UNIVERSIDAD DE VALLADOLID
Escuela Técnica Superior de Ingenieros de Telecomunicación

Universidad de Valladolid

Aplicación de algoritmos de Machine Learning no supervisados para el etiquetado automático de medidas de vibraciones en fachadas

TRABAJO DE FIN DE GRADO

Grado en Ingeniería de Tecnologías de Telecomunicación

Autora:
Elvira Blanco de Lapuerta

Tutora:
Lara del Val Puente

Valladolid, 2025

TÍTULO:	**Aplicación de algoritmos de Machine Learning no supervisados para el etiquetado automático de medidas de vibraciones en fachadas**
AUTORA:	**Elvira Blanco de Lapuerta**
TUTORA:	**Lara del Val Puente**
DEPARTAMENTO:	**Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE:	Alberto Izquierdo Fuente
SECRETARIO:	Lara del Val Puente
VOCAL:	Juan José Villacorta Calvo
SUPLENTE:	Ramón de la Rosa Steinz
SUPLENTE:	Alonso Alonso Alonso

FECHA:
CALIFICACIÓN:

Índice de figuras

Resumen del proyecto

Este trabajo aplica algoritmos de Machine Learning no supervisado (K-Means y DBSCAN) para el etiquetado automático de vibraciones recogidas durante el desmontaje de los mosaicos del Centro SCOP en México. Los datos, capturados con sensores de aceleración, contenían patrones previamente identificados (A-H). Tras un preprocesamiento cuidadoso (normalización, balanceo, PCA), los resultados evaluados con métricas ARI y NMI mostraron un rendimiento cuestionable de ambos algoritmos para replicar la clasificación previa existente. Se concluye y se proponen líneas de futuro para mejorar la identificación automática en monitorización estructural y conservación patrimonial.

Palabras clave

Machine Learning no supervisado, Clústering, Vibraciones estructurales, K-Means, DBSCAN, Conservación patrimonial, Patrones.

Abstract

This work applies unsupervised Machine Learning algorithms (K-Means and DBSCAN) for the automatic labeling of vibrations recorded during the dismantling of the mosaics at the SCOP Center in Mexico. The data, captured using accelerometer sensors, contained pre-identified patterns (A-H). After exhaustive preprocessing (normalization, balancing, PCA), the results evaluated using ARI and NMI metrics showed limited performance of both algorithms in replicating the manual classification. It concludes and proposes future lines of work to improve automatic identification in structural health monitoring and heritage conservation.

Keywords

Unsupervised Machine Learning, Clustering, Structural Vibrations, K-Means, DBSCAN, Heritage Conservation, Patterns.

Índice

Introducción

0.1 Estado del Arte

En los últimos años, el sector de la conservación del patrimonio ha empezado a usar nuevas tecnologías para proteger edificios y elementos arquitectónicos de gran valor histórico. Una de las áreas más importantes en ingeniería civil es la salud estructural de los edificios (SHM, structural health monitoring) (Kareem Eltouny, 2023). Esta se ocupa de observar y evaluar el estado de las estructuras para detectar daños o problemas que puedan afectar su seguridad. Un ejemplo claro de este tipo de trabajo se llevó a cabo en el conjunto de edificios conocido como centro SCOP, en Ciudad de México. Para cuidar los mosaicos de sus fachadas, antes del derribo se colocaron sensores acelerómetros en los murales. Con ellos se midieron las vibraciones que aparecieron durante las diferentes etapas de la operación.

El objetivo de este estudio es aplicar algoritmos de Machine Learning no supervisado para clasificar automáticamente las medidas de vibración obtenidas por los sensores. Se cuenta con datos de seis sensores, cada uno con registros en tres ejes (X, Y y Z). Estos sensores fueron ubicados en las fachadas, según la posición de los mosaicos y los momentos clave del derribo. Las maniobras generaron vibraciones en distintas frecuencias que necesitan ser analizadas para reconocer patrones importantes y posibles riesgos.

El uso de Machine Learning no supervisado resulta útil porque evita el trabajo manual de etiquetar grandes cantidades de datos. En este enfoque, el modelo trabaja con datos sin etiquetas y busca descubrir patrones o estructuras ocultas en los datos. El objetivo final es poder identificar y clasificar de manera autónoma los diferentes tipos de vibraciones que ocurren durante el proceso de derribo, lo cual puede contribuir a la prevención de daños en los mosaicos y asegurar su conservación.

En el ámbito del análisis de vibraciones para la monitorización estructural, (Kareem Eltouny, 2023) resalta el interés creciente en métodos no supervisados para detectar daños y anomalías en estructuras civiles. Estos métodos son especialmente útiles cuando obtener datos es difícil o muy caro, como en el caso de edificios históricos que necesitan ser demolidos o reconstruidos. En su revisión se presentan técnicas como el análisis de componentes principales (PCA), los modelos de mezcla gaussianos (GMM) y las redes neuronales, mostrando que son eficaces para reconocer cambios y patrones en los datos de vibración.

Este recurso respalda la aplicación de algoritmos no supervisados en nuestro trabajo, ya que los sensores del centro SCOP producen datos multidimensionales (aceleraciones en X, Y y Z). Estos datos pueden, junto con técnicas como el clústering o la reducción de dimensionalidad, detectar eventos clave durante el derribo. Además, el artículo destaca la importancia de la resistencia de los modelos frente a cambios en el ambiente o las condiciones de trabajo. Esto resulta esencial porque las vibraciones en los murales provienen de varias fuentes, como maquinaria o impactos. Incluir estos avances en el proyecto no solo mejoraría el etiquetado automático de datos, sino que también daría un procedimiento metodológico sólido para futuras intervenciones en patrimonio cultural.

En este estudio también se parte de documentos importantes que recogen información sobre los eventos del derribo, la ubicación de los sensores, los patrones de vibración y los informes de las maniobras. Además, el grupo de investigación GPA, desarrolló un software en LabVIEW que permite procesar y analizar las señales de los acelerómetros, tanto en el tiempo como en la frecuencia. Esta herramienta es necesaria para extraer patrones relevantes.

De esta forma, se busca no solo aportar al uso de Machine Learning en el análisis de vibraciones, sino también ofrecer un recurso útil para la conservación del patrimonio cultural. Demostrar cómo la tecnología puede ayudar a proteger elementos históricos mediante técnicas innovadoras.

0.2 Objetivos

0.2.1 Objetivos generales

El presente trabajo tiene como objetivo desarrollar un sistema que, aplicando algoritmos de Machine Learning no supervisados permita realizar un etiquetado automático de las medidas de vibración registradas en las fachadas de los mosaicos del centro SCOP. Con ello se busca identificar de manera autónoma los distintos patrones presentes en las señales, diferenciando entre vibraciones significativas y ruido, con el fin de facilitar el análisis estructural y contribuir a la conservación del patrimonio arquitectónico.

0.2.2 Objetivos específicos

Con el fin de alcanzar el objetivo general, se plantean los siguientes objetivos específicos:

- Analizar en profundidad los datos de vibración de los acelerómetros. El propósito es encontrar y describir patrones que se repitan, tanto en el tiempo como en la frecuencia.

- Construir una base de datos organizada y etiquetada a partir de los patrones reconocidos manualmente, que sirva como referencia para el entrenamiento y la evaluación de los modelos de aprendizaje automático.

- Diseñar un proceso para preprocesar y normalizar los datos. Esto garantizará que sean consistentes, quitando valores conflictivos o nulos y preparando para que los algoritmos de Machine Learning puedan usarlos correctamente.

- Aplicar y comparar distintos algoritmos de aprendizaje no supervisado, en particular métodos de clústering como K-Means y DBSCAN, evaluando su capacidad para clasificar los datos contrastando con patrones definidos previamente.

- Implementar el uso de técnicas de reducción de dimensionalidad (PCA) con el fin de facilitar la visualización la robustez y analizando si los resultados obtenidos mejoran.

- Contrastar los resultados de los modelos con la clasificación manual y con los informes de maniobra disponibles, verificando la correspondencia entre los clústeres generados y los eventos registrados en las obras. Evaluar la aplicabilidad de la metodología a futuros escenarios, proponiendo posibles líneas de mejora.

0.3 Organización de la memoria

La presente memoria se organiza de manera que el lector pueda comprender tanto el contexto del proyecto como el desarrollo metodológico seguido. En primer lugar, se incluye el estado del arte, donde se describe el caso de estudio de los mosaicos del centro SCOP y el proceso de conservación en el que se sitúa la instalación de los sensores, así como la importancia del análisis de vibraciones en este tipo de intervenciones. Después, se presenta el marco teórico, en el que se introducen los fundamentos del aprendizaje automático, con especial énfasis en las técnicas no supervisadas y en los algoritmos utilizados en este trabajo.

Posteriormente, se dedica un capítulo al análisis de los datos disponibles, en el que se caracterizan los distintos patrones de vibración identificados y se detalla el proceso de preparación de los conjuntos de datos. Luego se expone la aplicación de los algoritmos de clústering elegidos, explicando la metodología, las pruebas realizadas y los resultados obtenidos. Finalmente, se presentan las conclusiones y posibles líneas futuras de investigación, así como una reflexión acerca de la relación del proyecto con los Objetivos de Desarrollo Sostenible.

Marco Teórico

1.1 Machine Learning

El Machine Learning (ML), o Aprendizaje Automático, es una rama de la inteligencia artificial que busca crear algoritmos capaces de aprender de los datos sin que sea necesario programar cada operación de forma directa. Arthur Samuel, pionero en el área, lo definió como "el campo de estudio que da a las computadoras la capacidad de aprender sin ser programadas explícitamente" (Samuel, 1959). Más adelante, Tom Mitchell ofreció una definición más formal desde una perspectiva de ingeniería: "Un programa de computadora aprende de la experiencia E con respecto a una tarea T y una medida de rendimiento P, si su rendimiento en T, medido por P, mejora con la experiencia E" (Michell, 1997).

Por ejemplo, un filtro de spam puede ser un sistema de aprendizaje automático: si le proporcionamos ejemplos de correos etiquetados como spam o ham (no spam), el sistema aprende a clasificar nuevos correos. En este caso, la tarea T es clasificar correos electrónicos, la experiencia E son los ejemplos etiquetados (el conjunto de entrenamiento), y la medida P podría ser la precisión de clasificación (Keita, 2024).

