMSE y el error absoluto medio o MAE, que se aplicarán a los errores de la predicción de todo el año. El RMSE es una medida ampliamente utilizada en el cálculo de precisión, aunque cuenta con la desventaja de ser altamente sensible a valores atípicos debido a su carácter cuadrático; valores que, en nuestro caso, pueden ser medianamente frecuentes. Por otro lado, el MAE tiene en cuenta la media de todos los errores cometidos, con lo que es más robusta ante valores atípicos pero proporciona poca información acerca de posibles características de la predicción, como pudiera ser una muy buena predicción de valores de producción pero una muy mala de

valores nulos. Ambas métricas se definen de la siguiente forma:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{4.4}$$

$$MAE = \frac{\sum_{i=1}^{n}|\hat{y}_i - y_i|}{n} \tag{4.5}$$

Además de lo anterior, se contempló la posibilidad de añadir como una métrica más el error porcentual absoluto medio o MAPE. La característica principal de esta medición es que aporta un carácter relativo a los errores que las otras métricas no contemplan: no será lo mismo predecir 1500 cuando la máquina este produciendo, en torno a 2000, que cuando la máquina esté parada, en torno a 0. Sin embargo, tal y como puede identificarse en su definición, para valores nulos o cercanos a 0, pequeños errores pueden suponer porcentajes extremadamente altos, que desvirtúen por completo la medida. Es por esto por lo que finalmente no se ha optado por su inclusión, pues estaría fuertemente condicionada por la cantidad de datos de potencia nulos, que acabarían "contaminando" la métrica. La expresión correspondiente es la que sigue:

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{y_i} \tag{4.6}$$

Para completar el conocimiento sobre la calidad de las predicciones, usaremos gráficas que complementen aquellos aspectos que las métricas no terminen de definir. Concretamente, debemos poder apreciar tendencias en los resultados, que explicarían cómo las redes predecirían mejor en cierto tipo de condiciones, es decir, para cierto rango de magnitudes de entrada; algo que las métricas no podrían detectar por su carácter "medio". Estas tendencias podrían incluir estacionalidades de los datos, es decir, meses determinados en los que las predicciones sean más precisas; o según el régimen de funcionamiento, es decir, mejores o peores resultados según si la máquina está produciendo mucho, algo o poco. Por todo ello, observaremos una gráfica con la predicción del año completo, para visualizar a grandes rasgos la validez y una potencial estacionalidad en la predicción, y posteriormente veremos esta misma predicción centrada en un espacio de 12 horas, para apreciarla mejor. Asimismo, estudiaremos después tres gráficas, cada una correspondiente a periodos de producción alta, media y baja, donde podremos intuir, si las hubiera, las tendencias antes comentadas.

Una vez explicado el procedimiento a seguir, pasamos a revisar los resultados de cada uno de los modelos.

4.1 Modelo A

Tal y como explicamos previamente, es el modelo más sencillo y, por tanto, el que peores resultados presentará. Comencemos viendo las características del entrenamiento de la red.

4.1.1 Resultados del entrenamiento

Tabla 4.1 Parámetros del entrenamiento de la red neuronal A.

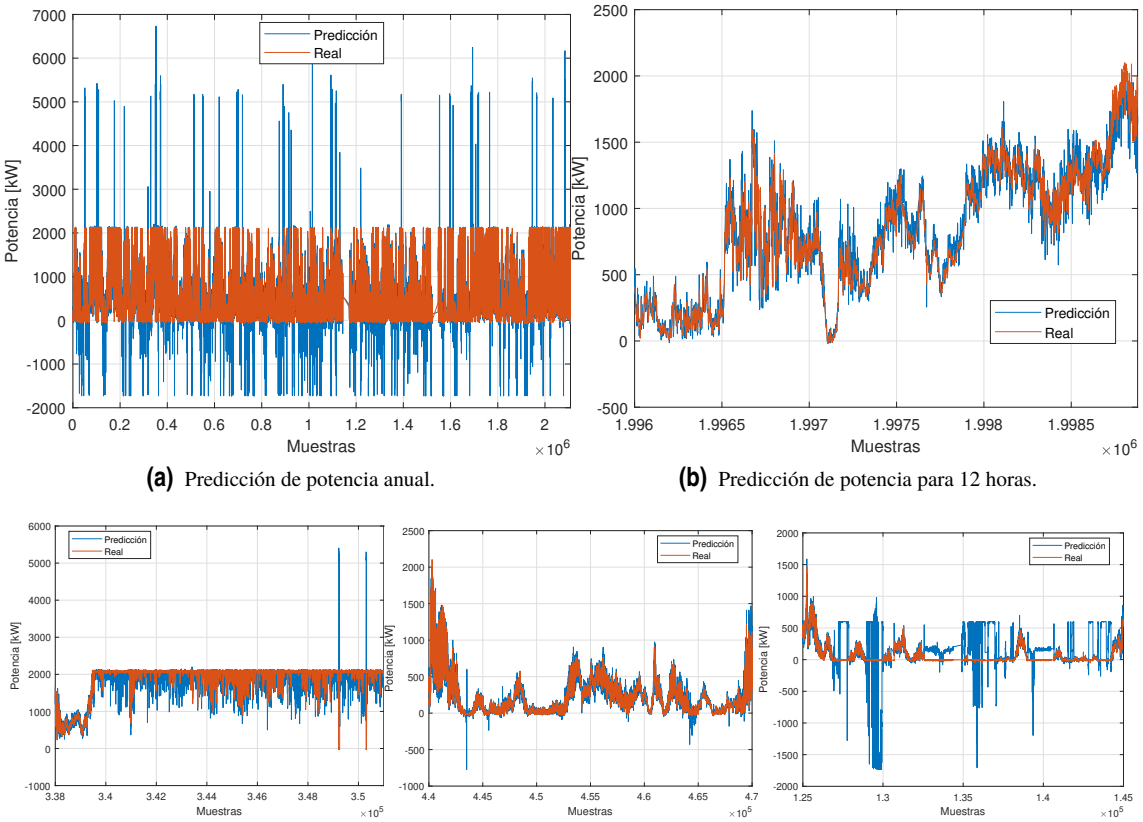
Parámetros	Valor
Iteraciones	727
Tiempo	7:30
Gradiente	$1.21 \cdot 10^3$

El tiempo requerido ha sido alto para la optimización del algoritmo, en perspectiva con la sencillez de la red. No obstante, el hecho de ser una red tan simple implica que es muy fácil poder mejorar la predicción, con lo que se entiende la necesidad de realizar tantas iteraciones.

Pasemos a analizar la predicción realizada.

4.1.2 Predicción

En primer lugar, analicemos las gráficas disponibles en la Figura 4.1.



(a) Predicción de potencia anual. **(b)** Predicción de potencia para 12 horas.

(c) Predicción potencia para mucho viento. **(d)** Predicción potencia para viento medio. **(e)** Predicción potencia para poco viento.

Figura 4.1 Predicción usando el modelo A.

Vemos que los resultados son, en general, malos. A lo largo de todo el año, en la Figura 4.1a, la red es incapaz de predecir con cierta constancia correctamente, aunque existen tramos con cierta precisión, como puede apreciarse en la Figura 4.1b. El problema parece especialmente focalizado en la predicción de valores muy bajos de producción, visible en la Figura 4.1e, donde la red comienza a producir valores extremadamente grandes. Es muy probable que esto se deba a que, con solo datos del aerogenerador cuando está en producción, no es capaz de predecir los valores de potencia en los momentos en los que la máquina no está funcionando. Para otros rangos en los que sí hay producción, como los de la Figura 4.1c y Figura 4.1d, parece que las predicciones no son del todo inadecuadas, pero siguen siendo ampliamente mejorables.

Veamos en qué se traducen estos errores en las métricas de la Tabla 4.2.

