Universidad de Cantabria

Facultad
de
Ciencias

DESARROLLO DE MODELO DE OPTIMIZACIÓN DEL RATIO PRODUCTIVO Y ENERGÉTICO EN PLANTAS DESALADORAS

Development of an optimization model of the production and energy ratio in desalination plants

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Samuel Carrera Garmendia

Director: Alfonso de la Vega Ruiz

Codirector: Miguel Sierra Sánchez

Junio – 2025

RESUMEN

En lo que se refiere al campo de la inteligencia artificial y el machine learning, hemos tenido grandes avances que permiten generar herramientas de ayuda y predicción. Este tipo de tecnología facilita desde recomendaciones de series personalizadas para el usuario o ámbitos más serios como llegar a predecir una enfermedad.

Es por esto por lo que se propuso el desarrollo de un modelo de predicción que mejore el ratio de producción de una bomba de agua, en función de los datos que se almacenan y con los que posteriormente se predecirá el ratio que va a tomar la bomba en el futuro y que podemos hacer para que no baje.

Cabe destacar que este algoritmo ha sido desarrollado durante unas prácticas realizadas en la empresa llamada “Soincon”, haciendo uso de unos de sus módulos llamado “Fast Monitoring” que se encarga de recoger todos los datos acerca de la bomba.

Los datos son capturados de forma automática, con el objetivo de permitir su estudio como posibles predictores del comportamiento de los procesos en el futuro.

El modelo ha sido implementado en Python, donde se han entrenado diversos modelos tales como XgBoost, Catboost y redes neuronales, a partir de datos recogidos en la bomba de agua, para que después estos realicen predicciones en función de lo aprendido. A su vez serán analizados los resultaos que nos den cada uno de los modelos mediante RMSE, MAE y diversas métricas que nos permitirán estimar con que exactitud trabaja el modelo.

Palabras clave: Datos industriales, Machine Learning, Modelos de Predicción, Regresión, Redes neuronales, xgboost, catboost.

ABSTRACT

In the field of artificial intelligence and machine learning, we have made great advances that allow for the development of support and prediction tools. This type of technology facilitates everything from personalized series recommendations to more serious areas such as predicting an illness.

This is why the development of a prediction model was proposed to improve the production rate of a water pump, based on the stored data. This model will then be used to predict the pump's future production rate and what we can do to prevent it from falling.

It should be noted that this algorithm was developed during an internship at a company called Soincon, using one of its modules called "Fast Monitoring," which is responsible for collecting all the data about the pump.

The data is captured automatically, allowing its study as potential predictors of future process behavior.

The model has been implemented in Python, where various models such as XgBoost, Catboost, and neural networks have been trained using data collected from the water pump and then made predictions based on what they learned. The results of each model will be analyzed using RMSE, MAE, and various metrics that will allow us to estimate the accuracy of the model.

Keywords: Industrial data, Machine Learning, Prediction Models, Regression, Neural networks, xgboost, catboost.

Contenido

Índice de Figuras

Índice de Tablas

Índice de Siglas

ERI: Energy Recovery Inc.

RO: Reverse Osmosis.

1. INTRODUCCIÓN Y OBJETIVOS

1.1. Contexto

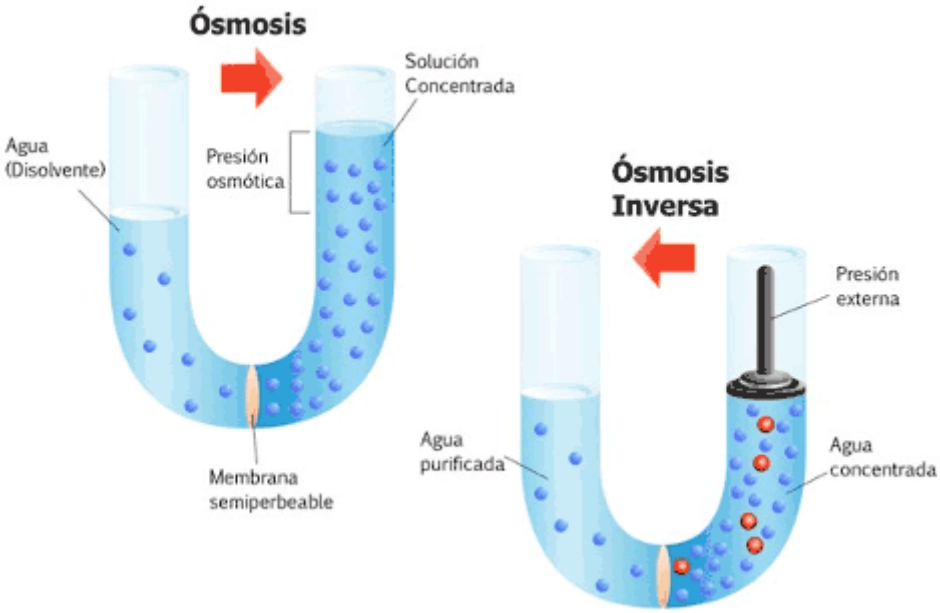

Figura 1: Explicación ósmosis inversa de https://www.carbotecnia.info

Actualmente, todo proceso industrial contempla dentro de sus objetivos económicos la reducción de los costos energéticos. Por ejemplo, la ósmosis inversa que podemos ver en la *Figura 1*, (también conocido como Reverse Osmosis, o RO, por sus siglas en inglés) que se encarga de que un agua con impurezas, a partir de aplicarle presión y hacer que pase por una membrana quede limpia. Al ser un proceso en la que la parte más importante de los costos de funcionamiento corresponden al bombeo de alta presión, a lo largo de los años se ha avanzado en la investigación de los procesos de recuperación de energía a través de aprovechar el agua de rechazo de alta presión, recuperando su energía.

Los sistemas de recuperación de energía aprovechan la gran presión del agua de rechazo generada en la ósmosis para devolverla, en gran parte, al agua de entrada a las membranas y así disminuir la cantidad de energía a suministrar por las bombas para alcanzar las grandes presiones de entrada a la ósmosis.

Para producir la ósmosis inversa, es preciso llevar el agua de mar a una presión de aproximadamente 60-70bar (es decir, 60-70 veces la presión atmosférica normal) en la entrada de las membranas. Esta presión no se pierde en el interior de las membranas, sino que la salmuera a la salida tiene esa misma presión menos las pérdidas de carga al pasar por las membranas. Como esta salmuera debe ser devuelta al mar, hay que quitarle previamente esa presión.

Las primeras plantas de ósmosis inversa solían tener una válvula reductora de presión para romper carga de la salmuera antes de su envío al mar. Esta situación duró muy poco, pues pronto se vio la mejora que suponía recuperar la energía de la salmuera en vez de tirarla.

La planta cuenta con sistemas de recuperación de energía de tipo ERI, denominación derivada de la empresa americana que los fabrica (Energy-Recovery Inc.). Este equipo recicla la energía que de lo contrario se perdería en el proceso de desalinización. Específicamente, captura la energía hidráulica de la corriente de agua de mar rechazada a alta presión y luego transfiere esta energía al agua de alimentación a baja presión con una eficiencia de hasta el 98 por ciento y sin energía eléctrica, además pueden reducir los costos de energía en un sistema RO (también conocido como Reverse Osmosis, o RO, por sus siglas en inglés) hasta un 60%.

Este proyecto tiene como objetivo minimizar el consumo específico de energía de cada *rack* de ósmosis inversa, entendido como una unidad autónoma dentro de la planta, compuesta por diversos componentes como bombas de alta presión, membranas y válvulas de control, que en conjunto permiten la operación independiente del proceso de desalinización.

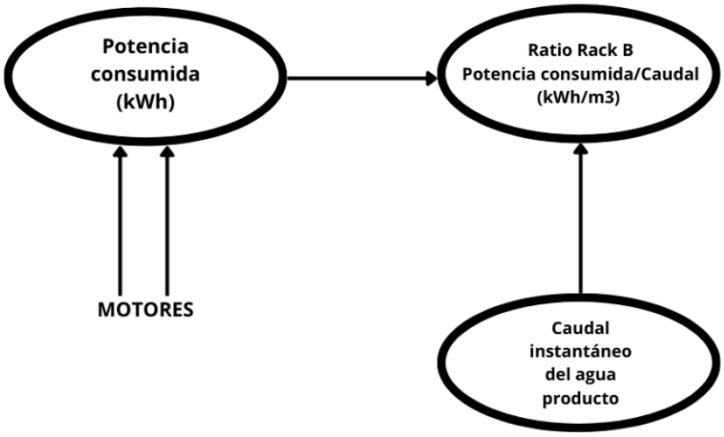

Figura 2: Cálculo variable objetivo

El consumo específico de energía(ratio) se define como la relación entre la energía eléctrica consumida (en kWh) y el volumen de agua desalada producida (en m³) que vemos en la *Figura 2*: Cálculo variable objetivo. Para calcular este valor, se suman los consumos energéticos (en kW) de todos los equipos motorizados involucrados en el proceso dentro del rack (como bombas y motores eléctricos), y se divide por la suma de los caudales instantáneos del agua producto generada por ese mismo rack.

La reducción de este consumo específico puede lograrse mediante el ajuste de variables operativas accesibles a través del sistema de control de la bomba, tarea que realizan los operarios. Si estos ajustes se aplican correctamente, es posible mantener estable —o incluso disminuir— el consumo energético. En cambio, una configuración inadecuada puede provocar un aumento significativo del mismo.

1.2. Motivación

Actualmente, se está dando un crecimiento bastante importante en el uso de técnicas de Inteligencia Artificial o Machine Learning, que cada vez se aplican a más campos. Ejemplos populares de esto son el auge de ChatGPT, o los numerosos algoritmos de recomendación que pueblan las aplicaciones de uso cotidiano.

Estas técnicas han adquirido también bastante popularidad para su aplicación en procesos industriales. Siguiendo con esta tendencia, en este proyecto se experimenta con el uso de técnicas de Machine Learning para la monitorización y predicción de comportamiento de la planta desaladora.

Es por este motivo por lo que se intenta utilizar machine learning en este caso en particular. Si queremos ahondar más en el tema, podemos matizar que el modelo propuesto permitirá anticipar el ratio de producción de la planta, así como, los cambios que se deben hacer en las variables que los operarios pueden gestionar.

La motivación principal se basa en la mejora económica que puede conseguirse si se es más eficiente en este campo. Si el modelo propuesto puede predecir con precisión el valor del ratio, podemos aprovechar los valores de las variables que el operario puede tocar, para que se ajusten los valores indeseados y se mantenga estable. Además de ser más seguro, también se podría ver beneficiada la vida útil de la maquinaria, contribuyendo esta última parte también a una mejora económica.

1.3. Objetivos

Este proyecto tiene como objetivo la minimización del ratio de consumo específico del Rack de ósmosis mientras que el Rack tiene un óptimo funcionamiento (que mantenga el rendimiento en su tarea, con el mínimo consumo posible, no superando barreras físicas). Esto se conseguirá realizando los siguientes pasos:

- Análisis de los datos que se proporcionan acerca del rack.
- Desarrollo de un modelo predictivo que aproxime los siguientes valores del ratio de consumo.
- Evaluar diferentes modelos y su desempeño en el mismo problema.
- Conseguir implementar un cambio automático en las variables que se pueden modificar, para que el ratio se minimice.

