Universidad de Valladolid

FACULTAD DE CIENCIAS

TRABAJO FIN DE GRADO

Grado en Estadística

Diagnóstico de fallos en motores de bombas eléctricas sumergidas mediante técnicas avanzadas de clasificación supervisada

Autor: **Juan Rubio Gómez**

Tutor: **Miguel Alejandro Fernández Temprano**

2024

Resumen

Los motores eléctricos de inducción están ampliamente presentes en los procesos industriales. Por ello, la monitorización del estado de los motores de inducción se ha vuelto cada vez más importante en los últimos años. En este Trabajo de Fin de Grado (TFG) se emplearán estadísticos de orden superior obtenidos en mediciones de motores de inducción trifásicos en pozos de agua. Estos estadísticos se emplearán para desarrollar algoritmos predictivos que permitan la detección temprana de fallos en este tipo de motores, permitiendo la realización de reparaciones a tiempo y evitando detenciones de procesos y pérdidas tanto de productividad como de dinero. Para la construcción de estos algoritmos predictivos, se utilizarán diversas técnicas avanzadas de aprendizaje automático, como Análisis Discriminante Lineal, Árboles de Decisión, técnicas de ensamblajes como Bosques Aleatorios y métodos de Boosting.

Este TFG esta desarrollado sobre un caso real, por lo que emplea datos reales de una empresa. Por tanto, este TFG no solo aborda un problema práctico real de gran relevancia para la industria, sino que también aplica métodos avanzados de aprendizaje automático para proporcionar soluciones innovadoras y efectivas. La utilización de datos reales garantiza que los resultados obtenidos sean aplicables en un entorno industrial real, ofreciendo así un valor añadido significativo.

Abstract

Electric induction motors are widely present in industrial processes. Therefore, the condition monitoring of induction motors has become increasingly important in recent years. In this Final Degree Project higher order statistics obtained from measurements of three-phase induction motors in water wells will be used. These statistics will be used to develop predictive algorithms that allow the early detection of failures in this type of motors, allowing timely repairs and avoiding process stoppages and losses of both productivity and money. For the construction of these predictive algorithms, several advanced machine learning techniques will be used, such as Linear Discriminant Analysis, Decision Trees, ensemble techniques such as Random Forest and Boosting methods.

This project is developed on a real case, so it uses real data from a company. Therefore, this project not only addresses a real practical problem of high relevance to the industry, but also applies advanced machine learning methods to provide innovative and effective solutions. The use of real data ensures that the results obtained are applicable in a real industrial environment, thus providing significant added value.

Agradecimientos

Quiero agradecer de corazón a mi familia por su cariño y apoyo inquebrantable a lo largo de toda mi vida. También quiero agradecer a las personas más cercanas a mí por estar siempre conmigo.

Por último, quiero agracecer a Miguel Alejandro Fernández Temprano por sus enseñanzas, consejos y ayuda, además de guiarme a lo largo de este proyecto.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Contexto del Problema

Un motor de inducción, también conocido como motor asíncrono, es un tipo de motor eléctrico que opera con corriente alterna. En este tipo de motores, la corriente eléctrica necesaria para hacer girar el rotor se induce mediante la interacción electromagnética con el campo magnético rotativo generado por el devanado del estator. Una de las mayores ventajas de estos motores es su alta eficiencia, que puede llegar hasta el 97 %. Además, los motores de inducción trifásicos proporcionan arranque automático, es decir, pueden iniciarse sin necesidad de intervención manual, y un control de velocidad variable [1]. Debido a esto, son ampliamente utilizados en multitud de aplicaciones industriales, y por ello, en los último años está creciendo la importancia de la monitorización de su estado para detectar fallos. La prevención de estos fallos es un aspecto crítico, puesto que predecir un futuro fallo y repararlo a tiempo puede evitar grandes pérdidas de dinero.

En este TFG se emplearán estadísticos de orden superior provenientes de los datos reales (como se detalla en el capítulo 4) obtenidos en las mediciones sobre las distintas fases de los motores de inducción trifásicos relativos a ciertos pozos de agua. Antes de nada, puesto que los datos son reales, es necesario un preprocesamiento donde se prepararán los datos para el análisis, limpiando valores atípicos, tratando con valores ausentes, reduciendo la dimensionalidad y seleccionando las variables oportunas, entre otros tratamientos. Posteriormente, se utilizarán los conjuntos de datos resultantes para entrenar distintos clasificadores que permitan predecir y clasificar las distintas mediciones de motores de otros pozos en su correspondiente estado.

1.2. Objetivos

Los objetivos fundamentales de este proyecto son los siguientes:

1. Comprobar si la intensidad es equivalente a la tensión en el contexto del problema.

2. Comprobar si las distintas fases, tanto de intensidad como de tensión, son mutuamente equivalentes en el contexto del problema.

3. Estudiar si es necesario el uso de diferentes clasificadores dependiendo del tipo de fallo que presenta el motor de un pozo.

4. Desarrollar ciertos clasificadores que permitan conocer el estado en el que se encuentra el motor de un pozo a través de los estadísticos de orden superior generados a través de mediciones en alguna de las fases, tanto de tensión como de intensidad, así como el tipo de fallo presente en dicho motor.

1.3. Asignaturas Relacionadas

Durante la realización de este proyecto, ha sido fundamental aplicar los conocimientos adquiridos tanto en el Grado de Estadística principalmente, como en el Grado en Ingeniería Informática. Destacan las siguientes asignaturas relacionadas con los temas tratados en este TFG:

- Análisis de Datos y Análisis Multivariante, donde se estudian los problemas de reducción de dimensionalidad y de clasificación, así como el desarrollo de modelos predictivos y sus métricas relacionadas.

- Técnicas de Aprendizaje Automático y Minería de Datos, enfocadas en el uso de Python para tareas predictivas de problemas de clasificación y *boosting*.

- Computación Estadística, Fundamentos de Programación, Paradigmas de Programación, Estructuras de Datos y Algoritmos y Programación Orientada a Objetos, así como cualquier asignatura del grado que haya aportado al estudio de la programación y desarrollo de código para el análisis de datos.

1.4. Estructura

La estructura de ese documento, que también se puede apreciar en el *Índice General*, se puede esquematizar según los siguientes capítulos:

- **Capítulo 1. Introducción:** En este presente capítulo se introduce el proyecto de fin de grado, su contexto, motivación y objetivos, así como las asignaturas del grado relacionadas y la estructura de la actual memoria donde se documenta el proyecto.

- **Capítulo 2. Motores y Bombas Eléctricas:** Durante este capítulo se trata el contexto y los conceptos claves teóricos relacionados con el tema del proyecto: los motores de bombas eléctricas sumergidas y sus principales fallos.
- **Capítulo 3. Metodología y Análisis:** En este cuarto capítulo se exponen los modelos y análisis realizados para los diferentes conjuntos de datos, con una breve introducción teórica a esos mismos modelos y análisis.
- **Capítulo 4. Datos:** En este capítulo se exponen los datos empleados durante el proyecto, su naturaleza, su descripción, su tipo y toda característica relevante propia de ellos. También se indica el preprocesamiento realizado sobre los datos inicales.
- **Capítulo 5. Resultados:** Durante este capítulo se describen y detallan los resultados, evaluaciones e interpretaciones relativas a los modelos y análisis realizados en el proyecto.
- **Capítulo 6. Conclusiones y Trabajo Futuro:** En el trascurso de este capítulo se presentan las conclusiones obtenidas al finalizar el proyecto. Además, se destacan principales dificultades encontradas y se exploran posibles áreas de mejora y puertas abiertas para futuras expansiones del proyecto.
- **Bibliografía:** Se referencian de las fuentes de información utilizadas durante el proyecto.
- **Anexos:** Se proporciona material complementario y detallado que respalda y enriquece la comprensión del trabajo realizado. Este material consiste principalmente en gráficos y tablas complementarias prodecentes de los análisis realizados, imágenes de los motores y el código desarrollado durante el proyecto.

Capítulo 2

Motores y Bombas Eléctricas

En este capítulo se introduce brevemente los conceptos teóricos acerca de los motores eléctricos de bombas sumergibles para pozos profundos, así como los principales fallos que el deterioro provoca en ellos. Un motor sumergible es una máquina diseñada para operar bajo el agua, sumergida en pozos o cuerpos de agua. Estos motores consisten en tres componentes principales [2]:

- **Motor eléctrico**: este dispositivo convierte la energía eléctrica en energía mecánica que impulsa la bomba. Los motores sumergibles suelen estar sellados herméticamente para evitar la entrada de agua y garantizar su correcto funcionamiento.
- **Carcasa hermética**: protege al motor y al resto de componentes eléctricos de la corrosión y el daño causado por el agua, ya que se construye con materiales resistentes a la humedad y las condiciones subacuáticas.
- **Sistema de bombeo**: se encarga de llevar el agua desde el pozo hasta la superficie y consta de una serie de piezas que incluyen impulsores, difusores y un eje o rotor.

Estos motores sumergibles funcionan según el principio de desplazamiento positivo, empujando el agua hacia arriba en lugar de aspirarla hacia abajo, siendo muy efectivas para elevar agua desde profundidades considerables, como las de los pozos de los que trata este TFG, que extraen agua de distintos acuíferos profundos. Además, son altamente eficientes en términos de consumo de energía, proporcionan una larga vida útil y requieren un mantenimiento mínimo [2].

2.1. Funcionamiento

Las bombas sumergibles pueden extraer agua de acuíferos a profundidades de entre 50 y 500 metros. Para ello, el conjunto del motor, que trabaja sumergido en el acuífero, eleva el agua a un tanque a través de una tubería. Los ejes del motor están directamente acoplados y posicionados

verticalmente, con el motor ubicado debajo de la bomba y la entrada de agua situada entre ambos. Una cubierta metálica lo envuelve, obligando al agua a fluir entre la cubierta y la superficie del motor antes de ingresar a la entrada, enfriando externamente el motor (figura 2.1). El bastidor del motor, que es liso y sin disipadores de calor, se enfría internamente al llenarse con una mezcla de agua y glicol. Debido a que el pozo perforado para acceder al acuífero tiene un diámetro pequeño para reducir los costos de excavación, el motor debe tener un diámetro reducido y un aislamiento adecuado, como se ha comentado anteriormente, para operar bajo el agua. Por ello, para facilitar el diseño estrecho y alargado, se emplean motores de inducción con jaula de rotor de barras oblicuas, donde su longitud es considerablemente grande en comparación con su pequeño diámetro. En los extremos del rotor, conectando sus barras conductoras, se encuentran los anillos de los extremos (figura 2.1), que se encargan de asegurar la continuidad eléctrica entre las barras del rotor. Esto es esencial para el funcionamiento del motor de inducción, ya que permite la circulación de corrientes inducidas en el rotor, lo que genera el par necesario para la rotación. Estos pares se logran una alta potencia en un tamaño de motor reducido, provocando que la alta velocidad de rotación incremente la potencia hidráulica de salida. Este aspecto tiene una desventaja, y es que también aumenta el desgaste de los componentes, como veremos en la siguiente sección 2.2. Los motores son alimentados al menos con un arrancador suave para evitar picos de corriente al arrancar y es posible ajustar la velocidad de la bomba. Actualmente, todos los motores nuevos funcionan con convertidores de frecuencia, operando a una velocidad fija, por lo que pueden aplicarse distintas técnicas para el análisis de señales con fines de monitoreo [3].

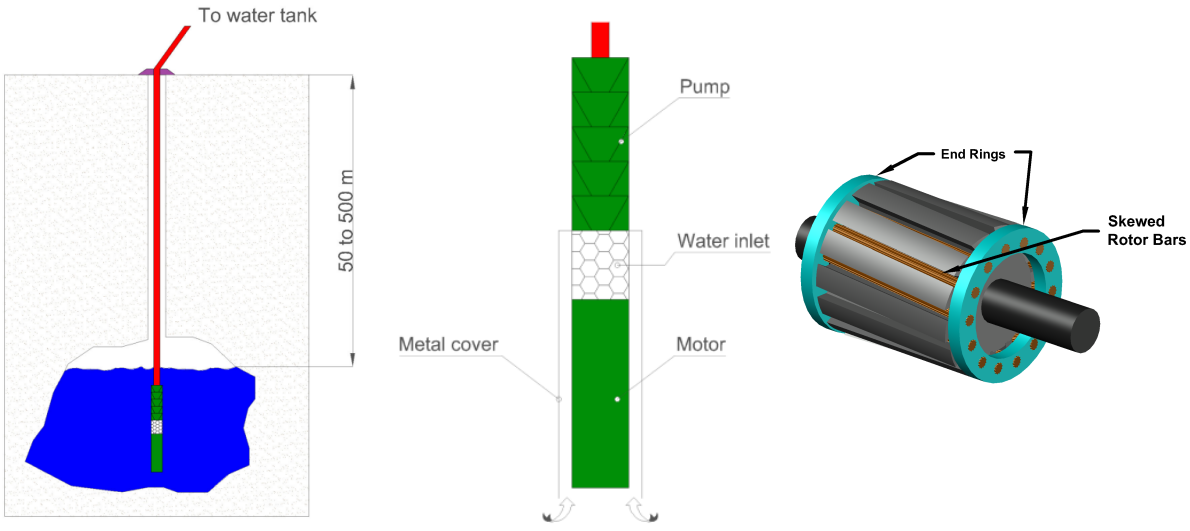


Figura 2.1: Esquema de las partes de un motor eléctrico sumergible [3] [4]

2.2. Fallos

En los motores convencionales, la falla del rotor principal viene dada por la rotura de las barras. Sin embargo, en los motores de pozo profundo, el principal causante de fallos se debe al desgaste de los anillos de los extremos. Estos anillos están hechos de varias láminas de cobre soldadas entre

sí, que son propensas a desgastarse debido al agua de refrigeración y a la rotación a alta velocidad, provocando pérdida de material en la soldadura y dejando al descubierto las láminas de cobre, que pueden desprenderse y causar mayores daños al motor. Existen varios tipos de fallas en los anillos terminales [3]:

- **Desgaste del material**: el desgaste gradual de los anillos supone una pérdida del material de soldadura que une las láminas de cobre, provocando un posible desprendimiento de estas láminas.
- **Pérdida de la soldadura**: consiste en el desgaste y erosión de la soldadura que mantiene unidas las láminas de cobre en los anillos de los extremos, aumentando las posibilidades del desprendimiento de las láminas de cobre.
- **Desprendimiento de las láminas de cobre**: finalmente, las láminas se desprenden y pueden causar daños graves, incluidos cortocircuitos o bloqueos mecánicos dentro del motor.
- **Calentamiento localizado**: la desigual distribución de la corriente, provocada por el desgaste y desprendimiento de láminas de cobre, forma puntos de mayor temperatura dentro del motor, degradando su aislamiento y otros componentes.
- **Interferencia electromagnética**: de nuevo, el desgaste y desprendimiento de láminas de cobre afectan la uniformidad del campo electromagnético, formando fuerzas electromagnéticas desiguales que aceleran el desgaste y pueden causar fallas en el motor.

Como se ha comentado en el capítulo introductorio, la detección temprana de estas fallas es fundamental para asegurar la productividad y buen funcionamiento del motor, aunque incluye notorias dificultades. El desgaste inicial es tan sutil que es muy dificil de detectar y genera la aparición de falsos negativos, además de que a medida que el desgaste se extiende uniformemente, las asimetrías iniciales se cancelan, dificultando su detección. Por ello, los métodos tradicionales para detectar fallas del rotor basados en la identificación de asimetrías no son efectivos para el desgaste del anillo terminal. Es necesario otra clase de enfoque para una mejor detección, como muestra el artículo [3]. Este artículo indica que, los experimentos realizados en él, confirman que las fallas en los anillos finales producen los mismos componentes armónicos en la corriente del estator que las fallas en las barras del rotor, empleando una combinación de corrientes de fase y otras magnitudes para mejorar la detección. Para esta detección se considera imprescindible el monitoreo continuo del motor, analizando los cambios en la amplitud de la LSH (*Lower Sideband Harmonic*) y RSH (*Reverse Sideband Harmonic*) a lo largo del tiempo, ya que estos pueden indicar la degradación de los anillos. En lugar de comparar la amplitud de LSH o RSH con los umbrales clásicos, parece resultar más efectivo observar las fluctuaciones en su amplitud. Realizar mediciones de vibraciones y emplear sensores de temperatura también es de gran utilidad para detectar este tipo de problemas y poder detener el motor en una etapa temprana de desgaste del anillo terminal, preveniniendo la contaminación del agua de refrigeración interna y los daños secundarios graves.

Capítulo 3

Metodología y Análisis

En este capítulo se describe la metodología empleada en este proyecto, contextualizando con una breve introducción teórica y mostrando los análisis y técnicas empleadas. La mayor parte de estos análisis y enfoques pertenecen al campo del aprendizaje automático o *machine learning*, cuya idea principal es es utilizar datos de observaciones pasadas para predecir resultados o valores desconocidos. Este concepto está basado en la estadística y el modelado matemático de datos, ya que básicamente un modelo de aprendizaje automático es una aplicación de *software* que encapsula una función para calcular un valor de salida en función de uno o más valores de entrada. Este proceso de definición de esa función se realiza sobre un conjunto apodado conjunto de entrenamiento (*train*). Una vez definida la función, esta se emplea para predecir nuevos valores en un proceso llamado inferencia.

3.1. Análisis de Componentes Principales

El Análisis de Componentes Principales (PCA) es un método estadístico que simplifica la complejidad de muestras multidimensionales al mismo tiempo que conserva, en la medida de lo posible, su información. Su objetivo principal es reducir la dimensionalidad del conjunto de datos analizado manteniendo la máxima cantidad posible de información. Considerando una muestra con n individuos, cada uno con p variables $(X_1, X_2, \ldots, X_p)$ el espacio muestral tiene p dimensiones. Este análisis identifica un número reducido de $z < p$ factores subyacentes que explican aproximadamente la misma variabilidad que las p variables originales. Estas nuevas z variables, llamadas componentes principales, son combinaciones lineales de las distintas variables originales que condensan la información de todas estas variables. Estas combinaciones lineales están ordenadas según el porcentaje de variabilidad explicada, de modo que la primera componente principal es la combinación lineal normalizada de dichas variables que tiene mayor varianza. Para extraer las componentes principales, es fundamental el cálculo de autovectores y autovalores sobre la matriz de correlaciones del conjunto, obteniendo así tantos pares de autovector-autovalor como dimensiones tiene el conjunto. Estos autovalores representan las direcciones de los ejes donde la varianza de

los datos es máxima y los autovalores indican cuánta varianza está contenida en cada componente principal. Siendo Z_i cada componente principal [5] [6]:

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \ldots + \phi_{p1}X_p \text{ con } \sum_{j=1}^{p} \phi_{j1}^2 = 1$$

Sobre el total de z componentes principales, el interés reside en seleccionar las m primeras componentes que expliquen una cantidad de variabilidad aceptable para el objetivo del análisis. Esta selección se puede basar en diferentes criterios, entre los que destacan:

- **Criterio del porcentaje de varianza explicada**: se puede fijar un porcentaje de variabilidad acumulada, como puede ser el 90 %, y seleccionar las primeras componentes que alcancen ese porcentaje.
- **Criterio del codo o del *Scree plot***: se grafica la varianza explicada por cada componente principal y se observa dónde se produce un quiebre abrupto en la pendiente de la curva, al que se denomina codo. Todas las componentes anteriores al codo se seleccionan.
- **Criterio del autovalor**: se seleccionan las componentes con autovalores mayores que 1, basándose en la idea de que los autovalores representan la cantidad de varianza explicada por cada componente, considerando que los componentes con autovalores mayores que 1 son significativos [7].

Por último, cabe resaltar que la detección y eliminación de outliers es un aspecto crítico en este análisis, al igual que estandarizar los datos, ya que en caso contrario variables con mayor escala pueden dominar sobre las demás.

3.2. Aprendizaje Automático No Supervisado

El aprendizaje automático no supervisado implica entrenar modelos utilizando datos que consisten únicamente en los valores de las variables (X), sin etiquetas conocidas. Este tipo de algoritmos determinan las relaciones existentes entre las características de las observaciones en los datos de entrenamiento [8].

3.2.1. Clustering

El *clustering* o la agrupación en clústeres es una técnica de aprendizaje automático no supervisado que agrupa observaciones similares en un conjunto de datos. Su objetivo principal es dividir los datos (sin etiquetar) en diferentes grupos, de modo que los datos que sean similares caigan en

el mismo grupo o clúster que aquellos que difieren de los demás, diferenciando datos con rasgos similares y asignándolos en grupos. Esto permite conocer la estructura del conjunto de datos total, mostrando posibles categorías en los que los datos están agrupados [9]. Según cómo se realice la formación de los clústers, estos métodos se dividen en dos distinciones [10]:

- **Métodos Jeráquicos**: estos métodos van generando grupos en cada una de las fases del proceso, buscando el número de clústeres que hacen una agrupación óptima. Por ello, no requieren una especificación previa del número de clústeres, siendo capaz de fijarlo el propio algortimo. Tienen gran utilidad como análisis exploratorio para posteriormente aplicar un análisis no jerárquico con el número de clústeres ya fijado. Existen dos estrategias, fácilmente representables con un dendograma:
  - **Estrategia Aglomerativa**: se parte de cada punto como un clúster individual y fusiona iterativamente los clústeres más cercanos hasta que todos los puntos están en un único clúster. Esta estrategia se puede representar con un dendrograma. Destacan los siguientes métodos:
    - ***Single Linkage*** o vecino más próximo, donde los clústeres se fusionan considerando la distancia más corta entre cualquier par de puntos de diferentes clústeres, provocando clústeres alargados.
    - ***Complete Linkage*** o vecino más lejano, fusiona los clústeres basándose en la distancia más larga entre cualquier par de puntos pertenecientes a diferentes clústeres, provocando clústeres más compactos y menos alargados.
    - ***Average Linkage*** o agrupación promedio, calcula la distancia promedio entre todos par de puntos para fusionarlos, provocando clústeres más equilibrados y menos sesgados por valores extremos.
    - ***Ward Linkage*** o de mínima varianza, se basa en minimizar la varianza total dentro de cada clúster al realizar una fusión. Esto a produce clústeres más compactos y homogéneos en términos de varianza, pero computacionalmente es más costoso.
  - **Estrategia Divisiva**: comienza con todos los puntos en un único clúster y se divide iterativamente los clústeres en subclústeres más pequeños, hasta que cada punto está en su propio clúster.
- **Métodos No Jerárquicos**: Estos métodos categorizan los elementos según un número de clústeres dado, por lo que a diferencia de los métodos jerárquicos sí necesitan que el número de particiones esté fijado a priori. Entre los algoritmos más empleados en el *clustering* no jeráquico destacan:
  - ***K-Means***: basado en centroides, divide los datos en K grupos, asignando cada punto de datos al grupo cuyo centroide (centro del clúster) esté más cercano. Es rápido y eficiente y puede converger a un mínimo local dependiendo de los centroides iniciales.
  - ***K-Medoids***: variante de *K-means* que utiliza puntos de datos reales (*medoids*) como representantes de los grupos en lugar de centroides. Un *medoid* es el punto real del

conjunto de datos más cercano al centroide de un clúster. Ete algortimo es más robusto frente a ruido y valores atípicos, pero también es más costoso computacionalmente.

- ***DBSCAN***: sus siglas corresponden a *Density-Based Spatial Clustering of Applications with Noise* y es un método de agrupamiento basado en densidad, que agrupa puntos de datos cercanos formando regiones de alta densidad e identificando valores atípicos en regiones de baja densidad. Para ello, emplea dos parámetros constantes importantes: *epsilon*, que define el radio para buscar puntos vecinos, y *min_pts*, el número mínimo de puntos para formar una región densa. Con ello, clasifica los puntos en tres categorías:
  - **Puntos centrales**: un punto se considera central si hay al menos *min_pts* puntos (incluyéndolo a él mismo) en su vecindario.
  - **Puntos frontera**: un punto es frontera si no es un punto central pero está en el vecindario de algún punto central.
  - **Puntos de ruido**: todo punto que no sea central ni frontera se considera punto de ruido (*outlier*).

  Por tanto, este algortimo es capaz de detectar *outliers*, es robusto frente al ruido y puede descubrir grupos de forma arbitraria, pero también puede tener dificultades con grupos de densidades variables o datos de alta dimensionalidad.

3.3. Aprendizaje Automático Supervisado

El aprendizaje automático supervisado implica, a diferencia del no supervisado, que los datos de entrenamiento incluyan tanto valores de etiquetas o *labels* (Y) conocidas como valores de características o *features* (X). Estas etiquetas clasifican las observaciones en diferentes grupos para algoritmos de clasificación, que se denominan binarios cuando hay solo dos categorías (0 o negativo y 1 o positivo), o multiclases cuando hay más de dos categorías. Además, también existen algoritmos de regresión, donde en vez de predecir un grupo o categoría la etiqueta predicha por el modelo es un valor numérico. Resumiendo, el aprendizaje automático supervisado se utiliza para entrenar modelos determinando una relación entre las características y las etiquetas en observaciones pasadas, de modo que se puedan predecir etiquetas desconocidas para características en casos futuros [8].

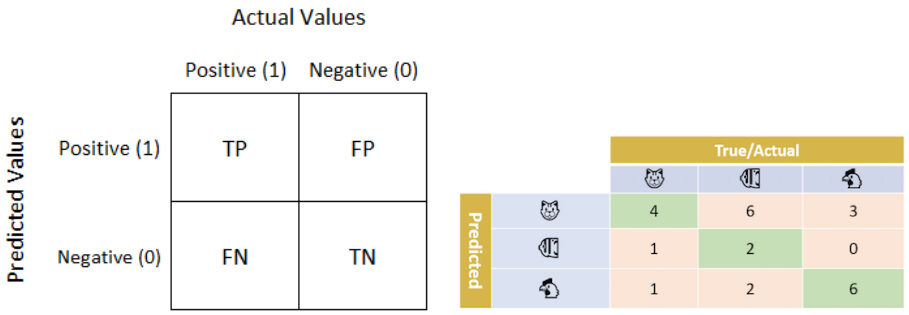
3.3.1. Análisis Discriminante Lineal

El Análisis Discriminante Lineal (LDA) es una técnica de aprendizaje supervisado principlamente utilizado para la clasificación, aunque también puede emplearse para la reducción de la dimensionalidad. La idea principal de se basa en encontrar una combinación lineal de características que separe mejor las diferentes categorías de las observaciones, intenta maximizar la varianza entre clases y minimizar la varianza dentro de la clase de los datos proyectados. Esta técnica tiene

varias ventajas, ya que al reducir la dimensionalidad optimiza el rendimiento de la clasificación eliminando variables irrelevantes o redundantes. Además, mejora la separabilidad entre clases, provocando decisiones más precisas y reduce tanto el costo computacional como la complejidad de los modelos. También facilita la visualización e interpretación de los resultados ya que proporcionaa una representación de menor dimensionalidad, conservando la estructura de clases [11]. A la hora de evaluar su rendimiento para medir la eficacia del modelo LDA para predecir las etiquetas de la clase correspondiente, se pueden utilizar las siguientes métricas [12]:

- **Matriz de confusión**: es una representación matricial de los resultados de las predicciones. En una dimensión de la tabla se muestran los valores predichos y en la otra dimensión los reales. Según el tipo de clasificacón que estemos llevando a cabo, la tabla tendrá diferente estructura, como se muestra en la figura 3.1. En el caso de clasificación binaria, se establecen los siguientes valores, mostrados en la figura 3.1 [13]:
  - **True Positive (TP)**: observaciones que el algoritmo clasifica como positivas y que realmente son positivas.
  - **True Negative (TN)**: observaciones que el algoritmo clasifica como negativas y que realmente son negativas.
  - **False Positive (FP)**: observaciones que el algoritmo clasifica como positivas cuando realmente son negativas.
  - **False Negative (FN)**: observaciones que el algoritmo clasifica como negativas cuando realmente son positivas.

  Con estas medidas se pueden calcular distintas tasas, como la tasa de falsos positivos (FPR), la tasa de verdaderos negativos (TNR), la tasa de verdaderos positivos (TPR) y la tasa de falsos negativos (FNR).



(a) Clasificación binaria (b) Clasificación multiclase

Figura 3.1: Matrices de confusión [13]

En la clasificación multiclases, la diagonal de la matriz indica la cantidad de observaciones que han sido predichas correctamente, mientras que los valores fuera de esa diagonal corresponden a observaciones clasificadas como una clase cuando realmente pertenecen a otra.

- ***Accuracy***: es la proporción de predicciones que el modelo acierta.

$$Accuracy = \frac{TN + TP}{TN + FN + FP + TP}$$

- ***Recall***: también llamada sensibilidad, mide la proporción de casos positivos reales que el modelo identificó correctamente.

$$Recall = \frac{TP}{TP + FN}$$

- ***Precision***: mide la proporción de casos positivos predichos donde la etiqueta real es positiva. Es otro enfoque de la métrica *recall*. Considerar tanto la capacidad para identificar correctamente los casos positivos como su habilidad para evitar falsos positivos.

$$Precision = \frac{TP}{TP + FP}$$

- ***F1-Score***: es una métrica general que combina las métricas *recall* y *precision*.

$$F1 - Score = 2 * \frac{Recall * Precision}{Recall + Precision}$$

- ***AUC* (Area Under the Curve)**: mide el área encerrada bajo la curva ROC (*Receiver Operating Characteristic*), de área máxima igual a uno. Esta curva es una representación de la tasa de verdaderos positivos (TPR) frente a la tasa de falsos positivos (FPR) y muestra cómo varía la capacidad de un modelo para discriminar entre clases positivas y negativas a medida que se ajusta un umbral de clasificación.

Por último, es importante recalcar que el LDA es un algoritmo paramétrico, por lo que asume dos asunciones fundamentales acerca de la distribución de los datos: **normalidad** y **homogeneidad** de varianzas. Estas dos asunciones o condiciones indican que los datos siguen una distribución normal multivariante, es decir, cada predictor que forma parte del modelo se distribuye de forma normal en cada una de las clases de la variable respuesta. Además, las clases tienen matrices de covarianza iguales, ya que la varianza del predictor es igual en todas las clases de la variable respuesta. Estas condiciones no son del todo indispensables, ya que si no se cumplen, el LDA pierde precisión pero aun así puede llegar a clasificaciones relativamente buenas.

3.3.2. Árboles de Decisión

Un árbol de decisión es un algoritmo de aprendizaje supervisado que se utiliza tanto para tareas de clasificación como de regresión. Es un algoritmo no paramétrico, por lo que no hace suposiciones específicas sobre la distribución de los datos. El algoritmo consta de una estructura de árbol jerárquica, que se compone de un nodo raíz, ramas, nodos internos y nodos hoja. Este algoritmo

permite una fácil representación que muestra la toma de decisiones y cómo se llega a clasificar una observación en un grupo u otro. Para la clasificación de observaciones, se sigue el camino desde el nodo raíz hasta una hoja, basándose en los valores de las características y las condiciones establecidas en cada nodo, asignando así la clase mayoritaria entre las observaciones de entrenamiento asociadas a esa hoja. Cuando un árnol crece demasiado en profundidad, puede dar lugar a *overfitting* o sobreajustes, lo que implica que el modelo se ajuste demasiado bien a los datos de entrenamiento, resultando en un rendimiento deficiente cuando se enfrenta a datos nuevos. Como criterio de división para selecionar el mejor atributo en cada nodo, existen principalmente dos métodos: la ganancia de información y la impureza de Gini. El criterio de ganancia de información está estrechamente relacionado con la entropía, que cuantifica la impureza de un conjunto de datos. Este criterio mide la reducción en la entropía al dividir los datos en subconjuntos basados en un atributo, por lo que un mayor valor de ganancia de información indica que el atributo es más relevante para la clasificación. En cuanto a la impureza de Gini, esta es la probabilidad de clasificar incorrectamente un punto de datos aleatorio en el conjunto de datos si se etiquetara en función de la distribución de clases del propio conjunto. Se busca minimizar la impureza de Gini en los nodos hojas para maximizar la homogeneidad de las clases en cada subconjunto.

En resumen, los árboles de decisión son fáciles de interpretar, no requieren ninguna preparación de los datos y son insensibles a las relaciones subyacentes entre los atributos, por lo que si dos variables están altamente correlacionadas, el algoritmo solo elegirá una de las características para realizar la división. Además, son muy flexibles y pueden emplearse para tareas de clasificación y de regresión. También presenta algunas limitaciones, ya que son altamente propensos al sobreajuste, más cosotosos computacionalmente que otros algoritmos y son muy susceptibles a estimadores de alta varianza, por lo que pequeñas variaciones en los datos pueden producir un árbol de decisión totalmente diferente [14].

3.3.3. Bagging: Random Forest

Un bosque aleatorio o *random forest* es un ensamble de bagging de clasificadores de aprendizaje automático. Un ensamble es una técnica que combina múltiples modelos de aprendizaje para mejorar el rendimiento general del sistema, basándose en la idea principal de que la combinación de varios modelos puede producir predicciones más precisas que cualquier modelo individual. El término *bagging* (*Bootstrap Aggregating*) indica el tipo de contrucción que tiene este ensamble, que en este caso consiste en entrenar múltiples modelos independientes utilizando diferentes subconjuntos aleatorios del conjunto de datos de entrenamiento para luego promediar o combinar sus predicciones. Estos subconjuntos de entrenamiento son generadas mediante muestreo con reemplazamiento llamado *bootstrap* y cada modelo en el ensamble tiene igual peso en la predicción final. Al predecir, se promedian las predicciones de todos los modelos para obtener la predicción final.

Los clasificadores individuales suelen ser clasificadores débiles, y en el caso de los bosques alea-

torios son árboles de decisión débiles o *stumps*, por lo que el ensamble combina el resultado de múltiples árboles para alcanzar un único resultado. Como se ha comentado en el apartado 3.3.2, individualmente estos árboles de decisión son altamente inestables y pueden inducir a sesgos y sobreajustes. Sin embargo, cuando varios árboles de decisión forman un conjunto en un bosque aleatorio, predicen resultados más precisos, reduciendo ampliamente estos problemas si los árboles son independientes y no están correlacionados entre sí, ya que el promedio de árboles no correlacionados reduce la varianza general y el error de predicción. Para lograr esto se emplea el ya mencionado *bagging* y la aletoriedad para generar un subconjunto aleatorio de características, garantizando una baja correlación entre los árboles de decisión.

Por tanto, la principal diferencia entre los árboles de decisión y los bosques aleatorios consiste en que, mientras los árboles de decisión consideran todas las posibles divisiones de características, los bosques aleatorios solo seleccionan un subconjunto de esas características y emplean múltiples árboles de decisión para combinar sus resultados en uno global. Esto permite reducir el riesgo de sobreajuste, obtener una gran facilidad para determinar la importancia de las características en el modelo y, al igual que los árboles de decisión, ofrece flexibilidad para emplearse tanto en modelos de regresión como en modelos de clasificación. En cuanto a sus limitaciones, requiere aún más recursos, tanto de tiempo como de computación, que los árboles de decisión, además de complicar su interpretación [15] [16].

3.3.4. Boosting: XG-Boost

El *boosting* es una metodología de aprendizaje automático en la que se combinan muchos modelos obtenidos mediante clasificadores con poca capacidad predictiva para dar lugar a un mejor predictor general. Por lo tanto, el *boosting* consiste, al igual que *bagging*, en un ensamble de clasificadores débiles, como árboles de decisión con poca profundidad. La principal diferencia entre el clasificador *XG-Boost* con los bosques aleatorios reside en la construcción del ensamble, es decir, en la diferencia entre una construcción bagging y una *boosting*. Como se comenta en la sección 3.3.3, el enfoque *bagging* entrena los clasificadores en paralelo, mientras que en boosting se entrenan secuencialmente, ya que cada modelo del ensamble se centra en corregir las observaciones mal clasificadas por los modelos anteriores. Esto se logra gracias a que, en cada iteración, los datos de entrenamiento se ponderan, dándole más peso a las instancias que fueron mal clasificadas, permitiendo combinar modelos débiles para formar un modelo más fuerte. Hay varios tipos de clasificadores *boosting*, entre los que destacan [17] [18] [19]:

- **AdaBoost** o ***Adaptative Boosting***: este método funciona de forma iterativa, identificando puntos de datos mal clasificados y ajustando sus pesos para minimizar el error de entrenamiento. El modelo continúa optimizándose de manera secuencial hasta que obtiene el predictor más fuerte. Es posible que este ensamble sea sensible al ruido y los valores atípicos.

- **Gradient Boosting**: en lugar de cambiar el peso de los puntos de datos como AdaBoost, este ensamble se centra en entrenar los errores residuales del clasificador anterior, ajustando un nuevo árbol de decisión para predecir los residuos (diferencias entre las predicciones actuales y los valores reales). Para ello, va agregando secuencialmente clasificadores a un conjunto en el que cada uno de ellos corrige los errores de su predecesor. Este enfoque es menos sensible al ruido que AdaBoost y generalmente ofrece un mejor rendimiento en la práctica.

