COMILLAS

UNIVERSIDAD PONTIFICIA

ICAI

SOLAR-IA: DESARROLLO DE UNA HERRAMIENTA DE SOFTWARE ANALÍTICA DE EVENTOS EN ACTIVOS RENOVABLES

Trabajo de Fin de Máster

Escuela Técnica Superior de Ingeniería (ICAI)

Máster en Big Data, Tecnología y Analítica Avanzada

Universidad Pontificia Comillas

Curso 2024-2025

Autora: Lucía López Aranzana

Tutora: Ana Laguna Pradas

Madrid, junio 2025

Declaro, bajo mi responsabilidad, que el proyecto presentado con el título

SOLAR-IA: DESARROLLO DE UNA HERRAMIENTA DE SOFTWARE ANALÍTICA DE EVENTOS EN ACTIVOS RENOVABLES

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2024/25 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Lucía López Aranzana Fecha: 27/06/2025

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Ana Laguna Pradas Fecha: 27/06/2025

Vº Bº del Coordinador de Proyectos

Fdo.: Carlos Morrás Ruiz-Falcó Fecha: .../.../...

Resumen

Este Trabajo de Fin de Máster tiene como objetivo guiar a la empresa Gamesa Electric en el desarrollo de un modelo para la clasificación del comportamiento anómalo de los inversores solares. Debido a la creciente digitalización del sector eléctrico, la empresa desea iniciarse en la implementación del mantenimiento predictivo de sus equipos.

El trabajo comienza con un análisis del estado actual de la industria eléctrica, marcado principalmente por la transición energética y la mencionada digitalización. Asimismo, se estudian los distintos tipos de mantenimiento eléctrico y se plantean algunos casos de uso del aprendizaje automático en este campo.

Los datos utilizados están en formato CSV y recogen información de variables numéricas y registros de alarmas de diferentes inversores solares. Al ser la primera vez que se analizan en profundidad, se aplican técnicas de preprocesado y visualización con el objetivo de detectar características importantes.

Dado que la empresa no podía aportar datos etiquetados, se aplican técnicas de *clustering* para obtener grupos de inversores diferenciados por su comportamiento. Además, ante el elevado número de variables, se emplea *Principal Component Analysis* (PCA) para la reducción de la dimensionalidad. No fue hasta la mitad del proyecto cuando se consiguió etiquetar los datos según un fallo específico de los inversores: el problema de condensación. Gracias a ello, el problema de clasificación se simplifica considerablemente.

Se prueban distintos modelos basados en árboles para clasificar este fallo concreto: *Decision Tree* y *XGBoost*. El objetivo es implementar un modelo lo más interpretable posible, facilitando a la empresa familiarizarse con estas técnicas de *machine learning* y comprender el funcionamiento interno del modelo. De los dos, el que presentó mejores métricas fue el basado en *XGBoost* junto con PCA. Sin embargo, aunque aceptables, las métricas muestran margen de mejora. Por ello, el trabajo finaliza con futuras líneas de mejora que la empresa puede seguir para conseguir un modelo de clasificación más robusto y apto para su implementación en entornos de desarrollo.

En definitiva, este trabajo supone un punto de partida para Gamesa Electric, sobre el cual continuar la implementación de un sistema de mantenimiento predictivo de sus equipos.

Palabras clave: mantenimiento predictivo, inversores solares, clasificación, *clustering*, reducción de dimensionalidad.

Abstract

This Master's Thesis aims to guide the company Gamesa Electric in developing a model for the classification of anomalous behavior in solar inverters. Due to the increasing digitalization of the electrical sector, the company seeks to begin implementing predictive maintenance for its equipment.

The work starts with an analysis of the current state of the electrical industry, mainly characterized by the energy transition and the aforementioned digitalization. Additionally, different types of electrical maintenance are studied and several use cases of machine learning in this field are proposed.

The data used are in CSV format and include numerical variables and alarm logs from various solar inverters. Since this is the first time these data are analyzed in depth, preprocessing and visualization techniques are applied to detect important features.

Given that the company could not provide labeled data, clustering techniques are applied to obtain groups of inverters differentiated by their behavior. Moreover, due to the high number of variables, Principal Component Analysis (PCA) is employed for dimensionality reduction. It was not until halfway through the project that the data were labeled according to a specific inverter fault: the condensation problem. Thanks to this, the classification problem is considerably simplified.

Different tree-based models are tested to classify this specific fault: Decision Tree and XGBoost. The objective is to implement the most interpretable model possible, making it easier for the company to become familiar with these machine learning techniques and understand the internal workings of the model. Of the two, the one with the best performance metrics was the XGBoost-based model combined with PCA. However, although acceptable, the metrics show room for improvement. Therefore, the work concludes with future improvement lines that the company can follow to achieve a more robust classification model suitable for deployment in development environments.

In short, this work serves as a starting point for Gamesa Electric to continue implementing a predictive maintenance system for its equipment.

Keywords: predictive maintenance, solar inverters, classification, *clustering*, dimensionality reduction.

Agradecimientos

Gracias a mi familia y en especial a mis padres por el apoyo recibido desde siempre. En los momentos más complicados y con mayor carga de trabajo, han sabido tranquilizarme y darme ánimos para seguir.

También me gustaría agradecer a Miguel Ángel, Ana y Vicente por su ayuda y consejos durante el desarrollo del trabajo. Han sido unos meses llenos de aprendizaje que estoy segura que me ayudarán en mi futuro profesional.

Índice general

1. Introducción **1**

1.1. Contexto y problemática de la empresa 2

1.2. Objetivo del trabajo 3

2. Estado de la cuestión **5**

2.1. Situación actual de la industria eléctrica 5

2.1.1. Transición energética 5

2.1.2. Digitalización de la industria eléctrica 9

2.2. Mantenimiento eléctrico 13

2.2.1. Tipos de mantenimiento eléctrico 14

2.2.2. Principales técnicas de mantenimiento predictivo 16

2.2.3. Algunas aplicaciones de inteligencia artificial en mantenimiento eléctrico solar 21

3. Definición del problema **25**

3.1. Objetivo 25

3.2. ¿Qué es un inversor solar? 26

3.2.1. El problema de condensación 27

3.3. Descripción de los datos 29
3.3.1. Descripción de las variables y alarmas a estudiar 29
3.3.2. Sistema de recolección 31
3.4. Etapas del trabajo 31
3.5. Recursos empleados 33

4. Análisis de las variables temporales y alarmas **35**

4.1. Preprocesado de los datos 35
4.2. Técnicas de análisis exploratorio 36
4.2.1. Distribución de las variables 36
4.2.2. Correlaciones lineales 38
4.3. Evolución temporal de las variables 39
4.4. Frecuencia de las alarmas 41

5. Modelo de *clustering* **43**

5.1. *Principal Component Analysis* 44
5.1.1. Fundamentos teóricos 44
5.1.2. Implementación 46
5.2. *K-Means* 49
5.2.1. Fundamentos teóricos 49
5.2.2. Implementación 51
5.3. *Fuzzy C-Means* 56
5.3.1. Fundamentos teóricos 56
5.3.2. Implementación 58

5.4. *Gaussian Mixture Models* 63

5.4.1. Fundamentos teóricos 64

5.4.2. Implementación 66

5.5. Resultados y conclusiones 72

6. Modelo de clasificación **75**

6.1. Etiquetado manual 75

6.2. Análisis exploratorio completo 76

6.3. Fundamentos teóricos de los modelos usados 79

6.3.1. *Decision Tree* 79

6.3.2. *XGBoost* 81

6.4. Modelos implementados 83

6.4.1. Modelo basado en DT considerando las variables T_SKIIP_MIN, TPOT_CAB_MAX, T_LIQ_MIN, DELTA_TEMP_AMB . . . 83

6.4.2. Modelo basado en DT considerando las dos primeras componetes principales de PCA 85

6.4.3. Modelo basado en *XGBoost* considerando las variables T_SKIIP_MIN, TPOT_CAB_MAX, T_LIQ_MIN, DELTA_TEMP_AMB . . . 87

6.4.4. Modelo basado en *XGBoost* considerando las dos primeras componentes principales de PCA 90

6.5. Resultados y comparativas de los modelos 90

6.5.1. Elección del mejor modelo 96

7. Conclusión del trabajo **99**

7.1. Conclusiones 99

7.2. Líneas de mejora y trabajo futuro 101

7.3. Conclusión final 102

Bibliografía **105**

Índice de figuras

2.1. Evolución del porcentaje de la demanda cubierta por la energía solar y eólica desde enero de 2017 hasta junio de 2024. La imagen fue extraida de [7]. 6

2.2. Esquema de la metodología SGAM aplicada a la digitalización del sector eléctrico, fuente [16]. 11

2.3. Esquema de las 4 etapas principales del mantenimiento predictivo, fuente [23]. 15

3.1. Imagen de un inversor solar en uno de los parques de la empresa. Este en concreto se encuentra situado en Myrtle, Estados Unidos. . 27

3.2. Esquema de funcionamiento de un inversor en el que se muestran los flujos del sistema de refrigeración y caldeo mediante líquido glicol. 27

3.3. Esquema con algunas componentes internas del inversor. Los cículos azules representan las dos cabinas de bloques de potencia mientras que los cículos rojos corresponden con los skiips. 28

4.1. Histogramas que representan la distribución de las variables temporales de los archivos CSV recibidos en la primera ronda de datos. No se graficaron la distribución de las variables `time`, `utc` y `sample` pues no aportaban información relevante. 37

4.2. *Boxplots* de las variables temporales de los archivos CSV recibidos en la primera ronda de datos. No se graficaron los correspondientes a las variables `time`, `utc` y `sample` pues no aportaban información relevante. 38

4.3. *Heatmap* que representa la matriz de correlaciones lineales de las variables temporales de los archivos CSV recibidos en la primera ronda de datos. Para su cálculo se utilizó el coeficiente de correlación de Pearson. Las celdas en tonos grises de tamaño menor se deben a la desviación estándar nula de la variable del denominador en la fórmula de Pearson. 39

4.4. Evaluación temporal de la variable `PREI` para los inversores 33 (naranja) y 34 (azul). Se observa cómo desde las 21h del día 11-02-25 hasta las 03h del 12-02-25 el inversor 34 experimenta un descenso de presión en la entrada de la bomba del inversor. Las líneas discontinuas marcan la aparición de alguna alarma. 40

4.5. Evaluación temporal de la variable `PREO` para los inversores 33 (naranja) y 34 (azul). Se observa cómo desde las 21h del día 11-02-25 hasta las 00h del 12-02-25 el inversor 34 experimenta un descenso de presión en la salida de la bomba del inversor. Las líneas discontinuas marcan la aparición de alguna alarma. 40

4.6. Evaluación temporal de la variable `PPC_PMG` (setpoint de potencia activa desde el SCADA) para los inversores 2 (morado), 4 (rojo), 11 (verde), 33 (naranja) y 34 (azul). Se observa cómo el sistema de potencia del inversor 11 está intentando aumentar el valor de esta variable sin éxito sobre las 7h del 13-02-25. 41

4.7. Evaluación temporal de la variable `IT1_RMS` (corriente de línea-1) para los inversores 2 (morado), 4 (rojo), 11 (verde), 33 (naranja) y 34 (azul). Se observa que el inversor 11 presenta, a partir de las 7h del día 13-02-25, un valor nulo para esta variable mientras que el resto de inversores tienen valores positivos. 41

4.8. Evaluación temporal de la variable `TCOOLINGI` (temperatura de entrada del líquido) para los inversores 2 (morado), 4 (rojo), 11 (verde), 33 (naranja) y 34 (azul). Se observa que el inversor 11 presenta, a partir de las 00h del día 13-02-25, un descenso que intenta solucionar el equipo sin éxito (pequeños picos a partir de valores de temperatura inferiores a 7^{o}C). Estos picos se observan también en el inversor 4 pero este si fue capaz de aumentar la temperatura. . . 41

5.1. Gráfico de barras y líneas que representa la varianza explicada por cada componente principal (barras) y la varianza explicada acumulada (línea). 47

5.2. *Heatmap* que representa las correlaciones de las 10 variables más contribuyentes en la primera componente principal. 48

5.3. *Heatmap* que representa las correlaciones de las 10 variables más contribuyentes en la segunda componente principal. 48

5.4. *Heatmap* que representa las correlaciones de las 10 variables más contribuyentes en la tercera componente principal. 48

5.5. Gráficos con los dos *clusters* obtenidos con el algoritmo de *K-Means*. Estos grupos se han proyectado en el espacio de las tres primeras componentes principales del algoritmo de PCA. En la imagen se muestran los centroides de ambos *clusters*. 52

5.6. Gráfico de barras con el valor de los dos centroides obtenidos con *K-Means* para la primera componente principal. 54

5.7. Gráfico de barras con el valor de los dos centroides obtenidos con *K-Means* para la segunda componente principal. 54

5.8. Gráfico de barras con el valor de los dos centroides obtenidos con *K-Means* para la tercera componente principal. 54

5.9. Distribución de los dos *clusters* obtenidos mediante *K-Means* usando los datos del inversor 34 el día 13-02-25. Este inversor presenta un comportamiento normal. 55

5.10. Distribución de los dos *clusters* obtenidos mediante *K-Means* usando los datos del inversor 11 el día 13-02-25. En ese día, se registró una explosión tras congelar del equipo. 55

5.11. Gráficos de barras en los que puede verse para cada *cluster* del algoritmo FCM, el número de puntos que hay para cada rango de grados de pertenencia. Se observa que para ambos *clusters* la gran cantidad de puntos tienen elevados grados de pertenencia. 60

5.12. Gráficos con los dos *clusters* obtenidos con el algoritmo de FCM. Estos grupos se han proyectado en el espacio de las tres primeras componentes principales del algoritmo de PCA. En la imagen se muestran los centroides de ambos *clusters.* 61

5.13. Gráfico de barras con el valor de los dos centroides obtenidos con FCM para la primera componente principal. 62

5.14. Gráfico de barras con el valor de los dos centroides obtenidos con FCM para la segunda componente principal. 62

5.15. Gráfico de barras con el valor de los dos centroides obtenidos con FCM para la tercera componente principal. 62

5.16. Distribución de los dos *clusters* obtenidos mediante FCM usando los datos del inversor 34 el día 13-02-25. Este inversor presenta un comportamiento normal. 63

5.17. Distribución de los dos *clusters* obtenidos mediante FCM usando los datos del inversor 11 el día 13-02-25. En ese día, se registró una explosión tras congelar del equipo. 63

5.18. Gráfico de barras con el índice BIC para cada modelo GMM entrenado con los distintos tipos de matriz de covarianza. Se observa que el tipo `full` es el que mejor equilibrio tiene entre ajuste y complejidad. 68

5.19. Gráficos con los dos *clusters* obtenidos con el algoritmo de GMM. Estos grupos se han proyectado en el espacio de las tres primeras componentes principales del algoritmo de PCA. En la imagen se muestran los centroides de ambos *clusters.* 69

5.20. Gráfico de barras con el valor de los dos centroides obtenidos con GMM para la primera componente principal. 70

5.21. Gráfico de barras con el valor de los dos centroides obtenidos con GMM para la segunda componente principal. 70

5.22. Gráfico de barras con el valor de los dos centroides obtenidos con GMM para la tercera componente principal. 70

5.23. Distribución de las log-densidades de probabilidad del modelo GMM para 10.000 muestras de los inversores escogidas de forma aleatoria. Se observan elevadas log-densidades indicando un buen ajuste a los datos del modelo. 71

5.24. Distribución de los dos *clusters* obtenidos mediante GMM usando los datos del inversor 34 el día 13-02-25. Este inversor presenta un comportamiento normal. 72

5.25. Distribución de los dos *clusters* obtenidos mediante GMM usando los datos del inversor 11 el día 13-02-25. En ese día, se registró una explosión tras congelar del equipo. 72

5.26. Distribución de los dos *clusters* obtenidos mediante GMM usando las muestras del inversor C07 del parque Myrtle etiquetado sin fallo de condensación. 74

5.27. Distribución de los dos *clusters* obtenidos mediante GMM usando las muestras del inversor B43 del parque solar Myrtle etiquetado con fallo de condensación. 74

6.1. Histogramas que representan la distribución de 43 variables temporales de los archivos CSV de los inversores del parque Myrtle y el situado en Cádiz. No se graficaron la distribución de las variables `time`, `utc` y `sample` pues no aportaban información relevante. . . . 77

6.2. *Boxplots* de las variables temporales de los archivos CSV recibidos en la primera ronda de datos. No se graficaron los correspondientes a las variables `time`, `utc` y `sample` pues no aportaban información relevante. 78

6.3. *Heatmap* que representa la matriz de correlaciones lineales de las variables temporales de los inversores del parque Myrtle. Para su cálculo se utilizó el coeficiente de correlación de Pearson. Las celdas en tonos grises de tamaño menor se deben a la desviación estándar nula de la variable del denominador en la fórmula de Pearson. . . . 79

6.4. Gráfico en el que se muestra el valor de la métrica *recall* para cada valor de `DT_min_impurity_decrease` que incluye todos los valores desde 0 hasta 0,05 a paso 0,005. El valor máximo de esta métrica para el modelo de DT con 4 variables se obtuvo con `DT_min_impurity_decrease = 0,01`. 84

6.5. Gráfico de barras en el que se muestra la importancia de cada variable en el modelo DT entrenado con 4 variables, utilizando para su cálculo el descenso de la impureza medido con el índice Gini. . . 85

6.6. Gráfico que muestra un esquema del árbol de decisión obtenido en el modelo DT entrenado con 4 variables. En cada nodo se indica el corte realizado, la impureza, número de muestras y distribución de clases. 85

6.7. Gráfico en el que se muestra el valor de la métrica *recall* para cada valor de `DT_min_impurity_decrease` que incluye todos los valores desde 0 hasta 0,05 a paso 0,005. El valor máximo de esta métrica para el modelo de DT con las dos primeras componentes principales se obtuvo con `DT_min_impurity_decrease` = 0,005. 86

6.8. Gráfico de barras en el que se muestra la importancia de cada variable en el modelo DT entrenado con las dos primeras componentes principales, utilizando para su cálculo el descenso de la impureza medido con el índice Gini. 87

6.9. Gráfico que muestra un esquema del árbol de decisión obtenido en el modelo DT entrenado con las dos primeras componentes principales. En cada nodo se indica el corte realizado, la impureza, número de muestras y distribución de clases. 87

6.10. Gráfico de barras en el que se muestra la importancia de cada variable en el modelo *XGBoost* entrenado con 4 variables, utilizando para su cálculo el descenso de la impureza medido con el índice Gini. 89

6.11. Gráfico en el que se muestra el número de nodos en cada árbol del ensamblado del modelo *XGBoost* entrenado con 4 variables. Además se marca en línea discontinua la media de nodos por árbol. 89

6.12. Gráfico de barras en el que se muestra la importancia de cada variable en el modelo *XGBoost* entrenado con las dos primeras componentes principales, utilizando para su cálculo el descenso de la impureza medido con el índice Gini. 90

6.13. Gráfico en el que se muestra el número de nodos en cada árbol del ensamblado del modelo *XGBoost* entrenado con las dos primeras componentes principales. Además se marca en línea discontinua la media de nodos por árbol. 90

6.14. Gráficas en las que se muestra la curva ROC, *calibration plot*, distribución de probabilidades y evolución *accuracy* según umbral de decisión para la clase positiva 1 en el modelo de DT entrenado con 4 variables. 93

6.15. Gráficas en las que se muestra la curva ROC, *calibration plot*, distribución de probabilidades y evolución *accuracy* según umbral de decisión para la clase positiva 1 en el modelo de DT entrenado con las dos primeras componentes principales. 94

6.16. Gráficas en las que se muestra la curva ROC, *calibration plot*, distribución de probabilidades y evolución *accuracy* según umbral de decisión para la clase positiva 1 en el modelo de *XGBoost* entrenado con 4 variables. 95

6.17. Gráficas en las que se muestra la curva ROC, *calibration plot*, distribución de probabilidades y evolución *accuracy* según umbral de decisión para la clase positiva 1 en el modelo de *XGBoost* entrenado con las dos primeras componentes principales. 95

Índice de cuadros

4.1. Top 3 alarmas por inversor. 42

4.2. Top 3 alarmas por día. 42

6.1. Resultados de métricas para los distintos modelos de clasificación. . 92

Capítulo 1

Introducción

En un mundo cada vez más digitalizado, son muchas las empresas que apuestan por las nuevas tecnologías. Es habitual escuchar en los medios de comunicación cómo las compañías se suman progresivamente a la implementación de técnicas de *machine learning* (ML) y de inteligencia artificial (IA), entre otras [1].

Uno de los sectores que más está apostando por la digitalización es el energético. Su compromiso con la transición hacia fuentes de energía renovables y el uso de maquinaria cada vez más compleja hace necesario plantearse nuevas formas de abordar los retos tecnológicos. Asimismo, gracias a la integración de tecnologías IoT y a la creciente monitorización de sus equipos, la recopilación de datos en este sector ha aumentado de forma exponencial.

Este incremento en la recolección de métricas y datos ha transformado las estrategias de mantenimiento de las infraestructuras eléctricas, tradicionalmente clasificadas en tres tipos: correctivo, preventivo y predictivo. Aunque antiguamente era habitual aplicar un mantenimiento de tipo preventivo, hoy en día se observa un cambio de tendencia hacia el mantenimiento predictivo, basado fundamentalmente en los datos. En este contexto, cobran especial relevancia las técnicas de aprendizaje automático y de *data mining* actuales.

Empresas punteras del sector ya están apostando por este nuevo enfoque de mantenimiento [2], [3]. De hecho, en 2022 se celebró el I Foro IndesIA, en el que se analizó el impacto que la IA tendrá en la industria española. En dicho foro se estimó que la inteligencia artificial generará un impacto en el PIB de entorno a 16.500 millones de euros para el año 2025 [4].

En este trabajo se presenta un desafío real proporcionado por la empresa Gamesa Electric, cuyo objetivo principal es desarrollar un modelo de clasificación del comportamiento —anómalo o normal— de los inversores solares. No obstante, como se detallará más adelante, el objetivo se reorientó hacia la detección específica del problema de condensación en dichos inversores.

En primer lugar, se contextualizará la situación actual de la industria eléctrica, destacando el papel de la transición energética y la digitalización. Asimismo, se describirán los distintos tipos de mantenimiento eléctrico y se presentarán algunos casos de uso de la IA en este ámbito.

A continuación, se definirá en detalle el problema abordado en este trabajo. Se presentarán los datos empleados, se explicarán los componentes principales de los inversores y se detallarán las distintas fases seguidas durante el proyecto: desde la recolección y el preprocesado de los datos, hasta el etiquetado manual y la comprensión del problema, pasando por el desarrollo del modelo.

Adicionalmente, se estudiarán en profundidad las técnicas de *machine learning* implementadas, mostrando los fundamentos matemáticos y estadísticos subyacentes. Posteriormente, se explicará en detalle la elección y aplicación de estas técnicas en el presente caso de uso.

Por último, se expondrán las conclusiones obtenidas y las propuestas de líneas futuras para avanzar hacia un modelo robusto de detección del problema de condensación.

1.1. Contexto y problemática de la empresa

Gamesa Electric es una empresa del sector eléctrico centrada en el diseño y la fabricación de componentes eléctricos, especialmente relacionados con placas fotovoltaicas, componentes hidráulicos y turbinas eólicas, entre otros. Su compromiso con la sostenibilidad la convierte en una compañía que apuesta firmemente por la tecnología y la innovación.

La empresa cuenta con distintas sedes, y en este trabajo se aborda un problema planteado por su planta especializada en paneles solares y convertidores eólicos, situada en Madrid.

Entre los diferentes componentes solares destacan los inversores solares, piezas fun-

damentales para el funcionamiento de los paneles. Estos dispositivos se encargan de convertir la corriente continua generada por los paneles solares en corriente alterna, que es la que se utiliza en los hogares y en la red eléctrica. En la actualidad, la mayoría de los inversores cuentan con mecanismos internos de *logs* que permiten a los técnicos tener un cierto grado de control y monitorización de estos dispositivos. A estos registros se suma la definición y el diseño de un sistema de alarmas que alerta sobre valores anómalos en distintas variables, como la temperatura ambiente o el voltaje de corriente continua (DC).

Hasta ahora, la empresa había planteado el mantenimiento de estos dispositivos mediante revisiones periódicas y medidas de control manuales. Sin embargo, debido al gran volumen de datos registrado por los *logs*, ha decidido apostar por un enfoque más avanzado de mantenimiento predictivo basado en el análisis de datos.

Entre los diversos problemas que pueden afectar a los inversores, uno de los más habituales y que más preocupa a la empresa es el fallo por condensación. Este problema es consecuencia de cambios drásticos de temperatura, los cuales generan humedad en el interior del inversor, dando lugar a fenómenos de corrosión en los circuitos eléctricos o incluso a cortocircuitos internos.

1.2. Objetivo del trabajo

El principal objetivo de este trabajo es guiar a la empresa en un caso de uso que sirva como punto de partida para la transición hacia un sistema de mantenimiento predictivo centrado en los datos. En concreto, se pretende desarrollar un modelo de clasificación capaz de identificar comportamientos de fallo en inversores solares.

Dado que hasta ahora la empresa no había trabajado en esta línea, el proyecto abarca tanto el preprocesado de los datos como el desarrollo e implementación de distintas técnicas de *machine learning*.

En primer lugar, será necesario familiarizarse con los datos y analizar la evolución temporal de las variables proporcionadas. Posteriormente, debido a la ausencia inicial de etiquetas, se emplearán técnicas de aprendizaje no supervisado para detectar patrones y posibles comportamientos anómalos.

A partir de la identificación de grupos diferenciados, se podrá comprobar la viabilidad de clasificar el comportamiento de los inversores en función de las variables disponibles. Gracias a ello, el objetivo del trabajo podrá concretarse y centrarse

en la clasificación de un tipo de fallo específico: el problema de condensación en los inversores.

Para alcanzar este objetivo se experimentará con distintos modelos basados en árboles y distintas variables en el entrenamiento, con el fin de identificar el modelo que mejor se ajuste a los datos.

Capítulo 2

Estado de la cuestión

Este capítulo, comenzará con un estudio sobre la situación actual de la industria eléctrica poniendo especial atención al aumento en el uso de energía solar así como la digitalización que está experimentando este sector.

Seguidamente, se explicarán los distintos tipos de mantenimiento eléctrico y se mostrarán las ventajas que aporta el uso del mantenimiento predictivo frente a los demás. En relación con ello, se estudiarán algunos artículos científicos en los que se han aplicado técnicas de aprendizaje automático para tareas de mantenimiento eléctrico.

2.1. Situación actual de la industria eléctrica

El sector eléctrico está viviendo una gran transformación en los últimos años. La mayor concienciación sobre el cambio climático junto con la revolución digital que está experimentando la sociedad son unos de los factores que más están impulsando esta transformación.

2.1.1. Transición energética

La transición energética es uno de los principales objetivos a nivel mundial. Prueba de ello, es la conocida Agenda 2030 cuyo objetivo número 7 es "Energía asequible y no contaminante". Desde la implantación de estos 17 objetivos de desarrollo

sostenible, se ha visto un aumento en la tasa mundial de acceso a la energía eléctrica pasando de un 87 % en 2015 a un 91 % en 2021 [5]. Aunque es el momento en el que más personas están teniendo acceso a la electricidad en el mundo, sigue siendo necesario impulsar nuevas medidas para conseguir que, en 2030, todas las personas tengan a su disposición electricidad. Asimismo, se tiene como objetivo que esta energía sea renovable y lo más eficiente posible.

Así, la descarbonización está jugando actualmente un papel muy importante en el sector eléctrico. La Unión Europea quiere conseguir para 2050, que las emisiones de gases de efecto invernadero se reduzcan en un 80 % con respecto a 1990 (un 40 % en 2030 y un 60 % para 2040) [6]. Por ello, la energías renovables están experimentando un fuerte crecimiento. Según un informe del Banco Central Europeo [7], las energías eólica y solar pasaron de cubrir un 26 % de la demanda eléctrica total en 2019 al 44 % en junio de 2024.

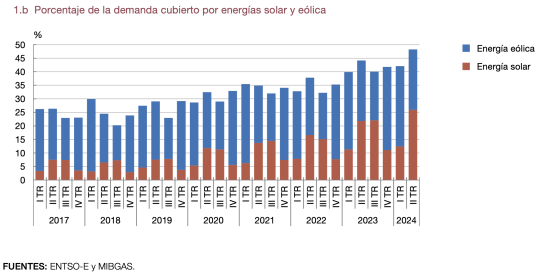

Figura 2.1: Evolución del porcentaje de la demanda cubierta por la energía solar y eólica desde enero de 2017 hasta junio de 2024. La imagen fue extraida de [7].

Son varios los factores que han potenciado este auge de las energías renovables. A la ya mencionada concienciación medioambiental, destacamos los siguientes:

1. Disminución en los costes de implementación de las energías renovables: gracias a una mayor apuesta por la innovación, el precio de las renovables compite cada vez más con los de los combustibles fósiles tradicionales. Por ejemplo, entre 2010 y 2020 el precio de los paneles solares fotovoltaicos disminuyó un 15 % por año [8].

2. Mejoras en los sistemas de almacenamiento de la energía: en los últimos años se está poniendo el foco en las baterías. El futuro de la descarbonización pasa por ellas. Debido al caracter estacional de las energías renovables, que dependen en gran medida de las condiciones meteorológicas, resulta indispensable disponer de baterías de gran capacidad para asegurar el suministro eléctrico en un sistema basado en las renovables.

En [9] se analiza la importancia del almacenamiento de la energía para la descarbonización del sector eléctrico. En este artículo se habla también de los sistemas de batería Li-ion (ion de Litio). Son uno de los sistemas de almacenamiento más competitivos de la actualidad pues permiten almacenar más energía en menos espacio. Además, su coste está en descenso, reduciéndose en un 85 % desde 2010 [10].

3. Búsqueda de la independencia energética: a raíz de la situación sociopolítica mundial, los países están tratando de implementar formas de energía que no dependan tanto de exportaciones de combustibles fósiles. Una de las situaciones que evidencian esta tendencia es la guerra de Ucrania. Países como Alemania o Francia han tomado medidas para no depender de países como Rusia y ser más independientes eléctricamente [11].

4. Aumento en ayudas e inversión en I+D: prueba de ello es el Plan Europeo de Recuperación Next Generation EU. Este consta de una serie de fondos que la Unión Europea proporciona a los países miembros para impulsar su economía. Uno de los requisitos que tienen que cumplir los países es que al menos un 37 % de los planes de recuperación económica involucren medidas de transición energética [12].

   Por otra parte, empresas como Iberdrola han duplicado su inversión en I+D, alcanzando los 4.000 millones de euros en 2030. Este presupuesto irá destinado en especial a soluciones tecnológicas limpias [13].