1.4. Organización de la memoria

En este apartado, se muestra de qué manera está estructurada la memoria a partir de aquí:

- El punto 2 explica las diferentes tecnologías y herramientas que se han utilizado durante el desarrollo del proyecto, así como los diferentes problemas y ventajas que se han dado.
- El punto 3 introduce el problema de datos al que hace referencia este TFG y aclara de qué forma se va a realizar.
- El punto 4 explica términos como el machine learning y el data mining, para después explicar sus fases de manera detallada, ofreciendo un buen marco teórico para todo lo que se ha realizado en el proyecto.
- El punto 5 contiene la metodología utilizada durante el proyecto, así como sus fases.
- El punto 6 explica el desarrollo del modelo, explicando de forma teórica cada uno de los modelos utilizadas, y, de forma práctica,

preprocesamiento y la diferente selección de diferentes parámetros durante el proceso.

- El punto 7 contiene el análisis del rendimiento de cada uno de los modelos, así como la visualización y comparación de los diferentes resultados.
- El punto 8 contiene las conclusiones sacadas del proyecto, así como los pasos futuros a seguir para poder continuar con él.

2.TECNOLOGÍAS Y HERRAMIENTAS

2.1. Python

El lenguaje que he utilizado a la hora de desarrollar el modelo de optimización del que trata el proyecto es Python. Esto se debe en mi caso, a que la facilidad y garantías que te dan las diferentes librerías que tiene para importar y que te aportan todo lo que necesitas para generar un buen modelo junto a sus métricas. En mi caso específico, se utilizan tanto Python 3.10 como la versión 3.13 en función de los distintos modelos implementados.

2.1.1.Librerías utilizadas

En cuanto a las librerías que se han utilizado, dejo aquí la explicación de cada una de ellas:

Pandas: es una librería que se utiliza para el manejo y procesado de datasets tabulares. En el contexto del proyecto, se ha utilizado a la hora de leer, procesar o escribir datos.

Numpy: para funciones matemáticas.

Seaborn: creación de algunas gráficas referentes a rendimiento o elección de unas señales frente a otras de manera más visual.

matplot.pyplot: permite crear visualizaciones en Python, por ejemplo: para gráficos de dispersión, histogramas etc.…

sklearn: derivada de scikit-learn mediante la cual podemos entrenar los modelos, elección o selección de los mejores hiperparámetros, normalización de los datos en caso de ser necesario e incluso métricas para la evaluación de los propios modelos.

Xgboost: librería que se encarga de darnos acceso al modelo de xgboost.

Catboost: librería que se encarga de darnos acceso al modelo de catboost.

Tensorflow: nos da acceso y la capacidad de construir y entrenar modelos de redes neuronales.

2.2. Fast Monitoring

Módulo de la empresa que se encarga de recoger todos los datos referentes a la planta sobre la que vamos a desarrollar el modelo del que se habla. Recoge todas las señales en tiempo real, permitiendo la explotación de esos datos, o simplemente su almacenaje.

2.3. Sinema Rc Client

Es un cliente de conexión de redes industriales que se utilizará para acceder de forma remota a los datos del cliente, en este caso para poder aprovechar el módulo de Fast Monitoring sobre el que hablamos anteriormente.

2.4. Visual Studio Code

Editor que se ha utilizado para la realización del código del problema debido a su adaptabilidad y sencillez a la hora de programar.

3.PLANTEAMIENTO DEL PROBLEMA DE ANÁLISIS

Para entender el proceso que se llevará a cabo, debemos de tener claro al problema de datos al que nos enfrentamos, como también debemos saber en el entorno de datos en el que nos encontramos.

En este problema analizamos una bomba de agua, que tiene un total de 35 señales que se recogen periódicamente de forma que tenemos datos siempre actualizados. Estas señales se muestran en *Tabla 1*, entre las cuales se encuentra la que se considera como variable objetivo (variable sobre la cual se quiere intentar entrenar a los modelos para que realicen predicciones precisas entorno a ella), y es el Planta_Ratio_Rack que se refiere al ratio de consumo de la bomba (kWh/m³).

El problema gira en torno a esta variable, ya que se busca minimizar en la medida de lo posible en todo momento el ratio de consumo, de manera que no haya picos de subido y sea estable.

Entorno al enfoque de este problema, primero debemos observar todas sus variables en su totalidad para poder entender de qué manera abordaremos el problema.

TAG	DESCRIPCIÓN	Modificable por operador	UNIT
BFV_315B_CVALVE2_PV	Válvula de control Rack Ósmosis B	MODIFICABLE POR OPERADOR	%
BFV_319B_CVALVE2_PV	Válvula del control de flujo permeado Rack Ósmosis B	MODIFICABLE POR OPERADOR	%
CIT_231_AINPUT2_PV	CIT-231 - Conductividad de entrada a bastidores	NO MODIFICABLE	%
CIT_311B_AINPUT2_PV	CIT_311B - Conductividad de permeado rack B	NO MODIFICABLE	mS/cm
CIT_431_AINPUT2_PV	CIT_431 - Conductividad de agua tratada	NO MODIFICABLE	mS/cm
DPT_312B_AINPUT2_PV	DPT_312B - Diferencial de entrada / salida rack B	NO MODIFICABLE	bar
PLANTA_ESTADO_RACK	Estado de Funcionamiento del RACK	MODIFICABLE POR OPERADOR	-
PLANTA _RATIO_RACK	Ratio Producción Rack B (kWh/m3)	NO MODIFICABLE	kWh/m³

FT_311B_AINPUT2_PV	FT_311B - Caudal de alimentación Rack B	NO MODIFICABLE	m³/h
FT_312B_AINPUT2_PV	Flujo de entrada de agua de mar al ERI Rack Ósmosis B	NO MODIFICABLE	m³/h
FT_313B_AINPUT2_PV	Caudales de entrada en las bombas Booster Rack Ósmosis B	MODIFICABLE POR OPERADOR	m³/h
FT_314B_AINPUT2_PV	FT_314B - Caudal de permeado Rack B	NO MODIFICABLE	m³/h
OI_311B_DF_HP_LP	OI_311B_DF_HP_LP - Factor de conversión Rack B	NO MODIFICABLE	%
PHT_121_AINPUT2_PV	PHT_121 - pH de agua bruta	NO MODIFICABLE	pH
PIT_313B_AINPUT2_PV	PIT_313B - Presión de entrada a rack B	MODIFICABLE POR OPERADOR	bar
PIT_315B_AINPUT2_PV	Presión de entrada del agua de mar al sistema ERI Rack Ósmosis B	NO MODIFICABLE	bar
PU_311B_SEPAM_AD1_5	Corriente Fase A Bomba alta presión Rack Ósmosis B	NO MODIFICABLE	A
PU_311B_SEPAM_AD1_6	Corriente Fase B Bomba alta presión Rack Ósmosis B	NO MODIFICABLE	A
PU_311B_SEPAM_AD1_7	Corriente Fase C Bomba alta presión Rack Ósmosis B	NO MODIFICABLE	A
PU_311B_SEPAM_AD3_6	Potencia instantánea Bomba alta presión Rack Ósmosis B	NO MODIFICABLE	kW
PU_312A_ATV_Puissance	Potencia instantánea Bomba Booster Rack Ósmosis B	NO MODIFICABLE	kW
PU_312B_ATV_AD1_2	Corriente Trifásica Bomba Booster Rack Ósmosis B	NO MODIFICABLE	A
TDT_231_AINPUT2_PV	TDT-231 - Turbidez entrada de bastidores	NO MODIFICABLE	NTU
TT_121_AINPUT2_PV	TT_121 - Temperatura de agua bruta	NO MODIFICABLE	ºC
TT_231_AINPUT2_PV	TT-231 - Temperatura de entrada a bastidores	NO MODIFICABLE	ºC
TT_311B_AINPUT2_PV	TT_311B - Temperatura entrada a bastidor B	NO MODIFICABLE	ºC
TT_311X_Moy	TT-311X - Temperatura de entrada a bastidores en funcionamiento (media)	NO MODIFICABLE	ºC
PIT_314B	"LP-out: low pressure out" Presión de concentrado en la salida de los ERIS	NO MODIFICABLE	bar
CIT_312B	Conductividad de concentrado en la salida de los ERIS "LP-out: low pressure out"	NO MODIFICABLE	mS/cm²
DPT_313B	Diferencial (HP-in y HP-out) entre alta presión de entrada concentrado y salida agua filtrada de los ERIS.	NO MODIFICABLE	bar
PC_312B	"HP-out: high pressure out" Presión calculada de agua filtrada salida de los ERIS & aspiración de la bomba booster.	NO MODIFICABLE	bar
PC_311B	"HP-in" Presión calculada de concentrado en la salida del bastidor & entrada a los ERIS	NO MODIFICABLE	bar

FC_311B	Caudal calculado de concentrado en la salida del bastidor & entrada a los ERIS	NO MODIFICABLE	m³/h
DPT_311B	Diferencial de Presión entre PC_312 & PIT_313	NO MODIFICABLE	bar
PIT_311	Presión colector de agua filtrada que alimenta a los bastidores	MODIFICABLE POR OPERADOR	bar

Tabla 1: Señales de la bomba de agua

En *Tabla 1*: Señales de la bomba de agua podemos ver todas las entradas de las que dispone la bomba de agua, tomando cada una un valor y con su propia unidad. Sobre estas características trabajará el modelo.

En cuanto al valor más importante en el que nos debemos fijar es nuestra variable objetivo o Planta_Ratio_Rack se mueve en unos valores entre 2.30 y 2.45, teniendo una media total de unos 2.37. En este problema de datos, el cambio de la variable a lo largo de 5 minutos suele ser sobre una o un par de centésimas por lo que es clave que el modelo tenga una alta precisión para que pueda llegar a detectar esto.

Cabe destacar que todos los datos mencionados se obtienen a través del módulo de la empresa “Soincon” llamado “Fast Monitoring” y que permite que pueda obtener todo lo mencionado de forma inmediata y con toma de datos en tiempo real (se recogen los datos en la planta en tiempo real, y accediendo a la plataforma podemos descargar los datos seleccionados).

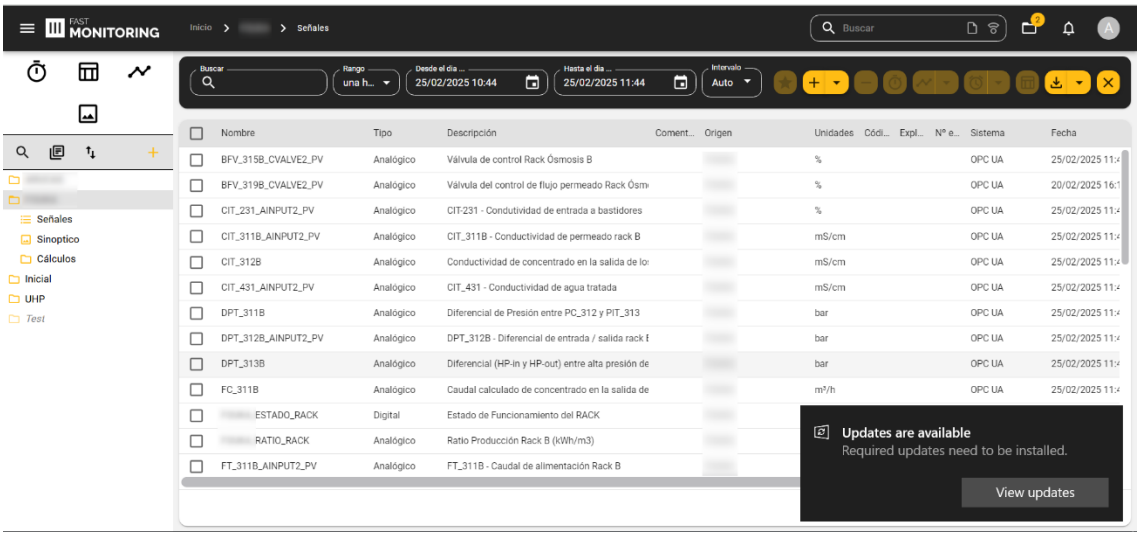

Figura 3: Módulo Fast Monitoring

En la *Figura 3* podemos ver el módulo indicado donde se nos permite elegir las diferente variables y fechas de las que queremos obtener los datos.