- **XGBoost** o ***Extreme Gradient Boosting***: es una implementación de *Gradient Boosting* diseñada para mejorar la velocidad y la escalabilidad computacional. El algoritmo XGBoost emplea múltiples núcleos en la CPU, que permiten que el aprendizaje ocurra en paralelo durante el entrenamiento, mejorando su eficiencia y rendimiento. Además de la paralelización, emplea técnicas como la regularización para evitar el sobreajuste y ofrece una amplia gama de hiperparámetros que permiten una mayor personalización del modelo.

Exiten muchos otros tipos de algoritmos *boosting* como **LightGBM** (*Light Gradient Boosting Machine*), de especial utilidad para conjuntos de datos muy grandes y con muchas características, o como **CatBoost**, que permite mejorar el rendimiento en conjuntos de datos que contienen variables categóricas gracias a *target encoding*, una técnica que permite la codificación de características [20].

Capítulo 4

Datos

En este capítulo se explica la procedencia de los datos empleados en el proyecto, así como sus características y atributos. También, se muestra y detalla el preprocesamiento previo a los análisis realizados, relacionado con la ingeniería de las características propias de los datos.

Los datos usados en este TFG provienen de casos reales de la empresa FACSA. Esta empresa española, ubicada en Castellón, está dedicada a ofrecer todos los servicios propios del ciclo integral del agua, desde su captación, potabilización y tratamiento hasta su distribución y posterior recogida y depuración de las aguas residuales [21]. Los datos han sido facilicitados, en colaboración con la Universidad Politécnica de Valencia (UPV), por el Departamento de Ingeniería Eléctrica de la Escuela de Ingenierías Industriales de la Universidad de Valladolid (UVa), más concretamente por Óscar Duque Pérez, director de este departamento. Estas dos universidades se han encargado de realizar las mediciones, cálculos previos y cargas necesarias para obtener los conjuntos de datos recibidos, en el formato .xlsx.

El código perteneciente a todos el preprocesamiento realizado se puede encontrar en los anexos, en el apéndice B, en la sección B.2.

4.1. Descripción

El conjunto total de datos consta de hasta seis conjuntos de datos independientes, cada uno perteneciente a un motor de un pozo específico. Por ello, se cuenta con observaciones de un mismo pozo a lo largo del tiempo. Cada uno de estos seis conjuntos de datos consta de hasta seis subconjuntos, equivalentes a cada una de las tres fases (A, B y C) tanto de intensidad como de tensión, en relación con la naturaleza trifásica del motor. A su vez, cada uno de estos subconjuntos cuenta con 27 variables, de las cuales 17 pertencen a los estadísticos de orden superior obtenidos sobre la distribución del conjunto de observaciones correspondientes. Otra variable indica la fecha y hora de medida de la observación y las 9 restantes corresponden a los valores normalizados de

algunas de las variables anteriores. Se ha comprobado que los análisis y modelos desarrollados usando estas variables normalizadas producen resultados muy similares a los obtenidos con las variables sin normalizar. Por ello, estas 9 variables normalizadas se han descartado y no se tendrán en cuenta de ahora en adelante.

De los seis pozos estudiados, existe una idea inicial de que los pozos Aljibes, Pedrizas1V y Pedrizas2A presentaron fallos en sus motores. El pozo de Aljibes no tuvo ningún fallo al inicio, pero con el transcurso del tiempo fue sufriendo un deterioro del material de los anillos terminales hasta un resultado fatal, en el que quedó inutilizable. La figura A.1, en los anexos, muestra el estado de sus anillos y se puede apreciar como ambos anillos extremos han perdido la mayor parte del cobre de soldadura, exponiendo las láminas y barras de cobre, destacando el desgaste en el anillo superior. Por otro lado, Pedrizas1V tuvo problemas por el desgaste provocado por un aumento en la temperatura y Pedrizas2A presentó un fallo de excentricidad en el rotor, probablemente desde el inicio de las mediciones. Las imágenes reales del motor de Pedrizas2A pueden encontrarse en los anexos, en la figura A.2. Los pozos Guillamón, Pellicer y Quintana, supuestamente, no mostraron ningún tipo de problemas o fallos. Un resumen de los distintos subconjuntos de datos para cada conjunto o pozo, se muestra en la tabla 4.1. Cabe destacar que para el conjunto Pedrizas1V no ha logrado obtener ningún dato para la fase C, tanto en intensidad como en tensión.

$Conjunto_i$	Intensidad	Fase A	$intensidadA\_pozo_i$
		Fase B	$intensidadB\_pozo_i$
		Fase C	$intensidadC\_pozo_i$
	Tensión	Fase A	$tensionA\_pozo_i$
		Fase B	$tensionB\_pozo_i$
		Fase C	$tensionC\_pozo_i$

Tabla 4.1: Subconjuntos de datos para cada conjunto/pozo

Además, la tabla 4.2 muestra un resumen de cada pozo, mostrando el número de observaciones, el estado en el que supuestamente se encuentraba y, en caso de que haya tenido fallos, el tipo de fallo:

Pozo	**Nº observaciones**	**Fallo**	**Descripción del fallo**
Aljibes	1005	SÍ	Anillos terminales deteriorados
Pedrizas1V	275	SÍ	Desgaste por exceso de temperatura
Pedrizas2A	416	SÍ	Fallo de excentricidad en el rotor
Pellicer	877	NO	Sin fallos
Guillamón	1085	NO	Sin fallos
Quintana	402	NO	Sin fallos

Tabla 4.2: Información relativa a cada pozo

Como se ha comentado anteriormente, para cada pozo se cuenta con 18 variables, las cuales pertenecen a ciertos estadísticos de orden superior propios de cada conjunto. En la figura 4.3, se muestran estas 18 variables obtenidas para cada subconjunto:

Variable	Descripción	Fórmula
Nombre	Fecha y hora de la medición	dd/mm/aa_hh/mm/ss
Valor_pico	Valor más alto	$\max \lvert x_i \rvert$
Valor_medio	Valor medio (m1, c1)	$\frac{1}{n}\sum_{i=1}^{n} x_i$
Valor_medio_abs	Valor medio de los absolutos	$\frac{1}{n}\sum_{i=1}^{n} \lvert x_i \rvert$
Valor_medio_cdr	Valor cuadrático medio	$\frac{1}{n}\sum_{i=1}^{n} x_i^2$
Valor_RMS	Raíz de la media cuadrática	$\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$
Momento_2	Varianza (m2)	$\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$
Momento_3	Momento de orden 3 (m3)	$\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3$
Momento_4	Momento de orden 4 (m4)	$\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^4$
Cumulante_2	Cumulante de orden 2 (c2)	$m_2 - m_1^2$
Cumulante_3	Cumulante de orden 3 (c3)	$m_3 - 3m_2m_1 + 2m_1^3$
Cumulante_4	Cumulante de orden 3 (c4)	$m_4 - 4m_3m_1 - 3m_2^2 + 12m_2m_1^2 - 6m_1^4$
Skewness	Coeficiente de asimetría	$\frac{m_3}{\sqrt{m_2}^3}$
Kurtosis	Coeficiente de apuntalamiento	$\frac{m_4}{{m_2}^2}$
Factor_cresta	Factor de cresta	$\frac{x_p}{\sqrt{m_2}}$, donde $x_p = \max \lvert x_i \rvert$
Factor_forma	Factor de forma	$\frac{n\sqrt{m_2}}{\sum_{i=1}^{n} \lvert x_i \rvert}$
Factor_borde	Factor de borde	$\frac{Valor\_RMS}{Valor\_medio\_cuadratico}$
Factor_afinidad	Factor de afinidad	$\frac{Valor\_pico}{Valor\_medio\_absoluto}$

Tabla 4.3: Listado de variables de cada conjunto/pozo

Hasta ahora, ninguna de las variables indica la etiqueta que permite conocer la categoría a la que pertenece una observación. Para lograr esto, se dispone de tres indicadores únicamente para el pozo de Aljibes, que deben ser interpretados, como se ve la sección 4.2.5, para asignar la etiqueta correspondiente a cada observación. Estos tres indicadores han sido facilitados en documentos .xlsx aparte, por lo que han sido relacionados con el conjunto de características de cada pozo con

un *inner-join*. Estos tres indicadores provienen de conceptos relacionados con la modulación de amplitud, una técnica utilizada en la transmisión de señales de radio [3], como se comenta en la sección 2.2:

- **Indicador 1**: medida LSH (Lower Sideband Harmonic), se refiere a las frecuencias más bajas en la banda lateral inferior generadas por la modulación de amplitud.
- **Indicador 2**: medida RSH (Reverse Sideband Harmonic) más 3. El concepto RSH se refiere a las frecuencias más bajas que se encuentran en la banda lateral superior.
- **Indicador 3**: medida RSH (Reverse Sideband Harmonic) menos 3.

También se cuenta con el indicador 1 (LSH) para Pedrizas2A, pero no se han podido relacionar con las observaciones de este mismo pozo debido a que las medidas de este indicador no coinciden ni en fecha ni en hora con las mediciones realizadas para extraer el conjunto de características del pozo. Esto hace que únicamente se cuente con la categoría real de uno de los pozos, el de Aljibes. Debido a esto, la línea de trabajo a seguir durante este proyecto consiste en emplear Aljibes para construir un clasificador que permita predecir el estado de cada una de las observaciones y, con ello, el estado general del resto de los motores. Las predicciones de este clasificador sobre el resto de pozos se deben comparar con la información, en principio correcta, acerca del estado general del pozo. También, estas predicciones deben ser consistentes en los pozos que no dieron ningún problema y, queda en el aire la duda de si podrá generalizarse el clasificador para los pozos Pedrizas2A y Pedrizas1V, ya que los motores de estos pozos tienen otro tipo de fallos diferentes al de Aljibes, motor con el que se entrena el clasificador.

Como se ha comentado, la primera variable 'Nombre' indica la fecha y hora en la que se realizó la medición en un pozo. Esta variable es por tanto común a todas las fases dentro de un mismo pozo, tanto de tensión como de intensidad, ya que las mediciones se realizaron simultáneamente para todas las fases. Estas mediciones se realizaron en un estado estacionario, lo que significa que el motor no solo había iniciado el bombeo de agua, sino que también había operado durante el tiempo suficiente para que la señal registrada fuera estable. Hay varias mediciones realizadas durante un mismo día en distintas horas, así como también hay días en las que no se realizó ninguna medición. Ante la falta de información sobre el porqué de estas ausencias de mediciones durante varios días, se ha supuesto que esos días no fue necesario extraer agua, y por tanto arrancar el motor, por lo que no se realizó ninguna medición. La figura 4.1 muestra el número de mediciones realizadas por día durante el periodo de medición del motor del pozo Aljibes. Se puede apreciar que hay algún día ausente de mediciones durante los meses comprendidos entre junio y octubre, destacando el periodo final de agosto. Además, la mayoría de días muestran al menos cuatro mediciones realizadas. Las mediciones realizadas durante un mismo día deberían clasificarse, salvo excepción, con el mismo estado, mientras que largos periodos sin recibir mediciones podrían producir cambios repentinos en el estado de las observaciones del motor. El equivalente a este gráfico para el resto de pozos se encuentra en los anexos en las figuras A.3, A.4, A.5, A.6 y A.7, para los pozos de Pedrizas2A,

Pedrizas1V, Pellicer, Guillamón y Quintana, respectivamente. Con estos gráficos podemos ver como todos estos pozos tienen grandes periodos de inactividad durante sus mediciones, por lo que no hay regularidad temporal en los datos, lo cual puede dificultar la tarea de clasificación. Estos periodos pueden ser de días, como en Aljibes o Pedrizas2A, de meses o incluso de años, como en Pedrizas1V.

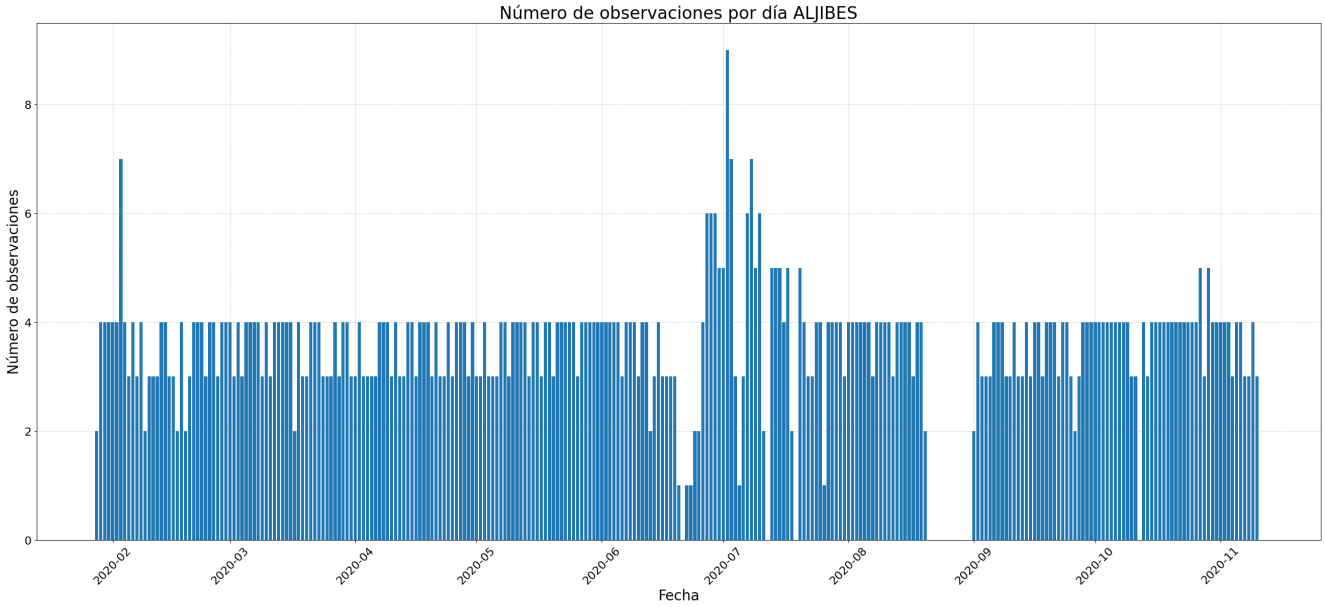


Figura 4.1: Número de mediciones por día para Aljibes

4.2. Preprocesamiento

Una vez descritos los datos y explicada su procedencia es estrictamente necesario llevar a cabo un preprocesamiento de los datos. Realizar un preprocesamiento de datos consiste en aplicar un conjunto de técnicas y pasos sobre los datos antes de ser utilizados para análisis o modelado, teniendo como objetivo principal limpiar y preparar los datos para que sean más adecuados y efectivos para su posterior análisis. Las actividades de este preprocesamiento suelen ser integración, limpieza, manejo de valores faltantes, discretización, reducción de dimensionalidad, selección de variables o características y transformación. Todos estos pasos han sido realizados, debido a que al tratarse de datos reales y a su naturaleza, la carga de trabajo en el preprocesamiento ha sido elevada.

4.2.1. Selección Inicial de Conjuntos

Antes de nada, es importante centrarse en alguno de los subconjuntos posibles de datos, eligiendo un conjunto como principal en el que basar el proyecto, entre las distintas fases de intensidad y de tensión. En principio, todas las fases deberían ser equivalentes entre sí para un mismo pozo, y tampoco debería hacer ninguna diferencia significativa entre elegir intensidad o tensión. Aún

así, en caso de que no fueran equivalentes, desde el Departamento de Ingeniería Eléctrica de la Universidad de Valladolid se ha facilitado cierta información que concluye que la intensidad es más relevante que la tensión, por lo que el proyecto se centrará en la intensidad. A la hora de seleccionar en qué fase enfocar el proyecto, por lo que se puede apreciar en los anexos en el apartado A.2.3, las fases A y B parecen comportarse de manera similar en la intensidad, siguiendo los patrones y cambios similares, mientras que la fase C es bastante distinta a las otras dos. Los gráficos mostrados en este apartado de los anexos contienen preprocesamiento posterior a este presente apartado, como la eliminación de valores atípicos, que se muestra en la sección 4.2.3. Estos gráficos invitan a pensar que existen diferencias entre tensión e intensidad y que las fases no son equivalentes entre sí, como se pensaba en un principio. Debido a que elegir la fase C supondría excluir los datos del pozo Pedrizas1V, puesto que no se dispone de los datos asociados a la fase C, la elección de fase es entre las fases A y B. Inicialmente, el análisis se centrará sobre la fase A de intensidad y será el conjunto sometido a los análisis durante este preprocesamiento. De forma simultánea, se mostrará el preprocesamiento de otras fases tanto de intensidad como de tensión, ya que también se someterán a los distintos algoritmos de aprendizaje automático para comprobar si los resultados son equivalentes a los obtenidos con la fase A de intensidad.

4.2.2. Integración de Datos

Como se ha comentado anteriormente, el conjunto de datos total consta de hasta seis pozos, cada uno con sus variables para cada una de las tres fases tanto en tensión como en intensidad. Todos ellos, en un archivo .xlsx con una hoja por cada fase han sido leídos como conjuntos independientes. Además, cada indicador se ha proporcionado en otro archivo .xlsx independiente, por lo que ha sido necesario integrarlos al conjuto de variables correspondiente con un *inner-join* a través de la variable 'Nombre' presente en ambos conjuntos. No todas las observaciones cruzan con los indicadores, por lo que hay valores ausentes en algunos valores de los indicadores, como se detalla en la sección 4.2.4.

4.2.3. Limpieza de Datos

Se ha llevado a cabo una exhaustiva detección de valores atípicos para cada fase de intensidad y de tensión en cada pozo. Esto ha implicado realizar la detección de outliers para hasta 34 conjuntos de datos. Este análisis se ha realizado visualmente, sobre gráficos descriptivos de líneas y de dispersión de los distintos conjuntos. En la mayoría de ocasiones, todas las fases de un mismo pozo, tanto en intensidad como en tensión, presentaban los mismos *outliers*. Debido al gran tamaño de los datos, la política acerca del tratamiento de *outliers* ha sido simplista y todos los *outliers* detectados visualmente han sido eliminados del conjunto de datos correspondiente. Estos valores atípicos detectados y eliminados para cada pozo se pueden observar en la tabla 4.4. Además, también se ha elimindado alguna observación puntual que presentaba valores extraños en ciertas variables. En total, se han detectado 41 *outliers* para Aljibes, 6 para Pedrizas1V, 15 para

Pedrizas2A, 24 para Pellicer, 45 para Guillamón y 15 para Quintana. Un ejemplo representativo de cómo estos valores atípicos afectaban a las variables correspondiente se muestra en la figura 4.2. Este ejemplo, hecho sobre la variable 'Valor_medio_absoluto' de las fases de intensidad pertenecientes al motor del pozo Aljibes, es representativo para el resto de pozos y variables.

	Outliers
Aljibes	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 15, 22, 23, 25, 27, 29, 75, 102, 103, 122, 145, 199, 246, 349, 381, 391, 401, 498, 536, 537, 540, 543, 558, 576, 651, 653, 760, 802, 837, 945
Pedrizas1V	60, 72, 154, 157, 179, 180
Pedrizas2A	4, 7, 10, 100, 191, 192, 200, 211, 225, 230, 261, 337, 338, 339, 386
Pellicer	0, 1, 2, 3, 4, 5, 6, 9, 18, 125, 180, 189, 226, 244, 278, 291, 382, 516, 536, 560, 698, 699, 752, 867
Guillamón	0, 7, 23, 47, 57, 69, 94, 136, 140, 156, 171, 173, 296, 361, 389, 424, 464, 476, 492, 562, 564, 572, 691, 694, 697, 714, 717, 733, 737, 748, 796, 801, 816, 892, 896, 901, 926, 927, 930, 980, 982, 1019, 1045, 1052, 1059
Quintana	15, 19, 22, 85, 126, 135, 198, 199, 201, 251, 257, 305, 350, 352, 401

Tabla 4.4: Resumen de los valores atípicos detectados por pozo

Figura 4.2: Influencia de los outliers en Aljibes

4.2.4. Tratamiento de Valores Ausentes

Ningún pozo presenta valores ausentes o NA's (*Not Available*) en su conjunto de características. Sin embargo, para el pozo de Aljibes, no todas las observaciones del conjunto de características están relacionadas con un valor del indicador, por lo que hay observaciones con valor Null o NA sus correspondientes indicadores. La tabla 4.5 muestra el número de valores ausentes para Aljibes, único pozo con indicadores y, con ello, valores ausentes.

	Indicador 1	Indicador 2	Indicador 3
Aljibes	195	208	208

Tabla 4.5: Resumen de la presencia de valores ausentes

Como solución a este problema, viendo la distribución de los valores ausentes para cada indicador, se ha tomado la decisión de interpolar dichos valores. Esta decisión basa en que la tendencia de la distribución de los indicadores parece mostrar dónde se encontrarían los valores reales de dichos indicadores, realizando así una aproximación y permitiendo no perder más observaciones, ya que hay un número significativo de NA's. Esta aproximación se ha realizado con un interpolado lineal, que rellena los valores faltantes calculando una línea recta entre los puntos de datos conocidos inmediatamente antes y después del valor faltante. Se ha decidido emplear el interpolado lineal en lugar de otros interpoladores más complejos debido a su simplicidad y a la aparente tendencia lineal de los datos. En la figura 4.3 se puede apreciar la distribución de los indicadores con los nuevos valores interpolados.

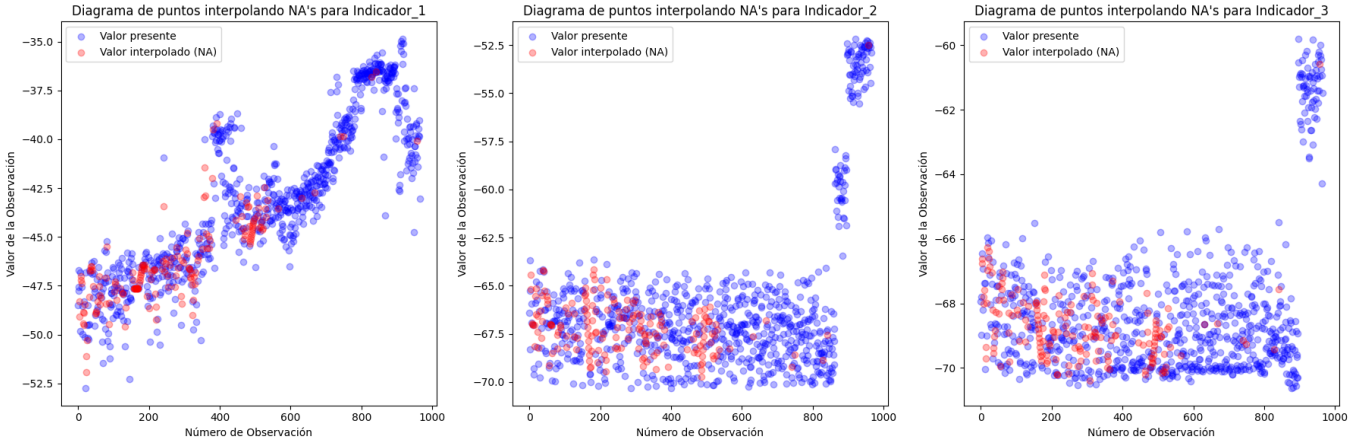


Figura 4.3: Interpolación de los valores ausentes en los indicadores de Aljibes

4.2.5. Discretización de la Categoría

Los tres indicadores existentes en Aljibes son de tipo continuo, es decir, los distintos indicadores toman valores númericos de tipo flotante. Para llevar a cabo la tarea de clasificiación requerida, es conveniente que estos valores sean categóricos o cualitativos, y estén distribuidos en distintos grupos o categorías. Para ello, hay que establecer puntos de corte entre los distintos valores de los indicadores para asignar a ciertos valores una categoría u otra. Para saber en cuántas categorías debemos agrupar los datos, se han realizado distintos análisis, llegando a las mismas conclusiones:

- **Visualización y dominio del problema**: antes de nada, es adecuado visualizar la distribución de los distintos indicadores para ver si a simple vista se pueden apreciar posibles estados del motor. Una posible elección de estados podría ser la mostrada en la figura 4.4,

donde se puede apreciar que los indicadores 1 y 2 separan en tres estados, mientras que el indicador 3 lo hace en solo dos. También se puede ver una gran similitud entre los indicadores 2 y 3, que parecen más coherentes con la previsible evolución del deterioro en el motor. El indicador 2 es aparentemente más sensible que el indicador 3, siendo capaz de detectar un estado intermedio. Esta discretización, además de basarse en la visible distribución de cada indicador, concuerda con los conocimientos extraídos del artículo [3].

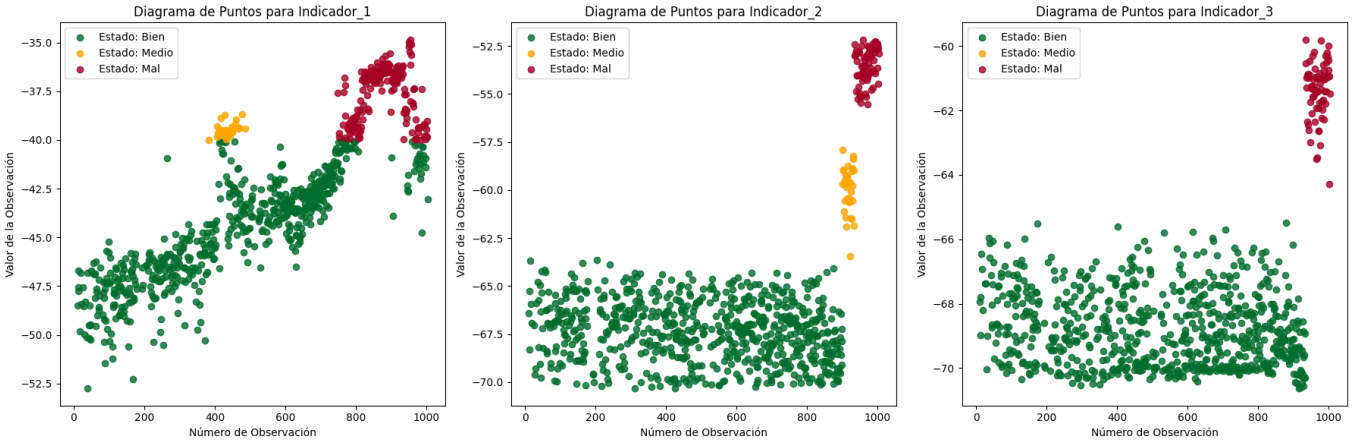


Figura 4.4: Distribución de los indicadores en Aljibes

Asignar tres como número de grupos o estados del motor permite clasificar los estados en bueno, malo o medio, lo cual tiene más interés que clasificarlos únicamente en buen o mal estado, ya que un estado medio puede preveer un próximo mal estado, prediciendo futuros fallos y evitando la indiponibilidad o el mal funcionamiento del motor.

- **Clustering**: utilizar técnicas de *clustering* antes de aplicar un algoritmo de clasificación supervisada puede ser una estrategia muy útil a la hora de comprender la estructura subyacente de los datos. En este caso, al no conocer los distintos estados en los que puede estar el motor, esto nos ayudará a ver en cuantos grupos se dividen los datos y poder decidir el número de estados de los motores. Se han realizado distintos métodos de *clustering* para agrupar el conjunto de características del motor del pozo Aljibes, todos ellos explicados en la sección 3.2.1. Los resultados obtenidos apoyan la idea anterior de seleccionar tres grupos, ya que el conjunto de características parece tener una óptima separación en tres clústeres. La figura 4.5 muestra un resumen de estos análisis. En ella se puede apreciar el método del codo y el índice de silueta para seleccionar el número óptimo de grupos. El método del codo se emplea para determinar el número óptimo de clústeres, buscando el punto o codo en el que se disminuye significativamente la componente Y en el gráfico. El índice de silueta calcula la cohesión intra-cluster y la separación inter-cluster, donde los valores cercanos a uno indican que una instancia está bien dentro de su propio clúster y lejos de otros clústeres. La figura 4.6 refleja cómo se agrupan las observaciones en los tres clústeres, obtenidos con KMeans con inicialización de centroides aleatoria, sobre la distribución de los distintos indicadores, donde se puede comprobar que los grupos formados según las similaridades de las observaciones en el conjunto de características coincide con la discretización anteriormente realizada según

el dominio del problema. Se puede apreciar que los valores que mejor distinguen estos tres clústeres son los propios del indicador 2, que detecta estas tres agrupaciones de manera clara y unequívoca. Como se puede ver los anexos A.2.5, este mismo *clustering* con otro método de inicialización supone otra asignación menos coherente con la naturaleza del problema. También en estos anexos (apéndice A.2.5) se puede apreciar las agrupaciones correspondientes al resto de análisis de *clustering* realizados, como son el aglomerativo, KMedoids y DBSCAN, produciendo todos conclusiones similares a los obtenidos por KMeans.

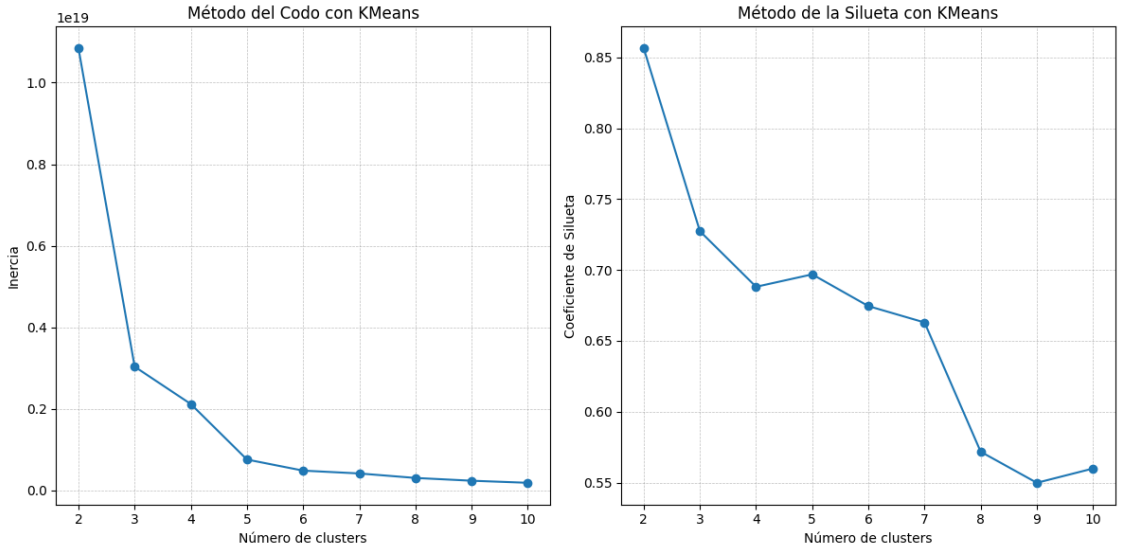


Figura 4.5: Resumen de los resultados del clustering

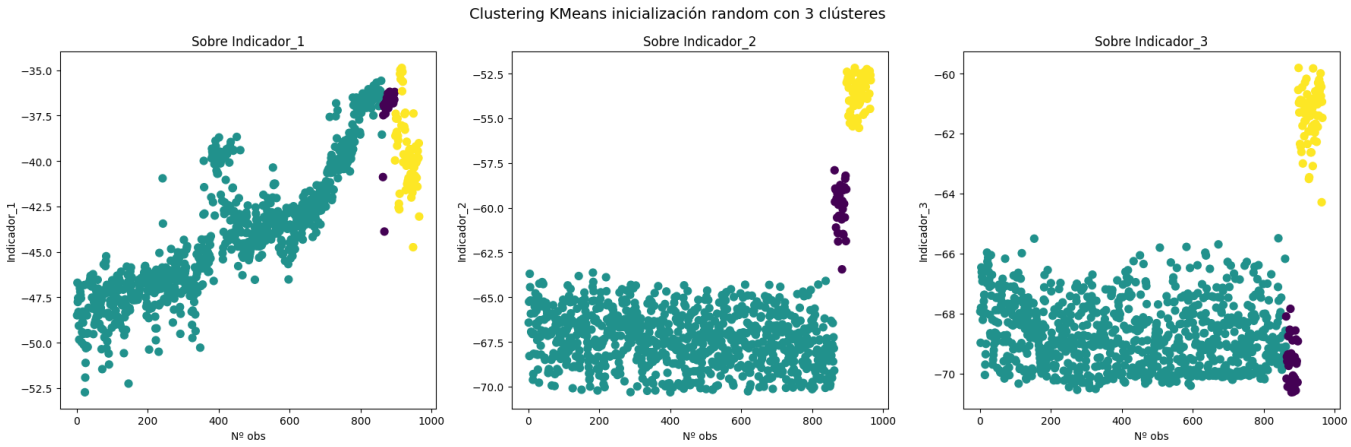


Figura 4.6: KMeans con inicialización aleatoria sobre Aljibes

- **Análisis de Componentes Principales**: otro análisis que refuerza la idea de tres estados es el ACP. Como se indica en el siguiente apartado 4.2.6, se emplea la técnica ACP para reducir la dimensionalidad del conjunto de características. En la figura 4.7, se puede observar que reduciendo el conjunto de características a únicamente tres dimensiones (o incluso dos), los datos ya son fácilmente separables para los indicadores 2 y 3. Por otro lado, el indicador 1 no parece separar bien los distintos estados en tan pocas dimensiones. La separación entre grupos es la misma que la establecida por el criterio visual y el dominio del problema.

Figura 4.7: Primeras componentes principales para los indicadores de Aljibes

Resumiendo, se ha optado por discretizar los indicadores en tres categorías para los indicadores 1 y 2, y en dos para el indicador 3. La tabla 4.6 muestra los valores de corte para cada estado:

	Bien	**Medio**	**Mal**
Indicador 1	$\text{Ind}_1 < -40$	$\text{Ind}_1 \geq -40$ y $n\_obs \leq 500$	$\text{Ind}_1 \geq -40$ y $n\_obs > 500$
Indicador 2	$\text{Ind}_2 < -63.5$	$-63.5 \leq \text{Ind}_2 < -57.5$	$\text{Ind}_2 \geq -57.5$
Indicador 3	$\text{Ind}_3 < -65$	-	$\text{Ind}_3 \geq -65$

Tabla 4.6: Discretización de indicadores en Aljibes

4.2.6. Reducción de Dimensionalidad

El conjunto de datos total de Aljibes cuenta con hasta 30 dimensiones por fase de intensidad y de tensión, de las cuales 27 pertenecen al conjunto de características y 3 a las etiquetas. De las 27 variables del conjunto de características, se desechan las 9 pertenecientes a valores normalizados, como ya se ha comentado y se detalla en el apartado 4.2.7. Las restantes, a excepción del identificador 'Nombre', se han sometido a un análisis de componentes principales. Los resultados de dicho análisis para intensidad fase A se pueden ver en la figuras 4.7 y 4.8, donde se aprecia que las dos primeras dimensiones son suficientes para diferenciar los datos en tres grupos, ademas de acumular el 77.55 % de la variabilidad total del conjunto. Con tres dimensiones podemos alcanzar hasta el 89.99 % de variabilidad.

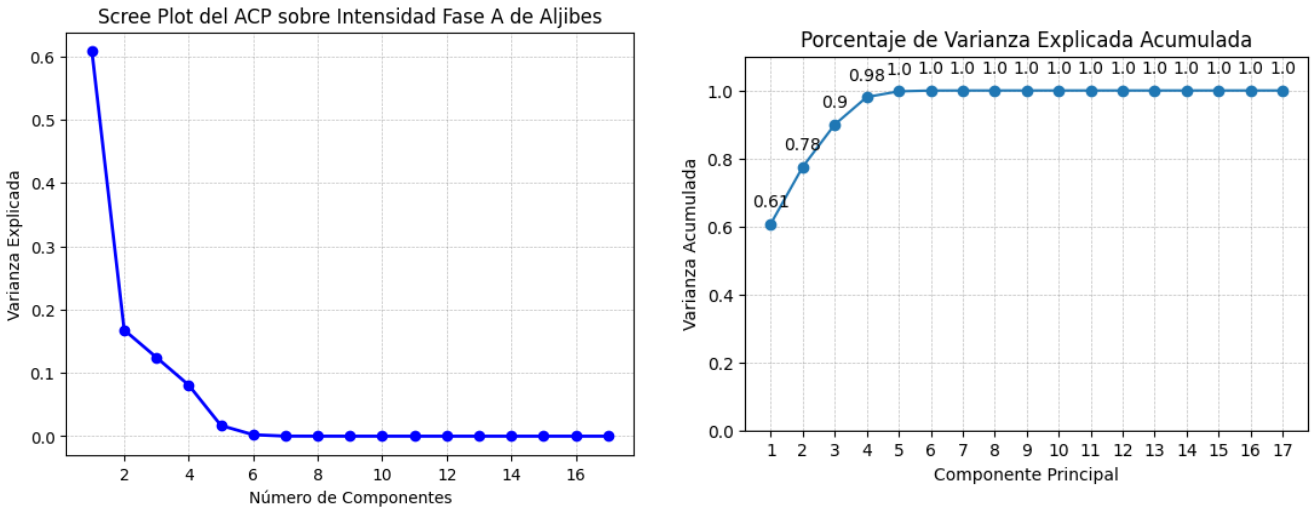


Figura 4.8: Varianza explicada por las primeras componentes para intensidad Fase A de Aljibes

Con estos resultados, es evidente que se puede reducir la dimensionalidad sin perder apenas información, alcanzando incluso el total de la variabilidad del conjunto entero en apenas cinco dimensiones.