A diferencia de los métodos tradicionales basados en reglas fijas, el Machine Learning ofrece soluciones más flexibles y escalables. Puede reconocer nuevas variantes simplemente observando patrones en los datos, sin necesidad de intervención humana. Esto lo hace especialmente útil en campos complejos donde los algoritmos hechos a mano no son eficaces, como el reconocimiento de voz, la visión por computadora o los sistemas de recomendación (Géron, 2019).

Los sistemas de Machine Learning pueden clasificarse según varios criterios:

- Aprendizaje incremental o por lotes, es decir, si el modelo aprende a medida que recibe nuevos datos o si se entrena con todo el conjunto de datos de una sola vez.

- Basado en instancias o basado en modelos, si compara nuevos datos con ejemplos previos o si construye una representación general de los datos.

- Supervisión durante el entrenamiento, se divide en: aprendizaje supervisado, no supervisado, semi-supervisado y por refuerzo.

El aprendizaje supervisado es una rama del Machine Learning donde el modelo se entrena con datos etiquetados, es decir, ejemplos que incluyen tanto las entradas como las salidas, ya asignadas. El objetivo es aprender una función (en base a los entrenamientos) que, dada una nueva entrada, prediga su salida correspondiente. En cambio, el aprendizaje no supervisado trabaja con datos no etiquetados, buscando descubrir patrones, agrupaciones o estructuras ocultas sin indicaciones previas. Mientras que el supervisado responde preguntas específicas, el no supervisado explora los datos para entender su organización interna.

En este caso de estudio, disponemos de datos que han recogido los sensores acelerómetros colocados en los mosaicos, pero estos datos no están clasificados o etiquetados, por lo que nuestro enfoque es usar Machine Learning no supervisado.

Dentro de esta técnica existen numerosos algoritmos, depende de lo que se quiera conseguir, unos se ajustarán mejor que otros.

1. 2 Algoritmos de clústering

K-Means:

El objetivo de este trabajo es hallar patrones, por lo que se deben estudiar métodos de 'clústering', es decir, aquellos que agrupan información en conjuntos. Uno de los métodos más conocidos es K-Means. Este algoritmo agrupa los datos alrededor de un número predefinido de "centros" llamados centroides, minimizando la varianza intra-cluster.

El funcionamiento de K-Means necesita conocer de antemano el parámetro 'k', que indica el número de clústeres que se desea encontrar. El proceso es iterativo: primero se inicializan 'k' centroides de forma aleatoria. Luego, en cada iteración, se asigna cada punto del conjunto de datos al centroide más cercano (usando la distancia euclidiana), formando 'k' clústeres temporales (Ilustración 1). A continuación, se recalcula la posición de cada centroide como la media de todos los puntos asignados a su clúster. Estos dos pasos se repiten hasta que las asignaciones no cambian significativamente o se alcanza un número máximo de iteraciones. Este número máximo de iteraciones es también un parámetro de entrada y evita que el algoritmo entre en bucle infinito (Kavlakoglu, 2024).

Su principal limitación, además de la necesidad de definir 'k', es que tiende a crear clústeres de forma esférica y de tamaño parecido, lo que no siempre se ajusta a la estructura real de los datos. Otro problema es que todos los puntos se asignan al grupo más cercano, lo que significa que los valores atípicos o outliers pueden alterar la posición de los centroides y afectar negativamente la calidad de los grupos.

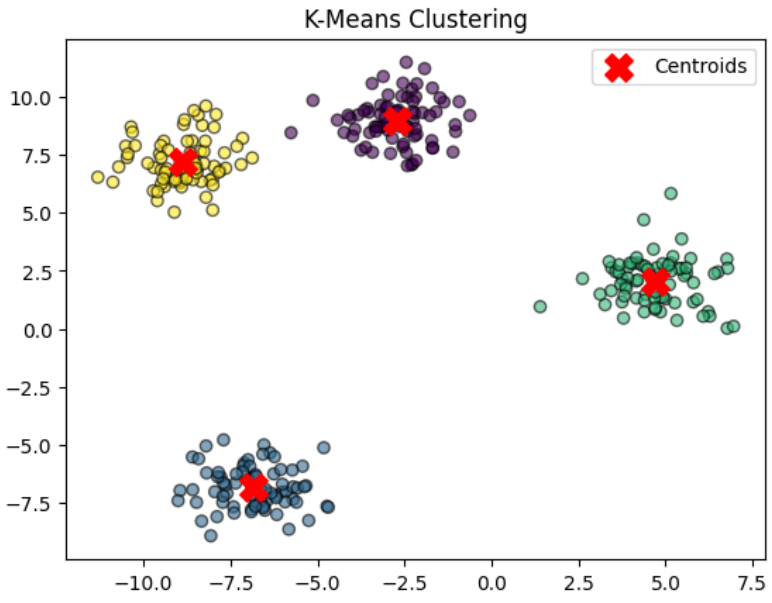

Ilustración 1: Funcionamiento de K-Means

DBSCAN

Frente a la limitación de tener que predefinir el número de grupos, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) surge como un método que no necesita que se especifique previamente el valor de los clústeres. Este algoritmo detecta grupos de cualquier forma y también identifica valores atípicos o ruido, ya que busca zonas donde los datos están densamente concentrados. Por eso se llama clustering basado en densidad. A diferencia de K-Means, DBSCAN puede encontrar grupos irregulares y reconocer puntos aislados que no pertenecen a ningún grupo.

Los parámetros principales de DBSCAN son 'ϵ' (épsilon), que representa la máxima distancia entre dos puntos para ser considerados vecinos, y 'MinPts', que es el número mínimo de puntos necesarios para formar una región densa. Se puede imaginar el vecindario de un punto como un círculo de radio ϵ alrededor de él. Un punto es alcanzable por densidad desde otro si se puede llegar a él moviéndose de vecindario en vecindario. Esta idea permite que DBSCAN expanda los clústeres a partir de los puntos núcleo (Kumar, 2024).

El funcionamiento de DBSCAN se basa en tres conceptos clave:

-Puntos núcleo: son aquellos que tienen al menos 'MinPts' puntos dentro de su vecindario de radio 'ϵ'.

-Puntos frontera: tienen menos de 'MinPts' en su vecindario, pero están cerca de un punto núcleo.

-Puntos de ruido: no son ni núcleo ni frontera, es decir, están demasiado aislados.

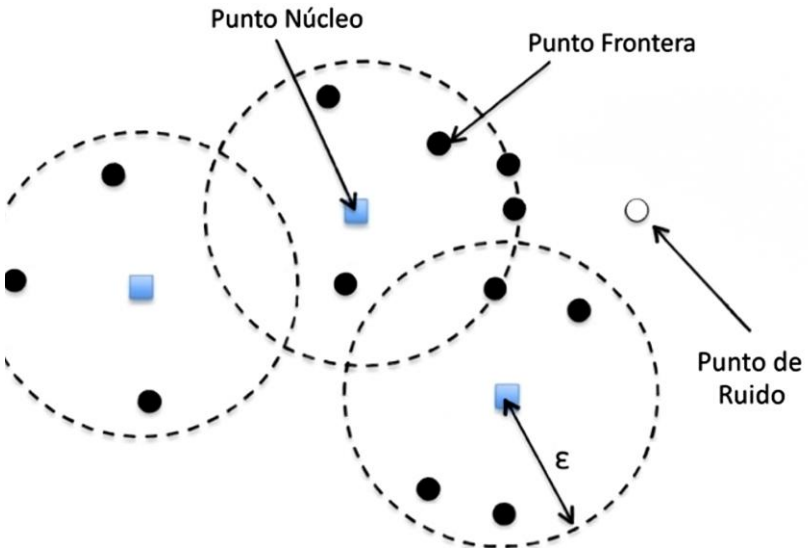

Ilustración 2: Funcionamiento de DBSCAN

El funcionamiento del algoritmo es el siguiente: primero se eligen los parámetros ϵ y MinPts. Luego se selecciona un punto que aún no haya sido visitado. Si tiene menos de *MinPts* en su vecindario, se marca como ruido. Si tiene al menos *MinPts*, se convierte en un punto núcleo y se inicia un nuevo clúster. A partir de ahí, se agregan todos sus vecinos al clúster. Si alguno de esos vecinos también es un punto núcleo, se expanden sus vecinos de manera recursiva. Si no lo es, se marca como punto frontera. Este proceso continúa hasta que todos los puntos han sido visitados. Los puntos inicialmente marcados como ruido pueden ser reconsiderados como frontera si están cerca de un punto núcleo.

Dos conceptos clave relacionados con este método son el alcance por densidad y la conectividad por densidad. El primero se refiere a que un punto 'q' es alcanzable desde 'p' si 'p' es un punto núcleo y existe una cadena de puntos núcleo conectados por distancias menores a 'ϵ'. La conectividad por densidad, por su parte, implica que dos puntos están conectados por densidad si ambos son alcanzables desde un mismo punto núcleo (Kumar, 2024).

Para elegir correctamente los parámetros, se recomienda usar conocimiento del dominio junto con herramientas como la gráfica k-distancia, que calcula la distancia al k-ésimo vecino más cercano. El codo ('elbow method') de esta gráfica suele ser un buen indicador para seleccionar el valor de 'ϵ'. En cuanto a 'MinPts', una regla general es usar el doble del número de características o dimensiones del conjunto de datos. Si los datos son ruidosos o contienen clústeres pequeños, puede usarse un 'MinPts' más bajo. En conjuntos de datos grandes, aumentar 'MinPts' ayuda a evitar clústeres pequeños.

Comparativa entre K-Means y DBSCAN

DBSCAN es una herramienta poderosa, ideal para trabajar con datos complejos y ruidosos donde no se conoce el número de clústeres. Su principal ventaja sobre K-Means es la capacidad para encontrar clústeres de forma arbitraria e identificar el ruido de forma explícita, lo que lo hace más adecuado para conjuntos de datos reales donde los outliers son comunes y no están identificados. No obstante, elegir los parámetros que la caracterizan ('ϵ' y 'MinPts') es clave para su efectividad y puede ser más compleja que elegir 'k' en K-Means.

