Revista Iberoamericana de Automática e Informática Industrial

Revista Iberoamericana de Automática e Informática Industrial 16 (2019) 114-127

www.revista-riai.org

Aprendizaje Automático en la Identificación de Sistemas. Un Caso de Estudio en la Predicción de la Generación Eléctrica de un Parque Eólico

Aguilar, R.M.*, Torres, J.M., Martín, C.A.

Departamento de Ingeniería Informática y de Sistemas. Universidad de La Laguna, Camino San Francisco de Paula, s/n 38271, La Laguna, España.

Resumen

Uno de los mayores desafíos tecnológicos de la actualidad es la obtención de modelos predictivos de sistemas complejos. En este artículo se propone darle valor a los datos recogidos sobre un proceso utilizándolos para la identificación del mismo mediante el empleo de algoritmos de aprendizaje automático. En concreto, se describe el desarrollo de un proyecto de determinación del modelo predictivo de un sistema, a partir de algoritmos de aprendizaje automático supervisado, usando como ejemplo el problema de determinar la generación de energía de un campo eólico. Para ello se estudian las transformaciones a realizar a los datos recogidos, la búsqueda del mejor algoritmo, cómo determinar la bondad del mismo y, finalmente, el entrenamiento y ajuste del modelo seleccionado. Todo ello usando el lenguaje de programación Python, que dispone de librerías que facilitan este tipo de proyectos, y en el entorno de Jupyter Notebook para realiza el proyecto y divulgar los resultados.

Palabras Clave:
Identificación de sistemas y estimación de parámetros, Incertidumbre de modelado, Inteligencia Artificial, Aprendizaje Automático

Automatic learning for the system identification. A case study in the prediction of power generation in a wind farm

Abstract

One of the greatest technical challenges of today is obtaining predictive models for complex systems. In this paper we propose using data collected during a process to identify said process by means of automatic learning algorithms. Specifically, we describe the development of a project to determine the predictive model of a system based on supervised automatic learning algorithms. As an example, we use the problem of determining the energy generated in a wind farm. We do so by studying how the data collected are transformed, the search for the best algorithm, how to determine its goodness, and finally, the training and adjustment of the selected model. This study relies on the Python programming language, which has libraries that facilitate this type of project, and the Jupyter Notebook environment to carry out the project and disseminate the results.

Keywords:
System Identification and Parameter Estimation, Modeling uncertainty, Artificial intelligence, Machine Learning

1. Introducción

La identificación de sistemas complejos consiste en la obtención de un modelo matemático que nos permita reproducir algunas propiedades del sistema original para su estudio (Ljung, 1998). Pero si consideramos que los sistemas complejos son aquellos que están formados por muchas partes interactuando entre sí o con pocas pero cuyo comportamiento resulta difícil de predecir; un sistema será tanto más complejo cuantas más partes tenga, más conexiones se den entre ellas o más dinámicas desconocidas presenten, necesitando por lo tanto más tiempo de estudio y de cómputo. Como los componentes del sistema no se pueden separar sin destruirlo, el método de análisis por descomposición en módulos independientes no se puede usar para el estudio o simplificación de tales sistemas. Esto significa, que los sistemas complejos son difíciles de modelar y los modelos resultantes son difíciles de usar para la predicción o el control y que, por lo tanto, son sistemas difíciles de resolver (Piñuela-Martín et al., 2016).

*Autor para correspondencia: raguilar@ull.edu.es
To cite this article: Aguilar R.M., Torres J.M., Martín C.A. 2019. Machine Learning for System Identification. A Case Study in Wind Farm Generation. Revista Iberoamericana de Automática e Informática Industrial 16, 114-127. https://doi.org/10.4995/riai.2018.9421

En la actualidad, con la industria conectada 4.0 donde todos los procesos están interconectados toma especial relevancia el estudio de este tipo de sistemas complejos.

Ese mismo avance en la conectividad nos ofrece una ventaja importante: la disponibilidad de gran cantidad de información en relación a diferentes procesos y sistemas (tanto industriales como logísticos), servicios (ventas, conexiones entre usuarios, consumo eléctrico, etc.) o tráfico de datos (registros de eventos en enrutadores, servidores y otros equipos). El análisis de estos datos puede proporcionar información muy valiosa acerca del comportamiento de los procesos. Por ejemplo, se pueden prevenir problemas en un determinado proceso industrial a través de la detección de resultados o de medidas anómalas (de forma inteligente, sin tener que haber definido previamente qué medida es o no es anómala) o detectar eventos relacionados dentro de procesos complejos, facilitando su gestión a través de la predicción, al saber de antemano la probabilidad con la que un evento desencadenará otro. A partir de toda esta información se pueden realizar simulaciones que, además, permiten predecir qué recursos van a ser necesarios; pudiendo optimizar su uso de forma automática y proactiva y anticipando los acontecimientos futuros. Para ello se debe trabajar en el campo del *Machine Learning* (ML) o Aprendizaje Automático (Bibault et al., 2016).

Tanto la disciplina de aprendizaje automático como la de identificación de sistemas tienen como objetivo la construcción de modelo a partir de los datos observados (Ljung, 2008). En el caso de la identificación de sistemas, se centra en la obtención de un modelo paramétrico, con estructura fija y, a ser posible, de bajo orden (es decir, el número de coeficientes base relativamente pequeño) que lo mantenga lo más simple posible para evitar problemas computacionales, etc. Este modelo se puede usar para hacer predicciones sobre el comportamiento futuro del sistema a partir de nuevos datos de entrada (Pillonetto et al., 2016). Por otro lado, el aprendizaje automático en su vertiente de algoritmos de regresión también diseña modelos con fines de predicción. La principal diferencia con las técnicas de identificación del sistema es que los algorimtos de ML generan modelos no paramétrico. Esto último significa que la predicción para una nueva entrada se da como una función de los puntos de datos utilizados para el *entrenamiento* (aprendizaje, identificación) del modelo. Los métodos ML son más flexibles ya que no requieren ninguna selección de estructura por parte del usuario, pero las limitaciones a resolver son el esfuerzo de cálculo para la convergencia del método.

Aunque las técnicas de identificación y de ML han evolucionado independientemente, se reconoce la necesidad de establecer un terreno común donde la disciplina de identificación de sistemas se nutra de métodos ampliamente utilizados en otros campos como ML (Ljung et al., 2011). El objetivo de este artículo es avanzar en esta línea de investigación, demostrando como se puede identificar el sistema de predicción de la generación de un campo eólico a partir de técnicas de aprendizaje automático.

El término de *Machine Learning* (ML) o Aprendizaje Automático (Bibault et al., 2016) se acuñó en los años 50, ha adquirido últimamente una gran relevancia debido al enorme aumento de la capacidad de cómputo y al gran volumen de datos que las empresas empiezan a almacenar y manejar. En este sentido, lo que nos aporta el Aprendizaje Automático es un conjunto de algoritmos cuyo objetivo es dotar a los ordenadores de la capacidad de aprender sin la necesidad de ser programados explícitamente.

En este artículo se propone el uso de algoritmos de aprendizaje automático para identificar sistemas a partir de grandes cantidades de datos almacenados sobre los mismos. Es decir, sin tener que hacer explícitas las relaciones existentes entre las distintas partes del sistema. La utilidad de este tipo de algoritmos se estudiará para el caso concreto de la predicción de la generación en un parque eólico (Torres et al., 2018).

1.1. Big Data

No es ninguna novedad que en la actualidad la captura y el almacenamiento de datos tiene un coste muy bajo. Este hecho nos permite disponer de una gran cantidad de datos almacenados. Además del gran volumen de información disponible, existe una gran variedad de tipos de datos que pueden ser representados de diversas maneras. Por ejemplo, audio, video, sistemas GPS, incontables sensores digitales en equipos industriales y dispositivos móviles, automóviles, medidores eléctricos, veletas, anemómetros, etc.; los cuales pueden medir y comunicar la posición, movimiento, vibración, temperatura, humedad y hasta los cambios químicos. Sin embargo, no sólo hay que tener presentes los datos procedentes de tecnologías que permiten conectarse a otros dispositivos *Machine-to-Machine* (M2M); también debemos incluir datos alojados en redes sociales como: Facebook, Twitter, LinkedIn, blogs, etc. Estos nos permiten conocer los gustos y preferencias de los usuarios e incluso, yendo más allá, sus estados de ánimo. También se registran datos procedentes de las transacciones realizadas entre cliente-empresa, donde se incluyen registros de facturación, registros detallados de las llamadas telefónicas, notas de voz, correos electrónicos, documentos electrónicos, estudios médicos, etc.

Estos datos esconden información de gran valor para saber, no sólo lo que sucede a nuestro alrededor, sino también lo que va a pasar en un futuro, en algunos casos con niveles de precisión muy altos. Para poder tratar estas grandes cantidades de datos y darle valor añadido extrayendo el conocimiento existen en ellos, las técnicas tradicionales de estadística y las herramientas de gestión clásicas no sirven, debido a que no están preparadas para trabajar con tantos datos ni tan variados (Peña, 2014). Por lo que se hace evidente la necesidad de nuevas herramientas de análisis.