4.2.7. Selección de Variables

La selección de variables debe realizarse tanto en el conjunto de características, como en la etiqueta, ya que se dispone de hasta tres indicadores que podrían ser empleados para catalogar las distintas observaciones.

Conjunto de Características

Como se ha visto en la sección anterior 4.2.6, está claro que se debe reducir la dimensionalidad del conjunto de características. En lugar de seleccionar componentes principales (combinaciones lineales de las variables originales) como nuevo conjunto de características, se ha optado por hacer una selección de variables debido a que facilita la interpretación de unos posteriores resultados en los distintos algoritmos de clasificación. Es importante recordar que no se tienen en cuenta las nueve de variables normalizadas, ya que fueron descartadas al inicio debido a que producían resultados muy similares a los obtenidos con las variables orginiles, además de mostrar los mismos patrones que estas.

Para selección de estas variables se ha tenido en cuenta la correlación entre las disintas variables del conjunto de características. Se han eliminado iterativamente las variables más correlacionadas, hasta mantener un número cercano a las dimensiones óptimas según el ACP anterior, teniendo en cuenta que las variables restantes sean incorreladas. La figura 4.9 muestra un mapa de calor indicando los valores absolutos de la matriz de correlación de las variables iniciales y de la selección final.

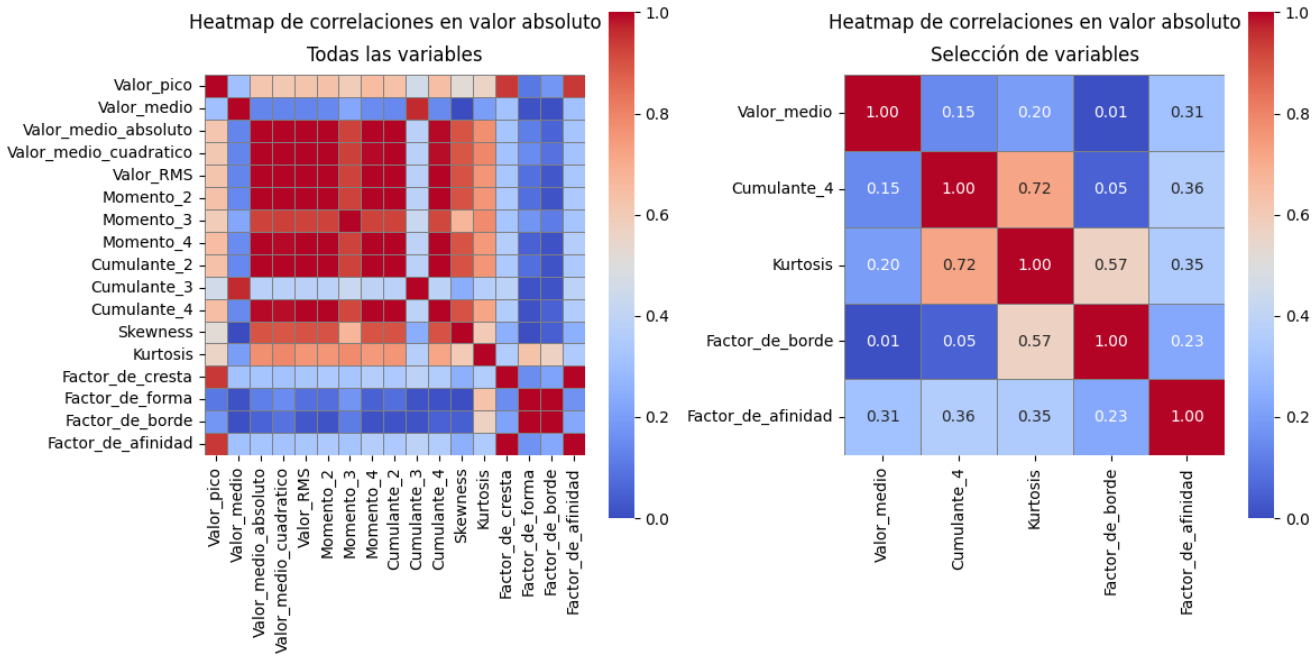


Figura 4.9: Mapa de calor para las variables de intensidad fase A de Aljibes

Por tanto, la selección de variables para la fase A de intensidad consiste en Valor_medio, Cumulante_4, Kurtosis, Factor_borde y Factor_afinidad. Se puede observar en los anexos, en la figura A.34, como otras fases de intensidad y tensión proporcionan otra selección de variables distinta, lo que refuerza la idea de que no exista una equivalencia estadística entre las distintas fases y entre tensión e intensidad. Esta selección es consistente con el gráfico de cargas proveniente del ACP (figura 4.10) realizado anteriormente, ya que las variables con altas correlaciones entre sí tienen aportes similares a las dos primeras componentes, las cuales acumulaban el 77.55 % de la variabilidad total del conjunto.

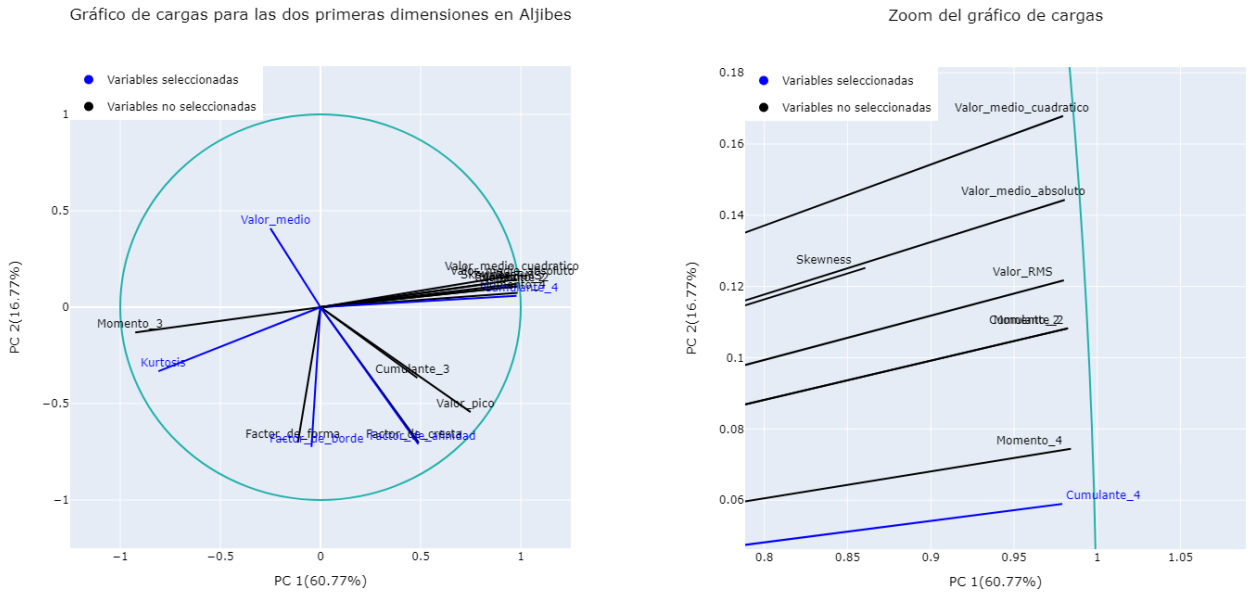


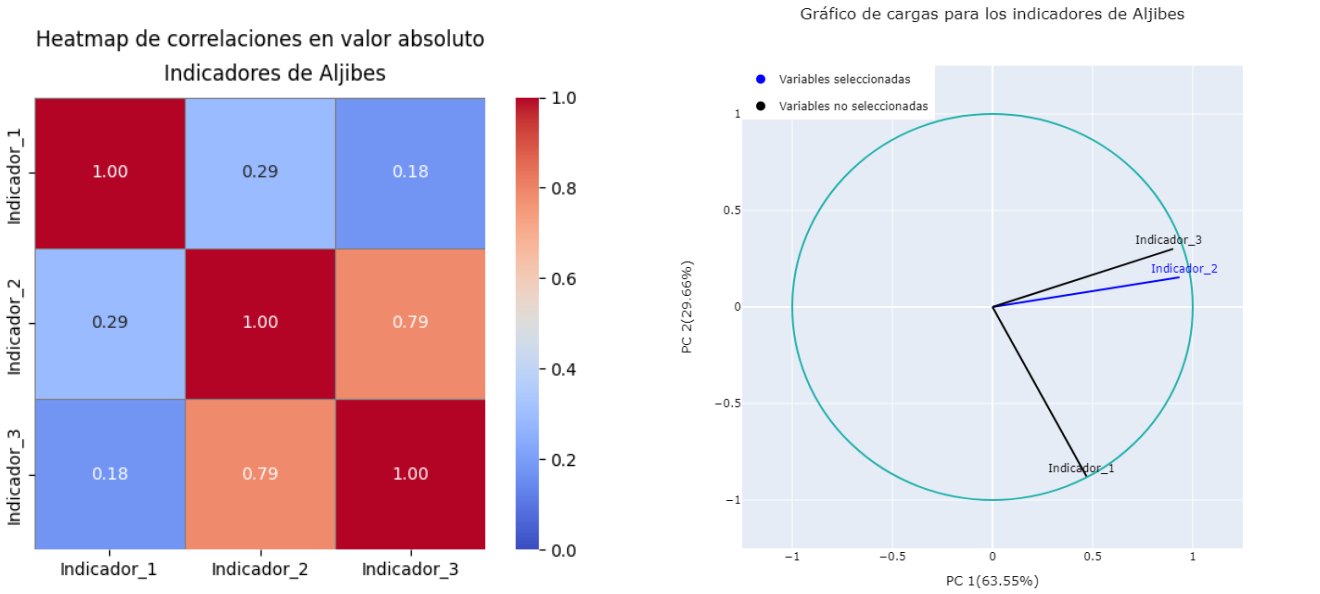
Figura 4.10: Gráfico de cargas del ACP sobre intensidad fase A de Aljibes

Es importante destacar que, en la fase A de intensidad, las correlaciones entre las variables seleccionadas en Aljibes presentan valores diferentes en comparación con los observados en el resto de los pozos, indiferentemente de su estado. Esto puede dificultar el aprendizaje y empeorar las predicciones en el resto de pozos, ya que diferencias en las correlaciones muestran que las relaciones entre las variables en Aljibes no se replican en los otros pozos, lo cual puede provocar que los modelos aprendan patrones que no son generalizables. Para el resto de fases, tanto de intensidad como de tensión, también ocurre esto. La sección A.2.6 del apéndice A.2 de los anexos, se pueden comparar las diferencias entre las correlaciones de las variables en la fase A de intensidad de Aljibes con las propias en el resto de pozos, destacando las variables seleccionadas anteriormente.

Indicadores

El estudio se centra en un solo indicador. Como ya se ha comentado y se ha visto en las distribuciones de la figura 4.4, los indicadores 2 y 3, que son ciertamente similares como se puede ver en las figuras 4.4 y 4.11, parecen más coherentes con la evolución del desgaste y deterioro del motor. Sin embargo, el indicador 1 muestra como antes de la mitad de las observaciones el motor

alcanza un estado 'Medio' para luego volver a un estado 'Bien', antes de deteriorarse demasiado y llegar al estado 'Mal'. Una vez llegado a ese estado 'Mal', finalmente vuelve a un estado 'Bien', lo que hace pensar que este indicador no mide el progresivo desgaste del motor como se necesita en este proyecto. También se ha podido ver como en la figura 4.7, en las dos primeras dimensiones los indicadores 2 y 3 separan los diferentes estados, mientras que el indicador 1 no lo logra ni en las tres primeras. Por otro lado, el indicador 2 parece indentificar una fase intermedia entre un estado bueno y malo, siendo más sensible que el identificador 3. Según estas conclusiones, el indicador seleccionado es el indicador 2. Esta decisión está apoyada también por los resultados obtenidos en el *clustering*, como se ha visto en la sección 4.2.5.



(a) Mapa de correlación de los indicadores

(b) Gráfico de cargas del ACP sobre los indicadores

Figura 4.11: Correlaciones y cargas de los indicadores de Aljibes

4.2.8. Transformación de Datos

Como muestran los diagramas de cajas por variable y la distribución de sus valores en las figuras de las seccines A.2.2 y A.2.3 de los anexos, las variables para cada pozo presentan distintos rangos de variabilidad y distintas distribuciones. Por ello se ha optado por normalizar los datos, dependiendo de la metodología empleada, realizando un escalado para que todos los datos se encuentren dentro del intervalo formado por [0, 1] o realizando una estandarización, para establecer la media en 0 y la desviación típica en 1.

$$X_{\text{escalado}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \quad \text{ó} \quad X_{\text{estandarizado}} = \frac{X - \text{media}(X)}{\text{desviación estándar}(X)}$$

Estandarizar suele ser más robusto frente a los *outliers* que escalar, anque la decisión de usar uno u otro dependerá del contexto de la metodología empleada. En caso de requerir una distribución comparable entre todas las variables, se deberá estadarizar, pero sin embargo, si se desea que todas las variables fluctúen en el mismo rango, se deberá escalar.

4.3. Conjunto de Datos Final

Concluyendo y resumiendo todo este proceso de preprocesamiento, destacamos las siguientes ideas acerca del conjunto de datos final empleado en este proyecto:

- Como únicamente se cuenta con los indicadores de Aljibes, la línea de trabajo principal se enfoca en tratar de crear clasificadores, entrenados con el subconjunto de la intensidad de fase A de Aljibes, que puedan predecir los diferentes estados de los otros cinco pozos, centrándose en los que idealmente están en buen estado, como son Guillamón, Pellicer y Quintana, e intentando generealizar estos clasificadores también a Pedrizas1V y a Pedrizas2A. Pese a la selección del subconjunto de la intensidad de fase A, a lo largo de este proyecto también se hablará de resultados procedentes de otras fases o de la tensión.
- Se ha reducido la dimensionalidad del conjunto de datos. La selección de variables sobre el subconjunto de la intensidad de fase A de Aljibes ha disminuido el total variables a seis: Valor_medio, Cumulante_4, Kurtosis, Factor_borde, Factor_afinidad y el Indicador_2.
- Se ha discretizado la vairable etiqueta Indicador_2 en tres categorías, que indican el estado del pozo en una observación concreta. Estas categorías son 'Bien', 'Medio' y 'Mal'.
- Se han limpiado los datos, eliminando los *outliers* o valores atípicos detectados visualmente y se han interpolado los valores ausentes para no perder observaciones. También se normalizarán los datos acorde al análisis o metodología aplicada.

Capítulo 5

Resultados

Durante este capítulo se detallan los resultados obtenidos con los análisis y métodologías realizadas. Como se comenta en el capítulo 4, se ha fijado el pozo de Aljibes para los conjuntos de datos empleados para el entrenamiento que se utilizan para ajustar el modelo, ya que es el único pozo con etiquetas en sus observaciones. Principalmente, los resultados se basan en la fase A de intensidad, como también se comenta en el capítulo 4, aunque alternativamente se detallan resultados procedentes de otras fases de intensidad y de tensión. Se han realizado análisis y se han obtenido resultados usando otros conjuntos como entrenamiento de los clasificadores, como la intensidad en fase C o la tensión en fase A, e incluso empleando las variables ya normalizadas comentadas en el capítulo 4, obteniendo resultados similares a los mostrados en este capítulo. Por este motivo, junto al concepto de garantizar la brevedad y concisión de la memoria y a las limitaciones de espacio impuestas, se ha tomado la decisión de excluir dichos análisis y resultados del documento. El código perteneciente a todos los análisis realizados se puede encontrar en el apéndice B de los anexos, en la sección B.3.

5.1. Análisis Discriminante Lineal

Inicialmente, se realiza un Análisis Discriminante Lineal empleando todas las variables. Sus resultados en Aljibes alcanzan un 100 % de tasa de aciertos en el entrenamiento y en el test, por lo que se toma la decisión de probar con la selección de variables realizada en el preprocesamiento con tal de evitar un posible sobreajuste. La selección de variables para la fase A de intensidad proporciona de nuevo un 100 % de tasa de aciertos tanto en entrenamiento como en test. A la hora de realizar el análisis, se escalan los datos a valores comprendidos entre cero y uno y se entrena y prueba el algoritmo con Validación Cruzada de 10 *folds* y con *Leave-One-Out (LOO)* para tratar de evitar el sobreajuste. La opción de LOO proporciona una estimación menos sesgada y maximiza el número de datos de entrenamiento y, al no contar con demasiadas observaciones, el coste computacional que supone no es muy elevado. Los resultados, mostrados en la figura 5.1, son iguales para ambas técnicas, obteniendo un valor máximo en todas las métricas analizadas,

tanto en el entrenamiento como en la prueba: *accuracy*, *recall*, *precision* y *f1-score*. Por tanto, los estados por los que pasa el motor del pozo de Aljibes parecen fácilmente separables linealmente.

		Valor medio
VC-10	Accuracy	100 %
	Recall	100 %
	Precision	100 %
	F-1Score	100 %
LOO	Accuracy	100 %
	Recall	100 %
	Precision	100 %
	F-1Score	100 %

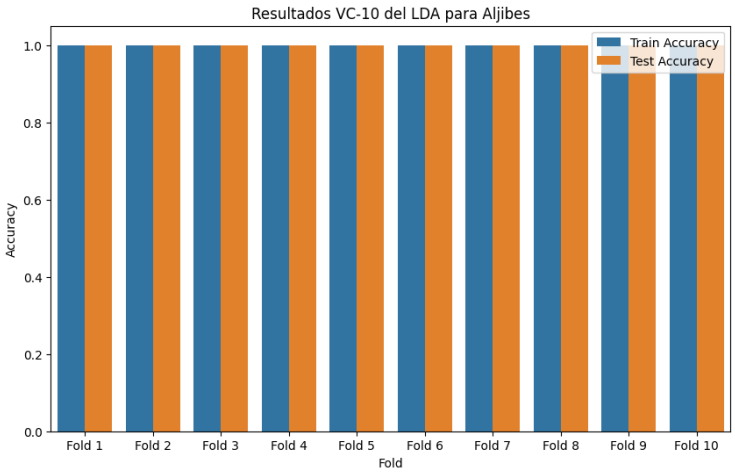


Figura 5.1: Resultados del LDA en entrenamiento y validación sobre Aljibes

5.1.1. Predicciones sobre otras Fases

Para comprobar si las fases entre intensidad y tensión, así como las propias magnitudes intensidad y tensión, son equivalentes, se prueba el desempeño en otras fases un clasificador entrenado con una fase distinta a las anteriores, tanto de intensidad como de tensión, de este mismo pozo de Aljibes. Aunque en principio se ha trasladado, desde el Departamento de Ingeniería Eléctrica de la Universidad de Valladolid, la idea de la equivalencia entre fases y tensión e intensidad, en el capítulo 4 se ha mostrado que esto puede no ser cierto. Hay que recordar que la selección de variables no es igual para cada fase dentro de la intensidad, ni para las mismas fases entre tensión e intensidad. Por ello, se emplean las variables de la selección realizada y, alternativamente, con la totalidad de las variables para entrenar al clasificador. Los resultados, con ambas opciones, dejan en evidencia que no se debe emplear una fase para predecir otra distinta, así como tampoco se debe entrenar un clasificador LDA con intensidad para predecir en tensión y viceversa. Destaca también el hecho de que para las fases de tensión son necesarias más variables que para intensidad para obtener los mismos resultados. Algunas de las matrices de confusión obtenidas se muestran en la figura A.41 de los anexos, como muestra de que la mayoría de las observaciones se clasifican incorrectamente. Esto ocurre también en el resto de combinaciones de conjuntos entre *train* y *test* no presentes en esta figura A.41.

5.1.2. Predicciones sobre otros Pozos

Se ha comprobado que un LDA no permite predecir en otra fase distinta a la de los datos de entrenamiento, ni permite mezclar intensidad y tensión. Por ello, se fija la fase A de intensidad en Aljibes como conjunto de datos para entrenar y tratar de generalizar el clasificador al resto de pozos. Al no disponer de etiquetas en el resto de pozos, el desempeño del clasificador se basa en la idea incial del estado del pozos, por lo que no se sabe con total certeza cuando un clasificador da buenos o malos resultados. Las predicciones correspondientes al estado de los motores del resto de pozos se muestan en la figura A.42 de los anexos. En esta figura se puede observar como el motor de Pedrizas2A se clasifica en mal estado desde el inicio de sus mediciones, coincidiendo con la información recibida acerca de este motor. Para Pedrizas1V no parece que la clasificación sea del todo correcta, puesto que clasifica la primera mitad de las observaciones alternando entre un buen y mal estado, estableciéndose más tarde en un estado óptimo. En Quintana se clasifica toda observación como buen estado, lo que puede ser correcto. Un caso similar ocurre con Pellicer, con la diferencia en que este último parece fallar un par de observaciones predichas como mal estado. Por último, el motor de Guillamón, en principio en buen estado, clasifica como mal estado las primeras y las últimas observaciones, lo que hace pensar que el clasificador no esta actuando bien y está generando falsos negativos, tal vez provocado por la simetría en el desgaste de los anillos terminales. Un resumen de estos resultados se presenta en la tabla 5.1. Se puede concluir que los resultados del LDA no son correctos en los motores de Pedrizas1V, Pellicer y Guillamón, presentando estos dos últimos falsos negativos en las predicciones. Por otro lado, las predicciones del clasificador proveniente del LDA son más alineadas a la realidad en Pedrizas2A y Quintana.

Pozo	Estado Real	Estado Predicho
Aljibes	Fallo al final	Fallo al final
Pedrizas1V	Fallo	Alterna fallo y buen estado
Pedrizas2A	Fallo desde el inicio	Fallo desde el inicio
Pellicer	Bueno	Bueno con dos observaciones malas
Guillamón	Bueno	Fallos al inicio y al final
Quintana	Bueno	Bueno

Tabla 5.1: Resumen de resultados del LDA

5.2. Árboles de Decisión

Los resultados obtenidos con el Análisis Discriminante Lineal parecen ser mejorables. Emplear un árbol de decisión puede aportar otro enfoque que permita comprar los resultados obtenidos con el LDA, a la vez que la facilita la interpretación de los resultados. Como primera aproximación, se ha entrenado el clasificador con el conjunto total de variables. Esto ha provocado múltiples cla-

sificadores distintos que proprocionaban el mismo resultado, una clasificación perfecta en Aljibes, debido a la gran intercorrelación entre las variables del conjunto. Por ello, de aquí en adelante se ha empleado la selección de variables del capítulo 4 sobre la fase A de intendidad. Para la búsqueda de hiperparámetros, sobre el conjunto de Aljibes se ha realizado primeramente un *hold-out* para separar el conjunto de datos en dos (66.66 % − 33.33 %). Sobre el conjunto de mayor tamaño, se ha realizado una validación cruzada de 10 *folds* para encontrar los hiperparámetros óptimos, que luego serán finalmente validados en el conjunto restante. La búsqueda de dichos hiperparámetros ha resultado en un árbol tal que:

- Su profundidad es igual a 2.
- El número mínimo de muestras para dividir un nodo interno es 10.
- El número mínimo de muestras necesarias para crear un nodo hoja igual a 5.
- La función que mide la calidad de una división en un nodo es *gini*, en lugar de *entropy* o *log_loss*.

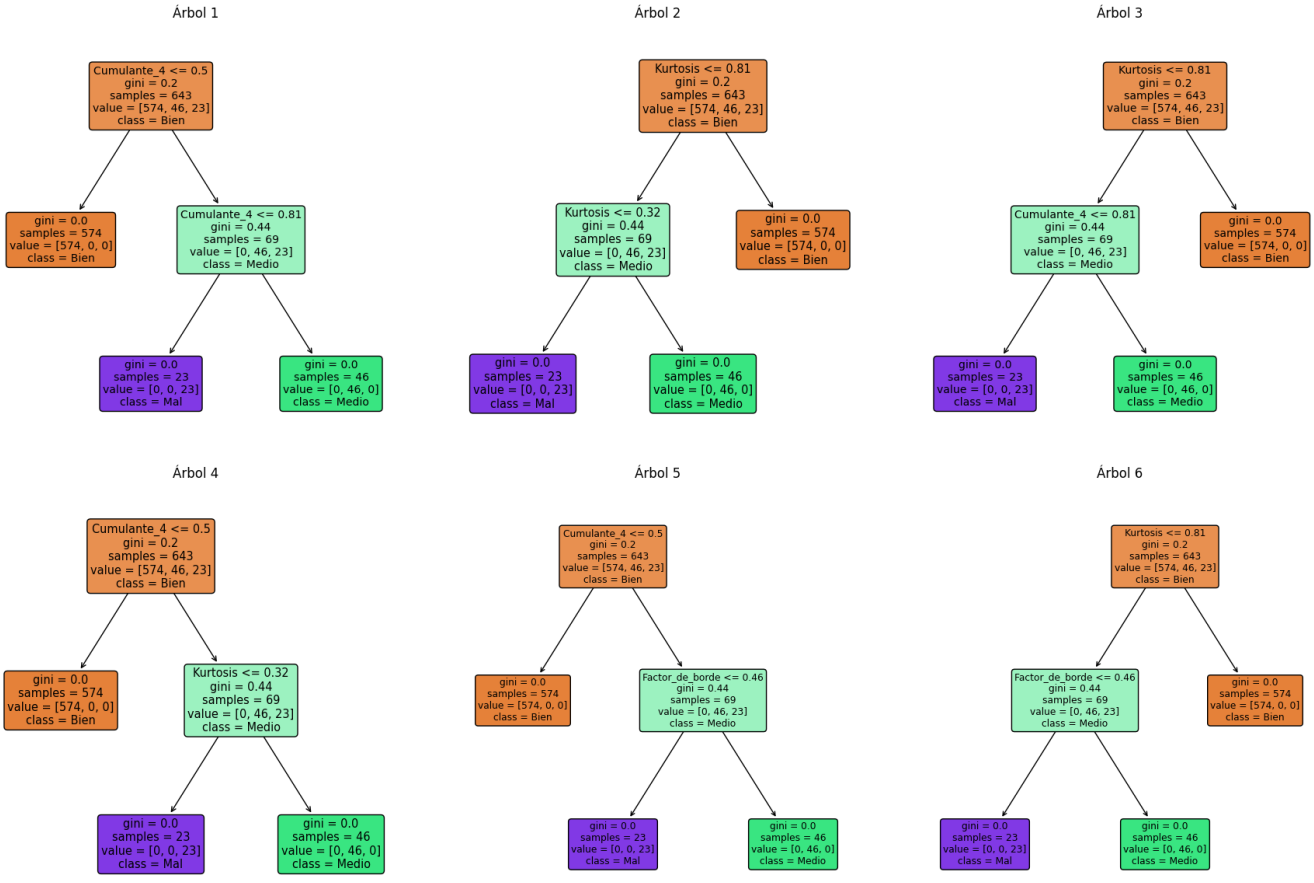


Figura 5.2: Árboles producidos con la configuración de hiperparámetros óptima

Pese a haber reducido el número de variables con la selección, distintas semillas han generado hasta 6 árboles distintos para la misma configuración del algoritmo. Todos los árboles ofrecen

resultados perfectos en la clasificación, al igual que con el LDA, por lo que todos ellos se han probado para realizar predicciones en otras fases de Aljibes y en otros pozos, como se detalla en el apartado 5.2.2. Los árboles obtenidos se muestran en la figura 5.2, donde destaca la presencia de las variables Cumulante_4 y Kurtosis en la mayoría de estos árboles.

5.2.1. Predicciones sobre otras Fases

Análogamente al correspondiente apartado del LDA de la sección 5.1.1, para evaluar la equivalencia entre las fases de intensidad y tensión se prueba el rendimiento de un clasificador entrenado con diferentes fases de intensidad y tensión para el mismo pozo, Aljibes. También, se evalua el rendimiento del clasificador estableciendo una fase fija y entrenándolo en intensidad para predecir en tensión y viceversa. Al igual que con el LDA, para obtener las mismas métricas sobre la clasificación se requieren más variables para la tensión que para la intensidad, por lo que los árboles para la tensión presentan una profundidad máxima igual a 4. Un ejemplo de estos árboles se muestra en la figura A.43. Los resultados y las conclusiones extraídas son idénticas a las ya obtenidas anteriormente, clarificando cada vez más la no equivalencia ni entre magnitudes ni entre fases.

5.2.2. Predicciones sobre otros Pozos

Al igual que en el LDA, se fija la fase A de intensidad de Aljibes como conjunto de datos para entrenar al clasificador. En la figura 5.2 se muestran los seis clasificadores distintos obtenidos, a los cuales se hará referencia como "Árbol X", siendo X un número del 1 al 6, según se indica en la propia figura. Todos estos clasificadores han sido empleados en la predicción del estado de los motores del resto de pozos y los resultados proporcionados son notablemente diferentes:

- Los árboles 1, 4 y 5 etiquetan toda observación de Pedrizas1V como correcta, mientras que el árbol 3 lo clasifica en un estado intermedio. Los árboles 2 y 6 lo categorizan en un mal estado.

- Los árboles 1 y 3 asginan un mal estado a todo el pozo de Pedrizas2A, mientras que el resto de árboles alternan el estado medio con el malo, con algún signo de evolución del deterioro con el tiempo.

- Todos los árboles clasifican todas las observaciones de Pellicer en buen estado, lo cual es consistente con el LDA.

- Prácticamente todos los árboles detectan el mismo patrón detectado por el LDA (figura A.42) en Guillamón: un estado medio/malo en el periodo inicial y final de las mediciones, mientras que en la época central se mantiene un buen estado en el motor.

- Los árboles 1, 4 y 5 asignan un buen estado a la completitud de las observaciones de Quintana. Mientras, los árboles 3 y 6 clasifican la mayoría de ellas como un estado medio y el árbol 2 como un estado malo.

Estos resultados demuestran, principalmente, una clara inestabilidad en los clasificadores, ya que una misma configuración provoca hasta seis árboles diferentes que clasifican contrariamente ciertas observaciones. Pese a que los árboles 1, 4 y 5 parecen los más consistentes con lo obtenido en el LDA, reduciendo el número de variables explicativas a únicamente dos estos clasificadores no parecen del todo adecuados para el problema. La figura A.44 de los anexos muestra los resultados procedentes del árbol 1, ya comentados con anterioridad, mientras que la tabla 5.2 refleja un resumen de los mismos.

Pozo	Estado Real	Estado Predicho
Aljibes	Fallo al final	Fallo al final
Pedrizas1V	Fallo	Bueno
Pedrizas2A	Fallo desde el inicio	Fallo desde el inicio
Pellicer	Bueno	Bueno
Guillamón	Bueno	Bueno con alguna observación media al inicio
Quintana	Bueno	Bueno

Tabla 5.2: Resumen de resultados del árbol de decisión 1

5.3. Random Forest

Un bosque aleatorio puede mejorar el problema de la inestabilidad mostrado por los clasificadores de un único árbol de decisión, ya que se basa de un conjunto de árboles simples e inestables, como los obtenidos anteriormente, para crear un clasificador más robusto y estable. Para este algoritmo se han seleccionado todas las variables disponibles, puesto que Random Forest permite medir la importancia de las variables en la clasificación. Se ha seguido el mismo procedimiento de división de conjuntos de datos para la obtención de los hiperparámetros óptimos que en los árboles de decisión, obteniendo la siguiente configuración de hiperparámetros:

- El bosque consta de 102 árboles de decisión.
- Cada árbol cuenta con una profundidad máxima igual a 5.
- La función que mide la calidad de una división en un nodo, al igual que en los árboles de decisión, es *gini*.

Esta configuración proporciona un valor inmejorable en todas nuestras métricas: *accuracy*, *recall*, *precision* y *f1-score*. En la figura 5.3 se puede apreciar la importancia de las distintas variables en

la clasificación para Aljibes, donde destaca la variable Kurtosis, también presente en la selección propuesta en el capítulo 4 y en la mayor parte de los árboles de decisión del apartado anterior.

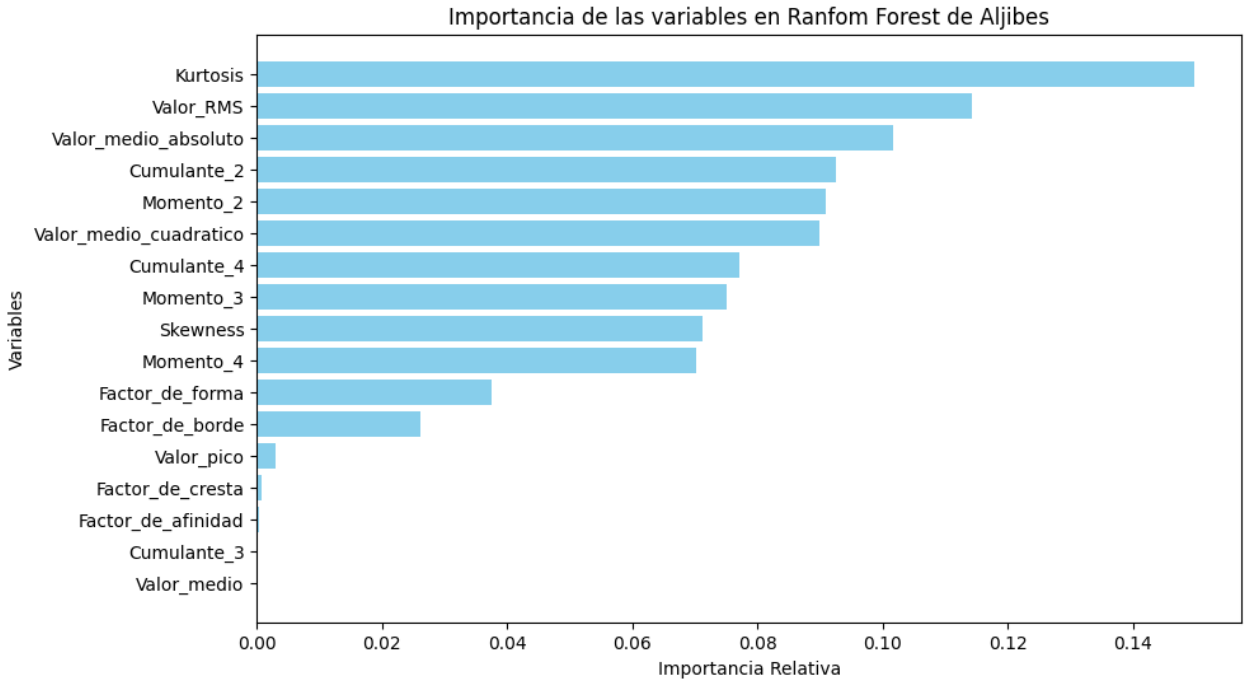


Figura 5.3: Importancia de las variables del Ranfom Forest en Aljibes para intensidad fase A

5.3.1. Predicciones sobre otras Fases

Como se ha visto con otros clasificadores, no exite ninguna equivalencia entre tensión e intensidad, así como en sus fases asociadas. Este hecho se puede volver a comprobar con un bosque aleatorio, como ya se hizo con el LDA y con los árboles de decisión. La importancia de las variables, con su correspondiente selección óptima de hiperparámetros, también es cambiante dependiendo de la fase y magnitud con la que se entrene el clasificador. En la sección A.3.3 de los anexos se muestran los gráficos correspondientes a las importancias relativas de los distintos conjuntos de entrenamiento en Aljibes. Se aprecian grandes diferencias fijando las fases y variando la magnitud entre tensión e intensidad. En la tensión se concentran altos valores de importancia en pocas variables, destacando Valor_medio_absoluto, mientras que en intensidad la relevancia a la hora de clasificar se reparte más entre varias variables. También se presentan variaciones entre las distintas fases para una misma magnitud, ya que para cada fase se asigna una importancia distinta a cada variable. Por otro lado, destacar que como ya se había mostrado anteriormente, la tensión requiere de una mayor complejidad a la hora de obtener buenas clasificaciones, aumentando el tiempo de entrenamiento levemente y fijando una nueva configuración de hiperparámetros tal que:

- El bosque aleatorio para la tensión en fase A cuenta con 82 árboles de decisión.
- La profundidad máxima de cada árbol se establece en 10.

- De nuevo, *gini* es la función elegida a la hora de evaluar la calidad en la división de un nodo.

Estos hiperparámetros provocan que el clasificador del bosque aleatorio, en la fase A de tensión, clasifique hasta un 2.48 % de observaciones incorrectamente en Aljibes y produzca un *recall* del 80.25 %, por lo que un número considerable de observaciones en mal estado no están siendo detectadas. Esto deja en evidencia como un predictor más simple para la intensidad obtiene mejores resultados que uno más complejo para la tensión.