K-Means, por su parte, es más sencillo y computacionalmente eficiente para grandes conjuntos de datos, pero su rendimiento puede verse afectado si los clústeres no son esféricos o si existen outliers. Por estas razones, es interesante comparar los resultados de ambos algoritmos para tener una visión más completa y robusta de la estructura del conjunto de datos.

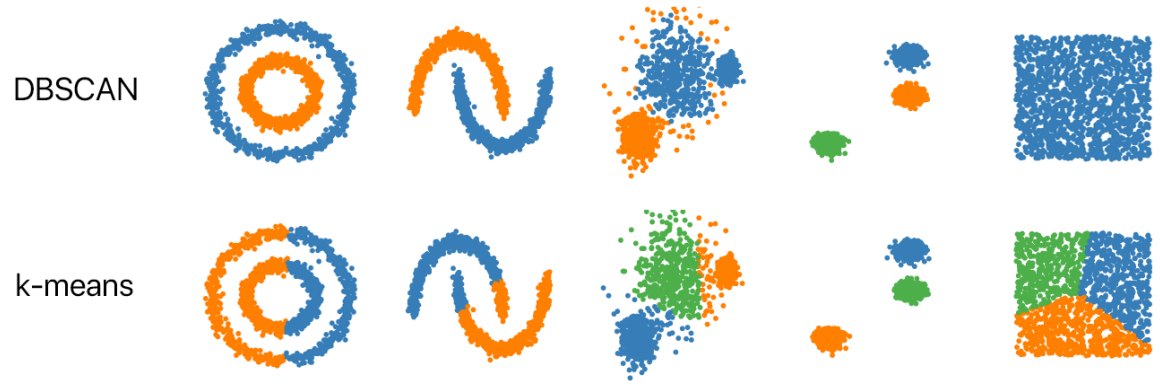

Ilustración 3: Comparativa clústering

Además de DBSCAN, existen otros métodos avanzados de aprendizaje no supervisado diseñados específicamente para la detección de anomalías o patrones novedosos en datos estructurales. Un estudio comparativo reciente (Wang, 2022), evaluó el rendimiento de varios algoritmos, incluyendo el modelo de mezcla gaussiana (GMM), máquinas de vectores de soporte de una clase (OC-SVM) y el clústering basado en densidad de picos (DPFC), aplicados a la detección de daños en estructuras utilizando datos de acelerómetros. Estos métodos comparten la ventaja de requerir únicamente datos de condiciones normales (no etiquetados) para entrenar modelos que luego identifican anomalías asociadas a daños. Entre ellos, el método DPFC demostró un rendimiento destacado en la localización de daños, gracias a su capacidad para identificar clústeres densos y puntos atípicos de manera automática. Este enfoque podría ser particularmente útil en el análisis de datos de mosaicos, donde la identificación de patrones anómalos podría correlacionarse con cambios estructurales o deterioro.

1.3 Herramientas utilizadas

A continuación, se describen las herramientas utilizadas para estudiar estos datos:

En primer lugar, para poder acceder a los datos de los sensores acelerómetros, se usó el software desarrollado por el Grupo de Investigación GPA, llamado AnalisisTAN. Su apariencia al ejecutarlo y cargar datos es la siguiente.

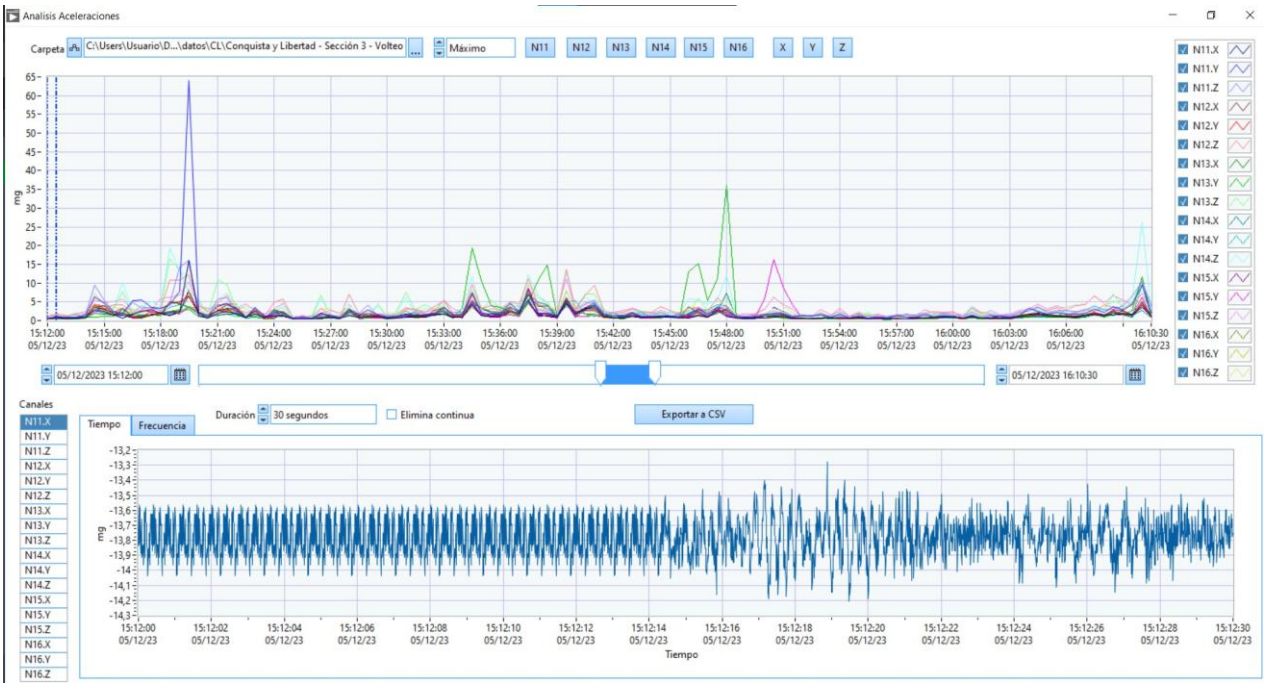

Ilustración 4: Apariencia del software.

Podríamos decir que la pantalla queda dividida en dos. En la parte superior se observan los datos seleccionados en el apartado de carpeta. En este caso se muestran los valores máximos, pero también se pueden ver la inclinación y la media.

En la pantalla se visualizan las señales de todos los sensores (y sus tres ejes) superpuestos, cada uno con un color, en las abscisas el tiempo (que se puede ampliar o recortar con la barra que tiene inmediatamente debajo) y en la ordenada la aceleración (medida en mg, $1g = 9{,}8\ \frac{m}{s^2}$).

Se puede seleccionar qué sensores visualizar en la parte derecha o en la de arriba, en la que se puede seleccionar datos solo del eje o de los tres ejes de un mismo sensor.

La parte inferior de la pantalla nos va a mostrar un intervalo más ajustado de aquel sensor+eje que seleccionemos en la parte izquierda. Se puede elegir entre visualizar intervalos desde 30 segundos hasta 3 horas. También existe una casilla que si se pulsa elimina la componente continua y una pestaña que nos permite visualizar en frecuencia (escala lineal o logarítmica) la ventana seleccionada. Por último, se permite exportar dichos datos a un documento .csv.

En segundo lugar, en el desarrollo de este proyecto se ha utilizado Google Colaboratory (Colab) como entorno de programación y ejecución de código en Python, debido a su facilidad de uso, accesibilidad y compatibilidad con bibliotecas ampliamente utilizadas en el ámbito del Machine Learning y la ciencia de datos. Google Colab (logotipo en la Ilustración 5) es una plataforma basada en la nube lanzada por Google en 2017, que permite crear y ejecutar cuadernos interactivos del tipo Jupyter Notebook desde un navegador web, sin necesidad de instalar software adicional en el equipo local (Bisong, 2019).

Ilustración 5: Google Colab

Los Jupyter Notebooks se han convertido en una herramienta fundamental en entornos académicos y profesionales por su capacidad de combinar código ejecutable, visualizaciones y texto en un único documento. Esto resulta especialmente útil en proyectos de análisis de datos, investigación reproducible y experimentación con modelos de aprendizaje automático (Google, 2023), Google Colab se basa en esta tecnología, pero añade la ventaja de ofrecer infraestructura en la nube, permitiendo a los usuarios ejecutar sus cuadernos en servidores remotos gestionados por Google.

No obstante, el uso de Google Colab también presenta ciertas limitaciones. Entre ellas destacan el tiempo restringido de uso continuo, la desconexión automática tras períodos de inactividad, y la necesidad de cargar manualmente los datos en cada nueva sesión, salvo que estén vinculados a Google Drive. A pesar de estos inconvenientes, Colab continúa siendo una opción muy valiosa para el desarrollo y prueba de modelos, especialmente en fases exploratorias o educativas, gracias a su entorno preconfigurado con librerías como NumPy, Pandas, Matplotlib, Scikit-learn y TensorFlow, entre otras (Kluyver, 2016).

Análisis de los datos

2.1 Primera aproximación

2.1.1 Contexto de los datos

Patrones:

Los datos de vibración registrados en los murales corresponden a las señales obtenidas durante el proceso de descolgado y traslado de las fachadas del Centro SCOP, en Ciudad de México (Lorenzana, Villacorta, Val, & Izquierdo, 2024), un edificio que había sufrido daños por varios terremotos. La intervención tuvo como objetivo preservar los mosaicos y murales, separándolos cuidadosamente de la estructura mediante cortes con hilo, perforaciones y maniobras con grúas, todo controlado mediante monitoreo. Cada panel, con dimensiones de hasta 20 × 10 m^2, se reforzó temporalmente con una estructura de acero en celosía para garantizar su integridad durante el levantamiento y giro posterior, minimizando el riesgo de daños a los mosaicos.