Big Data es el proceso de recolección de grandes cantidades de datos y su inmediato análisis para encontrar información oculta, patrones recurrentes, correlaciones, etc.. En Big Data el conjunto de datos es tan grande y complejo que los medios tradicionales de procesamiento son ineficaces. Y es que estamos hablando de desafíos como analizar, capturar, recolectar, buscar, compartir, almacenar, transferir, visualizar, etc. ingentes cantidades de información, para obtener conocimiento, muchas veces en tiempo real, poniendo al mismo tiempo especial cuidado en la protección de datos personales por motivos legales. Las características que exhibe este proceso son (Fichman et al., 2014):

- Volumen: Captar y organizar absolutamente toda la información que nos llega es esencial para tener registros

completos y no sesgados y que las conclusiones que obtengamos sirvan eficientemente a la hora de la toma de decisiones.

- Velocidad: Es importante el tiempo si afrontamos tanto la necesidad de generar información como de analizarla. Pero lo es más si necesitamos reaccionar inmediatamente. Todo el proceso pide agilidad para extraer valor de negocio a la información que se estudia sin perder oportunidades.
- Variedad: Hay que dar uniformidad a toda la información, que tendrá su origen en datos de lo más heterogéneos. Una de las fortalezas del Big Data reside en poder conjugar y combinar cada tipo de información y su tratamiento específico para alcanzar un todo homogéneo.
- Veracidad: Hay que considerar la calidad del dato y su disponibilidad, usando herramientas para comprobar la bondad de la información recibida.
- Valor: Trabajar con Big Data tiene que servir para aportar valor a la sociedad, las empresas, los gobiernos y, en definitiva, a las personas. Todo el proceso tiene que ayudar a impulsar el desarrollo, la innovación y la competitividad, pero también mejorar la calidad de vida de las personas.

El análisis de los datos nos puede ofrecer el modelo del proceso que lo ha generado (analítica predictiva). Antes de la irrupción del Big Data, ya existían algoritmos matemáticos que nos facilitaban descubrir información oculta en los datos, como son todos los que se engloban dentro del *Data Mining* o Minería de Datos: K-medias, arboles de decisión, redes neuronales, etc. Con el aumento de la potencia de cálculo de los ordenadores se pudieron acortar los tiempos que tardaban en obtener resultados. Sin embargo, no estaban pensados para ser utilizados en tiempo real y con el volumen tan grande de datos de que se dispone hoy en día.

Actualmente existen herramientas que facilitan obtener, almacenar, manipular e inferir modelos predictivos a partir de grandes cantidades datos de una manera muy intuitiva. En este artículo se presenta un conjunto de algoritmos de ML para la identificación de sistemas con diferentes complejidades. Nos centraremos en el aprendizaje supervisado, que se caracteriza por disponer de una experiencia o conocimiento previo en el que nos podemos apoyar para hacer predicciones o tomar decisiones. Frente a este, se encuentra el aprendizaje no supervisado, donde no tenemos un conocimiento previo de los datos recibidos y, por lo tanto, los algoritmos tratan de buscar estructuras o patrones en los datos.

La descripción de estos algoritmos de ML se realiza sobre un caso de uso: la predicción de la generación de un campo eólico. Y se presenta con la secuenciación necesaria para que sirva de guion en la enseñanza de modelado de sistemas complejos en asignaturas tales como 'Control Inteligente' en titulaciones de posgrado.

Para su incorporación en la docencia, se describe el proceso de identificación utilizando como lenguaje de programación Python (Brunner and Kim, 2016). Este es un lenguaje interpretado de propósito general que es fácil de aprender y que tiene una gran modularidad. Actualmente, es uno de los lenguajes de propósito general más populares y se encuentra en el top 10 según *Stack Overflow Trends*. Es un lenguaje dinámico, adecuado para el desarrollo rápido de prototipo y muy interactivo, por lo que es especialmente útil en la docencia. Todo ello sin perder su potencialidad en el desarrollo de aplicaciones de gran envergadura, lo que facilita la transición de la etapa de prototipado y desarrollo a la de producción. Además, es ampliamente utilizado en proyectos de ML y analítica de datos ya que cuenta con excelentes librerías en este ámbito.

Entre el ecosistema de las librerías de Python para matemáticas, ciencia e ingeniería nos encontramos con SciPy. El núcleo de esta librería está formado por:

- *Numpy*: que permite trabajar de forma eficiente con datos en vectores y matrices.
- *Matplotlib*: para crear gráficos.
- *Pandas*: ofrece herramientas y estructuras para acceder, organizar y analizar los datos

Sin embargo, la librería que permite realizar proyectos de ML en Python es *scikit-learn*, que ofrece algoritmos para clasificación, regresión y clustering. También ofrece herramienta para evaluar los modelos, ajuste de parámetros y pre-procesado de datos.

Para facilitar su uso se propone Jupyter como entorno de trabajo de fuentes abiertas, interactivo via web (La figura 1). Esta herramienta está organizada en pequeños bloques que pueden contener código de programación e ir ejecutándolo paso a paso o todo de golpe, obteniendo todos los resultados parciales. También se pueden incluir en los bloques texto para documentar el código o añadir las explicaciones oportunas, que pueden contener enlaces, imágenes, vídeos u otros elementos.

Esta serie de piezas de código, notas y resultados se guardan en un *notebook*, que es un fichero que contiene toda esta información. Uno de los principales objetivos de Jupyter es fomentar y simplificar la compartición de conocimiento y resultados a través de los block de notas en plataformas como *GitHub*. De esta manera pueden ser fácilmente difundidos y los resultados pueden ser reproducidos y validados en diferentes entornos. Por lo que es muy útil para la divulgación y la formación o en entornos educativos.

El artículo describe en la sección 2 los pasos a seguir en un proyecto de Machine Learning. En la sección 3 se presenta un conjunto de algoritmos de aprendizajes automático supervisado, tanto del grupo de algoritmos lineales (reresión lineal, Rigge, LASSO y ElasticNet), como no no lineales (k-vecinos más próximo, árboles de decisión, máquinas de soporte vectorial y Redes Neuronales Artificiales). La aplicación de estos métodos descritos se realizará sobre el modelado de la predicción de un parque eólico en la sección 4. El código desarrollado para este caso de uso se muestra en los apéndices, existiendo una versión del mismo en GitHub https://github.com/rmaguilarc/Modelos_Predictivos. Finalmente se termina con unas conclusiones.

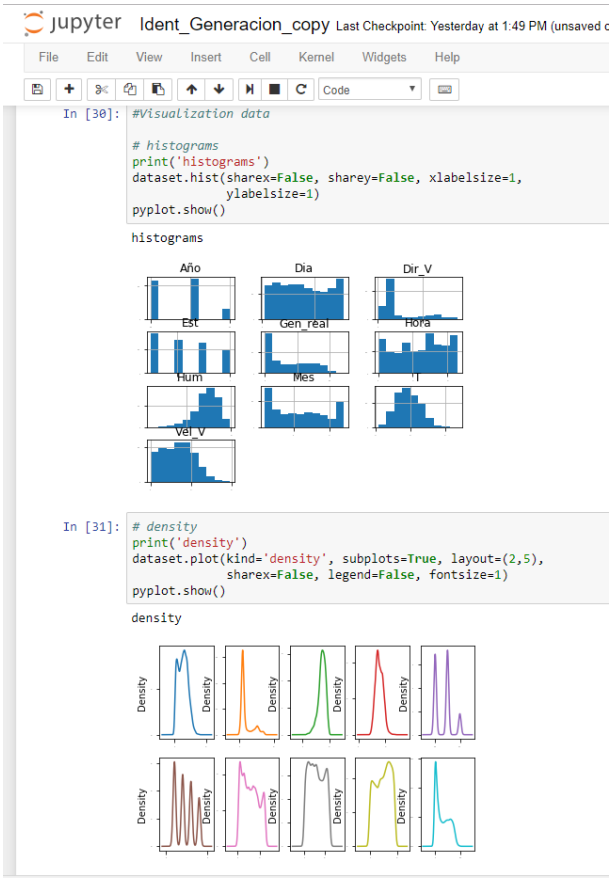

Figura 1: Jupyter Notebook

2. Método para el desarrollo de modelos predictivos

El campo del *Machine Learning* es muy amplio pero en este trabajo vamos a estudiar algoritmos para "modelado predictivo", ya que son muy útiles para la industria (Morris et al., 2018)(Forbes et al., 2015)(Domínguez-Tejo et al., 2018). Asimismo se ilustrará el uso de librerías tales como *scikit-learn* que hacen muy accesible la implementación de tales modelos.