5.3.2. Predicciones sobre otros Pozos

La figura A.48 de los anexos muestra las predicciones obtenidas sobre los distintos pozos con el clasificador entrenado, usando los hiperparámetros comentados anteriormente. La tabla 5.3 refleja un resumen de estos mismos resultados. Debe recordarse que este clasificador se ha entrenado con los datos de la fase A de intensidad en Aljibes, por lo que las correspondientes predicciones son también en las fases A de intensidad de los distintos pozos. Se puede apreciar como el modelo de clasificación no diferencia entre las observaciones de un mismo pozo y las clasifica todas en la misma categoría. Destaca la clasificación sobre Pedrizas1V, puesto que se conoce que el estado real de dicho motor no es bueno. El resto de predicciones parecen consistentes con la información acerca del estado de los motores y con los resultados obtenidos con otros clasificadores previos. Puede resultar sorprendente la ausencia de evolución del deterioro en Pedrizas2A; sin embargo, es importante recordar que se cree que este motor presentaba problemas y fallos desde el inicio de las mediciones. En Guillamón podemos ver que no se detectan fallas en las observaciones iniciales ni en las finales, a diferencia de los resultados obtenidos tanto con el LDA como con los árboles de decisión.

Pozo	Estado Real	Estado Predicho
Aljibes	Fallo al final	Fallo al final
Pedrizas1V	Fallo	Bueno
Pedrizas2A	Fallo desde el inicio	Fallo desde el inicio
Pellicer	Bueno	Bueno
Guillamón	Bueno	Bueno
Quintana	Bueno	Bueno

Tabla 5.3: Resumen de resultados del Random Forest

5.4. XGBoost

Alternativamente, se han desarrollado técnicas de *boosting*, empleando clasificadores XGBoost para resolver el problema. Al igual que en el bosque aleatorio, se han empleado todas las variables

disponibles en la fase A de intensidad para poder obtener resultados más precisos, debido a que el criterio de selección de variables propuesto provocaba distintas selecciones en los diferentes conjuntos de datos. Además, al igual que los bosques aleatorios, XGBoost permite medir la importancia de cada variable en la clasificación. A la hora de obtener los hiperparámetros que proporcionan mejores clasificaciones, se ha realizado una búsqueda aleatoria dentro del espacio de parámetros, realizando una validación cruzada de 5 *folds* con 5 repeticiones, lo que permite una evaluación más robusta del rendimiento del modelo al promediar los resultados de las cinco iteraciones, reduciendo la variabilidad del modelo. Los principales hiperparámetros de la configuración final seleccionada se muestran en la tabla 5.4 y producen una tasa de error media del 0.12 % en la clasificación sobre Aljibes, lo cual es levemente peor que los resultados ofrecidos por los clasificadores anteriores. La figura 5.4 indica las importancias de las distintas variables en esta clasificación. La variable Kurtosis sigue siendo importante en la clasificación, al igual que en el bosque aleatorio, pero ahora tanto Valor_medio_absoluto como Valor_RMS tienen más relevancia a la hora de clasificar una observación. Estas tres variables acumulan hasta un 97.67 % del total de importancia en la clasificación.

Hiperparámetro	Valor óptimo
learning_rate	0.21
gamma	0.1
n_estimators	200
max_depth	2

Tabla 5.4: Resumen de los hiperparámetros óptimos para XGBoost para intensidad fase A

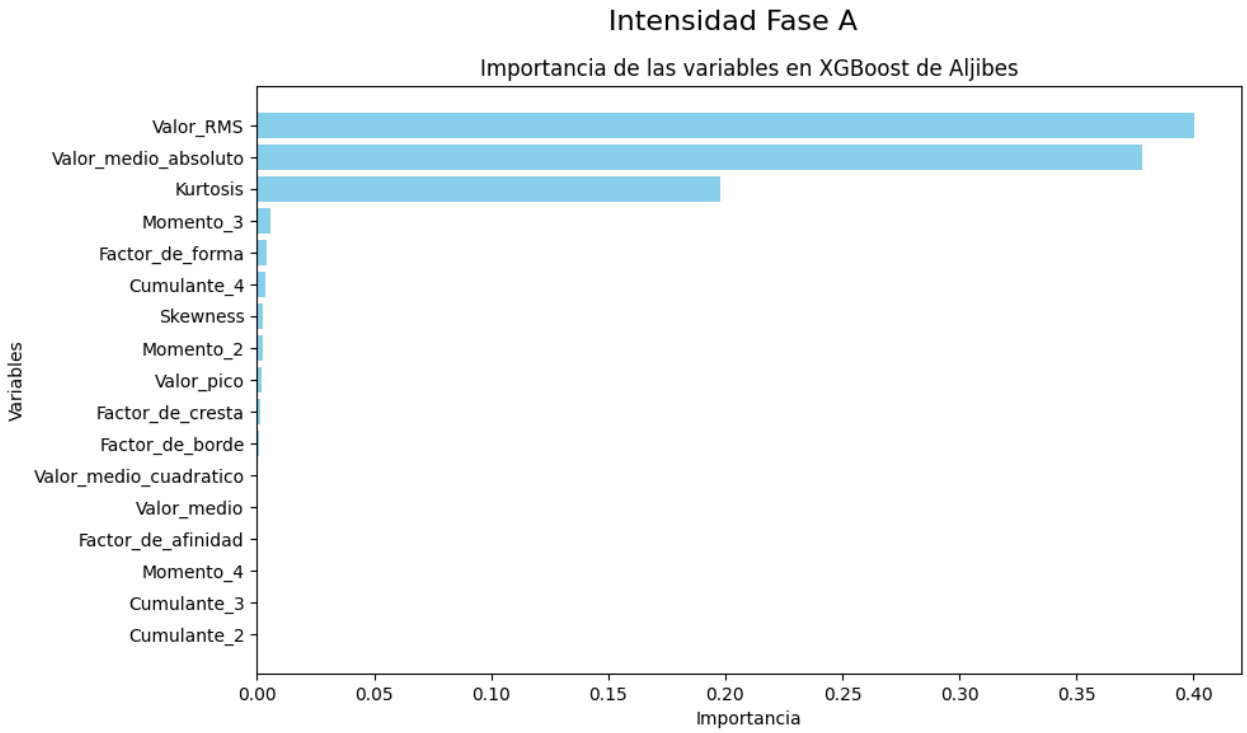


Figura 5.4: Importancia de las variables del XGBoost en Aljibes en intensidad fase A

5.4.1. Predicciones sobre otras Fases

De nuevo, se ha evaluado la equivalencia entre las magnitudes y las diferentes fases para este nuevo clasificador XGBoost. Los resultados, iguales a los ya comentados en anteriores apartados, hacen indicar la ausencia de dichas equivalencias. Al igual que ocurre en el bosque aleatorio, las importancias de las variables en la clasificación varían dependiendo del conjunto de entrenamiento seleccionado en Aljibes. La sección A.3.4 ilustra las diferentes importancias en este pozo Aljibes para cada magnitud y fase seleccionada. Ahora, es la intensidad la que acumula la mayor parte de importancia en dos o tres variables, mientras que la tensión reparte más equitativamente el peso de una variable a la hora de clasificar una observación. A pesar de esto, en la tensión destaca la importancia de Valor_medio_absoluto como variable más relevante. En cuanto a la complejidad del modelo, la tensión vuelve a requerir modelos más complejos que la intensidad, obteniendo para la fase A de tensión un *accuracy* entorno al 97.70 % en Aljibes, con la configuración de hiperparámetros de la tabla 5.5:

Hiperparámetro	Valor óptimo
learning_rate	0.11
gamma	0.1
n_estimators	1000
max_depth	7

Tabla 5.5: Resumen de los hiperparámetros óptimos para XGBoost en tensión fase A

5.4.2. Predicciones sobre otros Pozos

Este modelo entrenado con la fase A de intensidad de Aljibes se ha empleado, por última vez, para inferir predicciones sobre el resto de motores. Los resultados de estas predicciones pueden verse en la figura A.52 de los anexos. Estos resultados son equivalentes a los obtenidos con el bosque aleatorio, por lo que se pueden extraer las mismas conclusiones acerca de los datos sin la necesidad de emplear un modelo complejo como XGBoost.

Pozo	Estado Real	Estado Predicho
Aljibes	Fallo al final	Fallo al final
Pedrizas1V	Fallo	Bueno
Pedrizas2A	Fallo desde el inicio	Fallo desde el inicio
Pellicer	Bueno	Bueno
Guillamón	Bueno	Bueno
Quintana	Bueno	Bueno

Tabla 5.6: Resumen de resultados de XGBoost

Capítulo 6

Conclusiones y Trabajo Futuro

En este último capítulo se clarifican las conclusiones extraídas a lo largo de todo el proyecto. También se exponen las dificultades principales encontradas a lo largo de él y las posibles líneas de trabajo futuro e investigación.

6.1. Conclusiones

A partir del preprocesamiento y de los análisis desarrollados, se puede concluir:

- **Objetivos 1 y 2**. Los indicios mostrados en el capítulo 4 sobre la falta de equivalencia entre las diversas fases de intensidad y tensión, así como entre la intensidad y tensión dentro de una misma fase, se corroboran en el capítulo 5. En este capítulo, se observa cómo un modelo entrenado en una fase determinada no da buenos resultados cuando se utiliza para predecir en otras fases de intensidad o tensión. De hecho, se ha probado como para la tensión se requieren modelos más complejos que para la intensidad. Este hecho de no equivalencia discrepa con la idea inicial proporcionada por los expertos del departamento de Ingeniería Eléctrica de la Universidad de Valladolid y, debido a esto, durante el capítulo 5 los resultados y predicciones mostradas corresponden a la fase A de la intensidad. Sin embargo, se han repetido los mismos análisis y técnicas predictivas para la fase C de intensidad y las fases A y C de tensión, obteniendo resultados realmente similares o peores, y con ello no concluyentes, a los de la fase A de intensidad en las predicciones.

- **Objetivo 3**. Parece claro que el tipo de fallo presentado por el motor de Aljibes es significativamente diferente al presentado por el motor de Pedrizas1V. Como se muestra durante el capítulo 5, los distintos clasificadores desarrollados han sido incapaces de detectar fallas en el motor de Pedrizas1V, pese a saberse con certeza que este motor realmente sufrió daños. Tanto XGBoost, como el bosque aleatorio y muchos de los árboles de decisión individuales no predicen ninguna observación perteneciente a un estado que no sea bueno, mientras que

el LDA si detectó ciertas observaciones en medio/mal estado en la primera mitad de las observaciones. Esta primera mitad de las observaciones coincide con la primera época de las mediciones realizadas, como se ve en la figura A.3, por lo que tal vez en esa época de ausencia en las mediciones el motor se reparó adecuadamente. Pese a que las fallas reportadas en Aljibes son diferentes a las del motor de Pedrizas2A, ambos errores dan indicios de ser detectados por un mismo clasificador.

Esto permite concluir que no puede utilizarse un único clasificador para detectar todos los posibles fallos en las bombas. En otras palabras, diferentes tipos de fallos parecen manifestarse en distintas variables explicativas.

- **Objetivo 4**. El bosque aleatorio ha proporcionado las mismas predicciones sobre los motores de los pozos que XGBoost, siendo *random forest* una técnica más simple en términos de implementación, paralelización, ajuste de hiperparámetros y tiempos de entrenamiento. Estos resultados, a excepción del ya comentado motor de Pedrizas1V, son consistentes con la información proporcionada por los profesionales del departamento de Ingeniería Eléctrica de la Universidad de Valladolid. Gracias a estos resultados se confirma la no existencia de fallas en los motores de los pozos de Guillamón, Pellicer y Quintana, mientras que se ratifican los problemas en el motor de Pedrizas2A desde el inicio de sus mediciones. Los árboles de decisión individualmente provocaron resultados más inestables e inciertos, mientras que el LDA provocó, aparentemente, falsos negativos en Guillamón y clasificó incorrectamente dos observaciones en Pellicer.

- Mientras que la selección de variables del capítulo 4 ordenaba la selección de Valor_medio, Cumulante_4, Kurtosis, Factor_borde, Factor_afinidad como valores a emplear en los modelos, los árboles de decisión individuales se basaban principalmente en los valores de las variables Cumulante_4 y Kurtosis, junto a Factor_borde. Por otro lado, los resultados más fiables, proporcionados por el bosque aleatorio y XGBoost, también emplean Kurtosis como variable clave a la hora de clasificar pero Cumulante_4 pierde peso en la clasificación, sobre todo en XGBoost. Aquí ganan importancia otras variables como Valor_medio_absoluto y Valor_RMS, entre otras.

6.2. Trabajo Futuro

Este proyecto, a su vez, se ha encontrado con diferentes trabas por las que se han tenido que tomar decisiones a la hora de encaminarlo. Esto hace que queden numerosos frentes abiertos para dar continuidad al trabajo aquí presentado.

Las principales dificultades encontradas durante el proyecto tienen relación directa con la naturaleza de los datos. Como se ha comentado durante las conclusiones, cada motor que presentaba fallas lo hacía con un problema distinto, por lo que esto ha podido perjudicar a la predicción al emplear un clasificador entrenado con una falla distinta. Además, la ausencia de indicadores en el

resto de motores no permite conocer realmente si un clasificador estaba prediciendo bien fuera del pozo de Aljibes, ya que la información que se tiene acerca del estado real de un pozo no es completamente segura. Esto significa que realmente no se conoce cuándo un motor verdaderamente comenzó a dar problemas ni si un motor de los que supuestamente están en buen estado (Pellicer, Guillamón y Quintana) lo está en realidad, aunque los resultados mostrados durante el capítulo 5 parecen indicarlo. Por otro lado, como se ha visto en el capítulo 2, el monitoreo constante y continuo de estos motores es clave a la hora de detectar fallas y problemas en su funcionamiento. Sin embargo, los gráficos mostrados en la sección A.2.1 denotan que no hay una regularidad continuada en las mediciones, teniendo periodos de meses e incluso un año de ausencia de mediciones en algunos pozos. Otra dificultad surge del hecho de que los datos son reales, lo que se traduce en una gran carga en el preprocesamiento de estos datos. Esto ha implicado la toma de decisiones en aspectos como seleccionar uno de los tres indicadores, la forma en la que discretizar el indicador puesto que este era un valor continuo o la elección de fases y de magnitud eléctrica a la hora de entrenar y predecir con los clasificadores. Aunque se han replicado los análisis propuestos y desarrollados en este TFG para otras fases y magnitudes o empleando otros indicadores, estas opciones no se han evaluado tan a fondo como la fase A de intensidad con el indicador dos. También, en el capítulo 4, se ha podido apreciar cómo las distintas variables se correlacionan diferente manera según el motor, independientemente del estado y de la fase o magnitud seleccionada. Esta dificultad añadida ha podido provocar que los resultados del LDA y de los árboles de decisión, con la selección de variables realizada, no sean tan precisos como el bosque aleatorio y XGBoost, que empleaban todas las vairables en su conjunto.

Trabajo Futuro

Una vez expuestas las conclusiones y dificultades encontradas, se abren varias líneas de investigación y trabajo futuro:

- Ampliar el estudio acerca de los distintos tipos de fallo presentes en los motores del proyecto, con el objetivo de obtener un indicador que permita a un clasificador detectar el tipo de fallo de Pedrizas1V.
- Agregar al problema mediciones en la temperatura y/o en la vibración, lo cual sería de gran utilidad en este tipo de problemas como se comenta en el capítulo 2.
- Aumentar el conjunto de datos analizado con más observaciones para fortalecer las conclusiones obtenidas, así como obtener el mismo indicador en todos los conjuntos de datos para poder probar con certeza el verdadero desempeño de un clasificador.
- Optar por una búsqueda exhaustiva de hiperparámetros en lugar de aleatoria para los modelos desarrollados, especialmente en el caso de XGBoost, con el objetivo de obtener mejores resultados.

- Estudiar tanto la combinación de distintas fases como la combinación de variables en tensión e intensidad con la intención de detectar nuevos patrones y características no visibles durante este proyecto. Ampliar también el estudio de otras fases o magnitudes individualmente, análogamente a lo realizado en este TFG. Esto podría dar pie al desarrollo de modelos más complejos como redes neuronales y técnicas de interpretabilidad de modelos complejos como LIME y SHAP, también aplicables a XGBoost.

Bibliografía

[1] *Induction Motor: Working Principle, Types and Definition.* Electrical4U, 2024, https://www.electrical4u.com/induction-motor-types-of-induction-motor.

[2] *Motor y Bomba Sumergible: La Combinación Perfecta para la Extracción de Agua Subterránea.* Marteli, https://franklinelectric.mx/motor-y-bomba-sumergible.

[3] J. Bonet-Jara, O. Duque-Perez, L. Serrano-Iribarnegaray, and J. Pons-Llinares, *End-Ring Wear in Deep-Well Submersible Motor Pumps.* IEEE, 2022, https://ieeexplore.ieee.org/abstract/document/9756319.

[4] *Squirrel Cage Induction Motor: Construction, Uses and Specifications.* MangoEngineer, 2024, https://www.mangoengineer.in/2024/02/squirrel-cage-induction-motor.html.

[5] J. Amat Rodrigo, *Análisis de Componentes Principales (Principal Component Analysis, PCA) y t-SNE.* Cienciadedatos, 2017. [Online]. Available: https://cienciadedatos.net/documentos/35_principal_component_analysis

[6] Z. Jaadi, *A Step-by-Step Explanation of Principal Component Analysis (PCA).* BuiltIn, 2024, https://builtin.com/data-science/step-step-explanation-principal-component-analysis.

[7] *Análisis de Componentes Principales - Caso práctico resuelto Introducción.* Universidad de Murcia. [Online]. Available: https://www.um.es/estadempresa/multivarianteR/ACP/caso_ACP.html

[8] *Types of machine learning.* MicrosoftLearn, 2024, https://learn.microsoft.com/en-us/training/modules/fundamentals-machine-learning/3-types-of-machine-learning.

[9] S. Kaushik, *Clustering: Different Methods, and Applications.* Analytics Vidhya, 2024, https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering.

[10] D. Calvo, *Clúster Jerárquicos y No Jerárquicos*, 2018, https://www.diegocalvo.es/cluster-jerarquicos-y-no-jerarquicos.

[11] S. Ghodratnama, I. Bismi, and S. Paunikar, *What is the Linear Discriminant Analysis (LDA) algorithm?* LinkedIn, 2024, https://es.linkedin.com/advice/3/what-linear-discriminant-analysis-lda-algorithm-2rp9e?lang=en.

[12] *Binary classification.* MicrosoftLearn, 2024, https://learn.microsoft.com/en-us/training/modules/fundamentals-machine-learning/5-binary-classification.

[13] J. C. López Enriquez, *Definiciones y especificaciones de modelos de Clasificación.* HackMD, 2021, https://hackmd.io/@api-conabio-ml/SJz3eYNrL.

[14] *What is a decision tree?* IBM, 2024, https://www.ibm.com/topics/decision-trees.

[15] *What is random forest?* IBM, 2024, https://www.ibm.com/topics/random-forest.

[16] A. F. Alaminos Fernández, *Árboles de decisión en R con Random Forest.* Obets Ciencia Abierta. Alicante: Limencop, 2023. [Online]. Available: https://rua.ua.es/dspace/bitstream/10045/133067/1/Random_Forest_en_la_Investigacion_Social.pdf

[17] M. Astorgano Antón, *Diagnóstico de fallos de rodamientos en motores de inducción en estado estacionario mediante técnicas boosting y redes neuronales.* Trabajo Fin de Grado, Universidad de Valladolid, 2022, https://uvadoc.uva.es/handle/10324/57935.

[18] *AdaBoost, Gradient Boosting, XG Boost: Similarities and Differences.* Medium, 2023, https://medium.com/@thedatabeast/adaboost-gradient-boosting-xg-boost-similarities-differences-516874d644c6.

[19] *What is boosting?* IBM, 2024, https://www.ibm.com/topics/boosting.

[20] Nohman, *CatBoost v. XGBoost v. LightGBM.* Kaggle, 2020, https://www.kaggle.com/code/nholloway/catboost-v-xgboost-v-lightgbm.

[21] *La empresa: FACSA.* FACSA, 2017, https://www.facsa.com/la-empresa.

Todos los enlaces fueron comprobados correctamente el 19 de junio de 2024.

Anexos

Apéndice A

Imágenes y Gráficos Complementarios

En este primer apéndice se expone material complementario para la comprensión de resultados y toma de decisiones durante el proyecto, así como imágenes reales del desgaste de los motores.

A.1. Imágenes de los Motores

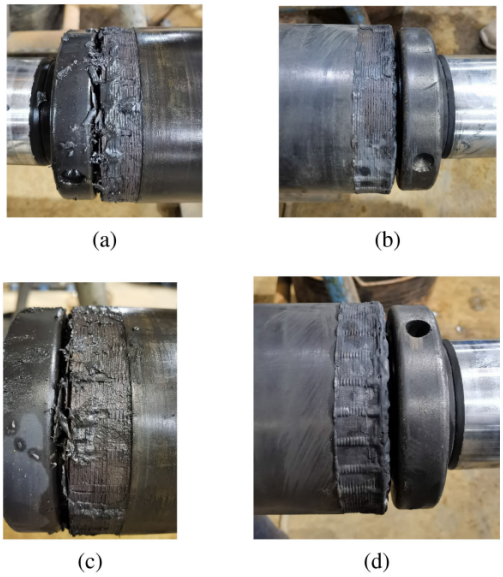


Figura A.1: Anillos desgastados del motor en Aljibes (superiores en *a* y *c*; inferiores en *b* y *d*) [3]

Figura A.2: Acoplamiento desgastado entre el motor y la bomba de Pedrizas2A [3]

A.2. Datos y Preprocesamiento

Durante esta sección se presentarán gráficos complementarios al preprocesamiento realizado, así como gráficos descriptivos de los datos que permitan completar la información acerca de ellos.

A.2.1. Distribución temporal de las observaciones

Gráfico de barras que indica el número de mediciones realizadas por día durante el periodo de medición de cada pozo.

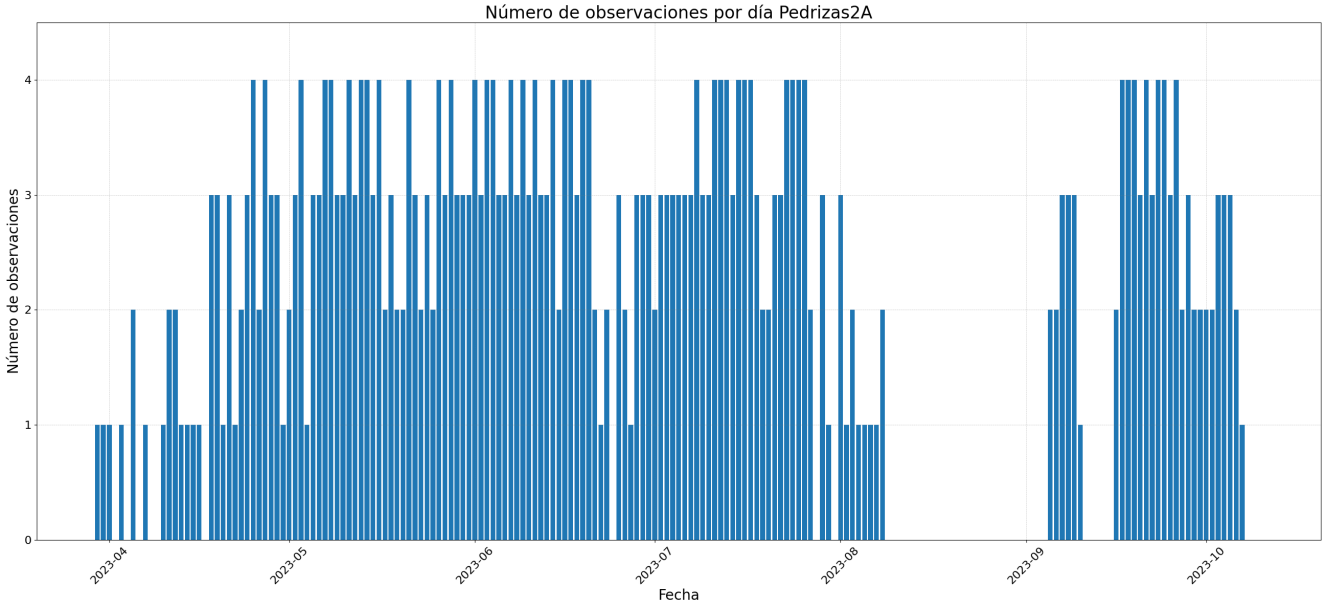


Figura A.3: Número de mediciones por día para Pedrizas2A

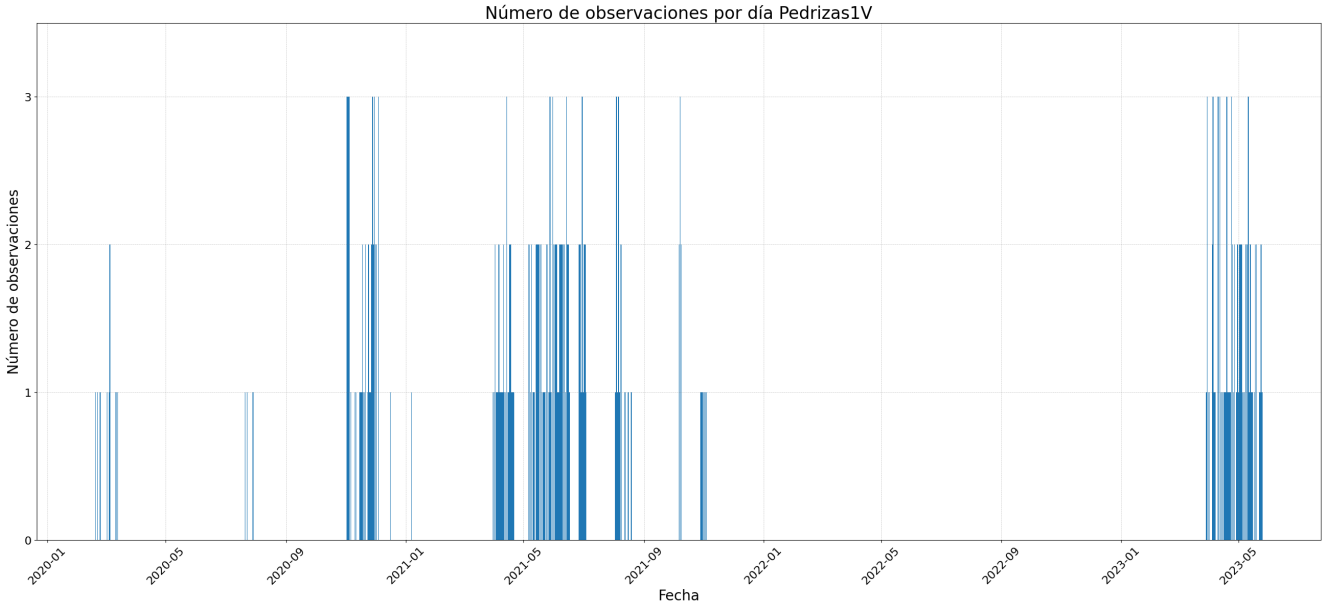


Figura A.4: Número de mediciones por día para Pedrizas1V

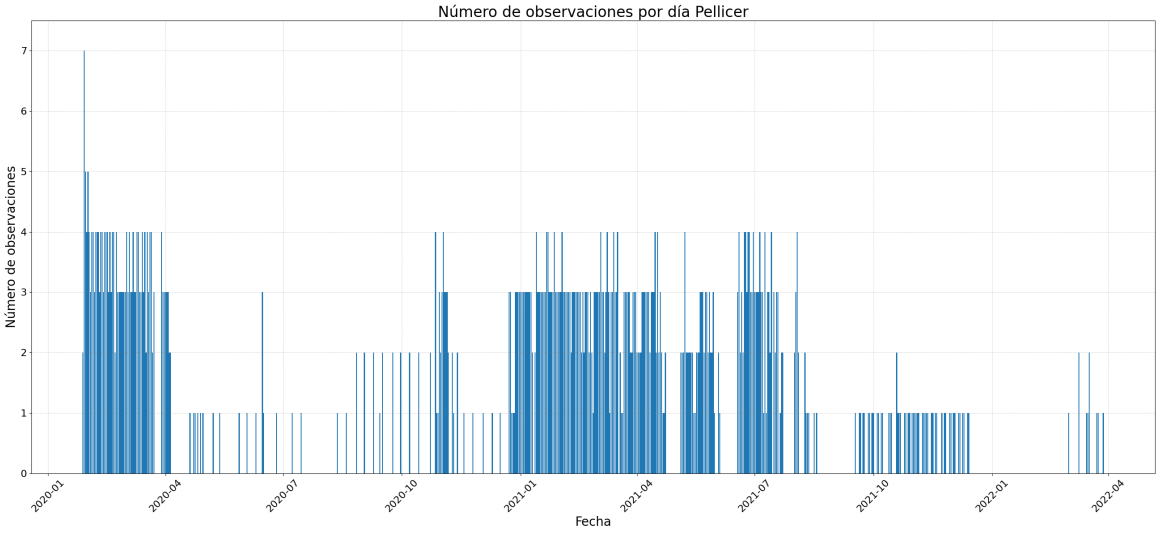


Figura A.5: Número de mediciones por día para Pellicer

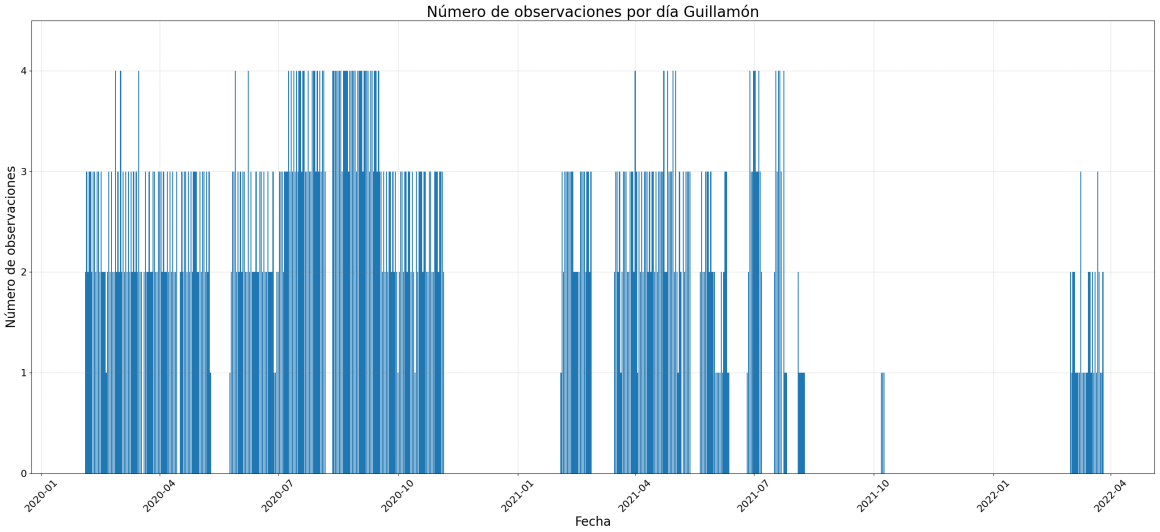


Figura A.6: Número de mediciones por día para Guillamón

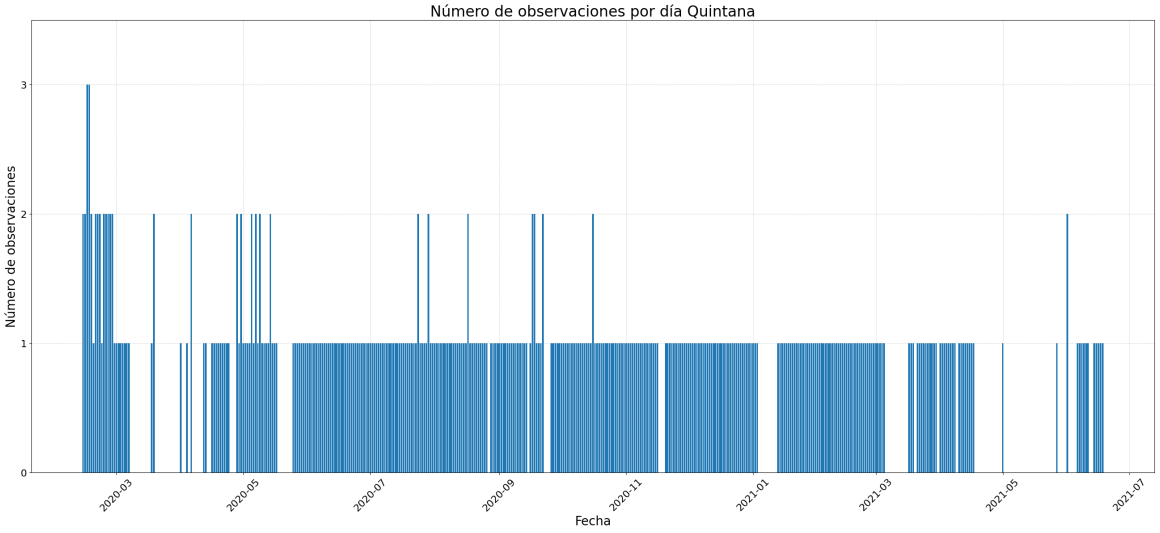


Figura A.7: Número de mediciones por día para Quintana

A.2.2. Diagramas de caja mútilples por variable

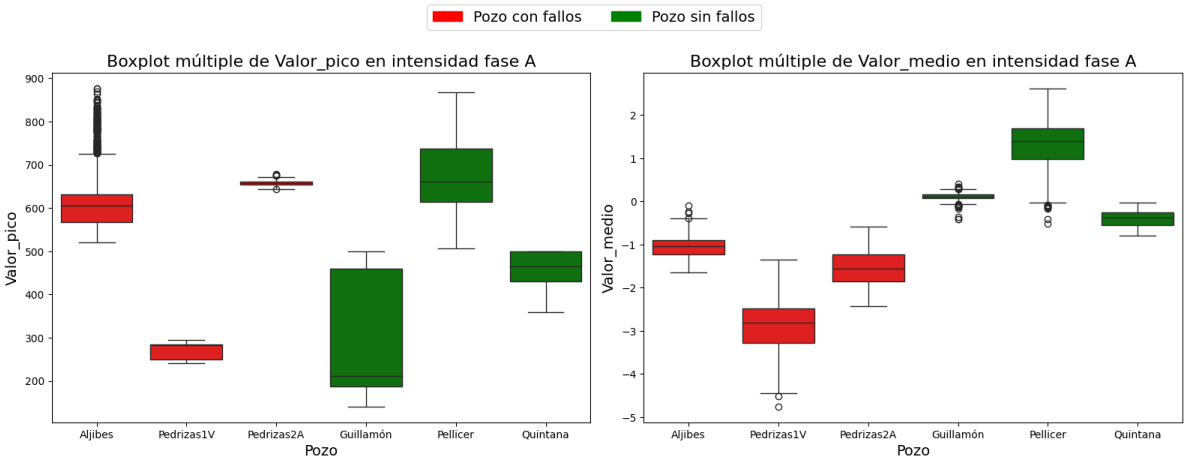


Figura A.8: Diagramas de caja mútilples para Valor_pico y Valor_medio

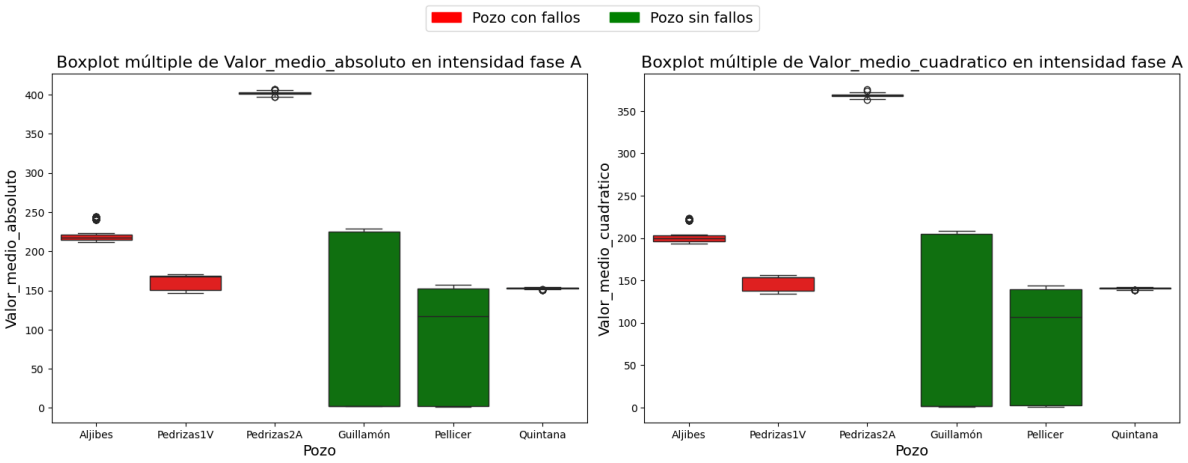


Figura A.9: Diagramas de caja mútilples para Valor_medio_absoluto y Valor_medio_cuadratico