Durante estas operaciones, se emplearon sensores acelerómetros MEMS para registrar la vibración inducida por la maquinaria, los impactos durante la liberación de los paneles y los movimientos de volteo, así como las perturbaciones ambientales. Este monitoreo permitió discriminar entre vibraciones significativas, asociadas a acciones estructurales o mecánicas, y ruido ambiental (por ejemplo, viento o pequeñas perturbaciones del entorno). De este modo, se estableció un umbral mínimo de 5 mg de aceleración, asegurando que el análisis se centrara en aquellos eventos que podían comprometer la integridad de los murales y facilitando la identificación de patrones de movimiento relevantes para su manejo seguro.

Para poder pedirle a un modelo de Machine Learning que clasifique los datos, previamente debemos analizarlos. Para ello se dispone de un estudio que realizó un compañero del grupo de investigación. En él se identifican una serie de patrones en el tiempo (clasificándolos con letras mayúsculas, A-H). Hubo que revisar minuciosamente los datos recogidos en los murales: 'Independencia y Progreso Norte' secciones 1 y 2, 'Independencia y Progreso Sur' secciones 1 y 2, 'Conquista y Libertad' secciones 1, 2 y 3, 'Libertadores Norte', 'Libertadores Sur', 'Canto y Patria' secciones 1 y 2 y '4 Siglos' secciones 1, 2 y 3. En la Ilustración 6 se muestra el plano completo de la instalación, con los 23 murales.

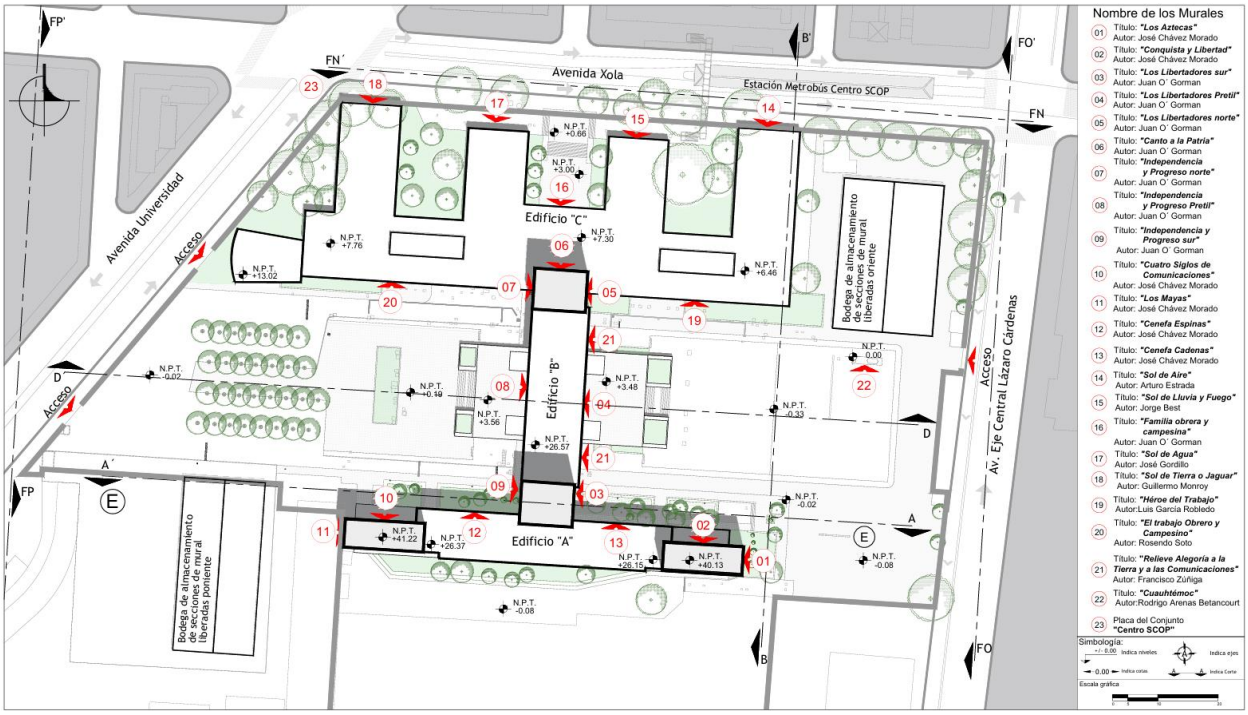

Ilustración 6: Planta de conjunto

En la ilustración 7 queda reflejado cómo cada mural queda dividido en distintas secciones para su separación del edificio y posterior descenso. Además, se establece también dónde están colocados los sensores que recogerán los datos.

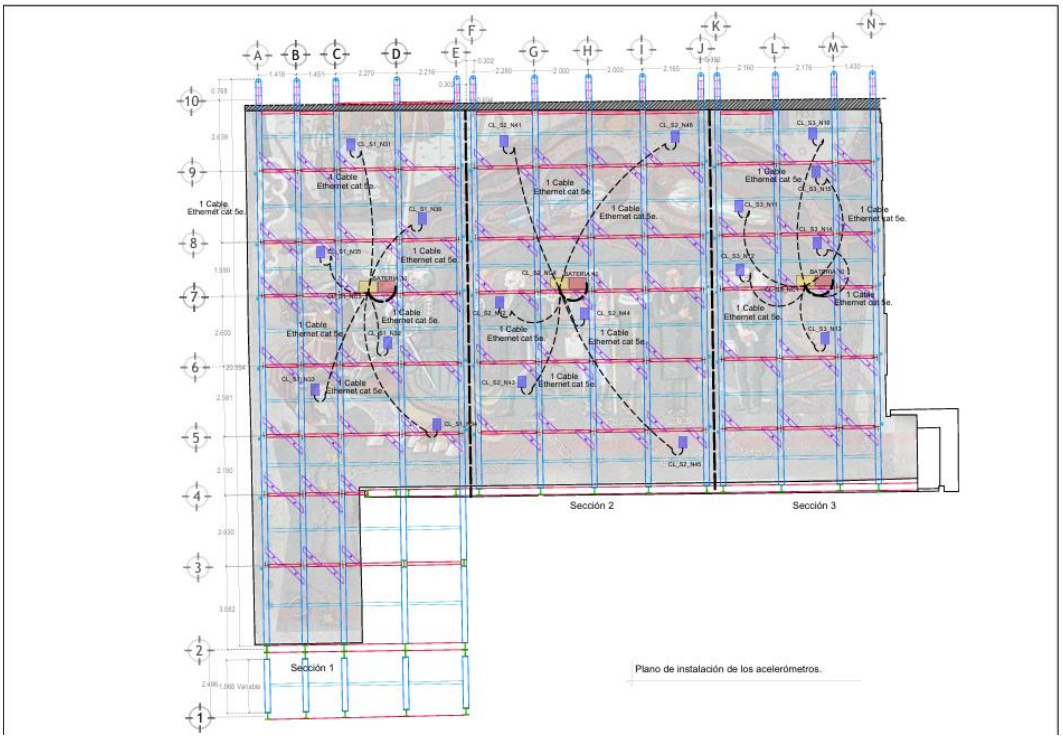

Ilustración 7: Mural 'Conquista y Libertad'

Para extraer dichos datos, se empleó el software descrito (Ilustración 4) y se fue revisando cada sensor de cada mural en busca de estos patrones. Con el fin de descartar vibraciones irrelevantes, se estableció un umbral mínimo de 5 mg en la amplitud de aceleración. Esto implica que solo se consideran significativas las señales con una intensidad mayor que el valor mencionado anteriormente, y aquellas con una amplitud por debajo de dicho valor se convierten en ruido para el entorno (vibraciones de bajo nivel causadas por factores externos, por ejemplo, por el viento, variaciones térmicas o pequeñas perturbaciones en el ambiente). Los cambios de baja magnitud en estructuras civiles pueden deberse a efectos ambientales y no necesariamente a excitaciones relevantes (Xia, 2012). En consecuencia, aplicar este umbral permite filtrar datos espurios y centrar el análisis en vibraciones potencialmente asociadas a fenómenos estructurales o a la acción directa de la maquinaria.

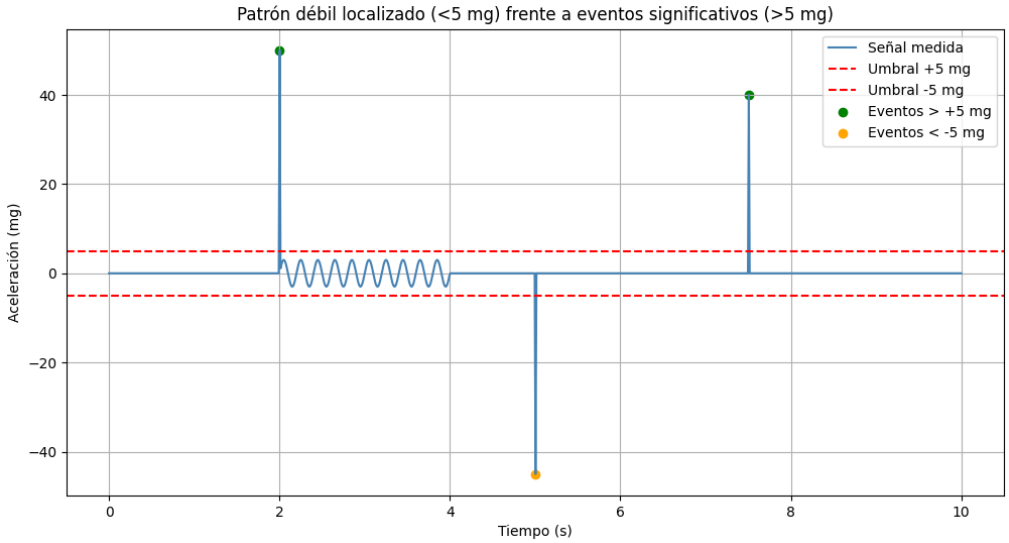

Ilustración 8: Umbrales de ruido

En la Ilustración 8 se muestra un ejemplo simulado de las vibraciones registradas. Entre los segundos 2 y 4 aparece un patrón senoidal de baja amplitud (≈3 mg), que al no superar el umbral de ±5 mg se considera ruido ambiental. En contraste, se observan picos de magnitud más significativa (≈40–50 mg) en momentos aislados, que sobresalen claramente por encima del umbral y se interpretan como eventos significativos asociados a excitaciones externas. El filtrado de señales, estableciendo un umbral permite descartar vibraciones menores debidas a factores ambientales y centrando el análisis en aquellas que pueden tener un impacto estructural (Xia, 2012).