Tabla 4.2 Métricas de validación de la red neuronal A.

Métrica	**Valor**
RMSE	$1.3634 \cdot 10^3$
MAE	409.6045

Existe una fuerte discrepancia en el orden de magnitud de las dos mediciones, debido fundamentalmente a los valores atípicos que tanto penalizan en el RMSE, tal y como preveíamos. No obstante, ambos dan una idea de la desviación existente en la mayoría de predicciones vistas como conjunto.

Una mejora que corregiría buena parte de los errores sería acotar la salida de la red neuronal a valores esperables, entre 0 y 2500. Aunque no modificara los valores erróneos, al menos se mitigaría su efecto en la predicción. De todos modos, los valores cuya predicción es mala son fácilmente intuibles por ser del todo anómalos, con lo que la mejor de las opciones sería intentar otro tipo de configuración de red. En cualquier

caso, parece probable que introduciendo los valores del estado de la máquina como parámetro de entrada, tal y como se hace en el Modelo B, buena parte de los errores desaparecerían. Veámoslo a continuación.

4.2 Modelo B

Recordemos que este modelo se trata de una pequeña mejora respecto del anterior, con la inclusión de los datos del estado de la máquina.

4.2.1 Resultados del entrenamiento

En lo que a entrenamiento se refiere, ha sido mucho menos costoso llevarlo a cabo, tal y como indican el tiempo y el número de iteraciones de la Tabla 4.3.

Tabla 4.3 Parámetros del entrenamiento de la red neuronal B.

Parámetros	Valor
Iteraciones	397
Tiempo	5:43
Gradiente	$1.25 \cdot 10^3$

Analicemos las predicciones obtenidas para juzgar la validez del modelo.

4.2.2 Predicción

En la Figura 4.2 se muestran las diversas gráficas disponibles. Podemos ver a simple vista en la Figura 4.2a como el modelo B mejora claramente al modelo A. Los valores anómalos son menores en cantidad y también menores en importancia. Es más, el modelo actual parece corregir en buena medida el principal fallo del anterior: la predicción en los valores de potencia de poco orden de magnitud. Tal y como se ve en la Figura 4.2e, pese a algunos errores llamativos, la predicción es aceptable en la mayor parte de las muestras. En el resto de condiciones no se aprecian grandes diferencias respecto del modelo A, tanto la Figura 4.2c como la Figura 4.2d presentan la misma dinámica q de aciertos y errores que antes.

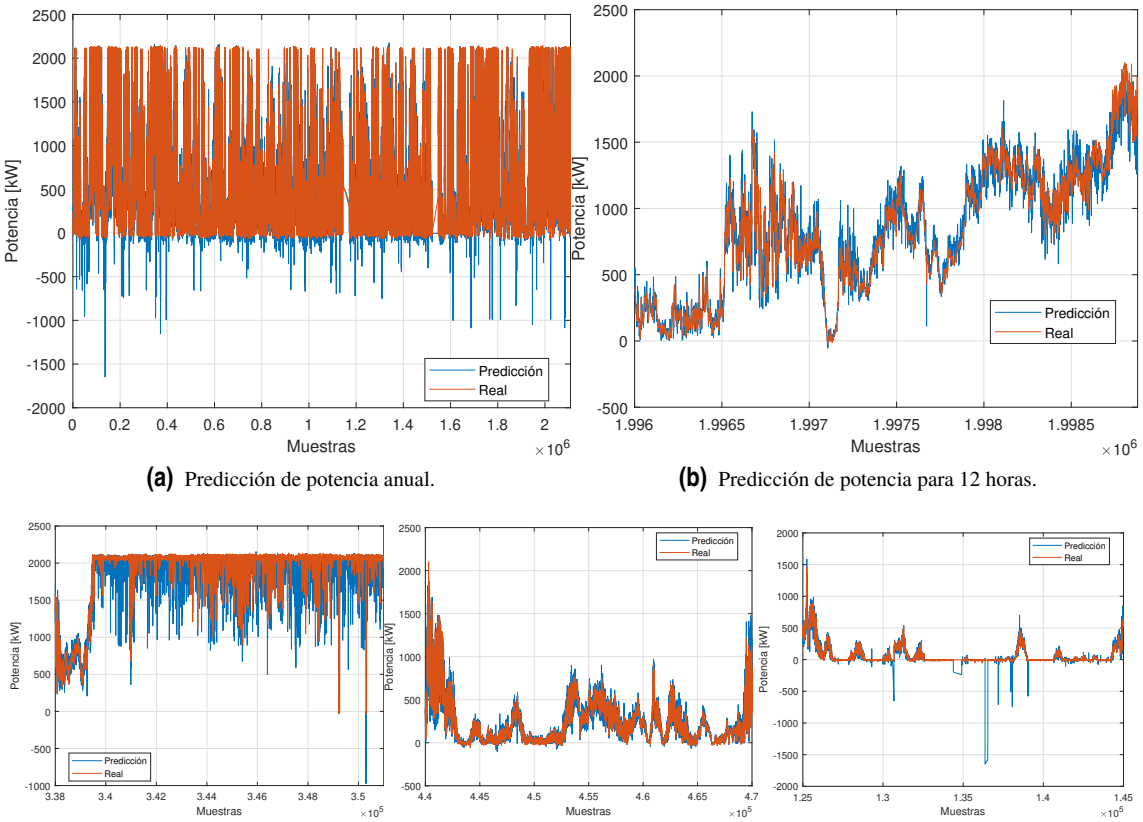
Veamos ahora si las métricas reflejan dicha mejoría, según recoge la Tabla 4.4.

Tabla 4.4 Métricas de validación de la red neuronal B.

Métrica	Valor
RMSE	110.6110
MAE	62.3169

Los resultados son enormemente mejores que los que teníamos previamente: el RMSE ya no tiene el valor desproporcionado debido a los valores anómalos y el MAE muestra un error medio de solo 62kW, poco en comparación a la potencia que produce nominalmente la máquina. En realidad, las métricas muestran unos resultados demasiado buenos para lo que puede apreciarse en las gráficas. En particular, llama la atención que el RMSE sea de 110kW cuando, en la Figura 4.2a, se observan muchos valores anómalos que deberían penalizar más en esta métrica. Es posible que estos fallos estén concentrados en datos que compartan características y que, al hacer la media con el resto de predicciones, que son muy precisas, se obtenga ese resultado.

En cualquier caso, no son resultados para darnos por satisfechos: las predicciones no son regularmente fiables y se aprecia en la mayoría un grado de error suficiente como para no tomar la red como un buen indicador. Por eso, veamos si el modelo siguiente es capaz de resolver los fallos que restan para considerar aceptable a una de las redes.



(a) Predicción de potencia anual.

(b) Predicción de potencia para 12 horas.

(c) Predicción potencia para mucho viento. **(d)** Predicción potencia para viento medio. **(e)** Predicción potencia para poco viento.

Figura 4.2 Predicción usando el modelo B.

4.3 Modelo C

De este modelo se esperan resultados aceptables, al suponer una mejora de los anteriores por tener una segunda capa oculta. Veamos, para empezar, como ha resultado el entrenamiento de la red.

4.3.1 Resultados del entrenamiento

Tal y como indica la Tabla 4.5, este modelo ha requerido menos tiempo aún para ser entrenado, así como menos iteraciones. En principio, esto no tiene por qué indicar muchas características de cómo serán las predicciones, pero sorprende (en un sentido positivo) que una arquitectura ligeramente más compleja que las anteriores sea menos costosa de entrenar. Puede deberse, en parte, a que el grado de abstracción mayor presente en este modelo facilite la configuración de los pesos y sesgos a su estado óptimo, mientras que para una sola capa existían muchas mejoras posibles a los valores de la red, debido a su sencillez.

