industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Estudio de nuevos modelos matemáticos de predicción de la potencia generada por un ciclista usando variables biométricas

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: **Carlos Antonio Mayancela Punin**
Director: Julio José Ibarrola Lacalle
Codirector: José Manuel Cano Izquierdo.

Cartagena, 15 de Julio de 2022

Universidad Politécnica de Cartagena

Agradecimientos

Durante mi etapa académica he tenido la posibilidad de conocer a muchos compañeros y profesores de los cuales me llevo un grato recuerdo. En muchas ocasiones he tenido que plantearme si realmente merecía la pena todo el esfuerzo realizado, debo decir que sí. Los esfuerzos se ven recompensados y gracias a él, tengo la posibilidad de presentar este trabajo fin de grado.

Quisiera agradecer a mi director, D. Julio José Ibarrola Lacalle, y a mi codirector, D. José Manual Cano izquierdo, por el apoyo brindado durante la elaboración de mi proyecto. Su confianza y predisposición durante el desarrollo de este TFG, me ha permitido resolver cualquier duda y, sobre todo, me ha permitido seguir creciendo y aprendiendo.

A mis padres por todo el esfuerzo a lo largo de este camino. Porque cada vez que sentía que no podía más, sabía que no podía defraudarlos. De igual manera, a mis hermanos que a pesar de la distancia siempre han estado apoyándome en todo momento. El valor y la constancia con la que decidí afrontar este camino fue gracias a ellos.

A Camila, mi compañera de aventuras con la que inicie esta etapa. Gracias por no haberme dejado tirar la toalla y por todos estos años en los que has sido un pilar fundamental en mis desafíos.

A todos ellos, a los que ya no están y a los me esperan siempre, gracias.

Índice general

Índice de figuras

Índice de tablas

Acrónimos y símbolos

P	Potencia
F	Fuerza
P_m	Potencia media
v	Velocidad
w	Velocidad angular
r	Radio
T	Torque
D	Resistencia aerodinámica
Cd	Coeficiente de arrastre [1]
ρ	Densidad
v_a	Velocidad del aire
A	Área frontal proyectada
FTP	Umbral de potencia funcional [2]
NP	Potencia normalizada
TSS	Factor de estrés
ANFIS	Sistema adaptativo de interferencia neuro-difusa[3]
FC	Frecuencia cardiaca
R	Coeficiente de Pearson
RMSE	Error medio cuadrático [4]
MAD	Desviación media absoluta [5]
MAE	Error medio absoluta [6]

[1] *Drag coefficient*
[2] *Funtional Threshold Power*
[3] *Adaptative neuro fuzzy interference system*
[4] *Root-mean-squared error*
[5] *Mean absolute desviation*

e_m	Error medio
R^2	Coeficiente de determinación
GMDH	Grupo de agrupamiento para el manejo de datos[7]
CR	Criterio de Regularidad

[6] *Mean absolute error*
[7] *Group Method of Data Handling*

Resumen

Los últimos años se ha incrementado el número de aficionados al ciclismo. Según el informe *El sector de la bicicleta en cifras 2021,* publicado en abril del 2022 por la *Asociación de Marcas y Bicicletas de España (Ambe),* donde se recoge el número de ventas del año anterior, se puede observar que el volumen de negocio del sector se ha visto incrementado en un 10.76%. Muchos de los clientes son aficionados y semiprofesionales que como cualquier deportista tiene la necesidad de monitorear su actividad.

En el ciclismo profesional es común usar potenciómetros como herramienta para medir la potencia de pedaleo, tanto es así que este parámetro se usa como medida de la eficiencia del ciclista, es decir, cuanto más eficiente sea el pedaleo, más rendimiento se consigue con el mismo esfuerzo. Entrenar con un ciclocomputador permite conocer la potencia desarrollada en vatios en tiempo real por lo que se puede planificar una carrera o un entrenamiento. Detrás del monitoreo existe una ciencia de análisis de datos que brindan una gran cantidad de información que permite al deportista entrenar con la intensidad necesaria en cada momento.

Aunque en últimos años esta herramienta ha bajado de precio considerablemente, para muchos aficionados y semiprofesionales, aún es inaccesible. Muchos de ellos optan por monitorear la potencia mediante aplicaciones, pulseras de actividad o bandas de cardio. Esto hace que sea necesario buscar alternativas a los ciclos computadores que puedan ser asequibles para los ciclistas *amateur*.

Por tanto, se propone el desarrollo de modelos matemáticos para la predicción de la potencia generada por un ciclista. Para estos modelos se tomará como entrada variables biométricas que son sencillos de obtener y, por tanto, accesibles a cualquier persona. La implementación de estos modelos tiene como objetivo el estudio de los modelos justificando su elección, seleccionar aquellas variables biométricas que sean más representativas para cada modelo, pues un modelo matemático robusto no es aquel que mejor se ajuste, sino el que mejor generaliza, y finalmente obtener resultados que permitan definir cuál de ellos proporciona mejores predicciones de la potencia generada.

Abstract

In the last few years, the number of amateur cyclists has increased. According to the report *El sector de la bicicleta en cifras 2021*, published in April 2022 by *Asociación de Marcas y Bicicletas de España (Ambe)* in which the number of sales from previous year has collected. We can see that the annual turnover has increased by 10.76%. Many customers are amateurs and semi-professional who want to monitor theirs progress and plan their training.

The professional cyclist uses power sensor or potentiometers as tool to measure pedal power. Pedalling power is used to measure training efficiency, in such a way that with little effort we can obtain grater performance in a race or training. If real time power is known, the training can be planning by the cyclist team. Behind monitoring there is a science of data analysis that provides a large amount of information which allows the athlete to train with the necessary intensity at each moment.

This device has dropper in price in recent years, but it still has a very high price for many semi-professionals cyclist. Some of them choose monitor the power through applications or activity bracelets. This has led to the search to develop a power prediction model that can be implemented in a computer cycle, making the price much lower.

Therefore, the development of mathematical models for the power generated by a cyclist is proposed. For these models, biometric variables that are easy obtain and accessible to anyone will be taken as input. The implementation of these models aims to study the models, justify the choice, select those biometric variables that are most representative for each model, since a robust mathematical model is not the one that best fits, but the one that best generalizes. Finally, we can obtain results to know which of them provides better predictions of the pedal power.

1. Introducción y objetivos

Como en cualquier otro deporte, en el ciclismo es imprescindible conocer la eficiencia del ciclista. Es bien conocido que los deportistas profesionales durante una subida miran hacia el manillar y no hacia el frente, pues están consultando datos de la potencia desarrollada para intentar optimizar la frecuencia del pedaleo. Hoy en día, esta práctica ya no es exclusiva de los profesionales y los aficionados también necesitan conocer estos datos para diseñar sus entrenamientos.

El sensor de potencia o potenciómetro es un aparato que suele estar situado en la biela de los pedales y gracias a él es posible medir la relación entre fuerza y velocidad o cadencia. Esta relación se envía a ciclocomputador donde se visualiza los vatios, por tanto, fijándonos en la fuerza y el número de veces que pedaleamos en un tiempo determinado es posible conocer la potencia, pero ¿no influyen otros factores?

1.1. Factores que actúan en el ciclismo

Es quizás el parámetro más influyente en el ciclismo es la potencia, pues poder medirla permite planificar una carrera o generar un plan de entrenamiento específico para cada deportista dentro de un equipo. La potencia mecánica se define como la cantidad de fuerza y la velocidad con la que se aplica.

$$P = F \cdot v$$

Si aplicamos la definición anterior al ciclismo, esta relación se define como el momento de fuerza o torque por la velocidad angular de la biela, es decir, la cadencia, luego se puede alcanzar mayor potencia llevando una candencia alta. Sin embargo, este parámetro se ve afectado por la longitud de la biela, usando bielas más cortas se conseguirá más cadencia, pero perdemos torque. Bielas más largas conseguirá que el torque sea mayor pero la cadencia disminuirá. La elección de uno u otro dependerá del deportista.

Figura 1: *Relación entre velocidad y fuerza en la biela.*

Otro factor a tener en cuenta es la resistencia aerodinámica, pues está directamente relacionado con la estructura de la bicicleta, la resistencia al viento del casco e incluso a la complexión y peso del ciclista. En los últimos años ha cobrado mayor protagonismo en el ciclismo profesional, tanto es así, que los fabricantes diseñan los cuadros y componentes teniendo en cuenta la aerodinámica de la bicicleta.

La ecuación que describe la resistencia aerodinámica depende de cuatro factores:

$$D = \frac{1}{2}\rho C_d A v^2$$

D: Resistencia aerodinámica (N)
ρ: Densidad del aire (kg/m^3)
A: Área proyectada por el ciclista y la bicicleta (m^2)
Cd: Coeficiente aerodinámico
v: Velocidad relativa (m/s)

Si observamos la expresión, la velocidad es el parámetro más destacado y los tres primeros factores influyen de manera lineal, por lo que, a menor densidad del aire, mayor será la potencia generada. De igual forma, si el área frontal del ciclista, junto con el de la bicicleta, es menor, ofrece menor resistencia aerodinámica, por tanto, mayor eficiencia en el pedaleo.

1.2. Instrumentos de medida

Un ciclista aficionado es común que utilice un pulsómetro como herramienta de medida. Estos dispositivos se anclan al manillar y pueden registrar la velocidad, cadencia y gasto calórico mediante una banda de cardio que se conecta por bluetooth. A partir de esta información el software puede ofrecer información en tiempo real de la potencia generada. Evidentemente estos datos no son los más aproximados, pero son muy versátiles y permiten planificar los entrenamientos.

En el ciclismo profesional se utiliza sensores de potencia, pues ofrece mayor precisión desde que se inicia el movimiento. Este sistema detecta deformaciones de galgas colocadas en bielas, cadenas, bujes, etc. Esta deformación se transforma a pulsos de señal eléctrica que se transmiten a ciclo computador mediante tecnologías inalámbricas. El protocolo utilizado para transmitir suele ser ANT+, compatible con la mayoría de las variedades de ciclo computadores del mercado, por lo que, el usuario puede usar diferentes marcas de sensores para una única unidad de procesado.

Un estudio realizado en 2017 (Maier, Schmid, Müller, Steiner, & Wehrlin), publicó resultados de una comparativa entre 54 sensores de potencia de 9 fabricantes, usados por

32 ciclistas. El estudio tenía como objetivo comparar la precisión de cada uno de los sensores mientras las ciclistas descendían cuesta abajo en una cinta rodante.

Entre los sensores que se pusieron a prueba encontramos modelos utilizados por competidores profesionales y sensores de gama baja. Entre los cuales observamos XX1 de la marca Quarq, Vector de la marca Garmin o G3 de la marca PowerTap. Como resultado se obtuvo la siguiente tabla.

n	Manufacturer	Mean deviation (%)	Coefficient of variation (%)	Cadence (RPM)
12	SRM	−0.5 ± 2.4	0.8 ± 0.4	83 ± 14
10	PowerTap	0.9 ± 2.1	0.8 ± 0.2	87 ± 5
11	Quarq	0.5 ± 3.0	1.3 ± 0.8	87 ± 6
13	Stages Cycling	−2.9 ± 3.9	2.0 ± 1.4 *	89 ± 6
3	Verve Cycling	−1.7 ± 1.1	0.6 ± 0.4	88 ± 3
2	power2max	−4.8 ± 3.4	1.5 ± 0.4	87 ± 16
1	Garmin	−2.0	1.6	86
1	Polar	−3.9	2.6	93
1	Rotor	2.1	0.4	84
54	All	−0.9 ± 3.2	1.2 ± 0.9	87 ± 8

Values are presented as mean ± standard deviation (if $n > 1$).
* Different from SRM and PowerTap $p < 0.05$

***Tabla 1**: Resultados de comparativa entre sensores*

Se puede observar que en cuanto al coeficiente de variación las marcas SRM y PowerTap son algo más precisos, pero todo depende de la forma en que se desarrollen los experimentos. Por tanto, podemos decir que existe grandes variaciones en la precisión en función del modelo y del fabricante, pero todos están dentro de unos rangos de valores aceptables. Lo que demuestra este estudio es que la precisión entre modelos de gama alta, utilizados por ciclistas profesionales, es mucho más alta que los utilizados por ciclistas aficionados. Como conclusión podemos decir que, si queremos datos más aproximados, es conveniente calibrar el sensor de potencia cada cierto periodo de tiempo. (Maier, Shmid, Müller, Steiner, & Wehrlin, 2017).

1.3. Indicadores de eficiencia

Si lo que se quiere es obtener el mayor rendimiento posible es necesario tener en cuenta algunos indicadores que permiten comprender cuándo una actividad está siendo óptima. El análisis de cada uno de ellos arroja resultados del estado físico del ciclista, los puntos débiles y fuertes y la cantidad de esfuerzo realizado.

El primero de ellos es la potencia media, representa el promedio de vatios que hemos generado en un lapso determinado de tiempo. Normalmente el periodo de tiempo es inferior a 5 segundos, por lo que, los datos son precisos y reales, pero no resultan ser nada útiles, pues no se tienen en cuenta cambios de ritmo, cambios en la pendiente, etc. (Cragnulini, s. f.).

Para valorar estas variaciones en los entrenamientos se debe recurrir a la potencia normalizada (NP). No es más que el resultado de un algoritmo que contempla cambios de intensidad en duraciones mayores a 30 segundos. Normalmente su valor es mayor que la media, esto es porque en su cálculo se contempla momentos en los que no se pedalea (bajadas o descansos).

Supongamos que una actividad consta de 5 min de subida a máxima potencia, 400w, y 5min de bajada en una pendiente a 100w. La potencia media de los últimos 10 min ha sido de 300w, pero ¿y si esta actividad se realiza en un terreno llano durante 10 minutos? La media en ambos casos es la misma, pero en el primer caso la media normalizada será mayor, pues a mayores cambios en la intensidad mayor diferencia entre la media y la media normalizada.

Hablamos de potencia absoluta cuando nos referimos al total de vatios generados, mientras la potencia relativa se define como los vatos por peso (w/kg). A la hora de comparar rendimientos entre varios ciclistas es mejor usar el segundo parámetro, pues estamos obteniendo la potencia generada relacionada a su peso. Cuanto mayor sea la ratio w/kg mejor será el rendimiento. También, es necesario definir la potencia máxima. Representa el valor máximo de potencia obtenido durante un entrenamiento completo. Este dato es más representativo en competiciones donde se realice grandes esfuerzos como un sprint. (Cragnulini, s. f.).

FTP *(Funcional Threshold Power)* o umbral funcional de potencia. Este parámetro mide la potencia que el ciclista es capaz de mover durante una hora de entrenamiento. Generalmente este parámetro se usa como medida de la evolución del ciclista, pues depende directamente de la condición física, los valores de FTP son representativos de la época donde se encuentra el deportista. A medida que se mejore las capacidades físicas mejor valor de FTP obtendremos. (Allen & Coggan, 2014).

TSS (Training Stress Score) o factor de estrés durante un entrenamiento. Se trata de un parámetro que mide la carga física que ha supuesto una actividad. A diferencia de la potencia normalizada que no tiene en cuenta el periodo donde se ha realizado la actividad, este término tiene en cuenta la intensidad, la carga física y el tiempo. De esta forma se puede medir la recuperación que puede tener un ciclista los días posteriores al entramiento o carrera. (Allen & Coggan, 2014).

1.4. Objetivos

Para este trabajo fin de grado nos planteamos como primer objetivo, el estudio y desarrollo de nuevos modelos para la predicción de potencia generada a partir de datos biométricos. En anteriores estudios se propusieron diferentes métodos predictivos, por lo que nuestro objetivo es ampliar el abanico de modelos. Desde modelos de regresión, hasta redes neuronales con la finalidad de obtener aquel que se adecue mejor a nuestros datos.

Además, es necesario estudiar las variables que nos interesan, es decir, se usará métodos de selección de variables con el fin de optimizar los modelos de predicción. El estudio de las variables es importante, pues nuestro objetivo en este caso no es obtener un modelo que se ajuste a los datos de entrada, si no que pueda generalizar sea cual sea los datos utilizados. De esta manera evitamos problemas de sobreparametrización o problemas de *overfitting*.

Por último, cada modelo generado deberá ser validado usando datos de actividades que no se haya utilizado para el entrenamiento. Mediante diferentes parámetros debemos ser capaces de comparar los cuatro modelos para obtener aquel que nos de mejores resultados.

2. Estado del arte

A lo largo de los años hemos tenido la suerte de observar el desarrollo tecnológico en diferentes deportes, el ciclismo no se queda atrás. La innovación en nuevos materiales, componentes o la mejora de su aerodinámica han sido el mayor avance en las últimas décadas. Una de ellas es la inclusión del sensor de potencia y ciclo computadores. Permite que ciclistas profesionales tengan una forma de medir su rendimiento y evaluar los puntos débiles durante una carrera.

Como todo avance, el desarrollo de estos componentes tiene un coste accesible solo a equipos de competición o profesionales, pero hay semiprofesionales o aficionados a los que también les gustaría contar con una herramienta como esta.

Desde finales de los años 70 se han publicado diferentes artículos relacionados con la predicción de potencia generada. Algunos de estos modelos basan en factores como la velocidad, aerodinámica, resistencia a la rodadura o principios fisiológicos. Muchos de ellos se usaban para la predicción de la velocidad o la predicción del rendimiento, y no es hasta finales de la noventa cuando se obtiene modelos para la predicción de potencia.

2.1. Modelos existentes

2.1.1. Modelo Di Prampero y colaboradores

El modelo presentado por Di Prampero et al. tuvo el objetivo de evaluar de forma cuantitativa como afecta el tamaño del cuerpo, la presión de los neumáticos, la temperatura del aire y la inclinación del terreno en la potencia mecánica desarrollada por el ciclista. Es un modelo basado en las fuerzas de oposición al movimiento que aparecen, como son la resistencia aerodinámica y la resistencia a la rodadura. (Di Prampero, Cortili, Mognoni, & Saibene, 1979).

Expone que durante la práctica del ciclismo las pérdidas por fricción en los radios de las ruedas son muy pequeñas. Por este motivo, la resistencia total (R_T) que se opone al movimiento del ciclista es la suma de: la resistencia a la rodadura (R_R), que son las pérdidas en las ruedas a lo largo del camino y la resistencia aerodinámica (D). La resistencia a la rodadura depende sustancialmente de la presión de los neumáticos y de las características de la superficie del terreno y de los neumáticos. Esta resistencia es proporcional al peso total del conjunto del ciclista y la bici y es constante, independientemente de la velocidad. La resistencia al aire es función del área proyectada en el plano frontal (A_P), la densidad del aire (ρ), y la velocidad del aire (v_a). (Di Prampero, Cortili, Mognoni, & Saibene, 1979).

$$R_T = R_R + D$$

$$D = \frac{1}{2} * CD * A_P * \rho * v_a^2$$

Los valores del coeficiente CD es función del número de Reynols. Sin embargo, en este rango de velocidades y para la postura dada en el ciclismo, CD es constante. De esta manera, para una densidad del aire concreta y una postura del ciclista, el valor de $(\frac{1}{2} * C_D * A_P * \rho)$ es constante y puede ser sustituido por la letra (k). (Di Prampero, Cortili, Mognoni, & Saibene, 1979).

$$R_T = R_R + k * v_a^2$$

Aproximadamente R_T tiene un valor de 3.2 N y k de 0.19 $Nm^{-2}s^2$. Estos valores son calculados por Pugh, usando el oxígeno consumido durante la actividad y la potencia medida con un ergómetro, y por Nonweiler en un túnel de viento. El valor de R_R obtenido fue aproximadamente el doble del propuesto por Pugh durante una carrera (7.2 N). Por lo tanto, el valor de la resistencia a la rodadura varía dependiendo del terreno y de los neumáticos. (Di Prampero, Cortili, Mognoni, & Saibene, 1979).

De esta manera, la potencia mecánica de salida (W) viene dada por el producto de la resistencia total por la velocidad del ciclista (s):

$$W = R_R * s + 0.19 * v_a^2 * s$$

La ecuación anterior no tiene en cuenta factores como el peso, el perfil del ciclista o la densidad del aire, por lo que su implementación requiere un estudio mucho más profundo a que debemos añadir factores externos.

2.1.2. Modelo Martin y colaboradores

Es un modelo predictivo basado en principios físicos y de ingeniería, con gran precisión y profundizando en los factores que influyen en la producción de la potencia. Los factores que incluyen en su modelo son la resistencia aerodinámica que está relacionada con la densidad del aire, la superficie frontal del ciclista y de la bicicleta, la velocidad del aire, la resistencia a la rodadura, que está relacionada con el peso de la bici y el ciclista, la presión y el material de los neumáticos, del gradiente y de la textura de la superficie del suelo. Los cambios en la energía cinética están relacionados con la masa, la inercia y la velocidad, y por otro lado, los cambios en la energía potencial están influidos por la masa, la gravedad y la velocidad vertical. (Martin, Milliken, Cobb, McFadden, & Coggan, 1998).

El propósito de su investigación fue determinar si la potencia generada por el ciclista podía ser precedida por un modelo matemático, mediante la validación de la potencia medida por un sensor de potencia SMR, la derivación de un modelo matemático basado en principios físicos y de ingeniería y la determinación de los valores de los diferentes parámetros del modelo y su posterior comparación entre los valores de potencia estimados y los medidos por el sensor de potencia. Llegaron a la conclusión de que la potencia podía ser medida y estimada mediante un modelo matemático, ya que el error estándar de medida entre la potencia modelada y la potencia medida fue de tan solo 2.7 W. (Martin, Milliken, Cobb, McFadden, & Coggan, 1998).

Este modelo tiene en cuenta varios factores: Resistencia aerodinámica, resistencia al giro de las ruedas por el aire, rozamiento de la rueda con la calzada, pérdidas por fricción y diferencias de energía potencial y cinética. Para cada factor se desarrolla una serie de ecuaciones que llevan a una ecuación matemática final.

$$P_{TOT} = \frac{P_{AT} + P_{RR} + P_{WB} + P_{PE} + P_{KE}}{Ec} =$$

$$P_{TOT} = (1/2 * \rho * (C_D * A + F_W) * V_A^2 * V_G + V_G * C_{RR} * m_T * g + V_G * (91 + 8.7V_G) * 10^{-3} + V_G * mT * g * sin(tan^{-1}(G_R)) + 1/2 * (m_T + I/r^2) * (V_{GF}^2 - V_{GI}^2)/(t_I - t_F)/Ec$$

Donde ρ es la densidad del aire, Cd es el coeficiente de rozamiento, A es el área frontal, v_a es la velocidad del aire de forma tangencial, m es la masa total de la bici y el ciclista, g es la gravedad y G_R es el gradiente de la carretera.

F_W es un factor asociado con la rotación de las ruedas que representa el área de rozamiento de los radios, P_{AT} es potencia aerodinámica total generada por el ciclista y la bici, y la rotación de las ruedas. P_{RR} es la fuerza de rozamiento del giro de las ruedas con el suelo. Además, P_{WB} son las pérdidas por fricción relacionadas con la velocidad del giro de las ruedas y P_{EP} y $W_{KE,}$ son los cambios en la energía potencial y cinética, respectivamente.

2.1.3. Modelo de Árbol de Regresión

El trabajo fin de grado, titulado *Desarrollo de un modelo matemático de predicción de la potencia generada por un ciclista usando variables biométricas* (Carrillo Vera, 2019) desarrolla, tanto los modelos de conocimiento de Di Prampero y colaboradores y de Martin y colaboradores, como un primer modelo matemático basado en herramientas de regresión. Este primer modelo se basa en el método predictivo de árbol de regresión que se implementa mediante la herramienta *Regresión Learner* de Matlab.

El planteamiento es similar al que se utilizará más adelante. En primer lugar, se obtiene los datos del sensor de potencia y se genera una matriz de datos con las siguientes variables: distancia, altura, frecar, cadencia, velocidad, velaire, pedaleo y gradiente. Esta información corresponde a la actividad 3 de los datos que se usará a lo largo de este trabajo fin de estudio.

Modelo	RMSE	Error medio	Potencia media	Potencia media real
Modelo Di Prampero	124.93	-17.93	153.05	135.12
Modelo Martin	115.98	5.87	129.25	135.12
Modelo Árbol de Regresión	50.15	12.24	122.89	135.12

Tabla 2: *Resultados globales de los modelos.*

Para generar los modelos de conocimiento se recurre a las ecuaciones de cada uno de los modelos mencionados, generando una respuesta. En ambos casos es necesario aplicar un filtro a la respuesta con una media de los 10 valores anteriores donde, además, se introduce la variable velocidad del aire. Para el modelo de regresión no se utiliza, pues esta variable no se puede recoger en tiempo real y se necesitaría un sensor para ello.