Son varias las energías renovables que se utilizan en la actualidad: eólica, hidráulica, biomasa, etc. . En este trabajo, se pondrá el foco en la energía solar.

Desarrollo y perspectivas de la energía solar

El Sol es una de las principales fuentes de energía disponibles en la Tierra. A través del aprovechamiento de su energía electromagnética, es posible generar electricidad o calor para distintos usos.

La energía solar ha experimentado un notable desarrollo gracias a su diseño simple, modular y fácilmente escalable, lo que permite procesos de aprendizaje acelerados y mejoras constantes. Además, ha recibido importantes contribuciones desde la industria tecnológica, como el desarrollo de células solares de perovskita, que prometen aumentar la eficiencia y reducir los costes. La disminución de los riesgos tecnológicos y la reducción del coste del financiamiento también han favorecido su

expansión. Por otro lado, los avances en reciclaje contribuyen a asegurar el suministro de materiales y a reducir los costes a lo largo del ciclo de vida de los sistemas. Finalmente, la diversidad y evolución de las tecnologías de almacenamiento, como las baterías, continúa reduciendo costes y reforzando el papel de la energía solar [8].

España es uno de los países con mayor potencial para la implantación de energía solar, debido a su elevada radiación solar y gran número de horas de sol anuales. Estas condiciones climáticas hacen que la instalación de sistemas fotovoltaicos sea especialmente eficiente y rentable en comparación con otros países europeos. Un estudio realizado por la Universidad de Sevilla analiza la viabilidad de un sistema eléctrico basado principalmente en energía solar en el territorio español [14].

Los principales objetivos del estudio fueron clasificar los municipios según su nivel de autosuficiencia energética mediante paneles solares instalados en tejados y evaluar la viabilidad y el coste en distintos escenarios, modificando la cobertura de paneles y la capacidad de almacenamiento. El análisis consistió en los siguientes pasos: determinación de perfiles de demanda horaria por municipio, estimación de la producción solar fotovoltaica, balance energético horario a nivel municipal considerando únicamente energía solar, y un segundo escenario integrando otras fuentes renovables disponibles.

El estudio destaca varias conclusiones relevantes:

- La autosuficiencia solar varía según la localización y la época del año. Por ejemplo, Sevilla podría ser autosuficiente durante todo el año, mientras que Asturias no alcanzaría dicha condición en ningún periodo. A nivel nacional, España no sería autosuficiente únicamente con energía solar en los meses de noviembre, diciembre y enero.
- Un sistema energético basado exclusivamente en energía solar requeriría una gran inversión en almacenamiento para compensar la variabilidad de la generación.
- Si un 10 % de la demanda eléctrica se cubriera con otras fuentes renovables, el 90 % restante podría satisfacerse mediante energía solar con almacenamiento estacional moderado.

Aunque el estudio se basa en datos de 2018, sus conclusiones siguen siendo una referencia sólida para impulsar la transición energética en España. Con el apoyo de otras fuentes renovables y mejoras en el almacenamiento, un sistema eléctrico

basado mayoritariamente en energía solar resulta técnicamente viable. Estudios similares han analizado escenarios equivalentes en otros países, como Estados Unidos [15].

A pesar del avance y el potencial de la energía solar, existen aún barreras relevantes para su expansión global, como se destaca en [8]:

1. Desigualdades financieras entre países: aunque las inversiones avanzan en las economías desarrolladas, los flujos financieros orientados al clima deben multiplicarse por entre 4 y 8 para permitir el despliegue global de esta tecnología.

2. Limitaciones en la cadena de suministro: la producción de paneles solares depende de minerales críticos como el litio, telurio, cobre o níquel, cuyas reservas son limitadas y su suministro está concentrado en unos pocos países, como China o la República Democrática del Congo.

En conclusión, la energía solar atraviesa un periodo de expansión impulsado principalmente por la disminución de sus costes asociados y una mayor concienciación ambiental. No obstante, aunque diversos estudios indican que un sistema eléctrico basado en esta fuente es posible, destacando el caso de países como España, será necesario continuar trabajando en cuestiones económicas y logísticas para garantizar una transición energética sostenible.

2.1.2. Digitalización de la industria eléctrica

El desarrollo tecnológico impulsa la descentralización del sector energético, permitiendo la integración de generación distribuida y el rol activo de los usuarios. Herramientas como IoT, redes inteligentes y plataformas digitales facilitan esta transformación del modelo tradicional [6].

En un artículo publicado por el instituto de Investigación Tecnológica se realiza un estudio sobre la digitalización en las redes eléctricas [16]. En él, se tratan diversos temas entre los que se encuentran las principales tecnologías utilizadas por la industria así como los costes y beneficios de la digitalización.

En este artículo, se recalca cómo el gran desarrollo tecnológico que está experimentando la sociedad y la mayor concienciación medioambiental están favoreciendo la digitalización en el sector. Prueba de ello es el cambio en el modelo del sector

eléctrico pasando de un enfoque vertical a uno más distribuido. En el modelo tradicional, se tiene un único generador eléctrico que distribuye la electricidad a los consumidores de forma que el flujo circula en la misma dirección mientras que, en el modelo distribuido, se puede generar electricidad en distintos puntos gracias a las energías renovables. En este último escenario, entra en escena un nuevo personaje: el prosumidor. Este término designa al cliente que, no solamente consume energía, si no que también es capaz de generarla bien por cuestiones económicas o por concienciación medioambiental. Por ejemplo, en España un prosumidor sería un particular que dispone de placas solares en su domicilio.

La aparición en escena de la figura del prosumidor y la generación distribuida hace que el sector tenga que pensar en nuevas operativas de negocio que involucren una mayor conectividad y monitoreo entre los sistemas. Así, en el artículo se destacan tres pilares fundamentales de la digitalización, los cuales están interconectados: los sensores y actuadores, la conectividad y el tratamiento de datos.

El elemento central sobre el que gira esta nueva etapa en la industria eléctrica es la red inteligente o *smart grid*. En el artículo del IIT, se define como la aplicación de las tecnologías de la información y comunicación (TIC) a la red eléctrica, de modo que se puedan detectar a distancia y en tiempo real cambios en la red, procesarlos y responder ante ellos. Es la responsable de la mayor eficiencia de la red y que los usuarios estén más involucrados en el consumo y producción eléctrica.

En España, los contadores inteligentes son la tecnología principal en este sentido. Gracias a ellos, se pueden recolectar grandes volúmenes de datos sobre el consumo de energía de los clientes. Esto resulta beneficioso tanto para los clientes, pues disponen de más información sobre sus facturas y consumos, como para las empresas pues pueden detectar patrones de consumo que pueden llevar a plantear nuevos casos de negocio. Otra tecnología ampliamente utilizada en el país es el telecontrol. Permite que los operarios de parques eléctricos interactúen de forma remota con los dispositivos que se encuentran a pie de campo reduciendo los tiempos de paradas de los equipos para el mantenimiento.

A raíz de esta mayor interoperabilidad y conectividad, la industria eléctrica está empezando a implementar una nueva metodología de trabajo, *Smart Grid Architecture Model (SGAM)*. Esta nueva metodología se basa en las relaciones entre tecnologías en dos ejes y cinco planos. El primer eje del SGAM representa los "Dominios" del sector eléctrico, desde la generación hasta el consumo de energía. El segundo eje clasifica las "zonas" que gestionan la información en diferentes niveles (campo, estación, operación, empresa y mercado), abarcando equipos y espacios físicos. Ambos ejes se aplican a cinco capas de interoperabilidad: negocio, fun-

ción, información y componentes. A continuación, se muestra una imagen sobre la implementación de esta metodología en el ámbito de las tecnologías digitales del sector eléctrico:

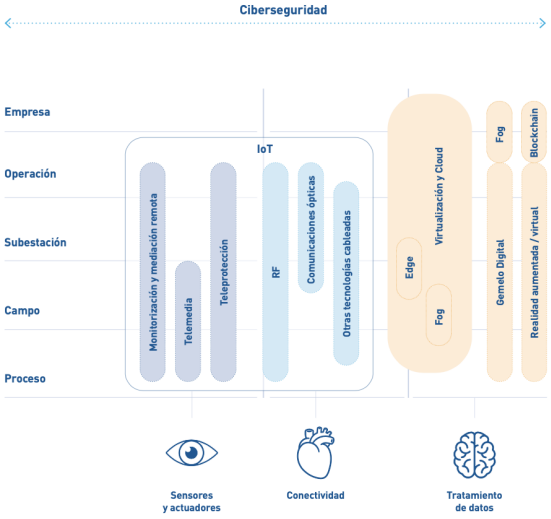

Figura 2.2: Esquema de la metodología SGAM aplicada a la digitalización del sector eléctrico, fuente [16].

Aunque es cierto que son muchas las tecnologías aplicables en la industria eléctrica, no todas presentan el mismo grado de madurez. Una tecnología puede estar lo suficientemente desarrollada y testeada como para ponerla en producción (madurez comercial) mientras que otras están en entorno de prueba (fase piloto) o incluso siguen probándose en el laboratorio (prueba de concepto).

Un ejemplo de tecnología con madurez comercial sería *Internet of Things* (IoT). Estos sistemas permiten la comunicación entre herramientas sin necesidad de intervención humana y tienen su origen en las redes de comunicación M2M (*Machine-to-Machine*). Algunas de sus aplicaciones incluyen la automatización y obtención de datos a través de sistemas SCADA (*Supervisory Control and Data Acquisition*). Así, el IoT constituye la base de la digitalización eléctrica al permitir tanto recolectar datos (que se usarán para aplicaciones de *machine learning* e inteligencia artificial como la que se presenta en este trabajo) como realizar intervenciones sobre equipos remotos.

En cuanto a tecnologías que se encuentran en una fase piloto o de prueba de concepto destacan:

- Virtualización y *cloud*: la virtualización permite crear una versión digital

(*software*) de un recurso físico o de un sistema que normalmente requiere *hardware* específico. El objetivo es que esta versión virtual funcione de forma equivalente al dispositivo físico original, sin afectar su rendimiento ni su operatividad. Como apoyo a la virtualización, se encuentra la computación en la nube, que no solo facilita el acceso a este *software*, sino también el almacenamiento y gestión de los datos generados por las *smart grids*.

- Analítica y *Big Data*: resulta indispensable tener herramientas que permitan un procesado y almacenamiento óptimo y eficiente de la gran cantidad de datos que se recogen de los sistemas. Para esta tarea, se encuentran herramientas como *Hadoop* que es un *framework* de *software* que incorpora sistemas de almacenamiento distribuido como HDFS y otras funcionalidades que permiten el tratamiento de *Big Data* [17].

  Gracias a la gestión de grandes volúmenes de datos, es posible realizar análisis avanzados mediante algoritmos matemáticos, lo que permite detectar patrones de comportamiento de los clientes, fallos en los sistemas, entre otros usos.

- Gemelo digital: es una réplica virtual de un objeto, proceso o sistema físico, conectada a su contraparte real mediante datos en tiempo real. Entre sus principales aplicaciones se encuentran el monitoreo en tiempo real, la simulación y pruebas (lo que permite reducir costos al evitar el uso de equipos físicos, que suelen ser costosos), y el mantenimiento predictivo [18], [19].

A parte de las tecnologías mencionadas se encuentran otras como realidad aumentada/virtual, *blockchain* o ciberseguridad. Asimismo, se pueden encontrar artículos como [20] en los que se explican algunos casos de uso en los suministros, la demanda o las redes.

Algunos retos de la digitalización eléctrica

Aunque las implementaciones de sistemas digitales en la industria eléctrica están cosechando resultados prometedores, todavía queda un largo camino por recorrer. Cabe destacar que el objetivo principal no es conseguir una red completamente inteligente si no conseguir una cada vez más inteligente. En la actualidad, existen una serie de factores que es necesario tener en cuenta para fomentar este desarrollo tecnológico en la industria eléctrica.

Por un lado, es imprescindible que las nuevas tecnologías sean compatibles con las infraestructuras existentes. Esto garantiza el correcto funcionamiento de la red, al

tiempo que permite avanzar hacia la innovación. Además, facilita que los clientes se adapten de forma gradual a los cambios en su interacción con el proveedor eléctrico.

Por otro lado, la red eléctrica abarca una gran extensión, suministrando tanto a zonas urbanas como rurales. Esta característica genera ciertas desigualdades, ya que en las zonas rurales la conectividad es mucho más limitada. Además, en muchos casos, la baja densidad de población hace que no resulte económicamente viable implementar nuevas tecnologías. Como ejemplo, el Tiempo de Interrupción Equivalente a la Potencia Instalada (TIEPI) en zonas rurales es hasta tres veces superior al registrado en zonas urbanas [16].

Otro reto importante a considerar es la ciberseguridad. Cada vez más, los datos son accesibles de forma remota, lo que exige la implementación de medidas que controlen el acceso y garanticen su protección. Las empresas no solo recopilan datos de los equipos y sistemas, sino también de sus clientes. Debido al carácter personal de estos últimos, es obligatorio que cumplan con el Reglamento General de Protección de Datos (RGPD) [21].

Por último, pero no menos importante, están las competencias digitales tanto de los empleados como de la ciudadanía en general. Es fundamental fomentar una mayor formación en competencias TIC para poder liderar la transformación digital que atraviesa el sector eléctrico.

2.2. Mantenimiento eléctrico

Uno de los temas centrales del trabajo es el mantenimiento en empresas eléctricas. Es por ello, que resulta importante definir en profundidad en qué consiste este área de la empresa así como las distintas técnicas que se emplean. Además, comparar los distintos tipos de mantenimiento será de utilidad para poder entender por qué cada vez más empresas apuestan por las técnicas predictivas.

El mantenimiento eléctrico es el conjunto de acciones y procedimientos destinados a conservar, reparar y mejorar los sistemas eléctricos para garantizar su correcto funcionamiento, seguridad y eficiencia. Su objetivo principal es evitar fallos, reducir el riesgo de accidentes y prolongar la vida útil de los equipos e instalaciones eléctricas.

2.2.1. Tipos de mantenimiento eléctrico

Son varias las técnicas de mantenimiento que emplean las empresas. No obstante, estas pueden englobarse en tres categorías [22], [23]:

1. Mantenimiento correctivo: Es aquel que se lleva a cabo cuando un equipo o sistema ha fallado o presenta un mal funcionamiento. En lugar de prevenir fallos o monitorear métricas se centra en reparar o restaurar los equipos tras la avería. Podemos distinguir a su vez dos tipos:

   - Programado: Se realiza cuando se detecta un fallo incipiente o se identifica que un equipo está funcionando mal, pero aún es posible seguir utilizándolo. Su reparación puede ser programada para que afecte lo mínimo posible a la producción global. Un ejemplo de ello es cuando una máquina comienza a hacer ruidos anormales y se programa su arreglo para el fin de semana.
   - No programado o de emergencia: La avería afecta a la producción y es necesaria una reparación inmediata. Por ejemplo, si una bomba hidráulica deja de funcionar en plena operación.

2. Mantenimiento preventivo: Se realiza de manera planificada y periódica para prevenir fallos y averías en los equipos antes de que ocurran. A diferencia del anterior, no es necesario que se produzca una avería para que se lleve a cabo. Su objetivo principal es prolongar la vida útil de los equipos, reducir el riesgo de fallos inesperados y minimizar los costos de reparaciones de emergencia. Se suele organizar en cuatro fases: inspección, detección, corrección y prevención. Además, puede realizarse según los siguientes aspectos:

   - Oportunidad: Aprovecha un momento en el que está parada la máquina.
   - Predictivo: Se detectan los momentos de mayor uso de la maquinaria y se realiza el mantenimiento en aquellos de menos uso.
   - Programado: Se realiza en periodos fijos (tiempo, kilometraje,...)

   Un ejemplo de uso es la limpieza y revisión de palas en aerogeneradores.

3. Mantenimiento predictivo: Se realiza un monitoreo continuo de los equipos para predecir fallos antes de que ocurran. Se basa en la recopilación y análisis de datos en tiempo real mediante sensores, inspecciones y herramientas tecnológicas.

   Podemos distinguir cuatro etapas en este proceso:

a) Recolección de datos y métricas a partir de sitemas IoT que permiten monitorear las máquinas

b) Transmisión en *real time* de estos datos a través de la red para hacerlos llegar al área de negocio de la empresa.

c) Utilización de métodos de inteligencia artificial y *machine learning* para obtener información estratégica y de valor para la empresa.

d) Implementación de medidas a partir de la información y conclusiones obtenidas.

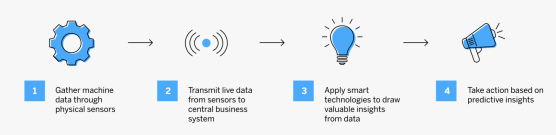

Figura 2.3: Esquema de las 4 etapas principales del mantenimiento predictivo, fuente [23].

Una de las situaciones en las que se aplica este tipo de mantenimiento es cuando se emiten *logs* de control durante el funcionamiento de las máquinas.

Cada vez son más las empresas punteras en el sector que están apostando por este último. A continuación, veremos algunas de las ventajas que ofrece con respecto al resto de técnicas.

Ventajas del mantenimiento predictivo

Entre las principales ventajas que ofrece el mantenimiento predictivo, se destacan las siguientes:

1. Monitorización de equipos en tiempo real: Se puede consultar en cualquier momento el estado actual de las máquinas y ver en directo sus necesidades. De esta forma, se evitan intervenciones innecesarias de los equipos.

2. Detección temprana de anomalías: Permite detectar fallos temprano en la máquina evitando llegar a anomalías críticas que afecten el funcionamiento del equipo. Esta es una gran ventaja frente al mantenimiento preventivo.

3. Alertas parametrizables: Se pueden establecer alarmas definidas anteriormente para avisar de valores anómalos en ciertas variables que hayan sido determinadas como importantes.

4. Liberación de personal para otras tareas: al automatizar los procesos de monitorización y vigilancia, el personal puede centrarse en tareas de organización y gestión.

2.2.2. Principales técnicas de mantenimiento predictivo

En esta sección se va a poner el foco en las técnicas de mantenimiento predictivo más habituales para la detección y diagnóstico de fallos en la industria eléctrica. En primer lugar, atendiendo al enfoque, recursos y técnicas empleadas se presenta la siguiente taxonomía [24]:

- *Model-based*: Este tipo de técnicas se centran en el desarrollo de sistemas basandos en gran medida en el conocimiento de expertos combinado con modelados matemáticos. Suelen ser modelos de tipo *white-boxes* pues son muy interpretables y predecibles.
- *Data-driven*: Utilizan *datasets* históricos junto con técnicas como ML para generar reglas de inferencia. En este caso, los modelos son de tipo *black-box* pues resulta más complicado conocer qué es lo que realiza el modelo internamente.

  Para diseñar un buen modelo en este sentido, resulta imprescindible seleccionar de forma cuidadosa el conjunto de entrenamiento y la técnica a emplear. Entre sus ventajas se encuentra que no necesita mucho conocimiento técnico.
- *Signal-based*: En ellos se parte de la suposición de que la señal medida refleja la condición de falla.
- Cuantitativos: Tratan de encontrar relaciones entre los *inputs* y *outputs* del sistema.
- Cualitativos: En este caso, las relaciones del sistema se expresan a partir de gráficos o reglas *if-then*.

Debido a que en este trabajo se presentará una estrategia de mantenimiento eléctrico basado en los datos, se profundizará en las técnicas principales de tipo *data-driven*. Así, en el artículo [24] se destacan las siguientes técnicas:

- *Fuzzy Systems*: Este tipo de sistemas se basan en la lógica difusa (*fuzzy logic*). Este tipo de lógica es bastante similar a la clásica pero con la diferencia

de que puede enunciar afirmaciones con cierto grado de seguridad (por ejemplo: 'Este inversor solar fallará con una probabilidad del 70 %'). Una parte principal de los *fuzzy systems* es que necesitan de conocimiento de expertos para poder crear las reglas difusas. Así, estas técnicas utilizan este conocimiento técnico junto con otro métodos como redes neuronales o *clustering*. En [25] y [24] se pueden encontrar algunos casos de uso, además, el primero presenta información más técnica sobre estos conceptos.

- *Qualitative Trend Analysis (QTA)*: Es una técnica utilizada para analizar tendencias en series temporales de forma cualitativa, es decir, sin requerir modelos matemáticos complejos ni cantidades exactas. Es especialmente útil cuando los datos son ruidosos, incompletos o inciertos, como suele ocurrir en sistemas industriales o eléctricos. Se distinguen dos pasos: extracción de tendencias (se transforman los datos para encontrar patrones de tendencia) y análisis de tendencia.

- Métodos estadísticos: Se suelen emplear para extraer características de los datos y así poder reconocer patrones. Son varias las técnicas estadísticas usadas en este sentido, pero las principales son las siguientes:

  1. *Principal Component Analysis (PCA)*: Es una de las técnicas lineales de reducción de la dimensionalidad más utilizadas. Trata de disminuir la dimensión de los datos minimizando la pérdida de información. Esta técnica se usará más adelante en el trabajo para reducir la dimensión de los datos obtenidos de los inversores solares.
  2. *Partial Least Squares (PLS)*: Combina técnicas de regresión lineal, PCA y reducción de la dimensionalidad. A partir de las variables originales extrae nuevas variables las cuales se encuentran proyectadas en el conocido como espacio latente.
  3. *Support Vector Machine (SVM)*: Es un algoritmo de aprendizaje supervisado que se utiliza para clasificación y regresión, aunque es más conocido por su uso en clasificación. Trata de encontrar el mejor hiperplano que separa las distintas clases de datos de forma óptima maximizando la distancia (margen) entre los datos de cada clase y su frontera. Para más información acerca de los conceptos matemáticos subyacentes, puede consultarse [26].

- Métodos estocásticos: Son técnicas matemáticas y estadísticas que se utilizan para modelar y analizar sistemas o procesos que evolucionan de manera aleatoria o probabilística en el tiempo o en el espacio. Estos métodos incorporan variables aleatorias, procesos estocásticos y distribuciones de probabilidad

para describir fenómenos cuyo comportamiento no es determinista. Muchos de estos métodos se basan en el conocido teorema de Bayes donde la probabilidad P de dos eventos distintos A y B se define como:

$$P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)}$$

Este tipo de métodos se pueden combinar con redes neuronales dando lugar a las redes neuronales bayesianas. En este tipo de redes, los pesos y *biases* de la red no son valores fijos si no distribuciones de probabilidad, permitiendo el manejo de la incertidumbre en el modelo. Puede consultarse [26] para una mayor profundidad técnica.

- *Artificial Neural Networks (ANN)*: El objetivo es ajustar los pesos y parámetros de una red neuronal. La red consiste en una capa de entrada, varias capas ocultas (opcionales) y una capa de salida. La entrada será el *dataset* a analizar al que se le aplicarán técnicas de reducción de dimensionalidad o extracción de características. Se pueden clasificar los tipos de ANN en función de la arquitectura de la red, objetivo de la salida y método de aprendizaje. Las más conocidas son: *MultiLayer Perceptron (MLP)*, *Radial Basis Function (RBF)*, *Convolutional Neural Networks (CNN)*, *auto-encoders* y *Recurrent Neural Networks* (RNN).

  Estas dos últimas son las más utilizadas para tareas de detección de anomalías. En [27] se utiliza la técnica de *clustering K-Means* junto con *Long Short-term Memory (LSTM)* (un tipo de RNN) para detectar anomalías en una planta fotovoltaica solar de grandes dimensiones. Por otra parte, en [28] se utilizan *auto-encoders* para extraer de forma no supervisada características de datos los cuales presentan cierto ruido. De esta forma, se consigue desarrollar un modelo más robusto frente a datos ruidosos en el conjunto de entrenamiento.

Desafíos y futuras líneas de investigación

Aunque en los últimos años, las investigaciones conducidas sobre la aplicación de la inteligencia artificial en el mantenimiento eléctrico han aumentado considerablemente, aún queda un largo camino por recorrer. En muchas empresas, sigue siendo complicado aplicar este tipo de soluciones debido a la calidad de los datos o la falta de medios. En el artículo [29], se presenta una revisión bastante interesante sobre los retos y futuras líneas de investigación en este campo.

En relación a los retos, en este artículo se destacan los siguientes:

1. Disponibilidad de los datos: La calidad de las predicciones de los modelos de *machine learning* dependen en gran medida de la calidad de los datos introducidos como *input*. Mala calidad en los datos implican baja fiabilidad y rendimiento de los modelos (lo que se conoce como el concepto de *Garbage In Garbage Out*)[30].

   El sector de las energías renovables es uno de los que menos datos de calidad tiene para el entrenamiento de modelos de mantenimiento predictivo. Esto puede deberse al formato de los datos que se obtienen de los sistemas de monitoreo como SCADA, o la falta de medidas de gobierno del dato. Esta situación, puede provocar que los modelos entrenados se aprendan muy bien los datos de entrenamiento e incluso el ruido intrínseco de los mismos y no sean capaces de generalizar bien a los datos de *test* (esto se conoce como *overfitting*).

2. Auditoría de datos: este concepto está muy ligado al punto anterior. Resulta necesario realizar ciertas revisiones sobre los datos para comprobar su calidad y adecuación para una aplicación en concreto. Gracias a estas auditorías, se puede asegurar la validez de los algoritmos implementados a lo largo de los años y comprobar que están aprendiendo lo esperado.

3. Selección de características *(Feature Engineering)*: Muchos de los datos de equipos obtenidos a partir de herramientas de monitoreo como SCADA presentan una gran cantidad de variables (elevada dimensionalidad). Cuantas más variables introduzcamos al modelo de ML más complejidad tendrá este y mayor tiempo computacional requerirá su entrenamiento, es el fenómeno conocido como *The curse of dimensionality* [31].

   Para evitar este problema, es necesario seleccionar unas pocas variables de todas las disponibles (las más informativas) para el caso de uso. Esta selección se puede realizar de forma manual (algoritmos de ML) o de forma automática (algoritmos de *deep learning*). En ambos casos, se necesitará del conocimiento de expertos para poder comprobar que las variables escogidas son adecuadas para el caso concreto.

4. Simulación simultánea de múltiples fallos: Es habitual que muchos de los fallos que presentan los equipos eléctricos se deban no solamente a un fallo si no al cúmulo de varios. Estas situaciones, resultan más difíciles de modelar debido a la interacción entre ellas y la dificultad de etiquetar los fallos de este tipo.

5. Equilibrio entre rendimiento y explicabilidad: A raíz de la aprobación del Reglamento de Inteligencia Artificial en Europa, es necesario que las empresas sean transparentes con los algoritmos de IA que implementan. Por ello,

resulta importante que se puedan entender y explicar las predicciones de los modelos. Muchos de los algoritmos más potentes, como las redes neuronales o algoritmos de *deep learning*, presentan baja explicabilidad por lo que muchas investigaciones se están conduciendo en ese sentido.

Como se verá más adelante, en el desarrollo del modelo para el fallo de condensación en inversores solares, aparecerán algunos de estos retos como la disponibilidad de los datos o la selección de características.

Para terminar esta sección, mostraremos algunas de las líneas de investigación más destacables en este campo. La principal línea de investigación es el desarrollo de técnicas de explicabilidad para redes neuronales y algoritmos de *deep learning*. Existen varios artículos que tratan este tema pero, en este trabajo, se destacan [32] y [33] pues introducen el término XAI (*Explainable Artificial Intelligence*) y ofrecen una recopilación sobre los últimos estudios.

Por otra parte, sectores como el energético son más propensos a sufrir ataques cibernéticos que afecten al rendimiento de los modelos provocando falsas alarmas de fallos en sistemas energéticos. Por tanto, es necesario seguir investigando para desarrollar modelos resistentes ante estos ataques [34], [35].

En relación con los datos, como la aparición de un fallo suele ser un evento atípico en las plantas energéticas, resulta difícil recolectar muestras de eventos de fallo. Algunas soluciones pasarían por crear datos sintéticos o utilizar modelos preentrenados que se ajusten al problema a resolver (*transfer learning*).

Por último, relacionado con la escasez de datos se destaca la técnica de *Novelty Detection*. Se trata de una técnica de aprendizaje automático que se encarga de identificar patrones o eventos que son significativamente diferentes o no vistos anteriormente durante el entrenamiento del modelo. Es algo distinto a *Outlier Detection* pues en esa técnica se introducen datos normales y anómalos en el conjunto de entrenamiento mientras que en *Novelty Detection* solo se entrena con datos normales. En este artículo [36], se hace un breve repaso sobre los estudios realizados sobre detección de novedades.

2.2.3. Algunas aplicaciones de inteligencia artificial en mantenimiento eléctrico solar

En los últimos años, han aumentado considerablemente los estudios sobre las aplicaciones de la IA en el mantenimiento eléctrico. Tras una revisión de la literatura, se ha observado una gran cantidad de artículos centrados en la energía eólica, mientras que el número de publicaciones en el ámbito solar es aún limitado. No obstante, debido al creciente auge de la energía solar en los últimos años, están aumentando las investigaciones científicas sobre la aplicación del aprendizaje automático al mantenimiento predictivo.

En esta última sección, se destacan algunas de las metodologías propuestas en ciertos *papers* científicos, seleccionadas por la relevancia de sus soluciones y su similitud con el caso de uso presentado en este trabajo. A parte de los presentados a continuación, se recomienda consultar [27] en el que se aplica *K-Means* y LSTM para la detección de anomalías, [37], [38] donde se comparan distintos algoritmos de ML en campos solares y [39] donde se aplica *Feed-Forward Back-Propagation Neural Neutwoks* mostrándose trambién la creación del KPI a estudiar.

Predicción de fallos en inversores solares utilizando un algoritmo de *Fast Clustering* y *Gaussian Mixture Model*

En este artículo [40], se propone un método de detección de fallos en inversores solares basado en la comparación entre *clusters* de inversores. La idea principal es evaluar la similitud entre el comportamiento de un inversor objetivo y un patrón de referencia construido a partir de inversores en buen estado, utilizando técnicas de reducción de dimensionalidad, agrupamiento y modelado probabilístico.

Primero, se construyó un espacio de características mediante el algoritmo *t-distributed stochastic neighbor embedding* (t-SNE), con el fin de reducir la elevada dimensionalidad de los datos. A continuación, se aplicaron técnicas de *Fast Clustering*, un conjunto de métodos de agrupamiento eficientes para grandes volúmenes de datos. En particular, se utilizó un enfoque basado en densidad (*density-based clustering*), que identifica el centroide del *cluster* como el punto con mayor densidad local, estimada mediante dos métodos: el *Gaussian kernel* y el *cut-off kernel* [41], [42].

Una vez definidos los *clusters*, se utilizó un modelo de mezclas gaussianas (GMM) para caracterizar la distribución tanto del centroide (inversor de referencia) como del inversor objetivo. El grado de solapamiento entre ambas distribuciones, medido

mediante la divergencia de Jensen–Shannon (JSD) [43], se interpretó como un indicador del estado de salud del inversor: cuanto mayor es el solapamiento, más normal es su comportamiento y menor la probabilidad de fallo.