2.1.2 Descripción de los patrones

Patrón A:

El Patrón A se caracteriza por una secuencia de picos de aceleración con amplitudes similares y periodicidad regular, típicamente entre 300 ms y 1 segundo (siendo 500 ms el intervalo más frecuente). Estos eventos suelen durar desde 2-3 segundos (con 3-5 picos) hasta varios minutos, con casos excepcionales de 30 minutos de actividad continua. Los picos aparecen tanto en ráfagas compactas (ej: 3 picos en 600 ms) como en secuencias estables (ej: picos cada 1.8 s durante 10+ minutos), manteniendo siempre una consistencia temporal notable.

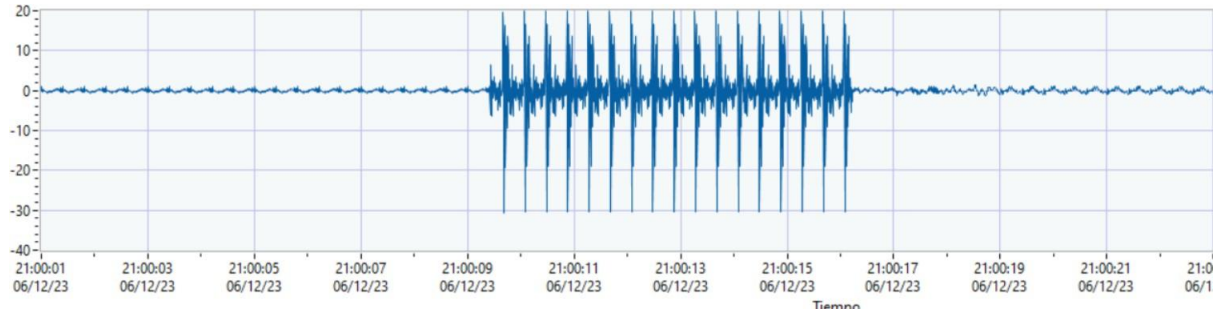

Ilustración 9: Patrón 'A'

Patrón B:

El Patrón B se caracteriza por formas de onda tipo "campana" sin picos definidos, con duraciones típicas entre 1 y 6 segundos (siendo 2-3 segundos los más frecuentes). Estos eventos aparecen tanto de forma aislada como encadenados en secuencias de varios minutos, manteniendo siempre perfiles suaves sin periodicidad clara. En algunos casos excepcionales, se observan patrones prolongados de hasta 10-12 segundos.

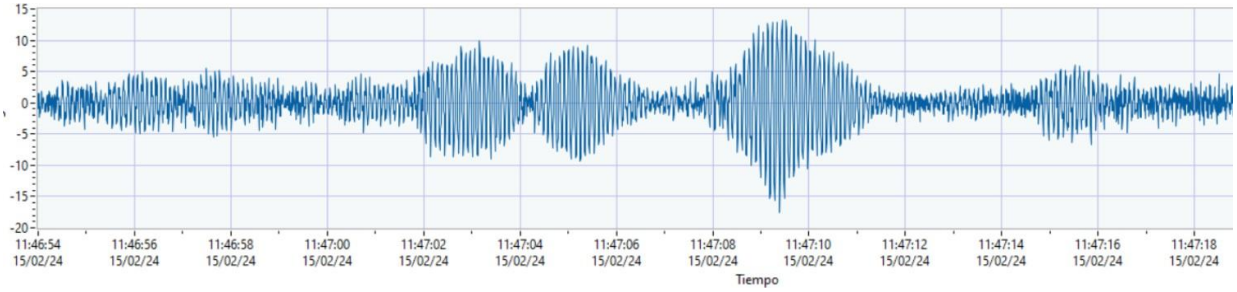

Ilustración 10: Patrón 'B'

Patrón C:

Se caracteriza por una secuencia de dos o más picos de aceleración cuya amplitud decrece progresivamente en el tiempo. Los eventos suelen durar entre 1 y 14 segundos, con intervalos entre picos que varían desde 0.5 segundos hasta 7 segundos (siendo los más comunes de 2-3 segundos).

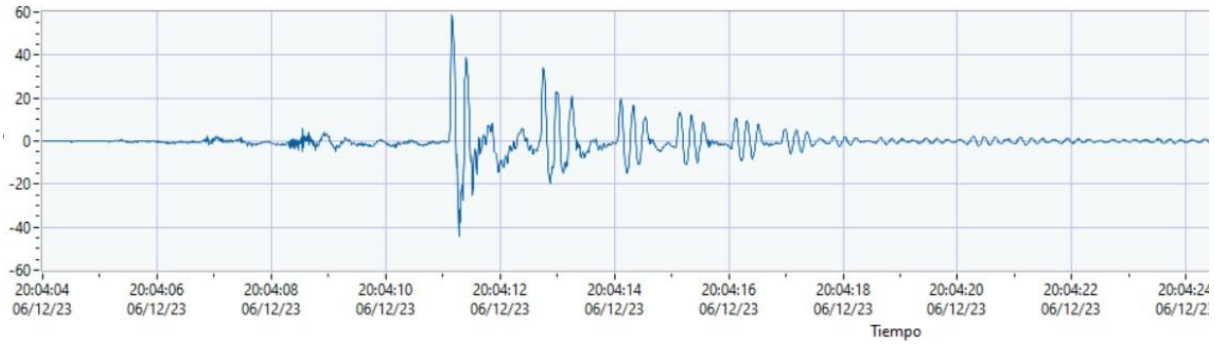

Ilustración 11: Patrón 'C'

Patrón D:

Presenta dos o más picos de aceleración cuya amplitud aumenta progresivamente en el tiempo. Los intervalos entre picos son regulares (típicamente 0.5 a 2 segundos), y la duración total del patrón varía entre 2 y 20 segundos.

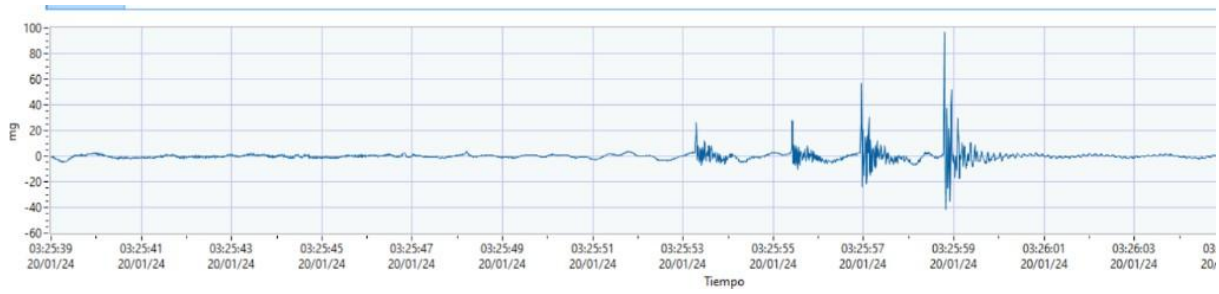

Ilustración 12: Patrón 'D'

Patrón E:

Consiste en picos de aceleración aislados y de alta amplitud, que destacan claramente sobre el ruido de fondo. Cada evento tiene una duración de aproximadamente 1 segundo, aunque el pico en sí es instantáneo.

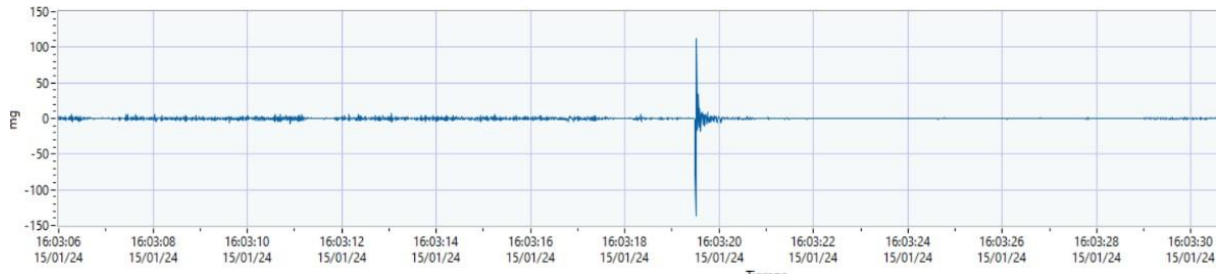

Ilustración 13: Patrón 'E'

Patrón F:

Se identifica por formas de onda senoidales de amplitud constante y larga duración (20 segundos a 1 minuto). La frecuencia dominante es clara y bien definida, con períodos que varían entre 0.5 y 3 segundos. Este patrón sugiere vibraciones sostenidas y estables, sin amortiguamiento apreciable.

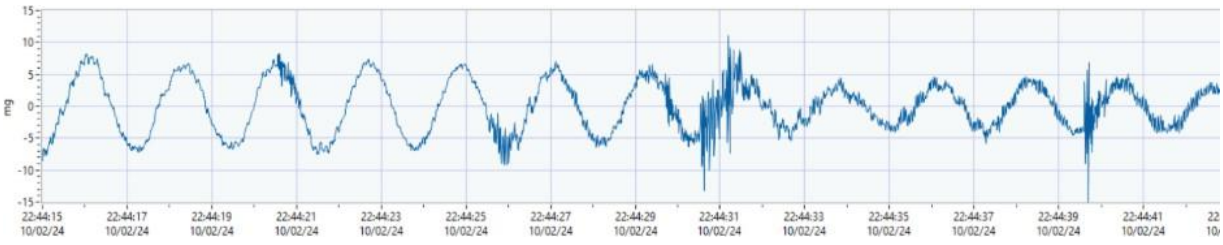

Ilustración 14: Patrón 'F'

Patrón G:

Muestra formas de onda senoidales complejas y amortiguadas, con una duración típica de 1 a 4 segundos. La amplitud decae rápidamente. No presenta picos definidos ni periodicidad clara, su espectro en frecuencia es ancho debido a la atenuación rápida.