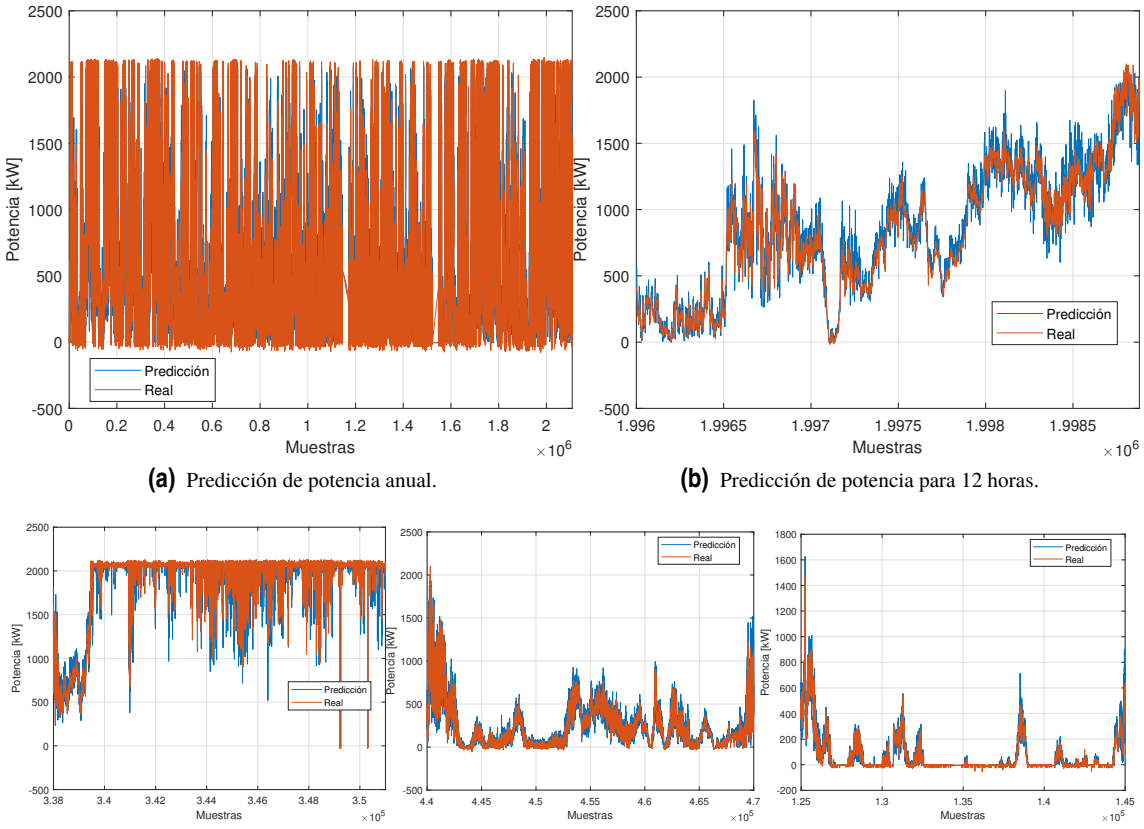
Tabla 4.5 Parámetros del entrenamiento de la red neuronal C.

Parámetros	Valor
Iteraciones	300
Tiempo	4:40
Gradiente	84.8

4.3.2 Predicción

Observemos los resultados de la predicción, presentes en la Figura 4.3 para extraer conclusiones.

Se generan mejores impresiones con este modelo que con los dos anteriores, solo con ver la Figura 4.3a. Los errores, no sabemos cuánto de frecuentes, ya no tienen magnitud suficiente como para resaltar, más allá de los valores extremos ordinarios. En la Figura 4.3b, apreciamos como existe cierta constancia en el error cometido, siendo la red capaz de mantener homogeneidad en todas las predicciones. En cuanto a las tendencias, vemos como se han corregido casi en su totalidad los errores en condiciones de producción bajas, como presenta la Figura 4.3e, así como un refinamiento de la predicción en las otras condiciones, donde en las Figura 4.3c y Figura 4.3d las líneas azules no son tan destacables, con lo que no existe tanta diferencia entre la salida de la red y la realidad.



(a) Predicción de potencia anual.

(b) Predicción de potencia para 12 horas.

(c) Predicción potencia para mucho viento. **(d)** Predicción potencia para viento medio. **(e)** Predicción potencia para poco viento.

Figura 4.3 Predicción usando el modelo C.

Confirmemos las impresiones anteriores con las métricas de regresión, que se especifican en la Tabla 4.6.

Tabla 4.6 Métricas de validación de la red neuronal C.

Métrica	Valor
RMSE	111.3938
MAE	64.2896

Resulta sorprendente comprobar que, tanto el RMSE como el MAE tiene valores muy similares al modelo B, e incluso ligeramente peores. Esto podría indicarnos que, como hipotetizábamos anteriormente, en general, las predicciones del B son muy precisas pero existen ciertos datos con mucho error, muy llamativos, que compensan los errores.

De todos modos, parece evidente la mejora de este modelo a todos los anteriores. De hecho, podríamos considerar a este modelo como válido para las predicciones, por la magnitud del error cometido y por la constancia de los errores ante diferentes condiciones de predicción.

Veamos el modelo que resta y si es capaz de mejorar a este.

4.4 Modelo D

Finalmente, nos encontramos ante este modelo, pensado para mostrar qué resultados se obtendrían si nos decidimos a poner en juego muchos más recursos en la red neuronal, independientemente de la complejidad que aparezca, añadiendo una capa y un número significativo de neuronas adicionales.

4.4.1 Resultados del entrenamiento

Comencemos por analizar los datos del entrenamiento de esta red, presentes en la Tabla 4.7.

Tabla 4.7 Parámetros del entrenamiento de la red neuronal D.

Parámetros	Valor
Iteraciones	154
Tiempo	4:24
Gradiente	633

Llama la atención el poco número de iteraciones requeridas, en comparación al tiempo de entrenamiento. Esto nos parece indicar que, debido a la arquitectura más compleja, cada iteración en el entrenamiento de la red es mucho más costosa.

Pasemos a las predicciones.

4.4.2 Predicción

Tenemos a nuestra disposición, en la Figura 4.4, las predicciones realizadas sobre el Modelo D. Sorprende la similitud aparente entre este modelo y el anterior. En todas las gráficas, incluida en la Figura 4.4b, parecemos detectar los mismos valores anómalos. La constancia en la predicción en todo tipo de condiciones se mantiene al igual que el Modelo C, como muestran la Figura 4.4c, la Figura 4.4d y la Figura 4.4e.

Comprobemos qué resultados obtienen las métricas en la Tabla 4.8.

Tabla 4.8 Métricas de validación de la red neuronal D.