Los resultados obtenidos para los modelos de conocimiento y para el modelo de Árbol de regresión, se recogen en la tabla 2. Observamos los resultados globales de los tres modelos, donde el modelo de Martin es el que menor error medio tiene, pero es el modelo de Árbol de regresión el que mejor RMSE tiene, con datos de potencia media aceptables.

2.1.4. Modelo Gillaume Lemaître

Es un modelo de *machine learning* desarrollado por Guillaume Lemaître en el que se toma datos de 417 actividades de 5 ciclistas utilizando diferentes potenciómetros. Para ello toma las variables de potencia, frecuencia cardiaca, velocidad, cadencia, distancia y elevación. Además, introduce otras variables como la aceleración, la pendiente y la derivada de la frecuencia cardiaca que son calculados a partir de los predictores iniciales.

Se propone comparar los resultados de utilizar un aprendizaje automático con el siguiente modelo matemático:

$$P_{meca} = (0.5 * \rho * SC_x * v_a^2 + Cr * mg * cos(\alpha) + mg * sin(\alpha)) * Vd$$

Donde ρ es la densidad del aire en $kg.m^{-3}$, S es la superficie frontal del ciclista en m^2, C_x es el coeficiente de arrastre, v_a es la velocidad del aire en $m.s^{-1}$, C_r es el coeficiente de rodadura, m es la masa del ciclista y la bicicleta en kg, g en la constante gravitacional que es igual a 9,81 $m.s^{-2}$, α es la pendiente en radianes y V_d es la velocidad del ciclista en $m.s^{-1}$.

Para obtener los resultados del modelo de *machine learning* se ha validado utilizando una validación cruzada, obteniéndose un R^2 de 0.71, mientras que el error absoluto medio (MAE) es de 25.1W. Para el modelo matemático obtiene un R2 de 0.26 y un MAE de 55.2 W.

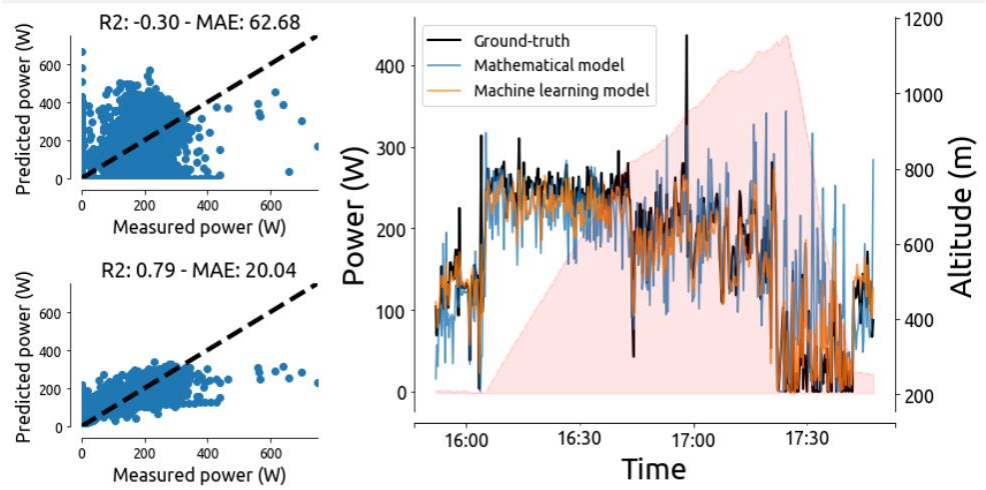

Figura 2: *Resultados del modelo Gillaume Lemaître*

En la figura anterior se recogen los resultados de este estudio, donde el modelo de aprendizaje automático tiende a ser más estable que el modelo matemático con el que se compara. Se concluye que este modelo permite estimar la potencia en tiempo real sin ninguna infraestructura importante como túneles de viento y, además, obtiene un mejor rendimiento que los modelos de regresión lineal simple. (Lemaître & Lemaître, 2018).

3. Metodología y método predictivos

3.1. Metodología

Evidentemente, antes de realizar cualquier desarrollo experimental, necesitamos datos reales tomados de un sensor de potencia. En este caso, el potenciómetro utilizado es de la marca Favero Electronics y el ciclo computador se corresponde con el modelo Edge 820 de la marca Garmin.

El sensor proporciona datos de potencia instantánea, junto con parámetros como la velocidad instantánea, cadencia, frecuencia cardiaca (FC), latitud, longitud, altura, distancia, entre otros muchos que no son relevantes para el objeto de este trabajo. Estos datos corresponden a una misma persona, tomando diferentes actividades en distintos momentos.

El ciclo computador permite la salida de archivos con extensión ".TCX", un tipo de archivo especifico de la marca y que no es de fácil manipulación, por lo que se opta por una conversión a un archivo Excel y posteriormente se convierte en matriz de Matlab de tipo ".mat". Esto facilita el acceso a todas las columnas anteriormente mencionadas, y podemos trabajar y manipular las variables según la necesidad de cada modelo.

A continuación, se describe la estructura que se implementará para generar modelos predictivos:

1. Una vez tenemos los datos, será necesario seleccionar los métodos predictivos más interesantes. Estos modelos están implementados en Matlab y mediante comando o ventanas podemos modificar parámetros que pueden mejorar la predicción.
2. También es necesario un procesado de datos previos, pues muchos de ellos tienen valores nulos o valores que no son posibles. Además, seleccionaremos las variables que sean útiles para nuestro trabajo. En concreto nos basaremos en: velocidad, inclinación, cadencia, FC, distancia y altura.
3. Otro factor a tener en cuenta es el filtrado de datos. Los datos de potencia instantáneos no son útiles para un ciclista que realice una actividad, pues el ciclo computador nos estaría arrojando datos cada segundo que apenas pueden variar. Por ello, estas herramientas están configuradas para darnos la potencia de media de las últimas cinco o diez muestras. De esta manera se reduce las grandes variaciones que puede sufrir la potencia cada segundo, y que no son representativas para un entrenamiento o actividad.

4. Además, debemos generar los tipos específicos de formato de datos que requieren todos los modelos, es decir, tablas, matriz de datos o formatos específicos.
5. Una vez se haya generado de manera correcta los datos, podemos iniciar con el desarrollo de los modelos. En primer lugar, estudiaremos la correlación de las 6 variables seleccionadas con el objetivo de descartar aquellas que no son relevantes para los modelos.
6. A continuación, generamos modelos con las variables disponibles.
7. Posteriormente, implementaremos un método de selección de variables. Aquellas variables que se hayan obtenido serán las que utilicemos para generar modelos definitivos.
8. Por último, es necesario validar los modelos. Para ello se utilizará datos que no se hayan utilizado para generar los modelos, de tal forma que obtendremos una tabla de validación cruzada para modelos generados con una actividad y validados con otra actividad distinta.

Este procedimiento se aplicará para los cuatro modelos seleccionados y de esta forma podremos comparar los resultados de todos ellos, pero ¿cómo medimos la calidad del modelo? Pues bien, utilizaremos diferentes parámetros como son: RMSE, R^2, MAD y MAE.

- RSMSE: *Root-mean-squared error* o error medio cuadrático mide la cantidad de error que hay entre dos conjuntos de datos, es decir, compara el valor predicho con el valor observado.

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

  Siendo $\hat{y}_i$ el valor esperado, y_i es el valor observado y *n* el número de observaciones.

- R^2: El coeficiente de determinación refleja el ajuste de bondad de un modelo a la variable que se pretende explicar.

- MAD: *Mean absolute deviation* o desviación media absoluta mide la dispersión de un valor observado en relación con valor medio de la muestra.

$$MAD = \frac{1}{n}\sum |x_i - m|$$

  x_i son los valores individuales, *m* es la media de la muestra y *n* el número de observaciones.

- MAE: *Mean absolute error* o error medio absoluto es una magnitud que mide el promedio de los errores sin importar la dirección que se tome.

$$MAE = \frac{1}{n}\sum|y_i - \hat{y}_i|$$

Donde $\hat{y}_i$ el valor esperado, y_i es el valor observado y n el número de observaciones.

3.2. Métodos predictivos

La modelización es una forma de crear una representación a través de ecuaciones o funciones matemáticas de la relación que existe entre dos o más variables. Los modelos matemáticos son usados para analizar, explicar o describir fenómenos que podrían ocurrir partiendo de la relación de datos de entrada.

Los modelos pueden ser simples o muy complejos, pero todos comparten características básicas:

- Variables: Son conceptos u objetos que buscamos analizar. La relación que exista entre ellas será la que permita conocer la variable predictora. Así, por ejemplo, una variable de entrada puede ser la cadencia de un ciclista y lo que queremos analizar es la potencia desarrollada durante un entrenamiento.
- Parámetro: Son aquellos valores conocidos o que se pueden controlar del modelo.
- Restricciones: Son límites que nos indican si los datos de entrada o salida son razonables. Por ejemplo, si una variable es la frecuencia cardiaca de un ciclista, una restricción será que el valor sea siempre positivo, pues nunca tendremos valores negativos.
- Relación entre variables: Será el modelo quien establezca la determinada relación entre variables basándose en teorías, algoritmos, etc.
- Representaciones: Cuando se desarrolla un modelo es habitual que se realicen representaciones de las diferentes relaciones que existen entre variables. Por ejemplo, podemos representar la potencia desarrollada en función del tiempo y cuantificar el comportamiento o tendencia que se observa.

Mediante estos elementos básicos del modelo podemos generar infinidad de modelos, pero ¿Cómo sabemos cuál es más indicado para nuestra aplicación? Cuando se diseña un modelo se busca que sea simple, es decir, sea entendible para cualquiera, que sea estable frente a cambios en las variables y robusto, pues buscamos generalizar y que no solo sea aplicable a un caso en concreto.

3.2.1. Regresión no lineal

Un modelo lineal simple es aquel cuya variable dependiente (Y) se determina a partir de una variable de entrada llamada predictor (X_1). Si tenemos un conjunto de predictores (X_1, X_2, X_3 …) se denomina regresión lineal múltiple. En ambos casos encontramos que son fácilmente interpretables, pero tienen grandes limitaciones en la predicción. Esto se debe a que al utilizar estos modelos asumimos que nuestros datos tienen un comportamiento lineal, pero muchas veces es una aproximación demasiado simple. (Rodrigo J. A., Estadística y Machine Learning con R, 2017).

Luego, nuestro primer modelo elegido es de tipo regresión polinómica. Conseguimos añadir cierta curvatura a nuestro modelo introduciendo predictores elevados a potencias. Esta es la forma más sencilla de incorporar mayor flexibilidad al modelo.

Si partimos de un modelo lineal,

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

Obtenemos un modelo de regresión polinómica de grado n al elevar los predictores a la potencia.

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \beta_3 X_i^3 + \beta_n X_i^n + \epsilon_i$$

Este modelo puede ajustarse por mínimos cuadrados, pues, aunque no sea un modelo lineal no deja de ser una ecuación lineal con predictores X, X^2, X^3, …, X^n.

Como ejemplo, tenemos la Figura 3, donde representamos un predictor y su respuesta. Observamos que la relación lineal no se podría aplicar en este caso, pues cometeríamos errores grandes, pero si subimos a un polinomio de grado 2 la curva se ajusta mejor.

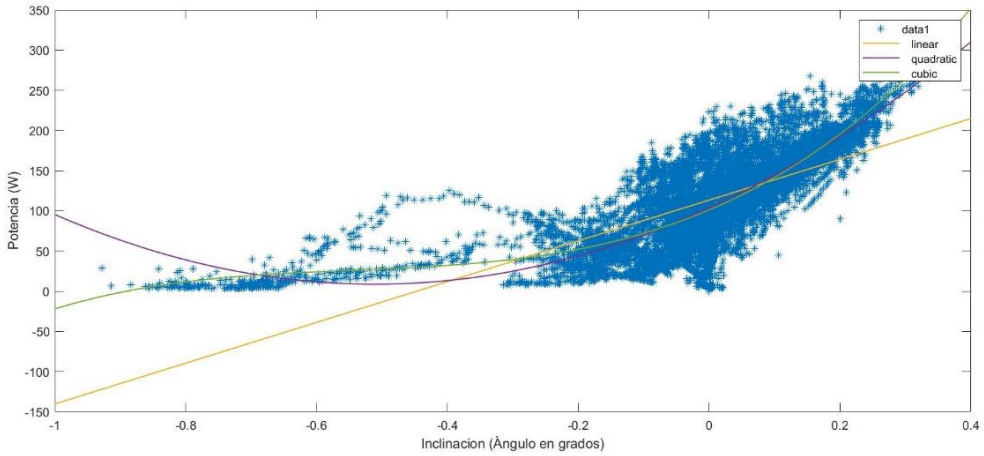

Figura 3: *Relación entre predictor y la salida*

Todo parece indicar que modelo de polinomio 3 es el que se ajusta mejor, pero se tendría que estudiar si realmente es necesario un modelo cúbico, pues puede darse el caso que un modelo este excesivamente ajustado a la ecuación de orden 3 y al introducir nuevos datos obtengamos peores resultados. En muchos textos se desaconseja el uso de modelos polinómicos de grado mayor a 3 o 4 debido a un exceso de flexibilidad. La implementación del modelo se realizará con Matlab, utilizando la función *fitnlm(tbl,modelfun,beta0),* donde *tbl* es la matriz de datos que utilizamos que integra la potencia, *modelfun* es el modelo no lineal a utilizar y *beta0* son valores iniciales de los coeficientes de la ecuación no lineal.

3.2.2. Árbol de Regresión

El segundo modelo elegido es el Árbol de Regresión. Cuando se trata de explicar la relación entre múltiples variables suele ser el más elegido por su fácil interpretación y no requiere manipulación de los datos. Los métodos predictivos como la regresión lineal o polinómica tienen excelentes resultados cuando tenemos pocos predictores, pero cuando se trata de estudiar la relación entre múltiples predictores que no se explica de manera lineal, debemos recurrir a método más avanzados.

Se basa en un conjunto de técnicas supervisadas que consiguen segmentar el espacio entre predictores en regiones mucho más simples y en las cuales sencillo manejar las interacciones. Su principal ventaja con respecto a otros métodos predictivos es que su estructura se asemeja a un árbol, facilitando su comprensión, tanto es así que no es necesario tener grandes conocimientos para poder interpretar los resultados. Además, su representación permite identificar de forma rápida aquellas variables que son más relevantes dentro del modelo. (Rodrigo J. A., Estadística y Machine Learning con R, 2017).

A continuación, generamos una estructura de árbol para nuestros datos de la actividad 1 en la que incluimos todas las variables disponibles. Esto nos permite entender mejor las principales características de este método predictivo.

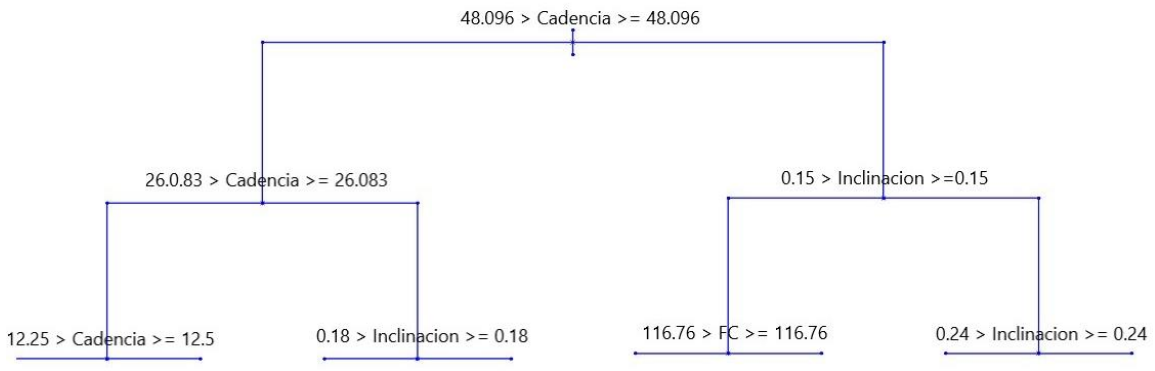

Figura 4: *Ejemplo de Árbol de Regresión para la actividad 1.*

- **Regiones:** A las regiones R se conoce como terminaciones o *terminal nodes*. Son puntos en los que el espacio de los predictores sufre una división, esto se denomina como *internal nodes.* Y a los segmentos que conectan dos nodos se conoce como ramas o *branches.*
    - R1 = {Cadencia < 48.096}: Potencia generada con una cadencia inferior a 48 W.
    - R2= {Cadencia ≥ 48.096 e Inclinación < 0.15}: Potencia generada cuando se produce una cadencia superior a 48 W con una inclinación inferior a 0.15°.
    - R3= {Cadencia ≥ 48.096 e Inclinación ≥ 0.15}: Potencia generada cuando tenemos una cadencia superior a 48 W con una inclinación superior a 0.15°.
    - R4= {Inclinación < 0.15 y FC < 116.76}: Potencia generada cuando se produce una cadencia superior a 48 W, Inclinación es inferior a 0.15° y la FC es inferior a 116.76 PPM.

- **Interpretación:** Luego de esta estructura se puede deducir que la variable más importante a la hora de determinar la potencia generada en la pedaleada es la Cadencia. En un segundo estrato encontramos a la Inclinación y la Frecuencia Cardiaca. Debemos recurrir hasta el sexto extracto para encontrar las demás variables.

Para este modelo la implementación se hará usando la herramienta Regresión Learner de Matlab. Esta aplicación permite la selección de diferentes algoritmos para entrenar y validar modelos de regresión. Una vez entrenados diferentes modelos, permite comparar de una manera muy ordenada y sencilla los distintos errores que presenta cada uno. Permite también realizar un aprendizaje automático supervisado introduciendo los datos de entrada o predictores y las respuestas conocidas a este conjunto de datos. (MathWorks, Aplicación de aprendizaje de regresión, s. f.).

En concreto hemos utilizado un tipo de modelo basado en los métodos de *ensemble*, cuya idea básica es combina múltiples modelos predictivos para lograr un equilibrio entre bios y varianza. ¿Qué entendemos por bios y varianza? Decimos que un modelo de pocas ramificaciones tiene alto bios y poca varianza, y si el árbol es grande, se ajusta mucho a los datos de entrenamiento, entonces decimos que tiene muy poco bios pero alta varianza.

3.2.3. Redes Neuronales

Como tercer modelo, elegimos una red neuronal. Debido a su construcción son capaces de adaptarse y aprender de las experiencias. Además, son capaces de autoorganizarse de manera que son tolerantes a fallos, por lo que es una buena alternativa a métodos de regresión clásicos.

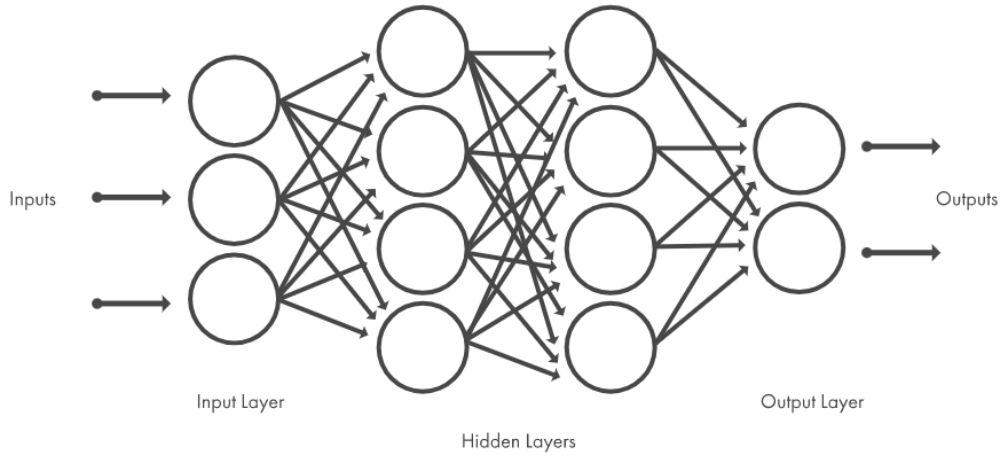

Figura 5: *Ejemplo de las capas de una red neuronal.*

Una red neuronal no es más que una estructura cuyas capas internas están interconectadas de forma similar a las neuronas del cerebro. Permite que la red pueda aprender, de manera que se puede entrenar para reconocimiento de patrones, clasificación o pronósticos. Combina diversas capas de procesamiento y utiliza elementos simples consta de una capa de entrada, una o varias capas ocultas y una capa de salida. Las capas están interconectadas mediante nodos y utilizan la salida de la capa anterior como entrada.

Las neuronas se organizan por niveles o capas representado en la Figura 5. En ella podemos describir tres tipos de capas:

- Capa de entrada: En ella se recibe directamente la información de entrada de una fuente externa (variables de entrada).
- Capas ocultas: Son propias de la red y no tienen contacto con el exterior. La combinación y la interconexión de estas capas es lo que permite obtener diferentes técnicas de las redes neuronales.
- Capa de salida: En esta capa se realiza la transferencia de información hacia el exterior (variables de salida).

Existen diversas técnicas utilizas hoy en día, habituales en *machine learning*, diseñadas específicas para cada aplicación. Algunas de ellas son el aprendizaje supervisado y no supervisado, clasificación, la regresión, reconocimiento de patrones y *clustering*. Nos centraremos en la técnica de regresión, pues describe bastante bien la relación entre una variable de salida y una o varias variables predictoras de entrada.

Nuestra red neuronal tiene dos capas retroalimentadas con una función de activación denominada *sigmoid*. Una función de activación se encarga de devolver una salida a partir de un valor de entrada, en este caso transforma los valores introducidos a valores 0-1, es decir, los valores altos tienden de forma asintótica hacia 1 y los valores bajos tienden hacia 0.

Para generar los modelos se ha usado la aplicación *Neural Net Fitting* de Matlab y se entrenará con un algoritmo de propagación llamado *Levenberg-Marquardt*. Es una técnica

estándar basada en técnicas descenso de gradiente o GN y el de Gauss-Newton. Es muy utilizada para solventar problemas no lineales de mínimos cuadrados. Se basa en el cálculo del jacobiano del modelo con respecto a los parámetros, y utilizando esta técnica se busca un mínimo local, es decir, buscamos una convergencia hacia el error mínimo. Desde la aplicación podemos exportar el modelo a la siguiente función:

[trainedNet, tr] = train(net, X, T, Xi, Ai, EW)

Donde *net* contiene las capas ocultas del modelo y el algoritmo con el que queremos entrenar, X es la matriz o tabla de datos donde no se incluye la salida, Y es la potencia real, Xi y Ai son condiciones iniciales que podemos pasar al modelo y EW son los pesos del error especificado como array. (MathWorks, Redes neuronales, s.f.).

3.2.4. ANFIS

El siguiente modelo elegido es un sistema adaptativo de interferencia neuro-difusa, ANFIS. Elegimos este modelo como una alternativa los métodos clásicos de redes neuronales por su gran capacidad adaptativa a diferentes áreas y porque utiliza reglas de aprendizaje automáticas.

Definimos las redes adaptativas como aquellas estructuras compuestas de varias capas con un conjunto de nodos conectados a través de enlaces. Cada nodo es una unidad de procesamiento que realiza una función sobre la señal que recibe. Los enlaces indican la dirección de flujo en la que avanza la señal. Son adaptativos puesto que dependen de un conjunto de parámetros o valores para generar una salida, estos parámetros cambian con la señal recibida.