Por último, se recomienda al lector la consulta directa del artículo, no solo por la calidad de las técnicas empleadas, sino también por las ilustraciones que incluye, las cuales resultan especialmente informativas.

Mantenimiento predictivo basado en detección de anomalías en sistemas fotovoltaicos usando datos SCADA y ML

El estudio [44] propone un enfoque de mantenimiento predictivo en plantas solares fotovoltaicas de reciente construcción, basado en los patrones diarios de irradiación y corriente alterna (AC). Dado que los datos no están etiquetados, opta por un enfoque no supervisado utilizando un modelo de *Long Short-Term Memory AutoEncoder* (LSTM-AE) para la detección de anomalías [45], [46].

Los datos fueron preprocesados mediante eliminación de duplicados, imputación de valores faltantes y ajuste de tipos de columnas. Posteriormente, se entrenó el modelo LSTM-AE con datos normales, aprendiendo el comportamiento típico del sistema. Este tipo de red neuronal está diseñada para procesar secuencias temporales y está formada por dos partes: un codificador, que resume la secuencia en una representación comprimida, y un decodificador, que intenta reconstruirla.

Asimismo, se definió un umbral de error basado en la raíz del error cuadrático medio (RMSE por sus siglas en inglés), el cual actúa como medida de desviación estándar del funcionamiento normal del sistema. Este umbral permite distinguir comportamientos anómalos.

Además, se implementaron alarmas predictivas basadas en la derivada de la tendencia del residuo de reconstrucción: si es positiva, sugiere una posible degradación del inversor; si es negativa, el sistema se comporta con normalidad. Para evitar revisiones innecesarias, se enfatizó la necesidad de minimizar los falsos positivos.

El estudio reporta buenos resultados con métricas MSE, RMSE y MAE de 10.95, 3.30 y 2.76, respectivamente. Como trabajo futuro, se propone combinar este enfoque con el modelo ARIMA, ampliamente utilizado en series temporales.

Nuevo método para el diagnóstico de fallos en un *array* fotovoltaico usando *Fuzzy C-Mean Clustering* y *Fuzzy Membership Clustering*

Este artículo [47] trata de implementar un método eficaz para detectar 6 fallos distintos en los elementos de un *array* fotovoltaico.

Lo primero que se realiza es aplicar *Fuzzy C-Mean Clustering* (FCC) para obtener los centroides de cada *cluster* (se tienen 6 *clusters*, uno por fallo). El siguiente paso, será calcular la función de pertenencia. Para ello, hay que tener en cuenta que los parámetros más representativos de un *array* van cambiando a medida que este está dañado o no. Por ello, se escoge la función de distribución normal de estos parámetros para calcular el grado de pertenencia entre las muestras y cada uno de los centroides, su expresión matemática sería la siguiente:

$$\mu(x) = e^{-\frac{(x-\mu)^2}{2\sigma}} \times 100\,\%$$

donde $\mu(x)$ es el grado de pertenencia del parámetro x y σ y μ la varianza y media respectivamente de la distribución normal. Se destaca que esta función de pertenencia tomará valores en el intervalo $[0, 1]$.

En el estudio, se comprueba que el algoritmo presenta un 96 % de *accuracy* por lo que desempeña notablemente la tarea de detección de fallos. También, se compara el uso de *K-Means* frente a FCC mostrándose que el *clustering* difuso se adapta mejor a este caso de uso. Además, se prueba a implementar una *Back Propagation Neural Network* pero los resultados son peores debido a la sensibilidad de este modelo a la escasez de datos de fallo [48], [49].

Por último, se muestra la adaptabilidad de este método a la aparición de nuevos tipos de fallos (basta con elevar el número de *clusters* en FCC) así como a periodos de transición entre fallos del sistema.

Capítulo 3

Definición del problema

Tras una extensa revisión sobre el estado actual de la industria eléctrica y las técnicas empleadas para el mantenimiento predictivo, en este capítulo se introducirá el problema a abordar y se explicarán algunos aspectos clave, como el funcionamiento de un inversor solar y el problema de la condensación.

Asimismo, se detallarán las fases que se seguirán en el proyecto. De esta forma, se podrá tener una visión global sobre el mismo.

3.1. Objetivo

Actualmente, la empresa cuenta con varios sistemas de adquisición de datos instalados en sus equipos, incluidos los inversores solares. Sin embargo, estos datos no están siendo plenamente explotados, a pesar de su potencial para aplicaciones de mantenimiento predictivo. En el contexto actual, donde los datos representan un recurso estratégico, Gamesa Electric ha manifestado su interés en iniciar el desarrollo de soluciones basadas en inteligencia artificial aplicadas al mantenimiento de sus equipos.

Aunque el objetivo ideal sería crear un modelo de clasificación general de anomalías, este se ve limitado por diversos factores. Por un lado, la calidad y homogeneidad de los datos disponibles no es la óptima, debido a la dificultad de su recolección y el hecho de ser la primera vez que se analizan de forma profunda. Por otro lado, el sistema actual de alarmas de los inversores, si bien proporciona

información útil, no ofrece etiquetas claras ni específicas que puedan emplearse directamente para entrenar modelos supervisados, ya que muchas de las alarmas son meramente informativas.

No obstante, tras conversaciones con personal técnico de la empresa, se identificó un problema específico que genera especial preocupación: la condensación en los inversores. Este fenómeno, además de crítico para el funcionamiento del equipo, está asociado directamente con una alarma concreta (la alarma 721), lo que permite contar con una etiqueta fiable para el entrenamiento del modelo.

Por tanto, el objetivo concreto de este proyecto será desarrollar un modelo de clasificación binaria capaz de detectar casos de condensación en inversores solares, diferenciándolos de su funcionamiento normal. Para ello, será necesario comprender en profundidad el problema de la condensación y algunos conceptos técnicos relacionados, así como realizar tareas de limpieza, análisis y procesamiento de datos, y llevar a cabo la posterior validación de los modelos probados.

3.2. ¿Qué es un inversor solar?

Este trabajo gira en torno a la figura del inversor solar, por lo que resulta pertinente incluir una breve descripción sobre su funcionamiento y funcionalidad con el fin de contextualizar el problema a abordar.

Un inversor solar es un dispositivo electrónico esencial en cualquier sistema fotovoltaico, ya que se encarga de convertir la energía eléctrica generada por los paneles solares —producida en corriente continua (CC)— en corriente alterna (CA), que es la forma de energía utilizada por la mayoría de los electrodomésticos y la que puede ser inyectada a la red eléctrica.

Más allá de esta conversión, los inversores modernos también cumplen funciones adicionales como la monitorización del rendimiento del sistema, la detección de fallos y la protección frente a condiciones anómalas. Su correcto funcionamiento es crucial para garantizar tanto la eficiencia energética como la seguridad del conjunto.

Por todo ello, se han convertido en uno de los componentes más importantes de cualquier parque solar, lo que ha motivado a la empresa a explorar nuevas estrategias para la detección temprana de fallos en estos equipos.

Figura 3.1: Imagen de un inversor solar en uno de los parques de la empresa. Este en concreto se encuentra situado en Myrtle, Estados Unidos.

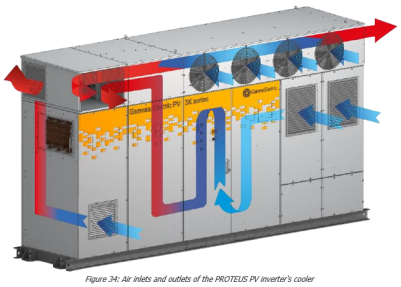

Figura 3.2: Esquema de funcionamiento de un inversor en el que se muestran los flujos del sistema de refrigeración y caldeo mediante líquido glicol.

Como se muestra en la Figura 3.2, el inversor solar presenta unos sistemas de refrigeración y caldeo necesarios para garantizar la temperatura óptima del equipo. Uno de los motivos por los que son necesarios este tipo de medidas es para evitar problemas como el de condensación que explicaremos a continuación.

3.2.1. El problema de condensación

La condensación es uno de los problemas más habituales y que más afectan al rendimiento de los inversores solares. Este, está estrechamente relacionado con la temperatura interna y externa del inversor.

En un estado de correcto funcionamiento, la temperatura interna de los mismos supera a la temperatura ambiente. Esta temperatura interna es medida a través de unas componentes conocidas como skiips. En el modelo de inversor solar con el que se está trabajando, se tienen 6 de estas componentes por lo que la temperatura interna del inversor solar será el mínimo de la temperatura de todos los skiips. Por otra parte, la temperatura externa del equipo se mide mediante unas cabinas de bloques de potencia situadas en la parte externa del inversor solar. En el caso de

estos inversores, se tienen dos cabinas de este tipo y la temperatura externa se calcula como el máximo entre estas dos.

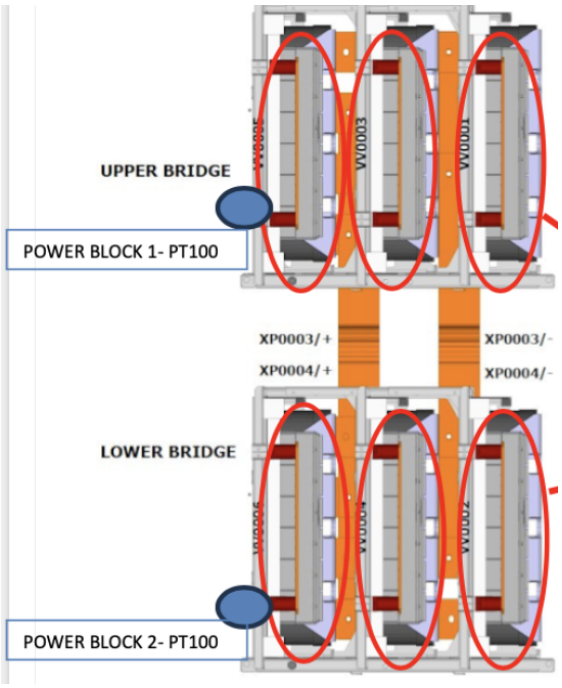

Figura 3.3: Esquema con algunas componentes internas del inversor. Los cículos azules representan las dos cabinas de bloques de potencia mientras que los cículos rojos corresponden con los skiips.

No obstante, cuando la temperatura ambiente supera a la interna, se generan ciertas gotas (similares al rocío) en el interior del inversor lo que afecta negativamente al rendimiento del mismo. Por ello, una vez calculadas estas dos temperaturas, se considera como indicador de condensación la diferencia entre la temperatura interna y la externa. A este valor se le conoce como el delta de temperatura.

Para evitar la situación de que esta variable sea negativa (condensación) se estableció que, cuando su valor sea inferior a 7^{o}C, se empiece a caldear el equipo. Este caldeo es llevado a cabo por un sistema interno hasta alcanzar una temperatura de 10^{o}C, momento en el cual se parará de caldear.

Por último destacar que hay situaciones en las que la temperatura de los skiips es inferior a 31^{o}C siendo necesario pasar líquido glicol caliente por ellos para aumentar su temperatura. En estos casos, la temperatura interna del inversor será el mínimo entre la temperatura de entrada y salida del líquido glicol.

3.3. Descripción de los datos

Durante todo el proyecto se trabajó principalmente con archivos en formato CSV, que contenían o bien información temporal sobre diversas variables medidas en los inversores (un archivo por inversor) o bien históricos de alarmas asociadas a cada uno.

En una primera fase, se dispuso de 15 archivos CSV correspondientes a 15 inversores en distintos días, cada uno con la evolución temporal de 50 variables. Además, se contaba con 10 CSV adicionales que recogían el histórico de alarmas de algunos de estos inversores. Solo dos inversores estaban etiquetados como defectuosos: uno por una explosión tras congelar y otro por una pérdida de presión.

Más adelante, ante la necesidad de disponer del registro completo de alarmas por inversor y las dificultades para generalizar el comportamiento de fallo, se recopilaron nuevos datos de un parque solar de la empresa ubicado en Myrtle (Estados Unidos). Esta nueva tanda incluyó 70 CSV con datos temporales de 70 inversores distintos (cada uno con 41 variables) y otros 70 CSV con los históricos de alarmas correspondientes a estos equipos.

Finalmente, debido a la baja presencia de fallos por condensación en el parque de Myrtle, se recopilaron más datos de otro parque solar, denominado Sigma, situado en Cádiz. En este caso, se obtuvieron 3 CSV con la evolución temporal de variables asociadas a 3 inversores distintos (las mismas que en el caso de Myrtle) y otros 3 CSV con los registros de alarmas correspondientes.

3.3.1. Descripción de las variables y alarmas a estudiar

Las variables medidas por los sistemas de monitorización de los inversores abarcan aspectos físicos como la temperatura interna y externa, la humedad, los voltajes o la potencia. Aunque inicialmente se tuvieron en cuenta todas estas variables, en una fase posterior del proyecto —tras consultas con perfiles técnicos de la empresa— se decidió centrar el análisis en un conjunto reducido de variables derivadas, calculadas a partir de las originales, que mostraban una mayor relación con el problema de condensación. Debido a su importancia durante el trabajo y el elevado número de variables en el *dataset*, a continuación se describen brevemente dichas variables asociadas con la condensación:

- T_SKIIP_MIN: valor mínimo de temperatura de los skiips. Se calcula de la siguiente forma:

  $\texttt{T\_SKIIP\_MIN} = \text{mín}(\texttt{TBRID11}, \texttt{TBRID12}, \texttt{TBRID13}, \texttt{TBRID21}, \texttt{TBRID22}, \texttt{TBRID23})$

- TPOT_CAB_MAX: valor máximo de las temperaturas de las cabinas de bloques de potencia 1 y 2. Calculada como:

  $$\texttt{TPOT\_CAB\_MAX} = max(\texttt{TCAB\_POT1}, \texttt{TCAB\_POT2})$$

- T_LIQ_MIN: valor mínimo entre las temperaturas de entrada/salida del líquido Glicol. Se obtiene de la siguiente forma:

  $$\texttt{T\_LIQ\_MIN} = min(\texttt{TCOOLINGI}, \texttt{TCOOLINGO})$$

- DELTA_TEMP_AMB: valor que indica la diferencia de temperatura entre el interior del inversor y la del exterior. En este caso, se tienen dos formas de calcularlo dependiendo si $\texttt{T\_SKIIP\_MIN} < 31°\text{C}$:

  1. Si $\texttt{T\_SKIIP\_MIN} \geq 31°\text{C}$, entonces:

     $$\texttt{DELTA\_TEMP\_AMB} = \texttt{T\_SKIIP\_MIN} - \texttt{TPOT\_CAB\_MAX}$$

  2. En caso contrario:

     $$\texttt{DELTA\_TEMP\_AMB} = \texttt{T\_LIQ\_MIN} - \texttt{TPOT\_CAB\_MAX}$$

En relación con las alarmas, estas también son bastante numerosas por lo que únicamente se definirá la alarma 721 pues es la que se utilizará principalmente en el proyecto. Esta alarma indica que el sistema de protección contra la humedad está activado y aparece cuando $\texttt{DELTA\_TEMP\_AMB} < 7\,°\text{C}$.

Por último, cabe destacar que aunque finalmente no se ha tenido en cuenta la alarma 720, esta también guarda cierta relación con el problema de condensación ya que está vinculada al sistema de caldeo del inversor. Esta alarma indica que el sistema no ha logrado calentar el equipo dentro del plazo de una hora.

En algunos casos, la alarma 721 puede estar precedida por una alarma 720. Sin embargo, la presencia de la alarma 720 no implica necesariamente la aparición de la 721 ni un fallo por condensación. Dado que se buscaba que la etiqueta de fallo por condensación fuera lo más precisa posible, se decidió considerar únicamente la alarma 721.

3.3.2. Sistema de recolección

A continuación se describe brevemente el sistema de recolección de datos actualmente implementado por la empresa.

La arquitectura de monitoreo de los inversores solares se basa en dos componentes principales: los *dataloggers* y la Unidad de Control Central (CCU). Los *dataloggers*, normalmente uno por inversor, registran datos procedentes de sensores físicos y los almacenan localmente. Actualmente, estos dispositivos no cuentan con capacidad de transmisión remota, por lo que la recolección de datos requiere conexión manual mediante ordenadores o memorias USB, lo que implica desplazamiento de personal al campo.

En cuanto a la CCU, esta actúa como nodo central del sistema y, en arquitecturas completas recibe, procesa y transmite los datos recogidos por los *dataloggers* hacia sistemas superiores. Por lo general, se instala una única CCU por parque solar, aunque este número puede variar según el tamaño del sistema. En cuanto a la arquitectura de comunicaciones, la CCU suele operar como servidor, mientras que los *dataloggers* funcionan como clientes.

La empresa tiene previsto implementar *dataloggers* remotos en el futuro, pero durante el desarrollo del presente trabajo aún no estaban habilitados, lo que dificultó el acceso automatizado a la información.

3.4. Etapas del trabajo

Para alcanzar los objetivos planteados, el desarrollo del trabajo se dividió en distintas etapas, cada una con un propósito específico:

1. Análisis de las variables temporales de los inversores y sus alarmas: En esta etapa se utilizarán los archivos CSV correspondientes a la primera fase, detallada en la sección 3.3. Se presentarán las distribuciones de cada variable, así como sus correlaciones y otros aspectos relevantes.

   Esta etapa será útil para obtener una interpretación general de los tipos de datos disponibles, su formato y los valores más representativos. Asimismo, mediante la identificación de valores atípicos, será posible detectar situaciones potencialmente asociadas a comportamientos anómalos.

Por último, se visualizará la evolución temporal de algunas variables en distintos inversores, y se realizará un análisis de frecuencia de los diferentes tipos de alarmas, con el objetivo de identificar las más recurrentes.

2. Modelo de *clustering* para la identificación de dos grupos de inversores (con comportamiento anómalo o normal): En esta fase se evaluarán tres modelos de *clustering*: *K-means*, *C-means* y *Gaussian Mixture Models (GMM)*, con el objetivo de clasificar las muestras de los inversores como anómalas o no anómalas. Debido a la alta dimensionalidad de los datos, se aplicará el algoritmo de *Principal Component Analysis (PCA)* como técnica de reducción de dimensionalidad.

   Para ello, se utilizarán los datos recolectados en el parque solar de Myrtle, a los cuales se añadirán las variables calculadas relacionadas con el problema de condensación. Cabe destacar que, dado que inicialmente no se contaba con datos etiquetados, se optó por desarrollar un modelo de *clustering* con dos grupos, destinado a separar las muestras correspondientes a inversores con fallos (de cualquier tipo) de aquellas sin fallos reportados.

3. Etiquetado manual de los datos: Una vez definido el objetivo de detectar el problema de condensación en los inversores, se procederá al etiquetado manual de los mismos. Tras mantener conversaciones con perfiles técnicos de la empresa, se concluyó que la aparición de la alarma 721 es un indicador suficiente para etiquetar a un inversor como afectado por fallo de condensación (representado con la etiqueta 1).

4. Desarrollo de un modelo de clasificación binaria para la detección de fallos por condensación: En esta última fase se desarrollarán dos modelos basados en árboles de decisión: *Decision Tree* y *XGBoost*. Ambos modelos serán evaluados utilizando dos tipos de *inputs*: uno conformado por las cuatro variables calculadas definidas en la sección 3.3.1, y otro en el que se aplicará nuevamente PCA como técnica de reducción de dimensionalidad.

En las secciones siguientes, se procederá a explicar más en detalle cada una de estas fases del trabajo y los resultados obtenidos en ellas. Asimismo, se mostrarán algunos de los principios matemáticos subyacentes en las técnicas empleadas y algunos parámetros importantes utilizados en el desarrollo del código.

3.5. Recursos empleados

En relación con los datos, la empresa fue la encargada de recolectarlos y facilitarlos a la autora a través del espacio de *SharePoint* de *Microsoft*. Además, proporcionaron varios archivos *Excel* con explicaciones sobre las variables y alarmas de los inversores. También se mantuvieron reuniones periódicas con personal técnico de la empresa para aclarar aspectos clave del proyecto, como el problema de condensación y el funcionamiento de los inversores.

Respecto al lenguaje de programación, se utilizó `Python` por su sintaxis clara, baja complejidad inicial y gran expresividad. Su versatilidad y legibilidad lo han consolidado como uno de los lenguajes más utilizados en ciencia de datos, aprendizaje automático e inteligencia artificial.

Durante el trabajo se emplearon varias librerías de `Python`. Se destaca el papel de las siguientes:

- `pandas`: Para la manipulación y análisis de datos.
- `numpy`: Para realizar cálculos numéricos en `Python` mediante estructuras de datos eficientes.
- `scikit-fuzzy (skfuzzy)`: Para lógica difusa y *clustering* difuso.
- `scikit-learn (sklearn)`: Para la implementación de modelos de aprendizaje automático.
- `matplotlib` y `seaborn`: Para visualización de datos.

El entorno de desarrollo elegido fue `Jupyter Notebook` [50], ideal para proyectos exploratorios por su carácter interactivo. Permite ejecutar código en bloques, visualizar resultados de forma inmediata y documentar el proceso en un solo lugar. Además, es compatible tanto con entornos locales como en la nube, lo que aporta flexibilidad.

En cuanto a los recursos *hardware*, se utilizó un ordenador personal para las tareas menos exigentes de procesamiento y entrenamiento de modelos. Sin embargo, debido a la ausencia de GPU y a la memoria limitada del equipo, fue necesario recurrir a `Google Colab` para entrenar modelos más pesados, como los basados en *XGBoost*.

En relación con esto último, cabe destacar que se intentó utilizar herramientas de procesamiento distribuido como `PySpark`, que no presentan las restricciones de memoria propias de las sesiones de `Google Colab`. Estas herramientas resultarían muy útiles para procesar el gran volumen de datos disponible y para el entrenamiento de modelos más complejos. Sin embargo, no fue posible debido a la falta de acceso a estos recursos.

Capítulo 4

Análisis de las variables temporales y alarmas

Este capítulo se centra en la primera fase del trabajo. En él se va a explicar con más detalle los distintos métodos llevados a cabo en el análisis exploratorio del primer grupo de datos recibidos.

Este análisis fue especialmente útil para familiarizarse con las principales variables físicas de los inversores. Además, también sirvió para entender en más detalle la estructura de los CSV, en especial de los correspondientes a las alarmas.

4.1. Preprocesado de los datos

Previo a llevar a cabo cualquier tipo de análisis, se tuvo que proceder a la lectura de los distintos archivos. Esta tarea resultó bastante sencilla para los archivos de alarmas pero no tanto para los CSV con las variables. Estos incluían algunas columnas finales que no aportaban información y desestructuraban el fichero. Se procedió a eliminar estas columnas y, haciendo uso de la librería `pandas`, se consiguió leer correctamente los archivos guardando la información en `dataframes` [51].

A continuación, se estudiaron los tipos de cada variable y la presencia de valores faltantes en el *dataset*. Casi todas las columnas eran de tipo `float` a excepción de `utc` y `sample` que eran de tipo `int` (representaban el tiempo en formato UTC y

el número de la muestra respectivamente) y `time` que era de tipo `datetime`. Por último, se vio que todas las variables tenían 26.943 registros por lo que no había ningún valor faltante (NaN).

4.2. Técnicas de análisis exploratorio

Seguidamente, se llevaron a cabo principalmente técnicas para visualizar la distribución de las variables así como para ver las correlaciones entre ellas.

4.2.1. Distribución de las variables

Para comprender mejor el rango de valores de las variables, se decidió analizar su distribución. Para ello, se utilizó el método `describe` de `pandas`, el cual proporciona información sobre la media, valores máximos y mínimos, cuartiles y desviación estándar de cada variable.

A partir de este análisis, se identificaron distintos grupos de variables, entre los que destacan los siguientes:

- Variables con un rango reducido de valores: Entre las variables de este grupo se encontraban algunas relacionadas con la tensión y distintas temperaturas.
- Rango de valores muy amplio en variables como las asociadas a la potencia o la corriente.
- Variables con mucha diferencia entre el tercer cuartil y su valor máximo: Esto podía indicar tanto la presencia de valores atípicos (*outliers*) como situaciones de comportamiento anómalo del inversor. Por esto último, no se decidió eliminar estos valores pues podrían resultar informativos en fases siguientes.
- Valores constantes en las variables: Se observó que 3 de las 50 variables eran nulas. Esto llevó a pensar que quizá siempre iban a ser nulas planteándose eliminarlas, pero la empresa aclaró que dicha situación fue casual y que estas variables no tienen por qué ser siempre nulas.

Tras este estudio, se graficaron las distribuciones de todas las variables, excepto `time`, `utc` y `sample` pues estas no aportaban información relevante por si solas . Los

histogramas resultantes se muestran a continuación y confirman las observaciones descritas.

Figura 4.1: Histogramas que representan la distribución de las variables temporales de los archivos CSV recibidos en la primera ronda de datos. No se graficaron la distribución de las variables `time`, `utc` y `sample` pues no aportaban información relevante.

Por último, también se graficaron los *boxplots* asociados a cada variable para obtener de forma más visual información sobre la mediana, cuartiles y posibles *outliers*. Los puntos representan valores atípicos y se puede observar que hay variables como `TBRID22` con un elevado número de ellos. Esto podría indicar situaciones anómalas del inversor debido a valores extraños en esas variables.

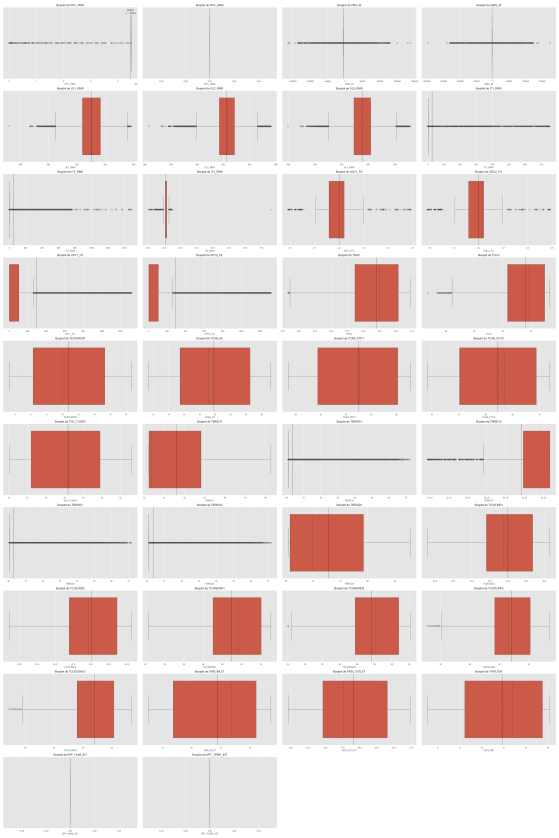

Figura 4.2: *Boxplots* de las variables temporales de los archivos CSV recibidos en la primera ronda de datos. No se graficaron los correspondientes a las variables `time`, `utc` y `sample` pues no aportaban información relevante.

4.2.2. Correlaciones lineales

Una vez hecho el análisis de las distribuciones, se estudiaron las posibles correlaciones lineales entre las variables. Para su cálculo, se utilizó la función `corr()` para `dataframes` de `pandas`. Esta se basa en el cálculo del coeficiente de correlación de Pearson cuya fórmula es la siguiente para dos variables X, Y:

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \cdot \sigma_Y}$$

Es importante destacar que si alguna de las dos variables tiene desviación estándar igual a 0, entonces el denominador es 0 y la correlación queda indefinida (NaN). Para que fuera más visual se representó la matriz de correlaciones mediante un *heatmap* de la librería `seaborn`.

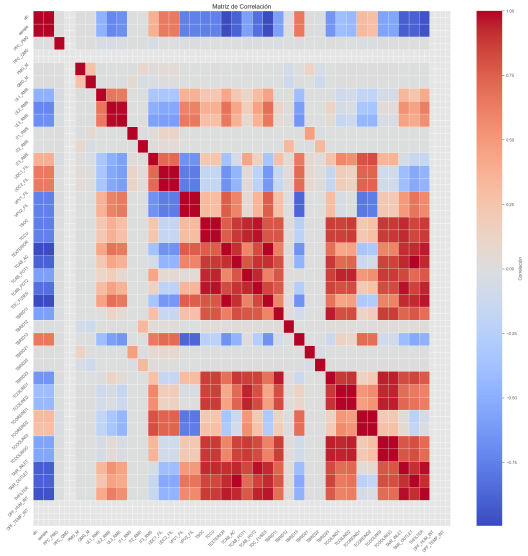

Figura 4.3: *Heatmap* que representa la matriz de correlaciones lineales de las variables temporales de los archivos CSV recibidos en la primera ronda de datos. Para su cálculo se utilizó el coeficiente de correlación de Pearson. Las celdas en tonos grises de tamaño menor se deben a la desviación estándar nula de la variable del denominador en la fórmula de Pearson.

Se pueden observar algunos grupos de variables altamente correlacionadas. Es el caso de las variables `TSOC`, `TCCU`, `TEXTERIOR`, `TCAB_AC`, `TCAB_POT1`, `TCAB_POT2`, `TDC_FUSES`, `TBRID11`, `TBRID23`, `TCOOLIND1` y `TCOOLIND2`. Por otra parte, llama la atención que variables como `TBRID` no estén altamente correlacionadas entre sí.

Este estudio resultó de ayuda para obtener una idea general sobre las variables de los inversores. No obstante, en fases posteriores del trabajo, se volverá a realizar un nuevo análisis pero con mayor cantidad de muestras obteniendo resultados más genéricos.

4.3. Evolución temporal de las variables

El siguiente paso que se llevó a cabo fue graficar la evolución temporal de las variables para los distintos inversores en ciertos días. En esta tanda de datos se tenía información para los días 12, 13, 14 y 15 de febrero de 2025. Además, se pudo detectar que en el día 12 hubo un fallo por pérdida de presión (no se sabe en qué inversor) y el día 13 sucedió una explosión tras congelarse el inversor 11. Por ello, los datos asociados a estos días se utilizaron para graficar la evolución temporal de las variables y poder ver diferencias de comportamiento entre los inversores normales

y los defectuosos. Además, se marcaron con líneas discontinuas las apariciones de alarmas en esos días y así poder ver cómo las variables cambiaban con la aparición de estas.

En relación con el día 12, se tenía información sobre dos inversores, el 33 y el 34 teniendo 86.193 muestras de estos inversores. Como no se había etiquetado el inversor que falló, se decidió dibujar la evolución temporal de las variables `PREO` y `PREI` que representan la presión de salida y entrada respectivamente de la bomba del inversor para ver si se podía identificar el fallido.