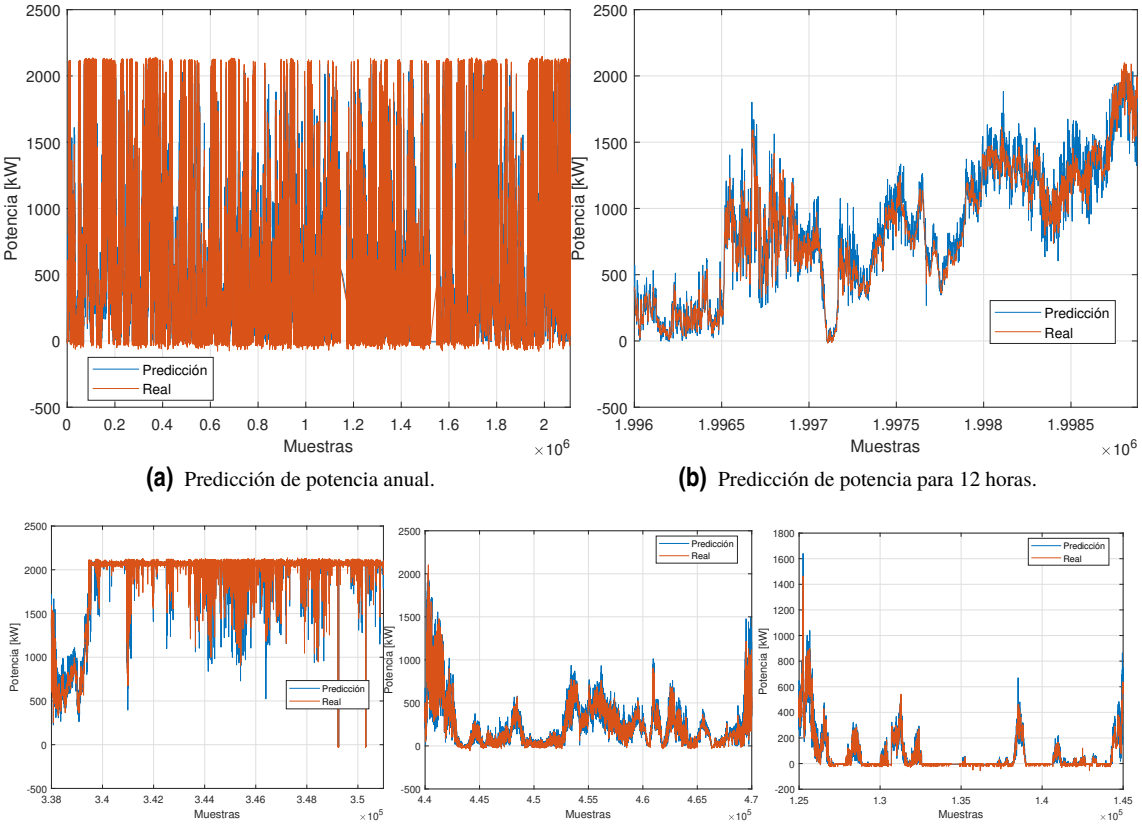
Métrica	Valor
RMSE	129.8829
MAE	64.5663

Sin embargo, pese a su apariencia similar, las métricas demuestran un ligero empeoramiento de las predicciones de este modelo con respecto a los anteriores.

4.5 Discusión

Para dar por finalizada la validación de los modelos, parece coherente compararlos y extraer conclusiones acerca de ellos y de la dinámica de nuestro problema.

Podemos comenzar identificando los parámetros que consideramos “estrictamente necesarios” a la hora de desarrollar la red. Se ha visto como, una vez introducido el estado de la máquina, gran parte de los errores



(a) Predicción de potencia anual.

(b) Predicción de potencia para 12 horas.

(c) Predicción potencia para mucho viento. **(d)** Predicción potencia para viento medio. **(e)** Predicción potencia para poco viento.

Figura 4.4 Predicción usando el modelo D.

que se cometían desaparecen, con lo que parece evidente que es una variable necesaria para el entrenamiento de la red. Por otro lado, los valores de la nacelle no parecen haber supuesto mejoras considerables. El resto de parámetros, pese a no haber comprobado su necesidad simulando una red en la que no aparecieran, parecen indispensables por ser responsables directos en la generación de potencia. De este modo, consideramos como magnitudes indispensables para el desarrollo de estas NN las que aparecen listadas en la Tabla 4.9.

Tabla 4.9 Magnitudes indispensables.

Dirección del viento
Temperatura ambiental
Velocidad del viento
Pitch
Estado

A continuación, trataremos de comparar los modelos a través de las distintas métricas, incluyendo una valoración subjetiva de la complejidad de la red en una escala de 1 a 5, en función de sus aspectos formales como son el número de capas y de neuronas. Estos datos se encuentran resumidos en la Tabla 4.10.

Como podemos observar y tal como adelantábamos, el primero de los modelos es del todo desechable, al igual que el resto de ellos presentaban mejorías cada uno respecto del anterior. Aunque a priori pudiéramos inferir que, a mayor complejidad de la red, mejores resultados, el último de los modelos descarta por completo esta idea. Las dinámicas y cálculos que hay detrás de una red neuronal son extremadamente complejos, de modo que no puede conocerse de antemano el rendimiento de la red simplemente por la arquitectura escogida.

Parece justo proponer al modelo C como el mejor de los desarrollados. El error, aunque presente, es homogéneo y de una magnitud asumible; a diferencia del B, que pese a contar con mejores métricas, los resultados de potencia negativos impiden considerarlo un buen estimador.

Tabla 4.10 Comparación entre modelos.

Modelo	Tiempo	RMSE	MAE	Complejidad
A	7:30	$1.3634 \cdot 10^3$	409.6045	1
B	5:43	110.6110	62.3169	1.5
C	4:40	111.3938	64.2896	2.5
D	4:28	129.8829	64.5663	4

Independientemente de lo anterior, siempre podríamos escoger otro de los modelos con buenos resultados, el B o el D, en situaciones concretas que requirieran mayor precisión. Por ejemplo, si nuestro objetivo principal es predecir con exactitud la potencia del aerogenerador en los instantes de poca producción, podríamos hacer uso del modelo D en esos instantes, mucho más preciso, y alternar al modelo de referencia en los momentos que no se cumpliera con esta condición.

Podría considerarse que las métricas escogidas no suponen una buena medida de la bondad de los resultados. Es cierto que pueden llegar a confusión, pero analizar los resultados en profundidad requeriría de una dedicación completa a esta temática. Se ha escogido analizar los resultados de todo el año, en conjunto, pues han sido los utilizados a lo largo de todo el trabajo. Podría haberse optado por hacer estas métricas a zonas concretas del año, como las que se han escogido para las gráficas, pero suponer como representativo el resultado obtenido en 20.000 muestras, en un conjunto de más de 2.000.000, no parecía una solución adecuada.

Otra de las posibilidades habría sido eliminar aquellas zonas en las que la interpolación primera de los datos tiene un peso significativo. De nuevo, consideramos que los datos interpolados, al ser los usados en todo el documento, son los "reales" en nuestro caso.

Tanto por los datos disponibles, que carecían de consistencia temporal al estar desincronizados, como por el objetivo del trabajo, que era el desarrollo y la validación de un modelo, y no la optimización del mismo, la filosofía a lo largo del documento no ha perseguido obtener los mejores resultados posibles. Si ese hubiera sido el objetivo, el primer y más importante de los pasos habría sido tratar los datos con mucha mayor profundidad, hasta quedar satisfechos con el registro listo para entrenar y validar.

En cualquiera de los casos, existe una solución indicada para cada tipo de exigencias y requerimientos, con lo que el desarrollo de los modelos se da por adecuado.

4.6 Otros posibles modelos

Los modelos validados y previamente desarrollados son solo unas pocas posibilidades de todas las que ofrecen las NN. Existen incontables modificaciones que podrían haber supuesto resultados muy diferentes en todos los modelos: desde la elección de los datos, pasando por el número total de neuronas o el propio algoritmo de entrenamiento. Eso, sin incluir otros posibles modelos que, manteniendo la configuración y filosofía de los desarrollados, hubiesen cambiado en la arquitectura de la red ligeramente o modificado el conjunto de datos de entrenamiento.

De hecho, los resultados obtenidos son, con casi toda seguridad, mejorables. Pese a que el modelo D, que era el que más capacidad de abstracción tenía, no haya dado buenos resultados, esto no es motivo suficiente como para descartar la idea de añadir una tercera capa oculta. Variar la configuración de esta capa añadida, las neuronas de cada una de las capas o simplemente permutarlas podría suponer una mejora considerable. En este tipo de técnica, el ensayo y error es clave para encontrar la solución óptima.

Como decimos, las posibilidades son prácticamente infinitas y, aunque los resultados obtenidos anteriormente son suficientes para mostrar la capacidad de predicción de una red neuronal, podemos enunciar otras posibilidades que podrían llevarse a cabo desde nuestra posición.