4.DATA MINING Y MACHINE LEARNING

En esta sección introducimos los términos data mining y machine learning, para facilitar la comprensión tanto de la metodología de trabajo utilizada (CrispDM) como del trabajo en sí realizado durante el desarrollo del proyecto.

4.1. ¿Qué es el Data Mining?

En la etapa en la que el ser humano se encuentra, la información es uno de los recursos más importantes que se pueden llegar a manejar, es por esto por lo que se buscan formas de explotar todo su potencial para predicciones y análisis.

Aunque la minería de datos como tal es relativamente reciente, sus fundamentos se remontan a técnicas estadísticas y matemáticas desarrolladas hace siglos, como el Teorema de Bayes o los métodos de regresión. Con la evolución tecnológica, estas bases dieron lugar a lo que hoy conocemos como Data Mining, una disciplina clave para transformar grandes volúmenes de datos en conocimiento útil.

4.2. ¿Qué es el machine learning?

Una vez tenemos claro lo que es el data mining podemos definir lo que consideramos como machine learning. El machine learning (aprendizaje automático) se basa en algoritmos que lo que buscan es aprender de unos datos sin tener una programación exacta para ciertas tareas. En general, son términos bastante similares, pero que tienen diferencias sutiles.

En la actualidad es algo que se usa mucho sin darnos cuenta, como en recomendaciones de ciertas películas o diagnósticos médicos, dándonos cuenta de que es muy versátil.

Al afrontar un problema referente a datos, necesitamos saber cómo vamos a conseguir unas explotación buena de los mismos y traducir los problemas de negocio en tareas de data mining, de forma que podamos aplicar posteriormente los algoritmos de machine learning en nuestro beneficio.

4.3. CrispDM y sus fases

CrispDM (Cross Industry Standard Process for Data Mining) se encarga de modelar de qué forma hay que comportarse frente a estos problemas, y como dividir las diferentes partes del procesos con fases comunes a todos. Es una manera de simplificar todas las partes implicadas. Además, al tener un estándar común el entendimiento en el sector aumenta al ser siempre lo mismo con diferente problema.

A su vez, un proyecto de data mining se puede dividir en distintas fases. En esta figura podemos ver cómo se dividen en 6 fases diferentes, con todo lo que se debe generar en cada una de ellas. Ahora entraremos a definir cada una de las fases de esta metodología de forma breve:

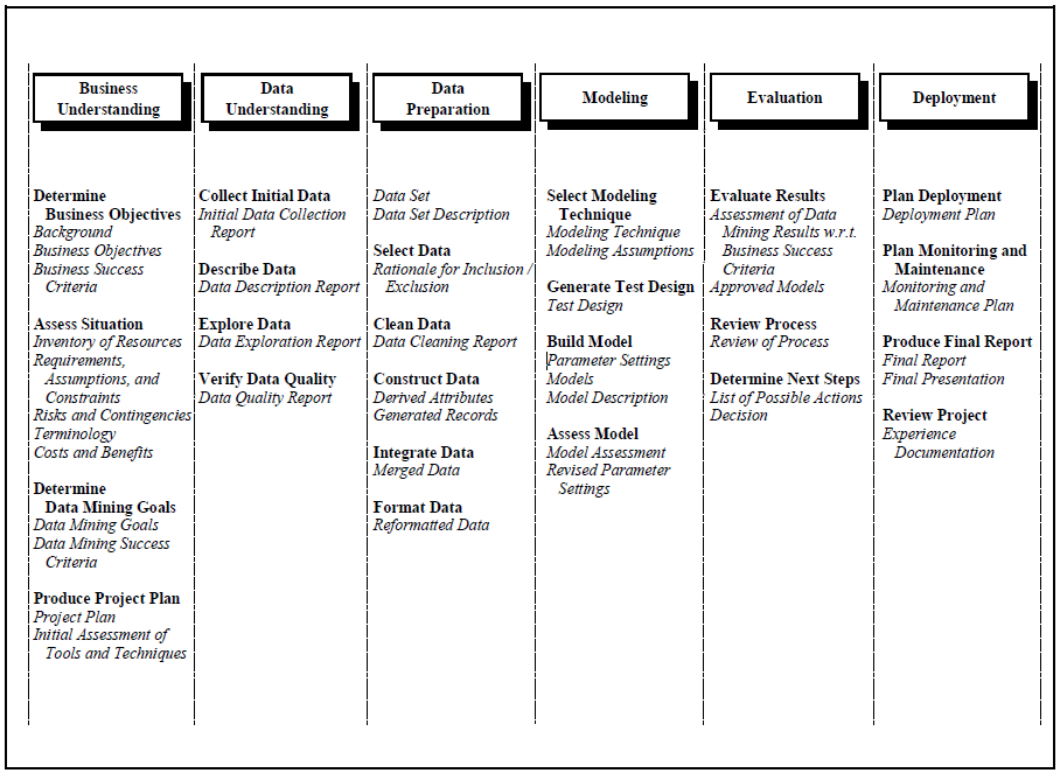

Figura 4: Fases de CrispDM

Business Understanding. Referente al entendimiento del negocio, valorando objetivos y requisitos del proyecto e irlo transformando a un problema de datos.

Data Understanding. Familiarizarnos acerca de los diferentes datos que disponemos, ver de qué manera están distribuidos, de que calidad son y que tipo de valores suelen tomar.

Data Preparation. Una vez hemos realizado un estudio acerca de los datos, nos centraremos en su preparación mediante técnicas de preprocesado y transformación de datos para lograr obtener buenos resultados.

Modeling. En el modelaje seleccionaremos en función de los datos que se nos ofrecen los distintos algoritmos a utilizar y así como los parámetros que va a utilizar el modelo en cada caso.

Evaluation. Una vez hemos preparado uno o varios modelos y antes de desplegarlo, se evaluará mediante diferentes tipos de métricas el rendimiento de este, para conocer de qué manera se desenvuelve frente a un problema real.

Deployment. Por último, se desplegará el modelo de manera que todo este conocimiento y datos que hemos obtenido se puedan llegar a utilizar para la predicción para la que ha preparado.

5.DESARROLLO DEL MODELO

5.1. Modelos

Los algoritmos de machine learning son el componente que adquiere mayor relevancia al abordar un problema relacionado con datos. La elección de el que creemos más correcto define como se lograrán tratar los datos y, por tanto, sacar unos buenos resultados o unos no tan buenos.

Un modelo de machine learning es un algoritmo que, en base a ciertos datos de los que se alimenta (los datos iniciales), detecte patrones y regularidades que dan en dichos datos, con intención de tomar decisiones y realizar predicciones sobre esos datos, como se puede ver en la siguiente figura.

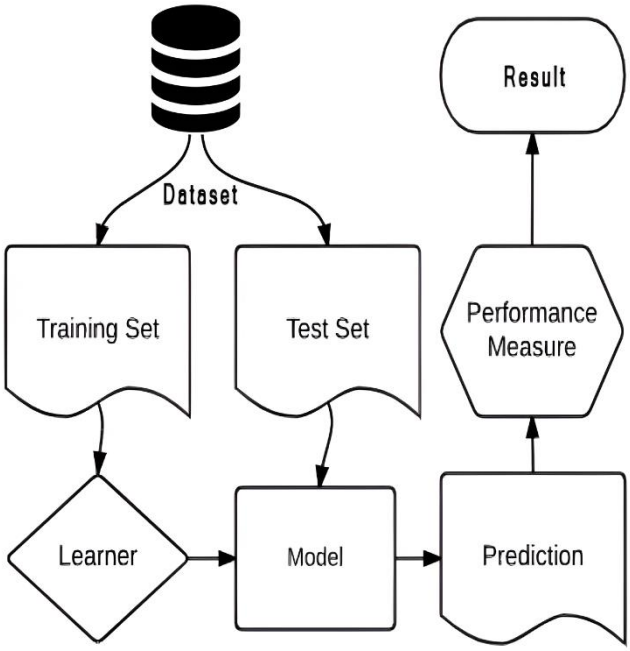

Figura 5: Proceso Modelo Machine Learning

Para comenzar, se realiza una división de los datos iniciales en un conjunto de entrenamiento y otro de prueba. Con los primeros el modelo realizará lo que hemos dicho anteriormente y, con el segundo conjunto que hemos definido se probará sus resultados entorno a la variable objetivo, realizando predicciones y midiendo en cuantos se acercan respecto a datos reales.

En este proyecto, es relevante explicar de qué manera trabajan dos tipos distintos de modelos de lenguaje: un Classifier o un Regressor.

Por un lado, un Classifier aborda problemas en los que la variable objetivo es de tipo discreto y, en el caso de este proyecto, binaria, ya que solo puede tomar dos valores concretos. Esto convierte el problema en uno de clasificación binaria, donde el objetivo es predecir cuál de los dos estados posibles tomará la variable. Un ejemplo típico de este tipo de problema es la predicción de enfermedades, como determinar si una persona tiene cáncer o no.

Por otro lado, un Regressor se utiliza cuando la variable objetivo es numérica, y lo que necesitamos es una aproximación de un número continuo de la forma más exacta posible (podrían ser los goles exactos que se van a meter en un partido).

En este problema se han utilizado un total de 3 modelos: XgBoost, Catboost y redes neuronales. Con estos modelos se busca un máximo rendimiento en el contexto del problema. Posteriormente, se comentará de qué forma trabaja cada uno de los modelos para su entendimiento.

En primer lugar, XgBoost es un modelo de los que se llama gradient boosting (consiste en utilizar árboles de decisión incrementalmente para conseguir realizar un modelo más fuerte). Se encarga de implementar varios árboles en paralelo que se encargan de ir aportando a la solución final. Un árbol de decisión es una herramienta en forma de diagrama de árbol que no solo permite visualizar las diferentes opciones y cómo afectan al problema, sino que también estructura las decisiones mediante 'preguntas' que se realizan a los datos en cada nodo del árbol.

Referente a los parámetros del modelo hay varios que merece la pena comentar. Los más utilizados son referentes a la tasa de aprendizaje, el número de iteraciones, máxima profundidad del árbol, etc. Además de estos cabe destacar algunos parámetros que se utilizan para regular cuanta información y de qué manera aprende el árbol como la submuestra con la que trabaja, columnas o por nodo de cada árbol. También argumentos importantes para su optimización y mejora de los resultados como el early stopping rounds que se explicará posteriormente.

Por otro lado, tenemos el Catboost que al igual que XgBoost es un algoritmo de la clase gradient boosting. Una característica extra que aporta es que maneja muy bien las variables categóricas.

Por último, las redes neuronales permiten construir distintos tipos de modelos sin necesidad de escribir código excesivamente complejo. La creación de una red implica definir las diferentes capas que la componen, seleccionar el optimizador a utilizar, la función de pérdida y las métricas con las que se evaluará el rendimiento. También es necesario especificar las entradas y salidas del modelo: las entradas corresponden a las variables que describen el estado del sistema

(en este caso, las señales recogidas de la bomba), mientras que la salida es la variable objetivo que se desea predecir. Para su entrenamiento, se indica el número de épocas, que representa cuántas veces la red verá por completo el conjunto de datos para ajustar sus parámetros, así como el tamaño de los lotes de datos (batch size) y el porcentaje de datos que se reservará para validación.