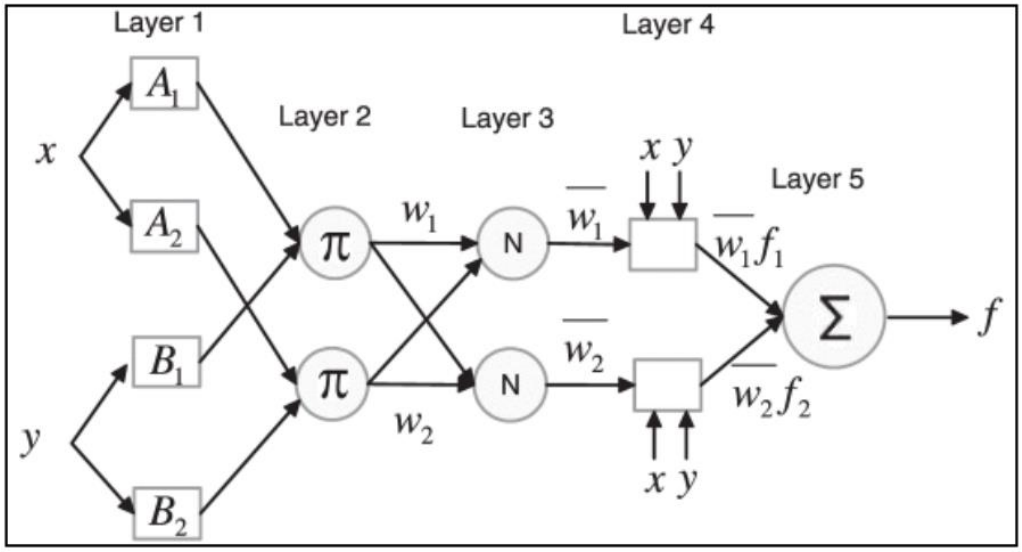

Figura 6: *Ejemplo de la estructura del modelo ANFIS*

El sistema adaptativo de interferencia fue desarrollado por J.R Jang en 1993 y es una red adaptativa cuyo fundamento radica en el sistema de interferencia *Takagi-Sugeno.* Es

decir, aprendizaje automático basado en reglas. Su filosofía consiste en predecir la salida a partir de una serie de puntos que se construyen de las entradas. Puesto que es una red adaptativa, su estructura consiste en una red *feedforward* multicapa, donde las funciones miembros están optimizadas usando una red neuronal. Utiliza dos tipos de algoritmos de aprendizaje: algoritmo de propagación y el algoritmo híbrido. (W & S, 2018).

Su estructura típica se constituye de las capas de la Figura 6:

- Layer 1: Capa difusa
  Cada nodo de la capa difusa representa un grado de pertenencia a la función miembro de la entrada ($\mu A_i(x)$, donde x es la entrada y μA_i es la salida.
- Layer 2: Capa producto
  En esta capa los nodos son no-adaptativos. Todos los nodos en esta capa se multiplican por la señal de entrada. Es decir, dos funciones miembros se multiplican obteniéndose un resultado w_1

$$w_1 = \mu A_1(x) \times \mu A_1(x)$$

- Layer 3: Capa de normalización
  En esta capa se calcula el peso de cada producto anterior para generar una función normalizada.

$$\overline{w1} = \frac{w1}{w1 + w2}, i$$

- Capa 4: Capa Adaptativa
  En esta capa los nodos son adaptativos. Cada nodo en esta capa se calcula mediante la función:

$$\overline{wi} fi = \overline{wi}(p_i x + q_i y + r_i)$$

  Donde p_i, q_i r_i son el conjunto de parámetros,
  x e y son las entradas del modelo,
  $(w_i)^-$es la función normalizada,
  y f_i es una salida del modelo.

- Capa 5: Capa salida
  Encontramos un único nodo no-adaptativo que calcula la salida total del sistema.

$$f(x) = \sum_i \overline{wi} fi = \sum_i \frac{wifi}{\overline{wi}}$$

La principal ventaja de este modelo es que utiliza métodos de predicción típicos de una red neuronal, por tanto, su estrategia de entrenamiento se basa en la configuración de sus parámetros y de la función de pertenencia.

Al igual que en el método anterior, también usaremos una aplicación implementada en Matlab para entrenar nuestro modelo. La herramienta se denomina *Neuro–fuzzy designer* y admite como datos de entrada archivos .dat o .mat, por lo que en el procesado de datos debemos crear estas extensiones para poder usar la aplicación. También tenemos la opción poder validar y testear los entrenamientos con datos diferentes o dividiendo los datos iniciales. Los modelos se han creado con un algoritmo de tipo *Substractive clustering* con un total de 30 épocas.

4. Desarrollo experimental

4.1. Procesado de datos

Si bien el ciclo computador nos da toda la información necesaria, los datos en crudo no nos sirven de mucho, por ello es necesario realizar antes un procesado. ¿Cuál es el objetivo del procesado de datos?

En primer lugar, debemos generar una matriz con las variables especialmente relevantes y que sean descriptivos para nuestros modelos. Anteriormente comentamos que los datos del sensor se han convertido a un archivo Excel para su fácil manejo. Con la herramienta Matlab estos archivos se han introducido en una tabla con cada una de las 8 actividades que tenemos disponibles. Contienen demasiada información, por lo que nosotros solo nos quedaremos con las columnas que nos interesan: cadencia, FC, potencia, distancia, altitud y velocidad.

Estos datos tampoco son de gran utilizad porque predictores como la altitud apenas es relevante, pues no suele haber grandes cambios de altura durante un entrenamiento. Por tanto, parece más sensato tomar la diferencia entre alturas entre dos instantes a la que llamaremos Inclinación, y será el predictor que nos indicará cuándo un ciclista está en una pendiente positiva o negativa y, por tanto, cuándo un ciclista realiza mayor o menos esfuerzo de pedaleo.

Ahora tenemos una matriz con seis variables: cadencia, FC, potencia, distancia, inclinación y velocidad, pero ¿realmente son necesario seis predictores? Si lo que queremos es asegurarnos de tomar aquellas que son más útiles recurrimos al estudio de la relación entre variables y la salida, en concreto tomamos el Coeficiente de Pearson (R). Este parámetro permite conocer el grado de variación conjunta entre dos variables

aleatorias y varían entre -1 y +1. Siendo +1 una correlación positiva perfecta y un cero una correlación nula, si la correlación es -1 decimos que posee correlación negativa perfecta. Además, debemos ser capaces de argumentar cuando una variable es o no es relevante y para ello se toma la siguiente tabla.

0	Correlación nula
0,01 a 0,19	Correlación positiva muy baja
0,2 a 0,39	Correlación positiva baja
0,4 a 0,69	Correlación positiva moderada
0,7 a 0,89	Correlación positiva alta
0,9 a 0,99	Correlación positiva muy alta
1	Correlación positiva grande y perfecta

Tabla 3: *Rangos de correlación.*

Si ahora generamos la correlación de las diferentes variables del estudio con relación a la salida, observamos que la variable altura y distancia tienen poca correlación con la potencia generada según los intervalos de la tabla 4, en consecuencia, nuestros datos finales incluirán: Velocidad, Inclinación, FC, Cadencia y Potencia como se puede observar en la tabla 5, donde se incluye una muestra de 15, correspondientes a la actividad 2.

Variable	Potencia
Velocidad	-0.22
Inclinación	0.71
FC	0.78
Cadencia	0.81
Distancia	-0.03
Altura	0.19

Tabla 4: *Correlación de las variables con la salida.*

Tiempo	Velocidad	Inclinación	FC	Cadencia	Potencia
1	5.00	0	92	89	145
2	9.87	-0.20	95	89	28
3	0	0	95	89	28
4	4.77	0	95	40	30
5	4.89	0	96	89	147
6	9.98	0	99	91	127
7	0	0	97	93	122
8	10.78	0	98	93	124
9	0	0	100	95	125
10	5.53	0	99	95	153
11	5.64	0.20	101	100	153
12	11.59	0	100	103	120
13	0	0.20	101	88	169

14	12.17	0	100	88	141
15	0	0	100	88	127

Tabla 5: *Ejemplo de las primeras 15 muestras de la actividad 2, sin aplicar ningún filtro.*

Además, necesitamos indicar de alguna manera situaciones que pueden ocurrir en un entrenamiento o carrea. Por ejemplo, que exista una pendiente negativa y el ciclista deje de pedalear o que simplemente se detenga en un semáforo. En ambos casos el ciclocomputador está recogiendo datos de frecuencia cardiaca, pero los datos de cadencia son nulos, como resultado la potencia generada también debe ser nula y en nuestros modelos debe estar contemplada esta situación.

La solución es implementar una función para cuando la Cadencia sea nula, la potencia sea cero, pero ¿Y si el ciclista se encuentra en una pendiente y deja de rodar los pedales? Esta situación no se puede contemplar en los modelos, por lo que sencillamente no se puede modelar.

Por último, es común que los valores de potencia que entrega el ciclo computador estén filtrados, es decir, nos de la potencia media de las últimas cinco o diez muestras. Esto es habitual puesto que un ciclista no necesita saber su dato de potencia cada segundo, tendría una variación muy amplia. Es más indicado que nos arroje la potencia media de un intervalo en concreto. Para simular este comportamiento realizaremos un filtrado a los datos de entrada para que de esta forma el entrenamiento de los modelos sea lo más ajustado posible a una curva suavizada, esto permite que los datos de potencia que nos entregue los modelos ya estén filtrados.

Tiempo	Velocidad	Inclinación	FC	Cadencia	Potencia
1	5.00	0	92	89	145
2	9.86	-0.2	95	89	28
3	0	0	95	89	28
4	4.77	0	95	40	30
5	4.88	0	96	89	147
6	9.9	0	99	91	127
7	0	0	97	93	122
8	10.78	0	98	93	124
9	0	0	100	95	125
10	5.08	-0.02	96.60	86.30	102.90
11	5.10	-0.002	97.26	86.53	98.690
12	5.21	-0.002	97.38	86.58	102.46
13	4.58	0.02	97.72	84.84	114.80
14	5.78	-0.000661	97.89	89.32	120.48
15	4.65	-0.000728	98.08	89.35	116.43

Tabla 6: *Ejemplo de las 15 primeras muestras de la actividad 2, aplicando el filtro de 10.*

En la tabla 5 se puede observar los 15 primeros datos correspondientes a la actividad 2, donde no se ha aplicado el filtro con la media de las diez últimas muestras, al contrario que en la tabla 6 donde si se ha aplicado este filtro. A partir de estos datos podemos representar una comparativa de la diferencia entre generar un modelo con datos filtrados y sin filtrar se puede observar en la figura 7, donde la variación de la Cadencia es mucho más pronunciada y, por tanto, generando un modelo que no se ajuste a curva de entrenamiento.

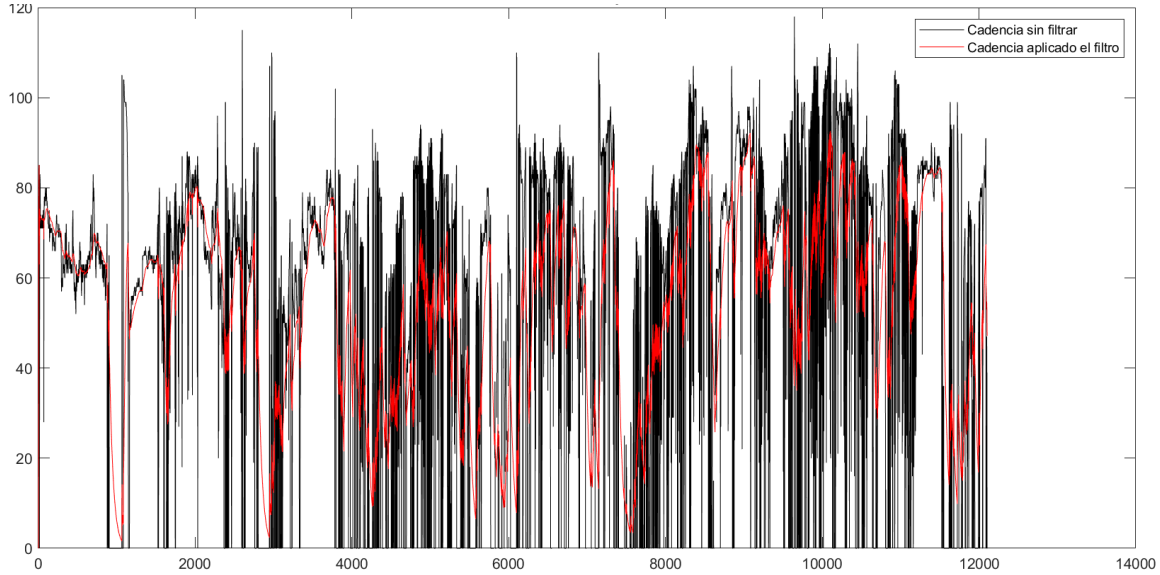

Figura 7: *Cadencia, actividad 2, sin aplicar el filtro y con el filtro aplicado.*

De igual forma, la figura 8 representa los datos de Potencia en watts que nos proporciona el sensor. Si aplicamos el filtro podemos observar que eliminamos gran parte de los picos de ruido que es causado por la toma de muestras a cada segundo. De este modo, a nuestros modelos les será mucho más fácil ajustarse a la curva con el filtro aplicado.

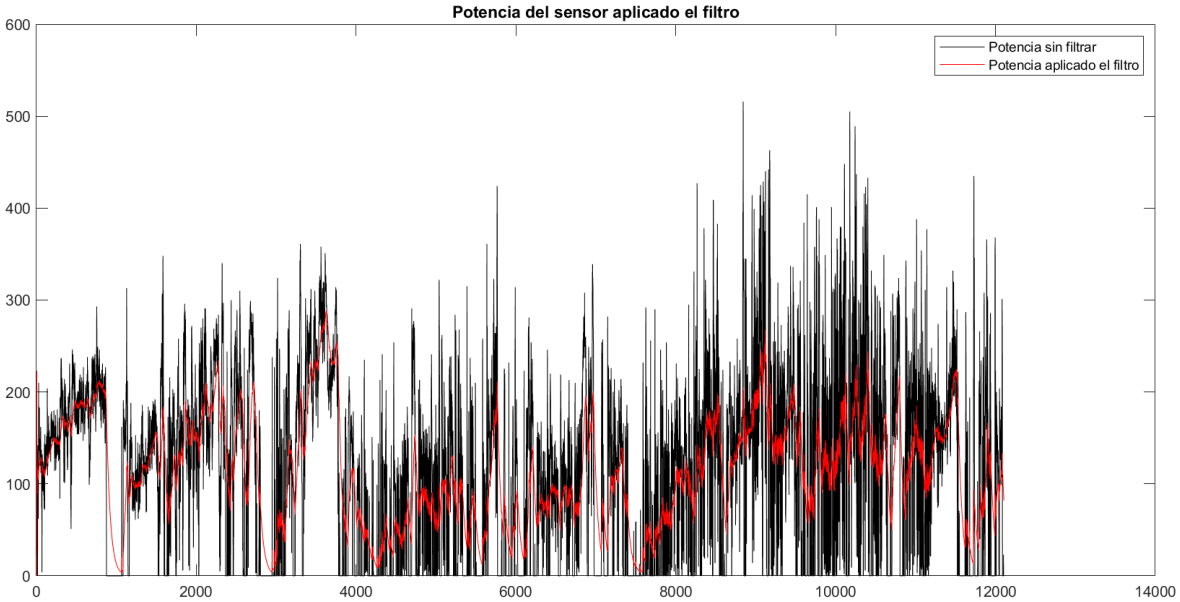

Figura 8: *Potencia, actividad 2, aplicado el filtro y sin el filtro.*

4.2. Implementación de los modelos

Como ya se pudo observar en la tabla 4, las variables que más correlación tienen con la Potencia son: velocidad, inclinación, FC y cadencia. A partir de estos datos generamos un script de Matlab para cada modelo y observamos la curva de entrenamiento utilizando las diferentes aplicaciones y funciones de las que dispone esta herramienta. Además, debemos mencionar que en este caso se ha utilizado los datos de la actividad 2 para realizar el entrenamiento de cada modelo.

4.2.1. Regresión no lineal

Para entrenar el modelo primero cargamos la matriz de datos que previamente han sido filtrados y mediante la función *fitnlm* procedemos a generar los modelos. Puesto que realmente no sabemos la ecuación a la que se ajustan los datos, nuestra metodología será de prueba y error hasta dar con una ecuación que nos proporcione los mejores resultados.

- Primera propuesta:

  *Potencia ~ b1 + b2*Velocidad+ b3*sin (Inclinacion) + b4*FC + cos (Cadencia)*
  Estimated Coefficients:

	Estimate	*SE*	*tStat*	*pValue*
b1	*-128.58*	*2.6731*	*-48.102*	*0*
b2	*15.263*	*0.27774*	*54.956*	*0*
b3	*387.35*	*3.8383*	*100.92*	*0*
b4	*1.1141*	*0.02643*	*42.153*	*0*

  Root Mean Squared Error: 29.6
  R-Squared: 0.662

  Con la primera propuesta obtenemos un R=0.66, es decir que el modelo es capaz de explicar el 66% de la variabilidad observada de la Potencia. Aun así, es un porcentaje bastante bajo, por lo que seguimos probando con otras combinaciones.

- Segunda propuesta:

  *Potencia ~ b1 + b2*Velocidad + b3*Inclinacion + b4*FC + Cadencia*

  Estimated Coefficients:

	Estimate	*SE*	*tStat*	*pValue*
b1	*-171.52*	*2.2975*	*-74.657*	*0*
b2	*-8.5806*	*0.1546*	*-55.503*	*0*
b3	*37.679*	*5.5212*	*6.8244*	*9.1852e-12*

b4	*2.2522*	*0.01838*	*122.53*	*0*

Root Mean Squared Error: 26.2
R-Squared: 0.736

Con la segunda propuesta logramos disminuir el error hasta un 26.2 W, pero seguimos combinando variables con el fin de obtener el mejor modelo posible.

- Tercera propuesta:

*Potencia ~ b1*Inclinación + b2*Cadencia + b3*(FC*FC) + b4*Velocidad*
Estimated Coefficients:

	Estimate	*SE*	*tStat*	*pValue*
b1	*82.802*	*3.3878*	*24.441*	*2.7264e-129*
b2	*1.5279*	*0.013165*	*116.06*	*0*
b3	*0.0046893*	*6.8773e-05*	*68.185*	*0*
b4	*-6.2647*	*0.22391*	*-27.979*	*9.2035e-168*

Root Mean Squared Error: 21.65
R-Squared: 0.82

Tras diferentes pruebas, hemos llegado a la conclusión de este modelo es el que mejores resultados obtiene. Evidentemente, habrá una relación entre predictores que mejor se ajuste a los datos, pero ese no es objetivo de este trabajo, por lo que damos por válido este modelo.

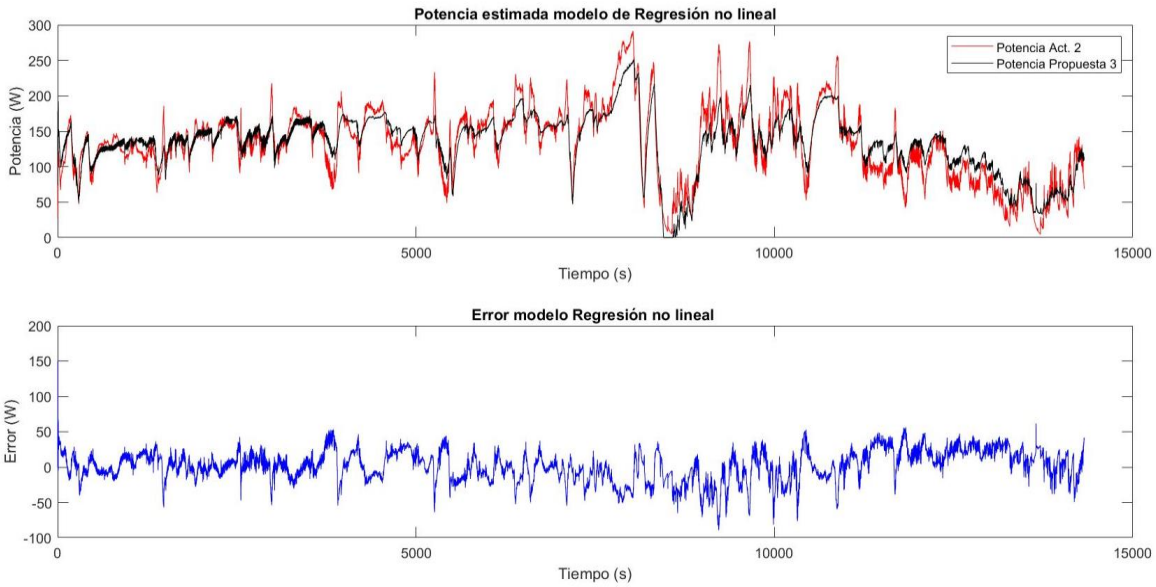

Figura 9: *Potencia predicha para el modelo de Regresión no lineal usando cuatro variables y el error acumulado.*

Con un ajuste de bondad del 82% conseguimos los resultados que se observan en la figura 9. Representamos la respuesta del modelo frente a la salida real. Además, en la gráfica inferior representamos el error que se comete cada instante.

4.2.2. Árbol de Regresión

Al igual que para el modelo anterior, también generamos un script en Matlab al que cargamos los datos de la actividad 2 para generar el modelo. Hacemos uso de la aplicación *Regresión Learner* al que introducimos la matriz de datos. Por defecto encontramos una partición de cinco para la validación cruzada, esto es para proteger la predicción contra el efecto de *overfitting*.

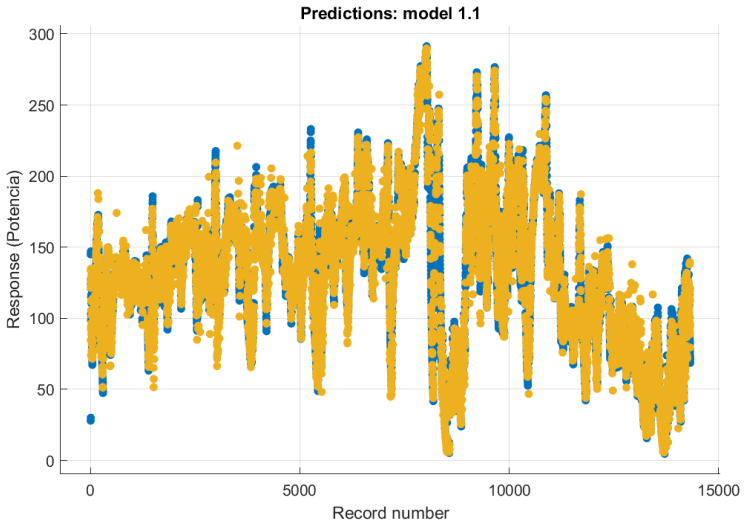

Figura 10: *Respuesta del modelo de Árbol de Regresión utilizando las cuatro variables.*

En la figura 10 podemos observar la respuesta que nos ofrece este modelo. Con un RMSE= 10.56 W, R^2=0.96 y MAD = 3.4 W en color azul tenemos datos de potencia en vatios para la respuesta verdadera y en amarillo para la estimada por este modelo. Además, la aplicación nos permite observar el comportamiento de la potencia real frente a la del modelo.

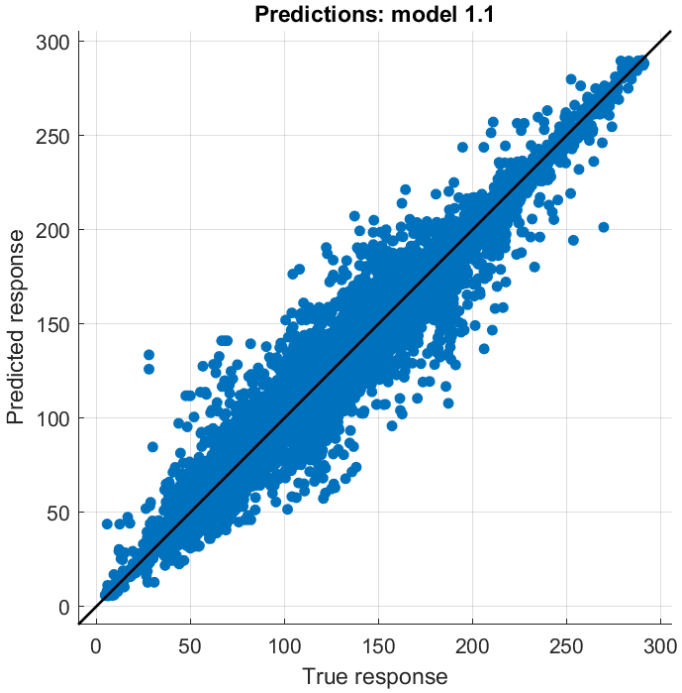

Figura 11: *Relación entre potencia real y potencia predicha por el modelo usando las cuatro variables.*

La respuesta que se observa en la figura 11 sigue una tendencia lineal siendo una predicción perfecta la diagonal. Como hemos filtrado los datos de entrada hemos eliminado los errores debido a los picos de potencia y evitamos la acumulación del error cuando el ciclista deja de pedalear, es decir, cuando la potencia del sensor es cero.