Datos de partida

Existe la posibilidad de, por un lado, usar los datos no sincronizados; y, por otro, tratar aún más los datos para usarlos en el entrenamiento con limitaciones en los valores extremos de la potencia o incluso con medias diezminutales. En nuestro caso, la opción de escoger los datos lo menos modificados posible nos pareció la mejor solución, pero puede no ser la única. Parte de técnica en el uso de NN es tener la habilidad de modificar los datos que se le introducen a la red, de forma que jueguen a favor de la predicción posterior.

Datos de entrenamiento

Escogimos un cuarto de los datos totales para el entrenamiento, pero el tamaño del conjunto podría haber sido otro. No es esperable un gran cambio ante tamaños ligeramente diferentes, teniendo en cuenta la imposibilidad de escoger muchos datos para que, el testar la predicción de la red, la validación no pierda su significado; pero es una posibilidad existente.

Número de neuronas

Desde el comienzo, se optó por imponer en 20 el número de neuronas, pero podría ser un número diferente: 24 o incluso hasta 50, todo dependerá de la complejidad que estemos dispuestos a asumir.

Algoritmo de entrenamiento

Tal y como se explicó, escogimos el algoritmo de Levenberg-Marquardt por reducir el tiempo de entrenamiento, pero podrían probarse otros más costosos en computadoras capaces de sobrellevarlo.

Otras potencias

Una opción para comprobar la robustez de las diferentes arquitecturas es aplicar las mismas NN para predecir otro tipo de potencias, como por ejemplo la reactiva. Entrenarlas con los datos correspondientes y comprobar si la validez de los datos es igual de buena o mala, según el modelo, que para el caso de la potencia activa.

Como indicamos, lo anterior es solo un pequeño resumen de todas las posibles opciones disponibles para desarrollar NN distintas para nuestro problema. Es la ventaja de esta tecnología: si el resultado no se enmarca dentro de lo aceptable, es cuestión de probar diferentes opciones hasta encontrar la adecuada.

5 Conclusiones y trabajos futuros

The scientific man does not aim at an immediate result. He does not expect that his advanced ideas will be readily taken up. His work is like that of the planter — for the future. His duty is to lay the foundation for those who are to come, and point the way. He lives and labors and hopes.

Nikola Tesla, 1934

Se han desarrollado varios modelos que predicen la potencia generada por una turbina eólica, todos ellos con distintos grados de precisión y complejidad. Si atendemos a las métricas y las figuras, nos encontramos con varios resultados similares en los tres últimos modelos; no obstante, debemos decantarnos por el Modelo C, no solo por la magnitud de los errores cometidos sino también por la homogeneidad de estos. El conjunto de modelos han servido para ilustrar el camino hacia un modelo cada vez más complejo, a base de introducir cambios sucesivos que alteraban las redes previas.

De lo anterior, podemos deducir que este trabajo no solo ha permitido obtener un modelo capaz de predecir la potencia con un buen rendimiento, sino que ha compuesto una guía de qué pasos realizar a la hora de construir una red neuronal y cuáles son las modificaciones y añadidos necesarios para mejorar paulatinamente los resultados.

Esto plantea la discusión de qué resultados se habrían obtenido con otro tipo de metodología diferente de la inteligencia artificial. Sabiendo que el objetivo último de un modelo es aproximarse lo máximo posible a la realidad, podemos concluir que las redes neuronales permiten, no solo resolver este cometido, sino desgranar el problema en varias cuestiones menores que sean mucho más realizables. Esto es, la ventaja de usar las redes neuronales es que pueden focalizarse tanto como se deseen, bien concentradas en problemas pequeños, como pudiera ser la predicción solo en días con vientos de más de una cierta velocidad, hasta el problema más genérico, como puede ser predecir la potencia para todo un año. La versatilidad de las redes y la inteligencia artificial es tal que, con la misma filosofía llevada en este trabajo, podría tratarse de averiguar métricas tan dispares (aunque relacionadas con la temática) como la probabilidad en cada mes de tener sin funcionamiento un aerogenerador del parque. Siendo así, nos parece que la decisión de haber usado las redes neuronales como herramienta para modelar el sistema ha sido útil e interesante, no solo por los resultados obtenidos, sino por la capacidad de mejora y extensión de esta solución a muchos otros problemas relacionados con la materia.

Relacionado con la capacidad de mejora, resulta evidente la posibilidad de hacer aún más precisos los modelos anteriormente presentados. Todo es cuestión de aumentar los grados de complejidad, el número de neuronas y de capas o el número de datos disponibles para entrenar, entre otras, y será cuestión de ensayo y error hasta dar con un modelo aún más preciso. Para el alcance de este trabajo, consideramos que los presentados cumplen con su objetivo, pero el margen de mejora existe y es realizable.

Una posible continuación de este trabajo es la mejora del gemelo digital tal y como se conceptualiza. Es decir, desarrollar una interfaz gráfica y ser capaces de interconectar los gemelos digital y real, para poder operar con información a tiempo real en ambos en todo momento.

Sin embargo, lo anterior requeriría de conocimientos en software gráficos, así como un desarrollo paralelo de redes neuronales para la predicción del resto de magnitudes presentes en los datos. Es por ello por lo que quizás resultara más inmediato otro tipo de trabajo futuro, consistente precisamente en desarrollar modelos

para predecir otro tipo de eventos. En concreto, ya que se ha cubierto el objetivo principal de un aerogenerador que es la producción de potencia, parece que la siguiente medición más deseable sería aquella que permitiera la detección de fallos. Dada la importancia de predecir la capacidad de generación de una máquina, no sería suficiente con saber cuál es dicha generación en condiciones de funcionamiento, sino en qué ocasiones puede encontrarse la máquina lista para funcionar. En este sentido, prever los fallos asociados a componentes propios o externos, se torna fundamental. Sería posible, conociendo cuáles son las principales causas de fallos, tomar las mediciones adecuadas relacionadas con esas causas y programar una red neuronal que fuera capaz de predecir aquello a lo que matemáticamente no le podemos encontrar relación directa. Por ejemplo: si conocemos que a partir de cierta temperatura de un líquido interno de la máquina, la producción se detiene, podríamos medir las condiciones de los elementos mecánicos en los que interviene dicho líquido, de donde podría obtenerse un modelo de la temperatura a partir de los elementos.

Lo anterior son solo unas propuestas de tantas que pueden imaginarse gracias a las redes neuronales. Las posibilidades de esta tecnologías son muchas y en el contexto actual parece más necesaria que nunca. Prever, predecir y actuar con anterioridad a que ocurran las situaciones se antoja fundamental en una época en la que el tiempo es la mayor de las inversiones y en la que la digitalización nos brinda infinitas posibilidades para el flujo constante de información actualizada.