La rama de ML relacionada con modelos predictivos se centra en el desarrollo de modelos que realicen predicciones fiables, a expensas de no conocer las razones de cómo esas predicciones se realizan (método de caja negra). Esto está en contraposición a como trabajaríamos desde la disciplina de la estadística, donde los modelos explicitan las relaciones existentes entre los distintos parámetros del sistema (Peña, 2014). Y aunque en el campo del ML es posible trabajar con datos en cualquier formato, cuando trabajamos en modelos predictivos normalmente se usan datos tabulados. Por ejemplo, tablas con números de velocidad de viento, dirección viento, potencia generada, etc.

La construcción de un modelo predictivo con algoritmos ML lleva asociado la realización de las siguientes 6 tareas:

1. Definir el problema: investigar y caracterizar el problema para tener totalmente identificados los objetivos perseguidos.
2. Analizar los datos: utilizar tanto estadística descriptiva como la visualización de los datos para tener conocimiento sobre el tipo de dato que tenemos disponible
3. Preparar los datos: realizar las transformaciones de datos oportunas para que los algoritmos de modelado puedan encontrar las relaciones entre los datos que permitan, por un lado, que el método converja y, por el otro, mejorar la predicción ofrecida.
4. Evaluar algoritmos: diseñar un test de evaluación de un conjunto de algoritmos estándares en tareas de predicción sobre los datos disponibles, para determinar aquellos que ofrecen mejores resultados.
5. Mejorar resultados: Conocido el algoritmo de ML que mejor resultados ofrece sobre nuestros datos, estudiar distintos ajustes en los parámetros de dicho algoritmo para mejorar su predicción.
6. Presentar resultados: finalmente se deben realizar predicciones con el modelo resultante y estudiar la sensibilidad del mismo para distintos conjuntos de datos.

La primera tarea consiste en determinar cuál es la pregunta que queremos responder con el modelo predictivo. Determinar las señales de entrada al sistema y la salida a predecir. Y localizar el conjunto de Big Data que tenemos disponible para identificar el sistema. La disposición de una gran cantidad de datos del sistema a modelar es condición necesaria (pero no suficiente) en cualquier proyecto de ML.

El proceso de análisis de datos es necesario para conocer cómo son los datos de los que disponemos con la finalidad de poder pre-procesarlos, de manera que mejoren tanto la convergencia de los algoritmos ML como la fiabilidad de los resultados. Para ello se estudian los siguientes ítems:

- Dimensionalidad de los datos: disponer de muchas filas pueden indicar tiempos de computación excesivos. Mientras que pocas podría significar que no se dispone de datos suficientes para entrenar los algoritmos. Por el contrario, si el número de características (columnas) son excesivas puede ocasionar búsquedas más grandes que lleven a un bajo rendimiento de las predicciones. Como se observa en la figura 6.

- Tipo de de atributo: ya que algunos tipos de datos se puede utilizar directamente, mientras que otros deben ser convertidos. Por ejemplo, los *strings* deben ser convertidos en valores reales o enteros que representen categorías o valores ordinales.

- Características estadísticas (cantidad, media, desviación estándar, valor mínimo, valor máximo, etc.) que ofrecen una visión de la forma de cada atributo.

- Correlaciones entre las variables. El método más común es el Coeficiente de Correlación de Pearson que asume distribuciones normales para los atributos (Bermudez-Edo et al., 2018). Así una correlación de -1 o 1 muestra una correlación negativa o positiva respectivamente. Mientras que un valor de 0 indica que no existe correlación. Algunos algoritmos de ML, tales como la regresión lineal o la regresión logística, presentan un bajo rendimiento si los atributos están altamente correlacionados.

Como tercer punto se preparan los datos para conseguir mejores resultados. Se pueden utilizar las siguientes transformaciones:

- Escalar: cuando los datos tiene diferentes escalas suele resultar útil escalarlos para que tengan todos la misma escala, normalmente entre 0 y 1. Este cambio es fundamental para aquellos algoritmos que ponderan las entradas tales como regresión o las redes neuronales; o aquellos que utilizan medidas de distancia como 'el K-vecino más próximo'

- Estandarizar: es útil cuando los atributos tienen distribución Gausiana pero diferentes medias y desviaciones estándares. Permite centrar todas las variables para que tengan media 0 y desviación 1. Esta transformación es adecuada para aquellos algoritmos que asumen en las variables de entrada una distribución gaussiana y mejoran con el escalado de datos; tales como regresión lineal, regresión logística o análisis del discriminante lineal.

- Normalizar: es un tipo de escalado en el que cada observación (fila) obtiene una longitud de 1 (conocido como unidad de norma o vector de longitud de 1 en el álgebra lineal). Este preprocesado puede ser interesante para conjuntos de datos dispersos (con muchos ceros) y cuyos atributos varían sus escalas. Mejora el rendimiento en los algoritmos que pesan las entradas, tales como redes neuronales o el k-vecino más próximo.

Para evaluar los algoritmos de ML entrenados, se necesita conocer cómo predicen sobre datos que no han sido utilizados en el entrenamiento y de los que conocemos su salida. La evaluación es una estimación de lo bien que el algoritmo responde en la práctica. Para ello tenemos que dividir aleatoriamente los datos en conjunto de entrenamiento (con los que entrenamos) y conjunto de test (con los que evaluamos el rendimiento). Aunque el tamaño de esta división depende del número de datos y especifidades del conjunto original, es común una división del 67 % para entreno y el resto 33 % para test. Como es necesario que los resultados sean reproducibles, sobre todo si queremos comparar distintos algoritmos de ML o distintas configuraciones del mismo algoritmo, en la división aleatoria tenemos que fijar también la semilla con la que se genera la aleatoriedad. Con ello nos aseguramos de que se crean los mismos números pseudoaleatorios y por lo tanto entrenamos los distintos algoritmos con el mismo conjunto de entrenamiento y de test.

Validación cruzada es un método para estimar el rendimiento de algoritmos ML con la menor varianza posible cuando se utiliza un único conjunto de entrenamiento y test. Para ello se divide el conjunto de entrenamiento en k partes (p.e. k=5 o k=10), donde a cada parte se le llama *fold*. El algoritmo se entrena con k-1 *fold* y testea con la parte que no se ha usado en el entrenamiento. Este entrenamiento se repite haciendo que cada una de las partes pase a ser un fold de test cada vez. Por lo que al final se obtendrán k diferentes medidas del rendimiento del algoritmo entrenado; que se puede concluir a partir de la media y la desviación estándar de las k medidas de rendimiento. Esta medida se ajusta mejor a lo esperado cuando se predice con nuevos datos, ya que el algoritmo ha sido entrenado y evaluado múltiples veces sobre diferentes particiones de datos. La elección de k tiene que ser aquella que permita un tamaño razonable de datos para cada conjunto, de manera que cada parte represente las características del sistema y además permita tener suficientes conjuntos como para repetir el proceso de entreno-test un número suficiente de veces. Para conjuntos de datos modestos, de entre 1.000 y 10.000, registros se suelen utilizar los valores de k=3, 5 y 10. Una variante del *k-fold cross-validation* es crear una división aleatoria de los datos para entreno y test, y repetir el proceso de división y evaluación en los algoritmos múltiples veces, siguiendo el *k-fold cross-validation*. Esta modificación mantiene la ventaja de usar un conjunto de entreno y test, y reduce la varianza en la estimación del rendimiento del algoritmo.

En esta cuarta tarea de evaluar los algoritmos ML es importante la elección de la métrica con la que se mide el rendimiento; ya que esta elección nos indica la importancia que le damos a las diferentes características del resultado (Walpole et al., 2011). Y además te indicará qué algoritmo elegir. En problemas de regresión las métricas más utilizadas son:

- Media del error absoluto (MAE – *Mean Absolute Error*) es la suma en valor absoluto de la media entre las predicciones y el valor real. Da una idea de la magnitud del error pero no de su dirección (por encima o por debajo del valor real) Ec. (1)

- Error cuadrático medio (MSE- *Mean Squared Error*) también ofrece una idea de la magnitud del error. Si realizamos la raíz cuadrada de MSE (RMSE - *Root Mean Squared Error*) podemos convertir esta medida a las unidades reales de la variable de salida y nos ofrece una descripción más significativa del resultado Ec. (2)(Hyndman and Koehler, 2006).

- Coeficiente de determinación (R^2), nos da una indicación de la bondad del ajuste de un conjunto de predicciones a los valores reales. Ofrece un valor entre 0 y 1, indicando 0 el peor ajuste y 1 un ajuste perfecto. Por lo tanto, valores de R^2 igual o menores a 0.5 corresponden a malos ajustes Ec. (3). (Vasileva-Stojanovska, 2015)

$$MAE = \frac{\sum_{i=1}^{n} SR_p(i) - SR_o(i)}{n} \tag{1}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(SR_p(i) - SR_o(i))^2} \tag{2}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(SR_o(i) - SR_p(i))^2}{\sum_{i=1}^{n}(SR_o(i) - \frac{1}{n}\sum_{i=1}^{n}(SR_o(i))^2} \tag{3}$$

Donde SR_o y SR_p son los datos reales y las predicciones respectivamente.