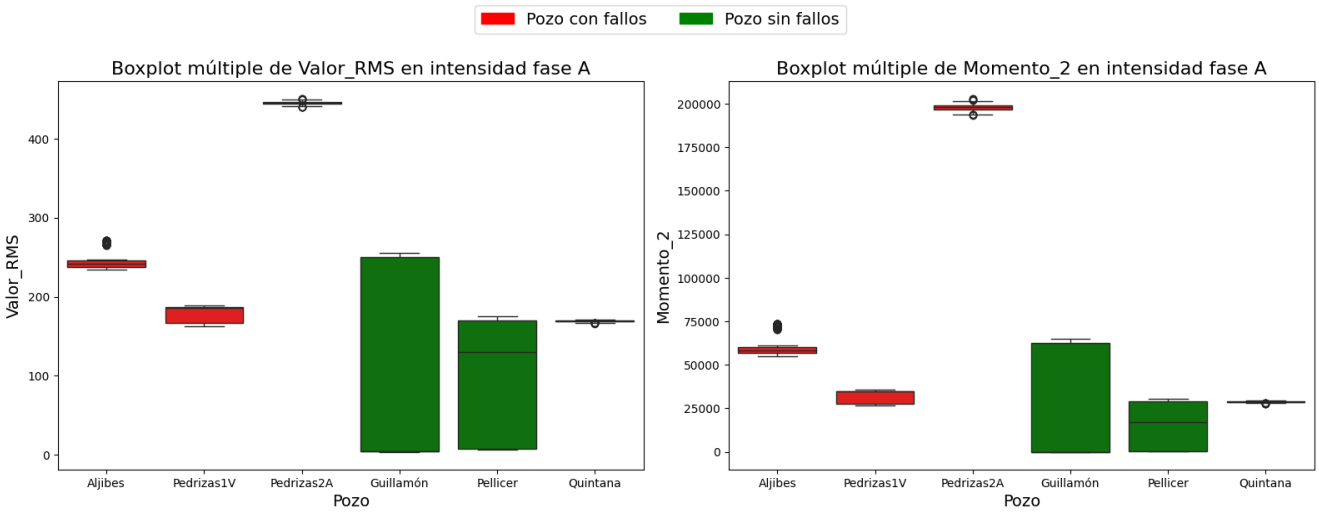


Figura A.10: Diagramas de caja mútilples para Valor_RMS y Momento_2

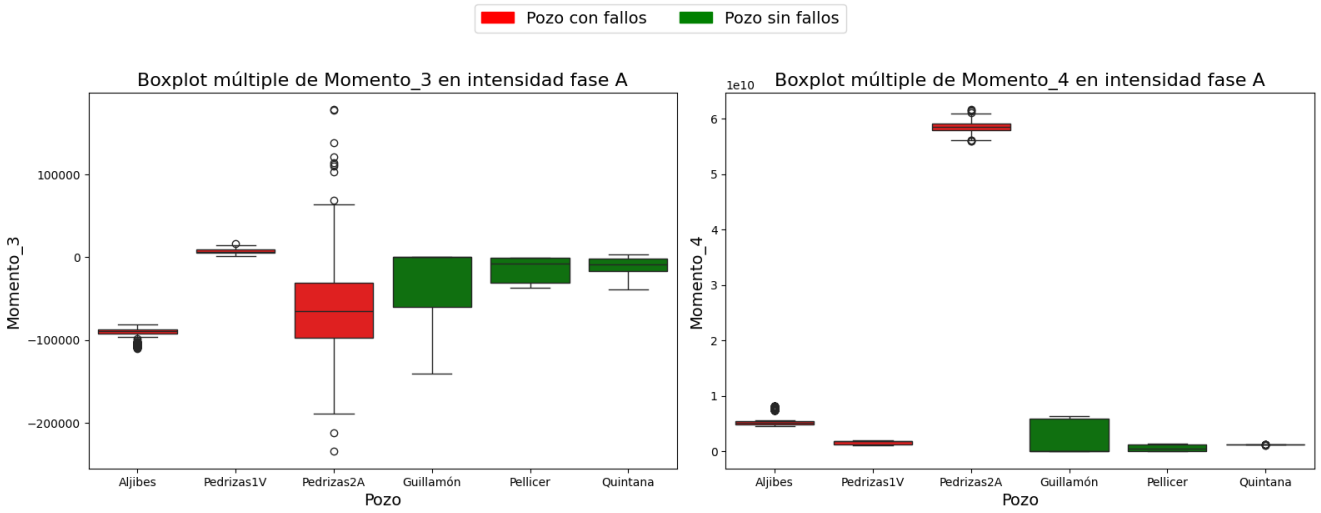


Figura A.11: Diagramas de caja mútilples para Momento_3 y Momento_4

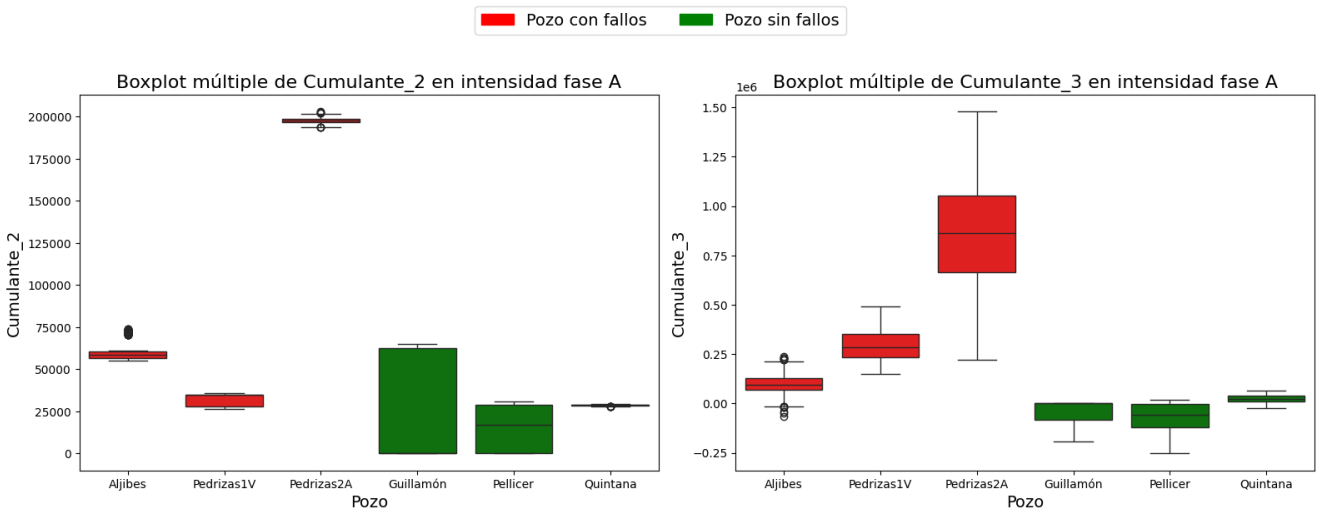


Figura A.12: Diagramas de caja mútilples para Cumulante_2 y Cumulante_3

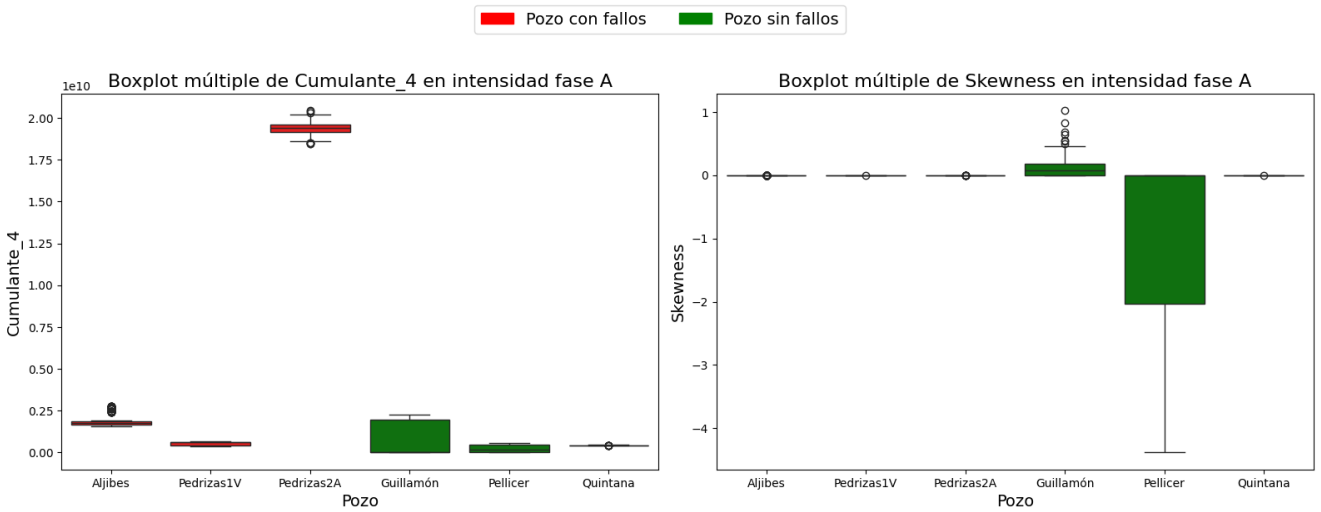


Figura A.13: Diagramas de caja mútilples para Cumulante_4 y Skweness

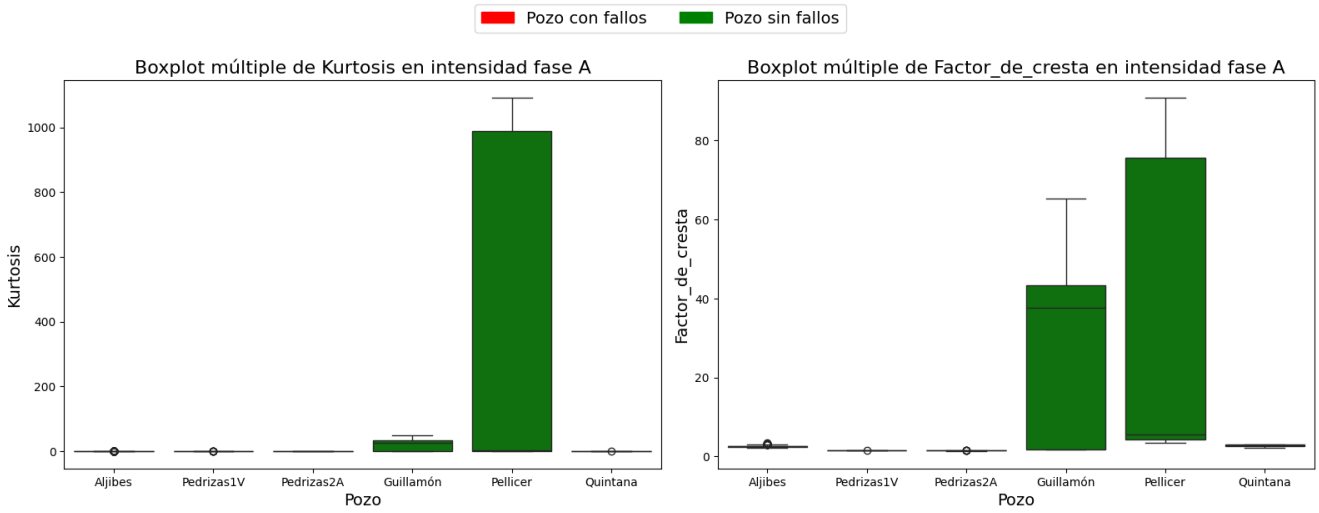


Figura A.14: Diagramas de caja mútilples para Kurtosis y Factor_cresta

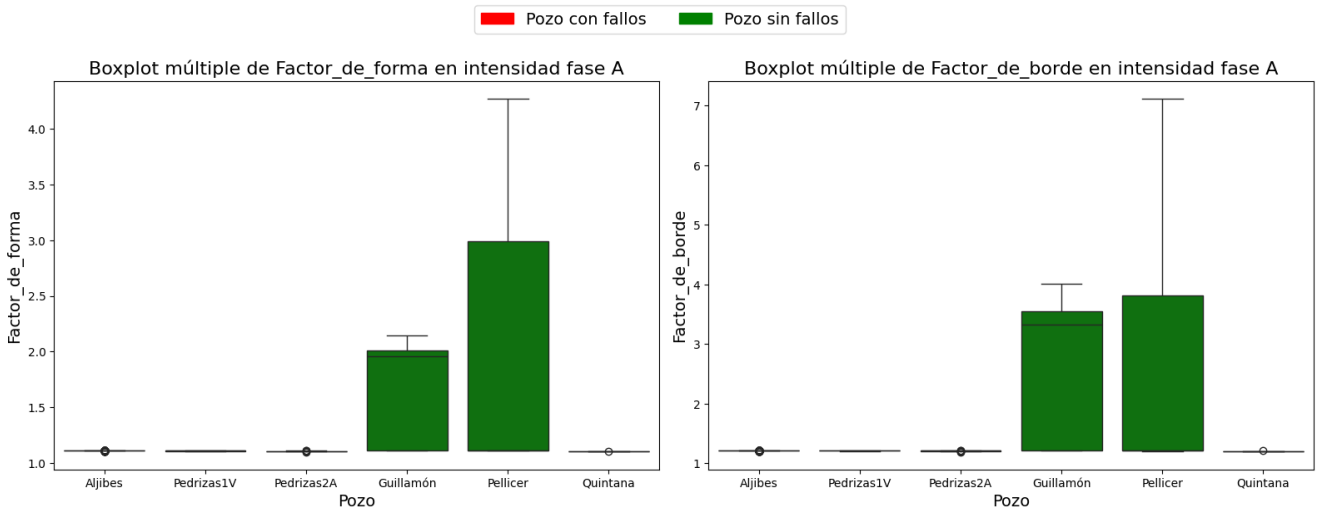


Figura A.15: Diagramas de caja mútilples para Factor_forma y Factor_borde

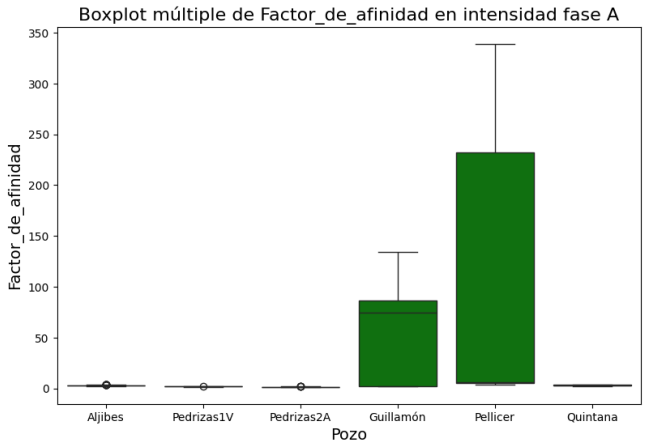


Figura A.16: Diagramas de caja mútilples para Factor_afinidad

A.2.3. Distribución en intensidad y tensión en Aljibes

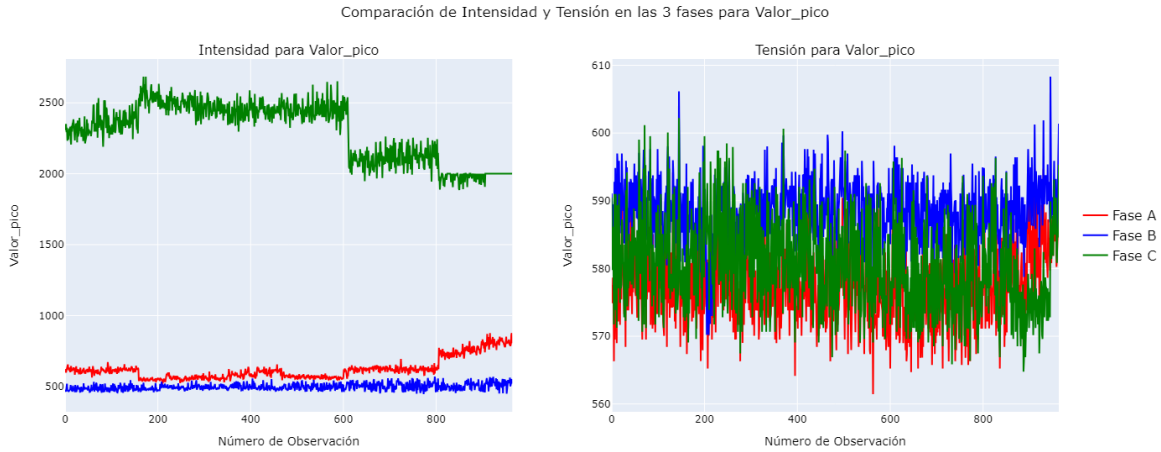


Figura A.17: Distribución de Valor_pico

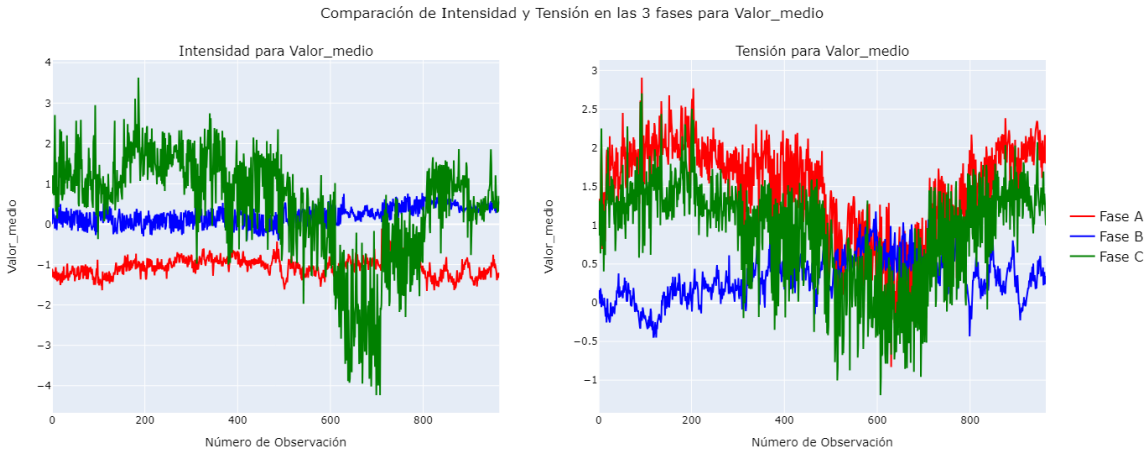


Figura A.18: Distribución de Valor_medio

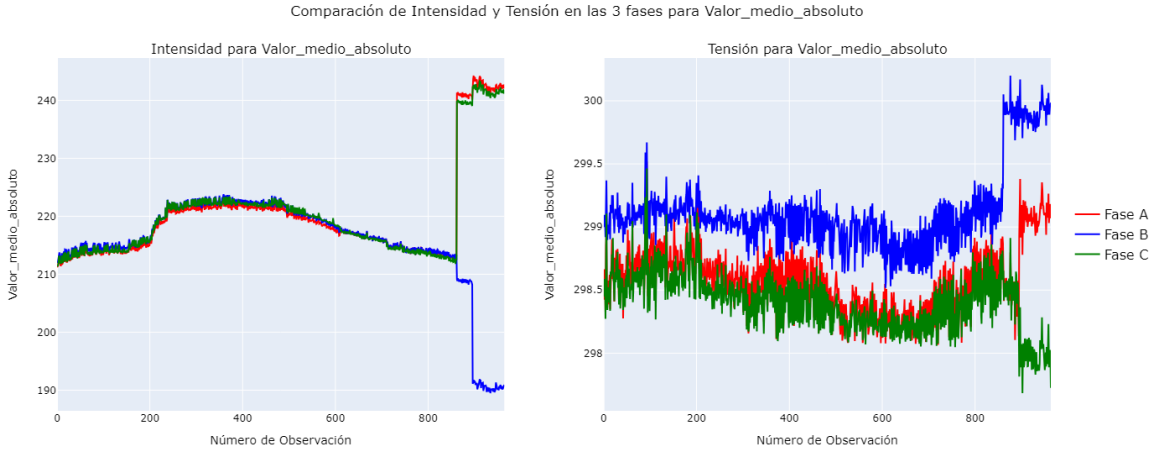


Figura A.19: Distribución de Valor_medio_absoluto

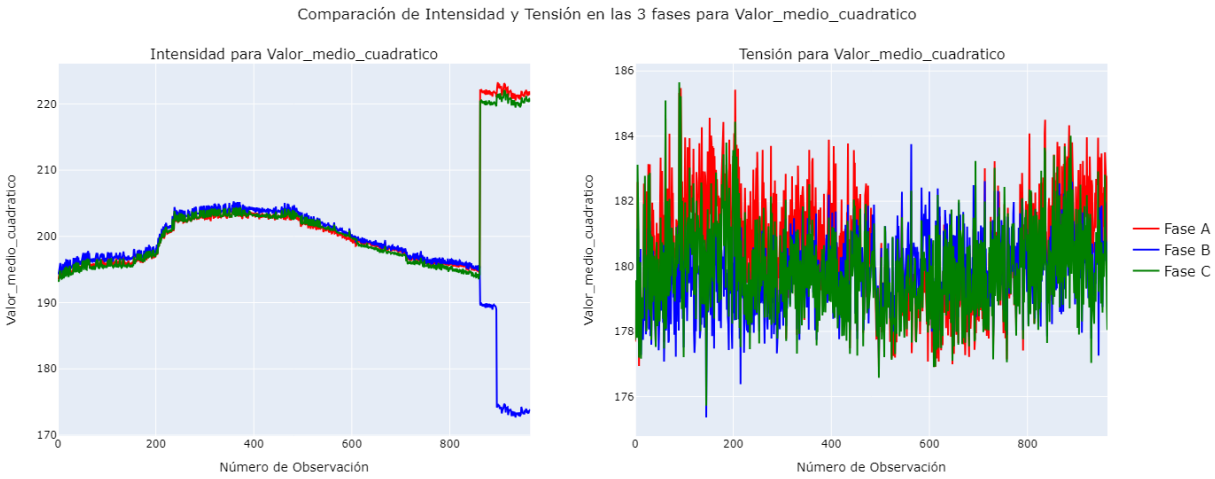


Figura A.20: Distribución de Valor_medio_cuadrático

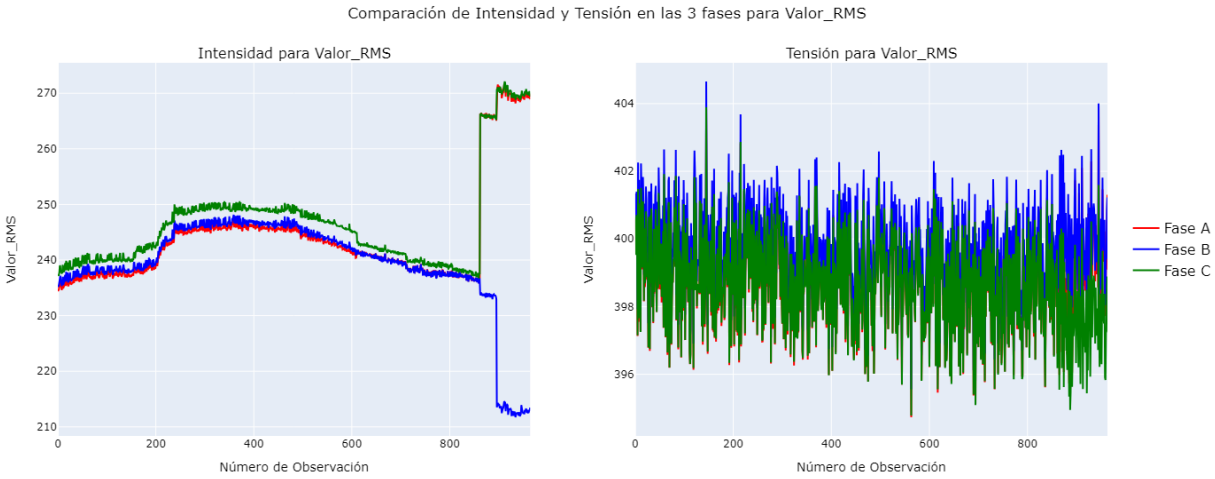


Figura A.21: Distribución de Valor_RMS

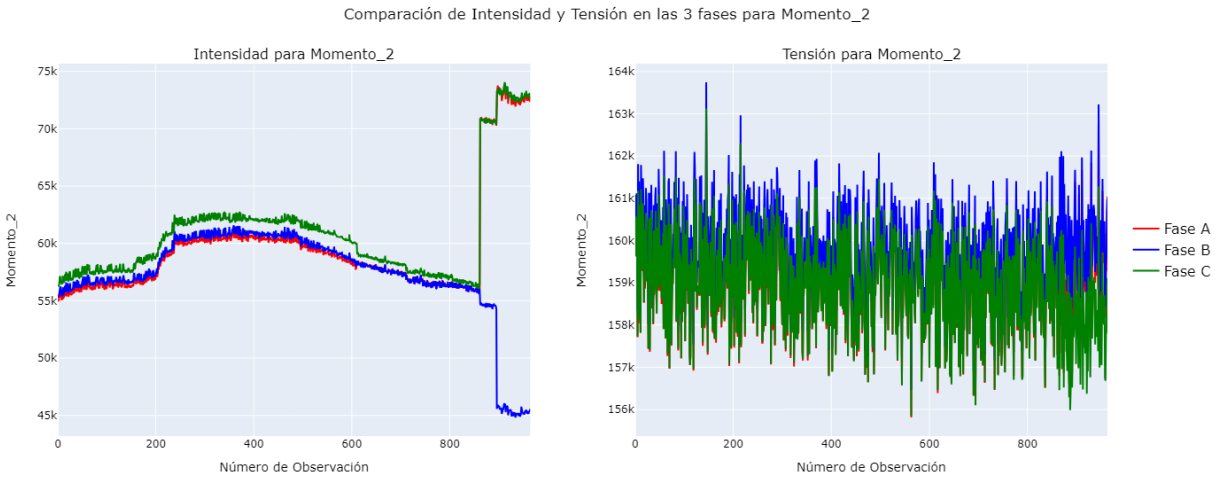


Figura A.22: Distribución de Momento_2

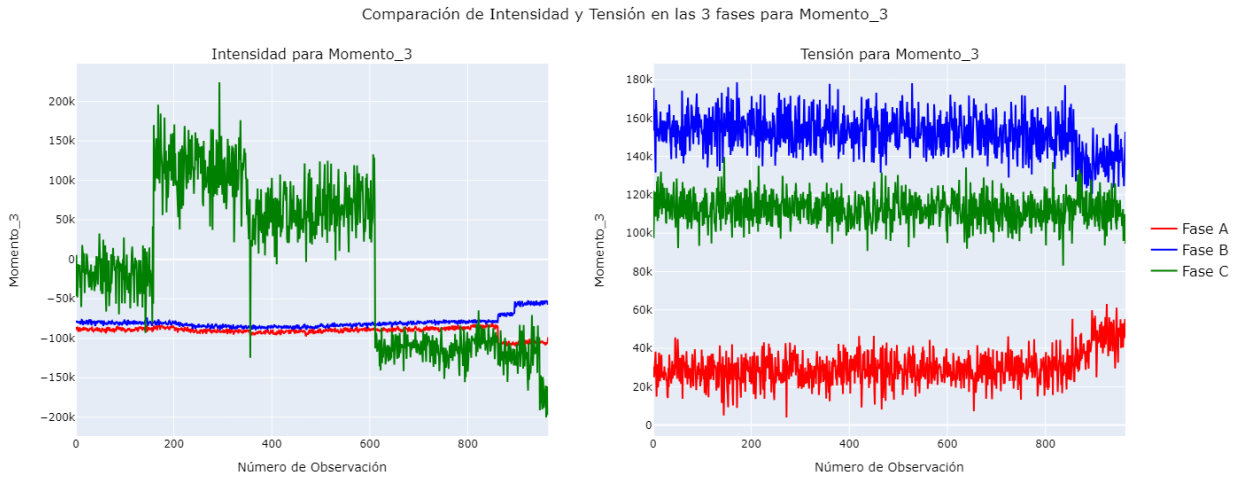


Figura A.23: Distribución de Momento_3

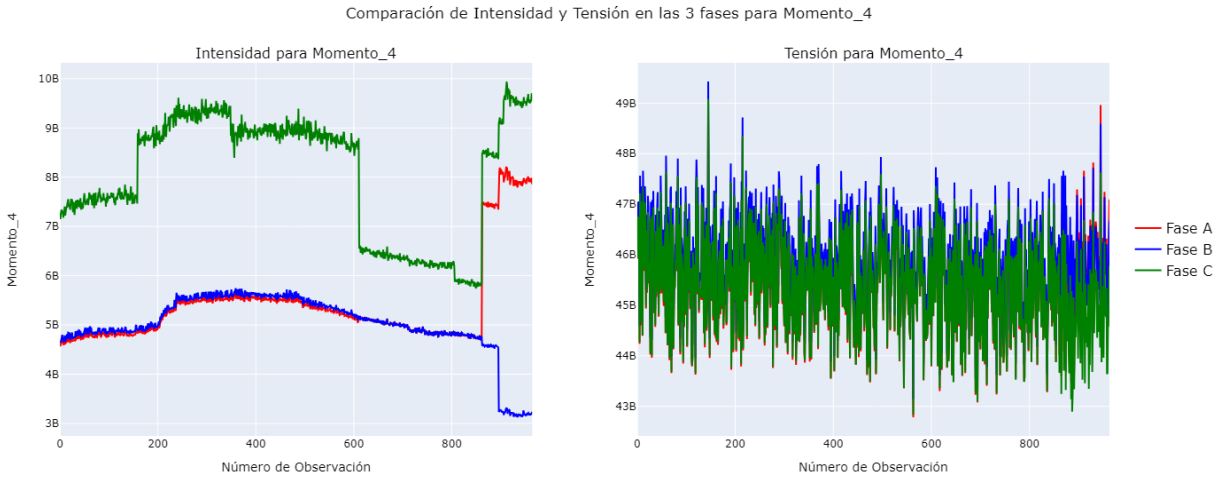


Figura A.24: Distribución de Momento_4

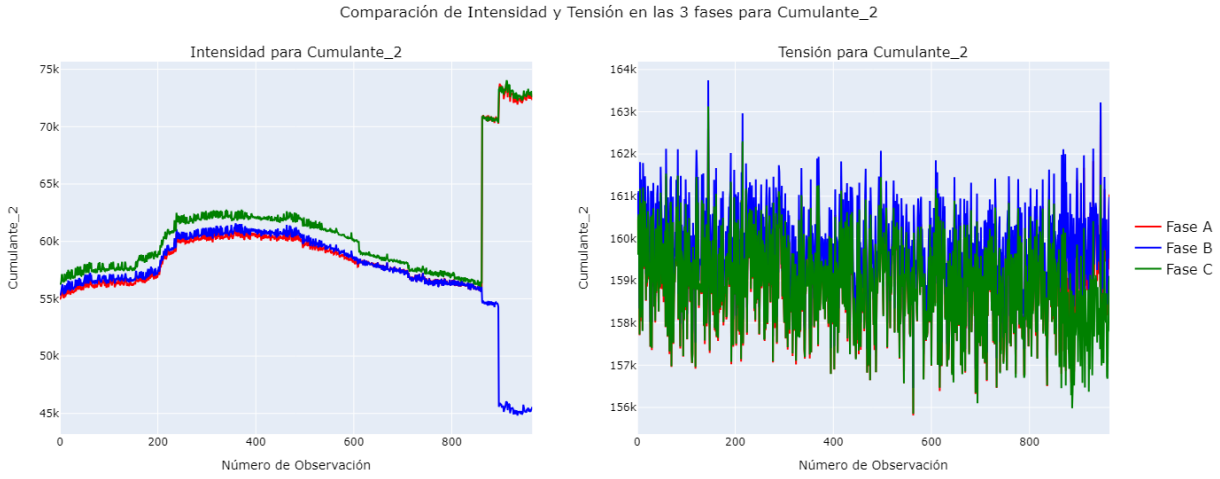


Figura A.25: Distribución de Cumulante_2

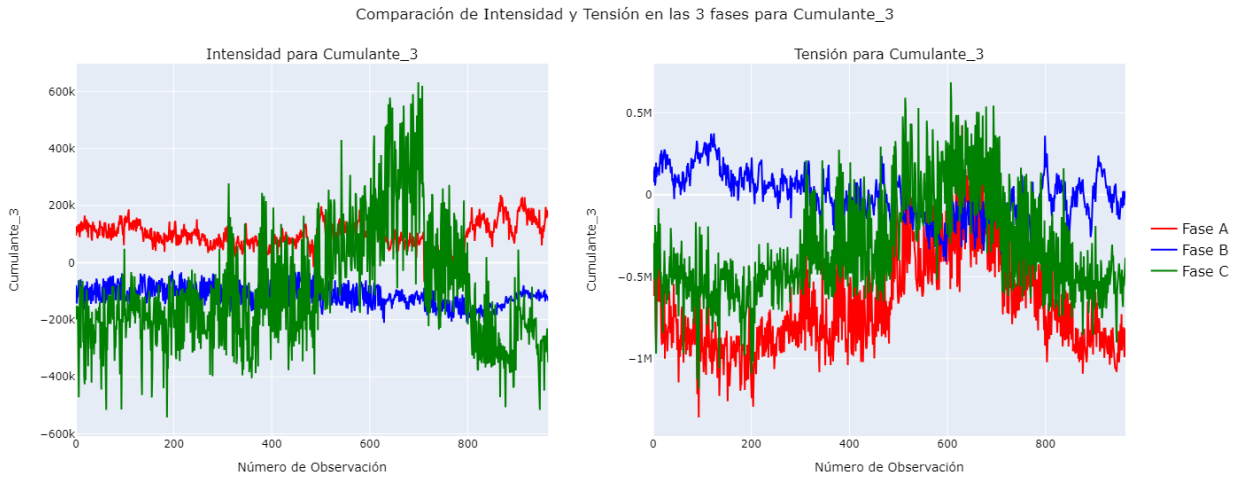


Figura A.26: Distribución de Cumulante_3

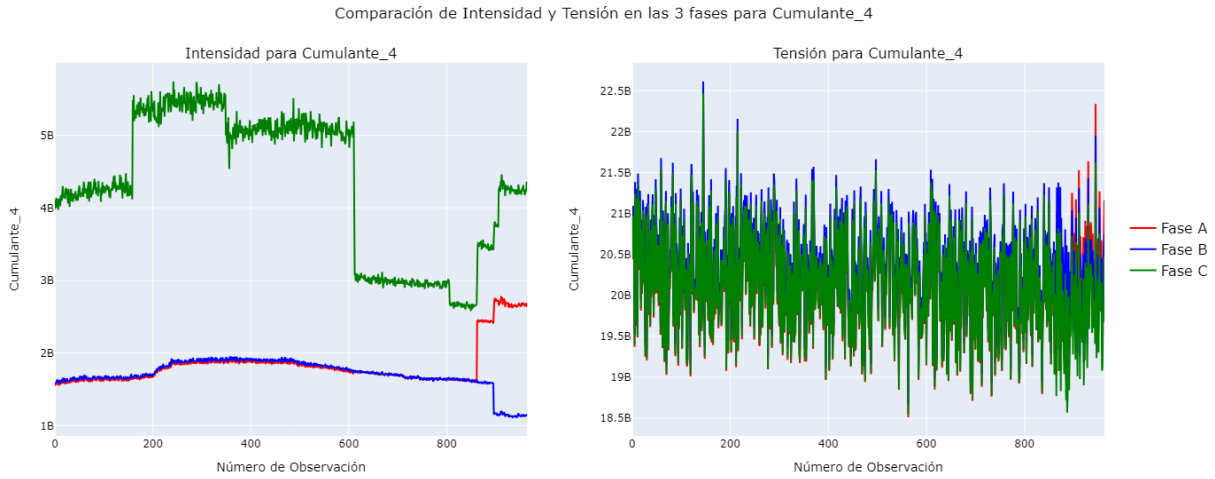


Figura A.27: Distribución de Cumulante_4

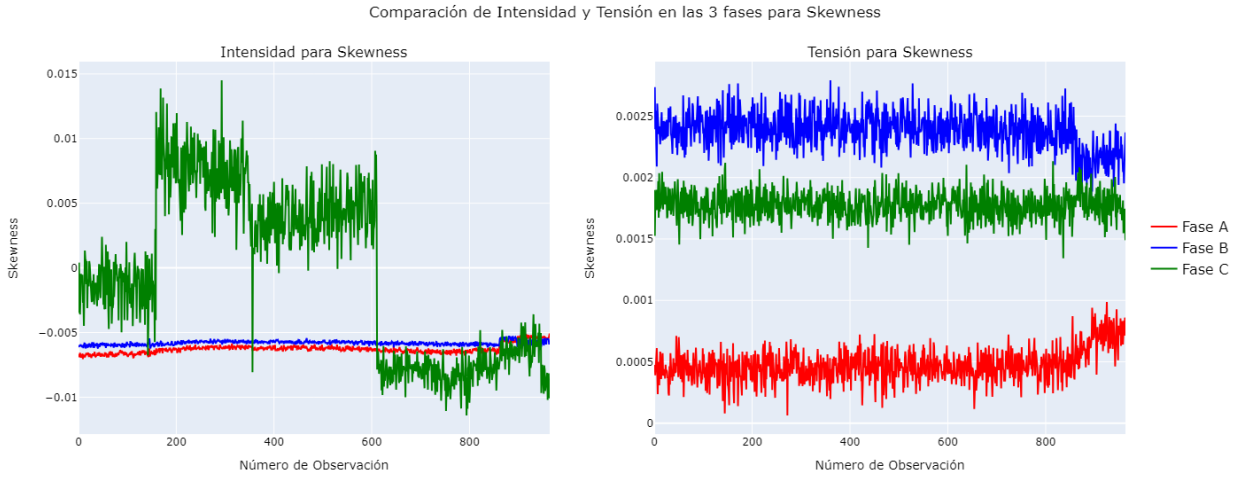


Figura A.28: Distribución de Skewness

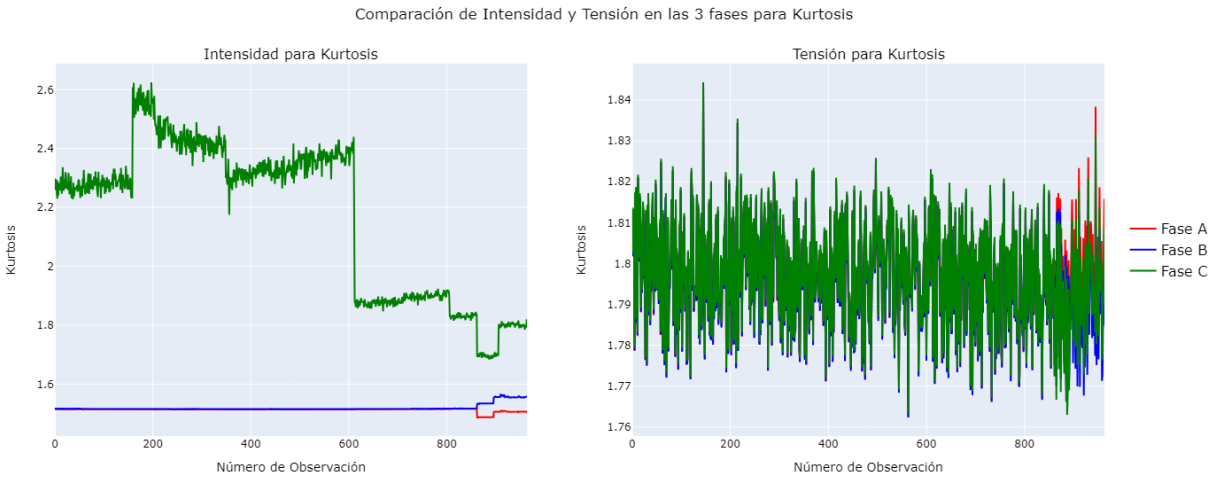


Figura A.29: Distribución de Kurtosis

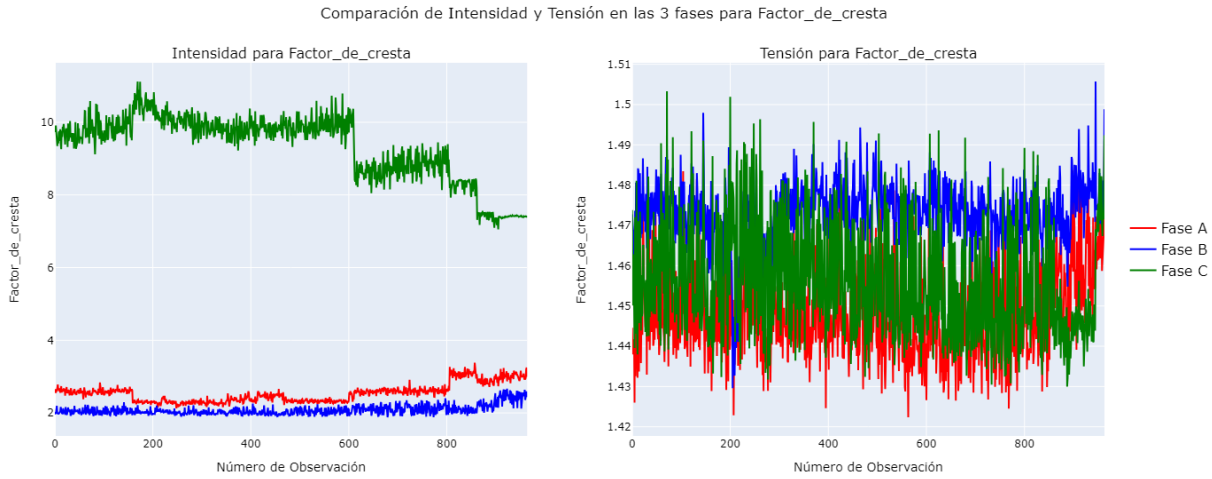


Figura A.30: Distribución de Factor_cresta

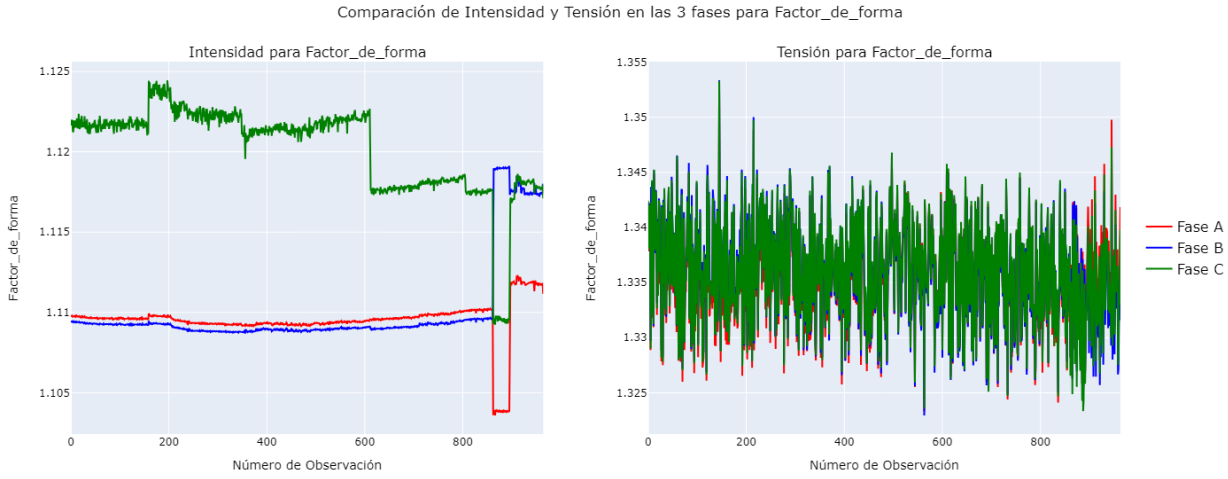


Figura A.31: Distribución de Factor_forma

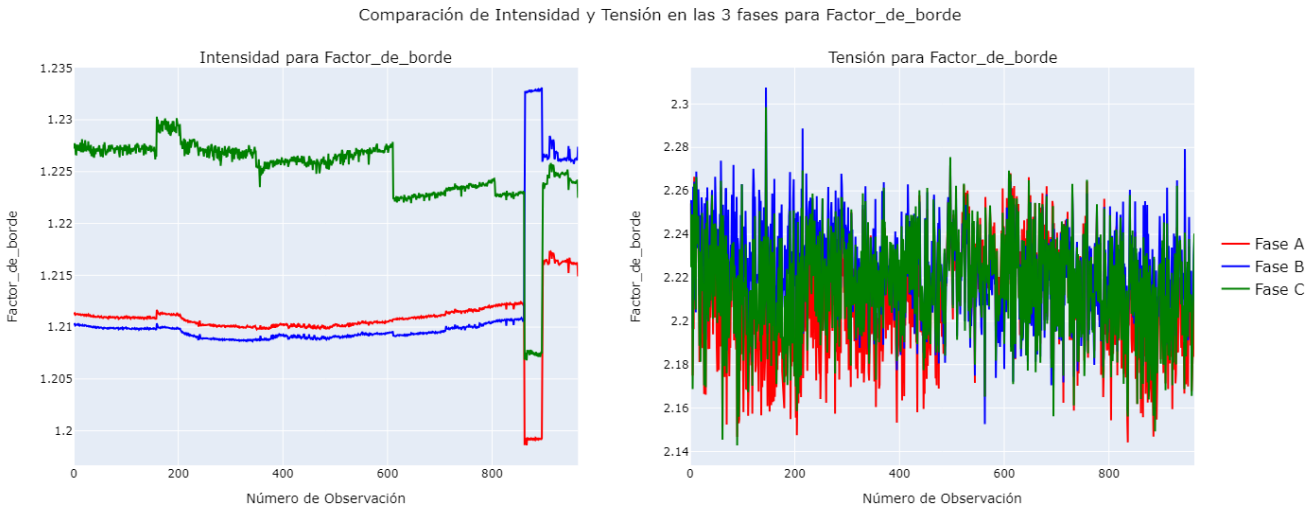


Figura A.32: Distribución de Factor_borde

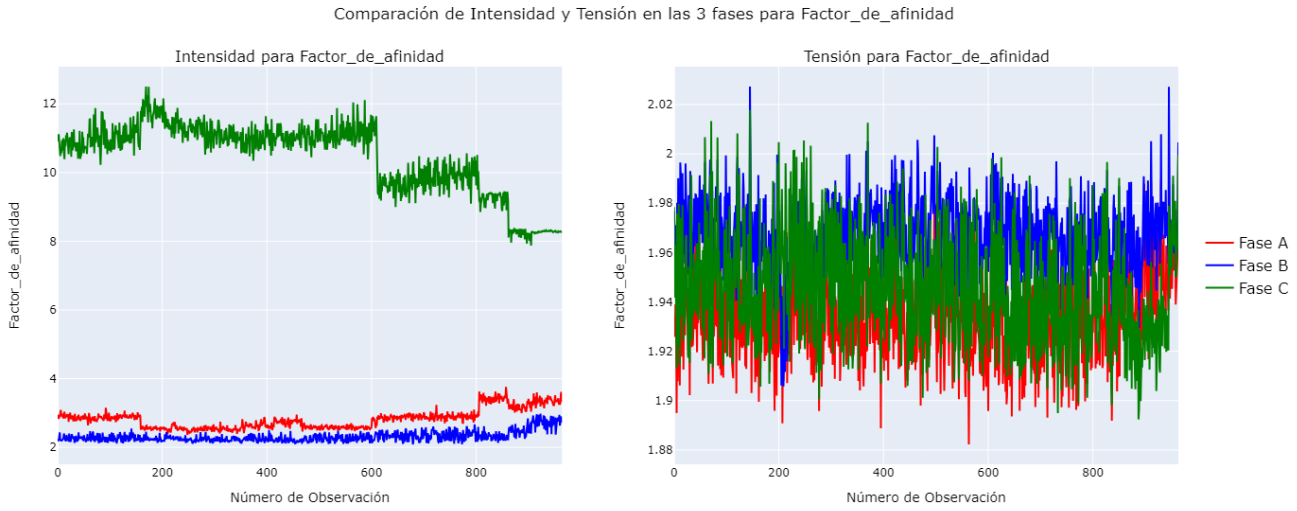


Figura A.33: Distribución de Factor_afinidad

A.2.4. Correlaciones de las variables en Aljibes

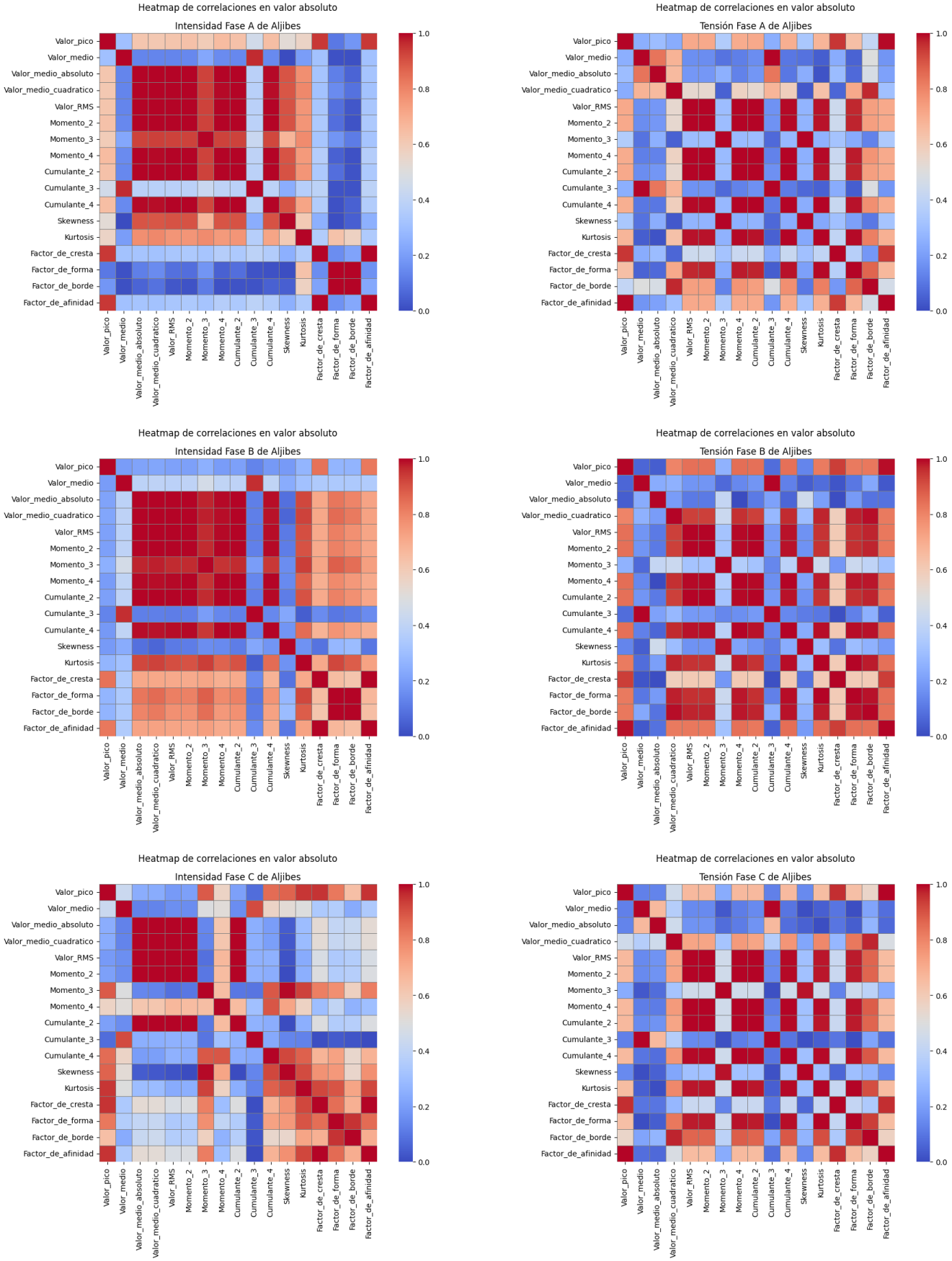