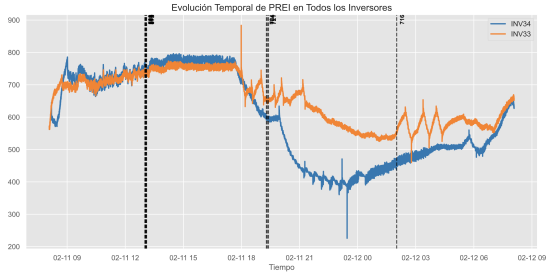

Figura 4.4: Evaluación temporal de la variable `PREI` para los inversores 33 (naranja) y 34 (azul). Se observa cómo desde las 21h del día 11-02-25 hasta las 03h del 12-02-25 el inversor 34 experimenta un descenso de presión en la entrada de la bomba del inversor. Las líneas discontinuas marcan la aparición de alguna alarma.

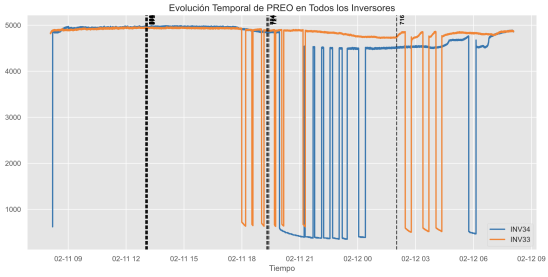

Figura 4.5: Evaluación temporal de la variable `PREO` para los inversores 33 (naranja) y 34 (azul). Se observa cómo desde las 21h del día 11-02-25 hasta las 00h del 12-02-25 el inversor 34 experimenta un descenso de presión en la salida de la bomba del inversor. Las líneas discontinuas marcan la aparición de alguna alarma.

Se puede ver en la evolución temporal de ambas variables que es en los gráficos correspondientes al inversor 34 donde más se aprecian descensos en la presión lo que lleva a pensar que fue este el inversor que experimentó pérdida de presión.

Por otra parte, para el día 13 se tenía información de los inversores 2, 4, 11, 33 y 34. Se sabía que el inversor 11 era el que había explotado tras congelarse por lo que se decidió graficar la evolución temporal de las variables temporales de este inversor junto con las del resto de inversores que en un principio habían tenido un comportamiento normal.

En la mayoría de variables se observa un comportamiento distinto del inversor 11. Por ejemplo, en las variables relacionadas con la potencia y la corriente se ve cómo para el inversor 11 estos valores son muy bajos en comparación con los otros equipos. Se observan ciertos picos en las gráficas de este inversor que indican intentos por aumentar dichos valores mientras que en otros casos los valores de las variables son nulos.

Por último, se observa un descenso en las variables de temperatura del inversor

11 en comparación con el resto de inversores. Este descenso normalmente coincide con momentos de mayor aparición de alarmas lo que indica que estas son muy informativas en estos casos.

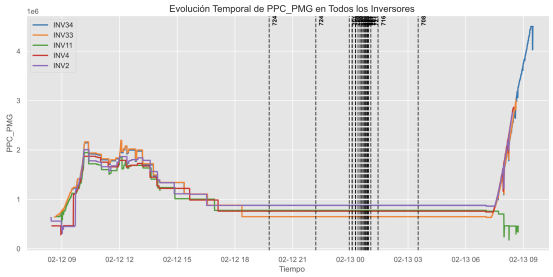

Figura 4.6: Evaluación temporal de la variable `PPC_PMG` (setpoint de potencia activa desde el SCADA) para los inversores 2 (morado), 4 (rojo), 11 (verde), 33 (naranja) y 34 (azul). Se observa cómo el sistema de potencia del inversor 11 está intentando aumentar el valor de esta variable sin éxito sobre las 7h del 13-02-25.

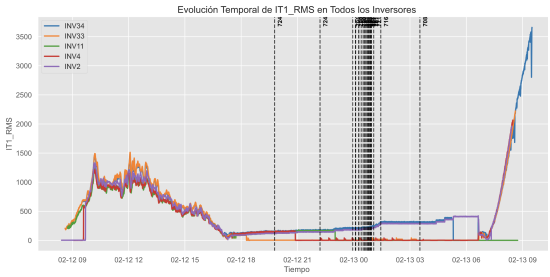

Figura 4.7: Evaluación temporal de la variable `IT1_RMS` (corriente de línea-1) para los inversores 2 (morado), 4 (rojo), 11 (verde), 33 (naranja) y 34 (azul). Se observa que el inversor 11 presenta, a partir de las 7h del día 13-02-25, un valor nulo para esta variable mientras que el resto de inversores tienen valores positivos.

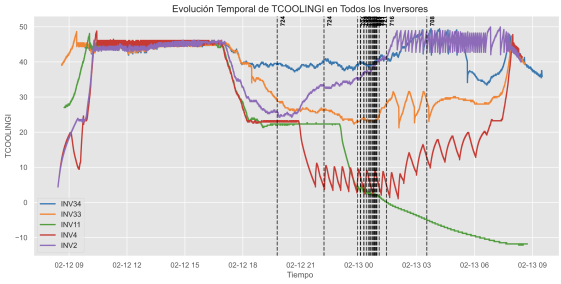

Figura 4.8: Evaluación temporal de la variable `TCOOLINGI` (temperatura de entrada del líquido) para los inversores 2 (morado), 4 (rojo), 11 (verde), 33 (naranja) y 34 (azul). Se observa que el inversor 11 presenta, a partir de las 00h del día 13-02-25, un descenso que intenta solucionar el equipo sin éxito (pequeños picos a partir de valores de temperatura inferiores a 7^{o}C). Estos picos se observan también en el inversor 4 pero este si fue capaz de aumentar la temperatura.

Como conclusión de este análisis, se puede afirmar que las variables proporcionadas son bastante informativas para identificar comportamientos anómalos en los inversores. Por tanto, resulta adecuado considerarlas en el desarrollo de modelos de clasificación orientados a detectar este tipo de comportamientos.

4.4. Frecuencia de las alarmas

Como último paso en este análisis exploratorio, se estudiaron las alarmas más frecuentes por día e inversor.

Por inversor, la alarma más habitual para casi todos los equipos fue la 723. Lamentablemente, el diccionario de alarmas proporcionado por la empresa no incluía detalles sobre esta alarma. Las dos siguientes alarmas más frecuentes fueron la 721 y la 720, también comunes en la mayoría de los inversores.

Por día, la frecuencia de alarmas mostró ligeras variaciones. En los días 12 y 13, identificados como días con algún tipo de fallo, el top 3 de alarmas más frecuentes estuvo formado por las alarmas 723, 720 y 721, siendo la 723 la más repetida en ambos casos, mientras que el orden de las otras dos varió. En cambio, en los días 14 y 15, considerados de funcionamiento normal, las alarmas más habituales fueron la 723, la 602 y la 720.

	Top1	Top2	Top3
Inv_1	723	400	535
Inv_2	723	720	721
Inv_4	723	721	720
Inv_33	723	721	720
Inv_34	602	723	720

Cuadro 4.1: Top 3 alarmas por inversor.

	Top1	Top2	Top3
12-02-25	723	721	720
13-02-25	723	720	721
14-02-25	723	602	720
15-02-25	723	602	720

Cuadro 4.2: Top 3 alarmas por día.

Gracias a este análisis se concluyó que las alarmas más relevantes para el estudio del comportamiento anómalo de los inversores son la 723, la 720 y la 721. Aunque en la fase siguiente del proyecto, correspondiente al modelo de *clustering*, todavía no se había puesto el foco en el fallo de condensación, posteriormente la empresa comunicó que las alarmas 720 y 721 estaban estrechamente relacionadas con un problema crítico que debía abordarse: el problema de condensación.

Capítulo 5

Modelo de *clustering*

Las técnicas de *clustering* tienen como objetivo dividir los datos en grupos (*clusters*) de forma que las diferencias entre las muestras asignadas a un mismo *cluster* sean menores que las existentes entre muestras de distintos grupos. Estas técnicas se encuentran dentro de la rama del aprendizaje no supervisado, por lo que no requieren de etiquetas previamente definidas para determinar la pertenencia a los distintos *clusters*.

En esta siguiente etapa del trabajo, se realizó un modelo de *clustering* que permitiera agrupar las muestras de los inversores según su comportamiento, normal o anómalo. En este caso, se utilizaron los datos del parque Myrtle, que presenta información sobre 70 inversores, con un total de 3.023.930 muestras y 41 variables originales a las que se sumaron las 4 variables asociadas con el problema de condensación (127.005.060 valores en total).

Para ello, fue necesario aplicar *Principal Component Analysis* como técnica de reducción de la dimensionalidad para simplificar el problema. Seguidamente, se probaron tres tipos de modelos: *K-Means*, *C-Means* y *Gaussian Mixture Models* (GMM).

En este capítulo se profundizará en los conceptos teóricos de estas técnicas y se mostrarán los resultados, comparativas y conclusiones obtenidas.

5.1. *Principal Component Analysis*

La técnica de PCA es una de las más utilizadas en la literatura para reducción de la dimensionalidad. En este trabajo se decidió emplearla debido al elevado número de variables que presentaban las muestras lo cual provocaba elevados tiempos de computación.

5.1.1. Fundamentos teóricos

El Análisis de Componentes Principales es una técnica estadística de reducción de la dimensionalidad que se utiliza para simplificar conjuntos de datos complejos. El objetivo principal de PCA es transformar un conjunto de variables posiblemente correlacionadas en un conjunto de variables no correlacionadas, denominadas componentes principales, ordenadas de manera que las primeras retengan la mayor parte de la variabilidad presente en los datos originales. De esta forma, se consigue reducir el ruido de los datos y detectar patrones que antes aparecían ocultos en el espacio original.

En términos algebraicos, lo que se pretende es aplicar un cambio de base al *dataset* original de forma que capture la mayor variabilidad posible de los datos, perdiendo la mínima información posible. Los vectores que constituyen esta base serán las llamadas componentes principales, denotadas como Z_i, y se calculan de la siguiente forma:

$$Z_i = \phi_{1i}X_1 + \phi_{2i}X_2 + \cdots + \phi_{ni}X_n \Leftrightarrow Z = \Phi X \tag{5.1}$$

donde $X \in \mathcal{M}_{N\times n}(\mathbb{R})$ representa el *dataset* original, con N muestras y n variables representadas como X_i, con $i = 1, \ldots, n$. Los valores ϕ_{ji}, con $j = 1, \ldots, n$, son los *loadings* de la componente principal i, que cumplen $\sum_{j=1}^{n} \phi_{ji}^2 = 1$. Se denota con $\phi_i = (\phi_{1i}, \ldots, \phi_{ni})$ al vector de *loadings* de la componente principal i. Estos valores representan la aportación de cada variable a la componente principal correspondiente; cuanto mayores sean en valor absoluto, mayor importancia tendrá su variable asociada en el cálculo de esa componente principal.

La primera componente principal, Z_1, representa la dirección en el espacio de características que captura la mayor variabilidad posible. La siguiente componente principal se calculará siguiendo la ecuación 5.1, pero teniendo en cuenta que debe ser una dirección ortogonal a la primera. Gracias a esta condición, se reduce bastante la arbitrariedad en la elección de una nueva componente principal.

¿Cómo se calcula la matriz Φ?

Para poder calcular Φ es importante recordar que se quiere maximizar la variabilidad de los datos, reduciendo la redundancia de los mismos. Esta redundancia se mide a partir de la matriz de covarianza, cuya expresión es la siguiente:

$$C_X = \frac{1}{n} X X^T$$

donde los elementos de la diagonal representan la varianza de cada variable, mientras que el resto de elementos de la matriz representan la covarianza entre variables dos a dos. Por tanto, el objetivo es encontrar una matriz Φ que transforme el *dataset* X en una nueva representación Z, la cual cumpla que su matriz de covarianza C_Z sea diagonal, pues así no hay correlaciones y, por tanto, no hay redundancia en sus variables, y que las variables de este nuevo espacio estén ordenadas de mayor a menor varianza.

Según [52] se deduce que:

$$C_Z = \Phi C_X \Phi^T$$

y como C_X es simétrica, tiene que ser diagonalizable ortogonalmente, es decir:

$$C_X = EDE^T$$

donde D es una matriz diagonal y E la matriz de autovectores de C_X, ordenados según sus autovalores. Escogiendo $\Phi = E$ se obtiene que C_Z es diagonal.

Como los procesos de diagonalización pueden ser bastante costosos computacionalmente, se aplica la descomposición matricial SVD, caracterizada por el siguiente teorema:

Sea una matriz $X \in \mathcal{M}_{n\times p}(\mathbb{R})$, sea $U \in \mathcal{M}_{n\times n}(\mathbb{R})$ ortogonal que representa los autovectores de XX^T, sea $V \in \mathcal{M}_{p\times p}(\mathbb{R})$ ortogonal que representa los autovectores de X^TX, y sea $\Sigma \in \mathcal{M}_{n\times p}(\mathbb{R})$ diagonal que contiene los valores singulares ordenados de forma decreciente. Entonces, la matriz X puede descomponerse de la siguiente forma:

$$X = U\Sigma V^T$$

Finalmente, volviendo a *PCA*, si se considera:

$$Z = \frac{1}{\sqrt{n}} X,$$

se tiene que:

$$Z^T Z = C_X$$

Por tanto, si se calcula la descomposición SVD de Z, la matriz V contiene los autovectores de C_X y, por consiguiente, las columnas de V son las componentes principales de X [53].

5.1.2. Implementación

Como se explicó anteriormente, se trabajó con los datos obtenidos del parque Myrtle. Es importante destacar que, originalmente, estos datos no contenían información sobre las variables de condensación `T_SKIIP_MIN`, `TPOT_CAB_MAX`, `T_LIQ_MIN` y `DELTA_TEMP_AMB` las cuales se quería tener en cuenta para el desarrollo de los modelos, por lo que fue necesario calcularlas siguiendo las fórmulas de la Sección 3.3.1, obteniéndose un total de 45 variables.

Para aplicar la técnica de *PCA* se empleó la función `PCA()` de `Python`, utilizando los argumentos `n_components = 10` y `svd_solver = 'randomized'`. Se escogió un número de componentes igual a 10 para asegurar que el método capturase al menos un 95 % de la varianza, representando así la mayor parte de la información de los datos. No obstante, existen técnicas como las descritas en [54], que permiten determinar de forma más óptima el número de componentes principales.

El parámetro `svd_solver = 'randomized'` indica que se emplea una técnica aproximada basada en proyecciones aleatorias para acelerar el cálculo de las componentes principales. Este método, conocido como *Randomized SVD*, es especialmente eficiente para conjuntos de datos grandes o de alta dimensión, ya que reduce el coste computacional y el uso de memoria, proporcionando resultados muy cercanos a los exactos.

Es relevante destacar que la lectura de los distintos archivos CSV se realizó guardando los conjuntos de datos en `dataframes` de `pandas`. Sin embargo, para poder aplicar el método mediante la función `fit_transform()`, fue necesario convertir estos `dataframes` a objetos de tipo `array`.

Tras realizar esta transformación, se llevó a cabo un estudio sobre las componentes principales obtenidas. Lo primero fue analizar el número de muestras resultantes tras la reducción de dimensionalidad. Se consiguió pasar de 127.005.060 valores a 9.071.790, por lo que la reducción de complejidad fue considerable.

El siguiente aspecto a analizar fue la evolución de la varianza explicada por cada

componente principal. La primera componente principal explicaba el 58 % de la varianza y, si se consideraba la varianza explicada acumulada, con las tres primeras componentes principales se conseguía capturar el 81 % de la variabilidad, siendo un porcentaje bastante elevado. Es por ello que, para el entreno de los siguientes modelos, se decidió considerar únicamente estas tres primeras componentes, pues si se incluían más, el aumento en la varianza explicada no compensaba el incremento de complejidad del problema.

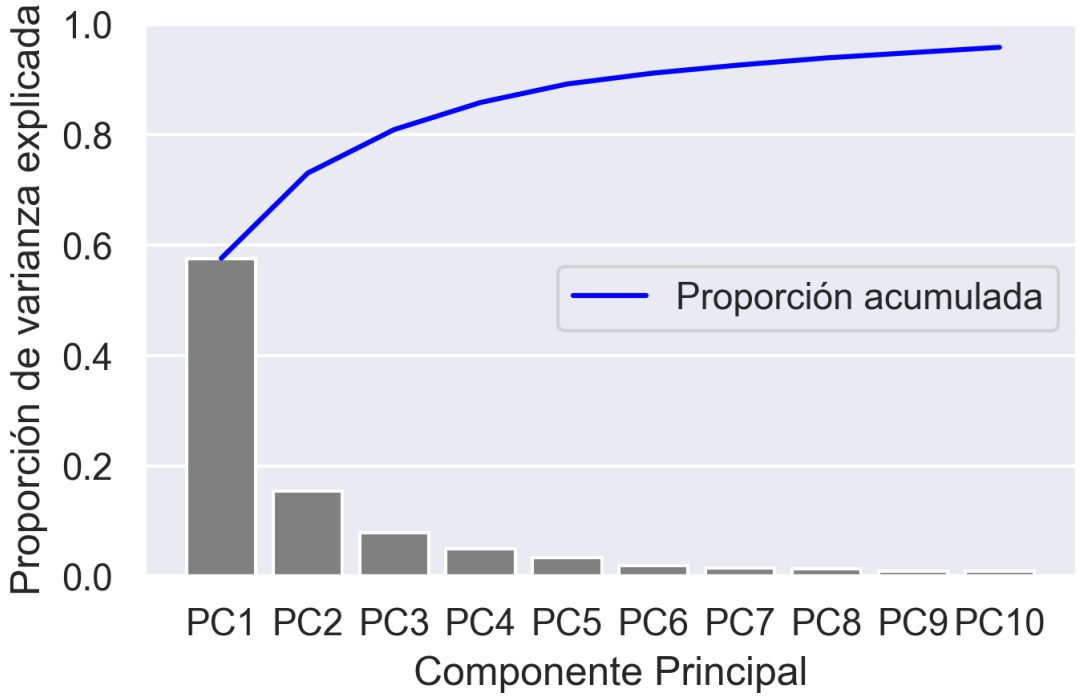

Figura 5.1: Gráfico de barras y líneas que representa la varianza explicada por cada componente principal (barras) y la varianza explicada acumulada (línea).

Seguidamente, se calcularon las 10 variables más importantes de cada una de estas tres componentes principales utilizando los *loadings*. Esto permitió interpretar el significado de cada una de las componentes:

- PC1: Engloba variables asociadas a temperaturas de componentes del inversor por los que pasa la corriente. Las 10 variables más contribuyentes son:

  `[TBRID11, TBRID22, TBRID12, T_SKIIP_MIN, TBRID23, TBRID13, TBRID21, TCOILIND2, TCOILIND1, TCOOLINGO]`.

- PC2: Formada principalmente por variables que representan temperaturas ambientales. Se obtuvieron las siguientes variables como las 10 más importantes:

  `[TAIR_INLET, TSOC, TCAB_POT1, TCCU, TDC_FUSES, TPOT_CAB_MAX, TEXTERIOR, TCAB_POT2, TCAB_AC, DELTA_TEMP_AMB]`.

- PC3: Compuesta en gran parte por variables asociadas a aspectos eléctricos. Las siguientes variables son las que más aportaron en el cálculo de esta componentes principal:

  `[UL1_RMS, PPC_QMG, QMG_M, UL3_RMS, UDC2_FIL, UDC1_FIL, UL2_RMS, VPV1_FIL, VPV2_FIL, THFILTER]`.

En relación con esta última, al personal técnico de la empresa le llamó la atención que las variables asociadas con la corriente (del tipo `IT`) no fueran de las que más contribuyen en PC3. Esto podría deberse a relaciones no lineales que no pueden ser detectadas mediante PCA por lo que en un futuro resultaría interesante aplicar técnicas de reducción de dimensionalidad no lineales. En cuanto a las otras dos componentes, como se verá en el apartado de clasificación, son las que mejor capturan los aspectos clave de la condensación de los inversores.

Después, se graficó la matriz de correlación de estas 10 variables correspondientes a cada componente principal. En dicha matriz, se observa que las variables asociadas a la primera componente principal (*PCA1*) son las que presentan mayor correlación entre ellas. Esto se debe a que las variables con mayor correlación tienden a explicar gran parte de la varianza de los datos. A medida que se consideran componentes de orden superior, las correlaciones entre las variables disminuyen.

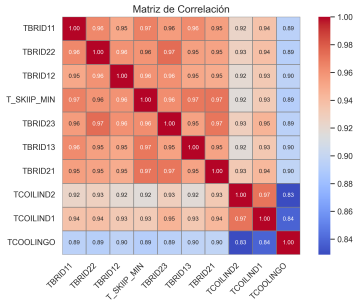

Figura 5.2: *Heatmap* que representa las correlaciones de las 10 variables más contribuyentes en la primera componente principal.

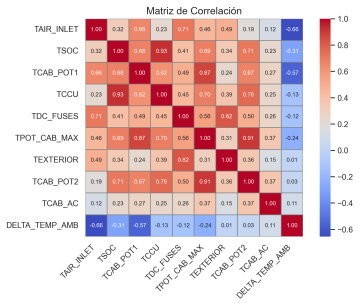

Figura 5.3: *Heatmap* que representa las correlaciones de las 10 variables más contribuyentes en la segunda componente principal.

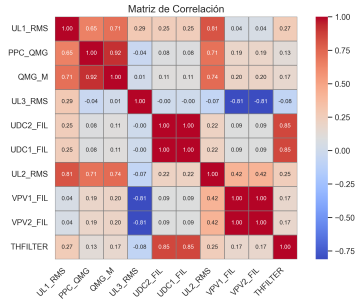

Figura 5.4: *Heatmap* que representa las correlaciones de las 10 variables más contribuyentes en la tercera componente principal.

Por último, se volvieron a analizar las evoluciones temporales (en especial del día en el que sucedió la explosión tras congelar, 13-02-2025) de las variables más importantes de las componentes principales para ver si en estas, los inversores normales y con fallo tenían diferencias de comportamiento. Efectivamente, en la mayoría de ellas el inversor que experimentó el fallo tenía un comportamiento distinto por lo que PCA está centrando su atención en las variables más informativas en situaciones de fallo.

5.2. *K-Means*

La técnica de *clustering K-Means* es de la familia *center-based*. En este tipo de modelos, cada *cluster* tiene un miembro representativo que puede ser de dos tipos:

- Centroide: La media de todas las muestras que forman el *cluster*. Este punto no tiene por qué ser un punto dentro del *dataset*.
- Medoide: Es el punto dentro del cluster cuya distancia total a todos los demás puntos del *cluster* es mínima.

En el caso de uso del trabajo, utilizaremos centroides como elementos representativos de los *clusters* debido al carácter numérico de los datos.

5.2.1. Fundamentos teóricos

Para ofrecer una descripción más detallada desde el punto de vista matemático del algoritmo de *K-Means*, se ha seguido la referencia [55].

En este modelo de *clustering*, el número de grupos que se quiere obtener (K) es definido previamente por el usuario. Existen varias variaciones de este algoritmo, pero en este trabajo se utilizó la versión clásica, cuyos pasos principales son los siguientes:

1. *Seleccionar K puntos del dataset original que serán los centroides iniciales.*
2. *Repetir hasta que los centroides no cambien:*

   a) *Crear K grupos asignando cada punto (muestra) al centroide más cercano.*
   b) *Recalcular el centroide para cada cluster.*

3. *Finalizar el algoritmo cuando los centroides se estabilicen.*

Cabe destacar que la condición de parada es bastante restrictiva y no todos los conjuntos de datos llegan necesariamente a converger por completo. Por ello, en ocasiones se considera como condición de parada que el porcentaje de puntos que cambian de *cluster* entre iteraciones sea inferior a un umbral muy reducido.

Para poder asignar cada punto a un *cluster*, es necesario definir una medida de distancia. Existen muchas opciones (Manhattan, coseno, entre otras), dependiendo de la naturaleza de los datos. La más utilizada es la distancia euclídea (L2), que es la empleada en este trabajo y que se define como:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{5.2}$$

Una vez escogida la distancia, el siguiente paso es definir una función objetivo que permita medir la calidad del *clustering* obtenido. En los casos en los que se utiliza la distancia L2, la función objetivo a minimizar es la suma de errores cuadráticos (SSE, por sus siglas en inglés):

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} d(c_i, x)^2 \tag{5.3}$$

donde K es el número de *clusters*, C_i es el conjunto de puntos que pertenecen al *cluster* i, c_i es el centroide del *cluster* i y d es la distancia L2. Este valor mide la compacidad de los *clusters*: cuanto menor sea, más compactos serán los *clusters* y mejor será la agrupación obtenida.

Por último, falta determinar cómo recalcular los centroides. Se puede comprobar que, al considerar la SSE, la elección de centroides que minimiza este valor es la media de los puntos del *cluster*. Por ello, la fórmula para calcular el centroide del *cluster* i es:

$$c_i = \frac{1}{m_i} \sum_{x \in C_i} x \tag{5.4}$$

donde m_i es el número de puntos del *cluster* i.

Una vez introducidos los conceptos teóricos principales del algoritmo, se procede a explicar el código utilizado para el desarrollo de este modelo.

5.2.2. Implementación

En el trabajo se utilizó el método `KMeans()` de la librería `scikit-learn` [56]. Se escogieron los siguientes parámetros: `k=2`, `random_state=10` y `n_init=10`.

El primero hace referencia al número de *clusters*. Como el objetivo era realizar un *clustering* para distinguir entre comportamiento anómalo y normal del inversor, se tenía claro que el número de *clusters* debía ser 2. No obstante, existen técnicas, como el cálculo de SSE o el coeficiente de Silhouette (definido más adelante), que permiten determinar el número óptimo de *clusters* aplicando el método del codo.

El segundo parámetro sirve para fijar la semilla y poder reproducir la aleatoriedad en la elección inicial de los K centroides. El último hace referencia al número de veces que el algoritmo se ejecutará con distintas inicializaciones de los centroides; de todas estas ejecuciones se escogerá aquella que obtenga el menor SSE. Gracias a este último parámetro se consigue evitar que el modelo caiga en soluciones locales, además de aportar una mayor robustez.

Una vez determinado el objeto `KMeans()`, este se ajustará a las muestras de los inversores del parque Myrtle transformadas a partir de PCA (considerando únicamente las tres primeras). A continuación, se muestran los dos *clusters* obtenidos proyectándolos en el espacio de las tres primeras componentes principales de PCA:

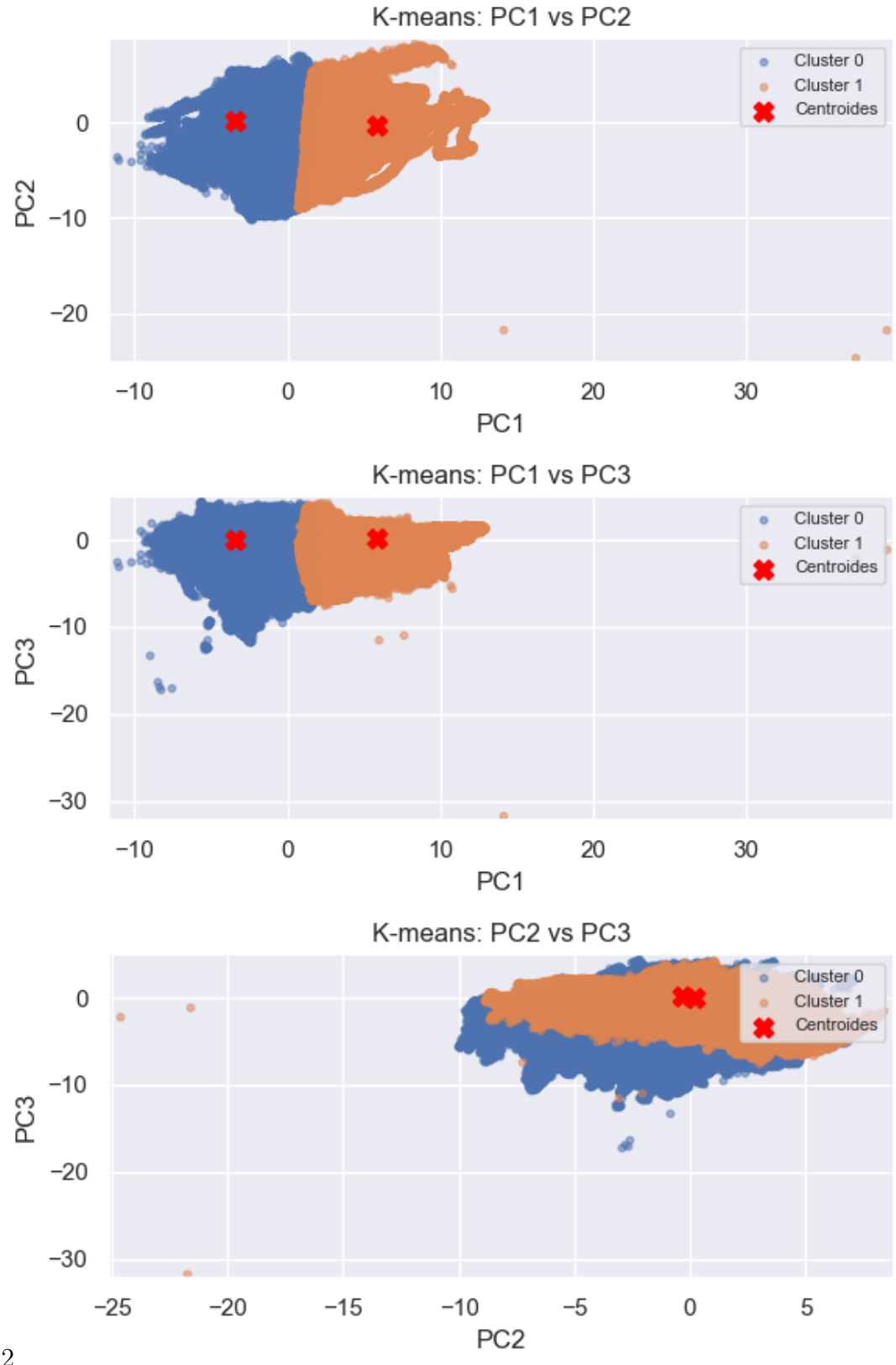

Figura 5.5: Gráficos con los dos *clusters* obtenidos con el algoritmo de *K-Means*. Estos grupos se han proyectado en el espacio de las tres primeras componentes principales del algoritmo de PCA. En la imagen se muestran los centroides de ambos *clusters*.

La separación de los *clusters* es clara en los planos PC1-PC2 y PC1-PC3, dado que la primera componente principal recoge la mayor parte de la varianza de los datos y aporta, por tanto, la mayor capacidad de diferenciación entre las muestras. Sin embargo, al proyectar sobre el plano PC2-PC3, donde no participa PC1, la separación entre *clusters* se reduce y aparecen solapamientos, ya que estas componentes representan una menor variabilidad de los datos. De hecho, esta superposición puede observarse también en la gran proximidad de los centroides de los *clusters*. En cuanto a los *outliers*, se observa que los datos están muy compactos, con muy pocos valores atípicos.