5.2. Prototipado

En cuanto vemos los valores en los que se mueve la variable objetivo (entre 2.30 y 2.45 normalmente) nos damos cuenta de que deberíamos utilizar un Regressor en vez de un Classifier (también referente al ratio de consumo). Sin embargo, con el objetivo de desarrollar un modelo de predicción simplificado de los valores, se decidió con ciertas features ya seleccionadas por la empresa que se realizase un Classifier del problema. Con esto simplificaríamos la dificultad de lo que queremos conseguir, haciendo distinción entre cuando un valor sube o baja en base a los valores de dichas features, siendo la variable objetivo un binario que toma valores de 1 si sube o 0 si baja (se utiliza un valor anterior de la variable objetivo para calcularlo como predictor).

Con esto me familiaricé con el modelo de Xgboost, que fue el que utilicé para este prototipo, y que más tarde se explicará junto con los otros dos modelos utilizados.

Sin embargo, a pesar de haber simplificado el problema de gran manera, en este punto se presentó uno de los mayores obstáculos en el desarrollo (a pesar de ser solo un prototipo). Esta es la primera vez que aparece la curva ROC, una herramienta gráfica utilizada para evaluar el rendimiento de modelos de clasificación binaria. En este caso, la curva mostró una diferencia notable entre el comportamiento del modelo en los datos de entrenamiento y en los de validación, lo que indicaba un problema en la generalización del modelo. Se

ofrece una explicación más detallada sobre este tipo de gráfica en *6.1. Resultados distintos modelos*.

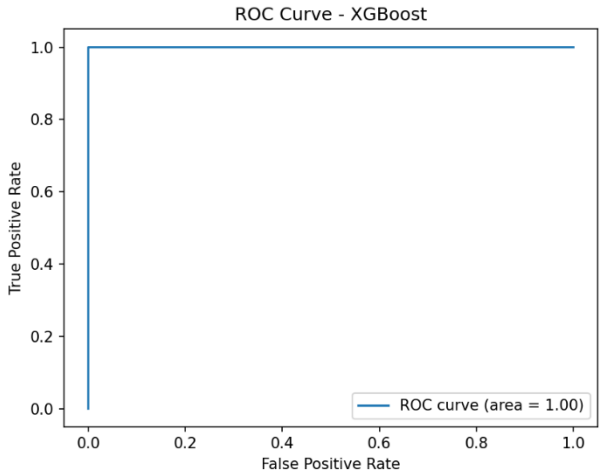

Figura 6: Caso de overfitting

En esta gráfica podemos ver cómo, según las métricas, el modelo tiene un desempeño perfecto a la hora de clasificar. Aunque podríamos pensar que esto es bueno, que el modelo tenga un desempeño tan perfecto es indicador de que el modelo no tiene la suficiente capacidad de generalización y simplemente, se aprende de memoria cada caso en lugar de aprender (suponemos que se está filtrando información, es decir, que el modelo está obteniendo y memorizando los datos que le pasamos para que evalúe en algún momento del proceso).Uno de los principales retos al entrenar modelos de machine learning es el overfitting o sobreajuste como en la *Figura 6*, que ocurre cuando un modelo aprende demasiado bien los datos que se le proporcionan, haciendo que el modelo no sepa diferenciar, solo aprendiéndose todos los casos de memoria. Para lograra acabar con estos problemas, se utilizan técnicas como la validación cruzada y el early stopping.

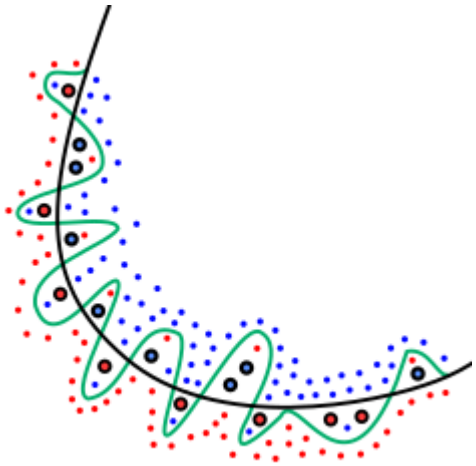

Figura 7: Overfitting

El método más común es la validación cruzada (en este caso la k-fold) que se encarga de partir los datos en los folds que queramos utilizar. Con esto ganamos en que se irán combinando y comprobando unos con otros de manera aleatoria lo que reducirá mucho las posibilidades de overfitting.

El early stopping es una técnica de regularización que se basa en parar el aprendizaje del modelo cuando ya avanza demasiado poco en cada iteración, o directamente ya no avanza nada. Esto aparte de hacer que el modelo no tenga overfitting, nos aporta ahorro de coste computacional, y, en caso de ser datos grandes ahorro de tiempo.

Después de estar varias semanas con problemas respecto al desempeño del modelo, implementé diferentes técnicas como el early stopping rounds y la validación cruzada para tratar de mitigar el comportamiento del modelo y que se centrase en aprender más. Al principio esto tuvo algo de impacto, pero también hice uso de optimización de hiperparámetros con el objetivo de obtener unos resultados razonables. Después de todos estos cambios obtuvimos estos resultados:

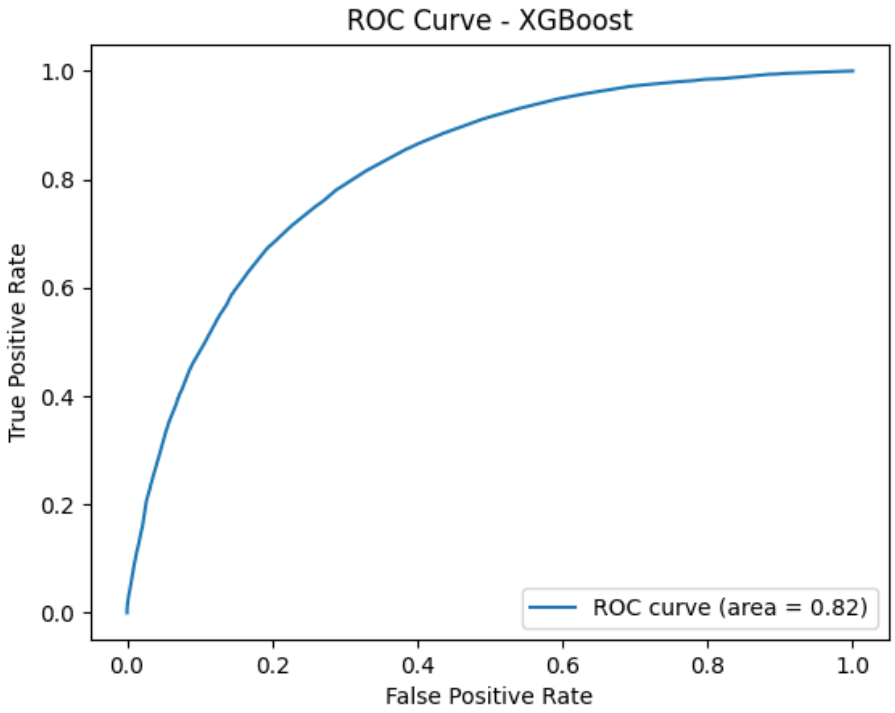

Figura 8: Resultado adecuado modelo

Podemos observar que el modelo tiene mejores resultados que el anterior, y, cuando se obtuvo esto se decidió pasar a la siguiente fase donde se iniciaría un modelo mediante un Regressor, pero no el Classifier.

5.3. Selección de la muestra

Como ya aclaramos anteriormente, los datos se obtienen de un módulo llamado Fast Monitoring. Aquí podemos elegir de qué forma queremos los datos o la cantidad deseada.

Respecto a los tiempos utilizados, se han cogido datos de todo el 2024 y noviembre y diciembre son los datos que utilizaremos como validación.

La primera duda que surgió en su desarrollo fueron los tiempos de recogida de estos datos, a la hora de entrenar al modelo. El módulo de Fast Monitoring nos permite la descarga/recogida de estos datos en intervalos de 1 minuto, 5 minutos y 1 hora.

Casi al principio del desarrollo se descartó casi totalmente el uso de los intervalos de 1 hora, ya que al ser demasiado grandes y una variable que tiene cambios pequeños en poco tiempo (la variable objetivo cambia unas pocas centésimas) y necesitamos que el modelo sea capaz de obtener patrones que ocurren en tiempos más cortos.

Respecto a los intervalos de 1 minuto, frente a los de 5, se empezaron haciendo pruebas con ambos valores. Mientras iba avanzando el desarrollo, se tomó la decisión de que se haría respecto a los valores en intervalos de 5 minutos. Esto se debe a que a parte que el objetivo final del modelo es que el operario pueda hacer los cambios correspondientes en las variables modificables. Este para poder controlar el ratio de producción, cuadra más que sea en espacios de 5 minutos donde se detecten esos patrones y lo hagan de forma similar a como se hará al final. A su vez, los tiempos de 1 minuto introducen mucho ruido y hace que las métricas no sean del todo correctas dependiendo de en qué modelos se utilice. Es por esto por lo que se tomó la decisión de tomar intervalos de 5 minutos.

5.4. Selección de características en los diferentes modelos

La selección de características (nos referiremos a ellas con el término “features” a lo largo del documento) es una de las partes más importantes entorno al machine learning y la explotación de los diferentes datos. Esto se debe a que es la fase donde elegimos sobre los datos que tenemos, y seleccionamos los que son más importantes o aportan algo a la hora de obtener buenos resultados.

Esto de no poder cambiar las decisiones que vamos tomando, puede ser un problema en el caso de que queramos cambiar algo. Este es el llamado efecto de anidamiento. Para arreglarlo, se inventaron métodos distintos métodos.

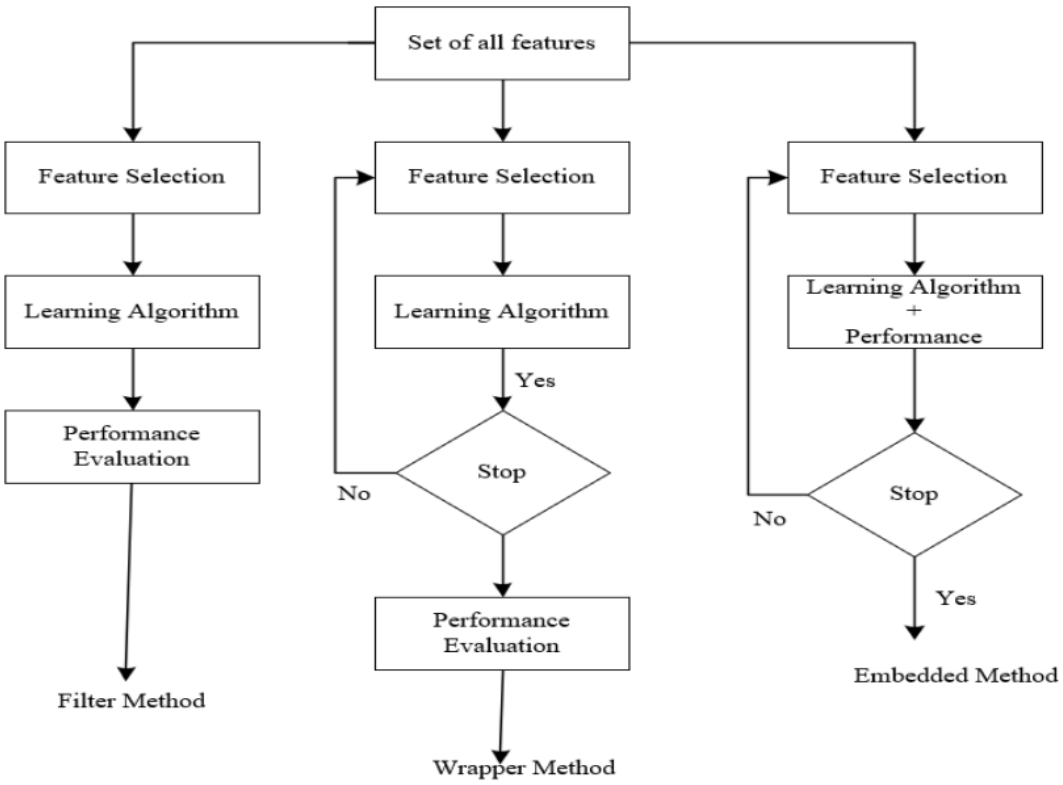

Figura 9: Métodos selección de características

En esta figura podemos observar los distintos métodos de selección de características que se explicarán posteriormente

En el caso de filtrado, son los que se basan en las relaciones entre las variables, sacando estadísticas sobre ellas. Tenemos por ejemplo la correlación de Pearson que se encargar de calcular la relación en torno a -1 y 1 (si es -1 la feature es inversamente proporcional a la otra, y cuanto más se acerque a 1 hay una relación posible).