Figura A.34: Correlaciones de las variables de las fases de tensión e intensidad en Aljibes

A.2.5. Clustering sobre Aljibes

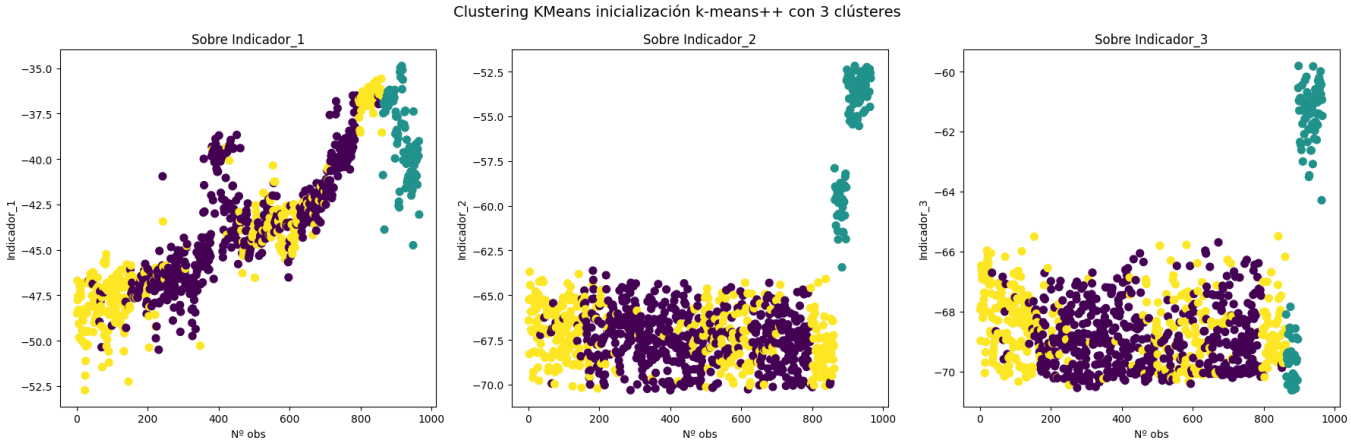


Figura A.35: KMeans con inicialización kmeans++ sobre Aljibes

Para el siguiente clustering agloremerativo, los resultados con los métodos *single*, *complete*, *ward* y *average* son equivalentes:

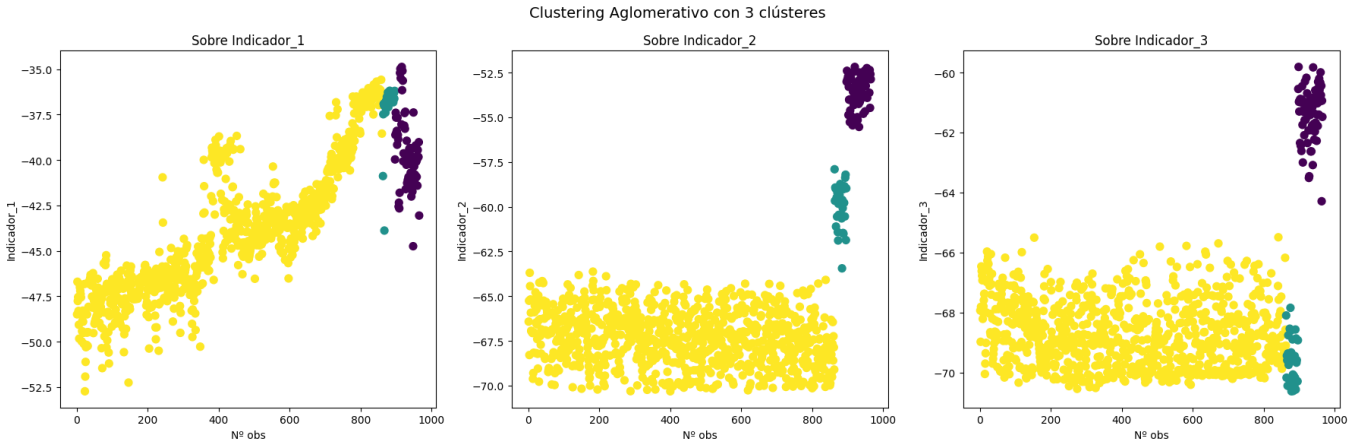


Figura A.36: Clusteting Aglomerativo sobre Aljibes

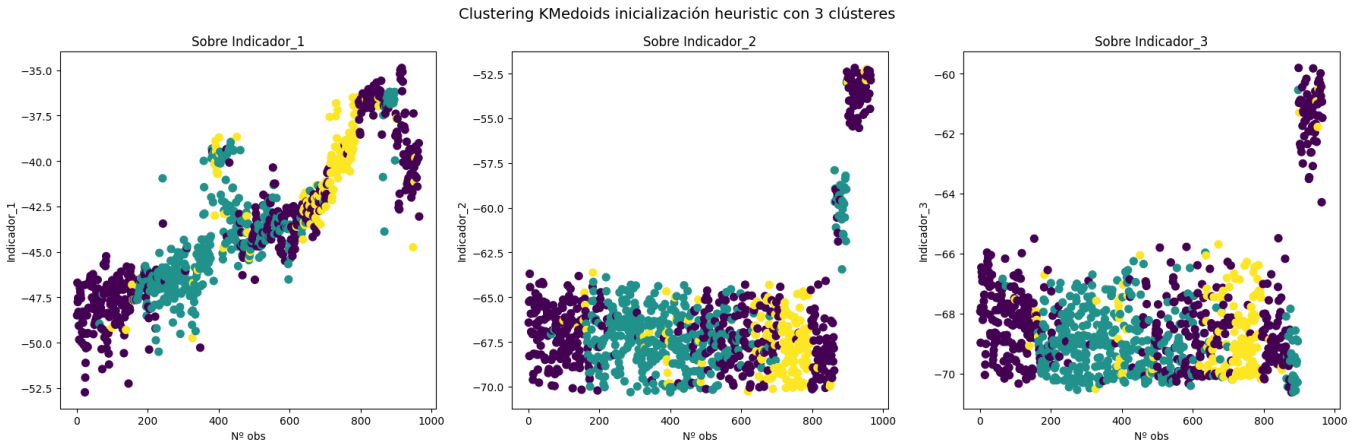


Figura A.37: KMedoids con inicialización heurística sobre Aljibes

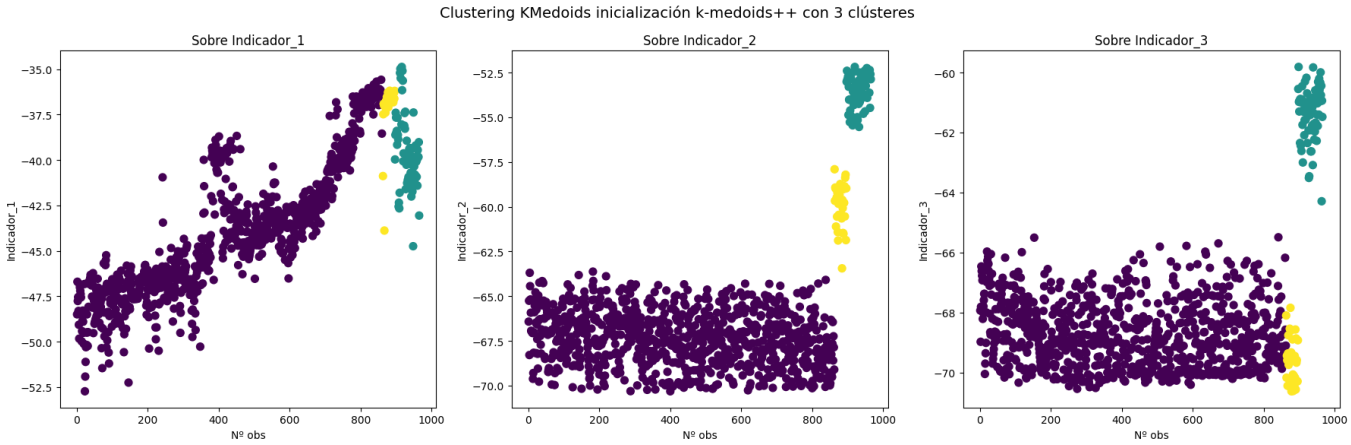


Figura A.38: KMedoids con inicialización k-medoids++ sobre Aljibes

Para DBSCAN, si se fija el número de clústeres en tres, se obtienen los mismos resultados (figura A.39) con los siguientes parámetros:

- Clústeres = 3, min_samples = [8, 10], eps = 0.4
- Clústeres = 3, min_samples = [3, 10], eps = 0.7
- Clústeres = 3, min_samples = [1, 10], eps = [0.8, 0.9, 1.0]

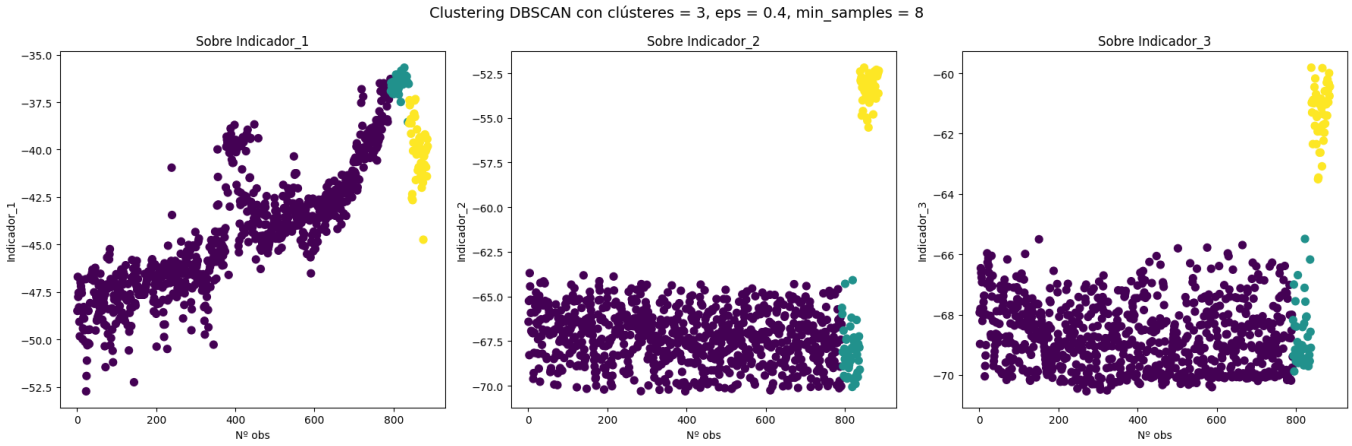


Figura A.39: DBSCAN sobre Aljibes

A.2.6. Diferencias entre las correlaciones en intensidad fase A

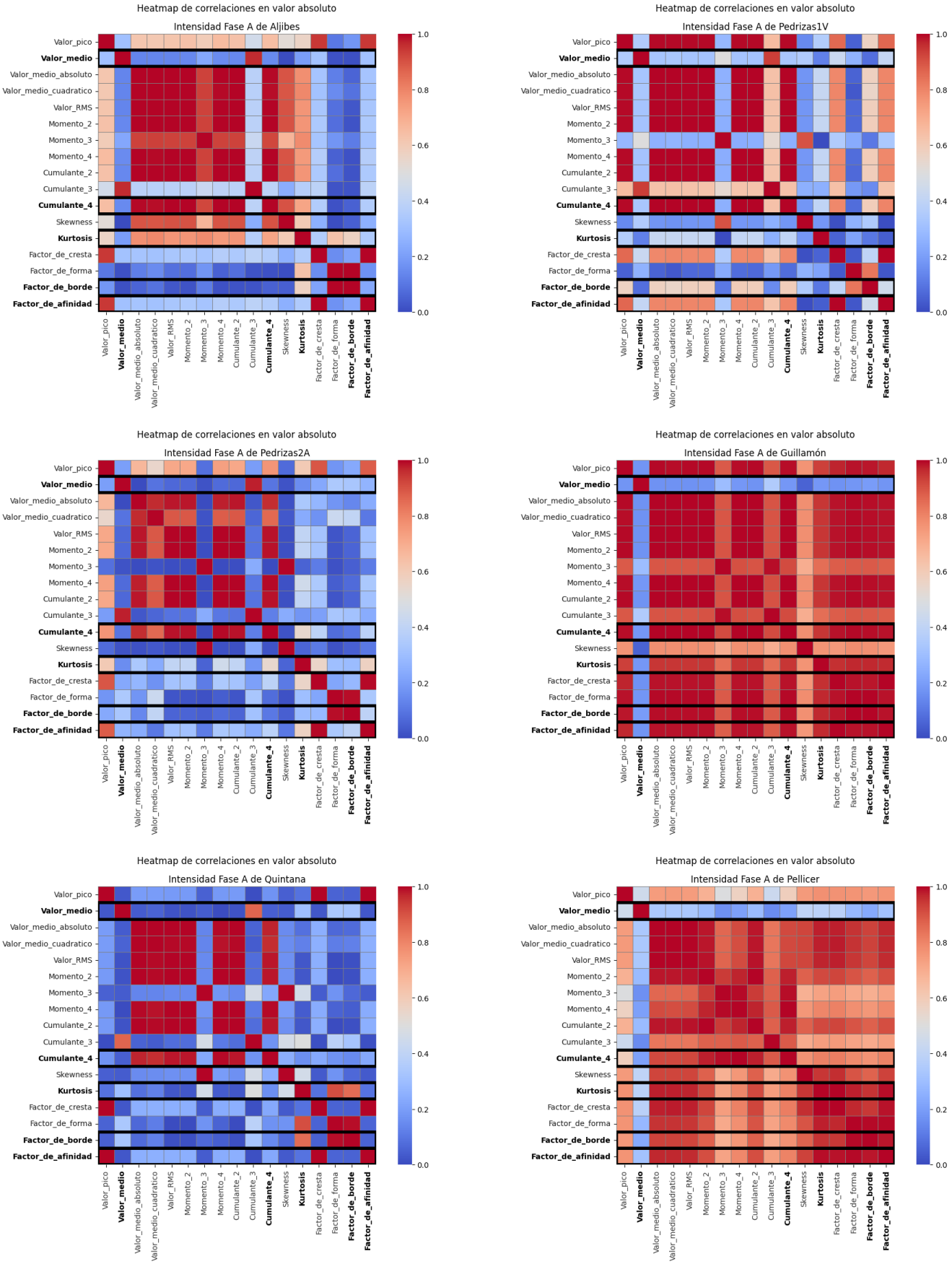


Figura A.40: Correlaciones en intensidad fase A para todos los pozos

A.3. Resultados

Durante esta sección se presentarán gráficos complementarios a los resultados mencionados a lo largo de la memoria, procedentes de los análisis mencionados y correspondientes al código desarrollado.

A.3.1. Análisis Discriminante Lineal

Gráficos complementarios a los resultados procedentes del Análisis Discriminante Lineal.

Predicción sobre otras fases

Matriz de confusión sobre algunas de las combinaciones entre conjunto de entrenamiento y conjunto de predicción.

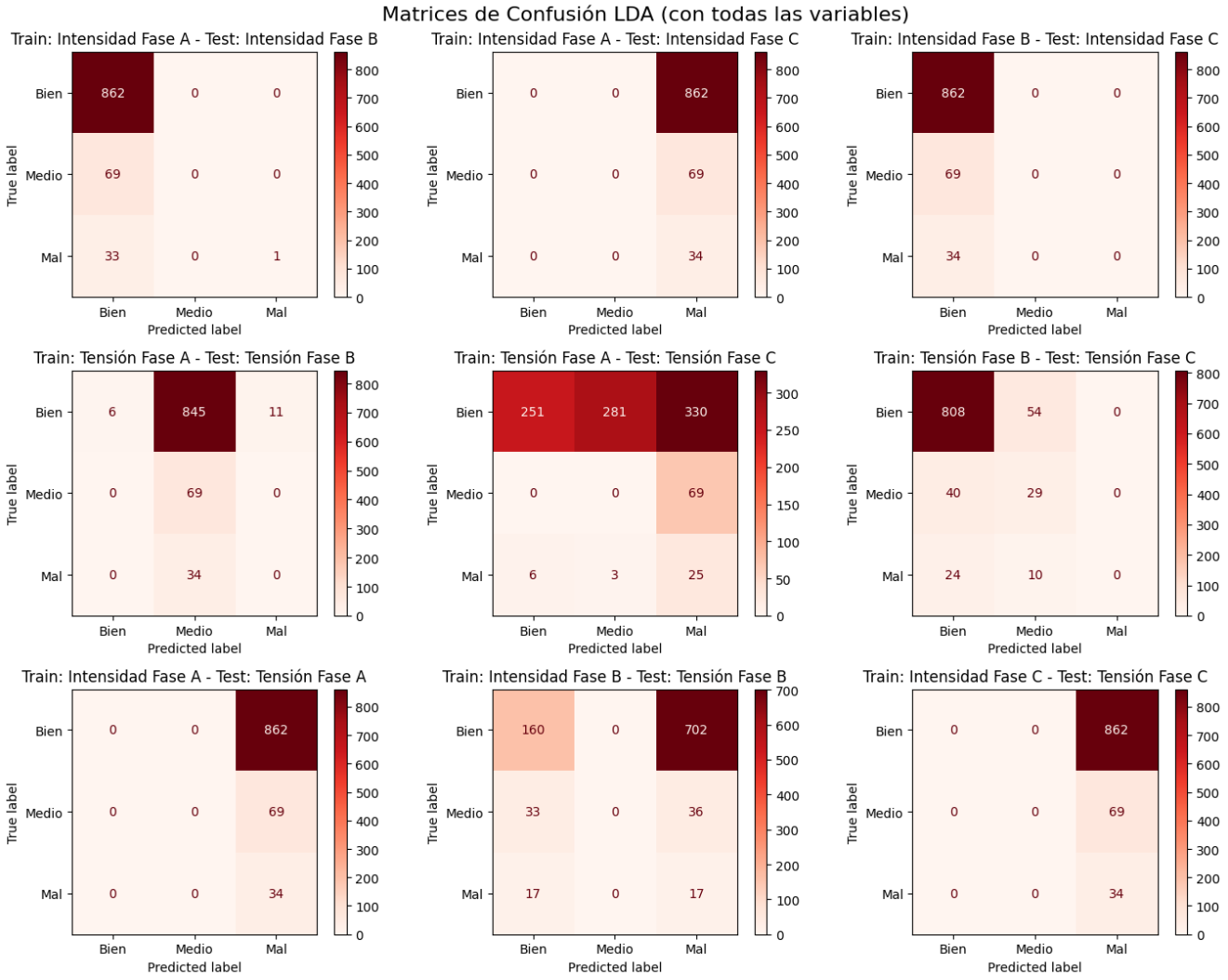


Figura A.41: Matrices de confusión del LDA entre las distintas fases y magnitudes de Aljibes

Predicción sobre otros pozos

Gráfico circular que muestra el porcentaje de observaciones asignadas a cada categoría, acompañado de una barra que indica en qué punto temporal de las mediciones se ha asignado un estado u otro. Este gráfico proviene de los resultados del Análisis Discriminante Lineal.

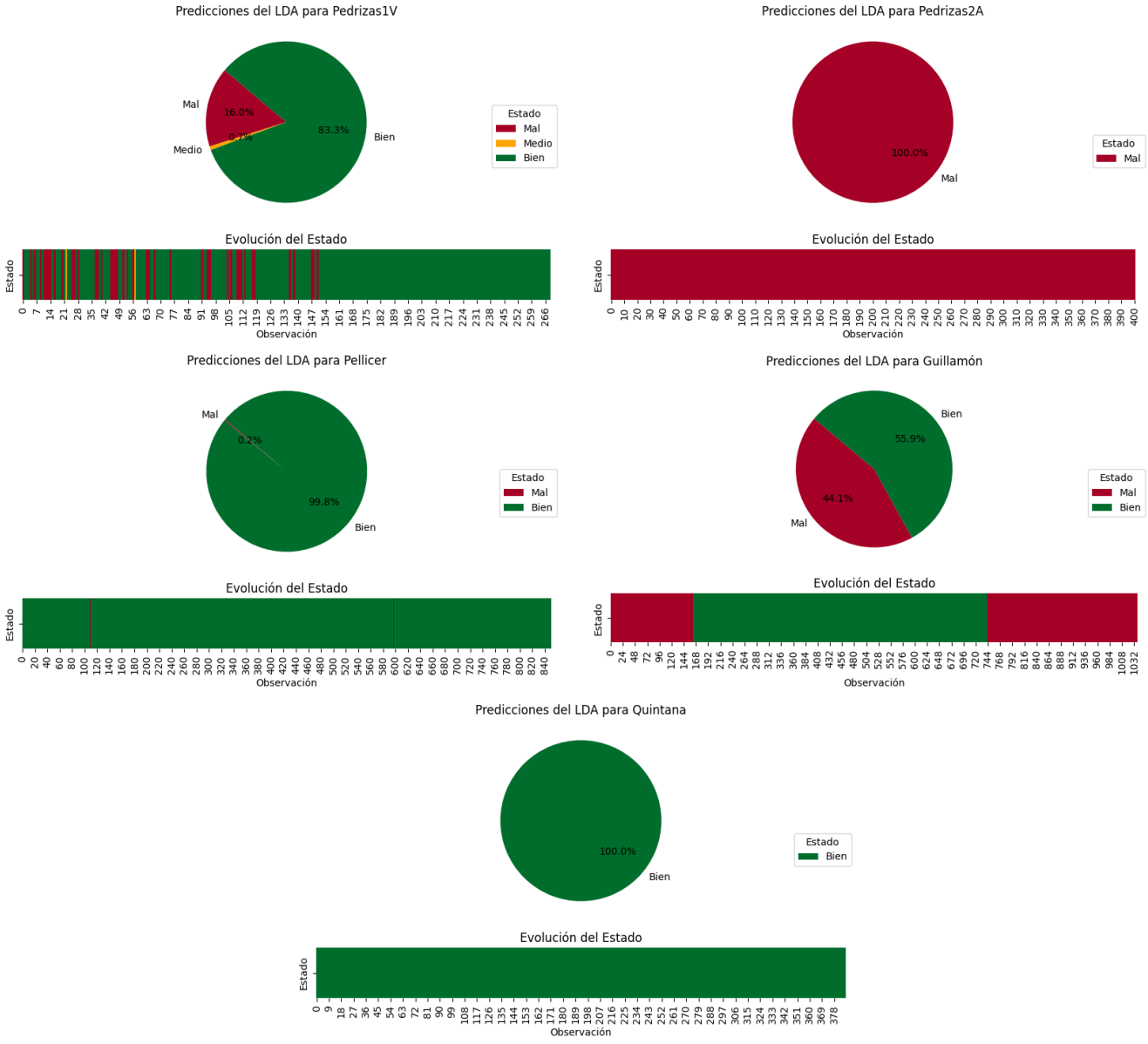


Figura A.42: Predicciones con el clasificador del LDA en los distintos pozos

A.3.2. Árbol de Decisión

Gráficos complementarios a los resultados procedentes del árbol de decisión.