Para poder estudiar la calidad del *clustering* obtenido, se calculó el coeficiente de Silhouette [57]. Su fórmula es la siguiente para un punto i del *dataset*:

$$s(i) = \frac{b(i) - a(i)}{\text{máx}\{a(i), b(i)\}}$$

donde $a(i)$ es la distancia media entre el punto i y todos los demás puntos del mismo *cluster*, y $b(i)$ es la distancia media mínima entre el punto i y todos los puntos del *cluster* más cercano al que no pertenece. Este coeficiente aporta información sobre la cohesión dentro de los *clusters* ($a(i)$) y la separación entre *clusters* ($b(i)$).

De la fórmula se deduce que $s(i) \in [-1, 1]$. Las situaciones en las que $s(i)$ es negativo se deben a que $a(i) > b(i)$, lo que implica que el punto i está más próximo a otros *clusters* que al *cluster* al que ha sido asignado (mala agrupación de los puntos). Por otra parte, cuanto mayor sea $s(i)$, mejor será la calidad del *clustering*, pues los elementos de un grupo estarán muy próximos entre sí y los *clusters* estarán más separados entre ellos.

El coeficiente de Silhouette de un *clustering* se obtiene como la media de los $s(i)$ para todos los puntos i del *dataset*. Debido al elevado número de muestras de inversores, se aplicó un cálculo simplificado de este coeficiente: en lugar de calcularlo para todos los puntos, se seleccionaron 1.000 puntos al azar. El resultado fue un coeficiente de Silhouette de 0,59, lo que indica que la calidad de la agrupación es razonablemente buena.

A continuación, se estudiaron en mayor detalle cada uno de los dos grupos obtenidos. En primer lugar, se graficaron los valores de los centroides de los *clusters* para obtener una idea del valor medio de cada componente principal en cada grupo.

Figura 5.6: Gráfico de barras con el valor de los dos centroides obtenidos con *K-Means* para la primera componente principal.

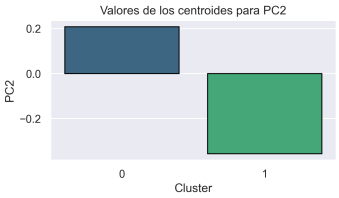

Figura 5.7: Gráfico de barras con el valor de los dos centroides obtenidos con *K-Means* para la segunda componente principal.

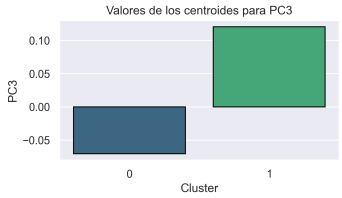

Figura 5.8: Gráfico de barras con el valor de los dos centroides obtenidos con *K-Means* para la tercera componente principal.

Gracias a estos gráficos, se puede concluir que los puntos que pertenecen al *cluster* 0 son aquellos que presentan valores negativos para PC1 y PC3, pero positivos para PC2. Por el contrario, los puntos del *cluster* 1 muestran valores positivos para PC1 y PC3, y negativos para PC2.

Seguidamente, se estudió el número de puntos pertenecientes a cada grupo. El *cluster* etiquetado como 0 (azul) contiene un total de 1.907.212 puntos, mientras que el *cluster* 1 agrupa 1.116.718 puntos. Es razonable pensar que lo más habitual es que los inversores no fallen, por lo que la mayoría de las muestras representarán un comportamiento normal de los equipos. Por ello, se planteó la hipótesis de que el *cluster* azul podría representar el comportamiento normal de los inversores.

Para comprobar esta hipótesis, se decidió aplicar el modelo de *K-Means* sobre dos inversores: uno con comportamiento a priori normal y otro etiquetado con un fallo. El inversor de comportamiento normal seleccionado fue el número 34 para el día 13-02-2025, mientras que el inversor con fallo fue el número 11 para ese mismo día, del cual se sabe que explotó tras congelarse. Además, el hecho de que ambos correspondan al mismo día permite que los resultados no se vean afectados por diferencias en las condiciones ambientales.

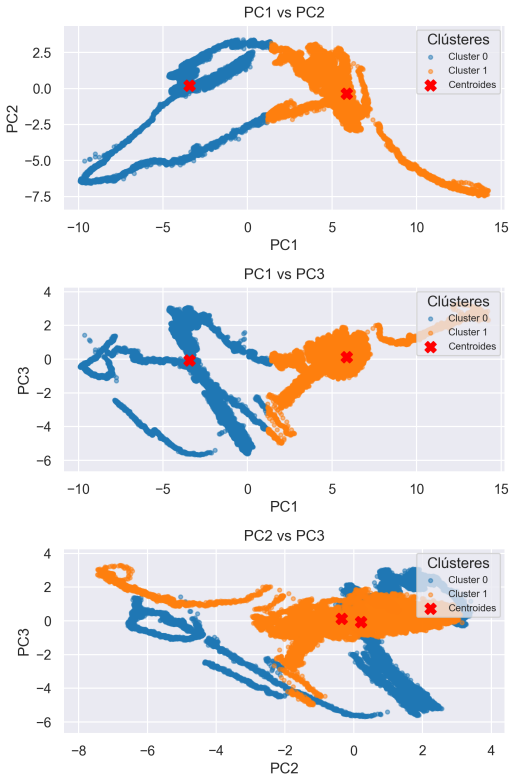

Figura 5.9: Distribución de los dos *clusters* obtenidos mediante *K-Means* usando los datos del inversor 34 el día 13-02-25. Este inversor presenta un comportamiento normal.

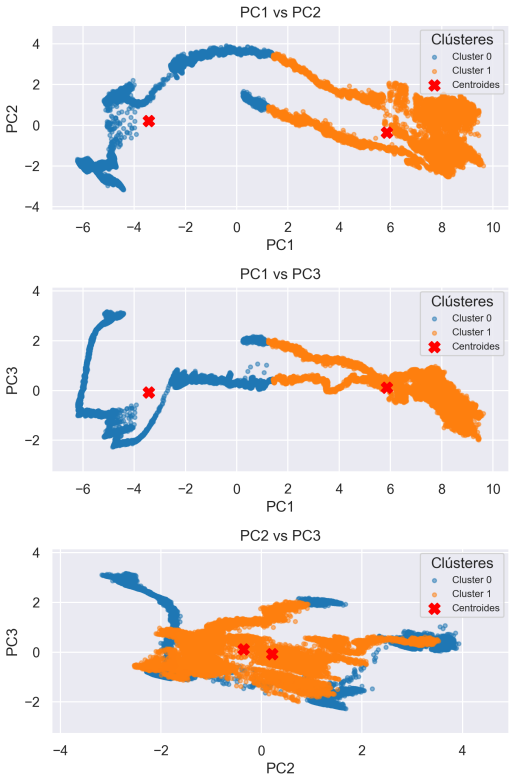

Figura 5.10: Distribución de los dos *clusters* obtenidos mediante *K-Means* usando los datos del inversor 11 el día 13-02-25. En ese día, se registró una explosión tras congelar del equipo.

En la distribución asociada al inversor 11, se observa que el 65.56 % de los datos pertenecen al *cluster* azul, mientras que para el inversor 34 este porcentaje es del 61.67 %. De este modo, no se puede afirmar con absoluta certeza que el grupo azul represente muestras de comportamiento normal. No obstante, podría ocurrir que los registros asociados al inversor 11 correspondan en su mayoría a un comportamiento previo al fallo, o bien que el inversor 34 haya presentado algún tipo de fallo no detectado, lo que podría explicar una mayor proporción de puntos en el *cluster* 1 (comportamiento fallido) para las muestras del inversor 34.

Aunque los resultados obtenidos son aceptables, se decidió probar con otros algoritmos de *clustering* con el objetivo de intentar mejorar estos resultados.

5.3. *Fuzzy C-Means*

El modelo que se va a probar a continuación es bastante similar a *K-Means*. Ambos pertenecen a la familia de modelos *center-based*. No obstante, en *K-Means* se realiza una partición dura del conjunto de datos, es decir, un punto solo puede pertenecer a un único *cluster*. Sin embargo, en *Fuzzy C-Means* (FCM) un punto puede pertenecer a varios *clusters* con distintos grados de pertenencia.

Este nuevo enfoque resulta especialmente interesante para el problema de clasificación del comportamiento de los inversores, ya que permite cuantificar la normalidad de una muestra. Supongamos que se dispone de un registro de las variables de un inversor durante un día en el que ha fallado y que dicho fallo ocurrió por la tarde. Durante la mañana, las muestras del inversor presentarán un comportamiento normal, ya que el fallo aún no se ha producido. No obstante, a medida que se aproxima la tarde, las muestras irán adquiriendo características propias de un comportamiento defectuoso.

En las siguientes secciones se explicará con mayor detalle el algoritmo de *Fuzzy C-Means*, incluyendo aspectos como la función objetivo o el cálculo de los centroides, para posteriormente describir cómo se ha implementado en `Python` y los resultados obtenidos con las muestras de los inversores del parque Myrtle. Para la parte teórica, se ha seguido principalmente como referencia [58].

5.3.1. Fundamentos teóricos

El objetivo del algoritmo de FCM es realizar una partición suave del conjunto de datos. Matemáticamente, este tipo de particiones se definen como:

Sea X un conjunto de datos y x_i un punto perteneciente a X. Se dice que una partición $P = \{C_1, \ldots, C_K\}$ es una partición suave de X si, y solo si, se cumplen las siguientes condiciones:

1. $\forall x_i \in X,\ \forall C_j \in P:\ 0 \leq \mu_{C_j}(x_i) \leq 1$
2. $\forall x_i \in X,\ \exists C_j \in P:\ \mu_{C_j}(x_i) > 0$

donde $\mu_{C_j}(x_i)$ denota el grado con el cual x_i pertenece al cluster C_j.

Se dice que una partición suave de X es restringida si además de las condiciones anteriores cumple también:

$$\sum_{j=1}^{K} \mu_{C_j}(x_i) = 1 \quad \forall x_i \in X$$

El algoritmo FCM realiza una partición suave restringida del *dataset*. Los pasos del algoritmo son los mismos que en el caso de *K-Means*, pero la función objetivo y el cálculo de los centroides varían ligeramente.

La función objetivo es similar a 5.3, pero añadiendo los grados de pertenencia difusos (μ_{C_j}) y un nuevo parámetro m, conocido como *peso exponente en la función de pertenencia*, que controla el grado de superposición entre *clusters*. Así, la función objetivo de FCM es el siguiente:

$$J_m(P, V) = \sum_{i=1}^{K} \sum_{x \in X} (\mu_{C_i}(x))^m \, \|x - v_i\|^2$$

donde $P = \{C_1, \ldots, C_K\}$ es una partición difusa de X, V el conjunto de centroides y v_i el centroide del *cluster* i. Esta función penaliza los puntos con alto grado de pertenencia a un *cluster* que estén alejados de su centroide.

Al igual que *K-Means*, FCM debe encontrar los centroides v_i y, además, las funciones de pertenencia μ_{C_j} que minimicen J_m. Estas condiciones se garantizan gracias al *Teorema de Fuzzy C-Means*, cuyo enunciado es:

Una partición difusa $P = \{C_1, \ldots, C_K\}$ puede ser un mínimo local de la función objetivo J_m solo si se cumplen las siguientes condiciones:

1. $$\mu_{C_i}(x) = \frac{1}{\sum_{j=1}^{K} \left(\frac{\|x - v_i\|^2}{\|x - v_j\|^2} \right)^{\frac{1}{m-1}}} \quad \forall i = 1, \ldots, K; \ \forall x \in X$$

2. $$v_i = \frac{\sum_{x \in X} \mu_{C_i}(x)^m x}{\sum_{x \in X} \mu_{C_i}(x)^m} \quad \forall i = 1, \ldots, K$$

Una vez determinado el número de *clusters* (K), el valor de m y el criterio de parada, el algoritmo FCM realiza dos pasos en cada iteración: primero calcula

las funciones de pertenencia mediante la primera ecuación del teorema anterior; después actualiza los centroides usando la segunda ecuación. Estos pasos se repiten hasta que se alcanza el criterio de parada.

5.3.2. Implementación

Para el desarrollo del código de este modelo se utilizó la librería `Scikit-Fuzzy (skfuzzy)`, un paquete especializado en lógica difusa y *clustering* difuso de `Python`.

El modelo FCM se entrenó utilizando la representación de los datos del parque Myrtle proyectados sobre las tres primeras componentes principales obtenidas mediante PCA. Es importante matizar que fue necesario transponer el *array* de `numpy` para poder utilizar la función `fuzz.cluster.cmeans` [59]. Este método implementa el algoritmo FCM y se configuró con los siguientes parámetros:

- `X_pca_3_T`: Datos del parque Myrtle tras aplicar PCA con tres componentes principales, transpuestos para cumplir con la entrada esperada.
- `c = n_clusters`: Número de *clusters* en los que dividir los datos. Al igual que en *K-Means*, se fijó `n_clusters = 2`.
- `m = 2.0`: Coeficiente de difusividad. Se escogió este valor porque es el habitual en la literatura, ya que equilibra la difusividad y la separación entre *clusters*.
- `maxiter = 1000`: Número máximo de iteraciones del algoritmo, que actúa como criterio de parada si no se alcanza convergencia antes.
- `init = None`: Inicialización automática de los grados de pertenencia.
- `seed = 42`: Semilla para garantizar la reproducibilidad.

Este método devuelve los siguientes elementos:

- `cntr`: Matriz con los centroides de los *clusters* difusos, de tamaño (*n_clusters*, *n_features*) (en este caso, *n_features* = 3). Es un `numpy.ndarray`.
- `u`: Matriz final con los grados de pertenencia, de tamaño (*n_clusters*, *num_muestras*) (en este caso, *num_muestras* = 3.023.930). Es un `numpy.ndarray`.

- `u0`: Matriz de pertenencias iniciales (antes de iterar), con las mismas dimensiones que `u` y también un `numpy.ndarray`.

- `d`: Matriz con las distancias de cada punto a cada centroide, de tamaño ($n\_clusters$, $num\_muestras$). Es un `numpy.ndarray`.

- `jm`: Función objetivo (J_m) evaluada en cada iteración, devuelta como una lista de `float` con longitud igual al número de iteraciones realizadas.

- `p`: Número de iteraciones realizadas hasta convergencia, de tipo `int`.

- `fpc`: Fuzzy Partition Coefficient, un valor entre $\frac{1}{n\_clusters}$ y 1 que mide la claridad de la partición. Cuanto más próximo a 1, mejor la partición, ya que indica asignaciones más definidas. Se calcula como:

$$\text{FPC} = \frac{1}{N} \sum_{i=1}^{K} \sum_{j=1}^{N} \mu_{ij}^2$$

  donde N es el número de muestras y K el número de *clusters*.

Una vez obtenido el *clustering* difuso, se evaluó su calidad mediante el FPC. El modelo FCM para el comportamiento de los inversores con dos *clusters* obtuvo un FPC de 0,84, valor bastante cercano a 1, lo que indica que la mayoría de los puntos tiene alta pertenencia a un único *cluster*, reflejando una asignación clara. Para visualizar cómo se distribuyen los grados de pertenencia de los distintos puntos a los *clusters*, se elaboró el siguiente gráfico:

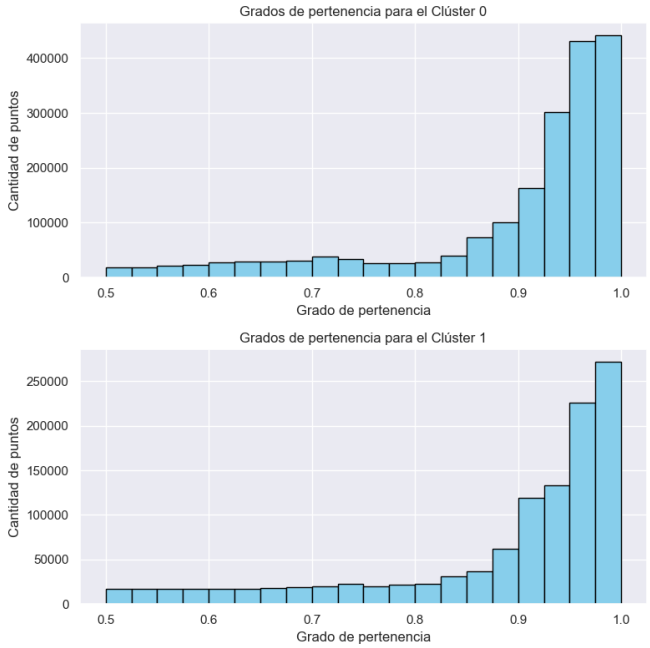

Figura 5.11: Gráficos de barras en los que puede verse para cada *cluster* del algoritmo FCM, el número de puntos que hay para cada rango de grados de pertenencia. Se observa que para ambos *clusters* la gran cantidad de puntos tienen elevados grados de pertenencia.

Se puede observar lo comentado anteriormente: hay pocos puntos con un grado de pertenencia cercano a 0,5, que serían aquellos sobre los que el modelo presenta mayor incertidumbre respecto a la asignación a un *cluster*, de forma que la pertenencia de los puntos a los *clusters* es bastante clara.

Para facilitar la visualización de los *clusters*, se asignó a cada punto el *cluster* correspondiente al mayor grado de pertenencia. Al hacerlo, el modelo FCM adopta un enfoque muy similar al de *K-Means*. Los *clusters* resultantes son los siguientes, donde se observa que el *clustering* es prácticamente igual al obtenido con *K-Means*, lo que resulta coherente dado que el algoritmo FCM produjo una partición con muy poca ambigüedad en la asignación de puntos.

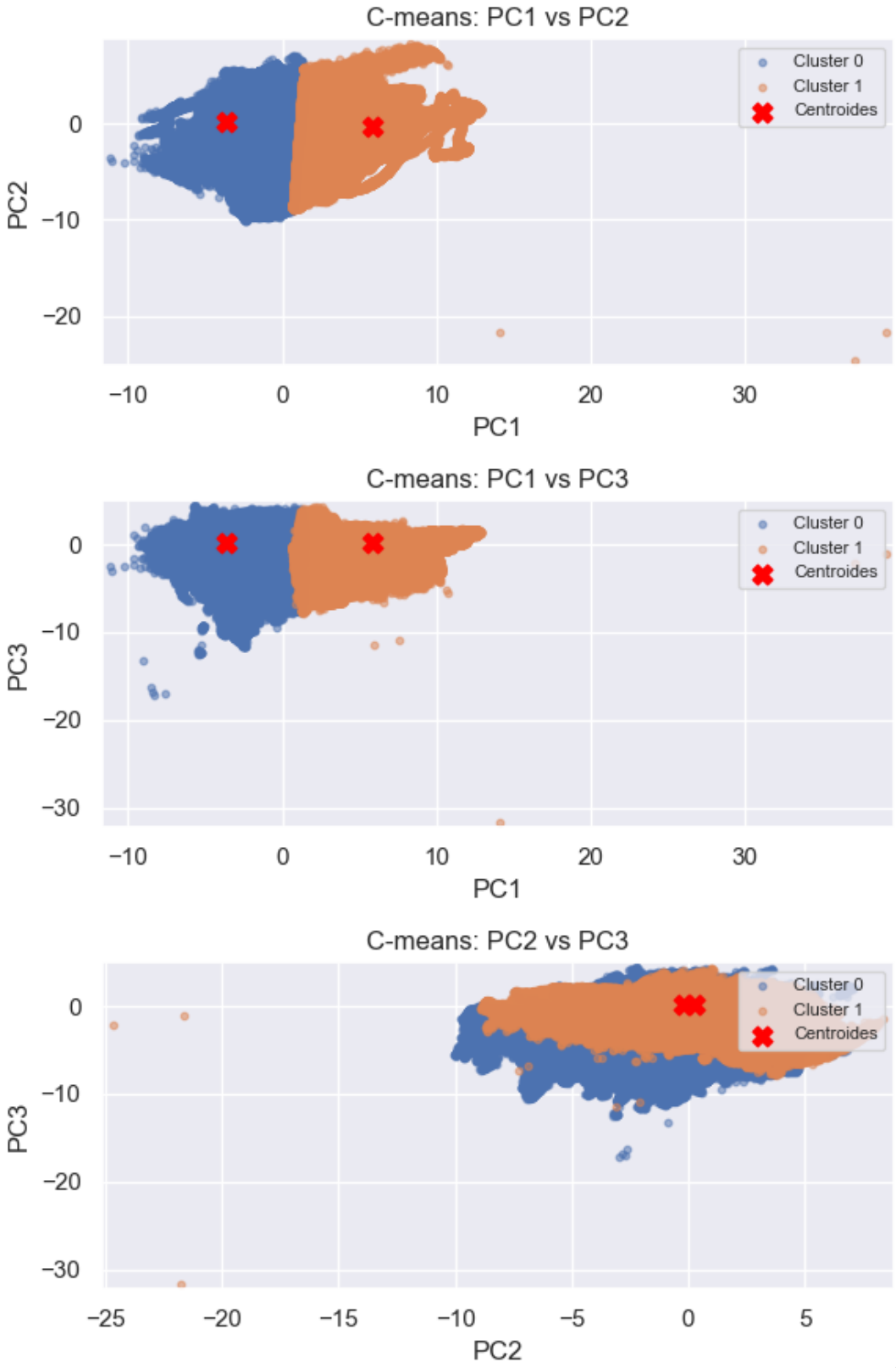

Figura 5.12: Gráficos con los dos *clusters* obtenidos con el algoritmo de FCM. Estos grupos se han proyectado en el espacio de las tres primeras componentes principales del algoritmo de PCA. En la imagen se muestran los centroides de ambos *clusters*.

Asimismo, se estudió el porcentaje de puntos que pertenecen a cada *cluster*. El 62.77 % correspondían al *cluster* 0 (azul) y el 37.23 % al *cluster* 1 (naranja). Nuevamente, se observa que el *cluster* 0 concentra la mayor cantidad de puntos, por lo que podría representar el comportamiento normal de los inversores.

Seguidamente, se analizó el valor de los centroides siguiendo un enfoque similar al realizado con *K-Means*. Se obtuvieron resultados parecidos para PC1 y PC2, pero se observó un ligero cambio en PC3. En este algoritmo, los puntos pertenecientes al *cluster* azul presentan un valor elevado para PC3, mientras que los del otro *cluster*, aunque también con valores positivos para esta componente, muestran valores menores. Por tanto, es en esta componente donde existe cierta diferencia entre el *clustering* obtenido con *K-Means* y con FCM. Esto se debe a que el primer algoritmo realiza una partición dura, por lo que es menos sensible a las variaciones aportadas por la componente que captura menor varianza (PC3). Por su parte, FCM, gracias a su asignación difusa, capta mejor las variaciones aportadas por PC3.

Figura 5.13: Gráfico de barras con el valor de los dos centroides obtenidos con FCM para la primera componente principal.

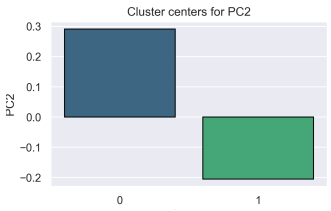

Figura 5.14: Gráfico de barras con el valor de los dos centroides obtenidos con FCM para la segunda componente principal.

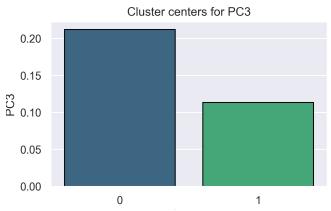

Figura 5.15: Gráfico de barras con el valor de los dos centroides obtenidos con FCM para la tercera componente principal.

Por último, se probó este *clustering* con las muestras de los inversores 11 y 34 del día 13-02-25 para comprobar si el *cluster* azul podría estar relacionado con el comportamiento normal de los equipos. Como resultado, se obtuvo que un 65.46 % de los puntos del inversor 11 pertenecían al *cluster* azul, frente a un 61.47 % de las muestras del inversor 34. Estos porcentajes son bastante similares a los obtenidos con *K-Means*, llegando a la misma conclusión.

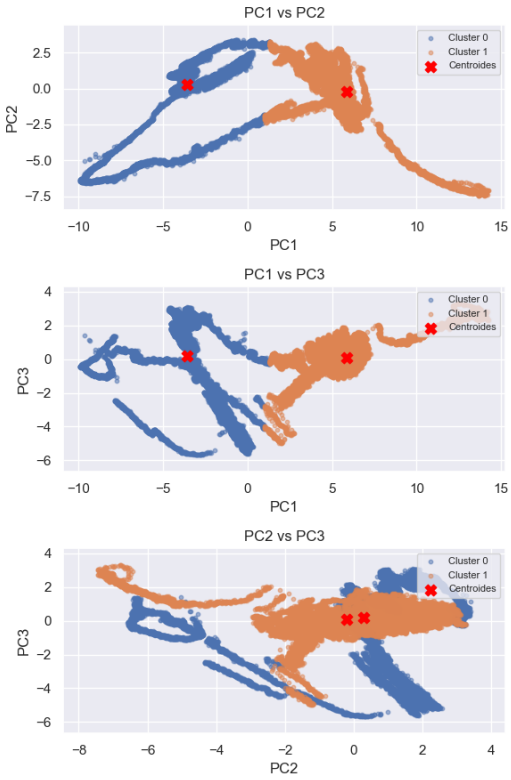

Figura 5.16: Distribución de los dos *clusters* obtenidos mediante FCM usando los datos del inversor 34 el día 13-02-25. Este inversor presenta un comportamiento normal.

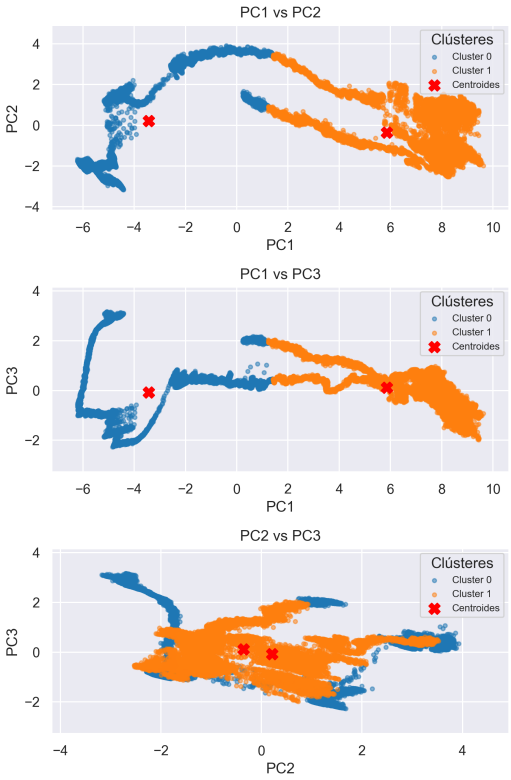

Figura 5.17: Distribución de los dos *clusters* obtenidos mediante FCM usando los datos del inversor 11 el día 13-02-25. En ese día, se registró una explosión tras congelar del equipo.

Dado que los resultados obtenidos con FCM fueron bastante similares a los obtenidos con *K-Means*, se optó por probar un algoritmo de *clustering* con un enfoque distinto: *Gaussian Mixture Models*. Este nuevo método permitirá una mayor flexibilidad en la forma de los *clusters*, pudiendo capturar mejor los patrones de los datos.

5.4. *Gaussian Mixture Models*

En las técnicas de *clustering* vistas anteriormente, la asignación de los puntos a los *clusters* se realizaba en función de la distancia, de manera que cada punto se asignaba al *cluster* cuyo centroide estaba más próximo. En cambio, el *Gaussian Mixture Model* (GMM) asigna los puntos a los *clusters* en base a probabilidades:

estima la probabilidad de que cada punto pertenezca a cada uno de los *clusters*.

El objetivo de GMM es representar los datos como una combinación de varias distribuciones gaussianas, donde cada gaussiana modela un *cluster* con su propia media y matriz de covarianza. En esta sección se procederá a introducir los conceptos técnicos de GMM y se explicará la implementación de esta técnica en `Python` para el caso de diferenciación en el comportamiento de los inversores, así como las conclusiones obtenidas con este último modelo. Se quiere destacar que, para la parte teórica del algoritmo, se siguió principalmente la referencia [60].

5.4.1. Fundamentos teóricos

Se empezará este apartado teórico recordando la definición de distribución normal (gaussiana) debido a su papel principal en esta sección.

Sea X una variable aleatoria de media μ y varianza σ^2, la función de densidad de la distribución normal viene dada por:

$$\mathcal{N}(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

A raíz de esta definición de distribución normal se puede introducir el concepto de distribución de mezclas gaussianas.

Una distribución de mezclas gaussianas (Gaussian Mixture Distribution) es una superposición lineal de K distribuciones gaussianas definida como:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k),$$

donde $x \in \mathbb{R}^d$, $d \in \mathbb{Z}$, π_k son los pesos de cada componente (o coeficientes de mezcla) cumpliendo $\pi_k \geq 0$ y $\sum_{k=1}^{K} \pi_k = 1$, y Σ_k es la matriz de covarianza de la gaussiana k.

El objetivo es determinar qué distribución gaussiana ha generado cada uno de los datos. Para ello, se considera una variable aleatoria Z, K-dimensional, que toma valores binarios tal que $\sum_{k=1}^{K} z_k = 1$. La interpretación es que si $z_k = 1$, entonces los datos han sido generados por la componente gaussiana k.

La distribución marginal de esta variable Z es:

$$p(z_k = 1) = \pi_k,$$

por lo que π_k indica la probabilidad de que un punto haya sido generado por la k-ésima distribución gaussiana.

Asimismo, relacionada con esta variable se encuentra otra cantidad fundamental en el algoritmo, la probabilidad condicional de Z dado X, que se expresa como:

$$\gamma(z_k) = P(z_k = 1 \mid x, \Theta) = \frac{\pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x \mid \mu_j, \Sigma_j)}, \tag{5.5}$$

donde esta cantidad representa la probabilidad posterior de que la observación x haya sido generada por la componente gaussiana k, dado el conjunto de parámetros actuales del modelo $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K}$.

Se busca encontrar los parámetros π_k, μ_k y Σ_k que mejor representen los datos. Para evaluar qué tan bien estos parámetros se ajustan a los datos, se utiliza la función de verosimilitud definida como:

$$L(\Theta) = p(X \mid \Theta) = \prod_{i=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k),$$

donde N es el número de muestras. No obstante, calcular un producto es costoso computacionalmente, por lo que se calcula la log-verosimilitud:

$$\ell(\Theta) = \ln L(\Theta) = \sum_{i=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k) \right). \tag{5.6}$$

Aunque esta medida reduce la complejidad del problema, sigue siendo difícil optimizarla directamente. En la literatura existen varias técnicas para optimizar el cálculo de los parámetros de GMM. En este trabajo se presenta el algoritmo clásico de *Expectation-Maximization* (EM) [60], aunque pueden encontrarse nuevas versiones de este algoritmo [61].