En el caso de los envolventes, que tratan la selección de características con lo que puede ser un proceso de caja negra. Básicamente a partir de un algoritmo de aprendizaje (como árboles o cualquier tipo de regresión) se trata de a partir del conjunto de características y ver de qué manera se desenvuelven respecto del problema.

Por último, para los incorporados, que se refieren a cuando el propio modelo de aprendizaje, durante su entrenamiento, es capaz de seleccionar las diferentes features. Tiene bastante precisión y menos coste que los casos envolventes. Algoritmos como el XgBoost, o muchos de los tipos de árboles son capaces de medir la importancia de las diferentes características.

Lo que se tenía planteado en un principio era utilizar para este estudio solo las características que eran modificables por el operador, porque al solo fijarnos en estas, se puede deducir que variable se puede modificar para provocar un cambio de la manera que deseamos en la variable objetivo. Sin embargo, esto no es posible ya que el número de características que tenemos que son modificables son muy pequeñas, por lo que no es suficiente información a la hora de hacer que el modelo aprenda. Es por esto por lo que se decide que se hará la selección con todas las características buscando el mejor rendimiento del modelo, para posteriormente fijarnos en el impacto de estas variables en concreto.

Ahora haremos uso de diferentes técnicas para ver con que características debemos contar a la hora de entrenar el modelo y cuales no debemos tener en cuenta.

Lo primero que se realizó fue la generación de una matriz de correlación referente a todas las features del problema (*Figura* 11). El objetivo de esto es ver de manera superficial si tenemos relaciones claras con la variable objetivo y si podemos explotarlo desde aquí:

```
# Calcular la matriz de correlación
corr_matrix = data.drop(columns=['Moment']).corr()

# Graficar el heatmap de correlación
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap="coolwarm", linewidths=0.5)
plt.title("Matriz de Correlación ")
plt.show()

# Ajustar pandas para mostrar todas las columnas y un mejor formato
pd.set_option('display.max_columns', None)  # Mostrar todas las columnas
pd.set_option('display.float_format', '{:.2f}'.format)  # Limitar decimales para mejor lectura
```

Figura 11: Código generación matriz

Figura 12: Matriz de correlación

De esta matriz podemos sacar mucha información acerca de la mayoría de las características del problema. En el caso de las variables de temperatura nos podemos dar cuenta de que comparten una correlación muy alta entre todas ellas. Por tanto, en el caso de querer añadir temperaturas, con una feature sería más que suficiente), que es el mismo caso que tenemos con las variables PU (variables para la potencia).

Entorno a la variable objetivo, que es a la que más atención le debemos de prestar (PLANTA_RATIO_RACK_DEFAULT), tenemos un par de relaciones importantes referentes a FC_331B_Default que tiene un índice del 0.68 por lo que ya tiene un patrón bastante importante como para que sea muy tomada en cuenta. Por otro lado, tenemos la feature FT_314B_AINPUT2_PV_Default que tiene una relación casi inversamente proporcional (-0.93) con nuestra variable, por lo que puede aportar información muy interesante.

Por último, en términos generales, las temperaturas también tienen una correlación negativa considerable con la variable objetivo, a parte de algunas más de las que se mueven sobre el 0.3 - 0.45 tanto con correlación negativa, como positiva.

También se realizó una prueba utilizando el modelo XgBoost, para lograr de encontrar patrones dentro de grandes cantidades de datos. El objetivo de esta prueba fue que el modelo identificara qué variables tienen más impacto en la predicción del consumo energético, como por ejemplo los llamados "lags", que son medidas del consumo en momentos anteriores (por ejemplo, hace 1 minuto, 5 minutos, etc.), utilizadas para ayudar a predecir el consumo actual.

En este caso, se generó una gráfica de "importancia de características" (feature importance), donde se observa qué valores históricos o condiciones tienen mayor peso a la hora de estimar la variable objetivo.

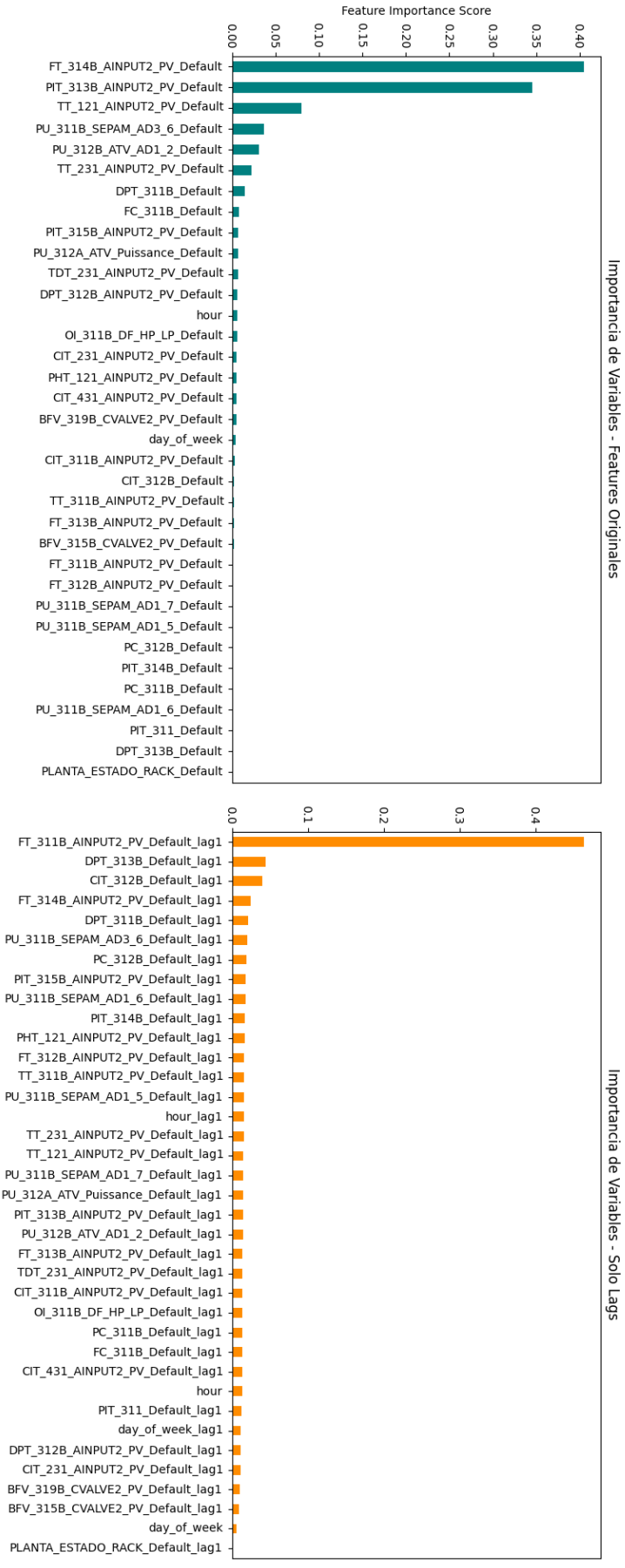

Figura 13: Importancia de Features

En esta gráfica se puede observar que el valor más grande se refiere a las variables que este modelo considera más importante. En el caso de las features, tienen especial importancia el FT_314B_Ainput2_PV_Default (Caudal de permeado Rack B) y el Pit_313B_Ainput2_PV_Default (Presión de entrada a rack B). En cuanto a los lags solo le da importancia al FT_311B_Ainput2_PV_Default (Caudal de alimentación Rack B).

Después de estas comprobaciones decidí que iba a hacer otra prueba de features mediante PCAS correspondiente a la *Figura 14*.

```
# Selección de características con SelectKBest
selector = SelectKBest(score_func=f_regression, k=10)  # Elegimos las 10 mejores
X_selected = selector.fit_transform(X_imputed, y)
selected_features = X_original.columns[selector.get_support()]
print("Características seleccionadas:", selected_features)
```

Figura 14: Features mediante PCAS

Al valorar todas las correlaciones *Figura 14*, con 10 features escogidas será suficiente para sacar los patrones correctos. En este caso las características seleccionadas son las siguientes:

```
Características seleccionadas: Index(['CIT_311B_AINPUT2_PV_Default', 'CIT_431_AINPUT2_PV_Default',
       'FC_311B_Default', 'FT_311B_AINPUT2_PV_Default',
       'FT_314B_AINPUT2_PV_Default', 'PU_311B_SEPAM_AD1_6_Default',
       'PU_311B_SEPAM_AD3_6_Default', 'TT_121_AINPUT2_PV_Default',
       'TT_231_AINPUT2_PV_Default', 'TT_311B_AINPUT2_PV_Default'],
      dtype='object')
```

Figura 15: Features seleccionadas

Este algoritmo se encarga de coger el número de características óptimas que le indiquemos, y estas son las 10 que elige. Una vez tenemos seleccionadas las features que vamos a utilizar, deberemos pasar al preprocesamiento. Pero antes se decidió dividir la feature 'Moment' en otras dos nuevas que sean hora y día de la semana y se incluyan a la hora de realizar el entrenamiento (*Figura 16*). También después de varias pruebas se consideró incluir dos lags, uno de temperatura y la otra referente del caudal del rack B.

```
data['hour'] = data['Moment'].dt.hour
data['day_of_week'] = data['Moment'].dt.dayofweek
```

Figura 16: Generación de nuevas columnas

Con esto, la selección de features estaba de momento realizada.

5.5. Preprocesamiento de datos

El preprocesamiento en el machine learning, se basa en pasar de unos datos generalmente ininteligibles en unos que sean más fáciles de manipular. Esto se debe a que normalmente los datos que se nos proporcionan pueden tener incongruencias (outliers) o simplemente no lo suficiente adecuados como nos gustaría. Este proceso se traduce en una herramienta para “ayudar” a que el modelo capte mejor las relaciones que se esconden entre los datos y que no se generen predicciones erróneas.

Esta parte del proceso es muy importante a la hora de poder empezara hacer uso de los datos y por consecuencia, empezar a sacar conclusiones entorno a ellos. Es por esto por lo que se le debe prestar mucha atención a este proceso.