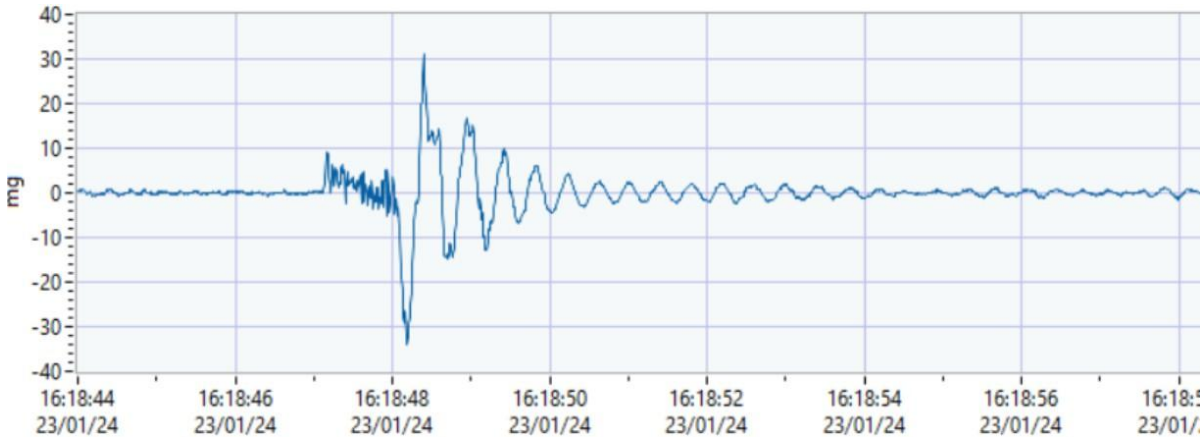

Ilustración 15: Patrón 'G'

Patrón H:

Se caracteriza por picos de aceleración negativos, que pueden aparecer aislados o agrupados con periodicidades variables (desde 0.3 segundos hasta 2 segundos). Estos eventos suelen durar entre 1 y 12 segundos, aunque en casos excepcionales se extienden hasta 10 minutos. La amplitud de los picos es consistente dentro de cada patrón, pero no sigue una tendencia creciente o decreciente.

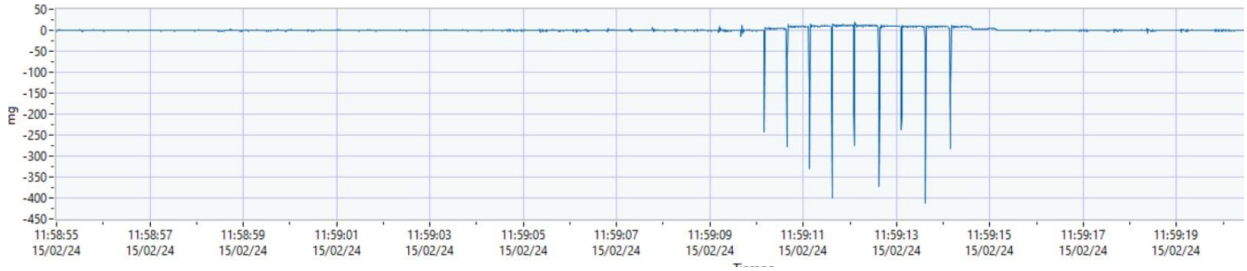

Ilustración 16: Patrón 'H'

Después de revisar los datos para localizar los patrones, se evidencia todavía más la ventaja que supone poder entrenar modelos de Inteligencia Artificial para realizar este tipo de tareas, ya que resultan muy tediosas. A menudo también es difícil distinguirlos, pues no son perfectos y en muchas ocasiones aparecen mezclados entre ellos (pueden aparecer al mismo tiempo picos positivos y negativos o picos muy fuertes junto con ondas senoidales…) o en presencia de distorsiones fuertes, lo que les hace perder la forma característica y es complicado saber qué es lo que predomina.

2.1.3 Patrones en la obra

En este punto, es interesante recordar que existen unos informes (Ilustración 17) escritos por los obreros que trabajaron en la operación. En ellos se recogen las operaciones que realizaban y también un cronograma en el que escribían qué tipo de maquinaria o movimiento llevaban a cabo y los minutos correspondientes. También registraron si en algún momento se encendían luces de aviso en los sensores estudiados.

Frente	Frente 2. Mural Los Libertadores norte		
Fecha	12 de diciembre de 2023		
Hora	**Evento**	**Tipo de equipo operando en alrededores**	**Ubicación**
21:36	Corte de varillas	Oxicorte	Nivel 3
21:37	Desbaste de trabe	Rotomartillo	Nivel 4
21:50	Corte	Motosierra	Nivel 3
21:50	Corte	Motosierra	Nivel 3
21:50	Corte en sección norte	Motosierra	Nivel 4
22:14	Desbaste de muro	Rotomartillo	Nivel 4
22:16	Desbaste de muro	Rotomartillo	Nivel 4
22:19	Desbaste de muro	Rotomartillo	Nivel 4
22:23	Corte	Motosierra	Nivel 5
22:26	Corte	Motosierra	Nivel 5
22:37	Corte de varillas	Oxicorte	Nivel 6
22:37	Desbaste de muro	Cinceladora	Nivel 5
22:37	Desbaste de muro	Maceta	Nivel 5
22:58-23:01	Refuerzo de perno	Maceta	Nivel 5
23:02	Colocación de calzas al reverso de la estructura	Martillo	Nivel 3
23:09	Corte	Oxicorte	Nivel 5

Ilustración 17: Ejemplo de informes de obreros

Se puede pensar que si se contraponen estos 'cuadernos de bitácora' de los obreros a los patrones encontrados, minuto por minuto, existirán correspondencias entre dichas maquinarias empleadas y patrones conocidos. Habrá que tener en cuenta el hecho de que en un mismo minuto pueden presentarse numerosos patrones y también en un mismo segundo podemos tener diferentes patrones dependiendo del eje o sensor que estemos midiendo. Estos informes de los obreros no son del todo minuciosos, a menudo se usan distintos términos para referirse a un mismo evento (quizás los realizaron personas diferentes). A veces no están completos para todos los mosaicos, en algunos no existe dicho informe. Aun así, se realiza una comparación para intentar sacar algún tipo de conclusiones

libertadores norte

Patrón A	17:31 Oxicorte	17:34/51 Martill	19:10/11 Martillo	20:06 Esméril	20:20 Martillo	20:34,35, 46 Marti	22:59 Maceta
Patrón B							
Patrón C	21:34 Martillo	01:07 Giro mural					
Patrón D	19:06 martillo	20:15 Esméril	00:52 mural en mov				
Patrón E	17:50 Martillo	20:15/45 esméril	22:28 rotomartillo	22:58 Maceta	00:06 Choque m	01:20 asentamien	01:25 retiro estrobos
Patrón F							
Patrón G	23:31 inicio elevación						
Patrón H	20:33 desconcet corrient	20:35 martillo	22:57 maceta	01:08 giromural			

4 siglos s3

Patrón A	16:05-07 mazo	16:12-30cote hilo	19:31 se suelta el tirfor		
Patrón B	9:43-49-54 demolición romp	10:19 demolición	15:39corte ménsulas r	16:11 corte hilo	18:36 rompedora
Patrón C					
Patrón D	16:29 corte hilo				
Patrón E	17:25 oxicorte	18:36 rompedora	22:24 asentam sección		
Patrón F	21:12-17 manibra volteo	22:09 asent horizontal			
Patrón G	20:18 carga 12 ton	20:30 orejap peq	21:13-14 volteo		
Patrón H					

Ilustración 18: Patrones con operaciones (1)

4 siglos s2

Patrón A	10:21 cincel y rotomartillo	10:36-38-40-52-53 corte con hilo	11:16-30 cortetrabe cortadora disco diama	12:03 taladroy ma	13:01-19 corte hilo	13:34-35-36-37-40	14:48 tension cable acom estrobo	18:01 torsión mural
Patrón B								
Patrón C								
Patrón D								
Patrón E	11:21 disco diamante	14:10 cincelador	14:41-59 mov sección	18:20 impacto viga	18:24 asentamiento mural			
Patrón F								
Patrón G								
Patrón H	10:33-51-53-54 corte hilo	12:03 taladro y n	13:36 mov grúa desprend sección					

4 siglos s1

Patrón A	11:48 rotomartillo	11:52-53 Mazo	12:00-03-06 ROtomart	13:04-06-13 Corte hilo	13:17taladro	14:22-34-36-40 co	17:23 giro mural
Patrón B	14:30 corte hilo	14:35 corte hilo					
Patrón C							
Patrón D	11:29 Rotomartilo	11:44 rotomartillo					
Patrón E			15:55-56 movimiento ç	16:28 movimiento secc			
Patrón F							
Patrón G	movimieno de la sección						
Patrón H	giro mura-movimiento grúa						

Ilustración 19: Patrones con operaciones (2)

libertadores sur

Patrón A	13:17 estrobos				
Patrón B	14:26 calzado reversos	14:54 cinceladora			
Patrón G	16:18 despega pieza	16:21 mazo, roto	16:31 rotomartillo	17:01 tensión cable grúa	
Patrón H	15:19 sierra				

zconq libert 3

Patrón A	16:30 maceta	17:15 estrobado				
Patrón E	17:33 maceta	18:18 maceta	19:32 elevado de la se	19:54 fin giro	20:06 toca suelo	20:20 toca suelo
Patrón F	19:21 elevación					
Patrón G	19:30 liberación seción	19:34 pieza libre				
Patrón H	17:35 maceta	18:18 maceta	20:29 maceta			

conq y libert 1

Patrón A	16:27 mazo			
Patrón B	17:50 corte com hilo			
Patrón C				
Patrón D	17:09 corte con hilo			
Patrón E	16:27 mazo	17:34 corte hilo		
Patrón F	21:04 impacto esquina			
Patrón H	16:13 cinceladora	20:39 golpea edi	21:36 volteo	21:57 asentamiento suelo

Ilustración 20:Patrones con operaciones (3)

El análisis de las tablas (Ilustración 17) debe ser interpretado. La información proviene de los cuadernos de trabajo, donde se recogen las acciones realizadas, la maquinaria empleada y un cronograma con las horas y minutos correspondientes. Sin embargo, presentan limitaciones: no siempre incluyen todos los eventos, y en ocasiones se concentran varias actividades o patrones en un mismo intervalo temporal. A pesar de estas limitaciones, la confrontación entre los cuadernos y los patrones identificados puede resultar útil para explorar posibles correspondencias.