Apéndice A
Códigos

A.1 Rellenado de datos

```
% Borrado previo
clear all
clc
% Carga de los datos
load('DATOS.mat')
% Inicializar datos rellenados
RELLENADOS = DATOS;
TF = zeros(size(DATOS));

%% Mantenemos los valores en las posiciones primeras y últimas sin dato
% Determinamos posiciones valores no nulos (primera y última)
indices = zeros(2,12);
matriz = table2array(DATOS(:,(2:end)));
for i = 1:12
    indices(1,i) = find(not(isnan(matriz(:,i))),1,'first');
    indices(2,i) = find(not(isnan(matriz(:,i))),1,'last');
    % Mantenemos el valor más cercano en los primeros y últimos valores nulos
    [RELLENADOS(1:indices(1,i),i+1),TF(1:indices(1,i),i+1)] = fillmissing(
        RELLENADOS(indices(1,i),i+1),'next');
    [RELLENADOS(indices(2,i):end,i+1),TF(indices(2,i):end,i+1)] = fillmissing(
        RELLENADOS(indices(2,i):end,i+1),'previous');
end

%% Interpolamos el resto de valores intermedios

% Interpolación lineal
[RELLENADOS(:,[2:4,6:11,13]),TF(:,[2:4,6:11,13])] = fillmissing(RELLENADOS
    (:,[2:4,6:11,13]),'linear');

% Mantenedor para STATUS y PITCH
[RELLENADOS(:,[5,12]),TF(:,[5,12])] = fillmissing(RELLENADOS(:,[5,12]),'
    previous');

%% Tener datos en formato timetable
DATOS_T = table2timetable(DATOS);
RELLENADOS_T = table2timetable(RELLENADOS);
```

A.2 Sincronización de datos

```
%% 1º parte año

% Carga de datos semestrales
load enejunTABLE.mat %enejunTABLE
% Crear tabla datos sincronizados
sz = [1059841;13];
varTypes = {'datetime','double','double','double','double','double'...
    ,'double','double','double','double','double','double','double'};
varNames = {'Time','WDIR','TEMP','WSPE','STATUS','APWR','STPWR',...
    'RPWR','RTPWR','GSPE','NACELLE','PITCH','RSPE'};
bien_enejun = table('Size',sz,'VariableTypes',varTypes,'VariableNames',varNames
    );
%Inicializar a NaN a partir de la segunda columna
bien_enejun{:,2:end} = NaN;
% Tomamos el tiempo primero de APWR como referencia y sumamos de 15 sec
bien_enejun.Time(1) = enejunTABLE.WNAC_WDDIRACQDEG(1);
for i=2:1048321
    bien_enejun.Time(i) = bien_enejun.Time(i-1) + duration(0,0,15);
end

% Creación de tablas de tiempo por magnitudes
tiemposEJ = enejunTABLE.WNAC_WDDIRACQDEG;
tiemposEJ(:,2) = enejunTABLE.WNAC_WDSPDACQMS;
tiemposEJ(:,3) = enejunTABLE.WNAC_WDSPDACQMS;
tiemposEJ(:,4) = enejunTABLE.WTUR_TURSTACQINT;
tiemposEJ(:,5) = enejunTABLE.WGEN_WACQKW;
tiemposEJ(:,6) = enejunTABLE.WGEN_STAWACQKW;
tiemposEJ(:,7) = enejunTABLE.WGEN_VARACQKVAR;
tiemposEJ(:,8) = enejunTABLE.WGEN_RTRWACQKW;
tiemposEJ(:,9) = enejunTABLE.WGEN_SPDACQRPM;
tiemposEJ(:,10) = enejunTABLE.WYAW_YAWANGACQDEG;
tiemposEJ(:,11) = enejunTABLE.WROT_PTAGVALBLACQDEG;
tiemposEJ(:,12) = enejunTABLE.WROT_ROTSPDACQRPM;

% Sincronización
for k = 1:12
    disp(k)
    cont = 1;
    i = 1;
    while cont < find(ismissing(tiemposEJ(:,k)),1,'first')
        while (bien_enejun.Time(i) < tiemposEJ(cont,k))
                i = i+1;
        end
    bien_enejun(i,k+1) = enejunTABLE(cont,2*k);
    cont = cont+1;
    end
end

%% 2º parte año

% Carga de datos semestrales
load juldic_bien.mat
% Crear tabla datos sincronizados
```

```
sz = [1059841;13];
varTypes = {'datetime','double','double','double','double','double'...
    ,'double','double','double','double','double','double','double'};
varNames = {'Time','WDIR','TEMP','WSPE','STATUS','APWR','STPWR',...
    'RPWR','RTPWR','GSPE','NACELLE','PITCH','RSPE'};
bien_juldic = table('Size',sz,'VariableTypes',varTypes,'VariableNames',varNames
    );
%Inicializar a NaN a partir de la segunda columna
bien_juldic{:,2:end} = NaN;
% Tomamos el tiempo primero de APWR como referencia y sumamos de 15 sec
bien_juldic.Time(1) = D_APWR(1);
for i=2:1059841
    bien_juldic.Time(i) = bien_juldic.Time(i-1) + duration(0,0,15);
end

% Creación de tablas de tiempo por magnitudes
tiemposJD=juldicTABLE.WNAC_WDDIRACQDEG;
tiemposJD(:,2)=juldicTABLE.WNAC_WDSPDACQMS;
tiemposJD(:,3)=juldicTABLE.WNAC_WDSPDACQMS;
tiemposJD(:,4)=juldicTABLE.WTUR_TURSTACQINT;
tiemposJD(:,5)=juldicTABLE.WGEN_WACQKW;
tiemposJD(:,6)=juldicTABLE.WGEN_STAWACQKW;
tiemposJD(:,7)=juldicTABLE.WGEN_VARACQKVAR;
tiemposJD(:,8)=juldicTABLE.WGEN_RTRWACQKW;
tiemposJD(:,9)=juldicTABLE.WGEN_SPDACQRPM;
tiemposJD(:,10)=juldicTABLE.WYAW_YAWANGACQDEG;
tiemposJD(:,11)=juldicTABLE.WROT_PTAGVALBLACQDEG;
tiemposJD(:,12)=juldicTABLE.WROT_ROTSPDACQRPM;

% Sincronizacion
for k = 1:12
    disp(k)
    cont = 1;
    i = 1;
    while cont < find(ismissing(tiemposJD(:,k)),1,'first')
        while (bien_juldic.Time(i) < tiemposJD(cont,k))
               i = i+1;
        end
    bien_juldic(i,k+1) = juldicTABLE(cont,2*k);
    cont = cont+1;
    end
end

%% Unión de las tablas semestrales en una única

% Crear tabla
sz = [2108161;13];
varTypes = {'datetime','double','double','double','double','double'...
    ,'double','double','double','double','double','double','double'};
varNames = {'Time','WDIR','TEMP','WSPE','STATUS','APWR','STPWR',...
    'RPWR','RTPWR','GSPE','NACELLE','PITCH','RSPE'};

DATOS = table('Size',sz,'VariableTypes',varTypes,'VariableNames',varNames);
DATOS{:,1} = NaT;
DATOS{:,2:end} = NaN;

DATOS{1:1048320,1} = bien_enejun{:,1};
```

```
DATOS{1:1048320,2:end} = bien_enejun{:,2:end};
DATOS{1048321:end,1} = bien_juldic{:,1};
DATOS{1048321:end,2:end} = bien_juldic{:,2:end};
```

A.3 Modelo A

```
%% Conjunto entrenamiento
clear all, clc
% Carga datos interpolados
load TIMETABLE.mat
% Transformamos tabla en array eliminando la columna de DATETIME
datos = table2array(RELLENADOS_T);
% Tamaño del conjunto de datos
tam = size(datos);
% Escogemos el tamaño del conjunto de datos que usar:
tam_rn = round(tam(1)/4); % Un cuarto
desp = 5e5; % Para no usar los N primeros datos
% Array con las magnitudes de entrada INPUT (input = X)
X = datos(desp:(desp+tam_rn),[1:3,11]);
% Array con las magnitudes de salida OUTPUT (output = Y)
Y = datos(desp:(desp+tam_rn),5); % Potencias
% Array de verificación: maáquina funcionando (RUNNING)
V = datos(desp:(desp+tam_rn),4);
% Buscamos los datos en los que el estado de la máquina sea RUNNING
not_running = find(V ~= 100);
% Tamaño del conjunto de datos retirado
tam_not_running = size(not_running);
% Eliminamos datos cuyo estado de la máquina sea distinto del funcionamiento
X(not_running,:) = [];
Y(not_running,:) = [];
% Tamaño del conjunto de datos actualizado
tam_running = size(X);
% Definir seed para obtener siempre mismo resultado
RandStream.setGlobalStream(RandStream('mt19937ar','seed',1));

%% Ejecución de la guía de uso del toolbox de fitnet de MATLAB

%% Conjunto datos validación
% Carga datos interpolados
load TIMETABLE.mat
% Transformamos tabla en array eliminando la columna de DATETIME
datos = table2array(RELLENADOS_T);
% Tamaño del conjunto de datos
tam = size(datos);
% Definimos entradas y salidas
% Array con las magnitudes de entrada INPUT (input = X)
XTest = datos(:,[1:3,11]);
% Array con las magnitudes de salida OUTPUT (output = Y)
YTest = datos(:,5); % Potencias

%% Predicción y resultados
% Ver red neuronal
view(net)
% Predicciones
```

```
testPredictions = net(XTest);
% Cálculo del error
errorPredictions = YTest - testPredictions;
% Performance
perf = perform(net,YTest,testPredictions)
% Métricas para la evaluación del error
testMSE = mse(errorPredictions)
testRMSE = sqrt(testMSE)
testMAE = mean(abs(errorPredictions))
testSD = std(errorPredictions)
% Para el MAPE discriminar los primeros n términos cercanos a cero
idx = abs(YTest)>100;
testMAPE = mean(abs(errorPredictions(idx)./YTest(idx)))*100
```