Este proceso generalmente se divide en las siguientes 6 categorías que se describirán a continuación, así como se indica en la figura:

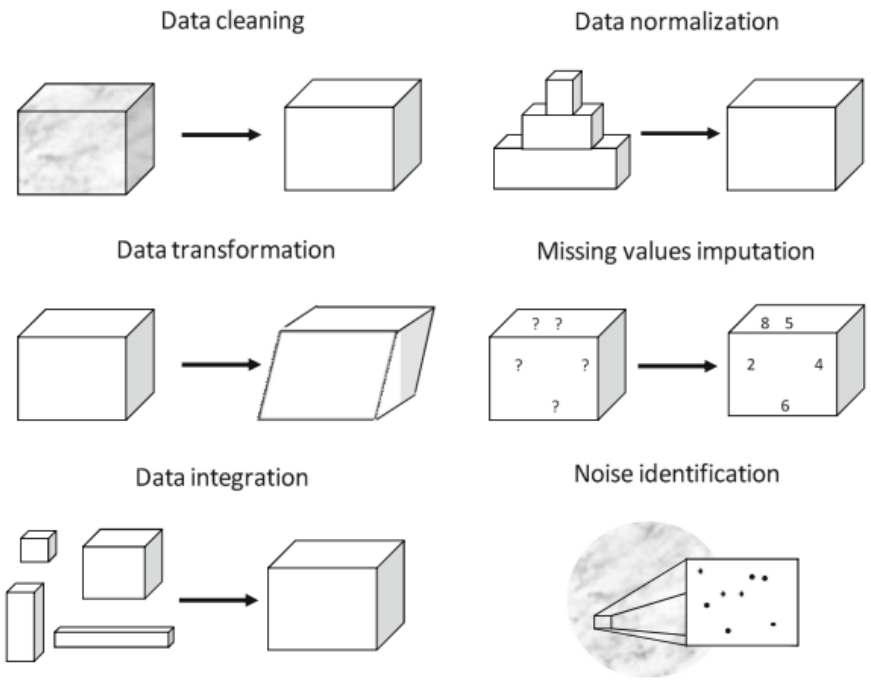

Figura 17: Tipos de preprocesamiento

En esta figura se muestran las diferentes formas en las que se deben de ir tratando los datos con el objetivo de ir mejorando su calidad y poder realizar una buena explotación, llegando a sacar unas conclusiones que nos aporten conocimiento acerca del problema.

Ahora nos centraremos en cada uno de los diferentes procesos a la hora de explotar los datos, y como se lleva a cabo (que se han utilizado en el proyecto).

En cuanto a la limpieza de datos (Data Cleaning), tenemos que valorar que muchas veces (o la gran mayoría) los datos proporcionados son inconsistentes, están incompletos o tienen muchos valores atípicos. En este punto es donde la limpieza de datos toma valor.

Un problema de falta de datos/incompletos (Missing values inputation) necesitamos tenerlo en cuenta para las métricas que después llevaremos a cabo, ya que esto puede generar malos resultados o que muchos datos no se puedan llegar a utilizar.

La solución fácil respecto a esto es borrar la columna que contiene esos datos si el porcentaje de nulos es muy elevado. Con esto nos ahorraremos limpiar fila por fila (lo cual sería rentable si el porcentaje de nulos no es demasiado).

En el otro caso de que los nulos no abarquen un porcentaje alto de los datos, tenemos una solución que puede seguir aprovechando esos datos. Esta se basa en rellenar los huecos con nulos con valores en la media de este campo, lo que asegura que son datos explotables (que cubre el punto de imputación de valores de la figura).

La transformación de los dato (Data Transformation). Por un lado, podemos simplificar tanto variables categóricas como numéricas, con el objetivo de ganar simplicidad en los datos que tenemos. Esto ayuda al modelo a identificar de manera sencilla relaciones, a pesar de perder algo de precisión en los datos. También tenemos en cuanto un suavizado de algunos datos, para que no tengamos ruido en la muestra, que puede generar dataos erróneos (similar a rellenar los huecos nulos con la media, comentado anteriormente). Por último, respecto a este punto, podemos llegar a generar campos (si tenemos ventas diarias en un problema de datos, quizás ver las semanales o mensuales nos ayude a captar alguna relación más) que nos ayuden con nuestro problema.

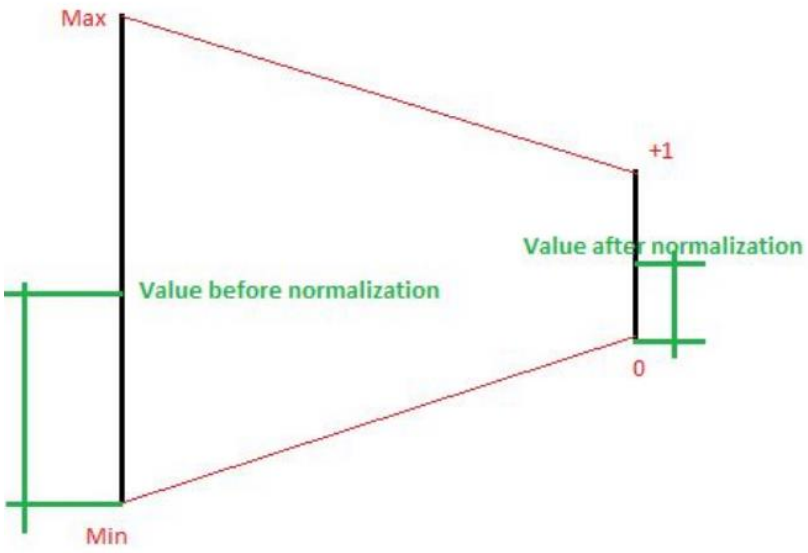

Figura 18: Normalización

La integración de los datos (Data Integration) es algo muy útil para un problema de datos en el que se utilizan varias fuentes de información que comparten datos comunes.

La normalización de los datos (Data Normalization) se puede meter en algo como transformación, es importante como para tratarlo como un tipo separado. Básicamente, podemos utilizar este método para que variables que manejan datos muy altos (imaginar una variable que se mueve en valores de 10000 a 20000) y otra con valores mucho más bajos (variable que se mueve entre valores de 0 a 10), así como podemos ver en la *Figura 18*. En mi caso se utiliza un Min-Max scaler para la normalización de los datos.

La identificación del ruido (Noise Identification). Con ruido nos referimos a una información incorrecta y que no cuadra con la media en la que se mueven los valores, y consigue que el modelo se retrase o incluso no aprenda de manera correcta.

También debemos nombrar a los outliers/valores atípicos. Son datos que pueden alterar los resultados de nuestra explotación de los datos. Esto se debe a que son valores muy alejados de la media que se maneja en los datos, por lo que hace que el modelo tenga predicciones erróneas y muy alejadas, lo que hace que los resultados finales se vean muy afectados.

Una vez tenemos identificado los distintos valores atípicos que contiene nuestra muestra de datos (como se mencionaba anteriormente), lo único que nos queda es eliminarlos. Para esto hay diferentes maneras que se comentarán posteriormente:

Forma manual: una vez vemos los datos que son considerados como outliers, podemos eliminarlos directamente. Esto es útil si son datos muy concretos y no nos requiere ir borrando a mano demasiados datos de nuestra muestra.

Cabe destacar que algunos de estos valores que no están sobre la media, puede que no deban de ser eliminados. Deberemos analizar el contexto y ver si esos datos se están dando debido a cierto fenómeno y, podrían estar dándonos informaciones muy valiosas en ciertas relaciones.

Esta fase es común de a todos los modelos que se han utilizado, por lo que se trata en conjunto. Al principio se hizo una limpieza general, de manera que todas las columnas con muchos nulos no se tuvieran en cuenta:

```
# Eliminar columnas con más del 20% de valores NaN
data = data.dropna(thresh=len(data) * 0.8, axis=1)
```

Figura 19: Limpieza de valores nulos en la muestra

Con esto *Figura 19* nos olvidamos de que features con pocos valores afecten los resultados de nuestras métricas. Es por esto por lo que una de las features que hemos elegido mediante PCAS antes queda descartada (FC_311B_Default) ya que tiene demasiados valores nulos como para trabajar con ella.

```
# Calcular la media, mínimo y máximo
stats = data.describe().loc[['mean', 'min', 'max']]
# Mostrar las estadísticas completas en la consola
print(stats)
```

Figura 20: Media, Mínimo y Máximo de cada feature

Una vez nos aseguramos de que no existen nulos en ninguna de las columnas que vamos a utilizar para entrenar el modelo, debemos de filtrar los datos que ya tenemos. Para esto tuve un problema respecto a ciertos outliers que me fui encontrando. El modelo tenía ciertos problemas o picos respecto a la variable

objetivo, que no sabía porque se daban, hasta que decidí observar los valores medios, mínimo y máximo de cada feature como en la *Figura 20.*

Una vez realizado esta vista nos damos cuenta de que la variable que consideramos objetivo en este problema tiene varios valores muy dispersos en comparación de la media. La variable objetivo, Planta_Ratio_Rack tiene una media de 2.37, marcando un valor mínimo de 1.3 y valor máximo de 300. Esto es totalmente incoherente sobre todo por lo alto ya que la diferencia es mucho mayor. Se decide investigar acerca de los diferentes valores y nos damos cuenta de que hay unos 15 valores que superan los 3 y unos 10 por debajo de los 2. Después de ver que estos casos aislados y que afectan de manera tan grave a las métricas, y, por lo tanto, al rendimiento del modelo, se eliminarán los valores que no se muevan entre 2 y 3 (*Figura 21*).

```
# Filtrar los datos de entrenamiento: Eliminar valores de 'PLANTA_RATIO_RACK_Default' fuera del rango [2, 3]
data = data[(data['PLANTA_RATIO_RACK_Default'] >= 2) & (data['PLANTA_RATIO_RACK_Default'] <= 3)]
```

Figura 21: Cutting aplicado en la variable objetivo

Esto, a pesar de que esta variable se mueve en rangos bastante pequeños (en los intervalos de 5 minutos se suele mover 1-2 décimas arriba o abajo) nos da margen para casos que no son los habituales (valores de 2.70), que el modelo sea capaz de dar un razonamiento más o menos adecuado en ese caso.

Evidentemente, estos valores tendrán menos cercanía de cara a predicciones ya que son más inusuales, pero se comprobará que el modelo se puede desenvolver más o menos bien en estos casos.

También tenemos que hacer un par de correcciones más sencillas respecto a otro par de features. Por un lado, en el caso de la variable objetivo la *Figura 22*, y la columna “Moment” encargada de la fecha y la hora en la *Figura 23*, las ajustamos para que sean tipo adecuado.

```
# Convertir 'PLANTA_RATIO_RACK_Default' a valores numéricos
data['PLANTA_RATIO_RACK_Default'] = pd.to_numeric(data['PLANTA_RATIO_RACK_Default'], errors='coerce')
```

Figura 22:Cambio de tipo de variable objetivo

```
# Convertir la columna 'Moment' a formato de fecha y hora
data['Moment'] = pd.to_datetime(data['Moment'], errors='coerce')
```

Figura 23: Cambio de tipo de variable a la columna Moment

```
# Eliminar filas con NaN en 'Moment' o 'PLANTA_RATIO_RACK_Default'
data = data.dropna(subset=['Moment', 'PLANTA_RATIO_RACK_Default'])
```

Figura 24: Eliminación de columnas con valores nulos en Moment o Variable objetivo

Además de tomar estas dos precauciones respecto a las variables, más importantes del problema, también debemos tener en cuenta que si alguna de las dos está nula se desechará la fila, como vemos en la *Figura 24*. Esto se debe a que si no tenemos la variable objetivo no nos servirán los datos y en el caso del tiempo es muy importante referente al problema.