3. Algoritmos de aprendizaje automático

Finalmente corresponde descubrir qué algoritmo de ML es el más adecuado al problema a tratar (Pospieszny et al., 2018). Para ello se ensaya con una variedad de métodos que se sospecha que podrían dar buenos resultados. Para tareas de modelado predictivo se suelen estudiar los siguientes:

3.1. Algoritmos lineales de ML

Regresión Lineal: el objetivo de una regresión lineal simple (una variable) es modelar la relación entre una característica (la variable explicativa, x) y el valor continuo de la respuesta del sistema (variable objetivo, y). La ecuación de un modelo lineal se representa como indica la Ec. (4). Donde el objetivo es aprender los pesos (w_0, w_1) que describen tal relación entre variable explicativa y variable objetivo.

$$y = w_0 + w_1 x \tag{4}$$

En base a la ecuación lineal que se define se puede predecir la respuesta ante nuevas variables explicativas que no formen parte del conjunto de entrenamiento. Esta ecuación corresponde con la línea recta que mejor ajuste realice al conjunto de datos de entrenamiento (figura 2).

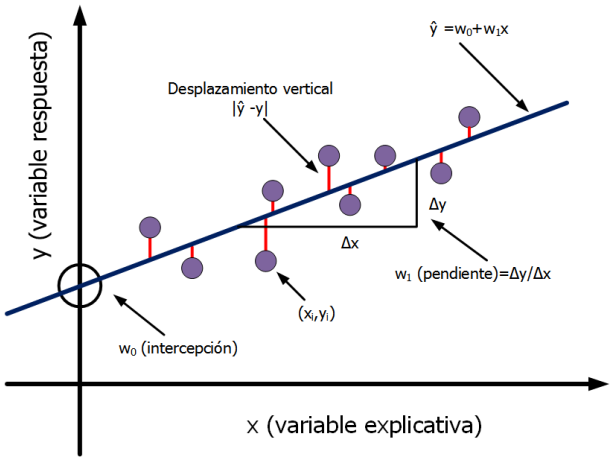

Figura 2: Descripción gráfica de ajuste lineal

Cuando existen múltiples variables explicativas se obtiene la regresión lineal múltiple como generalización de la regresión lineal. En este caso, el ajuste se realiza a un hiperplano en lugar de una línea recta, como muestra la Ec. (5). El objetivo de aprendizaje es determinar los pesos w que mejor ajustan el hiperplano al conjunto de datos de aprendizaje.

$$y = w_0 + w_1 x_1 + ... + w_m x_m = \sum_{i=0}^{m} w_i x_i = w^T x \tag{5}$$

Para determinar cuál es el mejor ajuste se pueden utilizar diferentes técnicas. La más común es el método de mínimos cuadrados (*Ordinary Least Squares* -OLS) donde se estiman los parámetros de la regresión lineal de tal manera que minimice la suma de las distancias verticales (error) al cuadrado de cada dato de entrenamiento. La función de coste a minimizar J es expresada en Ec. (6), siendo m el número de datos de entrenamiento.

$$J(W) = J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} ((w_0 + w_1 x_i) - y_i)^2 \tag{6}$$

Cuando se trabaja con aprendizaje automático es importante encontrar la manera de lograr la correcta generalización del modelo predictivo y de los datos de entrenamiento. En caso de no lograrlo podemos llegar a enfrentarnos a errores graves durante la predicción, conocidos como: sobreajuste (overfitting) o sobregeneralización (underfitting) de los datos de entrenamiento.

Se habla de un error de generalización cuando a pesar de que el modelo sea capaz de responder correctamente al predecir muestras del entrenamiento, falla al procesar nuevas muestras. Esto se da cuando las muestras dadas para el aprendizaje son incompletas o tienen demasiado ruido. En el caso del overfitting, sucede debido al exceso de complejidad en el diseño del ajuste y suele presentarse cuando el número de parámetros del modelo es muy alto, respecto al número de muestras de entrenamiento. Por otra parte el underfitting, se da cuando existe un exceso de generalización del modelo, el cual, prácticamente ignora todas o la mayoría de las muestras de entrenamiento.

Estos errores se pueden prevenir o corregir utilizando la regularización del modelo, que consiste en agregar un término en la función de coste con el fin de favorecer únicamente determinados valores con respecto a otros.

1. Regresión Ridge: también conocida como regulación L2-Normal, consiste en modificar la función de coste J(w), añadiéndole la suma de los coeficientes al cuadrado Ec. (7) . Lo que provoca la sumatoria desde i hasta n de los componentes al cuadrado del modelo, es que al minimizar la función, los valores muy altos queden compensados logrando que todos los componentes del modelo tengan un peso similar. Ec. (7) λ determina el grado de penalización que existe para los valores señalados.

$$J_{L2}(W) = J(W) + \frac{\lambda}{2m} \sum_{i=1}^{n} W_i^2 \tag{7}$$

2. Regresión LASSO (*Least Absolute Shrinkage and Selection Operator*): consiste en modificar la función de coste J(W) añadiéndole la suma de los valores absolutos de los coeficientes. También se la conoce como regulación L1-Normal Ec. (8). En este caso algunos de los coeficientes del modelo predictivo se anularán realizándose una selección de parámetros. Cuanto más grande sea el valor del parámetro de regularización λ, mayor número de parámetros se anularán.

$$J_{L2}(W) = J(W) + \frac{\lambda}{2m} \sum_{i=1}^{n} \|W_i\| \tag{8}$$

3. Regresión ElasticNet donde se aplica un mecanismo de regularización que combina las propiedades de ambas regresiones Ridge y LASSO. Busca minimizar la complejidad del modelo de regresión (magnitud y número de coeficientes de regresión) al penalizar el modelo utilizando tanto el L2-norma (valores de coeficientes de suma al cuadrado) como la L1-norma (suma de valores de coeficientes absolutos).

4. Algoritmos no lineales de *Machine Learning*

1. k-Vecinos más próximos (*K-Nearest Neighbors* - KNN) localiza las k instancias más próximas en el conjunto de entrenamiento para cada nueva instancia, resultando como predicción una media de la salida de las k instancias más cercanas. Comúnmente se utiliza la distancia

de Minkowski que generaliza tanto la distancia Euclidea (usada cuando todas las entradas tienen la misma escala) como la distancia Manhattan (que se usa cuando las escalas de las variables de entrada difieren).

2. Los árboles de decisión son una clase de algoritmo de aprendizaje supervisado que crean un diagrama de flujo que consiste en una secuencia de nodos, donde los valores de una muestra se utilizan para hacer una decisión sobre el siguiente nodo al que ir. El recorrido por el árbol nos dará la predicción del nuevo dato de entrada. En el proceso de entrenamiento se debe hacer crecer el árbol, decidiendo qué características elegir y qué condiciones usar para dividir, junto con saber cuándo parar. Como un árbol generalmente crece arbitrariamente, se tendrá que recortar para que se útil. La técnica más utilizada consiste en hacer crecer el árbol con todas las características y se prueban diferentes puntos de división utilizando una función de costo. Se selecciona la división con el mejor costo. Este algoritmo es de naturaleza recursiva ya que los grupos formados se pueden subdividir utilizando la misma estrategia Esto hace que el nodo raíz sea el mejor predictor.

3. Máquina de soporte vectorial (*Support Vector Machines* - SVM) está sustentado en las teorías estadísticas propuestas por (Vapnik, 1999). Este tipo de modelos pueden ser utilizados en clasificaciones binarias, multi-categorías (Cervantes et al., 2017) o en el caso más general, para aproximar una regresión.Su rendimiento se basa en encontrar los vectores que definan los hiperplanos de separación entre los grupos, con mayor margen. El modelo SVM determinará una superficie para predecir los valores con el mayor margen posible a ambos lados del hiperplano. En el caso en el que haya que recurrir a una superficie no lineal para predecir (La figura 3), el SVM transforma el espacio de atributos en un espacio lineal de mayor dimensionalidad. Para ello se mapea cada punto del conjunto de datos, mediante una transformación no lineal $\Phi(x) : \mathbb{R}^2 \rightarrow F$, en el espacio denominado de características F de dimensión mayor, de forma que cada punto X_n es proyectado en un punto $\Phi(x_n)$. Si la transformación es escogida adecuadamente, el espacio de características F sería un espacio de mayor dimensionalidad, donde la relación entre el conjunto de datos proyectado y los factores subyacentes a la variabilidad del conjunto de datos original sería lineal. Este procedimiento de mapear los datos en un espacio de características mediante una transformación no lineal, se agrupan en lo que se denomina métodos basados en kernel. En (La figura 3) a la izquierda se pueden observar dos conjuntos de datos en $\mathbb{R}^2$. En principio sería muy complicado entrenar un clasificador que pudiera separar ambos conjuntos. Sin embargo, cada punto puede ser mapeado mediante una transformación no lineal $\Phi(x)$ en un espacio de características F de dimensión M. Si la transformación no lineal se escoge adecuadamente, ambos conjuntos de datos serían fácilmente separables utilizando como frontera un hiperplano h en F.