La estrategia es revisar mural por mural, los minutos en los que hay un evento marcado, qué información nos proporcionan los sensores. Se toma de todas las formas de onda, la que predomine, puede ser porque está presente en varios ejes o porque tenga el valor de aceleración más elevado. De esta forma se asocian los eventos al patrón (Ilustración 18, 19, 20).

Tras una revisión y contraposición de los murales examinados, los patrones A, B, C, D y F, en los registros muestran una gran variedad de acciones: cortes con hilo, uso de rotomartillos, cinceladoras o esmeriles, entre otros. No se observa que estén claramente vinculados a un solo tipo de movimiento. Más bien, parecen intervenir en tareas puntuales o de apoyo que varían según lo que se necesitaba en cada momento de la obra. Por ello, no resulta fácil asociarlos a una acción concreta.

El Patrón E tiende a aparecer en fases de asentamiento de secciones sobre el suelo o contra estructuras, así como en choques o contactos con vigas y muros. Esto se aprecia, al menos parcialmente, en Libertadores Sur, Libertadores Norte y 4 Siglos S2. En algunos casos, su presencia se anticipa a movimientos de piezas o acciones de golpeo, lo que sugiere una relación con el ajuste o posicionamiento final. No obstante, al igual que ocurre con los otros patrones, la correspondencia no siempre es exacta, aunque parece lógico asociar un “golpe seco” con un pico de aceleración elevado, como indica este patrón.

El Patrón G aparece en varios casos asociado con acciones de elevación, liberación o movimiento de secciones mediante grúa. Por ejemplo, en Libertadores Sur coincide con el despegue de piezas, el uso de herramientas como el mazo y el rotomartillo, la tensión de cables y la posterior liberación de secciones. En 4 Siglos S1 se describe explícitamente un movimiento de sección en paralelo al Patrón G, mientras que en 4 Siglos S3 coincide con operaciones de carga, volteo y manipulación de piezas. En Libertadores Norte vuelve a aparecer en un contexto de elevación. Aunque estas coincidencias sugieren una relación, no siempre es posible establecer una correspondencia directa y única entre este patrón y un tipo específico de acción.

En cuanto al Patrón H, se encuentra con frecuencia en momentos donde se emplean herramientas de impacto como martillos, mazos o macetas. Así ocurre en Libertadores Sur, donde aparece vinculado tanto a fases de golpeo como al asentamiento de piezas en el suelo, o en Libertadores Norte, donde coincide con el uso explícito de martillo y maceta. En 4 Siglos S2 los registros son menos claros, ya que se documentan cortes y movimientos de grúa sin relación directa con el golpeo, mientras que en 4 Siglos S3 no se observa este patrón. Parece que el Patrón H se puede asociar habitualmente con acciones de impacto.

2.1.4 Patrones en frecuencia

En un primer estudio, después de hallar los patrones mencionados en el tiempo, se clasificó, junto a ellos, su espectro. Se intentó enfrentar estos patrones, que ya se sabía que iban a estar presentes, con su correspondencia en la frecuencia (tanto en escala lineal como logarítmica). A pesar de que un patrón tan claro en el tiempo sugiere un patrón en frecuencia, la realidad es que debido a que suelen ser de corta duración, la presencia de múltiples frecuencias y otros fenómenos como ruido, en el dominio frecuencial es difícil identificar un patrón (Ilustración 21, 22).

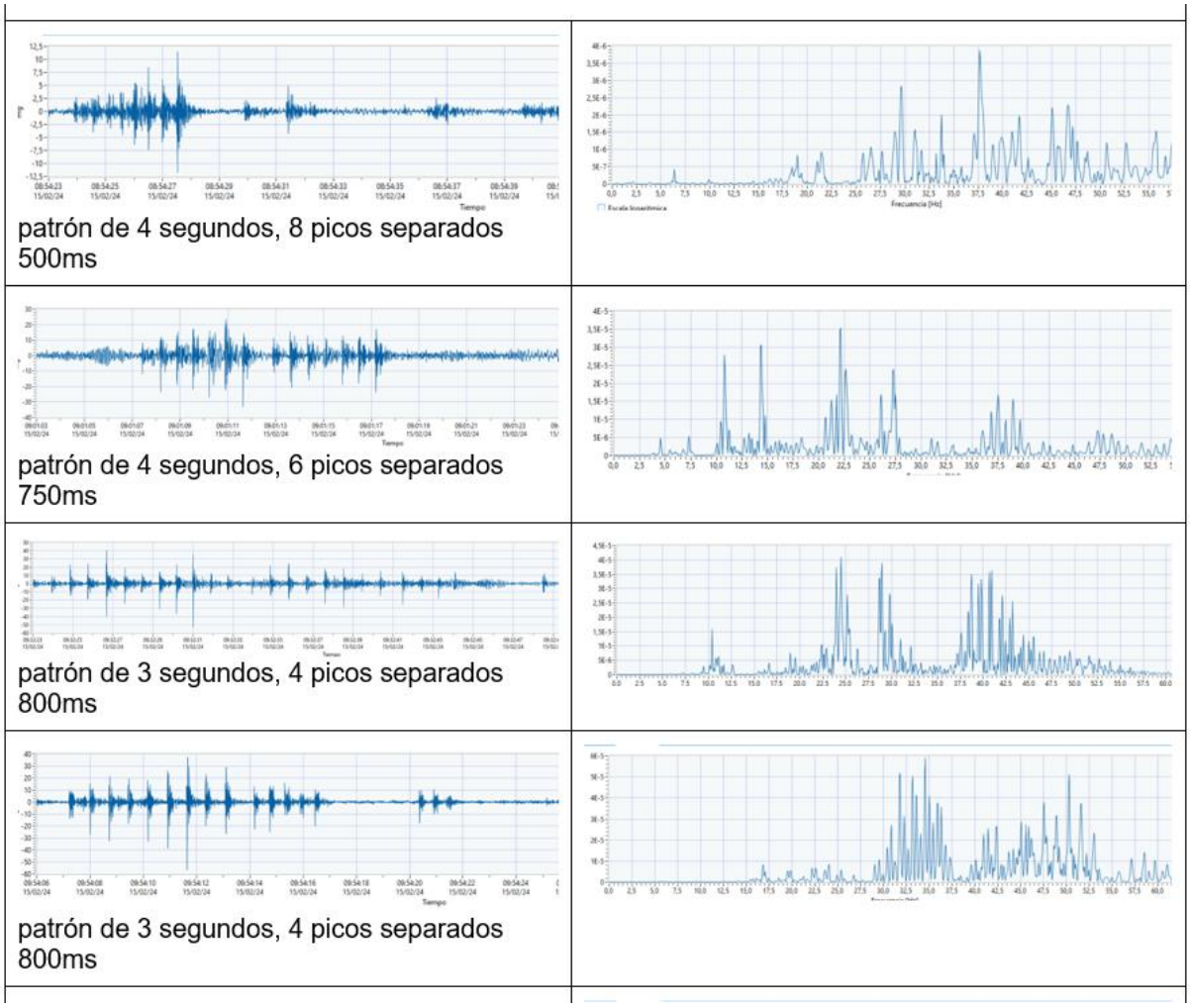

Ilustración 21: Correspondencia frecuencial ('D')

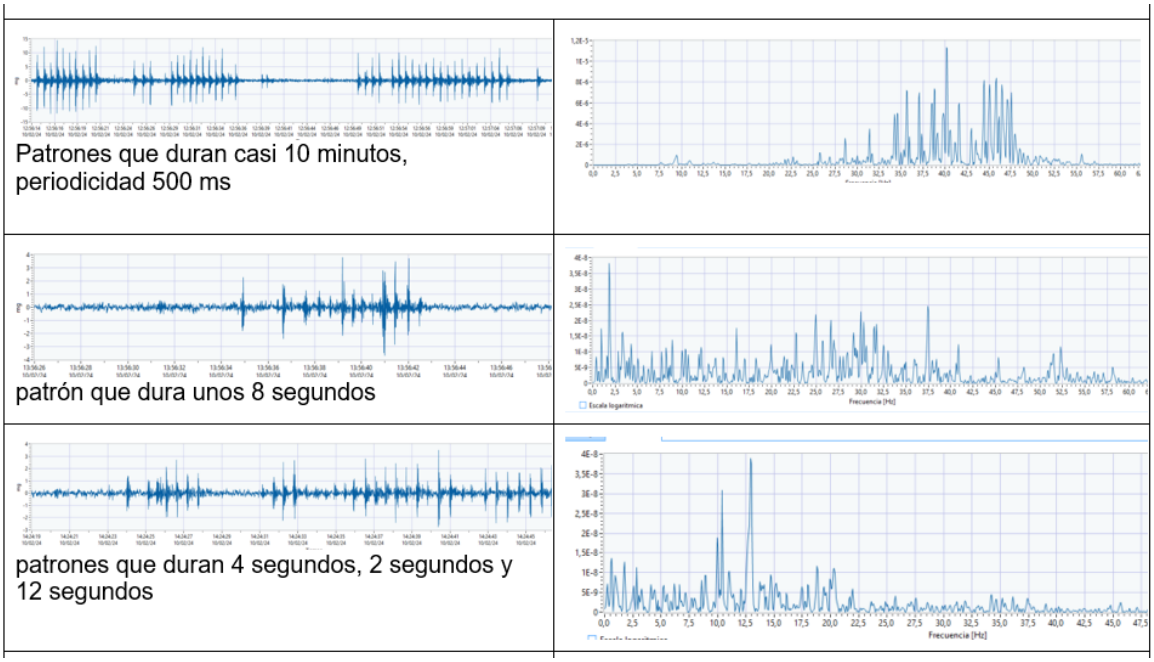

Ilustración 22: Correspondencia frecuencial ('A')

En algunos casos, como es el del patrón F, sí existe esa correspondencia, pues debido a que este patrón se identifica por formas de onda senoidales de amplitud constante, la correspondencia en frecuencia se muestra como un único pico muy claro en la frecuencia dominante (ilustración 23).