A.4 Modelo B

```
%% Conjunto entrenamiento
clear all, clc
% Carga datos interpolados
load TIMETABLE.mat
% Transformamos tabla en array eliminando la columna de DATETIME
datos = table2array(RELLENADOS_T);
% Tamaño del conjunto de datos
tam = size(datos);
% Escogemos el tamaño del conjunto de datos que usar:
tam_rn = round(tam(1)/4); % Un cuarto
desp = 5e5; % Para no usar los N primeros datos
% Array con las magnitudes de entrada INPUT (input = X)
X = datos(desp:(desp+tam_rn),[1:4,11]);
% Array con las magnitudes de salida OUTPUT (output = Y)
Y = datos(desp:(desp+tam_rn),5); % Potencias
% Definir seed para obtener siempre mismo resultado
RandStream.setGlobalStream(RandStream('mt19937ar','seed',1));

%% Ejecución de la guía de uso del toolbox de fitnet de MATLAB

%% Conjunto datos validación
% Carga datos interpolados
load TIMETABLE.mat
% Transformamos tabla en array eliminando la columna de DATETIME
datos = table2array(RELLENADOS_T);
% Tamaño del conjunto de datos
tam = size(datos);
% Definimos entradas y salidas
% Array con las magnitudes de entrada INPUT (input = X)
XTest = datos(:,[1:4,11]);
% Array con las magnitudes de salida OUTPUT (output = Y)
YTest = datos(:,5); % Potencias

%% Predicción y resultados
% Ver red neuronal
view(net)
% Predicciones
testPredictions = net(XTest);
```

```
% Cálculo del error
errorPredictions = YTest - testPredictions;
% Performance
perf = perform(net,YTest,testPredictions)
% Métricas para la evaluación del error
testMSE = mse(errorPredictions)
testRMSE = sqrt(testMSE)
testMAE = mean(abs(errorPredictions))
testSD = std(errorPredictions)
% Para el MAPE discriminar los primeros n términos cercanos a cero
idx = abs(YTest)>100;
testMAPE = mean(abs(errorPredictions(idx)./YTest(idx)))*100
```

A.5 Modelo C

```
%% Conjunto entrenamiento
clear all, clc
% Carga datos interpolados
load TIMETABLE.mat
% Transformamos tabla en array eliminando la columna de DATETIME
datos = table2array(RELLENADOS_T);
% Tamaño del conjunto de datos
tam = size(datos);
% Escogemos el tamaño del conjunto de datos que usar:
tam_rn = round(tam(1)/4); % Un cuarto
desp = 8000; % Para no usar los N primeros datos
% Array con las magnitudes de entrada INPUT (input = X)
X = datos(desp:(desp+tam_rn),[1:4]);
% Array con las magnitudes de salida OUTPUT (output = Y)
Y = datos(desp:(desp+tam_rn),5); % Potencias

%% Configuración de la red
x = X';
y = Y';
% Algoritmo Levenberg-Marquardt backpropagation.
trainFcn = 'trainlm';
% Tamaño de la red
hiddenLayerSize = [16,4];
% Creación de la red
net = fitnet(hiddenLayerSize,trainFcn);
% Funciones de normalización de datos
net.input.processFcns = {'mapstd'};
net.output.processFcns = {'mapstd'};
% Funciones de activación por capas
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'tansig';
% Definir seed para obtener siempre mismo resultado
RandStream.setGlobalStream(RandStream('mt19937ar','seed',1));
% División de los datos en los subconjuntos, aleatorio
net.divideFcn = 'dividerand';
net.divideParam.trainRatio = 60/100;
net.divideParam.valRatio = 35/100;
net.divideParam.testRatio = 5/100;
```

```
%% Entrenamiento de la red
% Mostrar entrenamiento
net.trainParam.show=1;
% Learning rate
net.trainParam.lr=0.05;
% Máximas iteraciones
net.trainParam.epochs=10000;
% Objetivo entrenamiento
net.trainParam.goal=0.05^2;
% Si quisieramos cambiar parámetro a optimizar (default: mse)
% net.performFcn = 'mae';
% Entrenamiento
[net,tr] = train(net,x,y);


%% Conjunto datos validación
% Carga datos interpolados
load TIMETABLE.mat
% Transformamos tabla en array eliminando la columna de DATETIME
datos = table2array(RELLENADOS_T);
% Tamaño del conjunto de datos
tam = size(datos);
% Definimos entradas y salidas
% Array con las magnitudes de entrada INPUT (input = X)
XTest = datos(:,[1:4])';
% Array con las magnitudes de salida OUTPUT (output = Y)
YTest = datos(:,5)'; % Potencias

%% Predicción y resultados
view(net)
% Predicciones
testPredictions = net(XTest);
% Cálculo del error
errorPredictions = YTest - testPredictions;
% Performance
perf = perform(net,YTest,testPredictions)
% Métricas para la evaluación del error
testMSE = mse(errorPredictions)
testRMSE = sqrt(testMSE)
testMAE = mean(abs(errorPredictions))
testSD = std(errorPredictions)
% Para el MAPE discriminar los primeros n términos cercanos a cero
idx = abs(YTest)>100;
testMAPE = mean(abs(errorPredictions(idx)./YTest(idx)))*100
```