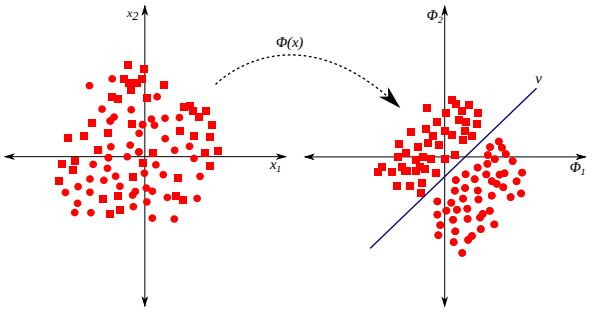

Figura 3: En las SVM dos conjuntos de datos difíciles de clasificar en R2 —izquierda—son proyectado mediante $\Phi(x)$ en el espacio de características —derecha—, donde pueden ser clasificados empleando un hiperplano h como frontera

4. Las Redes Neuronales Artificiales (RNA) son un modelo matemático de una red neuronal biológica formado por la unión de unidades más simples llamadas perceptrón. En las neuronas reales, la dendrita recibe señales eléctricas de los axones de otras neuronas, pero en un perceptrón, estas señales eléctricas se representan como valores numéricos. En las sinapsis entre la dendrita y los axones, las señales eléctricas se modulan en diversas cantidades. Esto también se modela en el perceptrón, multiplicando cada valor de entrada por un valor llamado peso. Una neurona real dispara una señal de salida solo cuando la potencia total de las señales de entrada excede un cierto umbral. Modelamos este fenómeno en un perceptrón calculando la suma ponderada de las entradas para representar la intensidad total de las señales de entrada y luego aplicando una función de activación a la suma para determinar su salida. Al igual que en las redes neuronales biológicas, esta salida se alimenta a otros perceptrones (La figura 4).
   El entrenamiento de un RNA consiste en presentar unos vectores a la entrada e intentar que la salida reproduzca los patrones deseados, mediante la modificación de los pesos según el algoritmo escogido.

5. Caso de Uso: modelado de la predicción de la generación en un campo eólico

En este apartado se realizará un proyecto de aprendizaje automático para la identificación de un campo eólico a partir del histórico de datos, siguiendo las 6 etapas anteriormente descritas. Se utilizará Python como lenguaje de programación, por lo que las tareas se adaptarán para ser implementadas en las librerías de Python existentes (p.e, *pandas* para cargar datos y *scikit-learn* para modelar). En las siguientes subsecciones se describirá cada una de estas tareas y su implementación correspondiente en Python. El block de notas de jupyter con el código del ejemplo del modelo de predicción de generación de campo eólico está disponible en Github https://github.com/rmaguilarc/Modelos_Predictivos.

5.1. *Definición del problema*

En esta tarea se debe determinar todo lo que se necesita para comenzar a trabajar en el problema. Esto incluye la definición

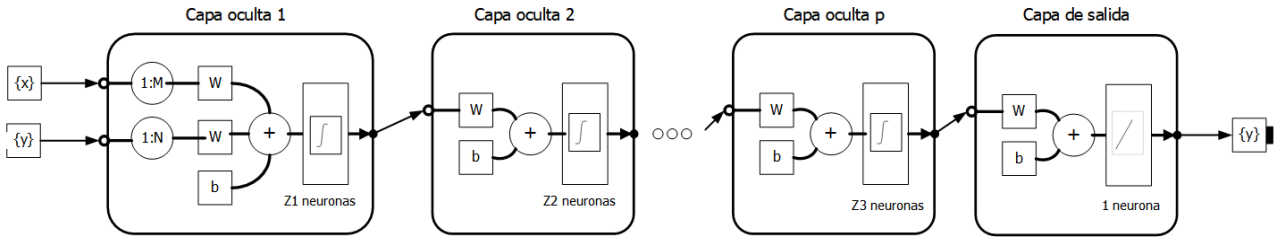

Figura 4: Esquema de una red neuronal

del mismo, cargar los módulos de Python a utilizar y obtener el conjunto de datos con el que se va a trabajar. Se debe realizar en este paso cualquier configuración global que se vaya a necesitar. Asimismo, si el conjunto de datos del que se dispone es demasiado grande se tiene que hacer una muestra más reducida. Idealmente, el conjunto de datos debe ser lo suficientemente pequeño como para construir un modelo o crear una visualización en un minuto (preferible 30 segundos). Una vez que se determinan los modelos que mejor funcionan en pasos posteriores, se puede escalar el problema al conjunto total de datos.

El problema en estudio es diseñar un modelo que nos proporcione una buena aproximación de la generación de un parque eólico. Para conocer la producción de energía eólica es necesario contar con un sistema de predicción preciso, capaz de vincular los cambios diarios asociados a las variaciones en la velocidad y dirección del viento del lugar en el que está instalado el parque eólico. Pero, ¿cuál es el mejor método para predecir la generación de un parque eólico? Para responder a esta pregunta vamos a implementar una metodología basada en ML que a partir de los datos históricos que tenemos para un parque concreto, obtengamos el modelo que mejor predicción hace de la generación producida por dicho parque.

Para ello trabajaremos con datos de uno de los parques eólicos del Instituto Tecnológico y de Energías Renovables de Canarias (ITER). El ITER cuenta con tres parques eólicos entre los que se encuentra el parque MADE, denominado así por el tipo de aerogeneradores que lo componen. Son ocho aerogeneradores tipo MADE AE-46, que proporcionan una potencia nominal al parque de 4,8 MW. Con los datos extraídos de este parque se ha construido una base de datos que contiene medidas, reales y previstas, tanto de las condiciones meteorológicas y variables temporales, como de la generación del parque. El conjunto de datos abarca un periodo comprendido entre el 01 de enero de 2014 y el 31 de marzo de 2016, para un total de dos años y tres meses.

Cada registro de la base de datos nos da para una hora concreta de una fecha concreta las características atmosféricas y la generación producida en esa hora. Los atributos se definen como se indica a continuación: Velocidad del viento (m/seg), Dirección del viento (grados), Humedad relativa (tanto por uno), Temperatura de la superficie (^{o}C), Año, Estación (verano o invierno), Mes, Día, Hora, Generación (kWh) (figura 6).

En esta primera tarea conocemos el problema y cargamos en el entorno de implementación las librerías de Python necesarias y el fichero con los datos. Detalles del código en Apéndice A.1, Apéndice A.2.

```
# shape
print('shape')
print(dataset.shape)
```

```
shape
(13087, 10)
```

```
# types
print('types')
print(dataset.dtypes)
```

```
types
Velocidad_viento    float64
Direccion_viento      int64
Humedad_Relativa    float64
Temperatura         float64
Año                   int64
Estacion              int64
Mes                   int64
Dia                   int64
Hora                  int64
Generacion_real     float64
dtype: object
```

Figura 5: Características del conjunto de datos

5.2. *Analizar los datos*

En esta tarea se trata de comprender mejor los datos de los que se dispone. Esto incluye tanto estadísticas descriptivas como visualizaciones de los datos.

Comenzamos por confirmar las dimensiones del conjunto de datos, el número de filas y columnas. A continuación conozcamos los tipos de datos de cada atributo. Se observa que se dispone de 13087 instancias para trabajar y podemos confirmar que los datos tienen 10 atributos que incluyen el atributo de salida Generación. Asimismo vemos que todos los atributos son valores numéricos, algunos reales (*float*) y otros han sido interpretados como enteros (*int*)(figura 5).

Al observar las primeras 10 filas de los datos, se confirma que las escalas de los atributos son muy diferentes debido a las distintas unidades que se utilizan. Para mejorar el ajuste se deben realizar algunas transformaciones (figura 6). El código en Python de estas transformaciones en: Apéndice A.3.

Las correlaciones entre los atributos se muestran en la figura 7. El color claro muestra una correlación positiva, mientras que el color oscuro indica correlación negativa. Al estar estas variables correlacionadas no aportan información adicional y sí que complican el modelo, ya que tendría mayor peso debido a estas

relaciones. Por lo tanto estas variables podrían ser candidatas a eliminar para mejorar la precisión de los modelos.