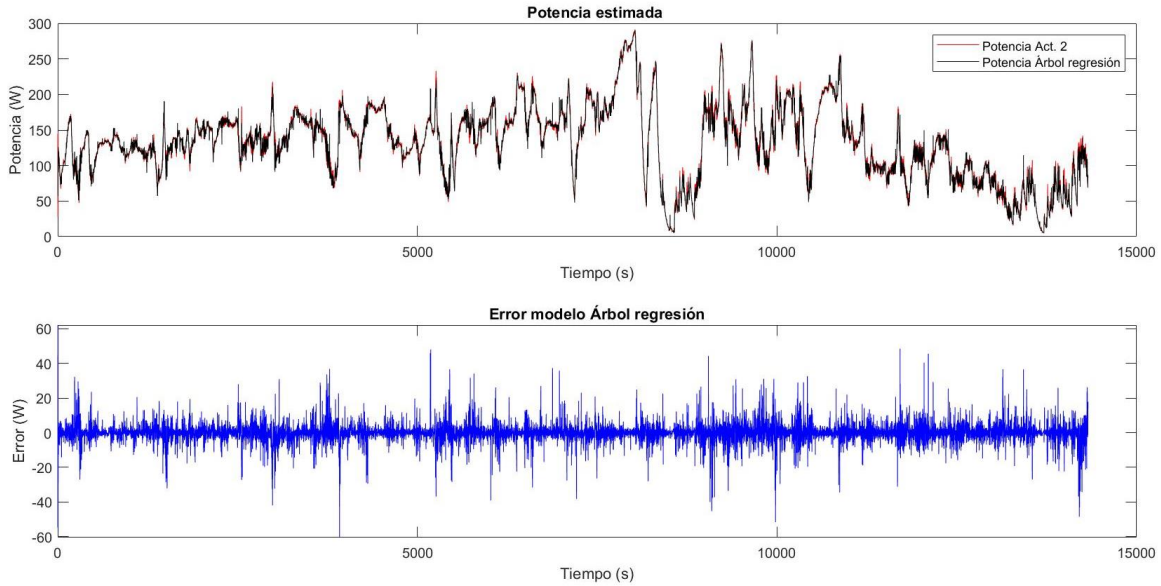

Figura 12: *Respuesta del modelo de Árbol de Regresión con cuatro variables y el error asociado.*

Hacemos la misma representación que en el modelo anterior. La potencia predicha superpuesta a la potencia real y en gráfico inferior el error acumulado. Si nos figamos ambas curvas están casi solapadas, algo que indica que el ajuste es muy bueno, pero que deberemos comprobar a lo largo del desarrollo de este trabajo.

4.2.3. Redes Neuronales

Para el entrenamiento del modelo usamos la función de Matlab anteriormente descrita, pero no hacemos uso de las condiciones iniciales ni del peso de los errores.

[trainedNet,tr] = train(net, Tbl2, Pot.real)

Donde Net contiene un total de 30 capas ocultas y el algoritmo de *Levenberg-Marquardt backpropagation*, pues tras varias pruebas realizadas se ha concluido que es el que mejor resultados obtiene. Al igual que para casos anteriores usamos la actividad 2 para generar los resultados.

Como ya hemos hecho con los modelos anteriores, representamos la potencia y el error asociado a la potencia predicha por el modelo. En este caso la aplicación nos permite visualizar el histograma de errores que se puede observar en la figura 14, así como la potencia de la actividad frente a la potencia predicha por el modelo de red neuronal. Los valores de los parámetros en este caso son RMSE= 12.40 W, MAD=9.22 W con un R^2=0.96.

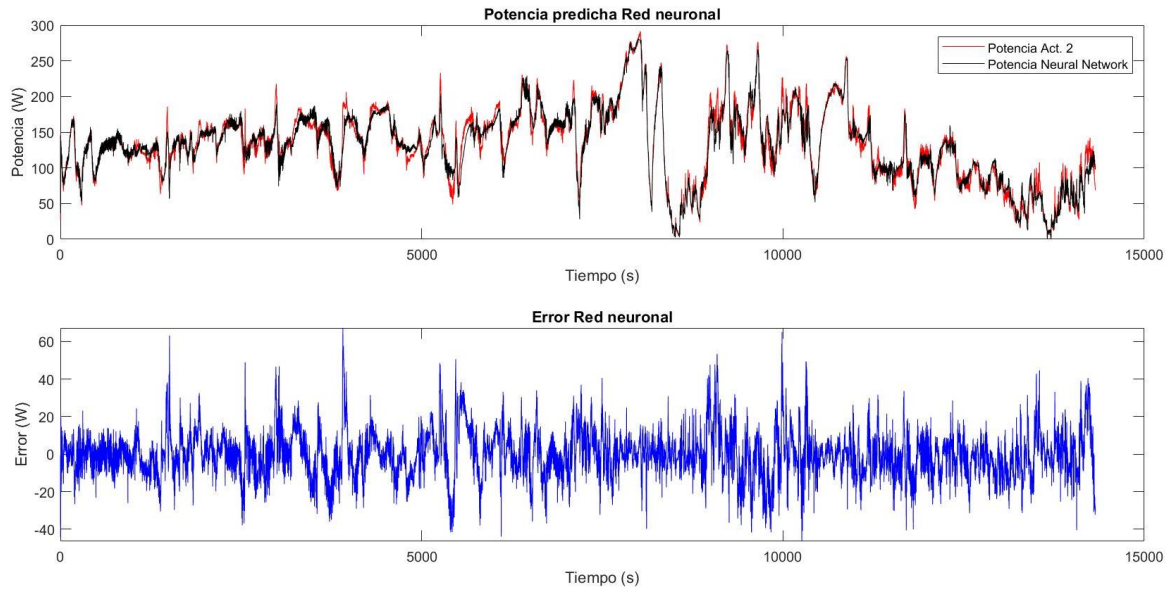

Figura 13: *Respuesta del modelo Red Neuronal usando las cuatro variables y error asociado.*

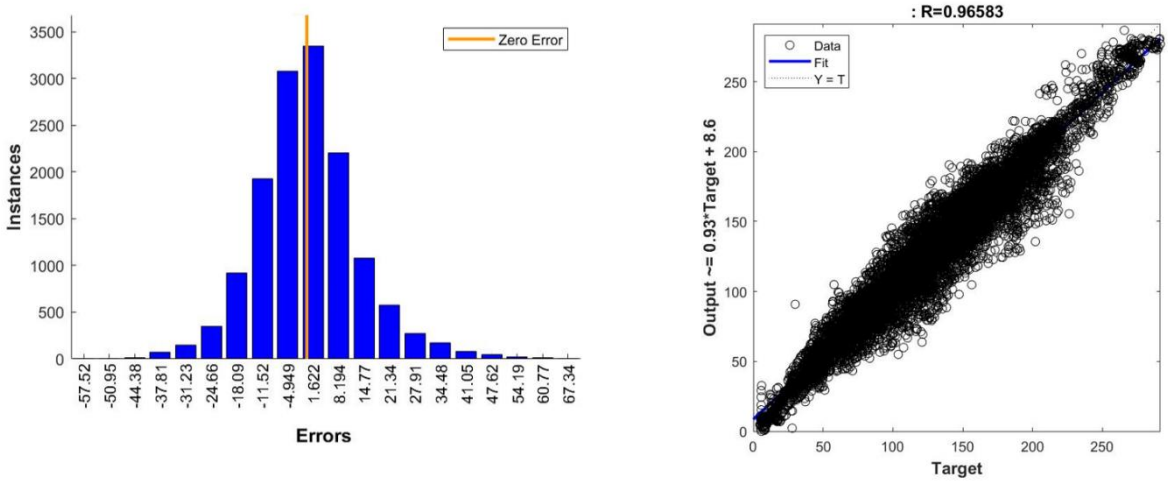

Figura 14: *Acumulación del error para el modelo de Red Neuronal usando las cuatro variables*

4.2.4. ANFIS

Para el entrenamiento del modelo ANFIS usaremos la aplicación integrada en Matlab denominada *Neuro-Fuzzy designer*. Necesitamos una extensión de archivos .dat y procedemos a particionar los datos para el testeo y/o validación. Además, elegimos 30 épocas para el entrenamiento pues tras varias pruebas hemos considerado que son suficientes para este modelo.

Generando el entrenamiento para este modelo se obtiene un RMSE de 17.39 W y un MAD de 14.19 W y al igual que antes podemos representar la salida con este modelo y generar el error asociado.

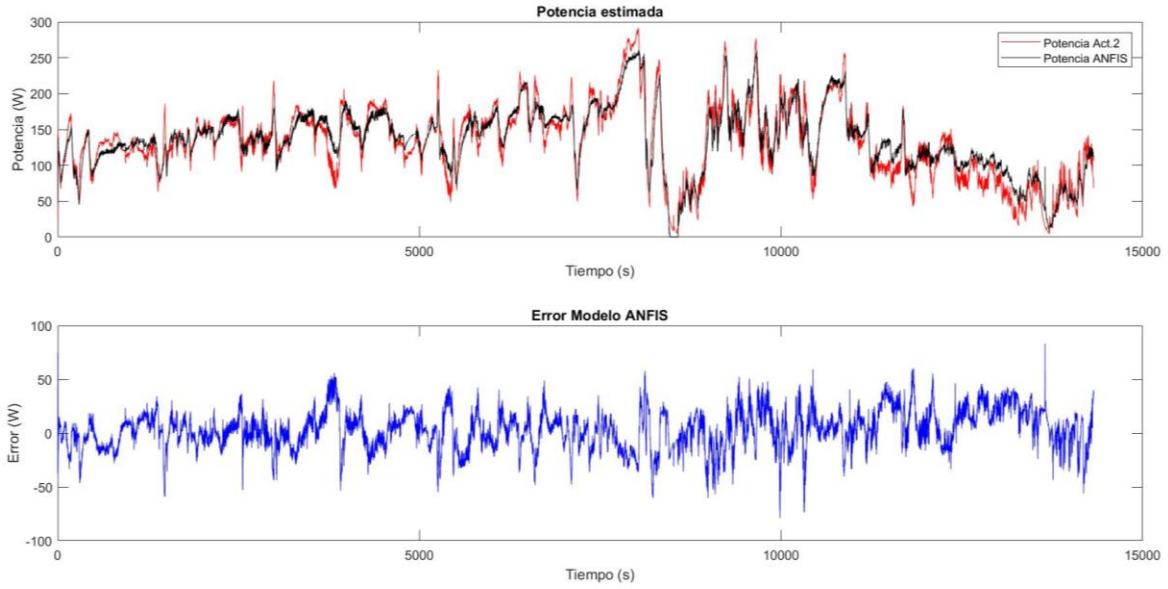

Figura 15: *Respuesta del modelo ANFIS utilizando las cuatro variables. La figura inferior representa la acumulación de error.*

5. Selección de variables y desarrollo final

Cuando se realiza predicciones con un modelo es importante tener en cuenta la relación que existe entre las variables de entrada y de salida. Decimos que un modelo tiene buena capacidad predictiva cuando se cumple que la relación entre los predictores y la variable dependiente es aproximadamente lineal y cuando el número de observaciones es mayor al número de predictores.

Los métodos por mínimos cuadrados tienden a considerar útiles todos los predictores, pero si el número de variables es muy alto puede afectar a la interpretabilidad del modelo, de tal forma que lo que perseguimos con la selección de variables es limitar el modelo solamente aquellos predictores que tengan una influencia notable sobre la respuesta o salida. Esto es especialmente útil si queremos evitar problemas de *underfitting* y *overfitting*, bastante comunes cuando se generan modelos predictivos como en los modelos de regresión que ya se comentó en el apartado 4.2.2.

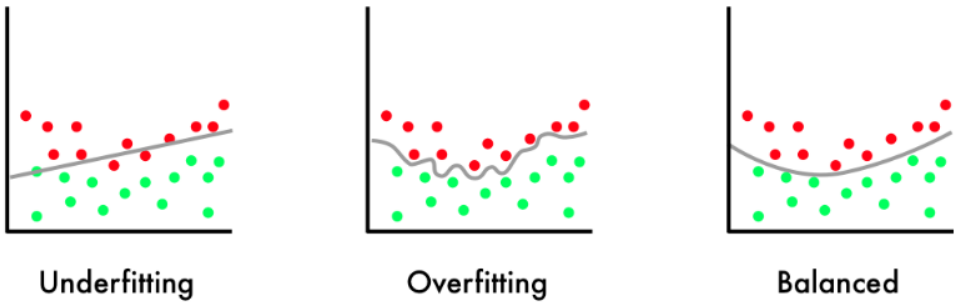

Figura 16: *Ejemplo de problemas cuando generamos modelos.*

- *Underfitting:* Son modelos que tienen escasa flexibilidad para ajustarse a los datos de entrada, de tal forma que generan errores de predicción altos. Una característica de estos modelos es tener una varianza baja, es decir, permanece similar cuando introducimos nuevos datos. En la primera imagen de la figura 16 se observa un ejemplo de este problema donde la respuesta corresponde a una ecuación lineal.

$$y = \beta_0 + \beta_1 x$$

- *Overfitting:* Generalmente es causado por modelos bastante potentes donde su ecuación se corresponde con un polinomio de alto grado. Se ajusta muy bien a los datos, pero tiene la desventaja de que al estar tan ajustado a los datos está capturando también el ruido. Al contrario que antes, ahora las varianzas son altas, lo cual indica que cualquier variación en los datos genera errores muy elevados.
  Un ejemplo de este problema se puede observar en la segunda imagen de la figura 16 donde su ecuación se ajusta un polinomio de grado n.

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_n x^n$$

En el apartado 4.2 hemos generado modelos con las cuatro variables disponibles, pero cabe la posibilidad de que los modelos tengan problemas de este estilo y cuando procedamos a introducir nuevos datos tengamos errores muy altos. Contemplada esta situación procedemos a emplear un método de selección de variables para identificar los más relevantes, pero antes, necesitamos una medida que permita compararlos y en este caso emplearemos los parámetros RMSE, MAD y R^2.

5.1. Método GMDH y método alternativo

Group Method of Data Handling, GMDH, es un algoritmo inductivo propuesto por Ivahnenkko en 1971 con el objetivo de identificar las relaciones no lineales entre variable de entrada y salida. Este algoritmo consiste en construir, a partir de un conjunto de variables explicativas y una variable respuesta, aquel polinomio que mejor ajusta los datos del problema *input-output* cumpliendo una serie de criterios previamente especificados.

Antes de iniciar el proceso los datos se dividen en dos grupos: el primero de entrenamiento, que se utiliza para calcular los parámetros de red, y el segundo para evaluar los resultados.

En una primera fase, se agrupa las variables de entrada por pareja, de forma que abarcamos todas las posibles combinaciones. La siguiente fase consiste en tomar cada pareja y generar un modelo, de tal forma que hay tantos modelos como combinaciones posibles. Por último, se evalúa cada modelo siguiendo un criterio conocido como el Criterio de Regularidad, CR. El valor más bajo de CR es el que tomamos como partida y para las siguientes iteraciones buscaremos que el CR sea siempre menor que la anterior. Si el CR de la siguiente iteración fuese mayor el criterio no se cumple y, por tanto, el proceso ha terminado. (Hernandez & Fernandez, 2012).

Este proceso indica que debemos generar para cada pareja de predictores un modelo distinto. Es decir, tenemos un total de seis modelos para iniciar la interacción, posteriormente debemos volver a crear nuevas combinaciones para generar nuevos modelos, pero siendo cuatro los métodos predictivos los planteamos al inicio, debemos buscar otra alternativa en la que no generemos demasiados modelos.

Una alternativa más sencilla sería estudiar la correlación entre predictores y la respuesta. Para iniciar la selección se genera el modelo con todas las variables disponibles, tomaremos aquella que tenga la correlación más alta con la salida. A partir de aquí generamos un modelo M_1. Si ahora obtenemos el error como $e_i = Y_{real} - \hat{Y}_{est,i}$, podemos estudiar la relación del error con los demás predictores mediante el coeficiente de Pearson, $R_{Vi,\ Ei}$, de tal forma que eliminamos el error de la primera variable seleccionada y nos quedamos con los residuos de las demás variables. Repetimos el proceso seleccionado

aquella variable con la correlación más alta, obteniendo un modelo M_2. El ciclo continúa hasta llegar al modelo M_n con todas las variables.

Parece que el método anterior es una buena alternativa que podemos seguir, pero tiene el inconveniente mencionado en el apartado 5, pues considera útiles todos los predictores y siempre será el mejor modelo aquel que contenga todas las variables. Esto podemos paliarlo añadiendo una condición de parada, es decir, una forma de que el proceso se detenga. Por tanto, se tendrá en cuenta las siguientes condiciones de parada:

1- La iteración se detiene cuando se obtenga la correlación y este nos indique añadir una variable que ya había sido introducida anteriormente.
2- El proceso se detiene cuando la correlación entre las variables y el error es nula o tiene valores cercanos a cero.

En cualquier caso, nos quedaremos con el último modelo generado, desechando la variable que no haya sido seleccionada.

5.2. Desarrollo del método alternativo

Se implementa el método anterior a cada uno de los modelos de predicción. Tenemos cuatro predictores con los que generamos la correlación de cada una con la potencia de salida. En la tabla 4 del apartado 4.1 se obtuvo que la variable cadencia resulta ser la que mayor correlación tiene con la potencia, por lo que será la primera variable que introduciremos a todos los modelos, posteriormente se tomará la correlación del error del modelo respecto a los predictores.

Para realizar el procedimiento se ha tomado la actividad 2, creado un script para cada método predictivo. Puesto que en el apartado 4.2 ya se obtuvo modelos utilizando todas las variables, tomamos los mismos resultados y seguimos el procedimiento. Al igual que antes, necesitamos parámetros para medir la calidad del modelo y en este caso utilizamos el MAD, RMSE y R^2 para generar una tabla comparativa donde introducimos los resultados en cada fase del método.

5.2.1. Regresión no lineal

Desarrollando la selección de variables para este método predictivo obtenemos la tabla 7. Podemos observar que tras generar el modelo M_1, la correlación de las variables con el error de este modelo nos indica que debemos introducir el predictor FC. Posteriormente introducimos la Inclinación, pero tras generar el modelo M_3, debemos volvemos a introducir la frecuencia cardiaca.

	Regresión no lineal	Velocidad	Inclinación	FC	Cadencia	R^2	RMSE	MAD
$R_{Pi, Pot}$	**M_4: Todas las variables**	-0.22	0.71	0.78	0.81	**0.82**	**21.65**	**17.16**
$R_{Vi, Ei}$	**M_1: Cadencia**	0.24	-0.45	-0.47	0.09	**0.62**	**31.45**	**22.87**
	M_2: Cadencia + FC	0.43	-0.48	0.02	0.23	**0.72**	**26.76**	**20.46**
	M_3: Cadencia + FC + Inclinación	-0.14	-0.10	-0.36	-0.26	**0.80**	**22.12**	**17.44**

Tabla 7: *Desarrollo de la selección de variables para en modelo de Regresión no lineal.*

Nuestro criterio de parada decía que si el modelo indicaba introducir una variable que ya estaba dentro no debemos seguir el procedimiento, quedándonos con el modelo actual. De esta forma, nuestro modelo final tendrá como predictores: cadencia, FC e inclinación con RMSE de 22.12 W, muy cercano al modelo con todas las variables. Como ya comentamos, los modelos de regresión tienden a ser mejores cuanto mayor sea el número de variables, pero el objetivo es que los modelos que se obtengan sean capaces de generalizar para todas las actividades que se introduzca.

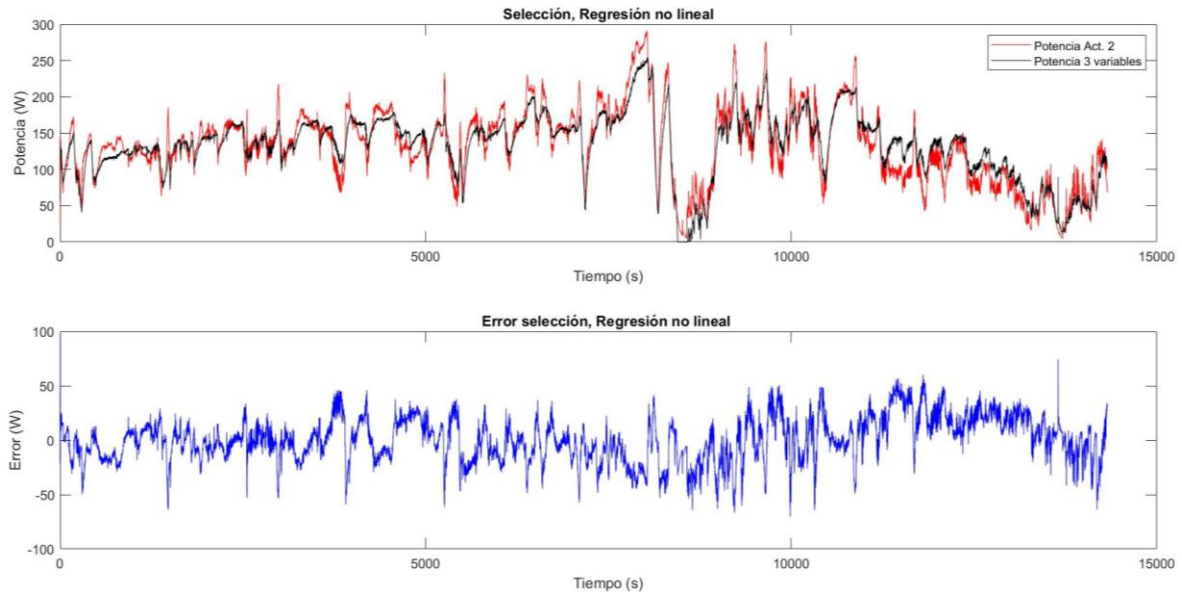

Figura 17: *Respuesta del modelo para el modelo de Regresión no lineal con las tres variables.*

5.2.2. Árbol de Regresión

	Árbol de regresión	Velocidad	Inclinación	FC	Cadencia	R^2	RMSE	MAD
R Pi, Pot	**M_4: Todas las variables**	-0.22	0.71	0.78	0.81	**0.96**	**5.61**	**3.40**
R Vi, Ei	**M_1: Cadencia**	0.16	-0.35	-0.39	0.00	**0.53**	**23.85**	**17.21**
	M_2: Cadencia + FC	0.16	-0.17	0.00	0.00	**0.86**	**11.08**	**6.43**
	M_3: Cadencia + FC + Inclinación	0.00	-0.01	0.00	0.00	**0.95**	**6.22**	**3.78**

Tabla 8: *Desarrollo de la selección de variables para el modelo de Árbol de Regresión.*

El procedimiento en este caso indica que debemos introducir las variables en el siguiente orden: cadencia, FC e inclinación, como primeros predictores, pero cuando generamos la correlación del último modelo observamos que son prácticamente nulos. Se cumple la segunda condición de parada, por lo que la variable Velocidad no se introduce al modelo y nos quedamos con las tres variables.

En la tabla 8 observamos que el valor de RMSE y MAD están muy cerca de los valores del modelo M_4 con todas las variables. Puesto que es un modelo de regresión, aquel modelo con el mayor número de predictores siempre obtendrá mejores resultados. Al igual que antes podemos representar la salida de este modelo junto con el error.

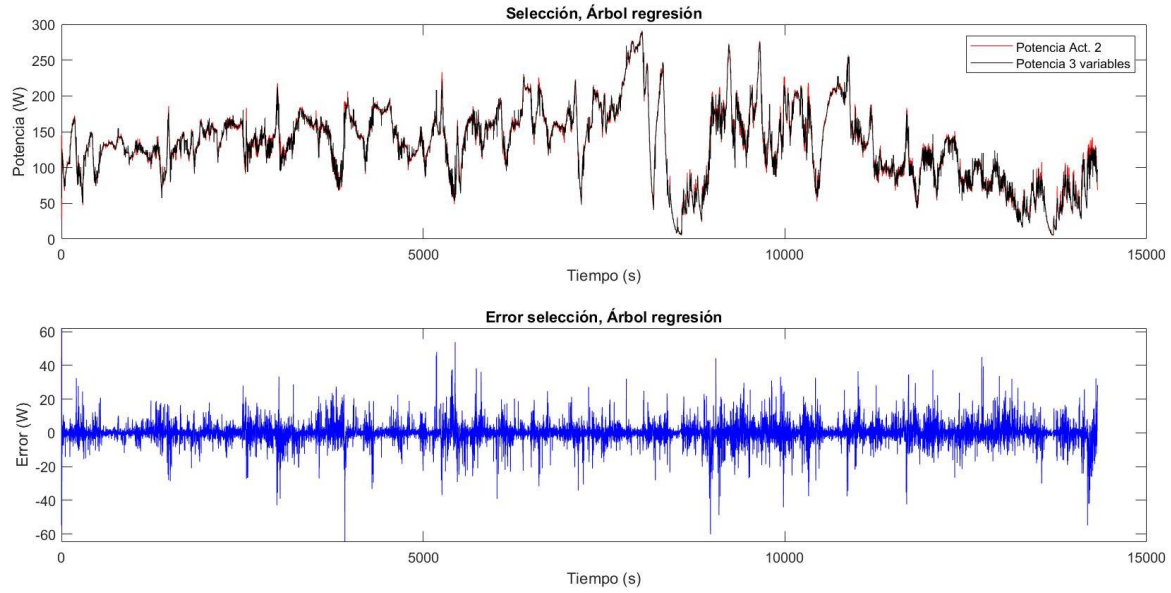

Figura 18: *Repuesta del modelo de Árbol de regresión usando las tres variables.*

5.2.3. Redes Neuronales

	Redes Neuronales	Velocidad	Inclinación	FC	Cadencia	R^2	RMSE	MAD
R Pi, Pot	**M4: Todas las variables**	-0.22	0.71	0.78	0.81	**0.96**	**13.37**	**9.89**
R Vi, Ei	**M1: Cadencia**	-0.18	0.44	0.49	0.01	**0.80**	**30.13**	**22.15**
	M2: Cadencia + FC	-0.36	0.37	0.00	0.00	**0.90**	**21.37**	**15.98**
	M3: Cadencia + FC + Inclinación	-0.01	0.00	0.00	0.01	**0.96**	**13.67**	**10.00**

Tabla 9: *Desarrollo de la selección de variables para el modelo de Redes Neuronales.*

Para este modelo predictivo encontramos resultados similares a los anteriores. El método de selección nos indica que con tres variables es suficiente. Lo parámetros para el modelo con tres variables son muy similares que si utilizáramos todas las variables e incluso igualamos en algunos casos.