Algoritmo *Expectation-Maximization*

Dado un modelo de mezcla gaussiana, el objetivo es maximizar la función de verosimilitud respecto a los parámetros entre los que se incluyen las medias y covarianzas de los componentes, así como los pesos de cada componente. Los principales pasos del algoritmo de *Expectation-Maximization* son:

1. Inicializar las medias μ_k, las matrices de covarianza Σ_k y los coeficientes de mezcla π_k, y evaluar el valor inicial de la log-verosimilitud.
2. Paso E (*Expectation*): calcular las responsabilidades ($\gamma(z_k)$) utilizando los valores actuales de los parámetros mediante la ecuación 5.5.
3. Paso M (*Maximization*): recalcular los parámetros usando las responsabilidades actuales y las fórmulas siguientes:

$$\mu_k^{\text{nuevo}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})\, x_n$$

$$\Sigma_k^{\text{nuevo}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})\, (x_n - \mu_k^{\text{nuevo}})(x_n - \mu_k^{\text{nuevo}})^\top$$

$$\pi_k^{\text{nuevo}} = \frac{N_k}{N}$$

   donde $N_k = \sum_{n=1}^{N} \gamma(z_{nk})$ y N el número total de muestras.
4. Evaluar la log-verosimilitud con la fórmula 5.6 y ver si hay convergencia de los parámetros o en la log-verosimilitud. Si no se satisface el criterio de convergencia, se vuelve al paso 2.

5.4.2. Implementación

Como en el resto de modelos, GMM se ha aplicado sobre las muestras de los inversores del parque Myrtle en su representación mediante las tres primeras componentes principales de PCA. La clase de `Python` que se ha utilizado ha sido `GaussianMixture()` de la librería `sklearn` [62].

Entre los parámetros disponibles para esta clase se encuentra `covariance_type`. Se encarga de especificar la forma de la matriz de covarianza de cada componente

de la mezcla gaussiana por lo que está directamente relacionado con la forma de los *clusters*. Puede tomar los siguientes valores:

- `full`: Cada componente tiene su propia matriz de covarianza. En este caso, los *clusters* pueden ser elipses de distintos tamaños y orientadas sobre distintos ejes. Matemáticamente, la matriz de covarianza tiene la siguiente expresión:

$$\Sigma_k = \begin{pmatrix} \sigma_{11}^{(k)} & \sigma_{12}^{(k)} & \cdots & \sigma_{1d}^{(k)} \\ \sigma_{21}^{(k)} & \sigma_{22}^{(k)} & \cdots & \sigma_{2d}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{d1}^{(k)} & \sigma_{d2}^{(k)} & \cdots & \sigma_{dd}^{(k)} \end{pmatrix}$$

- `tied`: Todas las componentes comparten la misma matriz de covarianza. En este caso, los *clusters* serán de la misma forma elíptica pero pueden estar rotadas:

$$\Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_{dd} \end{pmatrix}$$

- `diag`: Cada componente tiene su propia matriz de covarianza diagonal. Los *clusters* son elipses alineadas con los ejes pero de distintas anchuras:

$$\Sigma_k = \begin{pmatrix} \sigma_1^{2(k)} & 0 & \cdots & 0 \\ 0 & \sigma_2^{2(k)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_d^{2(k)} \end{pmatrix}$$

- `spherical`: Cada componente tiene su propia varianza escalar. En este último caso, los *clusters* son esferas pero pueden variar de tamaño de unas componentes a otras:

$$\Sigma_k = \sigma_k^2 I_d \quad \text{donde} \quad I_d = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

Dado que este parámetro debe especificarse antes del entrenamiento, se decidió ajustar un modelo GMM con dos gaussianas para cada uno de los tipos de matriz

de covarianza y calcular el índice BIC correspondiente. El objetivo era determinar qué elección de matriz de covarianza es la más adecuada para los datos.

Para el entrenamiento de los cuatro modelos, además del parámetro `covariance_type`, se utilizaron los parámetros `n_components = 2` y `random_state = 10`. El cálculo del BIC se realizó mediante el método `bic`, que forma parte de la clase `GaussianMixture()`. Este método aplica la siguiente fórmula:

$$\text{BIC} = -2\log\left(\hat{L}\right) + p\log(n)$$

donde $\hat{L}$ es la verosimilitud máxima del modelo, p el número de parámetros y n el número de observaciones. Este índice mide el equilibrio entre el ajuste y la complejidad del modelo: cuanto menor es su valor, mejor es el balance conseguido.

Los resultados obtenidos muestran que el mejor rendimiento corresponde al caso `covariance_type = full`, que permite total libertad en la forma de los *clusters*. Este resultado sugiere que este modelo es un candidato para ofrecer mejores resultados que los dos métodos anteriores, ya que la forma esférica de los *clusters* supuesta en dichos métodos, no es la que mejor se ajusta a los datos según el BIC.

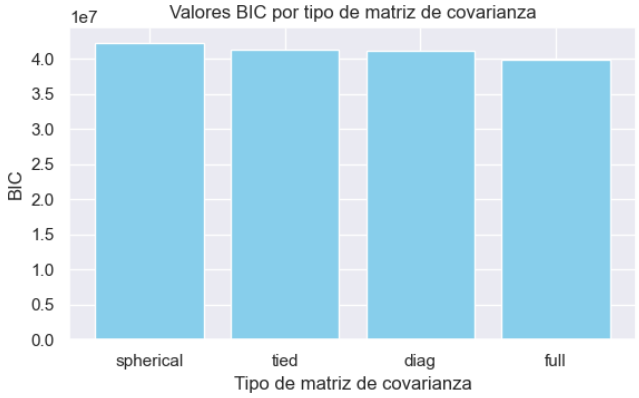

Figura 5.18: Gráfico de barras con el índice BIC para cada modelo GMM entrenado con los distintos tipos de matriz de covarianza. Se observa que el tipo `full` es el que mejor equilibrio tiene entre ajuste y complejidad.

Los *clusters* obtenidos se muestran en la siguiente imagen, donde pueden apreciarse sus formas no esféricas. En el plano PC2–PC3 se observa un menor solapamiento entre los *clusters* en comparación con los modelos anteriores además de una distancia algo mayor entre los *clusters* en este plano.

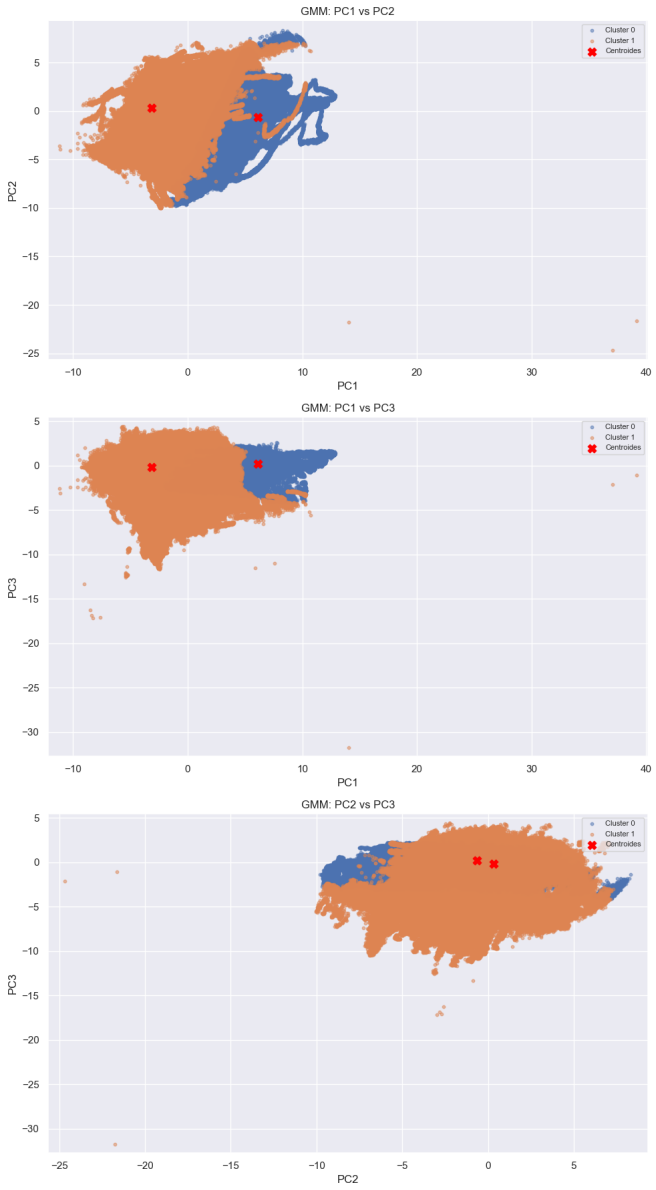

Figura 5.19: Gráficos con los dos *clusters* obtenidos con el algoritmo de GMM. Estos grupos se han proyectado en el espacio de las tres primeras componentes principales del algoritmo de PCA. En la imagen se muestran los centroides de ambos *clusters*.

A diferencia de los dos modelos anteriores, el *cluster* 1 es el de mayor tamaño, conteniendo un 66.1 % de los datos. Por tanto, en este modelo, este *cluster* es el principal candidato a representar el comportamiento normal de los inversores.

Para entender mejor los *clusters*, se graficaron también los valores de los centroi-

des para cada componente principal. Se observa que los valores medios de cada componente principal para cada *cluster* son muy distintos a los obtenidos en los otros dos modelos. El *cluster* 0 se caracteriza por valores positivos en PC1 y PC3 (siendo más elevados en la primera componente) y negativos en PC2, mientras que en el *cluster* 1 ocurre lo contrario.

Figura 5.20: Gráfico de barras con el valor de los dos centroides obtenidos con GMM para la primera componente principal.

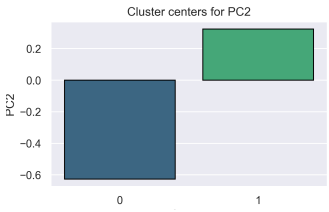

Figura 5.21: Gráfico de barras con el valor de los dos centroides obtenidos con GMM para la segunda componente principal.

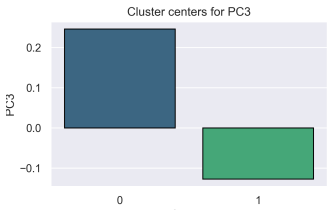

Figura 5.22: Gráfico de barras con el valor de los dos centroides obtenidos con GMM para la tercera componente principal.

Después se procedió a estudiar la calidad del *clustering* obtenido. En este caso se utilizó la distribución de las log-densidades de probabilidad de los puntos la cual se calcula para cada punto al ajustar el modelo siguiendo la fórmula:

$$\log p(x_i) = \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k) \right)$$

Las log-densidades indican cuán probable es para el modelo observar ese dato. Por tanto, si estas son elevadas, se interpreta que el modelo se ajusta bien a los datos.

La distribución se muestra a continuación y se calculó para 10.000 puntos aleatorios debido al coste computacional. Se observa que son pocos los puntos con baja log-densidad y que la zona de valores altos es estrecha, lo que indica que los puntos están muy agrupados en torno a los centros de los *clusters*. Por todo ello, se puede concluir que el *clustering* representa de forma adecuada los datos, capturando los patrones relevantes.

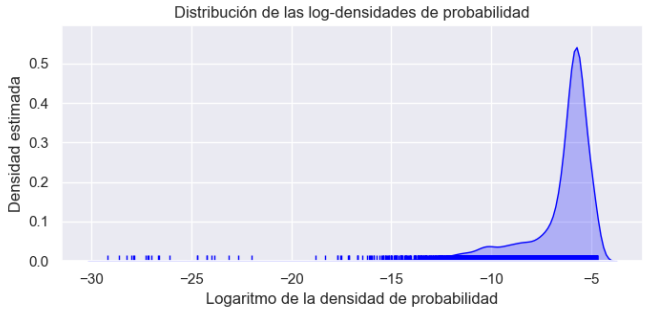

Figura 5.23: Distribución de las log-densidades de probabilidad del modelo GMM para 10.000 muestras de los inversores escogidas de forma aleatoria. Se observan elevadas log-densidades indicando un buen ajuste a los datos del modelo.

Como en los casos anteriores, se probó el modelo GMM sobre las muestras de los inversores 11 y 34, obteniéndose que el 68.76 % de las muestras del inversor 11 pertenecen al *cluster* 1 y el 66.49 % de las del inversor 34, llegando a las mismas conclusiones.

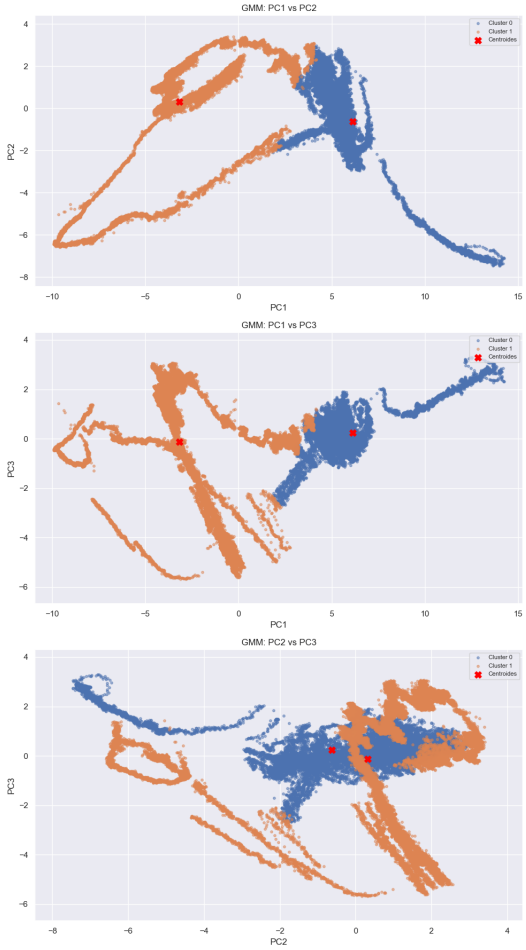

Figura 5.24: Distribución de los dos *clusters* obtenidos mediante GMM usando los datos del inversor 34 el día 13-02-25. Este inversor presenta un comportamiento normal.

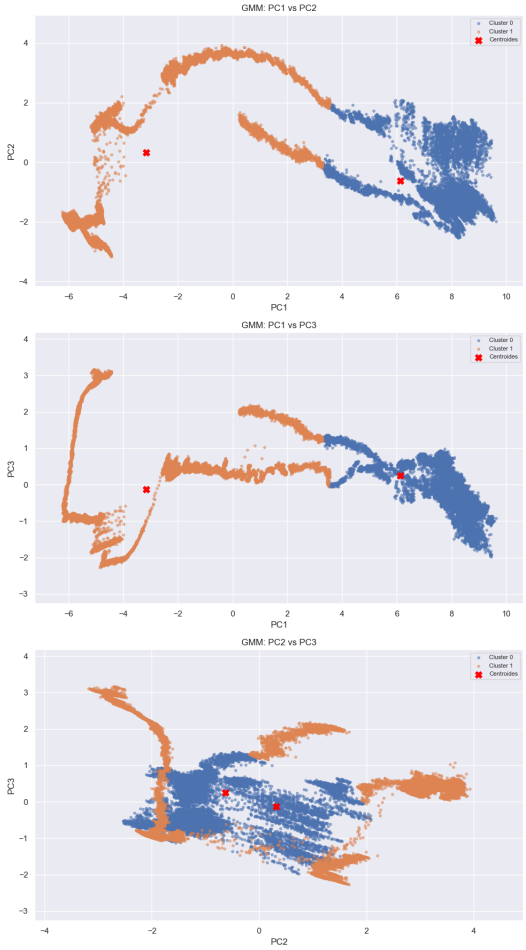

Figura 5.25: Distribución de los dos *clusters* obtenidos mediante GMM usando los datos del inversor 11 el día 13-02-25. En ese día, se registró una explosión tras congelar del equipo.

5.5. Resultados y conclusiones

Para el desarrollo del modelo de *clustering* que mejor detectara los comportamientos anómalos y normales de las muestras de los inversores solares, se probaron tres tipos de modelos.

En relación con los dos primeros modelos (*K-Means* y *Fuzzy C-Means*), ambos partían de la hipótesis de que la forma de los *clusters* era circular y la asignación de los puntos se realizaba a partir de las distancias a los centroides de los *clusters*.

La principal diferencia entre ellos era que FCM permitía que un punto perteneciera a varios *clusters* con distintos grados de pertenencia.

A partir de los grupos obtenidos con estas técnicas, se pudo comprobar que eran muy similares, presentando ligeras diferencias en la tercera componente principal. Para medir la calidad del *clustering*, en el primer modelo se utilizó el coeficiente de Silhouette y en FCM se empleó el FPC, obteniéndose valores de 0,59 y 0,84 respectivamente. Estos son valores aceptables, aunque el modelo FCM resultó más informativo para el caso de uso de los inversores.

El tercer modelo se planteó para dotar de un enfoque distinto al problema. Se empleó el modelo GMM, de carácter probabilístico, que permitía una mayor flexibilidad en la forma de los *clusters*. Se entrenó el modelo con distintas formas y se comprobó, mediante el cálculo del BIC, que los *clusters* que mejor se ajustaban a los datos eran aquellos con total libertad en su forma, mientras que los *clusters* esféricos (caso de los dos modelos probados anteriormente) presentaron el peor balance entre ajuste y complejidad. Además, entrenando el modelo con esta forma libre de los *clusters*, se constató la buena calidad del agrupamiento mediante el análisis de la distribución de las log-densidades de probabilidad.

Por todo ello, se concluyó que el modelo de *clustering* más óptimo para detectar los patrones entre comportamiento anómalo y normal de los inversores solares era el *Gaussian Mixture Model*.

A estas alturas, lo último que quedaba por determinar era cuál de los dos grupos correspondía al comportamiento normal y cuál al fallido. En GMM se partía de la hipótesis de que el *cluster* 1 (naranja) representaba el comportamiento normal al ser el más numeroso. No obstante, al probar el modelo con un inversor etiquetado con fallo y otro con comportamiento normal a priori, se observó que un mayor porcentaje de las muestras del inversor fallido formaban parte del *cluster* naranja. Por tanto, no se podía afirmar con total certeza la hipótesis.

No fue hasta tener etiquetadas las muestras de los inversores del parque Myrtle según el problema de condensación cuando se pudo determinar con mayor certeza cuál era el *cluster* de comportamiento normal. Para ello, se probó el modelo GMM con el inversor B43 (etiquetado con problema de condensación) y el inversor C07 (sin dicha etiqueta). Se obtuvo que un 64,28 % de las muestras del inversor C07 pertenecían al *cluster* 1 frente a un 60,40 % de las muestras del inversor B43. Así, se concluyó finalmente que el *cluster* 1 de GMM podría considerarse como el representante del comportamiento normal de los inversores solares del parque Myrtle.

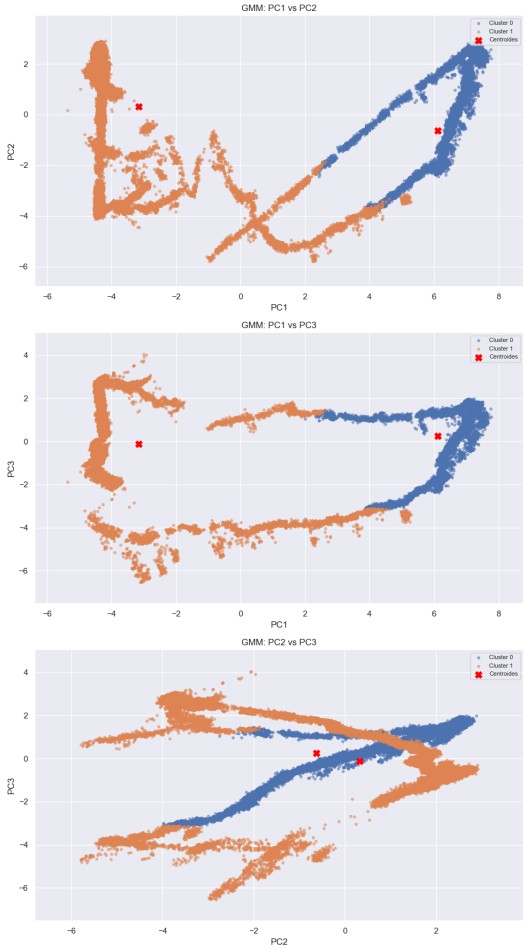

Figura 5.26: Distribución de los dos *clusters* obtenidos mediante GMM usando las muestras del inversor C07 del parque Myrtle etiquetado sin fallo de condensación.

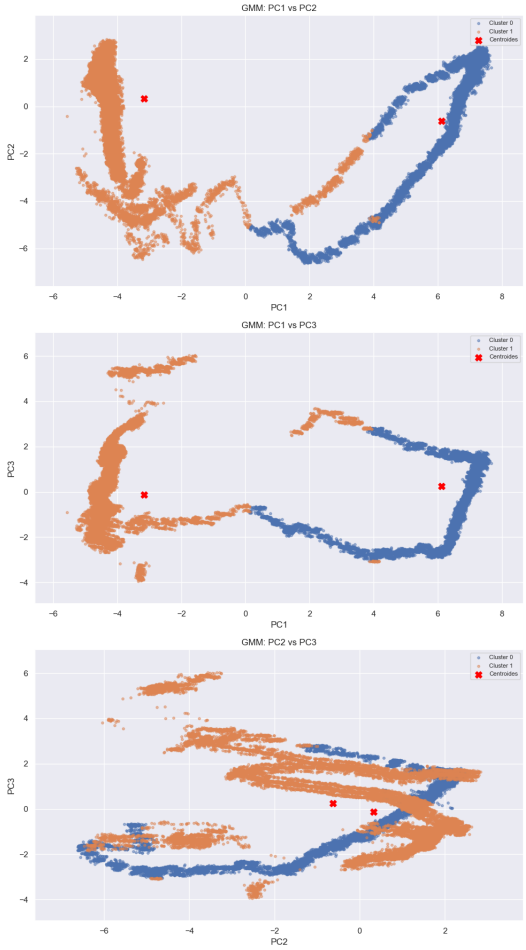

Figura 5.27: Distribución de los dos *clusters* obtenidos mediante GMM usando las muestras del inversor B43 del parque solar Myrtle etiquetado con fallo de condensación.

Por último, cabe destacar que varios factores pudieron influir en que no se pudiera confirmar la hipótesis hasta ese momento. Por un lado, se encontraba la escasa disponibilidad de inversores etiquetados con fallo (únicamente dos) con la que se contaba inicialmente. Por otro lado, aunque el resto de inversores no había registrado un fallo específico, desde la empresa se indicó que el proceso de etiquetado resulta muy complejo para ellos, por lo que podría haber sucedido que alguno de esos inversores hubiera fallado sin que se hubiera podido identificar y etiquetar adecuadamente.

Capítulo 6

Modelo de clasificación

Tras realizar el modelo de *clustering* para el comportamiento de los inversores, la empresa trasladó el deseo de centrarse en el problema de condensación, explicado en la Sección 3.2.1, que afectaba en gran medida a la empresa.

Por ello, fue necesario realizar un etiquetado manual de los datos del parque Myrtle para posteriormente implementar distintos modelos basados en árboles para encontrar aquel que mejor clasificara los inversores según las etiquetas de condensación.

En este capítulo, se detallarán los aspectos relevantes del etiquetado y el desarrollo del modelo de clasificación binaria. Además, se explicarán los aspectos teóricos más relevantes de los dos tipos de modelos probados: *Decision Tree* y *XGBoost*.

6.1. Etiquetado manual

Como se explicó en la Sección 3.3.1, la principal alarma relacionada con el problema de condensación era la 721. Por ello, para cada tipo de inversor, se realizó un `join` por fecha entre los archivos CSV que contenían las variables temporales y los correspondientes a las alarmas.

Dado que los archivos CSV de alarmas recogían un histórico desde el año 2021, fue necesario realizar un filtrado previo de las alarmas por la fecha correspondiente al archivo de variables temporales.

Una vez se disponía de las variables temporales y las alarmas de un inversor para

una fecha determinada, se decidió crear una nueva variable binaria denominada `Condensacion`, que tomara valores en el conjunto $\{0, 1\}$. El valor 1 se asignaba a todas las muestras del inversor si había saltado la alarma 721 en esa fecha, y el valor 0 en caso contrario.

Tras realizar este etiquetado para todos los inversores del parque Myrtle (70 en total), se observó que únicamente tres de ellos fueron etiquetados con el problema de condensación. Con el fin de aumentar el número de inversores con la etiqueta 1 en `Condensacion`, la empresa facilitó los archivos CSV de tres inversores solares de otro parque, ubicado en Cádiz, en los que se sabía que había saltado la alarma 721.

De este modo, aunque se seguía presentando un desbalanceo de clases, se logró incrementar el número de muestras con etiqueta 1, lo que ayudaría al modelo de clasificación a aprender mejor los patrones de fallo por condensación y ganar robustez.

6.2. Análisis exploratorio completo

En el capítulo 4.4 se realizó un análisis exploratorio (*EDA*, por sus siglas en inglés) con los primeros datos de inversores solares recibidos. Dado que desde ese momento se recibieron muchas más muestras de inversores y se han añadido nuevas variables, se consideró pertinente volver a realizar el EDA con los inversores de Myrtle. De este modo, se pudo comprobar si existían diferencias en la distribución y en las correlaciones de las variables.

En primer lugar, se obtuvo la distribución de las variables temporales, entre las que se encuentran aquellas asociadas con el problema de condensación. Comparándola con la Figura 4.1, se observa que en la Figura 6.1 las variables presentan un rango mayor, por lo que las distribuciones de la primera figura pueden interpretarse como distribuciones de la segunda restringidas a determinados valores. También destaca el hecho de que las variables `DPF_HUM_INT` y `DPF_TEMP_INT` siguen siendo nulas en este nuevo análisis, mientras que la variable `PPC_QMG` ya no lo es en estas muestras de los inversores.

En relación con estas dos primeras variables, la empresa informó de que se había incorporado una nueva funcionalidad para poder medirlas en los nuevos modelos de inversores solares, estando ya incluidas en el *software* interno de la CCU. En los equipos antiguos que no incorporaban esta novedad, las variables se consideraban

nulas, siendo el caso de los inversores de Myrtle.

Por último, centrándose en las variables asociadas con la condensación de los inversores, se observa cómo en la distribución de `DELTA_TEMP_AMB` la gran mayoría de los valores son positivos. Esto concuerda con el hecho de que en la distribución de la variable binaria `Condensacion` una gran cantidad de muestras se hayan etiquetado con 0 (sin fallo de condensación).

Figura 6.1: Histogramas que representan la distribución de 43 variables temporales de los archivos CSV de los inversores del parque Myrtle y el situado en Cádiz. No se graficaron la distribución de las variables `time`, `utc` y `sample` pues no aportaban información relevante.

En cuanto a los *boxplots* de las variables, se puede observar, al compararlos con la Figura 4.2, una mayor longitud en las cajas de la Figura 6.2. Cabe recordar que dicha longitud representa el rango intercuartílico (IQR, por sus siglas en inglés), y que el hecho de que las distribuciones de las variables presenten un rango mayor explica esta mayor longitud. También se observa que variables como las relacionadas con `TBRID` apenas tienen valores atípicos en este nuevo análisis en comparación con el anterior.

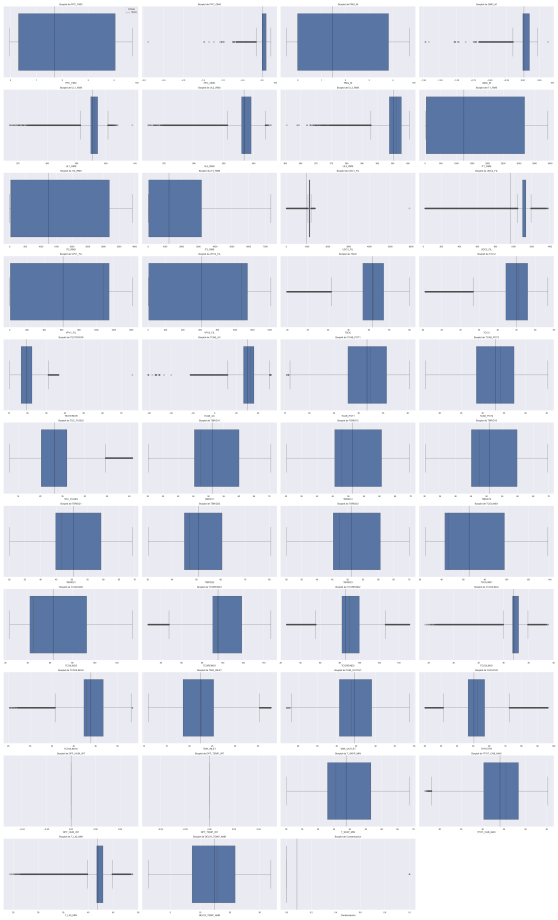

Figura 6.2: *Boxplots* de las variables temporales de los archivos CSV recibidos en la primera ronda de datos. No se graficaron los correspondientes a las variables `time`, `utc` y `sample` pues no aportaban información relevante.

Por último, se graficó la matriz de correlaciones de las variables, apreciándose algunas diferencias respecto a la Figura 4.8. Destacan las elevadas correlaciones entre `TBRID11`, `TBRID12`, `TBRID13`, `TBRID21`, `TBRID22`, `TBRID23`, `TCOOLIND1`, `TCOOLIND2`, `TCOREIND1`, `TCOREIND2`, `TCOOLINGI`, `TCOOLINGO`, todas ellas asociadas a temperaturas de elementos del inversor por los que pasa la misma corriente, lo que explica su correlación. Por el contrario, variables como `TSOC`, `TCCU`, `TEXTERIOR`, `TCAB_AC`, `TCAB_POT1`, `TCAB_POT2`, `TDC_FUSES`, relacionadas también con temperaturas pero en este caso ambientales, no presentan altas correlaciones con las anteriores, ya que el calentamiento de los componentes del inversor no implica necesariamente un aumento proporcional en dichas temperaturas. Esta diferencia entre ambos tipos fue detectada por la técnica PCA, como se indicó en el capítulo anterior.