5.3. *Preparar Datos*

Como se ha indicado anteriormente, la prueba de bondad del modelo predictivo diseñado nos la dan los valores de ajuste que proporcione el modelo sobre datos que no han sido utilizados en el entrenamiento. Para ello debemos tendremos que dividir el conjunto de datos en dos partes: el conjunto de entrenamiento y el conjunto de validación. Éste último conjunto se usará al final del proyecto para confirmar la precisión del modelo propuesto. Este último ejercicio nos indica cuánto nos equivocamos y ofrece la confianza en la precisión de las estimaciones en datos no vistos. En este caso, como no se dispone de una cantidad grande de datos, no se podrá utilizar la recomendación de 1/3 de datos para validación. Por lo que se utilizará el 80 % del conjunto de datos para entrenar y se guardará el 20 % para la validación. El código lo podemos encontrar en: Apéndice A.4

```
: # head
  print('head')
  print(dataset.head(20))
```

```
head
    Vel_V  Dir_V   Hum     T   Año  Est  Mes  Dia  Hora  Gen_real
0    4.75     22  59.0  14.5  2014    1    1    1     0     123.0
1    4.27     23  61.0  13.9  2014    1    1    1     1     161.0
2    3.82     20  57.0  13.1  2014    1    1    1     2     251.0
3    3.69     11  52.0  13.1  2014    1    1    1     3     179.0
4    3.90     11  54.0  12.9  2014    1    1    1     4     109.0
5    4.29     13  64.0  13.0  2014    1    1    1     5      90.0
6    4.24     14  70.0  12.8  2014    1    1    1     6     144.0
7    4.31     13  75.0  12.8  2014    1    1    1     7     187.0
8    4.04      9  75.0  12.6  2014    1    1    1     8     195.0
9    3.48     90  64.0  14.3  2014    1    1    1     9      47.0
10   4.67     30  74.0  16.8  2014    1    1    1    10      50.0
11   4.04     47  69.0  17.6  2014    1    1    1    11     204.0
12   4.98     66  70.0  17.9  2014    1    1    1    12     436.0
13   5.59     70  71.0  17.9  2014    1    1    1    13     471.0
14   5.16     73  72.0  17.8  2014    1    1    1    14     395.0
15   5.82     63  70.0  18.1  2014    1    1    1    15     319.0
16   5.14     63  69.0  17.9  2014    1    1    1    16     323.0
17   4.82     63  72.0  17.5  2014    1    1    1    17     327.0
18   3.96     48  74.0  16.9  2014    1    1    1    18     232.0
19   3.05     33  75.0  16.0  2014    1    1    1    19      68.0
```

Figura 6: Impresión de las primeras filas del conjunto de datos

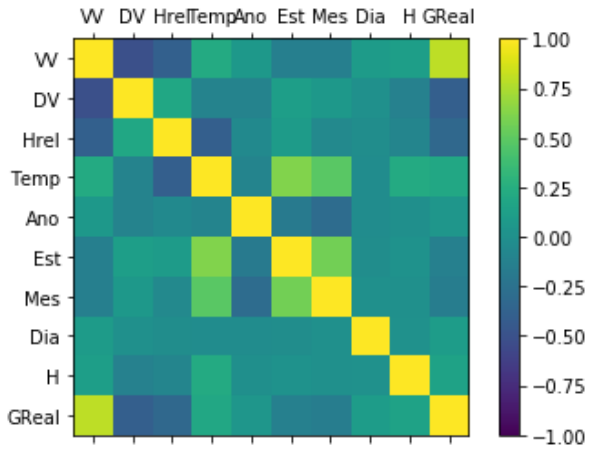

Figura 7: Gráfico de correlación de la variables del problema

5.4. *Evaluar Algoritmos*

A priori no conocemos qué algoritmos aprenderán bien en este problema. Podríamos intentar cualquiera de los anteriormente descritos. Sin embargo resulta más eficiente realizar validación cruzada (Stone, 1974).

La validación cruzada se inicia mediante el fraccionamiento de un conjunto de datos en un número k de particiones (generalmente entre 5 y 10). La validación cruzada luego itera entre los datos de evaluación y entrenamiento k veces, de un modo particular. En cada iteración de la validación cruzada, se elige una partición diferente como datos de evaluación. En esta iteración, las otras *k-1* particiones se combinan para formar los datos de entrenamiento. Por lo tanto, en cada iteración tenemos *(k-1)/k* de los datos utilizados para el entrenamiento y *1/k* utilizado para la evaluación. Cada iteración produce un modelo, y por lo tanto una estimación del rendimiento de la generalización de dicho modelo. Una vez finalizada la validación cruzada, todos los datos se han utilizado sólo una vez para evaluar pero *k-1* veces para entrenar. En este punto tenemos estimaciones de rendimiento de todas las particiones y podemos calcular la media y la desviación estándar de la precisión del modelo (Kokkinos and Margaritis, 2018). Como el conjunto de datos no es muy pequeño podemos utilizar k=10 (10 veces validación cruzada) donde incluiremos todos los algoritmos mencionados.

Evaluaremos los algoritmos usando la métrica del Coeficiente de determinación (R^2), que indicará la bondad de todas las predicciones. Mediante este coeficiente es posible seleccionar el mejor modelo de entre varios que tengan el mismo número de variables exógenas, ya que la capacidad explicativa de un modelo es mayor cuanto más elevado sea el valor que tome este coeficiente. Sin embargo, hay que tener cierto cuidado a la hora de trabajar con modelos que presenten un (R^2) muy cercano a 1 pues, aunque podría parecer que estamos ante el modelo "perfecto", en realidad podría encubrir ciertos problemas de índole estadística como la multicolinealidad (Walpole et al., 2011). Por otra parte, el valor del coeficiente de determinación aumenta con el número de variables exógenas del modelo por lo que, si los modelos que se comparan tienen distinto número de variables exógenas, no puede establecerse comparación entre sus (R^2). La programación está en: Apéndice A.5

Creemos un marco para evaluar el rendimiento de los 6 algoritmos seleccionados capaces de trabajar con este modelo predictivo: Regresión lineal (LR) con regularización Lasso y ElasticNet (EN); KNN, árbol de decisiones (CART), SVM y MLP (perceptrón multicapa). Todos los algoritmos utilizarán los parámetros de ajuste predeterminados. Para compararlos se muestra la media y la desviación estándar de (R^2) para cada algoritmo (Tabla 1), donde parece que el que mejor resultado ofrece es el de los k-vecinos más próximos. En la figura 8 se puede observar la distribución de puntuaciones en todos los k conjuntos de la validación cruzada por algoritmo.

Tabla 1: Resultados de la evaluación de los algoritmos

Algoritmo	Media R^2	Des.Estándar R^2
Regresión Lineal (LR)	0.647	0.022
Regresión LASSO	0.647	0.022
Regresión ElasticNet (EN)	0.0647	0.022
KNN	0.667	0.031
Árbol de decisión (CART)	0.536	0.032
SVM	-0.061	0.020
MLP	0.658	0.027

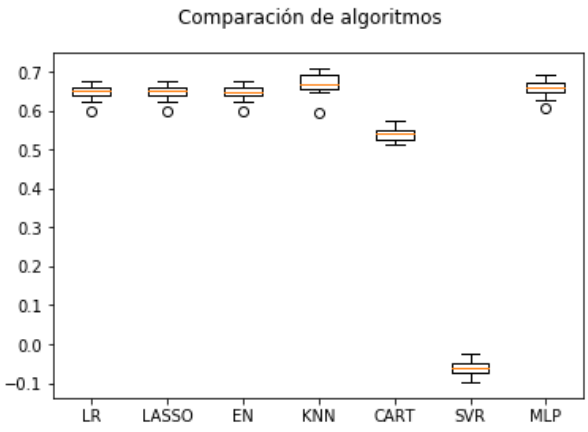

Figura 8: Comparación del rendimiento de los algoritmos

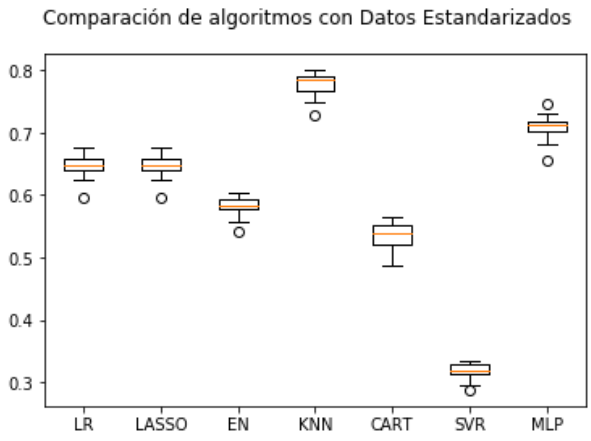

Figura 9: Comparación de los algoritmos datos estandarizados

Tabla 2: Evaluación de los algoritmos con datos estandarizados

Algoritmo	Media R^2	Des.Estándar R^2
Regresión Lineal (LR)	0.647	0.022
Regresión LASSO	0.647	0.022
Regresión ElasticNet (EN)	0.580	0.018
KNN	0.776	0.022
Árbol de decisión (CART)	0.531	0.022
SVM	0.317	0.015
MLP	0.709	0.024

El rendimiento de los algoritmos pueden verse disminuido por las diferentes escalas de los datos, por lo que ejecutaremos el mismo test pero utilizando una copia de los datos estandarizados. Para ello transformamos los datos de modo que cada atributo tiene un valor medio de cero y una desviación estándar de 1. Para evitar perder datos en esta transformación realizaremos la estandarización y el modelo del algoritmo a la vez para cada una de los pasos de la validación cruzada, Apéndice A.6. De esa forma podemos obtener una buena estimación de cómo funciona cada modelo con datos estandarizados (figura 9). Los resultados (Tabla 2) indican que con los datos estandarizados el algortimo KNN obtiene mejores resultados, es decir, (R^2) más próxima a 1.