Predicción sobre otras fases

Representación del árbol de decisión entrenado con la fase A de tensión de Aljibes, más profundo que si hubiera sido entrenado con la misma fase para la intensidad.

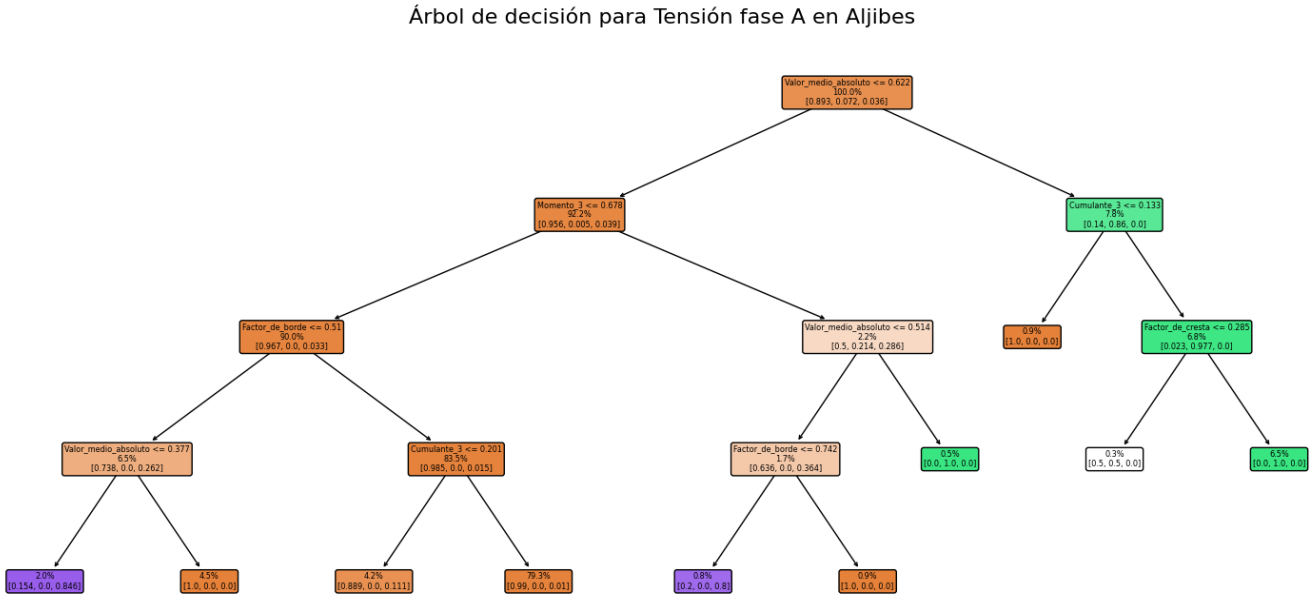


Figura A.43: Ejemplo de árbol de decisión para tensión en Aljibes

Predicción sobre otros pozos

Gráfico circular que muestra el porcentaje de observaciones asignadas a cada categoría, acompañado de una barra que indica en qué punto temporal de las mediciones se ha asignado un estado u otro. Este gráfico proviene de los resultados del árbol de decisión.

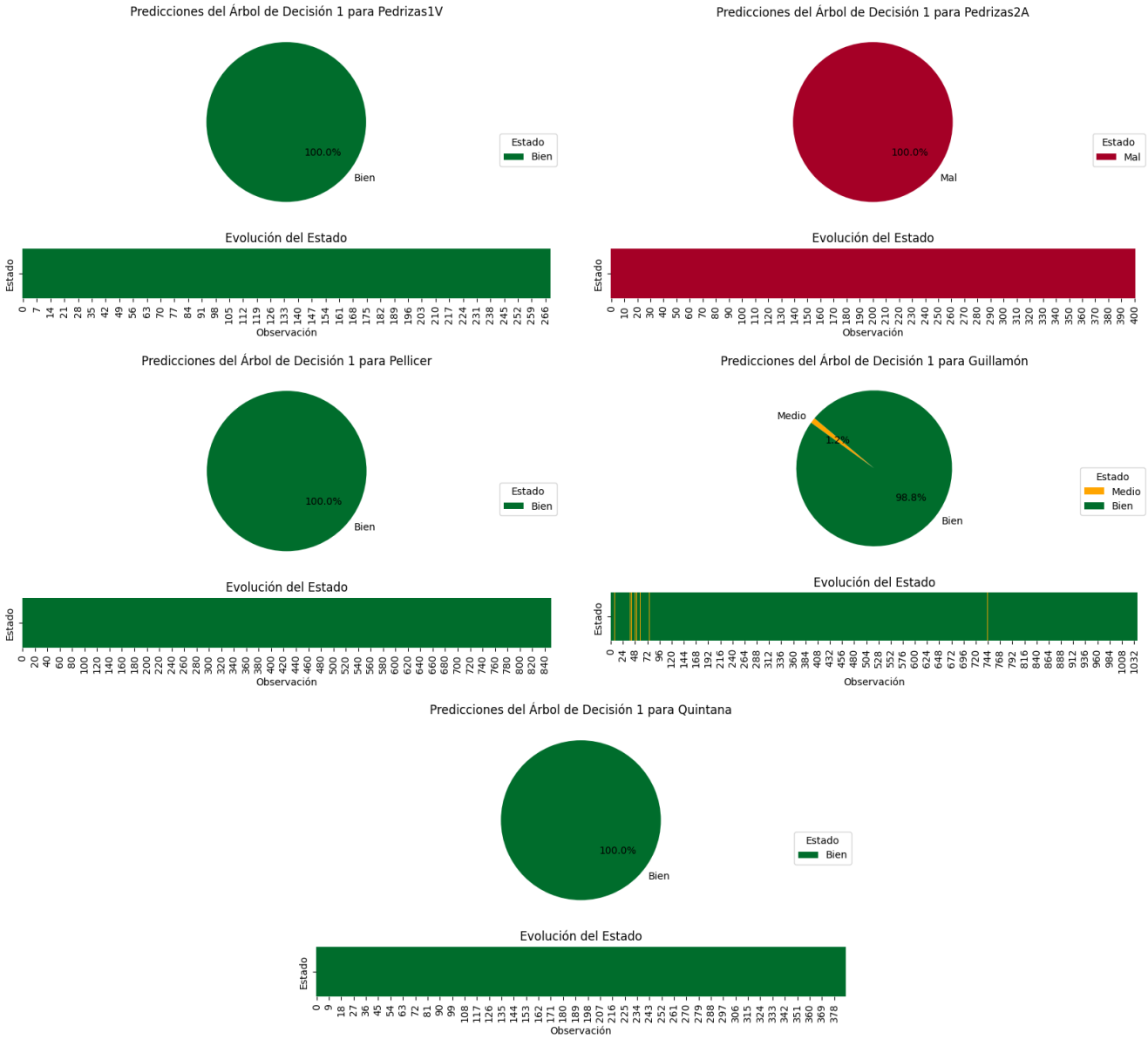


Figura A.44: Predicciones con el árbol de decisión 1 en los distintos pozos

A.3.3. Random Forest

Gráficos complementarios a los resultados procedentes del bosque aleatorio.

Predicción sobre otras fases

Gráfico de barras con las distintas importancias en Random Forest relativas a cada variable, para cada fase de intensidad y de tensión.

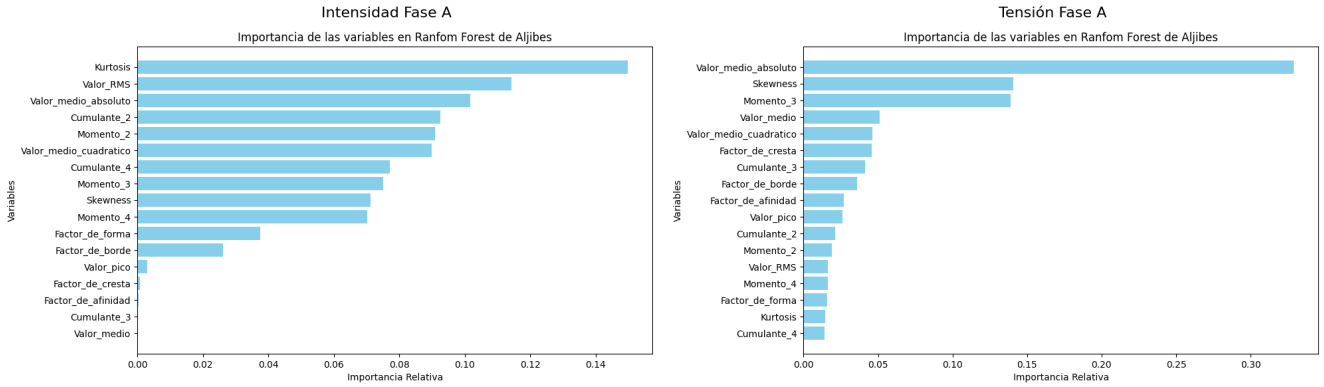


Figura A.45: Importancia de las variables en la clasificación de Random Forest en Aljibes Fase A

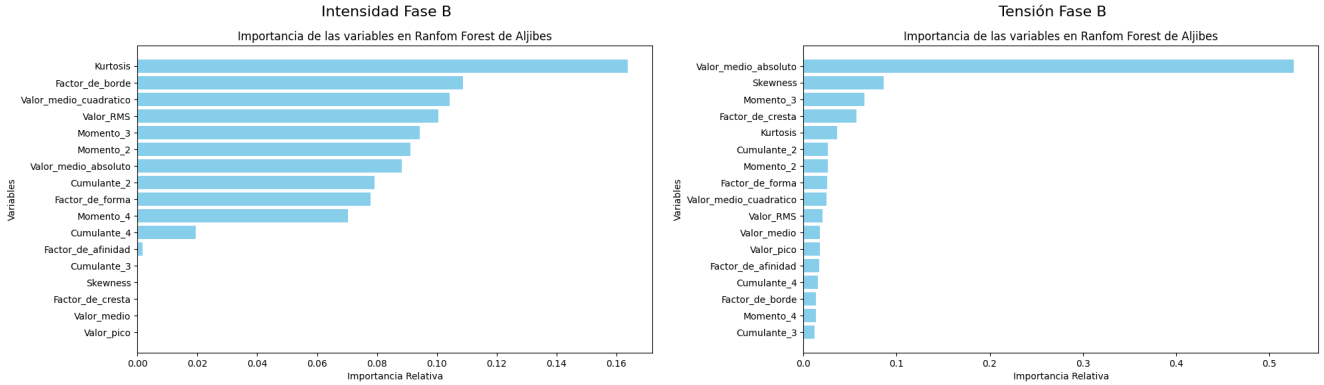


Figura A.46: Importancia de las variables en la clasificación de Random Forest en Aljibes Fase B

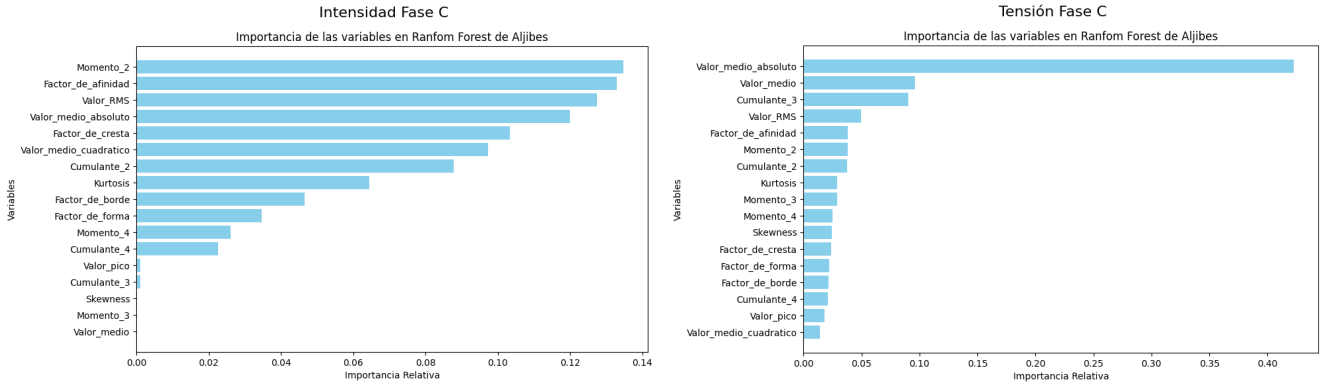


Figura A.47: Importancia de las variables en la clasificación de Random Forest en Aljibes Fase C

Predicción sobre otros pozos

Gráfico circular que muestra el porcentaje de observaciones asignadas a cada categoría, acompañado de una barra que indica en qué punto temporal de las mediciones se ha asignado un estado u otro. Este gráfico proviene de los resultados del *Random Forest*.

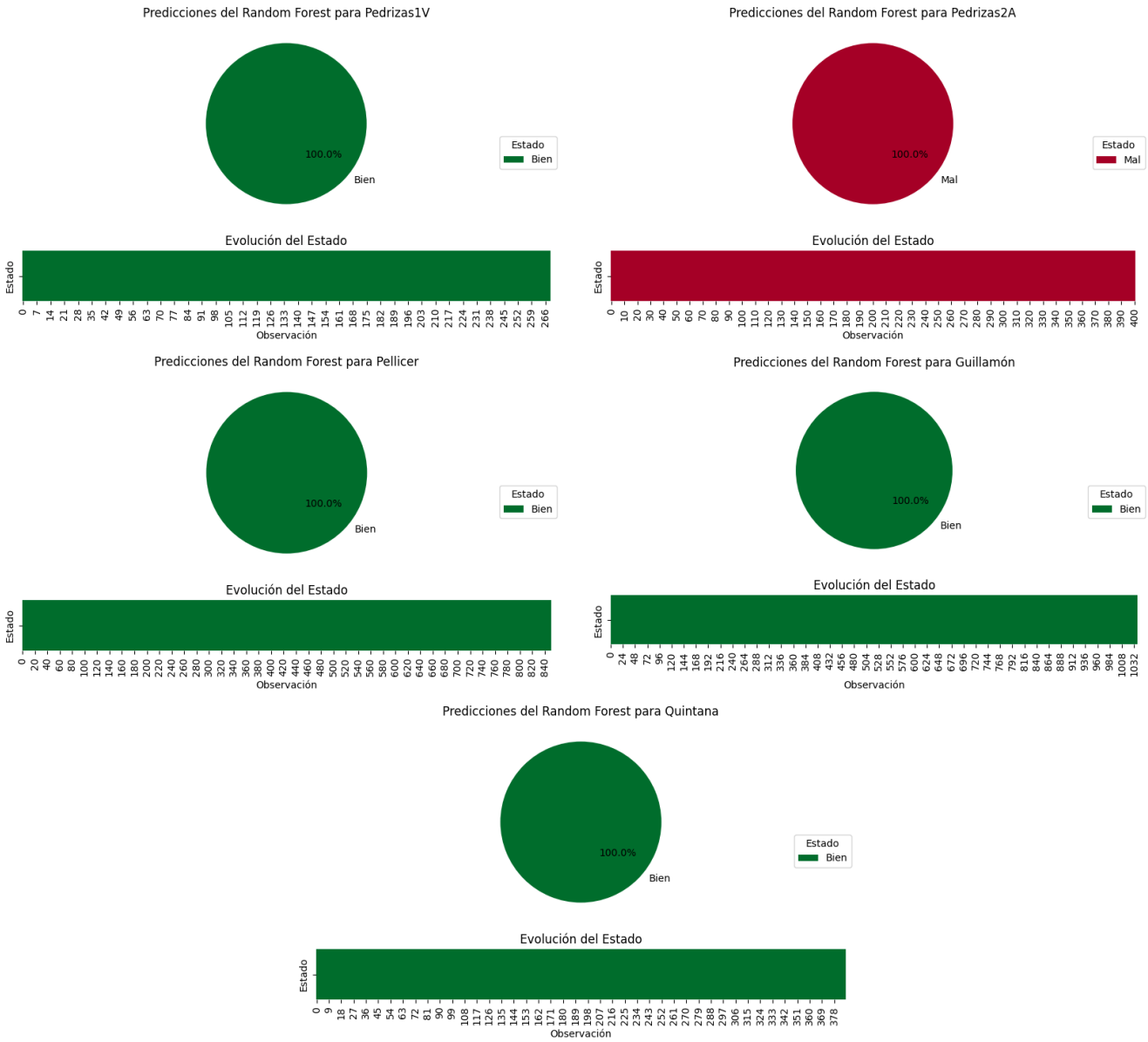


Figura A.48: Predicciones con el clasificador del Random Forest en los distintos pozos

A.3.4. XGBoost

Gráficos complementarios a los resultados procedentes del *boosting* XGBoost.

Predicción sobre otras fases

Gráfico de barras con las distintas importancias en XGBoost relativas a cada variable, para cada fase de intensidad y de tensión.

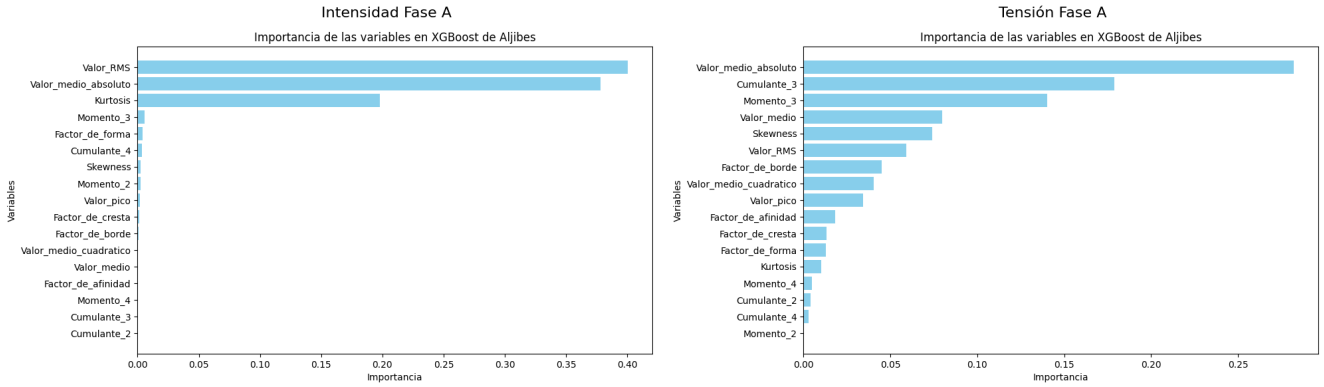


Figura A.49: Importancia de las variables en la clasificación de XGBoost en Aljibes Fase A

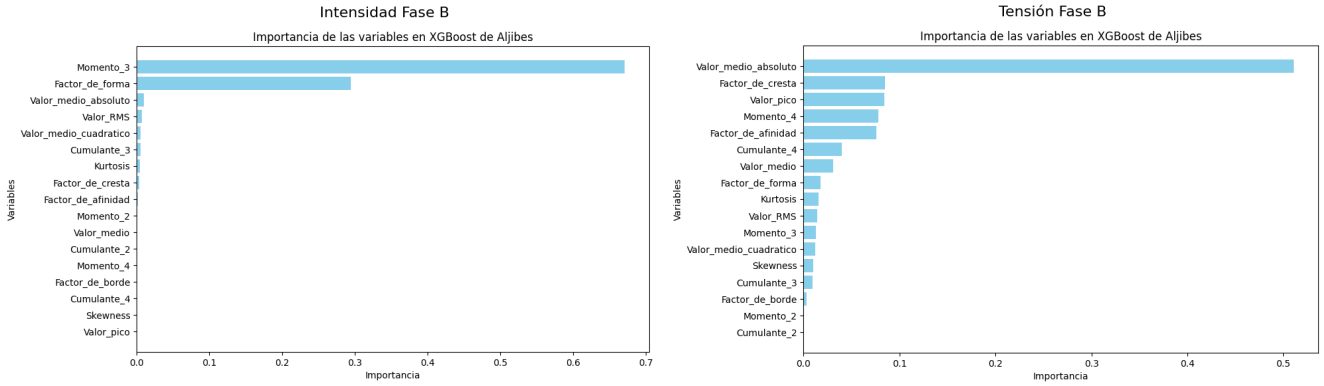


Figura A.50: Importancia de las variables en la clasificación de XGBoost en Aljibes Fase B

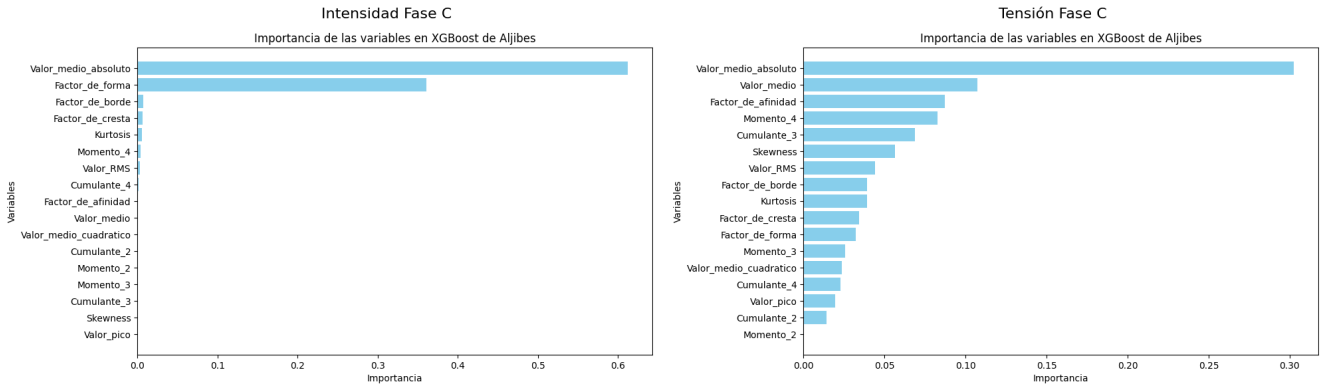


Figura A.51: Importancia de las variables en la clasificación de XGBoost en Aljibes Fase C

Predicción sobre otros pozos

Gráfico circular que muestra el porcentaje de observaciones asignadas a cada categoría, acompañado de una barra que indica en qué punto temporal de las mediciones se ha asignado un estado u otro. Este gráfico proviene de los resultados del *boosting* XGBoost.

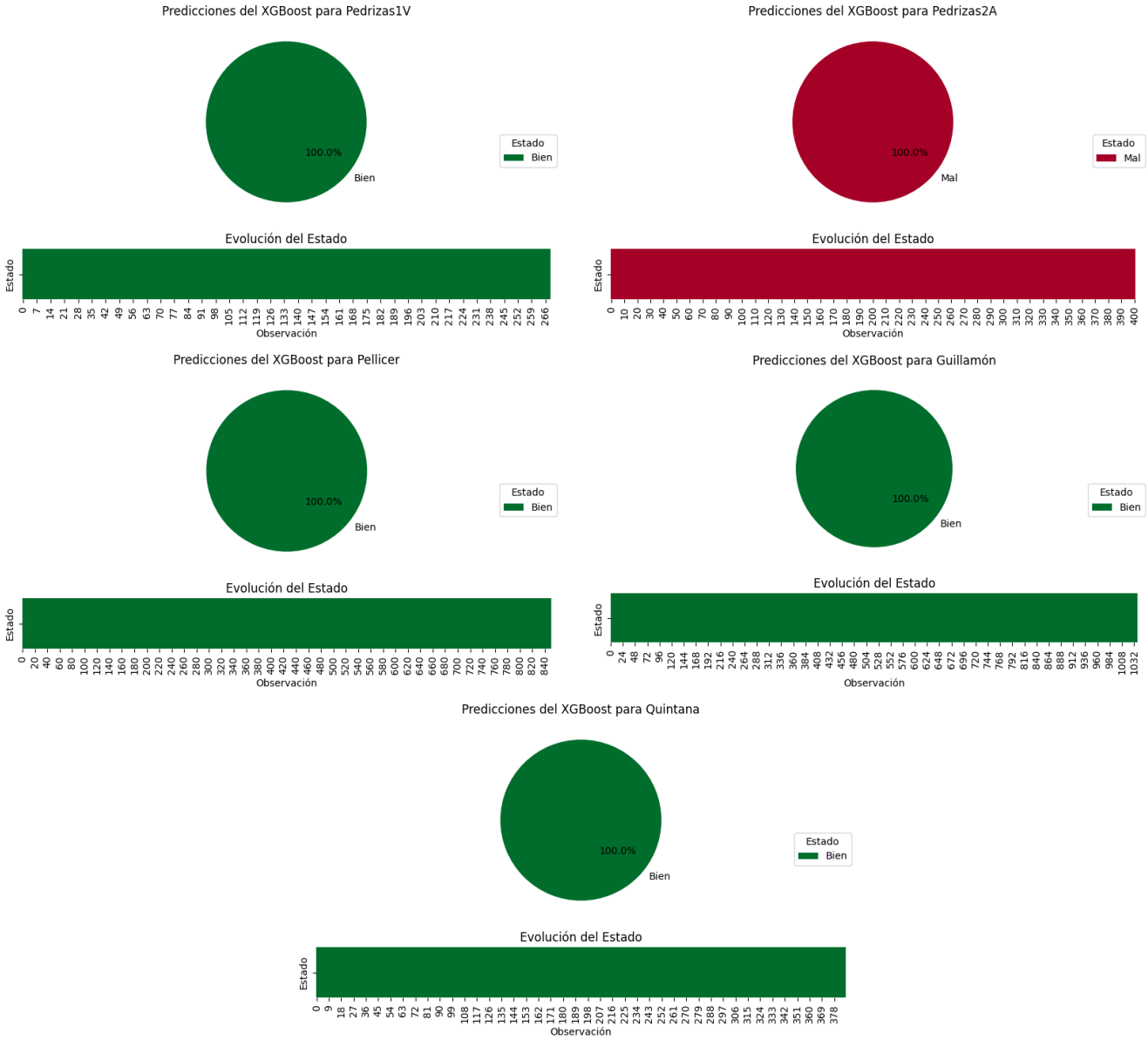


Figura A.52: Predicciones con el clasificador del XGBoost en los distintos pozos

Apéndice B

Código

En este segundo apéndice se muestra una selección principal del total del código desarrollado para llevar a cabo los análisis del proyecto. Se indica el conjunto de Aljibes como ejemplo base durante todo el código, siendo análogo el uso para el resto de conjuntos.

B.1. Lectura

```
    aljibes = "data/DatosAljibes2_Actualizado.xlsx"

    tensionA_alj = pd.read_excel(aljibes, sheet_name='TensionA')
    tensionB_alj = pd.read_excel(aljibes, sheet_name='TensionB')
    tensionC_alj = pd.read_excel(aljibes, sheet_name='TensionC')
    intensidadA_alj = pd.read_excel(aljibes, sheet_name='IntensidadA')
    intensidadB_alj = pd.read_excel(aljibes, sheet_name='IntensidadB')
    intensidadC_alj = pd.read_excel(aljibes, sheet_name='IntensidadC')
```

Listing B.1: Lectura de datos

B.2. Preprocesamiento

B.2.1. Tratamiento de NA's: Interpolación

```
for dataset in [tensionA_alj, tensionB_alj, tensionC_alj, intensidadA_alj, intensidadB_alj,
    intensidadC_alj]:
    dataset['Indicador_1'] = dataset['Indicador_1'].interpolate(method='linear')
    dataset['Indicador_2'] = dataset['Indicador_2'].interpolate(method='linear')
    dataset['Indicador_3'] = dataset['Indicador_3'].interpolate(method='linear')
```

Listing B.2: Interpolación de valores ausentes

B.2.2. Eliminación de Outliers

```
def remove_outliers(df, lista):
   return df.drop(lista)

outliers = [...]
for dataset in [tensionA_alj, tensionB_alj, tensionC_alj, intensidadA_alj, intensidadB_alj,
    intensidadC_alj]:
    dataset = remove_outliers(dataset, outliers)
```

Listing B.3: Borrado de outliers

B.2.3. Gráficos Descriptivos

```
num_obs = intensidadA_alj.groupby('Fecha').size().reset_index(name=obs)
dates = pd.date_range(start='2020-02-02', end='2020-11-11', freq='D')
dates_df = pd.DataFrame({'Fecha': dates})
num_obs_tot = pd.concat([dates_df, num_obs]).fillna(0)

plt.figure(figsize=(26, 12))
plt.bar(num_obs_tot['Fecha'], num_obs_tot[obs])
plt.xlabel('Fecha', fontsize=20)
plt.ylabel('Numero de observaciones', fontsize=20)
plt.title('Numero de observaciones por dia ALJIBES', fontsize=24)
plt.ylim(0, 9.5)
plt.xticks(rotation=45, fontsize=16)
plt.yticks(fontsize=16)
plt.grid(True, which='both', linestyle='--', linewidth=0.5, color='gray', alpha=0.5)
plt.tight_layout()
plt.show()
```

Listing B.4: Conteo de mediciones por día

```
def plot_dual_line_graphs(dataA_I, dataB_I, dataC_I, dataA_T, dataB_T, dataC_T, var):
    fig = make_subplots(rows=1, cols=2, subplot_titles=('Intensidad para ' + var, 'Tension
        para ' + var))

    # Intensidad
    fig.add_trace(go.Scatter(x=dataA_I[var].index, y=dataA_I[var], mode='lines', name='Fase
        A', line=dict(color='red'), showlegend=True, legendgroup='group1'), row=1, col=1)
    fig.add_trace(go.Scatter(x=dataB_I[var].index, y=dataB_I[var], mode='lines', name='Fase
        B', line=dict(color='blue'), showlegend=True, legendgroup='group1'), row=1, col=1)
    fig.add_trace(go.Scatter(x=dataC_I[var].index, y=dataC_I[var], mode='lines', name='Fase
        C', line=dict(color='green'), showlegend=True, legendgroup='group1'), row=1, col=1)

```

```
    # Tension
    fig.add_trace(go.Scatter(x=dataA_T[var].index, y=dataA_T[var], mode='lines', name='Fase A', line=dict(color='red'), showlegend=False), row=1, col=2)
    fig.add_trace(go.Scatter(x=dataB_T[var].index, y=dataB_T[var], mode='lines', name='Fase B', line=dict(color='blue'), showlegend=False), row=1, col=2)
    fig.add_trace(go.Scatter(x=dataC_T[var].index, y=dataC_T[var], mode='lines', name='Fase C', line=dict(color='green'), showlegend=False), row=1, col=2)

    for col in range(1, 3):
        fig.update_xaxes(title_text='Numero de Observacion', row=1, col=col)

    fig.update_layout(title_text='Comparacion de Intensidad y Tension en las 3 fases para ' + var, title_font=dict(color='black'), font=dict(color='black'), title_x=0.5, width=1400, height=600,
                      legend=dict(font=dict(size=16, color='black'), y=0.5))
    fig.show()

for est in intensidadA_alj.columns.tolist()[1:30]:
    plot_dual_line_graphs(intensidadA_alj, intensidadB_alj, intensidadC_alj, tensionA_alj, tensionB_alj, tensionC_alj, est)
```

Listing B.5: Gráficos de líneas para ver la distribución

```
def indAlj_groups(df):
    colors = {"Bien": (0/255, 109/255, 44/255), "Medio": "orange", "Mal": (165/255, 0/255, 38/255)}

    fig, axs = plt.subplots(1, 3, figsize=(18, 6))
    for i, indic in enumerate(["Indicador_1", "Indicador_2", "Indicador_3"], start=0):
        column_group = f"Grupo_ind_{i+1}"
        data = df[indic]
        for value, color in colors.items():
            ind_val = [i for i, v in enumerate(df[column_group]) if v == value]
            data_val = [data.iloc[i] for i in ind_val]
            axs[i].scatter(ind_val, data_val, color=color, label=f'Estado: {value}', alpha=0.8)

        axs[i].set_xlabel('Numero de Observacion')
        axs[i].set_ylabel('Valor de la Observacion')
        axs[i].set_title(f'Diagrama de Puntos para {indic}')
        axs[i].legend()
    plt.tight_layout()
    plt.show()

indAlj_groups(intensidadA_alj)
```

Listing B.6: Dispersión de los indicadores de Aljibes

```
dataset_name = ['Intensidad Fase A de Aljibes', 'Tension Fase A de Aljibes', 'Intensidad Fase B de Aljibes', 'Tension Fase B de Aljibes', 'Intensidad Fase C de Aljibes', 'Tension Fase C de Aljibes']
datasets = [intensidadA_alj.iloc[:,1:18], tensionA_alj.iloc[:,1:18], intensidadB_alj.iloc[:,1:18], tensionB_alj.iloc[:,1:18], intensidadC_alj.iloc[:,1:18], tensionC_alj.iloc[:,1:18]
]
figs_per_row = 2
rows = (len(datasets) + figs_per_row - 1) // figs_per_row
fig, axs = plt.subplots(rows, figs_per_row, figsize=(20, rows * 8))
for idx, (dataset, name) in enumerate(zip(datasets, dataset_name)):
    row = idx // figs_per_row
    col = idx % figs_per_row
    correlacion_mat = dataset.corr()
    sns.heatmap(abs(correlacion_mat), vmax=1, vmin=0, square=True, annot=False, fmt=".2f", cmap="coolwarm", linecolor='gray', linewidths=0.5, ax=axs[row, col])
    axs[row, col].set_title('Heatmap de correlaciones en valor absoluto', pad=30)
    axs[row, col].text(0.5, 1.025, name, fontsize=12, ha='center', va='center', transform=axs[row, col].transAxes)
plt.tight_layout(pad=3)
plt.show()
```

Listing B.7: Correlaciones entre las variables

```
datasets = ["Aljibes", "Pedrizas1V", "Pedrizas2A", "Guillamon", "Pellicer", "Quintana"]
dfs = [intensidadA_alj.iloc[:,1:], intensidadA_ped_1V.iloc[:,1:]
    ,intensidadA_ped_2A.iloc[:,1:], intensidadA_gui.iloc[:,1:], intensidadA_pelli.iloc[:,1:], intensidadA_quint2.iloc[:,1:]]
for i, df in enumerate(dfs):
    df['Dataframe'] = f'{datasets[i]}'
df_concat = pd.concat(dfs)
columns = intensidadA_alj.iloc[:,1:].columns.tolist()
for i in range(0, len(columns), 2):
    fig, axs = plt.subplots(1, 2, figsize=(16, 6))
    sns.boxplot(ax=axs[0], x='Dataframe', y=columns[i], data=df_concat, palette=["red","red","red", "green","green","green"])
    axs[0].set_title('Boxplot multiple de ' + columns[i] + " en intensidad fase A", fontsize=16)
    axs[0].set_xlabel('Pozo', fontsize=14)
    axs[0].set_ylabel(columns[i], fontsize=14)
    if i + 1 < len(columns):
        sns.boxplot(ax=axs[1], x='Dataframe', y=columns[i + 1], data=df_concat, palette=["red","red","red", "green","green","green"])
        axs[1].set_title('Boxplot multiple de ' + columns[i + 1] + " en intensidad fase A", fontsize=16)
        axs[1].set_xlabel('Pozo', fontsize=14)
        axs[1].set_ylabel(columns[i + 1], fontsize=14)
```

```
    red_patch = mpatches.Patch(color='red', label='Pozo con fallos')
    green_patch = mpatches.Patch(color='green', label='Pozo sin fallos')
    fig.legend(handles=[red_patch, green_patch], loc='upper center', ncol=2, fontsize=14,
        bbox_to_anchor=(0.5, 1.05))

    plt.tight_layout(rect=[0, 0, 1, 0.95])
    plt.show()
```