Por último, como podemos observar en la *Figura 25* se cambió la única variable categórica de la que dispone el problema PLANTA_ESTADO_RACK. Esta tomaba valores de 'Activo' o 'Parado' y para que el modelo la pueda procesar (en este caso porque por ejemplo catboost es capaz de tratar bien las variables categóricas, por lo que se suprimiría) y realizar un correcto aprendizaje.

```
data['PLANTA_ESTADO_RACK_Default'] = data['PLANTA_ESTADO_RACK_Default'].map({'Activo': 1, 'Parado': 0})
```

Figura 25: Cambio variable categórica a binaria

5.6. Selección de hiperparámetros en los diferentes modelos

Esta es una de las partes que coincide fuertemente con el apartado de selección de features, con el que tratamos en el punto anterior. Los hiperparámetros son parámetros de entrada que entran al modelo de lenguaje, de manera que condicionan su aprendizaje. Algunos ejemplos son el número de iteraciones, el número de hijos que puede tener cada padre y más opciones que tratan de que el modelo tenga los mejores resultados posibles, además de reducir el tiempo humano necesario para que el modelo funcione.

Es por esto por lo que aparte de poder obtener estos valores de forma manual, e ir actualizando los parámetros en función de si tenemos o no buenos

resultados, se han desarrollado diferentes mecanismos que consiguen que este proceso sea más sencillo.

Por un lado, tenemos el GridSearch. En este método le damos unos valores a cada hiperparámetro deseado, de manera que una vez le ponemos todos los valores a cada uno, se va iterando, haciendo todas las combinaciones que esto permite, quedándose con la más acertada. Esto, tiene un alto coste computacional si queremos varios hiperparámetros, pero es bastante preciso al poder elegir los valores.

Por otra parte, tenemos el RandomSearch. Este es muy similar al punto anterior, sin embargo, se le da un rango determinado a cada variable, con intención de conseguir unos valores adecuados para cada parámetro. Normalmente la búsqueda es en función de otro parámetro que indica cuando parar, o hasta cuando debemos de seguir buscando. Es mejor que el caso anterior cuando algunos hiperparámetros son más importantes unos que otros, como vemos en la *Figura 26*.

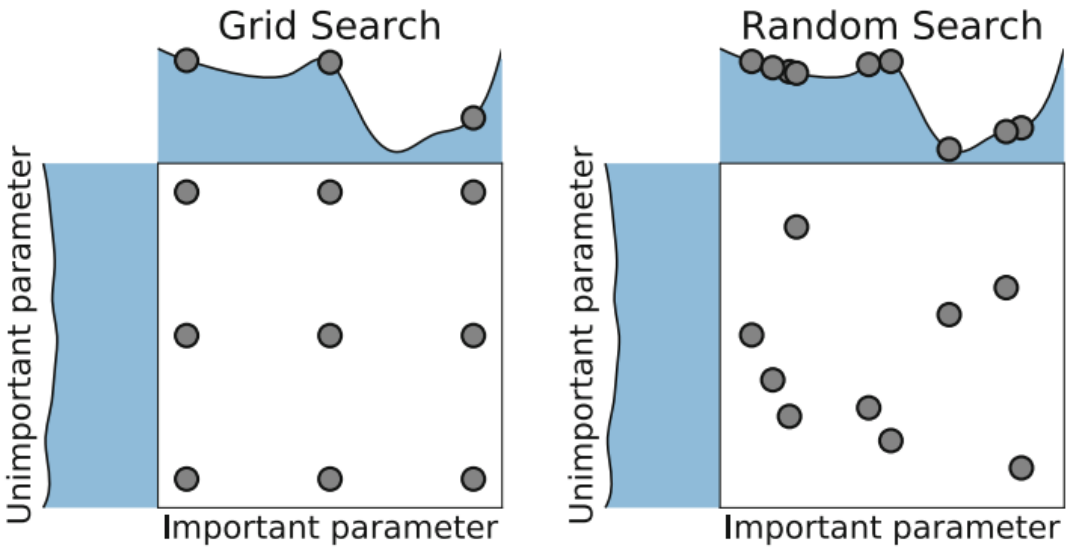

Figura 26: GridSearch vs RandomSearch

Podemos incluir también otro tipo importante, que es la optimización bayesiana. A partir de un modelo de probabilidad se van calculando os hiperparámetros óptimos (diciendo también cuanto queremos que explore y de qué manera) de

tal forma que el modelo se vaya actualizando en función de los hiperparámetros que se van calculando.

En XgBoost y Catboost se ha llevado un sistema de proceso similar. Se ha aplicado tanto GridSearch como RandomSearch y Optuna, en ambas situaciones para saber qué valor en los diferentes parámetros deben tomar para un rendimiento optimo del modelo. A pesar de esto y que diese unos datos bastante diferentes, los datos aportados por cualquiera de los dos métodos no fueron los hiperparámetros obtenidos no fueron la última configuración que se utilizó en el problema. Durante el desarrollo, el GridSearch destacaba en ser un método de selección de hiperparámetros que tarda demasiado en cuanto disponemos de una combinación de más de 6 argumentos (más de una hora en total en generar una respuesta, que no tiene por qué ser del todo consistente o la óptima). Sin embargo, sí que fueron muy útiles para acotar los valores adecuados para cada una de las variables.

Se diseñó una red neuronal sencilla compuesta por una capa oculta densa con 16 neuronas y activación ReLU (que deja pasar los valores positivos y convierte en cero los negativos), seguida de una capa de salida con una sola neurona, adecuada para tareas de regresión. Durante la fase de experimentación, se ajustaron distintos parámetros del modelo, como el número de épocas y el tamaño del lote (batch size), con el objetivo de optimizar el rendimiento y el tiempo de entrenamiento. Además, se evaluó el impacto de reducir el número de neuronas en la capa oculta a 2, 3 y 5, observándose diferencias mínimas en los resultados, con una leve mejora al emplear 5 neuronas. Sin embargo, esta mejora no fue suficientemente significativa como para justificar un aumento en la complejidad del modelo

6.ANÁLISIS DEL RENDIMIENTO

6.1. Resultados distintos modelos

En este apartado nos vamos a encargar de mostrar de qué manera se han logrado desenvolver los modelos, pudiendo observar que valores han tomado y cuanto se acerca cada predicción al valor real en cada caso.

Una vez hemos conseguido explotar los datos proporcionados y realizar algunas predicciones respecto a ellos, necesitamos medir su rendimiento. Para ello tenemos diferentes métricas en función de lo que queramos y el tipo de algoritmo que estamos utilizando, que explicaremos posteriormente.

La curva ROC (*Figura 27*) es una gráfica que sirve para evaluar el desempeño de un modelo binario, comparando cuántos positivos detecta correctamente (tasa de verdaderos positivos) frente a cuántos negativos clasifica erróneamente como positivos (tasa de falsos positivos). Cuanto más se acerque a una forma en "L" (subiendo recto y luego yendo a la derecha), mejor es el modelo. Sin embargo, si la curva es demasiado perfecta, podría indicar un sobreajuste (*overfitting*) a los datos de entrenamiento. Esta curva se construye asumiendo que el modelo devuelve una puntuación para cada predicción, y lo que se modifica es el umbral de decisión. Si no se tiene en cuenta esta puntuación y solo se genera una predicción final, no se obtiene una curva, sino un único punto con un TPR y un FPR específicos.

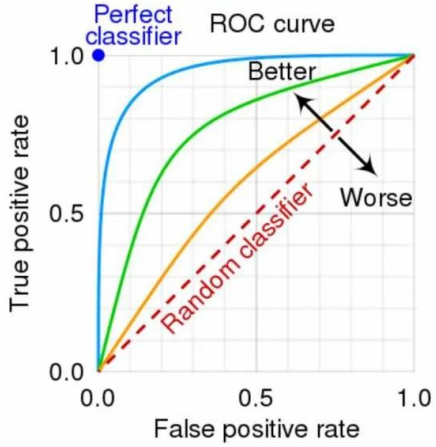

Figura 27: Curva de Roc

En el caso de Regressor tenemos las siguientes métricas para tener en cuenta:

El MAE (Mean Absolute Error) o error absoluto medio, mide el promedio de las diferencias absolutas entre las predicciones del modelo y los valores reales.

Por otro lado, el RMSE (Root Mean Squared Error) o raíz del error cuadrático medio, también mide la diferencia entre predicciones y valores reales, pero elevando las diferencias al cuadrado antes de promediarlas, y luego sacando la raíz cuadrada.

En este caso el primero de los errores, el MAE no penaliza con tanta fuerza como lo hace el segundo a valores muy grandes. Esto nos viene bien para tener unos resultados de forma equitativa con cada error, mientras que, por otro lado, si nos referimos al RMSE, podemos detectar errores grandes y eliminarlos al ser tan claramente marcados.

Una vez aclarados, se presentarán los distintos resultados que se han podido obtener en el uso de cada modelo.

XgBoost:

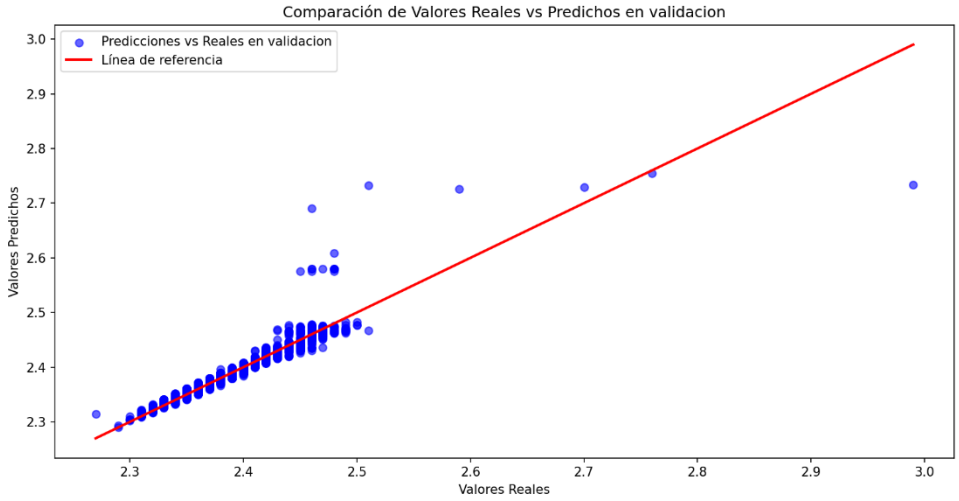

Figura 28: gráfica resultados XgBoost

En está gráfica podemos ver los valores predichos en comparación con el valor real de la variable objetivo. Lo más destacable y lo que más resalta son los 10 casos de valores que se salen de las métricas (pasan todos el mismo día en horas seguidas) que podríamos llegar a quitar al considerarlos outliers (el modelo no reacciona del todo mal a estos valores y no cambian demasiado las métricas).

Aquí podemos ver que el modelo tiene unos resultados bastante buenos, valorando el bajo error absoluto y la rápida ejecución del sistema.