5.5. *Mejorar los Resultados*

Los resultados de la sección anterior demuestran que el algoritmo KNN logra buenos resultados en un conjunto de datos escalados. Podríamos intentar mejorar este ajuste variando los parámetros de dicho algoritmo, ya que el experimento se realizó con los valores por defecto que es un k=7. En este paso probamos con un conjunto de diferentes números de vecinos con el objetivo de mejorar el rendimiento. En el ejemplo del Apéndice A.7 se prueba con valores impares k, del 1 al 21, en un rango arbitrario que cubre el mejor valor conocido de 7. Cada valor k (vecinos) se evalúa utilizando 10 veces en la validación cruzada con una copia estandarizada del conjunto de datos de entrenamiento. Obteniéndose los resultados que se muestran en la tabla 3, que indican que el parámetro a elegir es k=5.

5.6. *Presentar Resultados*

Finalmente, configuramos el modelo predictivo resultante (el código en Apéndice A.8), que es un KNN con k=5; y lo entrenamos con todo el conjunto de datos una vez estandarizado. Tras simular podemos obtener el rendimiento de este modelo para la predicción de generación del campo eólico. Cuyos resultados en cuanto a rendimiento son los mostrados en (Tabla 3).

Tabla 3: Ajuste de KNN

k_vecino	Media R^2	Des.Estándar R^2
1	0.708767	0.032005
3	0.774618	0.024086
5	0.776110	0.021747
7	0.768782	0.021747
9	0.762691	0.020683
11	0.757427	0.020880
13	0.753129	0.021292
15	0.750113	0.021786
17	0.746876	0.020717
19	0.743944	0.022265
21	0.741102	0.023158

6. Conclusiones

En este trabajo se proponen algoritmos de aprendizaje automático supervisado como mecanismos de identificación de sistemas complejos, cuando se dispone de una gran cantidad de datos del comportamiento del proceso. Para ello se describen los pasos a dar en cualquier proyecto de este tipo y se estudian distintos algoritmos. Su aplicación se realiza al modelado de la predicción de la generación de un campo eólico.

Se ha puesto especial énfasis en utilizar como lenguaje de programación Python, ya que dispone de librerías para la implementación de proyectos de ML que son muy accesibles, tanto por su curva de aprendizaje como en rendimiento computacional. Asimismo, como entorno de trabajo se propone Jupyter Notebook, por su facilidad para experimentar, evaluar los resultados y compartir los resultados de la investigación.

En el caso de uso propuesto se estudiaron los siguientes algoritmos: regresión lineal, regresión LASSO, regresión ElasticNet, árbol de decisión, máquina de soporte de vectores, k-vecino más cercano y redes neuronales artificiales. Se obtuvo que el mejor predictor es el algoritmo k-vecino más cercando, con un rendimiento de predicción de coeficiente de determinación de 0,77 (valor de 1 indica el mejor predictor). En el proceso de entrenamiento de los distintos algoritmos se utilizó la técnica de la validación cruzada, con el objeto de reducir la varianza.

Los algoritmos de ML funcionan como una caja negra, donde se construye un modelo en función de los datos de entrenamiento. La generalización de los resultados sólo es posible en problemas similares con conjunto de datos similares. Y las condiciones donde los algoritmos ofrezcan los mejores resultados serán aquellas en las que se disponga de un conjunto de datos lo suficientemente representativo de la dinámica que se pretende predecir. Además, de utilizar técnicas de reducción de la varianza que eviten el sobreajuste.

Como trabajo futuro se planea utilizar métodos de ensamblado (ensemble) para componer el modelo predictivo como combinación de diferentes algoritmos de ML (como los descritos) con el objeto de incrementar la precisión en la predicción.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Ciencia, Innovación y Universidades a través del proyecto titulado "Gestión de Redes Eléctricas Inteligentes con estimación De producción de Energías Renovables basado en modelos mesoescalares de alta resolución - GRIDER", RTC-2017-6409-3 .

Referencias

Bermudez-Edo, M., Barnaghi, P., Moessner, K., 2018. Analysing real world data streams with spatio-temporal correlations: Entropy vs. pearson correlation. Automation in Construction 88, 87 – 100.
URL: http://www.sciencedirect.com/science/article/pii/S0926580517303874
DOI: https://doi.org/10.1016/j.autcon.2017.12.036

Bibault, J.-E., Giraud, P., Burgun, A., 2016. Big data and machine learning in radiation oncology: State of the art and future prospects. Cancer Letters 382 (1), 110 – 117.
URL: http://www.sciencedirect.com/science/article/pii/S0304383516303469
DOI: https://doi.org/10.1016/j.canlet.2016.05.033

Brunner, R. J., Kim, E. J., 2016. Teaching data science. Procedia Computer Science 80, 1947 – 1956, international Conference on Computational Science 2016, ICCS 2016, 6-8 June 2016, San Diego, California, USA.
URL: http://www.sciencedirect.com/science/article/pii/S1877050916310006
DOI: https://doi.org/10.1016/j.procs.2016.05.513

Cervantes, J., Taltempa, J., García, F., Ruiz, J., Yee, A., Jalili, L., 2017. Análisis comparativo de las técnicas utilizadas en un sistema de reconocimiento de hojas de planta. Revista Iberoamericana de Automática e Informática Industrial RIAI 14 (1), 104–114.
URL: https://polipapers.upv.es/index.php/RIAI/article/view/9244
DOI: 10.1016/j.riai.2016.09.005

Domínguez-Tejo, E., Metternicht, G., Johnston, E. L., Hedge, L., 2018. Exploring the social dimension of sandy beaches through predictive modelling. Journal of Environmental Management 214, 379 – 407.
URL: http://www.sciencedirect.com/science/article/pii/S0301479718302238
DOI: https://doi.org/10.1016/j.jenvman.2018.03.006

Fichman, R., Dos Santos, B., Zheng, Z., 2014. Digital innovation as a fundamental and powerful concept in the information systems curriculum. MIS Quarterly: Management Information Systems 38 (2), 329–353, cited By 76.
URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84923459614\&partnerID=40\&md5=3e9979842834a08721716f33f1ae5ced

Forbes, M. G., Patwardhan, R. S., Hamadah, H., Gopaluni, R. B., 2015. Model predictive control in industry: Challenges and opportunities. IFAC-PapersOnLine 48 (8), 531 – 538, 9th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2015.
URL: http://www.sciencedirect.com/science/article/pii/S2405896315011039
DOI: https://doi.org/10.1016/j.ifacol.2015.09.022

Hyndman, R. J., Koehler, A. B., 2006. Another look at measures of forecast accuracy. International Journal of Forecasting 22 (4), 679–688.
URL: http://www.sciencedirect.com/science/article/pii/S0169207006000239
DOI: 10.1016/j.ijforecast.2006.03.001

Kokkinos, Y., Margaritis, K. G., 2018. Managing the computational cost of model selection and cross-validation in extreme learning machines via cholesky, svd, qr and eigen decompositions. Neurocomputing 295, 29 – 45.
URL: http://www.sciencedirect.com/science/article/pii/S0925231218300195
DOI: https://doi.org/10.1016/j.neucom.2018.01.005

Ljung, L., 1998. System Identification: Theory for the User. Pearson Education.
URL: https://books.google.es/books?id=fYSrk4wDKPsC

Ljung, L., 2008. Perspectives on system identification. IFAC Proceedings Volumes 41 (2), 7172 – 7184, 17th IFAC World Congress.
URL: http://www.sciencedirect.com/science/article/pii/S1474667016400984
DOI: https://doi.org/10.3182/20080706-5-KR-1001.01215

Ljung, L., Hjalmarsson, H., Ohlsson, H., 2011. Four encounters with system identification. European Journal of Control 17 (5), 449 – 471.
URL: http://www.sciencedirect.com/science/article/pii/S0947358011709712
DOI: https://doi.org/10.3166/ejc.17.449-471

Morris, D. H., Gostic, K. M., Pompei, S., Bedford, T., Łuksza, M., Neher, R. A., Grenfell, B. T., Lässig, M., McCauley, J. W., 2018. Predictive modeling of influenza shows the promise of applied evolutionary biology. Trends in Microbiology 26 (2), 102 – 118.
URL: http://www.sciencedirect.com/science/article/pii/S0966842X17302093
DOI: https://doi.org/10.1016/j.tim.2017.09.004

Peña, D., 2014. Big data and statistics: Trend or change? Boletín de Estadística e Investigación Operativa 30-3, 313 – 324.
URL: http://www.seio.es/BEIO/files/BEIOVol30Num3_opinion1.pdf

Pillonetto, G., Chen, T., Chiuso, A., Nicolao, G. D., Ljung, L., 2016. Regularized linear system identification using atomic, nuclear and kernel-based norms: The role of the stability constraint. Automatica 69, 137 – 149.
URL: http://www.sciencedirect.com/science/article/pii/S0005109816300449
DOI: https://doi.org/10.1016/j.automatica.2016.02.012

Piñuela-Martín, E., del Ama, A. J., Fraile-Marinero, J. C., Ángel Gil-Agudo, 2016. Modelización de la estimulación eléctrica neuromuscular mediante un enfoque fisiológico y de caja negra. Revista Iberoamericana de Automática e Informática Industrial RIAI 13 (3), 330 – 337.
URL: http://www.sciencedirect.com/science/article/pii/S1697791216300073
DOI: https://doi.org/10.1016/j.riai.2015.09.012

Pospieszny, P., Czarnacka-Chrobot, B., Kobylinski, A., 2018. An effective approach for software project effort and duration estimation with machine learning algorithms. Journal of Systems and Software 137, 184 – 196.
URL: http://www.sciencedirect.com/science/article/pii/S0164121217302947
DOI: https://doi.org/10.1016/j.jss.2017.11.066

Stone, M., 1974. Cross-validatory choice and assessment of statistical predictions. Journal of the Royal Statistical Society 36, 111–147.