Listing B.8: Boxplot en Aljibes

B.2.4. Análisis de Componentes Principales

```
# Creacion del ACP
def define_pca(dfi, n_comp, mode):
    if mode != 1 and mode != 2 and mode != 3:
        print("Selecciona modo: 1-> NO NORM  2-> NORM  3->IND")
        return
    if mode == 1:
        df = dfi.iloc[:,4:21] # DATOS NO NORMALIZADOS
    elif mode == 2:
        df = dfi.iloc[:,[21, 22, 23, 24, 8, 9, 25, 26, 27, 28, 29, 15, 16, 17, 18, 19, 20]] #
            DATOS NORMALIZADOS
    elif mode == 3:
        df = dfi.iloc[:,30:33] # INDICADORES

    scaler = StandardScaler()
    scaled_df = pd.DataFrame(scaler.fit_transform(df))

    pca = PCA(n_components=n_comp, random_state=42)
    pca_fit = pca.fit(scaled_df)
    pc_vals = np.arange(pca.n_components_) + 1
    plt.plot(pc_vals, pca.explained_variance_ratio_, 'o-', linewidth=2, color='blue')
    plt.grid(True, which='both', linestyle='--', linewidth=0.5, color='gray', alpha=0.5)
    plt.title('Scree Plot del ACP sobre Intensidad Fase A de Aljibes')
    plt.xlabel('Numero de Componentes')
    plt.ylabel('Varianza Explicada')
    plt.show()
    cum_var_prop = pca_fit.explained_variance_ratio_.cumsum()
    fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(6, 4))
    ax.plot(np.arange(n_comp) + 1, cum_var_prop, marker='o')
    for x, y in zip(np.arange(n_comp) + 1, cum_var_prop):
        label = round(y, 2)
        ax.annotate(label, (x, y), textcoords="offset points", xytext=(0, 10), ha='center')
    ax.set_ylim(0, 1.1)
    ax.set_xticks(np.arange(pca_fit.n_components_) + 1)
    ax.grid(True, which='both', linestyle='--', linewidth=0.5, color='gray', alpha=0.5)
    ax.set_title('Porcentaje de Varianza Explicada Acumulada')
    ax.set_xlabel('Componente Principal')
```

```
    ax.set_ylabel('Varianza Acumulada')
    plt.show()

    correlation_matrix = scaled_df.corr().rename(index=lambda x: x + 1, columns=lambda x: x + 1)
    print("Matriz de correlacion:")
    print(correlation_matrix)
    print("\nVarianza acumulada:")
    print(cum_var_prop)
    print("\nVarianza explicada:")
    print(pca.explained_variance_)
    print("\nPorcentaje de varianza explicada:")
    print(pca_fit.explained_variance_ratio_)

    # Autovectores
    loadings = pca.components_.T * np.sqrt(pca.explained_variance_)
    loading_matrix = pd.DataFrame(loadings[:, 0:n_comp], columns = [f"PC{i+1}" for i in range(n_comp)], index = df.columns.tolist())
    print("\nMatriz de autovectores:")
    print(loading_matrix)

# Grafico de individuos en 2 dimensiones
def pca_indiv2D(dfi, mode, indic):
    group = ["Grupo_ind_1","Grupo_ind_2", "Grupo_ind_3"]
    df_group = dfi.iloc[:,33:36]
    if mode != 1 and mode != 2 and mode != 3:
        print("Selecciona modo: 1-> NO NORM  2-> NORM  3->IND")
        return
    if mode == 1:
        df = dfi.iloc[:,4:21] # DATOS NO NORMALIZADOS
    elif mode == 2:
        df = dfi.iloc[:,[21, 22, 23, 24, 8, 9, 25, 26, 27, 28, 29, 15, 16, 17, 18, 19, 20]] # DATOS NORMALIZADOS
    elif mode == 3:
        df = dfi.iloc[:,30:33] # INDICADORES

    scaler = StandardScaler()
    scaled_df = pd.DataFrame(scaler.fit_transform(df))
    pca = PCA(n_components=2, random_state=42)
    components = pca.fit_transform(scaled_df)
    total_var = pca.explained_variance_ratio_.sum() * 100

    mapping = {"Bien": "#006D2C", "Medio": "orange", "Mal": "#A50026"}
    fig = px.scatter(components, x=0, y=1, color = df_group[group[indic - 1]].astype('category'), color_discrete_map = mapping, title = f"ACP de 2D para Indicador {indic}")
    fig.update_layout(title={'text': f"ACP sobre Intensidad Fase A para Indicador {indic} (2D)", 'x': 0.5, 'xanchor': 'center'}, width=600, height=600,
        xaxis_title=f'PC1({100*pca.explained_variance_ratio_[0]:.2f}%)',
```

```
        yaxis_title=f'PC2({100*pca.explained_variance_ratio_[1]:.2f}%)', legend_title='Estado')
78  fig.add_annotation(text=f'Varianza Explicada Acumulada: {total_var:.2f}%', xref="paper",
        yref="paper", x=0.5, y=1.1, showarrow=False, font=dict( size=12 ), xanchor='center',
        yanchor='top')
79  fig.show()
80
81  # Grafico de individuos en 3 dimensiones
82  def pca_indiv3D(dfi, mode, indic):
83      group = ["Grupo_ind_1","Grupo_ind_2", "Grupo_ind_3"]
84      df_group = dfi.iloc[:,33:36]
85      if mode != 1 and mode != 2 and mode != 3:
86          print("Selecciona modo: 1-> NO NORM  2-> NORM  3->IND")
87          return
88      if mode == 1:
89          df = dfi.iloc[:,4:21] # DATOS NO NORMALIZADOS
90      elif mode == 2:
91          df = dfi.iloc[:,[21, 22, 23, 24, 8, 9, 25, 26, 27, 28, 29, 15, 16, 17, 18, 19, 20]] #
              DATOS NORMALIZADOS
92      elif mode == 3:
93          df = dfi.iloc[:,30:33] # INDICADORES
94
95      scaler = StandardScaler()
96      scaled_df = pd.DataFrame(scaler.fit_transform(df))
97      pca = PCA(n_components=3, random_state=42)
98      components = pca.fit_transform(scaled_df)
99      total_var = pca.explained_variance_ratio_.sum() * 100
100
101     mapping = {"Bien": "#006D2C", "Medio": "orange", "Mal": "#A50026"}
102     fig = px.scatter_3d(components, x=0, y=1, z=2, labels={'0': f'PC
          ({100*pca.explained_variance_ratio_[0]:.2f}%)', '1': f'PC
          2({100*pca.explained_variance_ratio_[1]:.2f}%)', '2': f'PC
          ({100*pca.explained_variance_ratio_[2]:.2f}%)'}, width=600, height=600,
          color=df_group[group[indic - 1]].astype('category'), color_discrete_map=mapping)
103     fig.add_annotation(text=f'Varianza Explicada Acumulada: {total_var:.2f}%', xref="paper",
          yref="paper", x=0.5, y=1.05, showarrow=False, font=dict( size=12 ), xanchor='center',
          yanchor='top')
104     fig.update_layout(title={'text': f"ACP sobre Intensidad Fase A para Indicador {indicador}
          (3D)", 'x': 0.5, 'xanchor': 'center'} )
105     fig.show()
106
107 # Grafico de cargas para las dos primeras dimensiones
108 def loadings(dfi, mode):
109     if mode != 1 and mode != 2 and mode != 3:
110         print("Selecciona modo: 1-> NO NORM  2-> NORM  3->IND")
111         return
112     if mode == 1:
113         df = dfi.iloc[:,4:21] # DATOS NO NORMALIZADOS
114     elif mode == 2:
```

```
115         df = dfi.iloc[:,[21, 22, 23, 24, 8, 9, 25, 26, 27, 28, 29, 15, 16, 17, 18, 19, 20]] #
                DATOS NORMALIZADOS
116     elif mode == 3:
117         df = dfi.iloc[:,30:33] # INDICADORES
118
119     scaler = StandardScaler()
120     scaled_df = pd.DataFrame(scaler.fit_transform(df))
121     pca = PCA(n_components=2, random_state=42)
122     components = pca.fit_transform(scaled_df)
123     loadings = pca.components_.T * np.sqrt(pca.explained_variance_)
124     fig = go.Figure(layout_yaxis_range=[-1.25, 1.25], layout_xaxis_range=[-1.25, 1.25])
125     for i, feature in enumerate(scaled_df.columns):
126         if (feature + 1) in [2, 11, 13, 16, 17]:
127             fig.add_shape(type='line', x0=0, y0=0, x1=loadings[i, 0], y1=loadings[i, 1],
                    line=dict(color="blue"))
128             fig.add_annotation(x=loadings[i, 0] + 0.025, y=loadings[i, 1], ax=0, ay=0,
                    xanchor="center", yanchor="bottom", text=df.columns.tolist()[feature],
                    font=dict(color="blue"))
129         else:
130             fig.add_shape(type='line', x0=0, y0=0, x1=loadings[i, 0], y1=loadings[i, 1],
                    line=dict(color="black"))
131             fig.add_annotation(x=loadings[i, 0] - 0.025, y=loadings[i, 1], ax=0, ay=0,
                    xanchor="center", yanchor="bottom", text=df.columns.tolist()[feature],
                    font=dict(color="black"))
132
133     fig.add_shape(type="circle", xref="x", yref="y", x0=-1, y0=-1, x1=1, y1=1,
            line_color="LightSeaGreen")
134     fig.add_trace(go.Scatter(x=[None], y=[None], mode='markers', marker=dict(size=10,
            color='blue'), legendgroup='Variable seleccionada', showlegend=True, name='Variables
            seleccionadas'))
135     fig.add_trace(go.Scatter(x=[None], y=[None], mode='markers', marker=dict(size=10,
            color='black'), legendgroup='Variable no seleccionada', showlegend=True,
            name='Variables no seleccionadas'))
136
137     fig.update_layout(width=700, height=700,
            xaxis_title=f'PC1({100*pca.explained_variance_ratio_[0]:.2f}%)',
            yaxis_title=f'PC2({100*pca.explained_variance_ratio_[1]:.2f}%)', title="Grafico de
            cargas para las dos primeras dimensiones del ACP Aljibes", title_x=0.5,
            legend=dict(yanchor="top", y=1, xanchor="left", x=0, font=dict(size=12,
            color="black")), font=dict(color='black'), xaxis=dict(title_font=dict(color='black'),
            tickfont=dict(color='black')), axis=dict(title_font=dict(color='black'),
            tickfont=dict(color='black')))
138
139     define_pca(dfi = intensidadA_alj, n_comp = 20, mode = 2)
140     pca_indiv2D(dfi = intensidadA_alj, mode = 2)
141     pca_indiv3D(dfi = intensidadA_alj, mode = 2)
142     loadings(dfi = intensidadA_alj, mode = 2)
```

Listing B.9: Análisis de Componentes Principales y sus gráficos

B.2.5. Clustering

```
inertia = []
silhouette_scores = []
for i in range(2, 11):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(intensidadA_alj)
    inertia.append(kmeans.inertia_)
    score = silhouette_score(intensidadA_alj, kmeans.labels_)
    silhouette_scores.append(score)

# Grafica el metodo del codo
fig, axs = plt.subplots(1, 2, figsize=(12, 6))
axs[0].plot(range(2, 11), inertia, marker='o')
axs[0].set_title('Metodo del Codo con KMeans')
axs[0].set_xlabel('Numero de clusters')
axs[0].set_ylabel('Inercia')
axs[0].grid(True, which='both', linestyle='--', linewidth=0.5, color='gray', alpha=0.5)

# Grafica el metodo de la silueta
axs[1].plot(range(2, 11), silhouette_scores, marker='o')
axs[1].set_title('Metodo de la Silueta con KMeans')
axs[1].set_xlabel('Numero de clusters')
axs[1].set_ylabel('Coeficiente de Silueta')
axs[1].grid(True, which='both', linestyle='--', linewidth=0.5, color='gray', alpha=0.5)
plt.tight_layout()
plt.show()
```

Listing B.10: Método del codo y de la silueta

```
def kmeans_clustering(X, k, method, ind):
    norm = StandardScaler()
    X_norm = pd.DataFrame(norm.fit_transform(X), columns=X.columns.tolist())
    cluster = KMeans(n_clusters=k, init=method, random_state=42)
    cluster.fit(X_norm)
    y_pred = cluster.labels_
    fig, axes = plt.subplots(1, ind.shape[1], figsize=(18, 6))
    for i, indic in enumerate(ind.columns):
        plt.suptitle("Clustering KMeans inicializacion " + method + " con " + str(k) + "
            clusteres", fontsize=14)
        axes[i].scatter(np.arange(1, X_norm.shape[0] + 1), ind[indic], c=y_pred, s=50,
            cmap="viridis")
        axes[i].set_title("Sobre " + str(indic))
        axes[i].set_xlabel('Num obs')
        axes[i].set_ylabel(indic)
    plt.tight_layout()
    plt.show()

```

```
for mtd in ["k-means++", "random"]:
    kmeans_clustering(intensidadA_alj, 3, mtd, intensidadA_alj.iloc[:, 27:30])
```

Listing B.11: KMeans

```
def kmedoids_clustering(X, k, method, ind):
    norm = StandardScaler()
    X_norm = pd.DataFrame(norm.fit_transform(X), columns=X.columns.tolist())
    cluster = KMedoids(n_clusters=k, init=method)
    cluster.fit(X_norm)
    y_pred = cluster.labels_
    fig, axes = plt.subplots(1, 3, figsize=(18, 6))
    for i, indic in enumerate(ind.columns):
        plt.suptitle("Clustering KMedoids inicializacion " + method + " con " + str(k) + "
            clusteres", fontsize=14)
        axes[i].scatter(np.arange(1, X_norm.shape[0] + 1), ind[indicador], c=y_pred, s=50,
            cmap="viridis")
        axes[i].set_title("Sobre " + str(indic))
        axes[i].set_xlabel('Num obs')
        axes[i].set_ylabel(indic)
    plt.tight_layout()
    plt.show()

for mtd in ["random", "heuristic", "k-medoids++", "build"]:
    kmedoids_clustering(intensidadA_alj, 3, mtd, intensidadA_alj.iloc[:, 27:30])
```

Listing B.12: KMedoids

```
def agglomerative_clustering(X, k, method, ind, num):
    norm = StandardScaler()
    X_norm = pd.DataFrame(norm.fit_transform(X), columns = X.columns.tolist())
    cluster = AgglomerativeClustering(n_clusters=k, metric='euclidean', linkage=method)
    cluster.fit(X_norm)
    y_pred = cluster.labels_
    num_plots = ind.shape[1]
    fig, axes = plt.subplots(1, num_plots, figsize=(18, 6))
    mt = ["Single", "Complete", "Average", "Ward"]
    for i, indic in enumerate(ind.columns):
        plt.suptitle("Clustering Aglomerativo metodo " + mt[num] + " con " + str(k) + "
            clusteres", fontsize=14)
        axes[i].scatter(np.arange(1, X_norm.shape[0] + 1), ind[indicador], c=y_pred, s=50,
            cmap="viridis")
        axes[i].set_title("Sobre " + str(indic))
        axes[i].set_xlabel('Num obs')
        axes[i].set_ylabel(indic)
    plt.tight_layout()
```

```
    plt.show()

for i, mtd in enumerate(["single", "complete", "average", "ward"]):
    agglomerative_clustering(intensidadA_alj, 3, mtd, intensidadA_alj.iloc[:, 27:30], i)
```

Listing B.13: Clustering Aglomerativo

```
def dbscan_clustering(X, k, ind):
    eps_list = [i / 10 for i in range(3, 11)]
    min_samples_list = [i for i in range(1,11)]
    norm = StandardScaler()
    X_norm = pd.DataFrame(norm.fit_transform(X), columns=X.columns.tolist())
    for eps in eps_list:
        for min_samples in min_samples_list:
            clustering = DBSCAN(eps = eps, min_samples = min_samples).fit_predict(X_norm)
            y_pred = clustering
            num_clusters = len(set(clustering)) - (1 if -1 in clustering else 0)
            if (k == num_clusters):
                X_new = X_norm.iloc[y_pred != -1]
                dbscan_labels = y_pred[y_pred != -1]
                fig, axes = plt.subplots(1, 3, figsize=(18, 6))
                for i, indic in enumerate(ind.columns):
                    plt.suptitle("Clustering DBSCAN con clusteres = " + str(k) + ", eps = " +
                        str(eps) + ", min_samples = " + str(min_samples), fontsize=14)
                    indic = ind[indicador]
                    indic = indic[clustering != -1]
                    axes[i].scatter(np.arange(1, X_new.shape[0] + 1), indic, c=dbscan_labels,
                        s=50, cmap="viridis")
                    axes[i].set_title("Sobre " + str(indic))
                    axes[i].set_xlabel('Num obs')
                    axes[i].set_ylabel(indic)
                plt.tight_layout()
                plt.show()

dbscan_clustering(intensidadA_alj, 3, intensidadA_alj.iloc[:, 27:30])
```

Listing B.14: DBSCAN

B.3. Resultados

B.3.1. Entrenamiento-Validación

En esta sección se tomará como base el algoritmo del LDA y como conjunto de datos la intensidad fase A de Aljibes.

```
df = intensidadA_alj
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
scaler = MinMaxScaler()
X_sc = pd.DataFrame(scaler.fit_transform(X), columns=X.columns.tolist())
X_train, X_test, y_train, y_test = train_test_split(X_sc, y, test_size = 1/3, stratify=y)
classifier = LinearDiscriminantAnalysis()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
print("Tasa acierto: ", accuracy_score(y_test, y_pred))
confusion_matrix(y_test, y_pred)
```

Listing B.15: Hold out para LDA

```
def cross_validation(X_no_esc, y, cv):
    kf = StratifiedKFold(n_splits=cv, shuffle=True)
    fold_results = []
    test_error_sum = 0
    scaler = MinMaxScaler()
    X = pd.DataFrame(scaler.fit_transform(X_no_esc), columns=X_no_esc.columns.tolist())

    for train_index, test_index in kf.split(X, y):
        model = LinearDiscriminantAnalysis()
        X_train, y_train = X.iloc[train_index,:] , y[train_index]
        X_test, y_test = X.iloc[test_index,:] , y[test_index]
        model.fit(X_train, y_train)
        y_train_pred = model.predict(X_train)
        y_test_pred = model.predict(X_test)
        fold_result = {
           "Train Accuracy": accuracy_score(y_train, y_train_pred),
           "Test Accuracy": accuracy_score(y_test, y_test_pred),
           "Train Precision": precision_score(y_train, y_train_pred, average = "macro"),
           "Test Precision": precision_score(y_test, y_test_pred, average = "macro"),
           "Train Recall": recall_score(y_train, y_train_pred, average = "macro"),
           "Test Recall": recall_score(y_test, y_test_pred, average = "macro"),
           "Train F1": f1_score(y_train, y_train_pred, average = "macro"),
           "Test F1": f1_score(y_test, y_test_pred, average = "macro"),
           "Train Error": 1 - accuracy_score(y_train, y_train_pred),
           "Test Error": 1 - accuracy_score(y_test, y_test_pred)
        }
        test_error_sum += fold_result["Test Error"]
        fold_results.append(fold_result)
    test_error_average = test_error_sum / cv
    return fold_results, test_error_average
```

Listing B.16: Validación Cruzada para LDA

```
def loo(X_no_esc, y):
    loo = LeaveOneOut()
    fold_results = []
    test_error_sum = 0
    total_iterations = len(X_no_esc)
    scaler = MinMaxScaler()
    X = pd.DataFrame(scaler.fit_transform(X_no_esc), columns=X_no_esc.columns.tolist()) #
        escalamos los datos
    with tqdm(total=total_iterations, desc= "LOO Progress") as pbar:
        for train_index, test_index in loo.split(X):
            model = LinearDiscriminantAnalysis()
            X_train, y_train = X.iloc[train_index,:] , y[train_index]
            X_test, y_test = X.iloc[test_index,:] , y[test_index]
            model.fit(X_train, y_train)
            y_train_pred = model.predict(X_train)
            y_test_pred = model.predict(X_test)
            fold_result = {
               "Train Accuracy": accuracy_score(y_train, y_train_pred),
               "Test Accuracy": accuracy_score(y_test, y_test_pred),
               "Train Precision": precision_score(y_train, y_train_pred, average = None),
               "Test Precision": precision_score(y_test, y_test_pred, average = None),
               "Train Recall": recall_score(y_train, y_train_pred, average = None),
               "Test Recall": recall_score(y_test, y_test_pred, average = None),
               "Train F1": f1_score(y_train, y_train_pred, average = None),
               "Test F1": f1_score(y_test, y_test_pred, average = None),
               "Train Error": 1 - accuracy_score(y_train, y_train_pred),
               "Test Error": 1 - accuracy_score(y_test, y_test_pred)
            }
            test_error_sum += fold_result["Test Error"]
            fold_results.append(fold_result)
            pbar.update(1)
    test_error_average = test_error_sum/len(X)
    return fold_results, test_error_average
```

Listing B.17: Leave One Out para LDA

```
# Grafico de lineas para el accuracy (analogo para el resto de metricas)
def cv_accuracy(results):
    train_acc = [fold["Train Accuracy"] for fold in results]
    test_acc = [fold["Test Accuracy"] for fold in results]
    plt.figure(figsize = (10, 6))
    plt.plot(range(1, len(train_acc)+ 1), train_acc,"ro-", label = "Train Accuracy")
    plt.plot(range(1, len(test_acc)+ 1), test_acc,"bo-", label = "Test Accuracy")
    plt.xlabel("Fold")
    plt.ylabel("Accuracy")
    plt.title("Resultados de VC")
    plt.legend()
```

```
    plt.show()

# Grafico de barras para el accuracy (analogo para el resto de metricas)
def cv_accuracy_2(results):
    train_acc = [fold["Train Accuracy"] for fold in results]
    test_acc = [fold["Test Accuracy"] for fold in results]
    fold_labels = [f"Fold {i + 1}"for i in range(len(train_acc))]
    data = pd.DataFrame({"Fold": fold_labels,"Train Accuracy": train_acc,"Test Accuracy":
        test_acc })
    data = data.melt("Fold", var_name = "Dataset", value_name = "Accuracy")
    plt.figure(figsize = (10, 6))
    sns.barplot(x = "Fold", y = "Accuracy", hue = "Dataset", data = data)
    plt.xlabel("Fold")
    plt.ylabel("Accuracy")
    plt.title("Resultados de VC")
    plt.legend()
    plt.show()
```

Listing B.18: Visualización de los resultados en la validación cruzada

B.3.2. Análisis Discriminante Lineal

```
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(14, 11))
title = ["Train: Intensidad Fase A - Test: Intensidad Fase B", "Train: Intensidad Fase A -
    Test: Intensidad Fase C", "Train: Intensidad Fase B - Test: Intensidad Fase C", "Train:
    Tension Fase A - Test: Tension Fase B", "Train: Tension Fase A - Test: Tension Fase C",
    "Train: Tension Fase B - Test: Tension Fase C", "Train: Intensidad Fase A - Test: Tension
    Fase A", "Train: Intensidad Fase B - Test: Tension Fase B", "Train: Intensidad Fase C -
    Test: Tension Fase C"]
data = [[intensidadA_alj, intensidadB_alj], [intensidadA_alj, intensidadC_alj],
    [intensidadB_alj, intensidadC_alj], [tensionA_alj, tensionB_alj], [tensionA_alj,
    tensionC_alj], [tensionB_alj, tensionC_alj], [intensidadA_alj, tensionA_alj],
    [intensidadB_alj, tensionB_alj], [intensidadC_alj, tensionC_alj]]
for i, ax in enumerate(axes.flatten()):
    clas = LinearDiscriminantAnalysis()
    scaler = MinMaxScaler()
    X_train = pd.DataFrame(scaler.fit_transform(data[i][0].iloc[:,[1:18]]))
    y_train = intensidadA_alj.iloc[:, 31]
    clas.fit(X_train, y_train)
    y_test = intensidadA_alj.iloc[:, 31]
    x_test = pd.DataFrame(scaler.transform(data[i][1].iloc[:,[1:18]]))
    y_pre = clas.predict(x_test)
    conf_matrix = confusion_matrix(y_test, y_pre)
    disp = ConfusionMatrixDisplay(confusion_matrix=conf_matrix, display_labels=["Bien",
        "Medio", "Mal"])
    disp.plot(cmap=plt.cm.Reds, ax=ax)
    ax.set_title(title[i])
plt.suptitle('Matrices de Confusion LDA (con todas las variables)', fontsize=16)
```

```
plt.tight_layout()
plt.show()
```

Listing B.19: Comparación de matrices de confusión

```
df = intensidadA_alj
predict_df = intensidadA_ped1V
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

scaler = StandardScaler()
X_sc = pd.DataFrame(scaler.fit_transform(X), columns=X.columns.tolist())
predict_df_sc = pd.DataFrame(scaler.transform(predict_df),
    columns=predict_df.columns.tolist())

classifier = LinearDiscriminantAnalysis()
classifier.fit(X_sc, y)

y_pred = classifier.predict(predict_df_sc)
```

Listing B.20: Entrenamiento y predicción con LDA

B.3.3. Árbol de decisión

```
def hyp_srch_dectree(X_train_ho, y_train_ho, cv):
    param_dist = {
        "criterion": ["gini", "entropy"],
        "max_depth": [2, 3, 4],
        "min_samples_split": [2, 3, 4, 5, 10],
        "min_samples_leaf": [1, 2, 3, 4, 5, 10, 15, 20]
    }
    classifier = DecisionTreeClassifier(random_state=20)
    kf = StratifiedKFold(n_splits=cv, shuffle=True, random_state=42)
    random_search = RandomizedSearchCV(
        estimator=classifier,
        param_distributions=param_dist,
        n_iter=20,
        cv=kf,
        scoring='accuracy',
        random_state=42,
        n_jobs=-1
    )
    with tqdm(total=1, desc="Buscando hiperparametros optimos") as pbar:
        random_search.fit(X_train_ho, y_train_ho)
        pbar.update(1)

```

```
    best_model = random_search.best_params_
    best_score = random_search.best_score_
    best_estimator = random_search.best_estimator_

    return best_model, best_score, best_estimator

def hyp_val_dectree(X_train, y_train, X_test, y_test, criterion, depth, split, leaf):
    classifier = DecisionTreeClassifier(criterion=criterion, max_depth=depth,
        min_samples_split=split, min_samples_leaf=leaf, random_state=20)
    classifier.fit(X_train, y_train)

    y_train_pred = classifier.predict(X_train)
    y_test_pred = classifier.predict(X_test)
    fold_result = {"Train Accuracy": accuracy_score(y_train, y_train_pred),
                "Test Accuracy": accuracy_score(y_test, y_test_pred),
                "Train Precision": precision_score(y_train, y_train_pred, average = "macro"),
                "Test Precision": precision_score(y_test, y_test_pred, average = "macro"),
                "Train Recall": recall_score(y_train, y_train_pred, average = "macro"),
                "Test Recall": recall_score(y_test, y_test_pred, average = "macro"),
                "Train F1": f1_score(y_train, y_train_pred, average = "macro"),
                "Test F1": f1_score(y_test, y_test_pred, average = "macro"),
                "Train Error": 1 - accuracy_score(y_train, y_train_pred),
                    "Test Error": 1 - accuracy_score(y_test, y_test_pred)
    }

    return fold_result, classifier

df = intensidadA_alj
X = df.iloc[:, :-1]
y = pd.DataFrame(df.iloc[:, -1])

scaler = MinMaxScaler()
X_sc = pd.DataFrame(scaler.fit_transform(X), columns=X.columns.tolist())

X_train, X_test, y_train, y_test = train_test_split(X_sc, y, test_size = 1/3, stratify = y,
    random_state = 42)
best_model, best_score, best_clasif = hyp_srch_dectree(X_train, y_train, 10)
print("Mejor clasificador:" , best_clasif)
print("Hiperparametros del mejor clasificador: ", best_model)
print("Accuracy del mejor clasificador: ", best_score)
results, clasif = hyp_val_dectree(X_train, y_train, X_test, y_test,
    criterio=best_model["criterion"], depth=best_model["max_depth"],
    split=best_model["min_samples_split"], leaf=best_model["min_samples_leaf"])
print(results)
print(clasif)
```

Listing B.21: Búsqueda de hiperparámetros para un árbol de decisión

```
df = intensidadA_alj
predict_df = intensidadA_ped1V
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

scaler = StandardScaler()
X_sc = pd.DataFrame(scaler.fit_transform(X), columns=X.columns.tolist())
predict_df_sc = pd.DataFrame(scaler.transform(predict_df),
    columns=predict_df.columns.tolist())

classifier = DecisionTreeClassifier(criterion = "gini", max_depth= 2, min_samples_split= 10,
    min_samples_leaf=5, random_state=250)
classifier.fit(X_sc, y)

y_pred = classifier.predict(predict_df_sc)

# Graficar el arbol de decision
fig = plt.figure(figsize = (8, 6))
_ = tree.plot_tree(clf, feature_names = X_sc.columns.tolist(), class_names =["Bien", "Medio",
    "Mal"], filled = True, rounded = True, impurity = True, proportion = False, precision = 2)
```

Listing B.22: Entrenamiento y predicción con árbol de decisión

B.3.4. Random Forest

```
def hyp_srch_RF(X_train_ho, y_train_ho, cv, n_iter):
    param_dist = {
        'criterion': ["gini", "log_loss"],
        'n_estimators': randint(10, 150),
        'max_depth': 1, 2, 3, 5, 7, 10]
    }
    rf = RandomForestClassifier(random_state=42)
    kf = StratifiedKFold(n_splits=cv, shuffle=True, random_state=42)
    random_search = RandomizedSearchCV(estimator=rf,
                                       param_distributions=param_dist,
                                       n_iter=n_iter,
                                       cv=kf,
                                       scoring='accuracy',
                                       random_state=42,
                                       n_jobs=-1)
    with tqdm(total=n_iter, desc="Buscando hiperparametros optimos (RF)") as pbar:
        random_search.fit(X_train_ho, y_train_ho)
        pbar.update(n_iter)

    best_model = random_search.best_params_
    test_error_average = 1 - random_search.best_score_
    error_model = 1 - random_search.cv_results_['mean_test_score']
```

```
    best_estimator = random_search.best_estimator_

    return best_model, test_error_average, error_model.tolist(), best_estimator

def hyp_val_RF(X_train, y_train, X_test, y_test, criterion, nest, depth):
    classifier = RandomForestClassifier(criterion=criterion, n_estimators=nest,
        max_depth=depth, random_state=42)
    classifier.fit(X_train, y_train)
    y_train_pred = classifier.predict(X_train)
    y_test_pred = classifier.predict(X_test)
    fold_result = {"Train Accuracy": accuracy_score(y_train, y_train_pred),
                   "Test Accuracy": accuracy_score(y_test, y_test_pred),
                   "Train Precision": precision_score(y_train, y_train_pred, average =
                       "macro"),
                   "Test Precision": precision_score(y_test, y_test_pred, average = "macro"),
                   "Train Recall": recall_score(y_train, y_train_pred, average = "macro"),
                   "Test Recall": recall_score(y_test, y_test_pred, average = "macro"),
                   "Train F1": f1_score(y_train, y_train_pred, average = "macro"),
                   "Test F1": f1_score(y_test, y_test_pred, average = "macro"),
                   "Train Error": 1 - accuracy_score(y_train, y_train_pred),
                   "Test Error": 1 - accuracy_score(y_test, y_test_pred)
    }
    return fold_result, classifier

df = intensidadA_alj
X = df.iloc[:, :-1]
y = pd.DataFrame(df.iloc[:, -1])

scaler = MinMaxScaler()
X_sc = pd.DataFrame(scaler.fit_transform(X), columns=X.columns.tolist())
X_sc.columns, df.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X_sc, y, test_size = 1/3, stratify=y,
    random_state=42)
clasif, error, models, best_clasif = hyp_srch_RF(X_train, y_train, 10, 100)
print("Error del mejor clasificador:" , error)
print("Hiperparametros del mejor clasificador: ", clasif)
print("Mejor clasificador: ", best_clasif)
print("Errores de los modelos: ", models)
results, clas = hyp_val_RF(X_train, y_train, X_test, y_test, criterion=clasif["criterion"],
    nest=clasif["n_estimators"], depth=clasif["max_depth"])
print(results)
print(clas)
```

Listing B.23: Búsqueda de hiperparámetros para Random Forest

```
importances = pd.DataFrame(columns = ["Var", "Importance"])
importances["Var"] = clas.feature_names_in_
```

```
importances["Importance"] = clas.feature_importances_
var_candidates = importances.sort_values(by="Importance")["Var"]
importances_sorted = importances.sort_values(by="Importance")

plt.figure(figsize=(10, 6))
plt.barh(importances_sorted["Var"], importances_sorted["Importance"], color='skyblue')
plt.xlabel("Importancia Relativa")
plt.ylabel('Variables')
plt.title('Importancia de las variables en Ranfom Forest de Aljibes')
plt.show()
```

Listing B.24: Importancia de las variables en Random Forest

```
df = intensidadA_alj
predict_df = intensidadA_ped1V
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

scaler = StandardScaler()
X_sc = pd.DataFrame(scaler.fit_transform(X), columns=X.columns.tolist())
predict_df_sc = pd.DataFrame(scaler.transform(predict_df),
    columns=predict_df.columns.tolist())

classifier = RandomForestClassifier(criterion='gini', max_depth=5, n_estimators=102,
    random_state=42)
classifier.fit(X_sc, y)

y_pred = classifier.predict(predict_df_sc)
```

Listing B.25: Entrenamiento y predicción con Random Forest

B.3.5. Boosting: XGBoost

```
def hyp_srch_XGB(df, rep, cv):
    X = df.iloc[:, :-1]
    y_cat = pd.DataFrame(df.iloc[:, -1])
    label_encoder = LabelEncoder()
    y = label_encoder.fit_transform(y_cat)
    scaler = MinMaxScaler()
    X_sc = pd.DataFrame(scaler.fit_transform(X), columns=X.columns.tolist())
    alg = xgb.XGBClassifier()
    param_dist = {"learning_rate": np.arange(0.01, 0.51, 0.1),
            "n_estimators": [1, 10, 50, 100, 200, 1000],
            "min_child_weight": np.arange(1, 10, 2),
            "max_depth":[2, 3, 5, 7, 10],
            "gamma": [i/10.0 for i in range(0,5)],
```

```
            "subsample": [i/10.0 for i in range(4,9)],
            "colsample_bytree": [i/10.0 for i in range(4,9)],
            "reg_alpha": [0, 0.1, 0.5, 1, 2, 5],
            "reg_lambda": [0, 0.1, 0.5, 1, 2, 5],
            "objective": ["multi:softmax"],
            "num_class": [3],
            "use_label_encoder": [False]
            }
    error = []
    final_conf_matrix = np.zeros([3,3], dtype=int)

    for r in range(0, rep):
        print(f"Repeticion {r}")
        X_train, X_test, y_train, y_test = train_test_split(X_sc, y, stratify=y, test_size=1/3)
        clf = RandomizedSearchCV(alg, param_dist, cv=cv, n_iter=100, n_jobs=10, verbose=0,
            random_state=42)
        clf.fit(X_train, y_train, eval_set=[(X_train, y_train)], eval_metric="mlogloss",
            early_stopping_rounds=10, verbose=False)

        print(f"Mejores hiperparametros: ", clf.best_params_)
        y_pred = clf.predict(X_test)
        print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
        error.append(1-accuracy_score(y_test, y_pred))
        conf_matrix = confusion_matrix(y_test, y_pred)
        final_conf_matrix = final_conf_matrix + conf_matrix

    results = pd.DataFrame({"Error": error})
    print(final_conf_matrix)
    print(f"Accuracy medio: {1 - results['Error'].mean()}")
    print(f"Error medio: {results['Error'].mean()}")
    return clf.best_estimator_, clf.get_params
```

Listing B.26: Búsqueda de hiperparámetros para XGBoost

```
importances = pd.DataFrame(columns = ["Var", "Importance"])
importances["Var"] = clasificador.feature_names_in_
importances["Importance"] = clasificador.feature_importances_
var_candidates = importances.sort_values(by="Importance")["Var"]
importances_sorted = importances.sort_values(by="Importance")

plt.figure(figsize=(10, 6))
plt.barh(importances_sorted["Var"], importances_sorted["Importance"], color='skyblue')
plt.xlabel('Importancia Relativa')
plt.ylabel('Variables')
plt.title('Importancia de las variables en XGBoost de Aljibes')
plt.show()
```

Listing B.27: Importancia de las variables en XGBoost

```
df = intensidadA_alj
predict_df = intensidadA_ped1V
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

clasif_hyp, params = hyp_srch_XGB(df, 5, 5)

scaler = StandardScaler()
X_sc = pd.DataFrame(scaler.fit_transform(X), columns=X.columns.tolist())
predict_df_sc = pd.DataFrame(scaler.transform(predict_df),
    columns=predict_df.columns.tolist())

classifier = xgb.XGBClassifier(clasif_hyp.get_params())
classifier.fit(X_sc, y)

y_pred = classifier.predict(predict_df_sc)
```

Listing B.28: Entrenamiento y predicción con XGBoost

B.3.6. Visualización de las Predicciones

```
categories = ["Mal", "Medio", "Bien"]
colors = [(165/255, 0/255, 38/255), "orange", (0/255, 109/255, 44/255)]
labels = ['Mal', 'Medio', 'Bien']

values = [
    (pd.DataFrame({"Estado": y_pred})["Estado"] == "Mal").sum(),
    (pd.DataFrame({"Estado": y_pred})["Estado"] == "Medio").sum(),
    (pd.DataFrame({"Estado": y_pred})["Estado"] == "Bien").sum()
]
numb_state = {'Bien': 2, 'Medio': 1, 'Mal': 0}
numbers = [numb_state[cat] for cat in y_pred]
data = np.array(numbers).reshape(-1, 1)

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(8, 5), gridspec_kw={'height_ratios': [4, 1]})
ax1.pie(values, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)
ax1.set_title('Predicciones del LDA para Pedrizas1V')
sns.heatmap(data.T, cmap=colores, cbar=False, ax=ax2, yticklabels=['Estado'])
ax2.set_xlabel('Observacion')
ax2.set_title('Evolucion del Estado')
plt.tight_layout()
fig.legend(labels=etiquetas, loc='lower right', title='Estado', bbox_to_anchor=(1, 0.5))
plt.show()
```

Listing B.29: Visualización de las predicciones sobre otros pozos para todos los clasificadores