En cualquier caso, no quedamos con el modelo de las tres variables y representamos la respuesta del modelo frente a la potencia real. De igual forma en la gráfica inferior tenemos el error acumulado para este modelo.

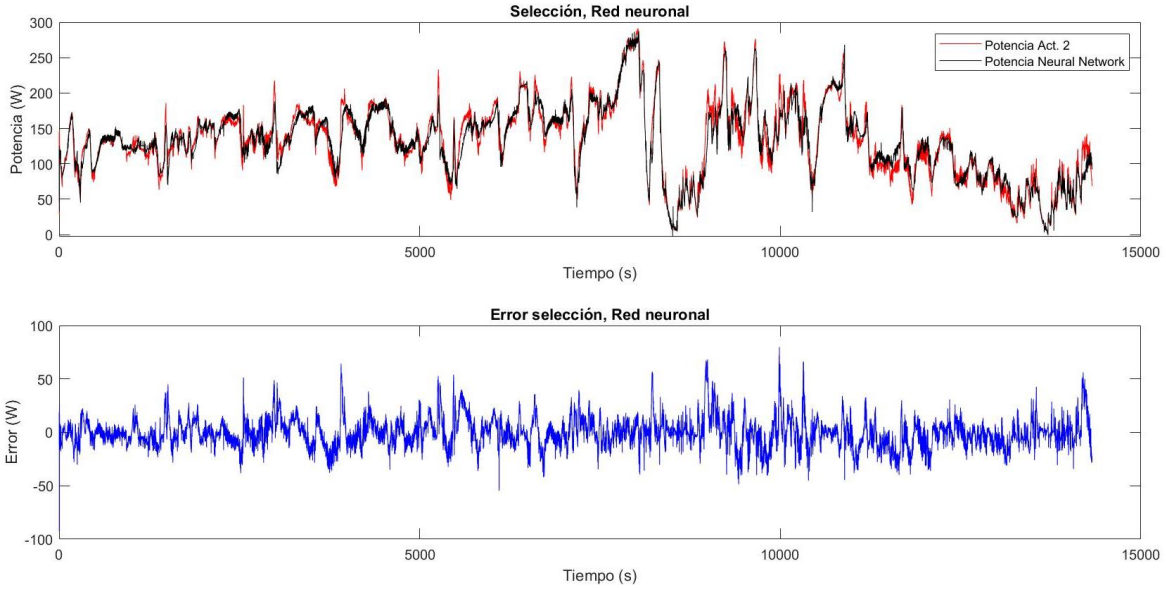

Figura 19: *Respuesta del modelo de Redes neuronales usando las tres variables.*

5.2.4. ANFIS

	ANFIS	Velocidad	Inclinación	FC	Cadencia	R^2	RMSE	MAD
R Pi, Pot	**M5: Todas las variables**	-0.22	0.71	0.78	0.81	**0.88**	**18.28**	**14.19**
R Vi, Ei	**M1: Cadencia**	0.15	-0.41	-0.53	-0.02	**0.62**	**31.42**	**23.15**
	M2: Cadencia + FC	0.44	-0.41	0.11	0.02	**0.76**	**25.40**	**19.78**
	M3: Cadencia + FC + Velocidad	0.04	-0.24	0.01	-0.17	**0.81**	**22.92**	**18.23**
	M4: Cadencia + FC + Inclinación	0.03	-0.04	0.02	-0.04	**0.88**	**17.80**	**13.68**

Tabla 10: *Desarrollo de la selección de variables para el modelo ANFIS.*

Para el modelo ANFIS observamos que, tras introducir la cadencia y la FC, la selección de variables nos indica que debemos incorporar la variable velocidad. Algo que en anteriores casos no había sucedido, pero al generar el modelo con esta variable nos indica que el siguiente sería la inclinación, es decir, todas las variables disponibles. Las condiciones de parada no se cumplen, por lo que podemos optar por elegir el modelo con todas las variables disponibles o elegir la inclinación como tercera variable para ver si realmente no se obtienen mejores resultados.

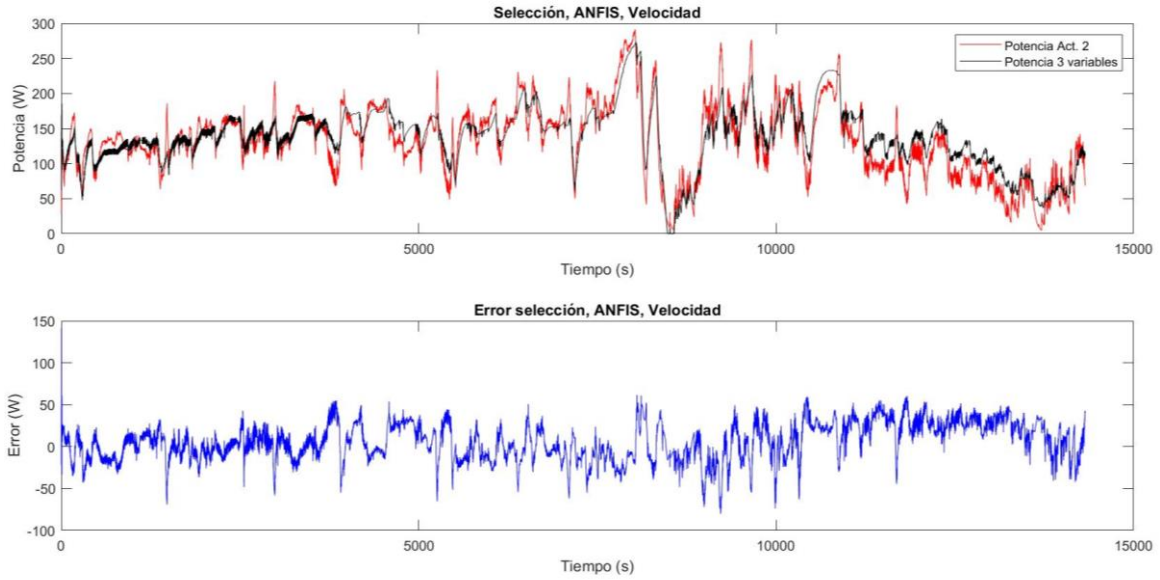

Figura 20: *Respuesta del modelo ANFIS si usamos la variable velocidad como tercera variable.*

La diferencia entre usar la velocidad o la inclinación como tercera variable, lo podemos observar en la tabla 10, donde los parámetros son notablemente mejores para el modelo que usa la variable inclinación. Además, en la figura 20 se observa la respuesta del modelo al usar la variable velocidad y vemos que se comete algo más de error que la repuesta de la 21 donde claramente al utilizar la inclinación, la repuesta esta mejor ajustada a la curva real. Por tanto, nuestro modelo final ANFIS contiene la cedencia, FC y la inclinación.

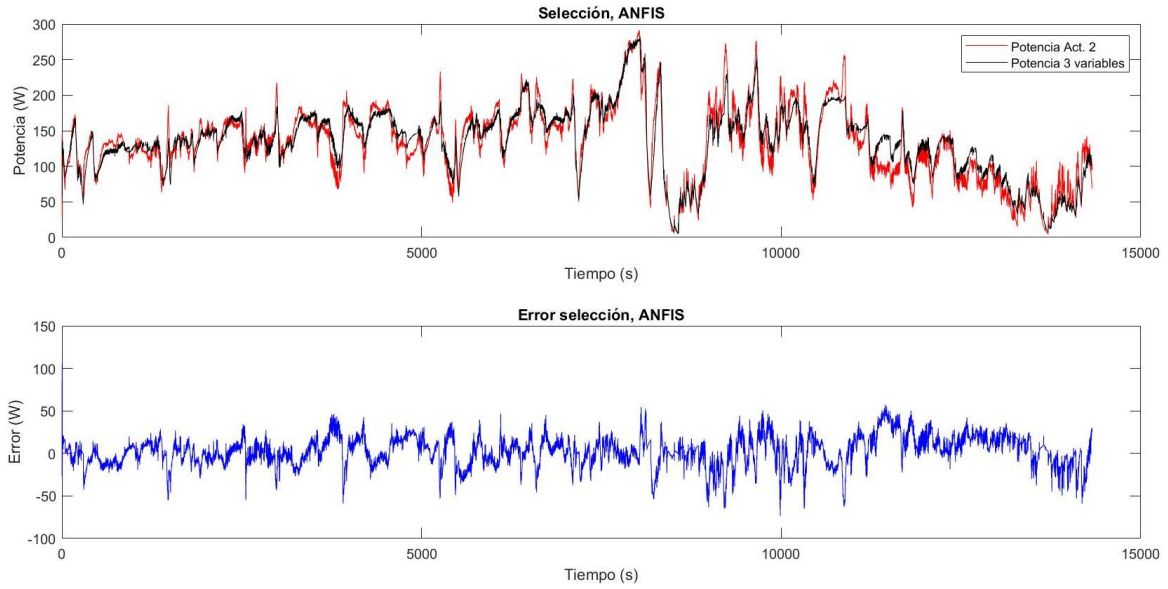

***Figura 21**: Respuesta del modelo ANFIS usando la inclinación como tercera variable.*

6. Resultados y discusión

Hemos podido comprobar que mediante la selección de variables se puede ajustar los modelos a los predictores necesarios, eliminando aquellos que no tienen demasiada influencia. El criterio de parada funcionó para todos los modelos, salvo para el modelo ANFIS, en el que tuvimos que valorar si utilizar todas las variables nos daba mejores resultados, pero compramos que no fue así. De esta forma, nos quedamos con las tres variables (inclinación, FC y cadencia) para todos los modelos predictivos.

Con los predictores se procede a testear los modelos con datos de las actividades que no ha visto aún. Creamos un script para cada método predictivo en el que se carga las ocho actividades disponibles y seguimos el siguiente procedimiento:

- Eliminamos de la matriz de todas las actividades la variable Velocidad.
- Generamos un modelo con la actividad 1, pero para la validación del modelo se toma todas actividades, incluida esta actividad.

- Para la validación se toma como parámetros el RMSE, MAD y MAE. De esta forma, generamos una tabla 8x8 para cada parámetro en el tenemos los modelos creados para cada actividad y validad con las demás actividades.
- Podemos representar la respuesta de una actividad aleatoria, validando con otra distinta para observar el ajuste de la curva de potencia.

6.1. Regresión no lineal

Regresión no lineal		Validación							
RMSE		Act.1	Act. 2	Act. 3	Act. 4	Act. 5	Act. 6	Act. 7	Act. 8
Construcción del modelo	Act. 1	21.07	24.12	21.98	25.23	24.42	26.01	22.49	28.12
	Act. 2	20.18	20.35	20.42	26.98	24.55	28.90	23.03	30.34
	Act. 3	19.61	22.37	19.66	26.65	23.26	29.55	25.03	28.16
	Act. 4	22.61	26.73	26.26	23.01	23.92	27.58	22.08	23.31
	Act. 5	21.17	26.45	22.82	24.10	23.58	26.42	22.15	24.97
	Act. 6	21.89	25.33	22.88	25.52	25.11	25.68	22.18	29.55
	Act. 7	21.69	24.70	24.20	23.89	24.49	26.27	20.91	27.68
	Act. 8	28.39	34.36	31.58	26.18	28.12	31.38	26.55	22.35
MAD									
Construcción del modelo	Act. 1	16.89	18.33	17.74	19.22	19.61	20.46	17.63	18.23
	Act. 2	15.62	16.19	16.86	17.89	18.04	21.78	16.74	16.33
	Act. 3	15.79	17.82	15.85	18.70	18.28	21.61	18.75	16.75
	Act. 4	16.67	17.10	17.91	18.16	19.10	21.22	16.71	17.04
	Act. 5	16.55	18.44	17.57	18.93	19.16	20.10	17.50	17.50
	Act. 6	17.27	19.34	18.80	19.69	19.92	19.86	17.23	19.30
	Act. 7	17.42	18.12	19.03	18.44	19.54	20.59	16.30	18.08
	Act. 8	16.96	17.82	17.59	18.92	19.64	21.00	17.43	17.74
MAE									
Construcción del modelo	Act. 1	16.95	19.12	18.16	19.51	19.56	20.45	17.98	21.99
	Act. 2	15.85	16.19	16.90	20.13	19.12	22.41	18.13	24.03
	Act. 3	15.80	17.81	15.85	20.76	18.44	21.37	20.27	22.15
	Act. 4	18.43	21.74	21.63	18.22	19.55	22.04	17.52	17.90
	Act. 5	17.06	20.71	18.65	18.90	19.22	20.65	17.47	19.76
	Act. 6	17.34	20.02	18.90	19.74	19.95	19.87	17.69	23.51
	Act. 7	17.42	19.77	20.09	18.27	19.52	20.69	16.30	21.60
	Act. 8	24.03	28.58	26.37	21.29	23.43	25.80	21.22	17.80

Tabla 11: *Resultados de la validación para el modelo de regresión no lineal.*

La tabla 11 recoge los resultados de la validación para el modelo regresión no lineal. A primera vista los resultados son correctos con valores de RMSE inferiores a 35 W, pero si nos fijamos ocurre algo particular. El error que se comete al crear el modelo con los datos de la actividad 2 y validarlos con los mismos datos, es superior a cuando validamos con la

actividad 1. Esto no debería suceder pues el modelo creado con unos datos debe de ser capaz de ajustarse a esos mismos datos, algo que no está sucediendo.

Una posible explicación puede estar en los propios datos con los que se han generado el modelo. Un modelo puede responder a una curva, pero si los datos nuevos están mucho más ajustados a esa misma curva, dará mejores resultados. Aun así, no se puede encontrar ninguna otra explicación a esta particularidad.

6.2. Árbol de Regresión

Árbol de Regresión		Validación							
RMSE		Act.1	Act. 2	Act. 3	Act. 4	Act. 5	Act. 6	Act. 7	Act. 8
Construcción del modelo	Act. 1	5.86	24.66	23.66	32.35	27.04	28.68	27.66	30.97
	Act. 2	25.33	6.23	22.52	28.70	27.46	25.85	29.35	26.70
	Act. 3	23.67	24.50	5.92	33.19	28.49	32.82	31.59	32.81
	Act. 4	31.69	32.26	31.56	6.48	32.04	36.70	28.60	31.25
	Act. 5	23.42	25.91	27.74	30.07	6.81	28.88	26.55	25.56
	Act. 6	25.66	26.86	24.54	30.91	29.46	6.46	32.02	29.66
	Act. 7	27.66	31.59	32.75	36.22	34.67	34.89	6.49	38.01
	Act. 8	30.32	33.58	36.51	33.08	29.58	33.41	31.48	5.35
MAD									
Construcción del modelo	Act. 1	3.71	18.79	18.52	22.59	20.55	21.28	19.92	21.20
	Act. 2	18.87	3.79	17.08	21.19	21.27	19.59	22.42	19.66
	Act. 3	18.43	18.62	3.57	22.32	21.39	24.20	23.18	21.28
	Act. 4	21.54	22.25	19.70	3.78	24.66	25.79	22.77	23.98
	Act. 5	17.85	19.93	22.08	22.20	4.10	22.38	20.21	18.81
	Act. 6	19.80	21.07	18.88	21.53	22.83	3.85	23.20	22.56
	Act. 7	20.79	24.15	23.61	27.00	25.98	27.00	3.96	27.12
	Act. 8	20.82	22.04	22.05	25.15	20.79	23.41	22.60	3.21
MAE									
Construcción del modelo	Act. 1	3.71	18.99	18.52	24.70	20.54	22.16	21.66	22.99
	Act. 2	20.30	3.79	17.45	22.09	21.38	19.59	23.36	20.35
	Act. 3	18.43	19.03	3.57	26.08	21.71	25.04	25.55	25.58
	Act. 4	25.94	24.81	25.10	3.78	25.30	29.67	22.70	24.02
	Act. 5	18.15	20.01	22.04	22.21	4.10	22.31	20.51	19.01
	Act. 6	20.27	21.37	18.94	23.62	23.03	3.85	25.46	22.81
	Act. 7	21.10	24.80	24.87	27.31	25.97	26.89	3.96	27.39
	Act. 8	23.28	26.34	27.37	25.83	23.14	24.80	23.78	3.21

Tabla 12: *Resultados de la validación para el modelo de Árbol de Regresión.*

A diferencia del modelo anterior, en este caso si encontramos que todos los modelos creados y validados con los mismos datos obtienen mejores resultados. Por otra parte, los valores de RMSE son bastante bajos cuando se valida con la misma actividad, algo

totalmente esperado, pero cuando validamos con otras actividades la diferentes es elevado. Lo que nos puede indicar que está excesivamente ajustado a los datos de entrenamiento.

6.3. Redes Neuronales

Redes Neuronales		Validación							
RMSE		Act.1	Act. 2	Act. 3	Act. 4	Act. 5	Act. 6	Act. 7	Act. 8
Construcción del modelo	Act. 1	12.17	24.67	18.45	34.29	23.81	43.72	25.66	36.48
	Act. 2	26.54	13.03	22.99	30.45	26.75	30.08	28.78	31.53
	Act. 3	24.45	29.26	12.58	42.20	27.74	39.05	28.65	34.08
	Act. 4	29.35	30.36	25.86	16.82	24.93	38.30	22.66	25.78
	Act. 5	23.95	26.17	26.53	29.94	14.79	33.44	33.23	33.77
	Act. 6	27.47	25.97	21.87	35.06	29.51	14.63	32.00	32.55
	Act. 7	28.21	33.76	31.03	37.37	32.59	35.99	14.85	34.44
	Act. 8	32.81	36.53	36.19	53.13	28.19	49.51	44.93	11.53
MAD									
Construcción del modelo	Act. 1	9.22	17.28	14.37	23.28	17.91	28.24	18.03	23.34
	Act. 2	18.66	9.61	17.56	22.58	20.77	21.74	22.27	23.92
	Act. 3	18.19	19.44	9.49	27.32	21.22	26.49	20.71	22.96
	Act. 4	18.14	19.31	16.63	12.30	18.64	24.53	17.35	19.40
	Act. 5	18.01	19.24	20.14	21.90	11.11	24.63	23.37	21.55
	Act. 6	19.84	19.95	16.31	21.93	21.81	10.70	22.32	23.88
	Act. 7	19.99	25.13	20.77	26.85	23.15	28.58	11.28	24.36
	Act. 8	21.87	25.00	24.84	33.39	19.86	34.02	29.11	8.44
MAE									
Construcción del modelo	Act. 1	9.22	17.92	14.33	25.33	17.92	26.99	19.60	25.24
	Act. 2	21.37	9.61	17.93	23.36	21.20	21.82	23.35	23.91
	Act. 3	18.19	19.47	9.49	29.71	21.54	26.41	22.45	25.74
	Act. 4	23.62	22.87	20.78	12.30	20.17	28.84	17.37	19.52
	Act. 5	18.01	19.45	20.16	22.31	11.11	24.33	23.51	22.93
	Act. 6	19.84	20.37	16.40	27.26	22.28	10.70	26.38	24.42
	Act. 7	21.67	26.14	23.86	26.50	23.99	28.52	11.27	24.80
	Act. 8	25.58	27.02	27.15	33.72	22.42	34.52	29.13	8.44

Tabla 13: *Resultados de la validación para el modelo de Redes Neuronales*

Para este modelo los valores de los parámetros en la diagonal son bastante buenos, pero crecen cuando validamos los modelos con otras actividades distintas. En caso del RMSE se puede observar valores superiores a 50 W para la actividad 8.

6.4. ANFIS

ANFIS		Validación							
RMSE		Act.1	Act. 2	Act. 3	Act. 4	Act. 5	Act. 6	Act. 7	Act. 8
Construcción del modelo	Act. 1	18.26	28.92	22.56	35.95	27.44	27.27	35.90	34.43
	Act. 2	17.73	17.69	18.85	21.98	20.88	24.17	21.86	23.08
	Act. 3	18.58	20.98	18.29	25.50	21.02	30.12	21.60	24.66
	Act. 4	23.10	24.17	22.07	18.42	23.35	29.20	21.66	23.31
	Act. 5	20.58	22.17	22.77	24.69	18.87	32.78	20.62	22.81
	Act. 6	22.73	24.61	22.26	32.06	29.88	22.17	27.79	35.34
	Act. 7	21.45	22.93	21.73	26.44	23.48	25.34	20.23	25.22
	Act. 8	25.58	25.37	29.09	25.10	27.55	25.28	31.42	25.11
MAD									
Construcción del modelo	Act. 1	13.73	19.04	15.55	21.73	19.28	19.12	22.26	21.57
	Act. 2	13.74	13.69	14.47	15.83	16.12	18.42	16.62	15.42
	Act. 3	14.11	15.87	13.95	18.97	16.68	22.36	16.98	16.71
	Act. 4	15.64	17.13	14.34	13.72	18.01	21.50	16.77	17.56
	Act. 5	15.26	16.64	17.37	18.91	14.52	23.95	16.33	17.29
	Act. 6	18.20	18.45	17.03	21.08	21.72	16.14	19.85	24.66
	Act. 7	17.19	16.91	17.11	20.65	18.80	19.68	15.83	16.82
	Act. 8	17.16	19.16	21.14	19.06	20.11	17.96	24.61	17.12
MAE									
Construcción del modelo	Act. 1	13.65	21.21	17.17	28.27	19.15	19.04	28.37	23.27
	Act. 2	13.86	13.79	15.15	15.99	16.21	19.05	17.04	16.80
	Act. 3	15.04	16.63	14.38	19.39	16.70	22.22	17.20	18.56
	Act. 4	18.69	19.33	17.55	13.70	18.47	23.25	16.80	17.36
	Act. 5	16.22	17.05	17.82	18.73	14.45	24.67	16.32	17.58
	Act. 6	18.20	18.53	17.04	23.95	22.10	16.16	21.91	26.47
	Act. 7	17.63	18.43	17.62	20.80	18.76	19.73	15.83	19.85
	Act. 8	20.41	20.26	23.01	19.94	21.02	17.80	24.97	17.32

***Tabla 14**: Resultados de la validación para el modelo ANFIS*

Por último, el modelo ANFIS nos arroja resultados bastante uniformes. Si observamos la diagonal y comparamos los valores de las demás validaciones no encontramos cambios elevados. Si nos fijamos en el error medio absoluto (MAE), obtenemos valores muy próximos cuando creamos el modelo con la actividad 3 y lo validamos con la actividad 7, que cuando creamos el modelo con la actividad 7 y lo validamos con la actividad 3.

La información contenida en cada una de las tablas permite observar los resultados al crear un modelo con una actividad y validarla con otra distinta, pero si lo que queremos es obtener un resultado global, lo que haremos es hallar los resultados globales, que se recogen en la tabla 15, donde el modelo ANFIS obtiene valores mucho mejores que los demás. A partir de esta información, podemos generar resultados para una actividad

aleatoria. Por ejemplo, creamos el modelo con la actividad 4 del modelo ANFIS y validamos con la actividad 5.