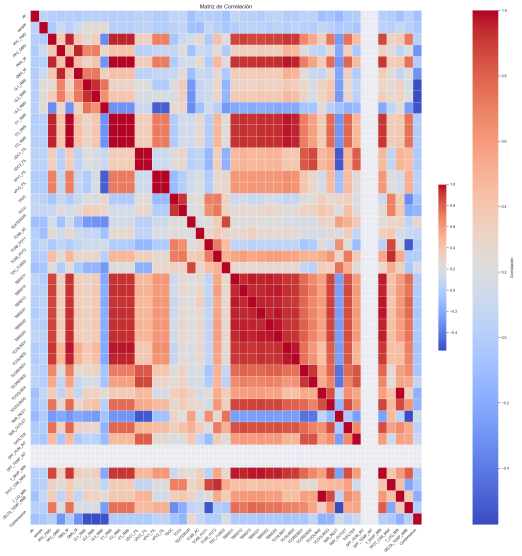

Figura 6.3: *Heatmap* que representa la matriz de correlaciones lineales de las variables temporales de los inversores del parque Myrtle. Para su cálculo se utilizó el coeficiente de correlación de Pearson. Las celdas en tonos grises de tamaño menor se deben a la desviación estándar nula de la variable del denominador en la fórmula de Pearson.

6.3. Fundamentos teóricos de los modelos usados

El objetivo de esta sección es introducir los conceptos matemáticos que subyacen a los modelos de *Decision Tree* y *XGBoost* utilizados en este trabajo. Estas nociones resultan útiles para comprender en mayor profundidad la implementación del código de dichos modelos y se seguirá principalmente la referencia [26].

6.3.1. *Decision Tree*

El objetivo de los modelos basados en árboles es dividir el espacio de características en subconjuntos, obtenidos mediante cortes paralelos a alguno de los ejes principales. En cada una de estas regiones, el modelo asigna un valor constante a todos los puntos contenidos en ella.

Dependiendo del objetivo, existen dos tipos de árboles: de regresión o de clasificación. En los primeros, el valor asignado en cada región suele ser la media de todos los puntos contenidos en ella. En cambio, en los de clasificación, suele ser la etiqueta mayoritaria. Este trabajo se centrará en estos últimos debido al caso de uso planteado.

Todo árbol de decisión (DT), ya sea de regresión o clasificación, puede modelarse como una función f que cumple:

$$f(x) = \sum_{m=1}^{M} c_m \, I(x \in R_m)$$

donde $x \in X$ es un punto en el espacio de características X, M es el número de subregiones en X (es decir, $X = \{R_1, \ldots, R_M\}$), c_m es una constante que, en árboles de clasificación, corresponde a alguna de las posibles etiquetas del problema, e $I(\cdot)$ es la función indicadora que vale 1 si se cumple la condición entre paréntesis y 0 en caso contrario.

Para determinar los mejores cortes que permitan clasificar los datos, es necesario definir una medida que cuantifique la impureza de cada nodo. El objetivo será que las clases sean lo más puras posible, es decir, que la mayoría de los puntos en cada región pertenezcan a la misma clase.

Sea y la variable objetivo, dada una región R_m con N_m observaciones y para cada etiqueta k, se define:

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

que representa la proporción de observaciones etiquetadas como k en el nodo m. De esta forma, las observaciones del nodo m se asignan a la clase $k(m) = \arg\max_k \hat{p}_{mk}$.

Existen diversas medidas de impureza para el nodo m en un árbol de decisión T, denotadas $Q_m(T)$:

- Error de clasificación: $1 - \hat{p}_{mk}$
- Índice de Gini: $\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$
- *Cross-Entropy*: $-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$

Es importante destacar que en las tres medidas se realiza una ponderación por el tamaño de los nodos hijos al evaluar un corte. Aunque las tres son similares, las dos últimas son diferenciables, lo que facilita la optimización numérica a la hora de escoger el corte. Además, el índice de Gini y la entropía cruzada son más sensibles a cambios en las probabilidades de clase, detectando mejor variaciones sutiles en la pureza de los nodos. Por ello, en la práctica se suelen utilizar alguna de estas dos. En el caso de la clasificación binaria de inversores solares, se empleó el índice de Gini, cuanto mayor sea este mejor será el corte asociado.

Por último, cabe destacar que los árboles de decisión son aproximadores universales, por lo que pueden ajustarse muy bien a los datos si se permite un número suficiente de nodos. Para evitar el sobreajuste y encontrar un equilibrio entre ajuste y complejidad, se puede establecer un número mínimo de observaciones necesarias para realizar un corte o para que una hoja sea válida. Este tipo de medidas se implementaron en el código como se verá en secciones siguientes.

6.3.2. *XGBoost*

Este algoritmo se basa en una técnica de ensamblado de modelos llamada *boosting*. Mientras que en un algoritmo tradicional de árbol de decisión se construye un único árbol, en las técnicas de *boosting* de árboles se construyen varios, formando el modelo:

$$f_M(x) = \sum_{m=1}^{M} T(x; \Theta_m)$$

donde M es el número de iteraciones y $\Theta_m = \{R_m, c_m\}$ representa los parámetros del árbol obtenido en la iteración m. Esta suma de árboles se construye de forma secuencial, resolviendo en cada etapa m el siguiente problema de optimización:

$$\hat{\Theta}_m = \arg\min_{\Theta_m} \sum_{i=1}^{N} L\big(y_i, f_{m-1}(x_i) + T(x_i; \Theta_m)\big)$$

donde Θ_m es el conjunto de parámetros del árbol que se ajusta en la etapa m dado el modelo actual $f_{m-1}(x)$, y $L(\cdot)$ es una función de pérdida. Como encontrar la región y los parámetros óptimos de cada árbol puede ser computacionalmente costoso, se emplean técnicas de optimización aproximada, como *gradient boosting*, que es el método utilizado en esta parte del trabajo.

El objetivo principal de esta técnica es construir secuencialmente un modelo que sea la suma de varios árboles de decisión simples, cada uno entrenado en una etapa distinta. Tras construir el árbol en la etapa m, se evalúa su rendimiento mediante la función de pérdida $L(\cdot)$ para obtener los errores (residuos), y el árbol de la etapa $m + 1$ se entrena para predecir estos errores del modelo anterior.

Algoritmo de *Gradient Boosting*

1. Se inicializa
$$f_0(x) = \arg\min_c \sum_{i=1}^{N} L(y_i, c)$$

2. Para $m = 1$ hasta M:

   a) Para $i = 1, \ldots, N$, se calculan los residuos r_{im} como:
   $$r_{im} = - \left. \frac{\partial L\big(y_i, f(x_i)\big)}{\partial f(x_i)} \right|_{f=f_{m-1}}$$

   b) Se ajusta un árbol de regresión a los residuos r_{im}, definiendo los nodos terminales R_{jm} con $j = 1, \ldots, J_m$.

   c) Para $j = 1, \ldots, J_m$, se calcula:
   $$c_{jm} = \arg\min_c \sum_{x_i \in R_{jm}} L\big(y_i, F_{m-1}(x_i) + c\big) \tag{6.1}$$

   d) Se actualiza el modelo:
   $$f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} c_{jm}\, I\big(x \in R_{jm}\big)$$

3. El modelo final será:
$$\hat{f}(x) = f_M(x)$$

En el caso de clasificación, se suele utilizar como función de pérdida:

$$L(y, f(x)) = \log\left(1 + e^{-2yf(x)}\right)$$

cuyo gradiente es:

$$\frac{\partial L(y, f(x))}{\partial f(x)} = -\frac{2y}{1 + e^{2yf(x)}}$$

Esta función de pérdida mide la discrepancia entre las probabilidades predichas por un modelo y las etiquetas verdaderas en un problema de clasificación. Sustituyendo en la ecuación (6.1) se obtiene la expresión:

$$c_{jm} = -\frac{\sum_{x_i \in R_{jm}} g_i}{\sum_{x_i \in R_{jm}} h_i}$$

donde g_i es el gradiente y h_i es la hessiana de la función de pérdida *log-loss*.

6.4. Modelos implementados

Una vez presentados los conceptos teóricos de estas técnicas, se describen los diferentes modelos implementados.

En total se evaluaron cuatro modelos: dos basados en *Decision Tree* y dos en *XGBoost*. Para su entrenamiento se usaron tanto los datos de los inversores del parque Myrtle como los situados en Cádiz siendo un total de 3.110.328 muestras de 73 inversores.

Debido al desbalanceo de clases, se quería que la división en el conjunto de *train* (80 %) y *test* (20 %) se hiciera manteniendo la proporción de las etiquetas. Además, como el etiquetado se había realizado a nivel de inversor, lo correcto era que la totalidad de las muestras de un inversor pertenecieran a uno de los dos conjuntos por lo que el 80 % de los inversores constituían el conjunto de entrenamiento (57, de los cuales 7 inversores habían tenido problema de condensación) y el 20 % restante al de *test* (15, de los cuales 1 falló por condensación). En número de muestras, el primer conjunto tenía 2.462.343 mientras que el segundo 647.985.

A continuación se detallan, de forma breve, el conjunto de entrenamiento y los parámetros utilizados en cada caso.

6.4.1. Modelo basado en DT considerando las variables T_SKIIP_MIN, TPOT_CAB_MAX, T_LIQ_MIN, DELTA_TEMP_AMB

Se utilizó la clase `DecisionTreeClassifier()` [63] con los siguientes parámetros:

- `criterion='gini'`: Utiliza el índice de Gini para evaluar la impureza.

- `min_samples_split=5`: Mínimo número de observaciones en un nodo para poder seguir realizando cortes.
- `min_samples_leaf=5`: Mínimo número de observaciones en un nudo terminal.
- `class_weight='balanced'`: Permite aumentar el peso de la clase minoritaria (en este caso de uso sería la etiqueta de `Condensacion` 1). Resulta muy útil para evitar el sesgo en la clase mayoritaria en problemas de clasificación con desbalanceo de clases.
- `random_state=150`: Semilla para asegurar la reproducibilidad.

A parte de estos parámetros también se encuentra `DT_min_impurity_decrease`. Se decidió buscar el valor óptimo de este parámetro mediante *Grid Search Cross Validation* usando la clase `GridSearchCV()`. Se optó por escoger como parámetro de `scoring` el *recall* (capacidad del modelo para detectar todos los casos positivos reales). La búsqueda del valor óptimo se realizó con valores que van desde 0 hasta 0,05 con un paso de 0,005. Como se observa en la siguiente figura, el valor de este parámetro que maximiza dicha métrica es 0,01.

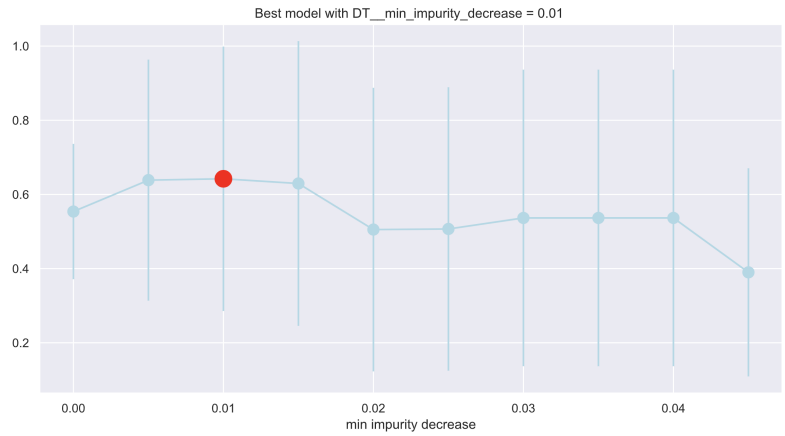

Figura 6.4: Gráfico en el que se muestra el valor de la métrica *recall* para cada valor de `DT_min_impurity_decrease` que incluye todos los valores desde 0 hasta 0,05 a paso 0,005. El valor máximo de esta métrica para el modelo de DT con 4 variables se obtuvo con `DT_min_impurity_decrease = 0,01`.

Una vez definidos todos los parámetros del modelo se procedió a entrenarlo con las muestras del conjunto de entrenamiento seleccionando únicamente las variables `T_SKIIP_MIN`, `TPOT_CAB_MAX`, `T_LIQ_MIN`, `DELTA_TEMP_AMB` que eran las más relacionadas con el problema de condensación.

Tras el entrenamiento del modelo se graficaron tanto la importancia de las variables como el árbol resultante, con el objetivo de entender mejor el funcionamiento del modelo y dotarlo de cierta interpretabilidad. Para obtener la Figura 6.5 se empleó el atributo `feature_importances_`, mientras que para la Figura 6.9 se utilizó la función `plot_tree()`.

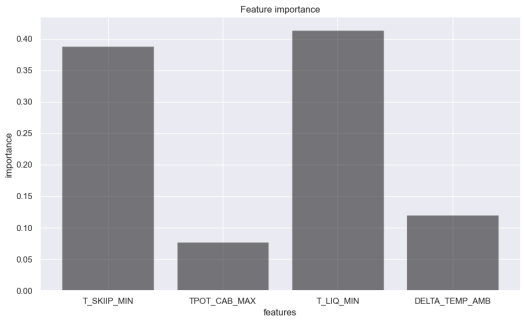

Figura 6.5: Gráfico de barras en el que se muestra la importancia de cada variable en el modelo DT entrenado con 4 variables, utilizando para su cálculo el descenso de la impureza medido con el índice Gini.

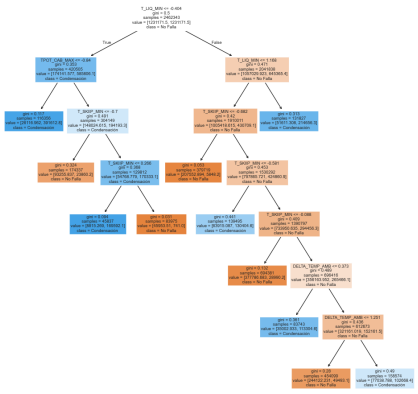

Figura 6.6: Gráfico que muestra un esquema del árbol de decisión obtenido en el modelo DT entrenado con 4 variables. En cada nodo se indica el corte realizado, la impureza, número de muestras y distribución de clases.

En la Figura 6.5 se observa que la variable en la que el modelo se basa principalmente para realizar los cortes es `T_LIQ_MIN`, lo que concuerda con el esquema mostrado en la Figura 6.9. En este último se aprecia que el primer corte realizado por el árbol corresponde a dicha variable. Los primeros cortes en los árboles de decisión se efectúan sobre las variables que más contribuyen a la reducción de la impureza de los nodos; de ahí que `T_LIQ_MIN` se identifique como la variable más relevante según el índice de Gini.

6.4.2. Modelo basado en DT considerando las dos primeras componetes principales de PCA

En este modelo se aplicó la transformación PCA de todas las variables descrita en la Sección 5.1, seleccionando las dos primeras componentes principales (en vez de tres) por ser las más relacionadas con el problema de condensación. Nuevamente

se utilizó la clase `DecisionTreeClassifier()`, manteniendo los mismos valores en casi todos los parámetros, salvo en `DT_min_impurity_decrease`.

Tras realizar un *Grid Search Cross Validation*, se determinó que el valor de `DT_min_impurity_decrease` que maximizaba el *recall* era 0,005, como se muestra en la Figura 6.7.

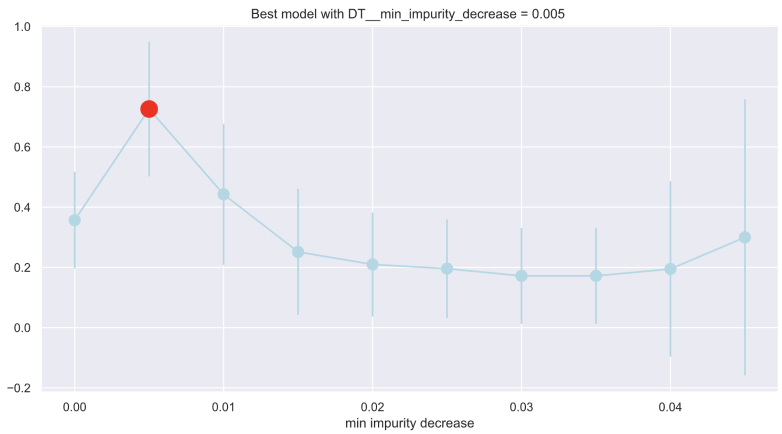

Figura 6.7: Gráfico en el que se muestra el valor de la métrica *recall* para cada valor de `DT_min_impurity_decrease` que incluye todos los valores desde 0 hasta 0,05 a paso 0,005. El valor máximo de esta métrica para el modelo de DT con las dos primeras componentes principales se obtuvo con `DT_min_impurity_decrease` = 0,005.

Una vez entrenado el modelo, se graficaron nuevamente la importancia de las variables y el árbol resultante. En este caso, la variable con mayor importancia fue `PCA_1`, la cual coincide con la variable utilizada por el árbol para realizar el primer corte. Cabe recordar que esta variable representa la temperatura de distintas componentes del inversor y captura la mayor variabilidad de los datos; es decir, contiene la mayor parte de la información. Por tanto, resulta coherente que el modelo haya utilizado `PCA_1` para efectuar el primer corte.

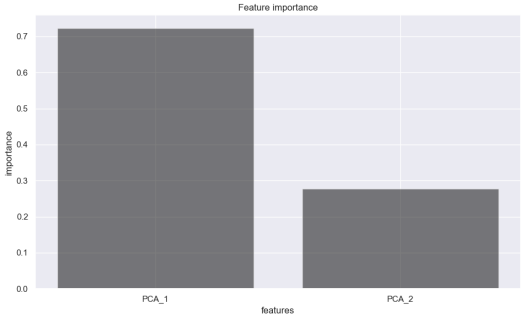

Figura 6.8: Gráfico de barras en el que se muestra la importancia de cada variable en el modelo DT entrenado con las dos primeras componentes principales, utilizando para su cálculo el descenso de la impureza medido con el índice Gini.

Figura 6.9: Gráfico que muestra un esquema del árbol de decisión obtenido en el modelo DT entrenado con las dos primeras componentes principales. En cada nodo se indica el corte realizado, la impureza, número de muestras y distribución de clases.

6.4.3. Modelo basado en *XGBoost* considerando las variables T_SKIIP_MIN, TPOT_CAB_MAX, T_LIQ_MIN, DELTA_TEMP_AMB

Para el entreno de este modelo se utilizó la clase `XGBClassifier()` con los siguientes parámetros:

- `scale_pos_weight = len(y_train[y_train == 0]) / len(y_train[y_train == 1])`: Factor de peso para panalizar más los errores de clasificación cometidos en la clase positiva.
- `tree_method = 'gpu_hist'`: Método de construcción del árbol optimizado para GPU, muy rápido para *datasets* grandes.
- `predictor = 'gpu_predictor'`: Indica que las predicciones también se harán usando GPU.
- `eval_metric = 'logloss'`: Métrica usada para evaluar el error durante el entrenamiento.

- `use_label_encoder = False`: Evita una advertencia por usar un codificador de etiquetas interno que ahora está obsoleto.
- `random_state = 150`

Como en el caso de los dos modelos anteriores, se tuvo que utilizar *Grid Search Cross Validation* para seleccionar el valor óptimo de los siguientes parámetros que maximiza el *recall*. A continuación, se determinan los parámetros y el rango de valores entre los que se hizo la búsqueda:

- `xgb__n_estimators: [100, 200]`: Número total de árboles (estimadores) que se construyen. Más árboles suelen mejorar el rendimiento, pero aumentan el tiempo de entrenamiento y el riesgo de *overfitting*.
- `xgb__max_depth: [3, 5, 7]`: Profundidad máxima permitida para cada árbol, controla la complejidad del árbol.
- `xgb__learning_rate: [0.01, 0.1]`: Paso de actualización en cada iteración, controla cuánto aporta cada árbol. Los valores bajos hacen el entrenamiento más lento, pero pueden mejorar la generalización mientras que los valores altos permiten un aprendizaje más rápido, aunque pueden causar sobreajuste.
- `xgb__subsample: [0.8, 1.0]`: Proporción de muestras usadas para construir cada árbol (*bootstrap sampling*). Ayuda a reducir el sobreajuste al usar solo parte de los datos en cada árbol.
- `xgb__colsample_bytree: [0.8, 1.0]`: Proporción de características (columnas) usadas para construir cada árbol. Similar a *Random Forest*, que emplea una muestra aleatoria de columnas para diversificar los árboles.

Los rangos escogidos para cada parámetro son los más habituales en la literatura siendo este el motivo por el que se escogieron. Los parámetros óptimos fueron:

```
{
    'xgb__colsample_bytree': [0.8],
    'xgb__learning_rate': [0.1],
    'xgb__max_depth': [5],
    'xgb__n_estimators': [200],
    'xgb__subsample': [1.0]
}
```

En este caso, debido al elevado número de combinaciones posibles, no se pudo obtener la gráfica con la evolución del *recall* para las distintas combinaciones.

Tras el entrenamiento del modelo, se volvió a graficar la importancia de las variables, obteniendo nuevamente que `T_LIQ_MIN` es la más relevante, al igual que en el modelo de *Decision Tree*. Dado que en los modelos de *XGBoost* se entrenan múltiples árboles, se representó gráficamente el número de nodos en cada árbol del ensamblado, lo que permitió estudiar la complejidad general de los árboles generados por este modelo.

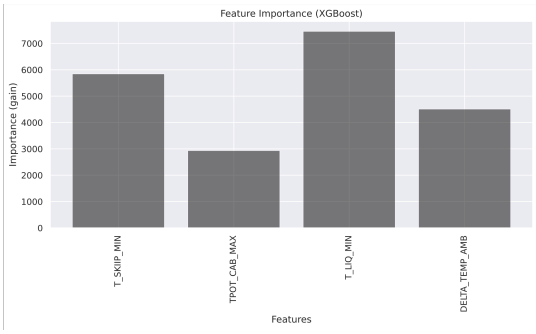

Figura 6.10: Gráfico de barras en el que se muestra la importancia de cada variable en el modelo *XGBoost* entrenado con 4 variables, utilizando para su cálculo el descenso de la impureza medido con el índice Gini.

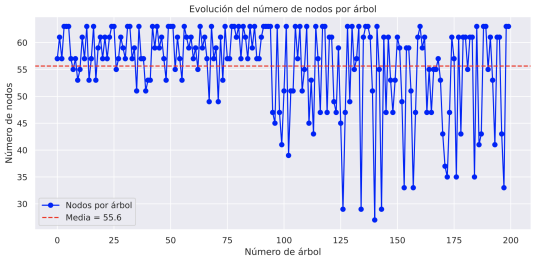

Figura 6.11: Gráfico en el que se muestra el número de nodos en cada árbol del ensamblado del modelo *XGBoost* entrenado con 4 variables. Además se marca en línea discontinua la media de nodos por árbol.

En la Figura 6.11 se observa que algunos de los últimos árboles del ensamblado presentan un menor número de nodos, siendo por tanto más sencillos. Esta diferencia en el número de nodos puede deberse a las variables disponibles en el momento de realizar cada corte. Es importante recordar que en este caso se estableció el parámetro `'xgb_colsample_bytree'` en 0,8, lo que implica que para cada árbol se selecciona aleatoriamente un 80 % de las variables totales (en este caso, 3). Es posible que en los árboles de menor profundidad, las variables seleccionadas incluyeran las más relevantes, permitiendo al modelo realizar cortes más eficaces sin necesidad de gran profundidad. Por el contrario, un mayor número de nodos podría deberse a que entre las variables disponibles no se encontraban las más relevantes, requiriendo más cortes para alcanzar un ajuste adecuado. En cuanto a la media de nodos por árbol, esta se situa en 55,6.

6.4.4. Modelo basado en *XGBoost* considerando las dos primeras componentes principales de PCA

Para este último modelo, se volvió a aplicar la transformación de las muestras de los inversores al espacio de las dos primeras componentes principales. Nuevamente, se utilizó la clase `XGBClassifier()` con los mismos valores de los parámetros y, en este caso, también se obtuvieron los mismos valores óptimos al aplicar *Grid Search Cross Validation*.

Una vez entrenado el modelo, se representaron gráficamente tanto la importancia de las variables como el número de nodos de cada árbol del ensamblado. En la Figura 6.12 se observa que, nuevamente, `PCA_1` es la variable más relevante del modelo. Por otra parte, en la Figura 6.13 se aprecia la variabilidad en el número de nodos entre los distintos árboles, con una media de 44,2 nodos por árbol, valor que resulta ligeramente inferior al observado en el modelo anterior.

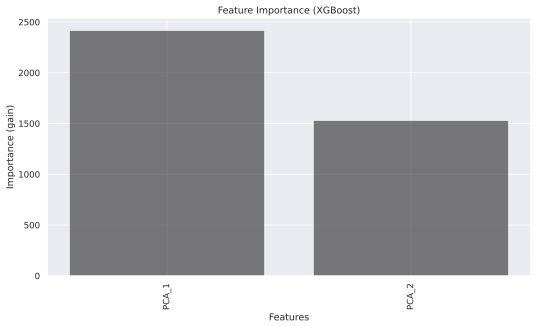

Figura 6.12: Gráfico de barras en el que se muestra la importancia de cada variable en el modelo *XGBoost* entrenado con las dos primeras componentes principales, utilizando para su cálculo el descenso de la impureza medido con el índice Gini.

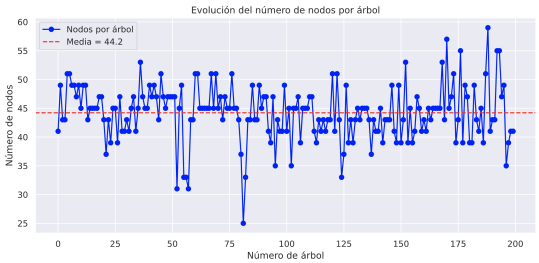

Figura 6.13: Gráfico en el que se muestra el número de nodos en cada árbol del ensamblado del modelo *XGBoost* entrenado con las dos primeras componentes principales. Además se marca en línea discontinua la media de nodos por árbol.

6.5. Resultados y comparativas de los modelos

En esta sección se presentan los resultados obtenidos para cada modelo, así como las comparaciones entre ellos. Este análisis permitió determinar el modelo que mejor ajuste presentaba para el problema de clasificación del fallo por condensación en los inversores.

En primer lugar, se calcularon las métricas de *accuracy*, *specificity*, *recall* y *F1 Score* para cada modelo y en cada conjunto de entrenamiento y *test*. Las fórmulas

de estas métricas son las siguientes:

$$
\begin{aligned}
\text{Accuracy} &= \frac{TP + TN}{TP + TN + FP + FN} \\
\text{Specificity} &= \frac{TN}{TN + FP} \\
\text{Recall} &= \frac{TP}{TP + FN} \\
F_1 \text{ Score} &= 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN} \\
\text{Balanced Accuracy} &= \frac{\text{Sensitivity} + \text{Specificity}}{2} = \frac{\frac{TP}{TP+FN} + \frac{TN}{TN+FP}}{2}
\end{aligned}
$$

donde TP (*True Positive*) son los casos positivos correctamente identificados por el modelo, TN (*True Negative*) los casos negativos correctamente identificados, FP (*False Positive*) los casos negativos clasificados incorrectamente como positivos y FN (*False Negative*) los casos positivos clasificados incorrectamente como negativos.

Las métricas empleadas permiten evaluar distintos aspectos del rendimiento de los modelos. El *accuracy* indica el porcentaje total de casos correctamente clasificados, combinando aciertos en ambas clases. El *specificity* mide la capacidad del modelo para identificar correctamente los casos negativos, es decir, aquellos que no corresponden a la clase de fallo. El *recall*, o sensibilidad, refleja la capacidad para detectar correctamente los casos positivos (fallos por condensación). El *precision* expresa la proporción de casos clasificados como positivos que son realmente positivos, lo que permite evaluar el nivel de falsos positivos. El *F1 Score* combina *precision* y *recall* en una única métrica que pondera ambos aspectos de forma equilibrada. Por último, el *balanced accuracy* es la media de la *recall* y la *specificity*, ofreciendo una medida más robusta frente al desbalanceo de clases.

Dado que el cálculo de algunas de estas métricas resultaba costoso computacionalmente, se optó por calcularlas por lotes (*batches*), obteniéndose los resultados que se muestran en la Tabla 6.1.

Cuadro 6.1: Resultados de métricas para los distintos modelos de clasificación.

Modelo	**Accuracy**	**Recall**	**Specificity**	**Precision**	**F1 Score**	**Bal. Acc.**
DT_4 (TRAIN)	0.80	0.91	0.79	0.29	0.44	0.85
DT_4 (TEST)	0.69	0.08	0.73	0.02	0.03	0.41
DT_PCA (TRAIN)	0.73	0.78	0.73	0.22	0.34	0.75
DT_PCA (TEST)	0.68	0.31	0.71	0.07	0.11	0.51
XGB_4 (TRAIN)	0.97	0.97	0.97	0.78	0.86	0.97
XGB_4 (TEST)	0.85	0.05	0.91	0.04	0.04	0.48
XGB_PCA (TRAIN)	0.72	0.74	0.72	0.20	0.32	0.73
XGB_PCA (TEST)	0.68	0.35	0.71	0.08	0.13	0.53

En general, se observan diferencias notables entre las métricas de los conjuntos de entrenamiento y de prueba, especialmente en *recall*, lo que podría indicar cierto grado de *overfitting*. Los modelos con mayor valor en esta métrica en el conjunto de prueba son los entrenados con las dos primeras componentes principales, obteniéndose un 0,31 para el *Decision Tree* y un 0,35 para *XGBoost*. Además, ambos presentan el mismo *accuracy* en el conjunto de prueba, aunque el *balanced accuracy* es ligeramente superior en el caso de *XGBoost*, lo que indica que este modelo clasifica mejor la clase minoritaria correspondiente al fallo por condensación.

Asimismo, se decidió estudiar el ajuste de cada modelo en cada una de las clases usando el conjunto de *test* para sacar conclusiones sobre cómo generalizan los modelos. Para ello, se representaron gráficamente la curva ROC, el *calibration plot*, la distribución de las probabilidades y la evolución del *accuracy* en función de distintos umbrales de decisión. Una breve descripción de estas métricas es la siguiente [64]:

- Curva ROC: Representa gráficamente la relación entre la tasa de verdaderos positivos (*recall* o *sensitivity*) y la tasa de falsos positivos para distintos umbrales de clasificación. Permite evaluar la capacidad discriminativa del modelo entre las clases. El área bajo dicha curva evalúa la capacidad del modelo para distinguir entre clases. Cuanto más próximo a 1, mejor será la distinción.

- *Calibration plot*: Compara las probabilidades predichas por el modelo con las probabilidades reales observadas. Un modelo bien calibrado proporcionará estimaciones de probabilidad que reflejen adecuadamente la frecuencia real de los eventos. Cuanto más diagonal sea la línea, mejor calibrado estará el modelo.

- Distribución de las probabilidades: Muestra cómo se distribuyen las probabilidades predichas para cada clase, permitiendo analizar si el modelo tiende a ser conservador o a asignar probabilidades extremas. Una buena distribución de las probabilidades se dará cuando la clase negativa se estime con probabilidades cercanas a 0 y la clase postiva con probabilidades cercanas a 1.

- *Accuracy* según umbral: Permite visualizar cómo varía el *accuracy* del modelo en función del umbral de decisión utilizado para clasificar una muestra como positiva o negativa. Esto es útil para seleccionar el umbral que optimiza el rendimiento en función del objetivo del problema.