A.6 Modelo D

```
%% Conjunto entrenamiento
clear all, clc
% Carga datos interpolados
load TIMETABLE.mat
% Transformamos tabla en array eliminando la columna de DATETIME
datos = table2array(RELLENADOS_T);
% Tamaño del conjunto de datos
```

```
tam = size(datos);
% Escogemos el tamaño del conjunto de datos que usar:
tam_rn = round(tam(1)/4); % Un cuarto
desp = 8000; % Para no usar los N primeros datos
% Array con las magnitudes de entrada INPUT (input = X)
X = datos(desp:(desp+tam_rn),[1:4,10:11]);
% Array con las magnitudes de salida OUTPUT (output = Y)
Y = datos(desp:(desp+tam_rn),5); % Potencias

%% Configuración de la red
x = X';
y = Y';
% Algoritmo Levenberg-Marquardt backpropagation.
trainFcn = 'trainlm';
% Tamaño de la red
hiddenLayerSize = [16,8,4];
% Creación de la red
net = fitnet(hiddenLayerSize,trainFcn);
% Funciones de normalización de datos
net.input.processFcns = {'mapstd'};
net.output.processFcns = {'mapstd'};
% Funciones de activación por capas
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'tansig';
% Definir seed para obtener siempre mismo resultado
RandStream.setGlobalStream(RandStream('mt19937ar','seed',1));
% División de los datos en los subconjuntos, aleatorio
net.divideFcn = 'dividerand';
net.divideParam.trainRatio = 60/100;
net.divideParam.valRatio = 35/100;
net.divideParam.testRatio = 5/100;

%% Entrenamiento de la red
% Mostrar entrenamiento
net.trainParam.show=1;
% Learning rate
net.trainParam.lr=0.05;
% Máximas iteraciones
net.trainParam.epochs=10000;
% Objetivo entrenamiento
net.trainParam.goal=0.05^2;
% Si quisieramos cambiar parámetro a optimizar
% net.performFcn = 'mse';
% Entrenamiento
[net,tr] = train(net,x,y);


%% Conjunto datos validación
% Carga datos interpolados
load TIMETABLE.mat
% Transformamos tabla en array eliminando la columna de DATETIME
datos = table2array(RELLENADOS_T);
% Tamaño del conjunto de datos
tam = size(datos);
% Definimos entradas y salidas
% Array con las magnitudes de entrada INPUT (input = X)
XTest = datos(:,[1:4,10:11])';
```

```
% Array con las magnitudes de salida OUTPUT (output = Y)
YTest = datos(:,5)'; % Potencias

%% Predicción y resultados
% Ver red neuronal
view(net)
% Predicciones
testPredictions = net(XTest);
% Cálculo del error
errorPredictions = YTest - testPredictions;
% Performance
perf = perform(net,YTest,testPredictions)
% Métricas para la evaluación del error
testMSE = mse(errorPredictions)
testRMSE = sqrt(testMSE)
testMAE = mean(abs(errorPredictions))
testSD = std(errorPredictions)
% Para el MAPE discriminar los primeros n términos cercanos a cero
idx = abs(YTest)>100;
testMAPE = mean(abs(errorPredictions(idx)./YTest(idx)))*100
```

Bibliografía

[1] A. Nordmann, "Esquema de una turbina eólica," 2007. [Online]. Available: https://commons.wikimedia.org/wiki/File:Wind_turbine_int.svg

[2] Y. Lei, A. Mullane, G. Lightbody, and R. Yacamini, "Modeling of the wind turbine with a doubly fed induction generator for grid integration studies," *Energy Conversion, IEEE Transactions on*, vol. 21, pp. 257 – 264, 04 2006.

[3] G. 88, "Gottlieb-daimler-schule sindelfingen. der fotograf bedankt sich für das freundliche angebot der schulleitung für die aufnahmen vom schuldach aus." 2019. [Online]. Available: https://commons.wikimedia.org/wiki/File:Gottlieb-Daimler-Schule_Sindelfingen_70.jpg

[4] H. Hillewaert, "Newly constructed windmills d4 (nearest) to d1 on the thornton bank, 28 km off shore, on the belgian part of the north sea." 2008. [Online]. Available: https://commons.wikimedia.org/wiki/File:Windmills_D1-D4_(Thornton_Bank).jpg

[5] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the Digital Twin: A systematic literature review," *CIRP Journal of Manufacturing Science and Technology*, vol. 29, pp. 36–52, may 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1755581720300110?pes=vor

[6] Avimanyu786, "How deep learning is a subset of machine learning and how machine learning is a subset of artificial intelligence (ai)," 2019. [Online]. Available: https://commons.wikimedia.org/wiki/File:AI-ML-DL.png

[7] "Esquema de una neurona de mcculloch-pitts," 2006. [Online]. Available: https://commons.wikimedia.org/wiki/File:Mccullochpitts.png

[8] Mcstrother, "Esquema de una red neuronal artificial de dos capas," 2010. [Online]. Available: https://commons.wikimedia.org/wiki/File:Two_layer_ann.svg

[9] A. Tummala, R. K. Velamati, D. K. Sinha, V. Indraja, and V. H. Krishna, "A review on small scale wind turbines," *Renewable and Sustainable Energy Reviews*, vol. 56, pp. 1351–1371, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032115014100

[10] R. E. España, "Potencia instalada." [Online]. Available: https://www.ree.es/es/datos/generacion/potencia-instalada

[11] D. intelligence for collaborative ENergy management in Manufacturing, "Concept." [Online]. Available: https://denim-fof.eu/concept/

[12] L. Söder and T. Ackermann, *Wind Power in Power Systems: An Introduction*. John Wiley & Sons Ltd, 2005, p. 55.

[13] T. Ackermann, *Historical Development and Current Status of Wind Power*, ser. Wind Power in Power Systems, Second Edition. John Wiley and Sons, 2012, pp. 41–42.

[14] *Offshore Wind Energy Utilisation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 615–652. [Online]. Available: https://doi.org/10.1007/3-540-29284-5_17

[15] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital Twin: Enabling Technologies, Challenges and Open Research," *IEEE Access*, vol. 8, pp. 108 952–108 971, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9103025

[16] R. M. Cabeza Gavira, "Industria 4.0 y sus aplicaciones a la optimización de procesos y eficiencia energética," pp. 16–17, 2018. [Online]. Available: https://hdl.handle.net/11441/82651

[17] F. Tao, H. Zhang, A. Liu, and A. Y. Nee, "Digital Twin in Industry: State-of-the-Art," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405–2415, apr 2019.

[18] M. Grieves, "Digital Twin : Manufacturing Excellence through Virtual Factory Replication - A White-paper by Dr . Michael Grieves," *White Paper*, no. March, pp. 1–7, 2014.

[19] M. Grieves and J. Vickers, *Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems*. Cham: Springer International Publishing, 2017, pp. 85–113. [Online]. Available: https://doi.org/10.1007/978-3-319-38756-7_4

[20] Mathworks, "No Title." [Online]. Available: https://es.mathworks.com/discovery/digital-twin.html

[21] Q. Qi, F. Tao, T. Hu, N. Anwer, A. Liu, Y. Wei, L. Wang, and A. Y. Nee, "Enabling technologies and tools for digital twin," *Journal of Manufacturing Systems*, vol. 58, pp. 3–21, jan 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S027861251930086X?via%3Dihub#fig0020

[22] Q. Qi and F. Tao, "Digital Twin and Big Data Towards Smart Manufacturing and Industry 4.0: 360 Degree Comparison," *IEEE Access*, vol. 6, pp. 3585–3593, jan 2018.

[23] A. Kusiak, "Smart manufacturing must embrace big data," *Nature 2017 544:7648*, vol. 544, no. 7648, pp. 23–25, apr 2017. [Online]. Available: https://www.nature.com/articles/544023a

[24] V. Lope Salvador, X. Mamaqi, and J. Vidal Bordes, "La inteligencia artificial," *Revista ICONO14 Revista científica de Comunicación y Tecnologías emergentes*, vol. 18, no. 1, p. 60, 2020.

[25] D. Hassabis, D. Kumaran, C. Summerfield, and M. Botvinick, "Neuroscience-Inspired Artificial Intelligence," pp. 245–258, jul 2017.

[26] Laughsinthestocks, "A plot of the identity activation function," 2015. [Online]. Available: https://en.wikipedia.org/wiki/File:Activation_identity.svg

[27] ——, "A plot of the rectified linear activation function," 2015. [Online]. Available: https://en.wikipedia.org/wiki/File:Activation_rectified_linear.svg

[28] ——, "A plot of the tanh activation function," 2015. [Online]. Available: https://en.wikipedia.org/wiki/File:Activation_tanh.svg

[29] ——, "A plot of the logistic activation function," 2015. [Online]. Available: https://en.wikipedia.org/wiki/File:Activation_logistic.svg

[30] ——, "A plot of the binary step activation function," 2015. [Online]. Available: https://en.wikipedia.org/wiki/File:Activation_binary_step.svg

[31] MATLAB, "Levenberg-Marquardt backpropagation." [Online]. Available: https://es.mathworks.com/help/deeplearning/ref/trainlm.html