Modelo	RMSE	MAD	MAE
Regresión no lineal	24.85	18.23	19.81
Árbol de regresión	26.76	19.51	20.56
Red Neuronal	29.12	20.53	21.59
ANFIS	24.48	17.87	18.81

Tabla 15: *Resultados globales de la validación.*

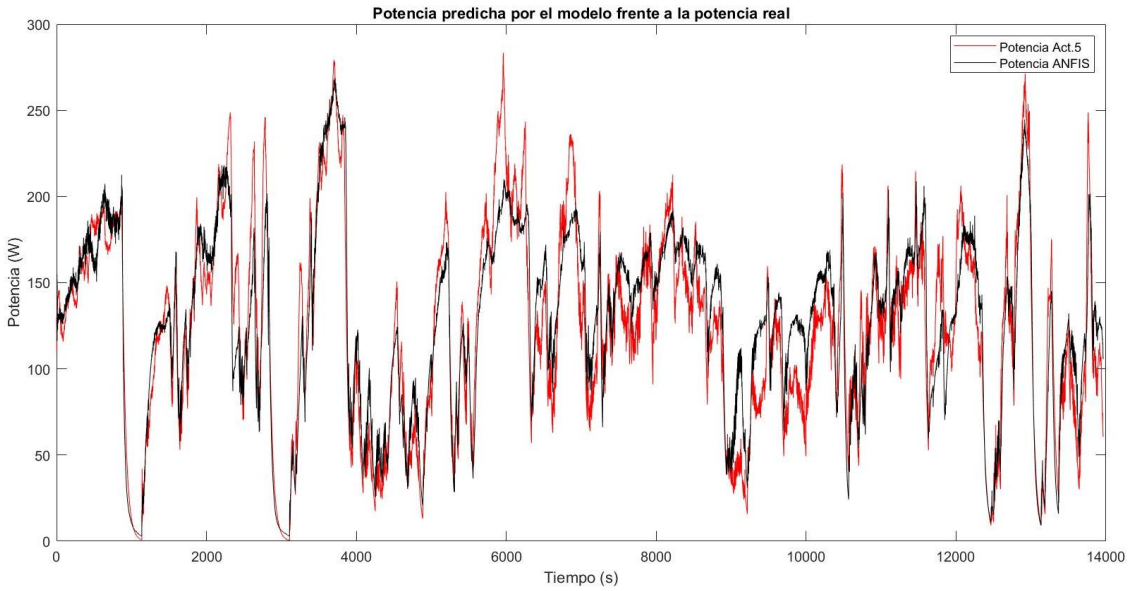

Figura 22: *Potencia predicha al elegir una actividad aleatoria.*

La figura 22 recoge la potencia predicha usando para la validación la actividad 5 y creada con la actividad 4. La curva se asemeja bastante a la real y si nos figamos la repuesta es mejor al inicio y al final de la sesión, pero si lo que queremos es observar el error recurrimos a las funciones representadas en la figura 23.

Mediante la función de densidad espectral de potencia (PSD) obtenemos el valor instantáneo del error esperado cuando este tiende a infinito, observamos que tiende a ser cero. Con la Función de autocorrelación analizamos la dependencia que tienen los datos en *k* periodos anteriores, con los mismos datos, pero en un periodo actual. También podemos indicar que el error alcanza valores cercanos a cero, donde no se observan picos elevados a excepción, de los valores iniciales.

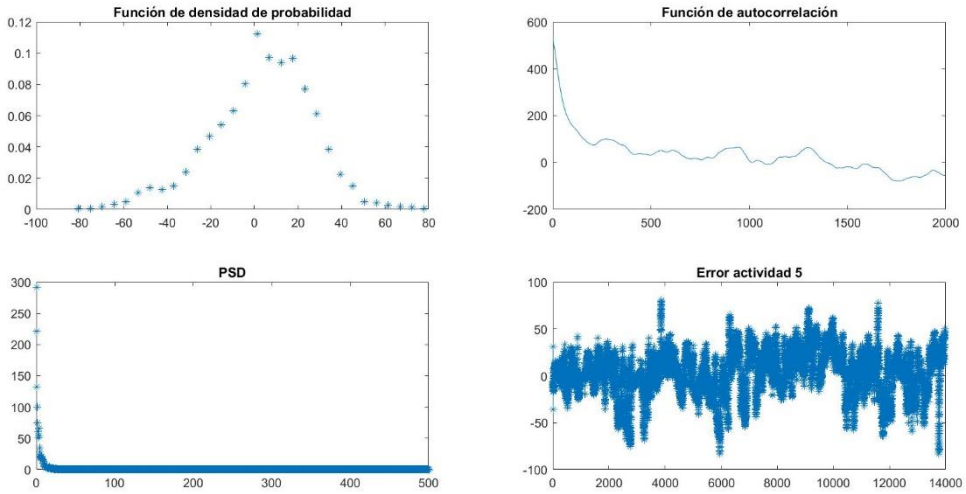

Figura 23: *Gráficas del error para la actividad aleatoria.*

Por último, podemos comparar los resultados creados y validados con las mismas actividades para todos los modelos. Como ya comentamos los resultados son mejores para el modelo ANFIS. El error medio está por debajo de los demás modelos y con la potencia media cercana a la real de la sesión 5. De esta forma respaldamos los resultados globales que se obtuvieron anteriormente.

Modelo	RMSE	MAD	MAE	e_m	P_m real	P_m modelo
Regresión no lineal	23.92	19.10	19.55	4.12	123.31	127.43
Árbol de Regresión	32.04	24.66	25.30	8.25	123.31	131.56
Redes Neuronales	24.93	18.64	20.17	-10.79	123.31	131.65
ANFIS	23.35	18.01	18.47	3.34	123.31	126.65

Tabla 16: *Comparativa de los resultados de la actividad aleatoria con los demás modelos.*

7. Conclusiones y trabajos futuros

Con este trabajo fin de grado nos propusimos implementar nuevos modelos matemáticos para predecir la potencia generada por un ciclista al pedalear. Se han elegido los algoritmos de aprendizaje y ajuste de modelos adecuados para la predicción de la potencia, se ha estudiado e implementado por software de cada uno de ellos

Por otra parte, los estudios previos mencionados en la bibliografía permitieron deducir que para la predicción de potencia de pedaleada podemos prescindir de factores externos como la velocidad del aire, ya que modelos basados en *machine learning* son capaces de arrojar resultados con errores bastante bajos en comparación con modelos matemáticos de regresión simple. Aun así, estos factores si tienen influencia, por lo que, para obtener resultados mucho más completos deberíamos ser capaces de medirlos.

Antes de realizar cualquier desarrollo procesamos y aplicamos un filtro con el fin de eliminar el ruido en los datos, pues son datos reales extraídos de un sensor que se han recogido durante las actividades de ciclistas en carretera. Se obtuvo resultados notables pues el filtro de diez segundos produjo que la curva de entrenamiento se suavice pudiendo descartar el ruido que no es relevante.

Para la selección de variables se ha usado una alternativa al método GMDH, ya que era necesario crear numerosos modelos para cada fase de la selección, algo que requiere un estudio más profundo. El método implementado se basa en el estudio de la correlación entre las variables y error acumulado del modelo. Nos dio como resultado que solamente es necesario introducir tres variables para todos los modelos: inclinación, FC y cadencia.

En la tabla 15 se puede visualizar los parámetros globales de todos los modelos. Observamos que son mejores para el modelo ANFIS, algo que tiene sentido, puesto que usa un aprendizaje basando en redes neuronales adaptativas ampliamente utilizado en diferentes campos.

El modelo Árbol de Regresión fue el que mejores resultados obtuvo en cuanto a los entrenamientos. Se ajusta muy bien a los datos como se puede deducir de la tabla 12, donde la diagonal presenta errores muy bajos. Sin embargo, al usar estos modelos con otras actividades vemos que los valores de los parámetros crecen y son bastante altos. Es significativo que para todos los modelos los parámetros utilizados son relativamente altos al usar la sesión número 8 a excepción del modelo ANFIS, en el que se mantiene uniforme y constante.

Como el modelo ANFIS obtiene los mejores resultados, se ha generado un entrenamiento con la actividad 4 y validado con la actividad 5. Estos resultados se recogen en la tabla 16, de tal forma que se obtiene un error medio muy bajo comparado con otros

modelos creados y validados con las mismas actividades. El RMSE es el menor de todos y la potencia media es la más cercana con una diferencia de 3.34 W.

En base a todos estos precedentes, podemos decir que hemos obtenido un modelo neuro difuso que es capaz de predecir la potencia generada usando tres variables que son fáciles de conseguir, por lo que podría llegarse a usar con otros datos que no se hayan usado en este trabajo. Por tanto, el objetivo planteado al inicio se ha cumplido. Hemos obtenido un modelo que no se ajusta perfectamente a los datos, pero que tiene alta capacidad de generalización.

8. Bibliografía

- Allen, H., & Coggan, A. (2014). *Entrenar y Correr con Potenciómetro.* Paidotribo. Recuperado el 12 de junio de 2022, de https://elibro.net/es/ereader/univupct/116247
- Asociación de Marcas y Bicicletas de España, AMBE. (Abril, 2022). *El sector de la bicicleta en cifras, 2021*. Recuperado el 14 de Mayo de 2022, de https://asociacionambe.com/
- Behar, A. A., & Iranzo, M. A. (2003). *Identificación y control adaptativo.* Alambra. Recuperado el 10 de Marzo de 2022, de https://www.casadellibro.com/libro-identificacion-y-control-adaptativo/9788420535708/883689
- Carrillo Vera, D. (2019). Desarrollo de un Modelo Matemático de predicción de la Potencia generada por un ciclista usando variables Biométricas. Recuperado el 21 de 06 de 2022, de https://repositorio.upct.es/handle/10317/9917
- Cragnulini, F. (s. f.). INTRODUCCIÓN AL ENTRENAMIENTO POR POTENCIA. 45.
- Di Prampero, P., Cortili, G., Mognoni, P., & Saibene, F. (1979). Equation of motion of a cyclist. *Jurnal of Applied Physiology*, 47. Obtenido de https://pubmed.ncbi.nlm.nih.gov/468661/
- Hernandez, F., & Fernandez, F. H. (2012). Identificación Inteligente de un Proceso Fermentativo Usando el Algoritmo GMDH Modificado. *Revista iberoamericana de automática e informática industrial (RIAI)*. Recuperado el 10 de Junio de 2022, de https://doi.org/10.1016/j.riai.2011.11.001
- Kataoka, Y., & Gris, P. (2019). Real-Time Power Performance Prediction in Tour de France. *Lecture Notes in Compute Science*. Recuperado el 12 de Julio de 2022, de https://link.springer.com/chapter/10.1007/978-3-030-17274-9_10
- Lemaître, G., & Lemaître, C. (2018). Estimate Power without Measuring it: a Machine Learning Application. *Journal of Science & cycling*. Obtenido de https://www.jsc-journal.com/index.php/JSC/article/view/420
- Maier, T., Schmid, L., Müller, B., Steiner, T., & Wehrlin, J. P. (s.f.). Accuracy of Cycling Power Meters against a Mathematical Model of Treadmill Cycling. *Internacional journal or sport medicine*. Recuperado el 21 de 06 de 2022, de https://pubmed.ncbi.nlm.nih.gov/28482367/

- Maier, T., Shmid, L., Müller, B., Steiner, T., & Wehrlin, J. P. (2017). Accuracy of Cycling Power Meters against a Mathematical Model of Treadmill Cycling. *International Journal of Sports Medicine*, 38. Obtenido de https://pubmed.ncbi.nlm.nih.gov/28482367/
- Martin, J. C., Milliken, D. L., Cobb, J. E., McFadden, K. L., & Coggan, A. R. (1998). Validation of a Mathematical Model for Road Cycling Power. *Journal of applied Biomechanics*. Obtenido de https://pubmed.ncbi.nlm.nih.gov/28121252/
- MathWorks, E. (s.f.). *Redes neuronales*. Recuperado el 2022 de Junio de 2022, de https://es.mathworks.com/discovery/neural-network.html
- MathWorks, E. (s.f.). *Regresión no lineal*. Recuperado el 12 de Marzo de 2022, de MATLAB & Simulink: https://es.mathworks.com/help/stats/nonlinear-models.html
- MathWorks, E. (s. f.). *Aplicación de aprendizaje de regresión*. Recuperado el 2022 de Marzo de 5, de MATLAB & Simulink: https://es.mathworks.com/help/stats/regression-learner-app.html?s_tid=CRUX_lftnav
- MathWorks, E. (s. f.). *Group Method of Data Handling (GMDH)*. Recuperado el 10 de Junio de 2022, de https://es.mathworks.com/matlabcentral/fileexchange/52906-group-method-of-data-handling-gmdh
- Rao, U., Sood, Y., & Jarial, R. (2015). *Subtractive Clustering Fuzzy Expert System for Engineering Applications.* Recuperado el 14 de Junio de 2022, de https://doi.org/10.1016/j.procs.2015.04.153
- Rodrigo, J. A. (2016). *Estadística y Machine Learning con R.* Recuperado el 2022 de Marzo de 15, de Correlación Lineal y Regresión Lineal Simple: https://github.com/JoaquinAmatRodrigo/Estadistica-con-R
- Rodrigo, J. A. (Febrero de 2017). *Estadística y Machine Learning con R.* Recuperado el 12 de Julio de 2022, de Árboles de predicción: bagging, random forest,: https://github.com/JoaquinAmatRodrigo/Estadistica-con-R
- Rodrigo, J. A. (2017). *Estadística y Machine Learning con R*. Recuperado el 17 de Marzo de 2022, de Métodos de regresión no lineal: Regresión: https://github.com/JoaquinAmatRodrigo/Estadistica-con-R
- W, S. T., & S, M. (2018). Investigating the performance of ANFIS model to predict the hourly temperature in Pattani, Thailand. *Journal of Physics: Conference Series*.

Recuperado el 15 de Mayo de 2022, de https://iopscience.iop.org/article/10.1088/1742-6596/1097/1/012085

9. Anexos

Anexo 1: Script para procesado de datos

```
%% Las actividades de cargan modificando el numero al final de
'datos'
%% Ejemplo:'datos5.*****'

clear;
load datos8;
DATOS(:,1)=datos8.ns1DistanceMeters3;
velo(:,1)=DATOS(2:end,1)-DATOS(1:end-1,1);
velo(end+1,1)=velo(end);
velocidad(:,1)=velo(:,1);
DATOS(:,2)=datos8.ns1AltitudeMeters;
incl(:,1)=DATOS(2:end,2)-DATOS(1:end-1,2);
incl(end+1,1)=incl(end);
inclinacion(:,1)=incl(:,1);
fcard(:,1)=datos8.ns1Value4;
cadencia(:,1)=datos8.ns1Cadence5;
potencia(:,1)=datos8.ns2Watts

%% Rellenamos celdas sin valor

inclinacion=fillmissing(inclinacion, 'constant',0);
cadencia=fillmissing(cadencia,'constant',0);
fcard=fillmissing(fcard,'constant',0);
velocidad=fillmissing(velocidad,'constant',0);
potencia=fillmissing(potencia,'constant',0);

DATOS8_SINFILTRO=[velocidad, inclinacion, fcard, cadencia,
potencia];

%% Relacionamos cadencia y potencia

DATOS8_CEROS=DATOS8_SINFILTRO;

for i=1 : size(DATOS8_CEROS)

    if DATOS8_CEROS(i:4) == 0  % Si la cadencia es cero
       DATOS8_CEROS(i:5)= 0;  % La potencia de salida es cero
    end

end

%% Filtro de diez muestras anteriores

DATOS8_FILTRADO=DATOS8_CEROS;

for i= 10 : length(DATOS8_FILTRADO)
    DATOS8_FILTRADO(i,:) = mean(DATOS8_FILTRADO(i-9:i,:));
end
```

```
save DATOS8_COMPLETO.dat DATOS8_FILTRADO -ascii

%% Eliminamos columna de potencia

DATOS8_SINPOTENCIA=DATOS8_FILTRADO;
DATOS8_SINPOTENCIA(:,5)=[];

Tbl8=table(DATOS8_FILTRADO(:,1), DATOS8_FILTRADO(:,2),
DATOS8_FILTRADO(:,3), ...
   DATOS8_FILTRADO(:,4),DATOS8_FILTRADO(:,5),...
   'VariableNames',{'Velocidad','Inclinacion','FCardiaca',
'Cadencia', 'Potencia'});

save DATOS8_SINPOTENCIA DATOS8_SINPOTENCIA;
save Tbl8 Tbl8;

%% Generamos la variable tiempo para cada actividad
load Tbl8;
for i=1 : height(Tbl8)
   tiempo8(i,1)=i;
end
save tiempo8 tiempo8;

clear
```

Anexo 2: Script para la selección de variables

- **Regresión no lineal**

```
%% SELECCIÓN DE VARIABLES: MODELO NO LINEAL
clear
load Tbl2;
load DATOS2_SINPOTENCIA;
load tiempo2;

Y_REAL=Tbl2.Potencia

%% MODELO 4VARIABLES:

Tbl_4VAR=table(Tbl2.Velocidad,Tbl2.Inclinacion,
Tbl2.FCardiaca,Tbl2.Cadencia, Tbl2.Potencia,...

'VariableNames',{'Velocidad','Inclinacion','FCardiaca','Cadencia',
'Potencia'});

%Propuesta 1:
% model_4VAR= @(b,x)b(1)+
b(2)*x(:,1)+b(3)*sin(x(:,2))+b(4)*x(:,3)+cos(x(:,4))
% beta_4VAR = [1, 1,1,1];

%Propuesta 2:
% model_4VAR= @(b,x)b(1)+
b(2)*x(:,1)+b(3)*(x(:,2).*x(:,2))+b(4)*x(:,3)+x(:,4)
```

```
% beta_4VAR = [1, 1,1,0.1];
%Propuesta 3:
% model_4VAR= @(b,x)b(1)*x(:,2)+b(2)*x(:,4)+
b(3)*(x(:,3).*x(:,3))+b(4)*x(:,1)
% beta_4VAR = [1, 1,1,0.1];

%Propuesta 4:
model_4VAR= @(b,x)b(1)+b(2)*x(:,2)+b(3)*x(:,4)+
b(4)*(x(:,3).*x(:,3))+b(5)*x(:,1)
beta_4VAR = [1, 1,1,1,1]

MODELO_4VAR= fitnlm(Tbl_4VAR,model_4VAR,beta_4VAR)

Y_4VAR=predict(MODELO_4VAR, Tbl_4VAR);

%Quitamos valores negativos de la potencia
for i=1 : height(Tbl_4VAR)
    if Y_4VAR(i)<=0
        Y_4VAR(i)=0;
    end
end

RMSE_4VAR=sqrt(mse(Y_4VAR, Y_REAL))
R_4VAR=MODELO_4VAR.Rsquared
e_4VAR=Y_4VAR-Y_REAL;
MAD_4VAR=mad(e_4VAR)

tiledlayout(2,1)
nexttile
plot(tiempo2, Y_REAL,'red')
hold on;
plot(tiempo2, Y_4VAR,'black')
legend('Potencia Act. 2','Potencia Propuesta 3')
title('Potencia estimada modelo de Regresión no lineal')
xlabel('Tiempo (s)')
ylabel('Potencia (W)')

nexttile
plot(tiempo2, e_4VAR,'b')
title('Error modelo Regresión no lineal ')
xlabel('Tiempo (s)')
ylabel('Error (W)')

C11=corr2(Tbl2.Velocidad, Y_REAL);
C12=corr2(Tbl2.Inclinacion,Y_REAL);
C13=corr2(Tbl2.FCardiaca, Y_REAL);
C14=corr2(Tbl2.Cadencia, Y_REAL);
Tabla1= [C11 C12 C13 C14];

Y_MEDIA_REAL=mean(Tbl2.Potencia);
Y_MEDIA_A3=mean(Y_4VAR)

%% SELECCIONAMOS LA PRIMERA VARIABLE: CADENCIA
% Tbl_CAD=table(Tbl2.Cadencia,
Tbl2.Potencia,'VariableNames',{'Cadencia','Potencia'});
%
```

```
% %MODELO
% model_CAD= @(b,x)b(1).*x(:,1)
% beta_CAD = [100];
% MODELO_CAD= fitnlm(Tbl_CAD,model_CAD,beta_CAD)
%
% Y_CAD=predict(MODELO_CAD, Tbl_CAD);
%
% %Quitamos valores negativos de la potencia
% for i=1 : height(Tbl_CAD)
%     if Y_CAD(i)<=0
%         Y_CAD(i)=0;
%     end
% end
%
% RMSE_CAD=sqrt(mse(Y_CAD, Y_REAL))
% R_CAD=MODELO_CAD.Rsquared
% e_CAD=Y_CAD-Y_REAL;
% MAD_CAD=mad(e_CAD)
%
% tiledlayout(2,1)
% nexttile
% plot(tiempo2, Y_REAL,'red')
% hold on;
% plot(tiempo2, Y_CAD,'black')
% legend('Potencia Act. 2','Potencia 1 Variable')
% title('Selección Regresión no lineal')
% xlabel('Tiempo (s)')
% ylabel('Potencia (W)')
%
% nexttile
% plot(tiempo2, e_CAD,'b')
% title('Error selección Regresión no lineal ')
% xlabel('Tiempo (s)')
% ylabel('Error (W)')
%
% C21=corr2(Tbl2.Velocidad, e_CAD);
% C22=corr2(Tbl2.Inclinacion, e_CAD);
% C23=corr2(Tbl2.FCardiaca, e_CAD);
% C24=corr2(Tbl2.Cadencia, e_CAD);
%
% Tabla2= [C21 C22 C23 C24]


%% MODELO 2 VARIABLES: CADENCIA + FCARDICA
% Tbl_2VAR=table(Tbl2.Cadencia,Tbl2.FCardiaca,
Tbl2.Potencia,'VariableNames',{'Cadencia',...
%     'FCardiaca','Potencia'});
%
% %MODELO
% model_2VAR= @(b,x)b(1).*x(:,1)+b(2).*(x(:,2).*x(:,1))
% beta_2VAR = [10, 1];
% MODELO_2VAR= fitnlm(Tbl_2VAR,model_2VAR,beta_2VAR)
%
% Y_2VAR=predict(MODELO_2VAR, Tbl_2VAR);
%
% %Quitamos valores negativos de la potencia
```

```
% for i=1 : height(Tbl_2VAR)
%     if Y_2VAR(i)<=0
%         Y_2VAR(i)=0;
%     end
% end
%
% RMSE_2VAR=sqrt(mse(Y_2VAR, Y_REAL))
% R_2VAR=MODELO_2VAR.Rsquared
% e_2VAR=Y_2VAR-Y_REAL;
% MAD_2VAR=mad(e_2VAR)
%
% figure;
% plot(tiempo2, Y_REAL,'red')
% hold on;
% plot(tiempo2, Y_2VAR,'black')
% legend('Real','2VAR')
%
% C31=corr2(Tbl2.Velocidad, e_2VAR);
% C32=corr2(Tbl2.Inclinacion, e_2VAR);
% C33=corr2(Tbl2.FCardiaca, e_2VAR);
% C34=corr2(Tbl2.Cadencia, e_2VAR);
%
% Tabla3= [C31 C32 C33 C34];

% %% MODELO 3 VARIABLES: CADENCIA + INCLINACIÓN + FCARD
% Tbl_3VAR=table(Tbl2.Cadencia,Tbl2.Inclinacion, Tbl2.FCardiaca,
Tbl2.Potencia,...
%
'VariableNames',{'Cadencia','Inclinacion','FCardiaca','Potencia'})
;
%
% %MODELO
% model_3VAR= @(b,x)b(1)+b(2)*x(:,2)+b(3)*x(:,1)+
b(4)*(x(:,3).*x(:,3))
% beta_3VAR = [1, 1,1,1];
% MODELO_3VAR= fitnlm(Tbl_3VAR,model_3VAR,beta_3VAR)
%
% Y_3VAR=predict(MODELO_3VAR, Tbl_3VAR);
%
% %Quitamos valores negativos de la potencia
% % for i=1 : height(Tbl_3VAR)
% %     if Y_3VAR(i)<=0
% %         Y_3VAR(i)=0;
% %     end
% % end
%
% RMSE_3VAR=sqrt(mse(Y_3VAR, Y_REAL))
% R_3VAR=MODELO_3VAR.Rsquared
% e_3VAR=Y_3VAR-Y_REAL;
% MAD_3VAR=mad(e_3VAR)
% e_MAE=abs(Y_REAL-Y_3VAR);
%
% tiledlayout(2,1)
% nexttile
% plot(tiempo2, Y_REAL,'red')
% hold on;
```

```
% plot(tiempo2, Y_3VAR,'black')
% legend('Potencia Act. 2','Potencia 3 variables')
% title('Selección, Regresión no lineal')
% xlabel('Tiempo (s)')
% ylabel('Potencia (W)')
%
% nexttile
% plot(tiempo2, e_3VAR,'b')
% title('Error selección, Regresión no lineal ')
% xlabel('Tiempo (s)')
% ylabel('Error (W)')
%
% C41=corr2(Tbl2.Velocidad, e_3VAR)
% C42=corr2(Tbl2.Inclinacion, e_3VAR)
% C43=corr2(Tbl2.FCardiaca, e_3VAR)
% C44=corr2(Tbl2.Cadencia, e_3VAR)
%
%
% Tabla4= [C41 C42 C43 C44];
```