Es importante destacar, que como los gráficos obtenidos para cada una de las clases son complementarios se analizaron únicamente los asociados con la clase 1 (fallo de condensación) al ser la más relevante.

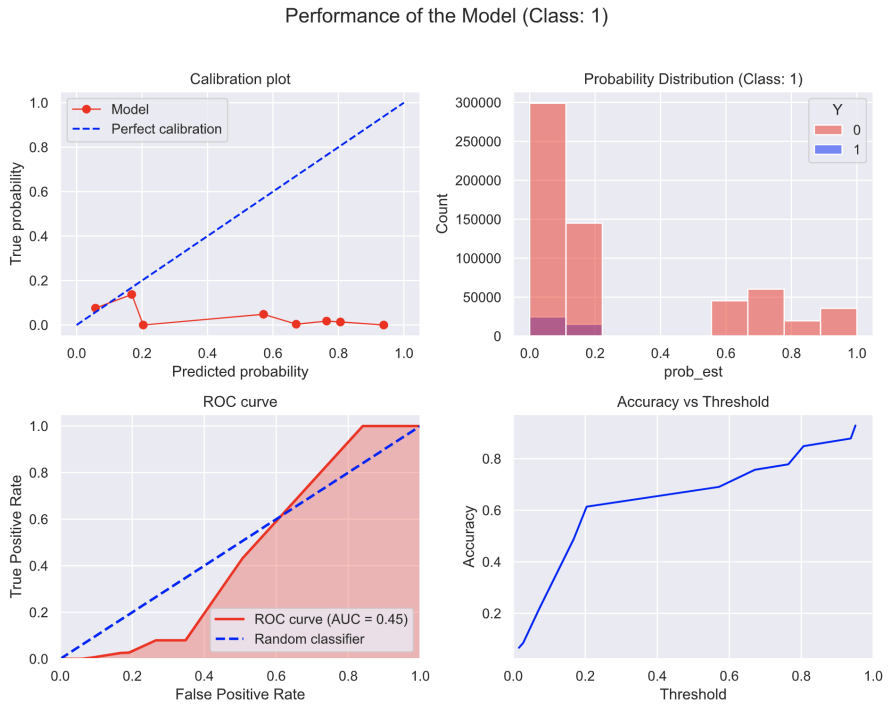

Figura 6.14: Gráficas en las que se muestra la curva ROC, *calibration plot*, distribución de probabilidades y evolución *accuracy* según umbral de decisión para la clase positiva 1 en el modelo de DT entrenado con 4 variables.

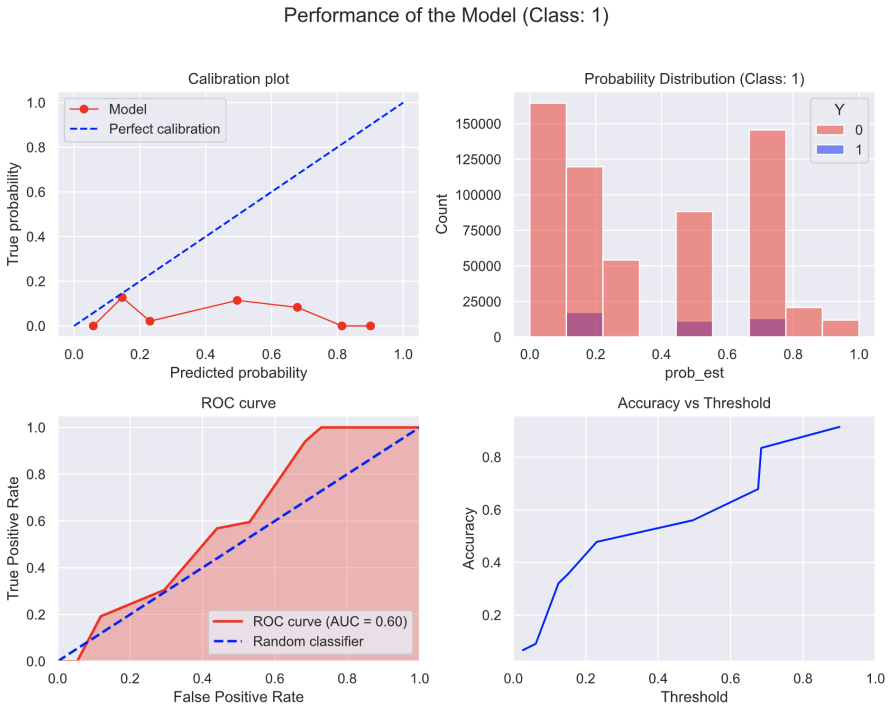

Figura 6.15: Gráficas en las que se muestra la curva ROC, *calibration plot*, distribución de probabilidades y evolución *accuracy* según umbral de decisión para la clase positiva 1 en el modelo de DT entrenado con las dos primeras componentes principales.

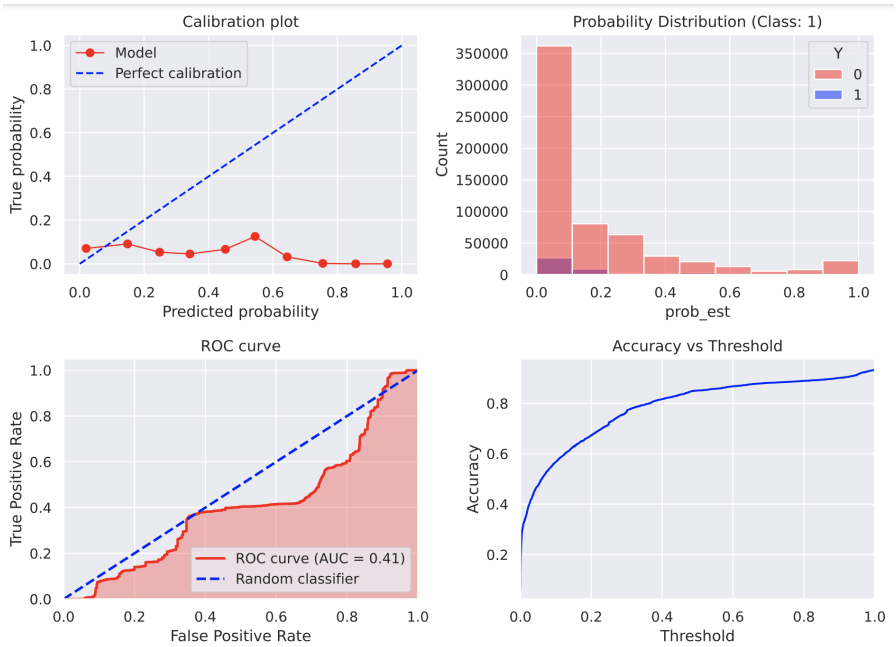

Figura 6.16: Gráficas en las que se muestra la curva ROC, *calibration plot*, distribución de probabilidades y evolución *accuracy* según umbral de decisión para la clase positiva 1 en el modelo de *XGBoost* entrenado con 4 variables.

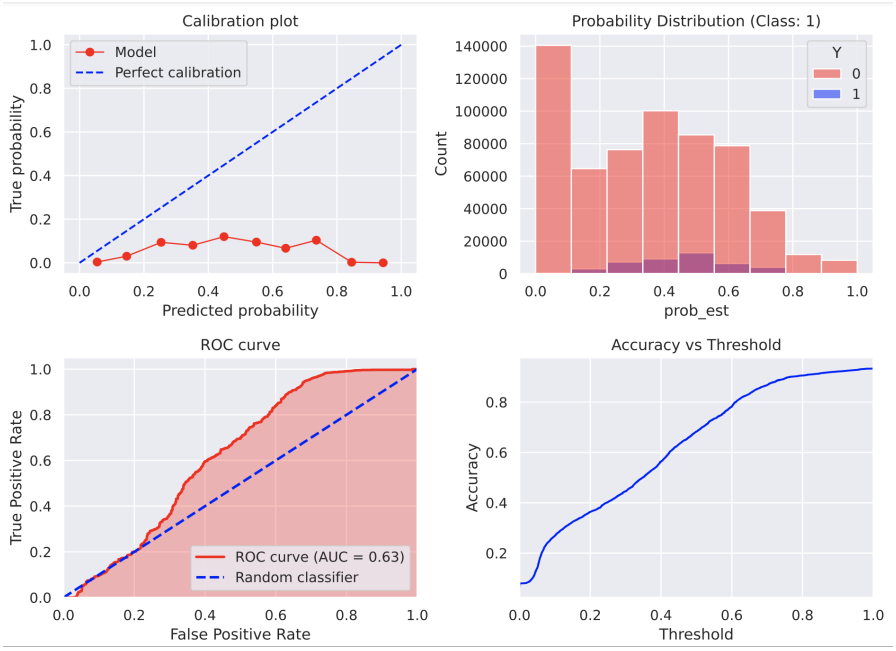

Figura 6.17: Gráficas en las que se muestra la curva ROC, *calibration plot*, distribución de probabilidades y evolución *accuracy* según umbral de decisión para la clase positiva 1 en el modelo de *XGBoost* entrenado con las dos primeras componentes principales.

En las Figuras 6.14, 6.15, 6.16 y 6.17 se observan *calibration plots* bastante similares que evidencian una mala calibración del modelo, posiblemente debida al desbalanceo de clases. Este desbalanceo puede provocar que el modelo tienda a predecir probabilidades sesgadas hacia la clase mayoritaria.

Respecto a la distribución de probabilidades, se observa que la clase negativa, 0, se asigna con bajas probabilidades, lo cual es habitual en un problema de clasificación binaria. En las Figuras 6.14 y 6.16 se puede ver que casi todas las muestras de la clase 1 fueron predichas con probabilidades muy cercanas a 0, asignándose erróneamente a la clase 0, lo que explica en parte los bajos *recall* obtenidos en el conjunto de *test* de estos modelos. En cambio, en figuras como 6.17, el modelo otorga probabilidades mayores a la clase 1.

En relación con la evolución del *accuracy* en función del umbral de decisión, en las cuatro gráficas se observa una tendencia creciente. Esto indica que el *accuracy* mejora a medida que se incrementa el umbral de probabilidad a partir del cual se considera que una muestra pertenece a la clase positiva. Esta tendencia se explica nuevamente por el desbalanceo de clases: a un umbral más alto, el modelo debe estar más seguro para etiquetar una muestra como positiva, lo que reduce los falsos positivos pero aumenta los falsos negativos. Esto resalta la importancia de utilizar métricas como la *balanced accuracy* en lugar del *accuracy* tradicional en problemas con gran desbalanceo de clases.

Finalmente, el modelo con mayor área bajo la curva ROC es el *XGBoost* entrenado con las dos primeras componentes principales (0,63). Por tanto, este modelo presenta nuevamente la mejor capacidad para distinguir entre las clases.

6.5.1. Elección del mejor modelo

Tras comparar los cuatro modelos basados en árboles y analizar cómo clasificaban cada una de las clases, se concluyó que el último modelo probado (*XGBoost* junto con PCA) fue el que ofreció los mejores resultados.

Aunque su *accuracy* en el conjunto de *test* fue similar al obtenido con el modelo de *Decision Tree* basado en PCA, su *balanced accuracy* resultó ligeramente superior, lo que indica una mejor capacidad para distinguir entre clases y mayor robustez ante el *overfitting*. Además, los mejores resultados obtenidos en la curva ROC y en la distribución de probabilidades confirmaron esta mayor capacidad de clasificación en el problema de detección de condensación en los inversores.

No obstante, las métricas obtenidas con este modelo aún presentan un amplio margen de mejora. La escasez de muestras etiquetadas con la clase positiva junto con el gran desbalanceo de clases provocan que el modelo no sea capaz de generalizar bien tendiendo a etiquetar los casos dudosos como comportamiento sin fallo. Por ello, este modelo no es lo suficientemente robusto para poder desplegarlo en un entorno productivo real.

Si en el futuro se dispone de un mayor número de muestras de inversores con el fallo de condensación, sería recomendable volver a entrenar el modelo y analizar si las métricas tanto en *train* como en *test*, especialmente el *recall*, mejoran significativamente. Asimismo, si se cuenta con mayores recursos computacionales, sería interesante entrenar el modelo con la totalidad de las variables para evaluar su ajuste, lo cuál no fue posible durante el trabajo debido a los límites de memoria de las sesiones de `Google Collab`.

Por todo ello, resulta necesario continuar avanzando en el tratamiento y procesado de las muestras de los inversores, así como explorar modelos más complejos —por ejemplo, basados en técnicas de *deep learning*— para incrementar el rendimiento alcanzado.

Capítulo 7

Conclusión del trabajo

En este último capítulo se presentan las principales conclusiones del trabajo, así como las posibles líneas de mejora para incrementar la robustez del modelo y ampliar sus casos de uso.

7.1. Conclusiones

En este trabajo de fin de máster se ha ayudado a una empresa del sector eléctrico a desarrollar un modelo de clasificación para la detección del problema de condensación en los inversores solares. Una vez completadas todas las etapas del proyecto, se pueden extraer varias conclusiones relevantes:

1. Definición del problema: Hasta la mitad del proyecto no se centró la atención específicamente en el problema de condensación. En un principio, el objetivo era desarrollar un modelo de clasificación para detectar fallos genéricos en los inversores. No obstante, la dificultad de obtener etiquetas fiables de comportamientos anómalos llevó a reconducir el objetivo hacia la detección de un fallo concreto.

2. Interpretación de las variables temporales y de las alarmas: Esta fue una de las etapas más relevantes del trabajo, junto con el preprocesado de los datos. Los archivos CSV contenían información de numerosas variables y alarmas, lo que hizo necesarias varias reuniones con el personal técnico de la empresa

para identificar las más relevantes y comprender en detalle el problema de la condensación.

3. Recolección de los datos: Aunque el sistema de los inversores permitía obtener los archivos CSV con las variables temporales y el histórico de alarmas, el proceso resultó ser ineficiente. Era necesario acudir presencialmente a los distintos parques para realizar la recolección, lo que retrasó en gran medida el desarrollo de los modelos. En un futuro próximo, la empresa tiene previsto implementar *dataloggers* para facilitar la recolección remota y agilizar este proceso.

4. Preprocesado de los datos: Los archivos CSV incluían columnas innecesarias o vacías, lo que dificultó su correcta lectura. Además, como se disponía de un histórico de alarmas y no de un registro sincronizado exactamente con las muestras de las variables temporales, fue necesario aplicar un filtrado por fecha. Por otro lado, para cada tipo de modelo los datos debieron transformarse a formatos distintos: `numpy arrays` para PCA y *clustering*, y `pandas dataframes` para clasificación.

5. PCA y modelos de clustering: La reducción de dimensionalidad fue clave para entrenar los modelos de forma eficiente y simplificar las variables. Los modelos de *clustering* permitieron identificar dos grupos representativos de comportamiento normal y anómalo. Aunque el mejor modelo, GMM, no alcanzó métricas excelentes, resultó útil para visualizar los datos y comprobar que, a partir de las variables disponibles, era posible detectar patrones diferenciadores.

6. Etiquetado de los datos en función del problema de condensación: Poner el foco en el problema de la condensación permitió etiquetar los datos de forma fiable según la aparición de la alarma 721. Esto no solo facilitó el desarrollo de un modelo de clasificación, sino que también permitió obtener métricas que ayudaron a evaluar su robustez y fiabilidad.

7. Modelos de clasificación: Dado que la empresa no había implementado previamente técnicas de *machine learning*, se decidió comenzar por métodos con cierto grado de interpretabilidad, con el fin de facilitar la comprensión de su funcionamiento. Aunque las métricas del mejor modelo fueron modestas, el trabajo sienta una base sobre la que se podrá seguir mejorando la clasificación del problema de condensación en los inversores.

7.2. Líneas de mejora y trabajo futuro

Los avances conseguidos en este trabajo han sentado las bases para que la empresa continúe desarrollando nuevos modelos y avance en su transición hacia un sistema de mantenimiento predictivo.

A partir de este punto, se identifican diversas líneas de mejora que podrían abordarse en el futuro con el objetivo de desarrollar un modelo de clasificación más robusto y superar las limitaciones detectadas en este trabajo:

1. Implementación de dataloggers en los inversores solares: Esto permitirá agilizar la recolección de datos y evitar desplazamientos presenciales a los parques solares.

2. Recolección de más muestras etiquetadas con fallo de condensación: Incrementar el número de ejemplos con etiqueta positiva en el conjunto de entrenamiento permitirá a los modelos aprender patrones más representativos, dado que en este trabajo solo se contaba con 7 inversores etiquetados con el problema de condensación en este conjunto. Además, disponer de un mayor número de muestras positivas en el conjunto de prueba facilitará una evaluación más fiable y representativa de la capacidad de generalización del modelo.

3. Implementación de técnicas no lineales de reducción de dimensionalidad: En este trabajo se utilizó PCA que es una técnica lineal pero los datos podrían presentar relaciones no lineales que esta técnica no podría capturar. Por tanto, sería recomendable explorar métodos como *kernel PCA* [26] o *Independent Component Analysis* (ICA) [65] para mejorar la representación de los datos.

4. Incorporación de nuevas técnicas de interpretabilidad: Dado el auge en el campo de la interpretabilidad en IA, sería beneficioso emplear métodos como DALEX [66] o SHAP [67] para comprender el comportamiento del modelo tanto a nivel global como local.

5. Exploración de técnicas de deep learning: En este trabajo se utilizaron modelos basados en árboles. Sin embargo, en tareas similares, modelos de *deep learning* como LSTM o redes neuronales recurrentes (RNN), han mostrado buenos resultados y podrían ser objeto de implementación para el caso de uso del trabajo.

6. Aumento de recursos computacionales: La disponibilidad de *hardware* más potente y con GPU así como el uso de herramientas de *software* de procesamiento distribuido como `Spark` o la librería `PySpark` de `Python`, permitirá entrenar modelos más complejos y manejar mayores volúmenes de datos.

7. Desarrollo de una plataforma online: Esta plataforma permitiría la incorporación de datos de los inversores en tiempo real y la detección inmediata de posibles problemas de condensación.

8. Identificación y clasificación de nuevos problemas en los inversores: Detectar y etiquetar nuevos tipos de fallos para desarrollar modelos específicos que ayuden a su diagnóstico.

9. Desarrollo de un modelo para clasificación de fallos genéricos: Esto requeriría un etiquetado más exhaustivo basado en alarmas y tiempos de ocurrencia. Aunque es un reto mayor, aportaría un valor añadido para la empresa.

Si bien algunas mejoras son más sencillas de implementar que otras, todas contribuirán a perfeccionar los modelos de clasificación de condensación, ampliar la metodología a nuevos problemas de los inversores y, en definitiva, optimizar el mantenimiento de los inversores solares por parte de la empresa.

7.3. Conclusión final

Este trabajo de fin de máster ha permitido a Gamesa Electric dar un primer paso sólido en el uso de técnicas de *machine learning* e *inteligencia artificial* para el desarrollo de un sistema de mantenimiento predictivo en sus inversores solares. A lo largo del proyecto se ha conseguido no solo entrenar y evaluar distintos modelos de clasificación, sino también establecer una metodología para la recolección, el procesado y el análisis de los datos procedentes de los inversores.

Uno de los principales logros ha sido acotar el problema a abordar. Inicialmente el objetivo era detectar fallos genéricos, pero la dificultad para obtener etiquetas fiables hizo necesario centrar los esfuerzos en un fallo concreto: la condensación, identificada mediante la alarma 721. Esto permitió etiquetar los datos de forma consistente y desarrollar modelos que, aunque con márgenes de mejora, demostraron ser capaces de detectar el problema en los inversores.

Se han explorado distintas aproximaciones, como la reducción de dimensionalidad mediante *PCA*, el uso de modelos de *clustering* para la exploración de patrones

de comportamiento y la implementación de modelos de clasificación basados en árboles. El análisis de los resultados mostró la importancia de usar métricas como la *balanced accuracy* en contextos con fuerte desbalanceo de clases, así como la necesidad de disponer de un mayor número de muestras etiquetadas para mejorar la capacidad de generalización de los modelos.

Además, se han identificado distintas líneas de mejora que permitirán seguir avanzando en el desarrollo de modelos más robustos: la recolección telemática de datos mediante *dataloggers*, el incremento del número de muestras etiquetadas o la implementación de modelos basados en *deep learning*, entre otras.

En definitiva, este trabajo ha sentado las bases para que la empresa continúe desarrollando soluciones basadas en *machine learning*, contribuyendo a la transición hacia un mantenimiento predictivo más eficiente. El camino iniciado abre nuevas oportunidades para aplicar estas tecnologías en la detección de otros tipos de fallos y en la optimización del mantenimiento de los inversores solares.

Bibliografía

[1] KPMG Ana Jiménez Carrillo. *La IA generativa por sectores*. https://www.tendencias.kpmg.es/2024/02/ia-generativa-sectores/. 2024.

[2] Iberdrola. *Mantenimiento predictivo: la técnica basada en datos clave para anticipar errores*. https://www.iberdrola.com/innovacion/mantenimiento-predictivo.

[3] Endesa. *La digitalización llega a los parques eólicos*. https://www.endesa.com/es/proyectos/todos-los-proyectos/transicion-energetica/renovables/digitalizacion-aerogeneradores-parques-eolicos.

[4] IndesIA. *I Foro IndesIA: el impacto de la Inteligencia Artificial en la industria*. https://www.indesia.org/i-foro-indesia-el-impacto-de-la-inteligencia-artificial-en-la-industria/.

[5] Naciones unidas. *Informe de los Objetivos de Desarrollo Sostenible*. https://unstats.un.org/sdgs/report/2023/The-Sustainable-Development-Goals-Report-2023_Spanish.pdf?_gl=1*1rumwib*_ga*NDA3Nzc2Nzk3LjE3NDc4NDEwNDI.*_ga_TK9BQL5X7Z*czE3NDc4NDEwNDEkbzEkZzEkdDE3NDc4NDEwOTkkajAkbDAkaDA.*_ga_SCSJZ3XC0L*czE3NDc4NDEwNDEkbzEkZzEkdDE3NDc4NDEwOTkkajAkbDAkaDA.. 2023.

[6] Asociación española para la digitalización. "La digitalización en el sector energía. Transformación tecnológica y energética." En: *Revista DigitalEs* (2019).

[7] Javier Quintana. *El impacto de las energías renovables sobre el precio mayorista de la electricidad*. https://www.bde.es/f/webbe/SES/Secciones/Publicaciones/InformesBoletinesRevistas/BoletinEconomico/24/T3/Fich/be2403-art09.pdf. 2024.

[8] Femke J. M. M. Nijsse et al. "The momentum of the solar energy transition". En: *Nature* (2023).

[9] Fernando J. de Sisternes, Jesse D. Jenkins y Audun Botterud. "The value of energy storage in decarbonizing the electricity sector". En: *ScienceDirect* (2016).

[10] Iberdrola. *Las baterías de ion de litio, fundamentales para el almacenamiento de energía*. https://www.iberdrola.com/innovacion/baterias-ion-litio.

[11] National Geographic Andrew Curry. *Cómo la guerra de Ucrania está acelerando la transición energética renovable en Alemania*. https://www.nationalgeographic.es/medio-ambiente/2022/05/como-la-guerra-de-ucrania-esta-acelerando-la-transicion-energetica-renovable-en-alemania.

[12] Comisión Europea. https://ec.europa.eu/economy_finance/recovery-and-resilience-scoreboard/green.html.

[13] Iberdrola. https://www.iberdrola.com/sala-comunicacion/noticias/detalle/duplicamos-nuestra-inversion-en-idi-hasta-alcanzar-los-4000-millones-de-euros-en-2030.

[14] Antonio Gomez-Exposito, Angel Arcos-Vargas y Francisco Gutierrez-Garcia. "On the potential contribution of rooftop PV to a sustainable electricity mix: The case of Spain". En: *ScienceDirect* (2020).

[15] Sanzana Tabassum et al. "Solar Energy in the United States: Development, Challenges and Future Prospects". En: *Energies* (2021).

[16] Dr. Jose Pablo Chaves Dr. Rafael Cossent et al. "La digitalización de las redes eléctricas de distribución en España". En: *Instituto de Investigación Tecnológica* (2021).

[17] Apache Hadoop. https://hadoop.apache.org.

[18] Gonzalo Álvarez, Ernesto Martínez y Dan Ezequiel Kröhling. "Bases para el desarrollo y aplicación de Gemelos Digitales en la industria de la energía eléctrica". En: *ResearchGate* (2023).

[19] Nabil Mchirgui et al. "The Applications and Challenges of Digital Twin Technology in Smart Grids: A Comprehensive Review". En: *MDPI* (2024).

[20] Francisco Boshell et al. "Innovation landscape for a renewable-powered future: solutions to integrate variable renewables". En: *ResearGate* (2019).

[21] Parlamento Europeo. https://www.boe.es/doue/2016/119/L00001-00088.pdf.

[22] Inés Uribe. https://secmotic.com/mantenimiento-industrial-correctivo-preventivo-y-predictivo/.

[23] SAP. https://www.sap.com/spain/products/scm/apm/what-is-predictive-maintenance.html.

[24] Marek Mole da et al. "From Corrective to Predictive Maintenance. A Review of Maintenance Approaches for the Power Industry". En: *MDPI* (2023).

[25] Guanrong Chen y Trung Tat Pham. *Introduction to fuzzy sets, fuzzy logic and fuzzy control systems*. CRC Press LLC, 2001.

[26] Trevor Hastie, Robert Tibshirani y Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2ª. Springer, 2017.

[27] Irfan Adam Zulfauzi et al. "Anomaly detection using K-Means and long-short term memory for predictive maintenance of large-scale solar (LSS) photovoltaic plant". En: *ScienceDirect* (2023).

[28] Vicky Zilvan et al. "Denoising Convolutional Variational Autoencoders-Based Feature Learning for Automatic Detection of Plant Diseases". En: *IEEE* (2020).

[29] Yasir Saleem Afridi, Kashif Ahmad y Laiq Hassan. "Artificial Intelligence Based Prognostic Maintenance of Renewable Energy Systems: A Review of Techniques, Challenges, and Future Research Directions". En: *ArXiv* (2021).

[30] Simon Jelley. https://www.forbes.com/councils/forbesbusinesscouncil/2023/11/16/garbage-in-garbage-out-the-role-of-data-management-in-effective-ai/.

[31] Michel Verleysen y Damien François. "The Curse of Dimensionality in Data Mining and Time Series Prediction". En: *ResearchGate* (2005).

[32] Saranya A. y Subhashini R. "A systematic review of Explainable Artificial Intelligence models and applications. Recent developments and future trends". En: *ScienceDirect* (2023).

[33] Rabia Saleem et al. "Explaining deep neural networks. A survey on the global interpretation methods". En: *ScienceDirect* (2022).

[34] Ioannis Zografopoulos, Juan Ospina y Charalambos Konstantinou. "Cyber Physical Energy Systems Security: Threat Modeling, Risk Assessment, Resources, Metrics, and Case Studies". En: *ResearchGate* (2021).

[35] Chee Wooi Ten, Manimaran Govindarasu y Chen Ching Liu. "Cybersecurity for electric power control and automation systems". En: *ResearchGate* (2007).

[36] Dubravko Miljkovic. "Review of novelty detection methods". En: *ResearchGate* (2010).

[37] Muhammad Salik Qureshi, Muhammad Usman Nawaz y Shayan Umar. "Machine Learning for Predictive Maintenance in Solar Farms Introduction". En: *ResearchGate* (2024).

[38] Fabiola Pereira y Carlos Silva. "Machine learning for monitoring and classification in inverters from solar photovoltaic energy plants". En: *ScienceDirect* (2024).

[39] Alessandro Betti et al. "Predictive maintenance in photovoltaic plants with a big data approach". En: *ArXiv* (2019).

[40] Zhenyu He et al. "Fault Prognostics for Photovoltaic Inverter Based on Fast Clustering Algorithm and Gaussian Mixture Model". En: *Energies* (2020).

[41] Yizhang Wang et al. "Density peak clustering algorithms: A review on the decade 2014–2023". En: *Science Direct* (2023).

[42] Alex Rodriguez y Alessandro Laio. "Clustering by fast search and find of density peaks". En: *Research Gate* (2014).

[43] M. L. Menéndez et al. "The Jensen-Shannon Divergence". En: *Science Direct* (1996).

[44] Agussalim Syamsuddina et al. "Predictive maintenance based on anomaly detection in photovoltaic system using SCADA data and machine learning". En: *Science Direct* (2024).

[45] Federico Landi et al. "Working Memory Connections for LSTM". En: *Science Direct* (2021).

[46] Umberto Michelucci. "An Introduction to Autoencoders". En: *ArXiv* (2022).

[47] Qiang Zhao et al. "A New PV Array Fault Diagnosis Method Using Fuzzy C-Mean Clustering and Fuzzy Membership Algorithm". En: *Energies* (2018).

[48] Robert Hecht-Nielsen. "Theory of the backpropagation neural network". En: *International 1989 Joint Conference on Neural Networks* (1989).

[49] Mingfeng Li. "Comprehensive Review of Backpropagation Neural Networks". En: *ResearchGate* (2024).

[50] Jupyter. https://docs.jupyter.org/en/latest/.

[51] Pandas. `https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html`.

[52] Jonathon Shlens. "A Tutorial on Principal Component Analysis". En: *ArXiv* (2014).

[53] I. T. Jolliffe. *Principal Component Analysis*. 2.ª ed. Springer, 2002.

[54] Edgar Dobriban y Art B. Owen. "Deterministic parallel analysis: An improved method for selecting factors and principal components". En: *ArXiv* (2017).

[55] University of Minessota Twin Cities. https://www-users.cse.umn.edu/~kumar001/dmbook/ch8.pdf.

[56] Scikit-learn. URL: %5Curl%7Bhttps://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html%7D.

[57] Peter J. ROUSSEEUW. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis". En: *Journal of Computational and Applied Mathematics* (1987).

[58] Jerónimo Rojas Díaz, Julio César Chavarro Porras y Ricardo Moreno Laverde. "Técnicas de lógica difusa aplicadas a la minería de datos". En: (2008).

[59] Scikit-fuzzy. https://pythonhosted.org/scikit-fuzzy/api/skfuzzy.cluster.html.

[60] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[61] Miin-Shen Yang, Chien-Yo Lai y Chih-Ying Lin. "A robust EM clustering algorithm for Gaussian mixture models". En: *Science Direct* (2012).

[62] Scikit-learn. *GaussianMixture*. https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html.

[63] Scikit-learn. *DecisionTreeClassifier*. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

[64] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2nd. O'Reilly Media, 2019.

[65] Aapo Hyvarinen, Juha Karhunen y Erkki Oja. *Independent Component Analysis*. John Wiley & Sons, 2001.

[66] https://github.com/ModelOriented/DALEX.

[67] https://shap.readthedocs.io/en/latest/index.html.