CatBoost:

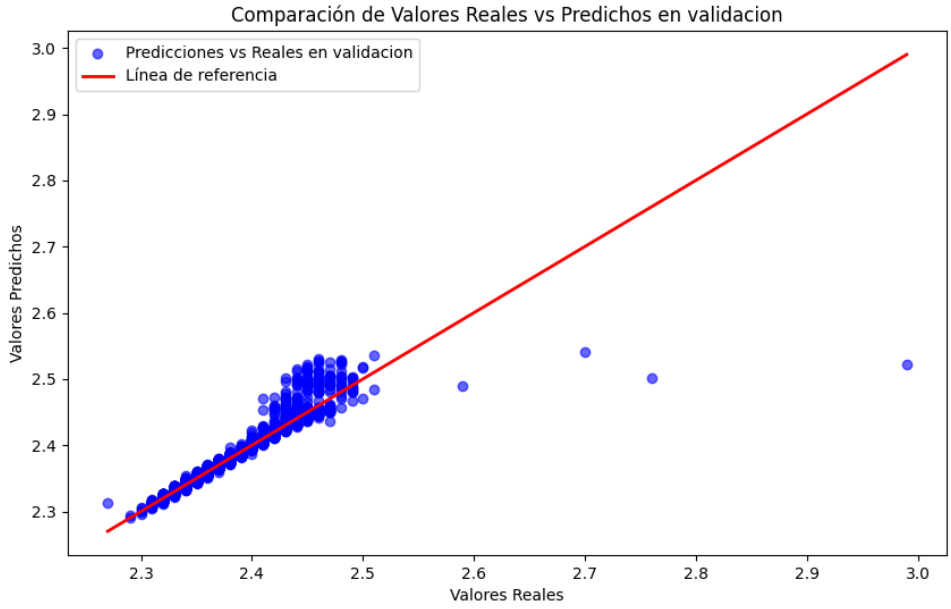

Figura 29: gráfica resultados CatBoost

En el caso de catboost, vemos unos valores bastante buenos también en la gráfica, pero se concentran algo más de valores no muy bien ajustados entorno al 2.45 que ocasionan que se distorsione el resultado.

Redes Neuronales:

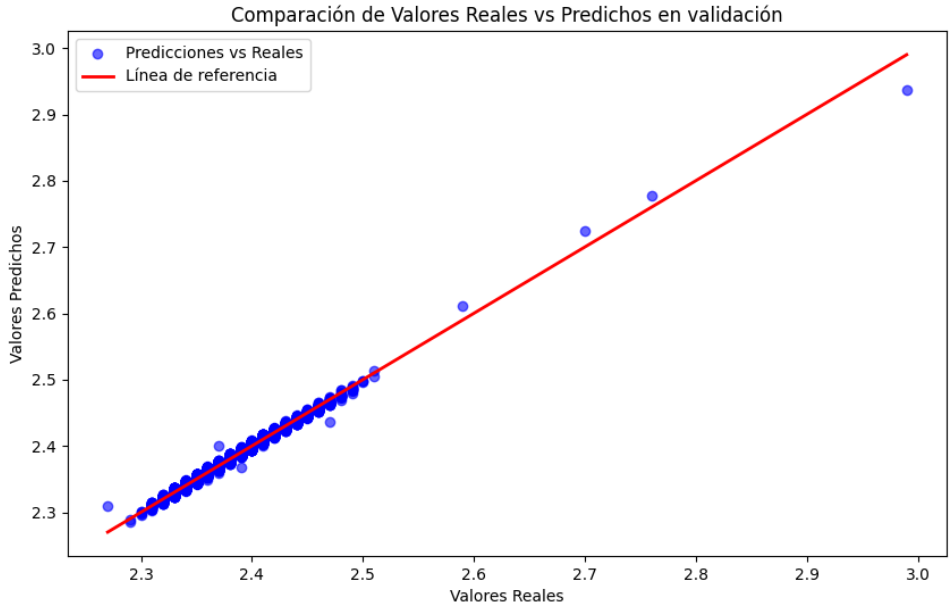

Figura 30: gráfica resultados Redes Neuronales

En este último caso, podemos ver cómo se adaptan las redes neuronales. En este caso se adaptan muy bien a todos los diferentes casos, a pesar de tener algún outliers, de manera que tiene unas predicciones bastante precisas.

Por último, muestro en esta tabla los diferentes valores que han obtenido en validación las distintas métricas en cada modelo:

Modelo	MAE	RMSE
XgBoost	0.00262852	0.00577217
CatBoost	0.00333565	0.00687148
Redes Neuronales	0.00268424	0.00324871

Tabla 2: Resultados de las métricas de cada modelo

6.2. Comparativa entre ellos

Como podemos observar en la *Tabla 2*: Resultados de las métricas de cada modelo, tenemos como claro mejor modelo, el de las redes neuronales por sus resultados. A pesar de esto XgBoost da unos resultados muy similares respecto a valores de error absoluto, en términos de errores grande y valores RMSE tiene muchos más fallos en comparación con las redes. En el caso de catboost, no es

superior a ninguna de las otras dos en este contexto, por ejemplo, habiendo solo una variable categórica y no pudiendo del todo desenvolverse de forma adecuada.

En este caso podemos ver una superioridad de las redes neuronales en cuanto a acercarse en valores más acertados y no tener fallos más grandes como en sus respectivos contrincantes (XgBoost y CatBoost).

7.CONCLUSIONES Y MEJORAS FUTURAS

7.1. Conclusiones sacadas

A pesar de lo que se creía en un primer momento, el modelo de XgBoost a pesar de tener muy buenos resultados, no acaba siendo el mejor. Esto se debe a que las redes neuronales se adaptan de mejor forma a los diferentes valores de las features en comparación del primero. Sin embargo, los dos siguen mejorando el rendimiento de catboost.

En conclusión, respecto a nuestro problema de datos actual, los mejores resultados nos los proporcionan las redes neuronales (a pesar de ser algo más lentas).

7.2. Pasos futuros por seguir

Una vez hemos conseguido un modelo con unos resultados con bastante buen rendimiento nos tocaría hacer un trabajo “inverso” a lo que se ha estado realizando. Para ello deberíamos fijarnos en los valores de las variables que un operador de la planta desaladora puede modificar de forma manual. Una vez veamos y comprendamos deberemos idear algún sistema que permita avisar al operario de los valores que debería cambiar para que el ratio de producción continue sin subidas.

Como última cosa a realizar podríamos tratar de mejorar el rendimiento de los modelos mediante algo más de refinamiento en el preprocesamiento y la selección de los distintos hiperparámetros.

BIBLIOGRAFÍA

[1] "TensorFlow." Accessed: May 14, 2025. [Online]. Available: https://www.tensorflow.org/?hl=es

[2] "CatBoost." Accessed: May 14, 2025. [Online]. Available: https://catboost.ai/docs/en/

[3] "XGBoost Documentation — xgboost 3.0.1 documentation." Accessed: May 13, 2025. [Online]. Available: https://xgboost.readthedocs.io/en/release_3.0.0/

[4] "scikit-learn: machine learning in Python — scikit-learn 1.6.1 documentation." Accessed: May 13, 2025. [Online]. Available: https://scikit-learn.org/stable/

[5] "Matplotlib — Visualization with Python." Accessed: May 13, 2025. [Online]. Available: https://matplotlib.org/

[6] "seaborn: statistical data visualization — seaborn 0.13.2 documentation." Accessed: May 13, 2025. [Online]. Available: https://seaborn.pydata.org/

[7] "pandas - Python Data Analysis Library." Accessed: May 13, 2025. [Online]. Available: https://pandas.pydata.org/

[8] "Sobreajuste - Wikipedia, la enciclopedia libre." Accessed: May 13, 2025. [Online]. Available: https://es.wikipedia.org/wiki/Sobreajuste

[9] S. Taylor, "Bayes' Theorem - Definition, Formula, and Example." Accessed: May 13, 2025. [Online]. Available: https://corporatefinanceinstitute.com/resources/data-science/bayes-theorem/

[10] N. Van Otten, "ROC And AUC Curves In ML Made Simple & How To Tutorial." Accessed: May 13, 2025. [Online]. Available: https://spotintelligence.com/2024/06/17/roc-auc-curve-in-machine-learning/

[11] N. Van Otten, "Confusion Matrix: A Beginners Guide & How To Tutorial." Accessed: May 13, 2025. [Online]. Available: https://spotintelligence.com/2024/09/06/confusion-matrix-a-beginners-guide-how-to-tutorial-in-python/

[12] Keerthana, "DATA PREPROCESSING TECHNIQUES. Data preprocessing is a Data Mining... | by Keerthana | AlmaBetter | Medium." Accessed: May 13, 2025. [Online]. Available: https://medium.com/almabetter/data-preprocessing-techniques-6ea145684812

[13] Codeacademy Team, “Introduction to Regression Analysis | Codecademy.” Accessed: May 13, 2025. [Online]. Available: https://www.codecademy.com/article/introduction-regression-analysis

[14] J. Holdsworth, “What is Data Mining? | IBM.” Accessed: May 13, 2025. [Online]. Available: https://www.ibm.com/think/topics/data-mining

[15] “¿Qué es y cómo funciona la ósmosis inversa? - Carbotecnia.” Accessed: May 13, 2025. [Online]. Available: https://www.carbotecnia.info/aprendizaje/osmosis-inversa/que-es-la-osmosis-inversa-purificador/

[16] R. Routledge, “Bayes’s theorem | Definition & Example | Britannica.” Accessed: May 13, 2025. [Online]. Available: https://www.britannica.com/topic/Bayess-theorem

[17] The Editors of Encyclopaedia Britannica, “Thomas Bayes | Bayesian Statistics, Probability Theory, Logic | Britannica.” Accessed: May 13, 2025. [Online]. Available: https://www.britannica.com/biography/Thomas-Bayes

[18] Z. H. Zhou, *Machine Learning*. Springer Nature, 2021. doi: 10.1007/978-981-15-1967-3.

[19] P. Jamal, M. Ali, R. H. Faraj, P. J. M. Ali, and R. H. Faraj, “1-6 Data Normalization and Standardization: A Technical Report,” 2014. [Online]. Available: https://docs.google.com/document/d/1x0A1nUz1WWtMCZb5oVzF0SVMY7a_58KQulqQVT8LaVA/edit#

[20] H. A. Ahmed, P. J. Muhammad Ali, A. K. Faeq, and S. M. Abdullah, “An Investigation on Disparity Responds of Machine Learning Algorithms to Data Normalization Method,” *ARO-THE SCIENTIFIC JOURNAL OF KOYA UNIVERSITY*, vol. 10, no. 2, pp. 29–37, Sep. 2022, doi: 10.14500/aro.10970.

[21] O. Rainio, J. Teuho, and R. Klén, “Evaluation metrics and statistical tests for machine learning,” *Sci Rep*, vol. 14, no. 1, Dec. 2024, doi: 10.1038/s41598-024-56706-x.

[22] F. Hutter, L. Kotthoff, and J. Vanschoren, “The Springer Series on Challenges in Machine Learning Automated Machine Learning Methods, Systems, Challenges.” [Online]. Available: http://www.springer.com/series/15602

[23] R. Wirth and J. Hipp, “CRISP-DM: Towards a Standard Process Model for Data Mining.”

[24] B. Venkatesh and J. Anuradha, “A review of Feature Selection and its methods,” *Cybernetics and Information Technologies*, vol. 19, no. 1, pp. 3–26, 2019, doi: 10.2478/CAIT-2019-0001.

[25] A. A. Abro, A. Ayub Khan, M. Sajjad, H. Talpur, I. Kayijuka, and E. Yaşar, “Machine Learning Classifiers: A Brief Primer,” 2021.

[26] S. García, S. Ramírez-Gallego, J. Luengo, J. M. Benítez, and F. Herrera, “Big data preprocessing: methods and prospects,” *Big Data Anal*, vol. 1, no. 1, Dec. 2016, doi: 10.1186/s41044-016-0014-0.