- **Árbol de regresión**

```
%% SELECCIÓN DE VARIABLES: ÁRBOL DE REGRESIÓN
clear
load Tbl2;
load tiempo2;
load DATOS2_SINPOTENCIA;
Y_REAL=Tbl2.Potencia;
%% MODELO CON TODAS LAS VARIABLES DISPONIBLES: DATOS2
load MODELO_ARBOL_4VAR;
Y_4VAR = MODELO_ARBOL_4VAR.predictFcn(Tbl2);

MSE_4VAR=loss(MODELO_ARBOL_4VAR.RegressionTree,Tbl2,
Tbl2.Potencia)
RMSE_4VAR=sqrt(immse(Y_4VAR, Y_REAL))
%mae_6VAR=mae(Y_6VAR,Y_REAL)
e=Y_4VAR-Y_REAL;
MAD_4VAR=mad(e)

Corr_Velocidad=corrcoef(Tbl2.Velocidad, Y_REAL);
Corr_Velocidad=Corr_Velocidad(1,2)

Corr_Inclinacion=corrcoef(Tbl2.Inclinacion, Y_REAL);
Corr_Inclinacion=Corr_Inclinacion(1,2)

Corr_Fcard=corrcoef(Tbl2.FCardiaca, Y_REAL);
Corr_Fcard=Corr_Fcard(1,2)

Corr_Cadencia=corrcoef(Tbl2.Cadencia, Y_REAL);
Corr_Cadencia=Corr_Cadencia(1,2)

tiledlayout(2,1)
nexttile
plot(tiempo2, Y_REAL,'red')
hold on;
```

```
plot(tiempo2, Y_4VAR,'black')
legend('Potencia Act. 2','Potencia Árbol regresión')
title('Potencia estimada ')
xlabel('Tiempo (s)')
ylabel('Potencia (W)')
nexttile
plot(tiempo2, e,'b')
title('Error modelo Árbol regresión')
xlabel('Tiempo (s)')
ylabel('Error (W)')

%% MODELO 1 VARIABLE: CADENCIA
% load MODELO_ARBOL_1VAR;
% Tbl2_1VAR=table(Tbl2.Cadencia, Tbl2.Potencia,...
%     'VariableNames',{'Cadencia', 'Potencia'});
%
% Y_1VAR = MODELO_ARBOL_1VAR.predictFcn(Tbl2_1VAR);
%
% MSE_1VAR=loss(MODELO_ARBOL_1VAR.RegressionTree,Tbl2_1VAR,
Tbl2_1VAR.Potencia)
% RMSE_1VAR=sqrt(immse(Y_1VAR, Y_REAL))
% e_1VAR=Y_1VAR-Y_REAL;
% MAD_1VAR=mad(e_1VAR)
%
% C11=corr2(Tbl2.Velocidad, e_1VAR);
% C12=corr2(Tbl2.Inclinacion,e_1VAR);
% C13=corr2(Tbl2.FCardiaca, e_1VAR);
% C14=corr2(Tbl2.Cadencia, e_1VAR);
% Tabla1= [C11 C12 C13 C14];
%
% tiledlayout(2,1)
% nexttile
% plot(tiempo2, Y_REAL,'red')
% hold on;
% plot(tiempo2, Y_1VAR,'black')
% legend('Real','1VAR')
% title('Potencia real y predicha por el modelo')
% nexttile
% plot(tiempo2, e_1VAR,'b')
% title('Error Modelo Arbol regresión 1 variable')


%% MODELO 2 VARIABLES: CADENCIA + F.CARDIACA
% load MODELO_ARBOL_2VAR;
% Tbl2_2VAR=table(Tbl2.FCardiaca, Tbl2.Cadencia, Tbl2.Potencia,...
%     'VariableNames',{'FCardiaca','Cadencia', 'Potencia'});
%
% Y_2VAR = MODELO_ARBOL_2VAR.predictFcn(Tbl2_2VAR);
%
% MSE_2VAR=loss(MODELO_ARBOL_2VAR.RegressionTree,Tbl2_2VAR,
Tbl2_2VAR.Potencia)
% RMSE_2VAR=sqrt(immse(Y_2VAR, Y_REAL))
% e_2VAR=Y_2VAR-Y_REAL;
% MAD_2VAR=mad(e_2VAR)
%
%
```

```
% C11=corr2(Tbl2.Velocidad, e_2VAR);
% C12=corr2(Tbl2.Inclinacion,e_2VAR);
% C13=corr2(Tbl2.FCardiaca, e_2VAR);
% C14=corr2(Tbl2.Cadencia, e_2VAR);
% Tabla1= [C11 C12 C13 C14];
%
% tiledlayout(2,1)
% nexttile
% plot(tiempo2, Y_REAL,'red')
% hold on;
% plot(tiempo2, Y_2VAR,'black')
% legend('Real','2VAR')
% title('Potencia real y predicha por el modelo')
% nexttile
% plot(tiempo2, e_2VAR,'b')
% title('Error Modelo Arbol regresión 2 variables')


% %% MODELO 3 VARIABLES: CADENCIA + INCLINACION + FCARD
%
% load MODELO_ARBOL_3VAR;
% Tbl2_3VAR=table(Tbl2.Inclinacion, Tbl2.FCardiaca,Tbl2.Cadencia,
Tbl2.Potencia,...
%    'VariableNames',{'Inclinacion','FCardiaca','Cadencia',
'Potencia'});
%
% Y_3VAR = MODELO_ARBOL_3VAR.predictFcn(Tbl2_3VAR);
%
% MSE_3VAR=loss(MODELO_ARBOL_3VAR.RegressionTree,Tbl2_3VAR,
Tbl2_3VAR.Potencia)
% RMSE_3VAR=sqrt(immse(Y_3VAR, Y_REAL))
% e_3VAR=Y_3VAR-Y_REAL;
% MAD_3VAR=mad(e_3VAR)
%
% C11=corr2(Tbl2.Velocidad, e_3VAR);
% C12=corr2(Tbl2.Inclinacion,e_3VAR);
% C13=corr2(Tbl2.FCardiaca, e_3VAR);
% C14=corr2(Tbl2.Cadencia, e_3VAR);
% Tabla1= [C11 C12 C13 C14];
%
% tiledlayout(2,1)
% nexttile
% plot(tiempo2, Y_REAL,'red')
% hold on;
% plot(tiempo2, Y_3VAR,'black')
% legend('Potencia Act. 2','Potencia 3 variables')
% title('Selección, Árbol regresión')
% xlabel('Tiempo (s)')
% ylabel('Potencia (W)')
% nexttile
% plot(tiempo2, e_3VAR,'b')
% title('Error selección, Árbol regresión')
% xlabel('Tiempo (s)')
% ylabel('Error (W)')
```

- **Red Neuronal**

```
%% SELECCIÓN DE VARIABLES: NEURAL NETWORK FITING
clear;
load Tbl2;
load DATOS2_SINPOTENCIA;
load tiempo2;
Y_REAL=Tbl2.Potencia;

%% MODELO TODAS LAS VARIABLES:
Tbl2_4VAR=[DATOS2_SINPOTENCIA(:,1), DATOS2_SINPOTENCIA(:,2),
DATOS2_SINPOTENCIA(:,3), ...
 DATOS2_SINPOTENCIA(:,4)];

% Levenberg-Marquardt backpropagation.
trainFcn = 'trainlm';
% Create a Fitting Network
hiddenLayerSize = 30;
net = fitnet(hiddenLayerSize,trainFcn);
% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
[net,tr] = train(net,Tbl2_4VAR',Y_REAL');

% Test the Network
Y_4VAR = net(Tbl2_4VAR');
for i=1 : length(Y_4VAR)
    if Y_4VAR(i)<=0
        Y_4VAR(i)=0;
    end
end
e = gsubtract(Y_REAL',Y_4VAR);
RMSE_4VAR= sqrt(perform(net,Y_REAL',Y_4VAR))
MAD_4VAR=mad(e')

ploterrhist(e)
figure;
plotregression(Y_REAL,Y_4VAR)

C11=corr2(Tbl2.Velocidad, Y_REAL)
C12=corr2(Tbl2.Inclinacion, Y_REAL)
C13=corr2(Tbl2.FCardiaca, Y_REAL)
C14=corr2(Tbl2.Cadencia, Y_REAL)

Tabla1= [C11 C12 C13 C14];
figure;
tiledlayout(2,1)
nexttile
plot(tiempo2, Y_REAL,'red')
hold on;
plot(tiempo2, Y_4VAR,'black')
```

```
legend('Potencia Act. 2','Potencia Neural Network')
title('Potencia predicha Red neuronal')
xlabel('Tiempo (s)')
ylabel('Potencia (W)')
nexttile
plot(tiempo2, e,'b')
title('Error Red neuronal')
xlabel('Tiempo (s)')
ylabel('Error (W)')


%% MODELO 1 VARIABLE: CADENCIA
% Tbl2_1VAR=[DATOS2_SINPOTENCIA(:,4)];
%
% trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.
%
% % Create a Fitting Network
% hiddenLayerSize = 30;
% net = fitnet(hiddenLayerSize,trainFcn);
%
% % Setup Division of Data for Training, Validation, Testing
% net.divideParam.trainRatio = 70/100;
% net.divideParam.valRatio = 15/100;
% net.divideParam.testRatio = 15/100;
%
% % Train the Network
% [net,tr] = train(net,Tbl2_1VAR',Y_REAL');
%
% % Test the Network
% Y_1VAR = net(Tbl2_1VAR');
% for i=1 : length(Y_1VAR)
%     if Y_1VAR(i)<=0
%         Y_1VAR(i)=0;
%     end
% end
% e_1VAR = gsubtract(Y_REAL',Y_1VAR);
% RMSE_1VAR = sqrt(perform(net,Y_REAL',Y_1VAR))
% MAD_1VAR=mad(e_1VAR')
%
% ploterrhist(e_1VAR)
% figure;
% plotregression(Y_REAL,Y_1VAR)
%
% C11=corr2(Tbl2.Velocidad, e_1VAR')
% C12=corr2(Tbl2.Inclinacion, e_1VAR')
% C13=corr2(Tbl2.FCardiaca, e_1VAR')
% C14=corr2(Tbl2.Cadencia, e_1VAR')
%
% Tabla1= [C11 C12 C13 C14];
% figure;
% tiledlayout(2,1)
% nexttile
% plot(tiempo2, Y_REAL,'red')
% hold on;
% plot(tiempo2, Y_1VAR,'black')
% legend('Real','1VAR')
```

```
% title('Potencia real y predicha por el modelo')
% nexttile
% plot(tiempo2, e_1VAR,'b')
% title('Error Modelo NET 1 variable')

%% MODELO 2 VARIABLES: CADENCIA + FCARDIACA

% Tbl2_2VAR=[DATOS2_SINPOTENCIA(:,3), DATOS2_SINPOTENCIA(:,4)];
%
% trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.
%
% % Create a Fitting Network
% hiddenLayerSize = 30;
% net = fitnet(hiddenLayerSize,trainFcn);
%
% % Setup Division of Data for Training, Validation, Testing
% net.divideParam.trainRatio = 70/100;
% net.divideParam.valRatio = 15/100;
% net.divideParam.testRatio = 15/100;
%
% % Train the Network
% [net,tr] = train(net,Tbl2_2VAR',Y_REAL');
%
% % Test the Network
% Y_2VAR = net(Tbl2_2VAR');
% for i=1 : length(Y_2VAR)
%     if Y_2VAR(i)<=0
%         Y_2VAR(i)=0;
%     end
% end
% e_2VAR = gsubtract(Y_REAL',Y_2VAR);
% RMSE_2VAR = sqrt(perform(net,Y_REAL',Y_2VAR))
% MAD_2VAR=mad(e_2VAR')
%
% ploterrhist(e_2VAR)
% figure;
% plotregression(Y_REAL,Y_2VAR)
%
% C11=corr2(Tbl2.Velocidad, e_2VAR')
% C12=corr2(Tbl2.Inclinacion, e_2VAR')
% C13=corr2(Tbl2.FCardiaca, e_2VAR')
% C14=corr2(Tbl2.Cadencia, e_2VAR')
%
% Tabla1= [C11 C12 C13 C14];
% figure;
% tiledlayout(2,1)
% nexttile
% plot(tiempo2, Y_REAL,'red')
% hold on;
% plot(tiempo2, Y_2VAR,'black')
% legend('Real','2VAR')
% title('Potencia real y predicha por el modelo')
% nexttile
% plot(tiempo2, e_2VAR,'b')
% title('Error Modelo NET 2 variables')
```

```
%% MODELO 3 VARIABLES: CADENCIA + INCLINACIÓN + F CARDIACA

% Tbl2_3VAR=[DATOS2_SINPOTENCIA(:,2), DATOS2_SINPOTENCIA(:,3),
DATOS2_SINPOTENCIA(:,4)];
%
% trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.
%
% % Create a Fitting Network
% hiddenLayerSize = 30;
% net = fitnet(hiddenLayerSize,trainFcn);
%
% %Setup Division of Data for Training, Validation, Testing
% net.divideParam.trainRatio = 70/100;
% net.divideParam.valRatio = 15/100;
% net.divideParam.testRatio = 15/100;
%
% %Train the Network
% [net,tr] = train(net,Tbl2_3VAR',Y_REAL');
%
% %Test the Network
% Y_3VAR = net(Tbl2_3VAR');
% % for i=1 : length(Y_3VAR)
% %     if Y_3VAR(i)<=0
% %         Y_3VAR(i)=0;
% %     end
% % end
% e_3VAR = gsubtract(Y_REAL',Y_3VAR);
% RMSE_3VAR = sqrt(perform(net,Y_REAL',Y_3VAR))
% MAD_3VAR=mad(e_3VAR')
%
% ploterrhist(e_3VAR)
% figure;
% plotregression(Y_REAL,Y_3VAR)
%
% C11=corr2(Tbl2.Velocidad, e_3VAR')
% C12=corr2(Tbl2.Inclinacion, e_3VAR')
% C13=corr2(Tbl2.FCardiaca, e_3VAR')
% C14=corr2(Tbl2.Cadencia, e_3VAR')
%
% Tabla1= [C11 C12 C13 C14];
% figure;
% tiledlayout(2,1)
% nexttile
% plot(tiempo2, Y_REAL,'red')
% hold on;
% plot(tiempo2, Y_3VAR,'black')
% legend('Potencia Act. 2','Potencia Neural Network')
% title('Selección, Red neuronal')
% xlabel('Tiempo (s)')
% ylabel('Potencia (W)')
% nexttile
% plot(tiempo2, e_3VAR,'b')
% title('Error selección, Red neuronal')
% xlabel('Tiempo (s)')
% ylabel('Error (W)')
```

Anexo 3: Validación de modelos

- **Regresión no lineal**

```
%% VALIDACIÓN: REGRESIÓN NO LINEAL
clear;
load Tbl1;load Tbl2;load Tbl3;load Tbl4;load Tbl5;load Tbl6;load
Tbl7;load Tbl8;
load tiempo1;load tiempo2;load tiempo3;load tiempo4;load tiempo5;
load tiempo6;load tiempo7;load tiempo8;
Y_REAL_A1=Tbl1.Potencia;
Y_REAL_A2=Tbl2.Potencia;
Y_REAL_A3=Tbl3.Potencia;
Y_REAL_A4=Tbl4.Potencia;
Y_REAL_A5=Tbl5.Potencia;
Y_REAL_A6=Tbl6.Potencia;
Y_REAL_A7=Tbl7.Potencia;
Y_REAL_A8=Tbl8.Potencia;

% Eliminamos la variable velocidad

Tbl1_s=table(Tbl1.Cadencia,Tbl1.Inclinacion, Tbl1.FCardiaca,
Tbl1.Potencia,...

'VariableNames',{'Cadencia','Inclinacion','FCardiaca','Potencia'})
;
Tbl2_s=table(Tbl2.Cadencia,Tbl2.Inclinacion, Tbl2.FCardiaca,
Tbl2.Potencia,...

'VariableNames',{'Cadencia','Inclinacion','FCardiaca','Potencia'})
;
Tbl3_s=table(Tbl3.Cadencia,Tbl3.Inclinacion, Tbl3.FCardiaca,
Tbl3.Potencia,...

'VariableNames',{'Cadencia','Inclinacion','FCardiaca','Potencia'})
;
Tbl4_s=table(Tbl4.Cadencia,Tbl4.Inclinacion, Tbl4.FCardiaca,
Tbl4.Potencia,...

'VariableNames',{'Cadencia','Inclinacion','FCardiaca','Potencia'})
;
Tbl5_s=table(Tbl5.Cadencia,Tbl5.Inclinacion, Tbl5.FCardiaca,
Tbl5.Potencia,...

'VariableNames',{'Cadencia','Inclinacion','FCardiaca','Potencia'})
;
Tbl6_s=table(Tbl6.Cadencia,Tbl6.Inclinacion, Tbl6.FCardiaca,
Tbl6.Potencia,...

'VariableNames',{'Cadencia','Inclinacion','FCardiaca','Potencia'})
;
Tbl7_s=table(Tbl7.Cadencia,Tbl7.Inclinacion, Tbl7.FCardiaca,
Tbl7.Potencia,...
```

```
'VariableNames',{'Cadencia','Inclinacion','FCardiaca','Potencia'})
;
Tbl8_s=table(Tbl8.Cadencia,Tbl8.Inclinacion, Tbl8.FCardiaca,
Tbl8.Potencia,...

'VariableNames',{'Cadencia','Inclinacion','FCardiaca','Potencia'})
;

%% ELEGIMOS ACTIVIDAD: 'MODELO_A1 - A8', cambiando los datos de
actividad: 'Tbl1_s'

% actividа1
model_fun= @(b,x)b(1)+b(2)*x(:,2)+b(3)*x(:,1)+
b(4)*(x(:,3).*x(:,3))
beta0 = [1, 1,1,1];
MODELO_A1= fitnlm(Tbl1_s,model_fun,beta0)
Y_A1=predict(MODELO_A1, Tbl1_s);

for i=1 : size(Y_A1)
    if Y_A1(i)<=0
        Y_A1(i)=0;

    end
end
e_A1=Y_A1-Y_REAL_A1;

% actividа2
Y_A2=predict(MODELO_A1, Tbl2_s);

for i=1 : size(Y_A2)
    if Y_A2(i)<=0
        Y_A2(i)=0;

    end
end
e_A2=Y_A2-Y_REAL_A2;
% actividа3

Y_A3=predict(MODELO_A1, Tbl3_s);

for i=1 : size(Y_A3)
    if Y_A3(i)<=0
        Y_A3(i)=0;

    end
end
e_A3=Y_A3-Y_REAL_A3;
% actividа4

Y_A4=predict(MODELO_A1, Tbl4_s);

for i=1 : size(Y_A4)
    if Y_A4(i)<=0
        Y_A4(i)=0;
```

```
    end
end
e_A4=Y_A4-Y_REAL_A4;
% actividа5

Y_A5=predict(MODELO_A1, Tbl5_s);

for i=1 : size(Y_A5)
    if Y_A5(i)<=0
        Y_A5(i)=0;

    end
end
e_A5=Y_A5-Y_REAL_A5;
% actividа6

Y_A6=predict(MODELO_A1, Tbl6_s);

for i=1 : size(Y_A6)
    if Y_A6(i)<=0
        Y_A6(i)=0;

    end
end
e_A6=Y_A6-Y_REAL_A6;
% actividа7

Y_A7=predict(MODELO_A1, Tbl7_s);

for i=1 : size(Y_A7)
    if Y_A7(i)<=0
        Y_A7(i)=0;

    end
end
e_A7=Y_A7-Y_REAL_A7;
% actividа8
Y_A8=predict(MODELO_A1, Tbl8_s);

for i=1 : size(Y_A8)
    if Y_A8(i)<=0
        Y_A8(i)=0;

    end
end
e_A8=Y_A8-Y_REAL_A8;

RMSE_A1=sqrt(mse(Y_A1, Y_REAL_A1))
RMSE_A2=sqrt(mse(Y_A2, Y_REAL_A2))
RMSE_A3=sqrt(mse(Y_A3, Y_REAL_A3))
RMSE_A4=sqrt(mse(Y_A4, Y_REAL_A4))
RMSE_A5=sqrt(mse(Y_A5, Y_REAL_A5))
RMSE_A6=sqrt(mse(Y_A6, Y_REAL_A6))
RMSE_A7=sqrt(mse(Y_A7, Y_REAL_A7))
RMSE_A8=sqrt(mse(Y_A8, Y_REAL_A8))
MAD_A1=mad(e_A1')
```

```
MAD_A2=mad(e_A2')
MAD_A3=mad(e_A3')
MAD_A4=mad(e_A4')
MAD_A5=mad(e_A5')
MAD_A6=mad(e_A6')
MAD_A7=mad(e_A7')
MAD_A8=mad(e_A8')
POT_MEDIA_A1=mean(Y_A1)
POT_MEDIA_A2=mean(Y_A2)
POT_MEDIA_A3=mean(Y_A3)
POT_MEDIA_A4=mean(Y_A4)
POT_MEDIA_A5=mean(Y_A5)
POT_MEDIA_A6=mean(Y_A6)
POT_MEDIA_A7=mean(Y_A7)
POT_MEDIA_A8=mean(Y_A8)
MAE_A1= mae(e_A1);
MAE_A2= mae(e_A2);
MAE_A3= mae(e_A3);
MAE_A4= mae(e_A4);
MAE_A5= mae(e_A5);
MAE_A6= mae(e_A6);
MAE_A7= mae(e_A7);
MAE_A8= mae(e_A8);

TB_VAL=[RMSE_A1 RMSE_A2 RMSE_A3 RMSE_A4 RMSE_A5 RMSE_A6 RMSE_A7
RMSE_A8;
    MAD_A1 MAD_A2 MAD_A3 MAD_A4 MAD_A5 MAD_A6 MAD_A7 MAD_A8;
    MAE_A1 MAE_A2 MAE_A3 MAE_A4 MAE_A5 MAE_A6 MAE_A7 MAE_A8;
    POT_MEDIA_A1 POT_MEDIA_A2 POT_MEDIA_A3 POT_MEDIA_A4
POT_MEDIA_A5 POT_MEDIA_A6 POT_MEDIA_A7 POT_MEDIA_A8]
```