Torres, J. M., Aguilar, R. M., Zuñiga-Meneses, K. V., 2018. Deep learning to predict the generation of a wind farm. Journal of Renewable and Sustainable Energy 10 (1), 013305.
URL: https://doi.org/10.1063/1.4995334
DOI: 10.1063/1.4995334

Vapnik, V. N., Sep 1999. An overview of statistical learning theory. IEEE

Transactions on Neural Networks 10 (5), 988–999.
DOI: 10.1109/72.788640
Walpole, R., Raymond, S. L. M., Myers, H., 2011. Probability & Statistics for Engineers & Scientists. Pearson, 9 edition.
URL: https://www.amazon.ca/Probability-Statistics-Engineers-Scientists-9th/dp/0321629116

Apéndice A. Código en Python para la Identificación de un parque eólico

Apéndice A.1. Cargar librerías para ML

```
# Load libraries
import numpy
import pandas as pd

from matplotlib import pyplot

from pandas import set_option
from pandas.plotting import scatter_matrix

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import KFold, \
    train_test_split, cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.neural_network import MLPRegressor
from sklearn.pipeline import Pipeline
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import median_absolute_error
```

Apéndice A.2. Cargar datos del parque eólico

```
# Load dataset
filename = 'Datosgeneracion.xlsx'
dataset = pd.read_excel(filename)
```

Apéndice A.3. Características del conjunto de datos

```
# shape
print('shape')
print(dataset.shape)

# types
print('types')
print(dataset.dtypes)

# head
print('head')
print(dataset.head(20))

# descriptions
print('descriptions')
set_option('precision', 1)
print(dataset.describe())

# correlation
print('correlation')
set_option('precision', 2)
print(dataset.corr(method='pearson'))

# Visualization data

# histograms
print('histograms')
dataset.hist(
    sharex=False,
    sharey=False,
    xlabelsize=1,
    ylabelsize=1
)
pyplot.show()

# density
print('density')
dataset.plot(
    kind='density',
    subplots=True,
    layout=(4, 4),
    sharex=False,
    legend=False,
    fontsize=1
)
pyplot.show()

# box and whisker plots
dataset.plot(
    kind='box',
    subplots=True,
    ayout=(4, 4),
    sharex=False,
    sharey=False,
    fontsize=8
)
pyplot.show()

# scatter plot matrix
scatter_matrix(dataset)
pyplot.show()

# correlation matrix
names = [
    'VV', 'DV', 'Hrel', 'Temp', 'Ano', 'Est',
    'Mes', 'Dia', 'H', 'GReal'
]
fig = pyplot.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(
    dataset.corr(),
    vmin=-1,
    vmax=1,
    interpolation='none'
)
fig.colorbar(cax)
ticks = numpy.arange(0, 10, 1)
```

```
ax.set_xticks(ticks)
ax.set_yticks(ticks)
ax.set_xticklabels(names)
ax.set_yticklabels(names)
pyplot.show()
```

Apéndice A.4. Separación de datos

```
# Split-out validation dataset
array = dataset.values
X = array[:, 0:8]
Y = array[:, 9]
validation_size = 0.20
seed = 7
X_train, X_val, Y_train, Y_val =
    train_test_split(X,
                     Y,
                     test_size=validation_size,
                     random_state=seed)
```

Apéndice A.5. Test para evaluación de los algoritmos

```
# Test options and evaluation metric
num_folds = 10
seed = 7

#scoring = 'neg_mean_squared_error'
scoring = 'r2'

# Spot-Check Algorithms
models = []
models.append(('LR', LinearRegression()))
models.append(('LASSO', Lasso()))
models.append(('EN', ElasticNet()))
models.append(('KNN', KNeighborsRegressor()))
models.append(('CART', DecisionTreeRegressor()))
models.append(('SVR', SVR()))
models.append(('MLP', MLPRegressor()))

# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = KFold(
        n_splits=num_folds,
        random_state=seed
    )
    cv_results = cross_val_score(
        model,
        X_train,
        Y_train,
        cv=kfold,
        scoring=scoring
    )

    results.append(cv_results)
    names.append(name)
    msg = "%s R^2: %.3f (%.3f)" % (
        name,
        cv_results.mean(),
        cv_results.std()
    )
    print(msg)

# Compare Algorithms
fig = pyplot.figure()
fig.suptitle('Comparación de algoritmos')
ax = fig.add_subplot(111)
pyplot.boxplot(results)
ax.set_xticklabels(names)
pyplot.show()
```

Apéndice A.6. Test algoritmos con datos estandarizados

```
pipelines = []

pipelines.append((
    'ScaledLR',
    Pipeline([
        ('Scaler', StandardScaler()),
        ('LR', LinearRegression())
    ])
))

pipelines.append((
    'ScaledLASSO',
    Pipeline([
        ('Scaler', StandardScaler()),
        ('LASSO', Lasso())
    ])
))

pipelines.append((
    'ScaledEN',
    Pipeline([
     ('Scaler', StandardScaler()),
        ('EN', ElasticNet())
    ])
))

pipelines.append((
    'ScaledKNN',
    Pipeline([
        ('Scaler', StandardScaler()),
        ('KNN', KNeighborsRegressor())
    ])
))

pipelines.append((
    'ScaledCART',
    Pipeline([
        ('Scaler', StandardScaler()),
        ('CART', DecisionTreeRegressor())
    ])
))
pipelines.append((
    'ScaledSVR',
    Pipeline([
        ('Scaler', StandardScaler()),
        ('SVR', SVR())
    ])
```

```
))

pipelines.append((
    'ScaledMLP',
    Pipeline([
        ('Scaler', StandardScaler()),
        ('MLP', MLPRegressor(max_iter=2000))
   ])
))

results = []
names = [
    'LR', 'LASSO', 'EN', 'KNN', 'CART',
    'SVR', 'MLP'
]
for name, model in pipelines:
    kfold = KFold(
        n_splits=num_folds,
        random_state=seed
    )
    cv_results = cross_val_score(
        model,
        X_train,
        Y_train,
        cv=kfold,
        scoring=scoring
    )

    results.append(cv_results)
    names.append(name)
    msg = "%s R^2: %.3f (%.3f)" % (
        name, cv_results.mean(), cv_results.std()
    )
    print(msg)

    # Compare Algorithms
    fig = pyplot.figure()
    fig.suptitle(
        'Comparación Datos Estandarizados')
    ax = fig.add_subplot(111)
    pyplot.boxplot(results)
    ax.set_xticklabels(names)
    pyplot.show()
```

Apéndice A.7. Ajuste de los parámetros del algoritmo KNN

```
# KNN Algorithm tuning
scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
k_values = numpy.array([
    1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21
])
param_grid = dict(n_neighbors=k_values)
model = KNeighborsRegressor()
kfold = KFold(
    n_splits=num_folds,
    random_state=seed
)
grid = GridSearchCV(
    estimator=model,
    param_grid=param_grid,
    scoring=scoring, cv=kfold
)
grid_result = grid.fit(rescaledX, Y_train)

print("Best: %f using %s" % (
    grid_result.best_score_,
    grid_result.best_params_)
)
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(
        means, stds, params):
    print("%f (%f) with: %r" % (
        mean, stdev, param))
```

Apéndice A.8. Construcción y evaluación del modelo final

```
# prepare the model
scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = KNeighborsRegressor(n_neighbors=5)
model.fit(rescaledX, Y_train)

rescaledValidationX = scaler.transform(X_val)
predictions = model.predict(rescaledValidationX)
print(r2_score(Y_val, predictions))
print(mean_squared_error(Y_val, predictions))
print(median_absolute_error(Y_val, predictions))
```