- Árbol de regresión

```
%% VALIDACIÓN: ÁRBOL DE REGRESIÓN
clear;
load Tbl1;load Tbl2;load Tbl3;load Tbl4;load Tbl5;load Tbl6;load
Tbl7;load Tbl8;
load tiempo1;load tiempo2;load tiempo3;load tiempo4;load
tiempo5;load tiempo6;load tiempo7;load tiempo8;

Y_REAL_A1=Tbl1.Potencia;
Y_REAL_A2=Tbl2.Potencia;
Y_REAL_A3=Tbl3.Potencia;
Y_REAL_A4=Tbl4.Potencia;
Y_REAL_A5=Tbl5.Potencia;
Y_REAL_A6=Tbl6.Potencia;
Y_REAL_A7=Tbl7.Potencia;
Y_REAL_A8=Tbl8.Potencia;

Tbl1_s=table(Tbl1.Cadencia,Tbl1.Inclinacion, Tbl1.FCardiaca,
Tbl1.Potencia,...
```

```
'VariableNames',{'Cadencia','Inclinacion','FCardiaca','Potencia'})
;
Tbl2_s=table(Tbl2.Cadencia,Tbl2.Inclinacion, Tbl2.FCardiaca,
Tbl2.Potencia,...

'VariableNames',{'Cadencia','Inclinacion','FCardiaca','Potencia'})
;
Tbl3_s=table(Tbl3.Cadencia,Tbl3.Inclinacion, Tbl3.FCardiaca,
Tbl3.Potencia,...

'VariableNames',{'Cadencia','Inclinacion','FCardiaca','Potencia'})
;
Tbl4_s=table(Tbl4.Cadencia,Tbl4.Inclinacion, Tbl4.FCardiaca,
Tbl4.Potencia,...

'VariableNames',{'Cadencia','Inclinacion','FCardiaca','Potencia'})
;
Tbl5_s=table(Tbl5.Cadencia,Tbl5.Inclinacion, Tbl5.FCardiaca,
Tbl5.Potencia,...

'VariableNames',{'Cadencia','Inclinacion','FCardiaca','Potencia'})
;
Tbl6_s=table(Tbl6.Cadencia,Tbl6.Inclinacion, Tbl6.FCardiaca,
Tbl6.Potencia,...

'VariableNames',{'Cadencia','Inclinacion','FCardiaca','Potencia'})
;
Tbl7_s=table(Tbl7.Cadencia,Tbl7.Inclinacion, Tbl7.FCardiaca,
Tbl7.Potencia,...

'VariableNames',{'Cadencia','Inclinacion','FCardiaca','Potencia'})
;
Tbl8_s=table(Tbl8.Cadencia,Tbl8.Inclinacion, Tbl8.FCardiaca,
Tbl8.Potencia,...

'VariableNames',{'Cadencia','Inclinacion','FCardiaca','Potencia'})
;

load MODELOS_ARBOL_VALIDACION

%% Cargamos el modelo de la actividad correspondiente:
MODELO_ARBOL_A1 - A8

% actividad 1
Y_A1 = MODELO_ARBOL_A1.predictFcn(Tbl1_s);
for i=1 : size(Y_A1)
    if Y_A1(i)<=0
        Y_A1(i)=0;

    end
end
e_A1=Y_A1-Y_REAL_A1;

% actividad 2
Y_A2 = MODELO_ARBOL_A1.predictFcn(Tbl2_s);
```

```
for i=1 : size(Y_A2)
    if Y_A2(i)<=0
        Y_A2(i)=0;

    end
end
e_A2=Y_A2-Y_REAL_A2;
% actividad 3

Y_A3 = MODELO_ARBOL_A1.predictFcn(Tbl3_s);
for i=1 : size(Y_A3)
    if Y_A3(i)<=0
        Y_A3(i)=0;

    end
end
e_A3=Y_A3-Y_REAL_A3;

% actividad 4
Y_A4 = MODELO_ARBOL_A1.predictFcn(Tbl4_s);
for i=1 : size(Y_A4)
    if Y_A4(i)<=0
        Y_A4(i)=0;

    end
end
e_A4=Y_A4-Y_REAL_A4;

% actividad 5
Y_A5 = MODELO_ARBOL_A1.predictFcn(Tbl5_s);
for i=1 : size(Y_A5)
    if Y_A5(i)<=0
        Y_A5(i)=0;

    end
end
e_A5=Y_A5-Y_REAL_A5;

% actividad 6
Y_A6 = MODELO_ARBOL_A1.predictFcn(Tbl6_s);
for i=1 : size(Y_A6)
    if Y_A6(i)<=0
        Y_A6(i)=0;

    end
end
e_A6=Y_A6-Y_REAL_A6;

% actividad 7
Y_A7 = MODELO_ARBOL_A1.predictFcn(Tbl7_s);
for i=1 : size(Y_A7)
    if Y_A7(i)<=0
        Y_A7(i)=0;

    end
end
```

```
e_A7=Y_A7-Y_REAL_A7;

% actividad 8
Y_A8 = MODELO_ARBOL_A1.predictFcn(Tbl8_s);
for i=1 : size(Y_A8)
    if Y_A8(i)<=0
        Y_A81(i)=0;

    end
end
e_A8=Y_A8-Y_REAL_A8;

RMSE_A1=sqrt(mse(Y_A1, Y_REAL_A1))
RMSE_A2=sqrt(mse(Y_A2, Y_REAL_A2))
RMSE_A3=sqrt(mse(Y_A3, Y_REAL_A3))
RMSE_A4=sqrt(mse(Y_A4, Y_REAL_A4))
RMSE_A5=sqrt(mse(Y_A5, Y_REAL_A5))
RMSE_A6=sqrt(mse(Y_A6, Y_REAL_A6))
RMSE_A7=sqrt(mse(Y_A7, Y_REAL_A7))
RMSE_A8=sqrt(mse(Y_A8, Y_REAL_A8))
MAD_A1=mad(e_A1)
MAD_A2=mad(e_A2)
MAD_A3=mad(e_A3)
MAD_A4=mad(e_A4)
MAD_A5=mad(e_A5)
MAD_A6=mad(e_A6)
MAD_A7=mad(e_A7)
MAD_A8=mad(e_A8)
POT_MEDIA_A1=mean(Y_A1)
POT_MEDIA_A2=mean(Y_A2)
POT_MEDIA_A3=mean(Y_A3)
POT_MEDIA_A4=mean(Y_A4)
POT_MEDIA_A5=mean(Y_A5)
POT_MEDIA_A6=mean(Y_A6)
POT_MEDIA_A7=mean(Y_A7)
POT_MEDIA_A8=mean(Y_A8)
MAE_A1= mae(e_A1);
MAE_A2= mae(e_A2);
MAE_A3= mae(e_A3);
MAE_A4= mae(e_A4);
MAE_A5= mae(e_A5);
MAE_A6= mae(e_A6);
MAE_A7= mae(e_A7);
MAE_A8= mae(e_A8);
TB_VAL=[RMSE_A1 RMSE_A2 RMSE_A3 RMSE_A4 RMSE_A5 RMSE_A6 RMSE_A7
RMSE_A8;
    MAD_A1 MAD_A2 MAD_A3 MAD_A4 MAD_A5 MAD_A6 MAD_A7 MAD_A8;
    MAE_A1 MAE_A2 MAE_A3 MAE_A4 MAE_A5 MAE_A6 MAE_A7 MAE_A8;
    POT_MEDIA_A1 POT_MEDIA_A2 POT_MEDIA_A3 POT_MEDIA_A4
POT_MEDIA_A5 POT_MEDIA_A6 POT_MEDIA_A7 POT_MEDIA_A8]
```

- **Red Neuronal**

```
%% VALIDACION: NEURAL NETWORK
clear;
load Tbl1;load Tbl2;load Tbl3;load Tbl4;load Tbl5;load Tbl6;load
Tbl7;load Tbl8;
load DATOS1_SINPOTENCIA;load DATOS2_SINPOTENCIA;load
DATOS3_SINPOTENCIA;load DATOS4_SINPOTENCIA;
load DATOS5_SINPOTENCIA;load DATOS6_SINPOTENCIA;load
DATOS7_SINPOTENCIA;load DATOS8_SINPOTENCIA;
load tiempo1;load tiempo2;load tiempo3;load tiempo4;load
tiempo5;load tiempo6;load tiempo7;load tiempo8;
Y_REAL_A1=Tbl1.Potencia;
Y_REAL_A2=Tbl2.Potencia;
Y_REAL_A3=Tbl3.Potencia;
Y_REAL_A4=Tbl4.Potencia;
Y_REAL_A5=Tbl5.Potencia;
Y_REAL_A6=Tbl6.Potencia;
Y_REAL_A7=Tbl7.Potencia;
Y_REAL_A8=Tbl8.Potencia;
Tbl1_s=[DATOS1_SINPOTENCIA(:,2), DATOS1_SINPOTENCIA(:,3),
DATOS1_SINPOTENCIA(:,4)];
Tbl2_s=[DATOS2_SINPOTENCIA(:,2), DATOS2_SINPOTENCIA(:,3),
DATOS2_SINPOTENCIA(:,4)];
Tbl3_s=[DATOS3_SINPOTENCIA(:,2), DATOS3_SINPOTENCIA(:,3),
DATOS3_SINPOTENCIA(:,4)];
Tbl4_s=[DATOS4_SINPOTENCIA(:,2), DATOS4_SINPOTENCIA(:,3),
DATOS4_SINPOTENCIA(:,4)];
Tbl5_s=[DATOS5_SINPOTENCIA(:,2), DATOS5_SINPOTENCIA(:,3),
DATOS5_SINPOTENCIA(:,4)];
Tbl6_s=[DATOS6_SINPOTENCIA(:,2), DATOS6_SINPOTENCIA(:,3),
DATOS6_SINPOTENCIA(:,4)];
Tbl7_s=[DATOS7_SINPOTENCIA(:,2), DATOS7_SINPOTENCIA(:,3),
DATOS7_SINPOTENCIA(:,4)];
Tbl8_s=[DATOS8_SINPOTENCIA(:,2), DATOS8_SINPOTENCIA(:,3),
DATOS8_SINPOTENCIA(:,4)];

%% Para entrenar el modelo con actividades, cambiamos: Tbl1_s

trainFcn = 'trainlm';
hiddenLayerSize = 30;
net = fitnet(hiddenLayerSize,trainFcn);
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;
[net,tr] = train(net,Tbl1_s',Y_REAL_A1');
%Actividad 1
Y_A1 = net(Tbl1_s');
for i=1 : length(Y_A1)
    if Y_A1(i)<=0
        Y_A1(i)=0;
    end
end
e_A1= gsubtract(Y_REAL_A1',Y_A1);
RMSE_A1 = sqrt(perform(net,Y_REAL_A1',Y_A1))
```

```
MAD_A1=mad(e_A1')
%Actividad 2
Y_A2 = net(Tbl2_s');
for i=1 : length(Y_A2)
    if Y_A2(i)<=0
        Y_A2(i)=0;
    end
end
e_A2= gsubtract(Y_REAL_A2',Y_A2);
RMSE_A2 = sqrt(perform(net,Y_REAL_A2',Y_A2))
MAD_A2=mad(e_A2')
%Actividad 3
Y_A3 = net(Tbl3_s');
for i=1 : length(Y_A3)
    if Y_A3(i)<=0
        Y_A3(i)=0;
    end
end
e_A3= gsubtract(Y_REAL_A3',Y_A3);
RMSE_A3 = sqrt(perform(net,Y_REAL_A3',Y_A3))
MAD_A3=mad(e_A3')
%Actividad 4
Y_A4 = net(Tbl4_s');
for i=1 : length(Y_A4)
    if Y_A4(i)<=0
        Y_A4(i)=0;
    end
end
e_A4= gsubtract(Y_REAL_A4',Y_A4);
RMSE_A4 = sqrt(perform(net,Y_REAL_A4',Y_A4))
MAD_A4=mad(e_A4')
%Actividad 5
Y_A5 = net(Tbl5_s');
for i=1 : length(Y_A5)
    if Y_A5(i)<=0
        Y_A5(i)=0;
    end
end
e_A5= gsubtract(Y_REAL_A5',Y_A5);
RMSE_A5 = sqrt(perform(net,Y_REAL_A5',Y_A5))
MAD_A5=mad(e_A5')
%Actividad 6
Y_A6 = net(Tbl6_s');
for i=1 : length(Y_A6)
    if Y_A6(i)<=0
        Y_A6(i)=0;
    end
end
e_A6= gsubtract(Y_REAL_A6',Y_A6);
RMSE_A6 = sqrt(perform(net,Y_REAL_A6',Y_A6))
MAD_A6=mad(e_A6')
%Actividad 7
Y_A7 = net(Tbl7_s');
for i=1 : length(Y_A7)
    if Y_A7(i)<=0
        Y_A7(i)=0;
```

```
    end
end
e_A7= gsubtract(Y_REAL_A7',Y_A7);
RMSE_A7 = sqrt(perform(net,Y_REAL_A7',Y_A7))
MAD_A7=mad(e_A7')
%Actividad 8
Y_A8 = net(Tbl8_s');
for i=1 : length(Y_A8)
    if Y_A8(i)<=0
        Y_A8(i)=0;
    end
end
e_A8= gsubtract(Y_REAL_A8',Y_A8);
RMSE_A8 = sqrt(perform(net,Y_REAL_A8',Y_A8))
MAD_A8=mad(e_A8')
POT_MEDIA_A1=mean(Y_A1)
POT_MEDIA_A2=mean(Y_A2)
POT_MEDIA_A3=mean(Y_A3)
POT_MEDIA_A4=mean(Y_A4)
POT_MEDIA_A5=mean(Y_A5)
POT_MEDIA_A6=mean(Y_A6)
POT_MEDIA_A7=mean(Y_A7)
POT_MEDIA_A8=mean(Y_A8)
MAE_A1= mae(e_A1');
MAE_A2= mae(e_A2');
MAE_A3= mae(e_A3');
MAE_A4= mae(e_A4');
MAE_A5= mae(e_A5');
MAE_A6= mae(e_A6');
MAE_A7= mae(e_A7');
MAE_A8= mae(e_A8');

TB_RMSE=[RMSE_A1 RMSE_A2 RMSE_A3 RMSE_A4 RMSE_A5 RMSE_A6 RMSE_A7
RMSE_A8;
  MAD_A1 MAD_A2 MAD_A3 MAD_A4 MAD_A5 MAD_A6 MAD_A7 MAD_A8;
  MAE_A1 MAE_A2 MAE_A3 MAE_A4 MAE_A5 MAE_A6 MAE_A7 MAE_A8;
  POT_MEDIA_A1 POT_MEDIA_A2 POT_MEDIA_A3 POT_MEDIA_A4 POT_MEDIA_A5
POT_MEDIA_A6 POT_MEDIA_A7 POT_MEDIA_A8]
```

- **ANFIS**

```
%% VALIDACIÓN: ANFIS
clear;
load Tbl1;load Tbl2;load Tbl3;load Tbl4;load Tbl5;load Tbl6;load
Tbl7;load Tbl8;
load tiempo1;load tiempo2;load tiempo3;load tiempo4;load
tiempo5;load tiempo6;load tiempo7;load tiempo8;
Y_REAL_A1=Tbl1.Potencia;
Y_REAL_A2=Tbl2.Potencia;
Y_REAL_A3=Tbl3.Potencia;
Y_REAL_A4=Tbl4.Potencia;
Y_REAL_A5=Tbl5.Potencia;
Y_REAL_A6=Tbl6.Potencia;
```

```
Y_REAL_A7=Tbl7.Potencia;
Y_REAL_A8=Tbl8.Potencia;
load DATOS1_COMPLETO.dat;
load DATOS2_COMPLETO.dat;
load DATOS3_COMPLETO.dat;
load DATOS4_COMPLETO.dat;
load DATOS5_COMPLETO.dat;
load DATOS6_COMPLETO.dat;
load DATOS7_COMPLETO.dat;
load DATOS8_COMPLETO.dat;

% ELIMINAMOS LA VARIBLE VELOCIDAD
DATOS_A1=DATOS1_COMPLETO;
DATOS_A1(:,1)=[];
DATOS_A2=DATOS2_COMPLETO;
DATOS_A2(:,1)=[];
DATOS_A3=DATOS3_COMPLETO;
DATOS_A3(:,1)=[];
DATOS_A4=DATOS4_COMPLETO;
DATOS_A4(:,1)=[];
DATOS_A5=DATOS5_COMPLETO;
DATOS_A5(:,1)=[];
DATOS_A6=DATOS6_COMPLETO;
DATOS_A6(:,1)=[];
DATOS_A7=DATOS7_COMPLETO;
DATOS_A7(:,1)=[];
DATOS_A8=DATOS8_COMPLETO;
DATOS_A8(:,1)=[];

% PARTIMOS LOS DATOS PARA EL TRAINING Y EL TESTEO
j=1;
for i=1 : length(DATOS_A1)
    if i<=8474
        DATOS_A1_TR(i,:)=DATOS_A1(i,:);
    else
        DATOS_A1_TS(j,:)=DATOS_A1(i,:);
        j=j+1;

    end
end

j=1;
for i=1 : length(DATOS_A2)
    if i<=10740
       DATOS_A2_TR(i,:)=DATOS_A2(i,:);
    else
       DATOS_A2_TS(j,:)=DATOS_A2(i,:);
        j=j+1;

    end
end
j=1;
for i=1 : length(DATOS_A3)
    if i<=10850
        DATOS_A3_TR(i,:)=DATOS_A3(i,:);
    else
```

```
        DATOS_A3_TS(j,:)=DATOS_A3(i,:);
        j=j+1;

    end
end
j=1;
for i=1 : length(DATOS_A4)
    if i<=6459
        DATOS_A4_TR(i,:)=DATOS_A4(i,:);
    else
        DATOS_A4_TS(j,:)=DATOS_A4(i,:);
        j=j+1;

    end
end
j=1;
for i=1 : length(DATOS_A5)
    if i<=10038
        DATOS_A5_TR(i,:)=DATOS_A5(i,:);
    else
        DATOS_A5_TS(j,:)=DATOS_A5(i,:);
        j=j+1;

    end
end
j=1;
for i=1 : length(DATOS_A6)
    if i<=7735
        DATOS_A6_TR(i,:)=DATOS_A6(i,:);
    else
        DATOS_A6_TS(j,:)=DATOS_A6(i,:);
        j=j+1;

    end
end
j=1;
for i=1 : length(DATOS_A7)
    if i<=7973
        DATOS_A7_TR(i,:)=DATOS_A7(i,:);
    else
        DATOS_A7_TS(j,:)=DATOS_A7(i,:);
        j=j+1;

    end
end
j=1;
for i=1 : length(DATOS_A8)
    if i<=3800
        DATOS_A8_TR(i,:)=DATOS_A8(i,:);
    else
        DATOS_A8_TS(j,:)=DATOS_A8(i,:);
        j=j+1;

    end
end
load MODELOS_ANFIS_VALIDACION;
```

```
%% Para generar los resultados de diferentes actividades
cambiamos:
    % MODELO_ANFIS_A1 - A8
    % DATOS_EVAL_A1 - A8
    % DATOS_A1 - A8
 % ACTIVIDAD 1
DATOS_EVAL_A1=DATOS_A1;
DATOS_EVAL_A1(:,4)=[];
Y_A1=evalfis(MODELO_ANFIS_A1,DATOS_EVAL_A1);

for i=1 : size(Y_A1)
    if Y_A1(i)<=0
        Y_A1(i)=0;

    end
end
e_A1=Y_A1-Y_REAL_A1;

%ACTIVIDAD2
DATOS_EVAL_A2=DATOS_A2;
DATOS_EVAL_A2(:,4)=[];
Y_A2=evalfis(MODELO_ANFIS_A1,DATOS_EVAL_A2);

for i=1 : size(Y_A2)
    if Y_A2(i)<=0
        Y_A2(i)=0;

    end
end
e_A2=Y_A2-Y_REAL_A2;

%ACTIVIDAD3
DATOS_EVAL_A3=DATOS_A3;
DATOS_EVAL_A3(:,4)=[];
Y_A3=evalfis(MODELO_ANFIS_A1,DATOS_EVAL_A3);

for i=1 : size(Y_A3)
    if Y_A3(i)<=0
        Y_A3(i)=0;

    end
end
e_A3=Y_A3-Y_REAL_A3;

%ACTIVIDAD4
DATOS_EVAL_A4=DATOS_A4;
DATOS_EVAL_A4(:,4)=[];
Y_A4=evalfis(MODELO_ANFIS_A1,DATOS_EVAL_A4);

for i=1 : size(Y_A4)
    if Y_A4(i)<=0
        Y_A4(i)=0;

    end
end
```

```
e_A4=Y_A4-Y_REAL_A4;

%ACTIVIDAD5
DATOS_EVAL_A5=DATOS_A5;
DATOS_EVAL_A5(:,4)=[];
Y_A5=evalfis(MODELO_ANFIS_A1,DATOS_EVAL_A5);

for i=1 : size(Y_A5)
    if Y_A5(i)<=0
        Y_A5(i)=0;

    end
end
e_A5=Y_A5-Y_REAL_A5;

%ACTIVIDAD6
DATOS_EVAL_A6=DATOS_A6;
DATOS_EVAL_A6(:,4)=[];
Y_A6=evalfis(MODELO_ANFIS_A1,DATOS_EVAL_A6);

for i=1 : size(Y_A6)
    if Y_A6(i)<=0
        Y_A6(i)=0;

    end
end
e_A6=Y_A6-Y_REAL_A6;
%ACTIVIDAD7
DATOS_EVAL_A7=DATOS_A7;
DATOS_EVAL_A7(:,4)=[];
Y_A7=evalfis(MODELO_ANFIS_A1,DATOS_EVAL_A7);

for i=1 : size(Y_A7)
    if Y_A7(i)<=0
        Y_A7(i)=0;

    end
end
e_A7=Y_A7-Y_REAL_A7;

%ACTIVIDAD8
DATOS_EVAL_A8=DATOS_A8;
DATOS_EVAL_A8(:,4)=[];
Y_A8=evalfis(MODELO_ANFIS_A1,DATOS_EVAL_A8);

for i=1 : size(Y_A8)
    if Y_A8(i)<=0
        Y_A8(i)=0;

    end
end
e_A8=Y_A8-Y_REAL_A8;

RMSE_A1=sqrt(mse(Y_A1, Y_REAL_A1))
RMSE_A2=sqrt(mse(Y_A2, Y_REAL_A2))
RMSE_A3=sqrt(mse(Y_A3, Y_REAL_A3))
```

```
RMSE_A4=sqrt(mse(Y_A4, Y_REAL_A4))
RMSE_A5=sqrt(mse(Y_A5, Y_REAL_A5))
RMSE_A6=sqrt(mse(Y_A6, Y_REAL_A6))
RMSE_A7=sqrt(mse(Y_A7, Y_REAL_A7))
RMSE_A8=sqrt(mse(Y_A8, Y_REAL_A8))
MAD_A1=mad(e_A1')
MAD_A2=mad(e_A2')
MAD_A3=mad(e_A3')
MAD_A4=mad(e_A4')
MAD_A5=mad(e_A5')
MAD_A6=mad(e_A6')
MAD_A7=mad(e_A7')
MAD_A8=mad(e_A8')
POT_MEDIA_A1=mean(Y_A1)
POT_MEDIA_A2=mean(Y_A2)
POT_MEDIA_A3=mean(Y_A3)
POT_MEDIA_A4=mean(Y_A4)
POT_MEDIA_A5=mean(Y_A5)
POT_MEDIA_A6=mean(Y_A6)
POT_MEDIA_A7=mean(Y_A7)
POT_MEDIA_A8=mean(Y_A8)
MAE_A1= mae(e_A1);
MAE_A2= mae(e_A2);
MAE_A3= mae(e_A3);
MAE_A4= mae(e_A4);
MAE_A5= mae(e_A5);
MAE_A6= mae(e_A6);
MAE_A7= mae(e_A7);
MAE_A8= mae(e_A8);
TB_VAL=[RMSE_A1 RMSE_A2 RMSE_A3 RMSE_A4 RMSE_A5 RMSE_A6 RMSE_A7
RMSE_A8;
    MAD_A1 MAD_A2 MAD_A3 MAD_A4 MAD_A5 MAD_A6 MAD_A7 MAD_A8;
    MAE_A1 MAE_A2 MAE_A3 MAE_A4 MAE_A5 MAE_A6 MAE_A7 MAE_A8;
    POT_MEDIA_A1 POT_MEDIA_A2 POT_MEDIA_A3 POT_MEDIA_A4
POT_MEDIA_A5 POT_MEDIA_A6 POT_MEDIA_A7 POT_MEDIA_A8]
```

Anexo 4: Script para generar gráficas de errores

```
%% Generamos resultados aleatorios
plot(tiempo5, Y_REAL_A5,'red')
hold on;
plot(tiempo5, Y_A5,'black')
legend('Potencia Act.5','Potencia ANFIS')
title('Potencia predicha por el modelo frente a la potencia real')
xlabel('Tiempo (s)')
ylabel('Potencia (W)')

figure;
Error_medio_A5=mean(e_A5)
% error
plot(e_A5,'-*');
```

```
% Densidad de probabilidad
N=length(e_A5)
n_cajas=30;
e_maximo=max(e_A5);
e_minimo=min(e_A5);
d=(e_maximo-e_minimo)/n_cajas;
caja=zeros(n_cajas,1);
for i=1 : 1 : N
for k=1 : 1 : n_cajas
ep(k) = e_minimo+(k-0.5)*d;
if (e_A5(i) >= e_minimo+(k-1)*d) && (e_A5(i) <= e_minimo+k*d)
caja(k) = caja(k)+1;
end
end
end
caja=caja/N;
figure;
plot(ep,caja,'*');

%Funcion de autocorrelación
e_corr=e_A5-mean(e_A5);
Re=correlacion(e_corr,2000);
figure;
plot(Re);
title('FUNCION DE AUTOCORRELACION');

%PSD
[Ge,f]=espectro(e_A5,0.001,0);
figure;
plot(f,Ge,'*');
title('PSD')

% Agrupación
subplot(2,2,1)
plot(ep,caja,'*');
title('Función de densidad de probabilidad');
subplot(2,2,2)
plot(Re);
title('Función de autocorrelación');
subplot(2,2,3)
plot(f,Ge,'*');
title('PSD');
subplot(2,2,4)
plot(e_A5,'-*');
title('Error actividad 5')
```