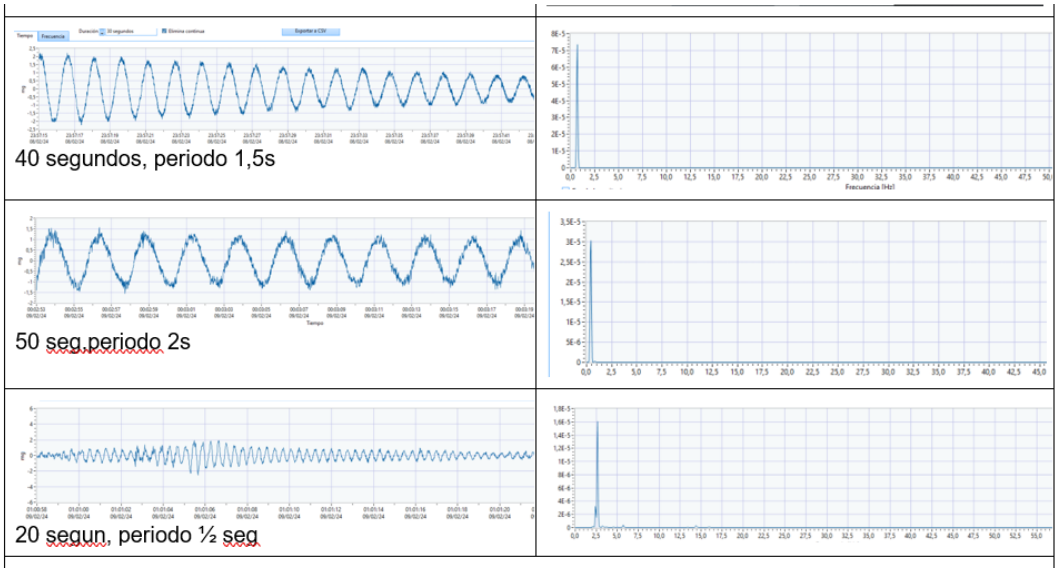

Ilustración 23: Correspondencia frecuencial ('F')

Es por esto, que el análisis se centra en el dominio temporal.

2.1.5 Exportar los patrones

Para poder probar distintos algoritmos de Machine Learning y apreciar cuál puede funcionar mejor para nuestro caso de estudio, nos será útil tener estos datos clasificados. Una vez hemos localizado los patrones significativos de cada sección de los datos, se guardan capturas de cada dato, junto con su duración y periodicidad si lo fuese, (Ilustraciones 21, 22, 23).

Estos patrones tienen duración variable, desde menos de un segundo (mayoritariamente patrón E), hasta patrones de un par de minutos (mayoritariamente patrones A o B). Se toma la decisión de capturar estos datos con una duración fija de 5 segundos. Los patrones que sean más largos se recortarán en varias capturas y aquellos que sean más cortos tendrán valores nulos al principio y al final de la captura.

Una vez tomada esta decisión, se vuelve a recorrer el conjunto de datos de los mosaicos y se descargan los 'paquetes de datos' de duración 5 segundos, para ello se encuentra el patrón en la gráfica y se pulsa el cuadro 'Exportar a CSV' (Ilustración 4). Pero hay que tener en cuenta que en la pantalla de visualización se ve el sensor y eje concreto en el que puede existir el patrón que buscamos, al descargar el CSV correspondiente a esos datos, se descargarán también los datos del resto de sensores y ejes (6 sensores con 3 ejes cada uno en total) y en ellos puede que el patrón sea diferente o puede no existir ningún patrón.

Los CSV exportados tienen el siguiente aspecto:

Tiempo	N31.X	N31.Y	N31.Z	N32.X	N32.Y	N32.Z	N33.X	N33.Y	N33.Z	N
32:29,5	-1,14E-02	1,10E-02	3,66E-03	-9,21E-03	3,14E-02	1,24E-01	2,22E-03	3,53E-03	6,74E-02	
32:29,5	1,30E-02	-4,65E-03	1,06E-01	2,70E-02	-1,92E-02	-1,34E-01	2,15E-02	-1,32E-02	-2,93E-02	
32:29,6	-5,18E-03	1,05E-02	-3,40E-03	-1,67E-02	8,07E-03	3,94E-02	-1,05E-02	5,46E-03	-2,32E-02	
32:29,6	6,25E-03	-2,86E-02	-1,27E-01	-2,21E-03	1,64E-03	8,51E-02	-1,78E-03	4,72E-03	-3,73E-02	
32:29,6	-1,21E-02	-1,74E-03	4,45E-02	1,61E-02	-1,96E-02	-6,71E-02	8,90E-03	-7,07E-03	3,81E-02	
32:29,6	-1,76E-02	1,80E-02	6,90E-02	-6,06E-02	9,78E-03	2,13E-02	-3,76E-02	-1,19E-02	6,40E-02	
32:29,6	3,34E-03	2,41E-02	8,33E-03	3,64E-03	8,68E-03	1,69E-02	-2,85E-03	-2,16E-03	-7,39E-02	
32:29,6	-3,36E-03	-1,59E-02	1,56E-02	7,07E-02	-1,13E-02	-1,30E-01	2,59E-02	1,84E-02	8,60E-03	
32:29,6	1,80E-02	-3,14E-02	-8,59E-02	-4,22E-02	1,19E-02	1,15E-01	-1,15E-02	-4,38E-03	2,53E-02	
32:29,6	-2,94E-02	3,19E-02	6,54E-03	-2,05E-02	8,60E-03	1,13E-01	-1,39E-02	-5,53E-03	-1,71E-02	
32:29,6	-1,55E-02	-3,61E-03	8,43E-02	-1,16E-03	-1,29E-02	-1,18E-01	6,43E-03	-3,95E-03	4,01E-02	
32:29,6	-2,03E-03	-8,82E-03	2,64E-02	-4,08E-03	-1,17E-02	-3,99E-02	1,75E-04	-5,20E-03	1,42E-02	
32:29,6	2,70E-02	-8,28E-03	-5,69E-02	-2,32E-03	1,29E-02	1,46E-01	-7,78E-03	4,66E-03	-5,53E-02	
32:29,6	-1,59E-02	6,14E-03	1,24E-02	7,70E-03	-2,73E-03	-1,58E-02	1,41E-03	3,51E-03	1,89E-02	
32:29,7	-9,70E-03	1,53E-02	3,96E-02	3,71E-03	7,74E-03	-9,52E-02	2,08E-03	7,18E-03	1,22E-02	
32:29,7	-3,44E-03	2,75E-03	-2,61E-03	-1,41E-02	6,65E-03	5,97E-02	-4,41E-03	-4,84E-03	9,99E-03	
32:29,7	2,57E-03	-1,60E-02	-2,36E-03	-3,04E-03	-1,34E-02	3,10E-02	5,28E-03	-2,00E-03	5,32E-03	
32:29,7	1,57E-02	5,57E-03	-7,75E-03	1,95E-03	8,46E-03	4,16E-02	5,40E-03	4,53E-03	2,27E-02	
32:29,7	-8,68E-03	1,14E-02	2,64E-02	1,31E-02	-4,28E-03	-7,95E-02	-5,51E-03	8,73E-03	-1,14E-02	
32:29,7	-1,48E-03	-3,14E-03	8,45E-03	-1,34E-02	6,55E-03	1,06E-02	-1,04E-02	-1,60E-03	-7,45E-03	

Ilustración 24: 'Aspecto de los ficheros exportados'.

El aspecto de los ficheros resultantes puede verse en la Ilustración 24. La primera columna corresponde siempre a la marca temporal, en nuestros datos de duración 5 segundos, contendrá 625 filas de datos. Después hay 18 columnas, una para cada sensor y eje. Por lo que el próximo paso es añadir una última fila en la que se escribirá el patrón presente en esos cinco segundos. Para que podamos trabajar con algoritmos de Machine Learning con estos datos, tendremos que renombrar los patrones, ahora se les designan números. El patrón A pasa a ser el 1, el B el 2 y así sucesivamente hasta llegar al H, que será el 8.

Otro aspecto observado durante esta fase es que los patrones rara vez se limitan a una única señal: en muchos casos se manifiestan en varios sensores a la vez, aunque tal vez solo en uno de los tres ejes; o, al contrario, aparecen en los tres ejes pero solo en un sensor concreto. En algunos patrones, como el Patrón E, lo más habitual es que la señal sobresalga en una sola columna, mientras que otros, como el Patrón A, tienden a aparecer de manera simultánea en diferentes sensores. También se detectaron combinaciones frecuentes, por ejemplo entre los patrones A y H, lo que refleja la complejidad de las obras. En cuanto a que esté presentes en un solo sensor, parece lógico mirando la Ilustración 7, en la que se muestra la disposición geográfica de los sensores. Si en un determinado momento los obreros trabajan en una zona aislada, en la que solo hay un sensor, podría solo recoger las vibraciones este acelerómetro.

Dado que uno de los objetivos es estudiar en qué medida los patrones son distinguibles entre sí, resulta interesante profundizar en qué eje y qué sensor muestran con mayor claridad cada patrón. En algunos casos, el eje vertical (Z) refleja mejor los impactos o asentamientos (como ocurre en los patrones E y H), mientras que los ejes horizontales (X o Y) capturan de forma más evidente las vibraciones sostenidas (como en los patrones F y G).

Pese a la dificultad añadida de que en muchos intervalos se superponen diferentes formas de onda, esta estrategia ofrece una base sólida para entrenar y evaluar algoritmos de Machine Learning, así como para estudiar qué sensores y ejes proporcionan información más útil en la identificación de cada patrón. Una vez se tienen todos los datos descargados y bien clasificados y etiquetados, se tiene todo lo necesario para comenzar con los algoritmos de Inteligencia Artificial.

2.2 Aplicación de algoritmos

2.2.1 Preprocesado de los datos

Existe un trabajo intermedio entre la descarga de los datos y su preprocesado. En la herramienta AnalisisTAN, se visualiza un sensor en un eje, se identifica un patrón y se 'experta como .csv'. Como ya se ha mencionado, este documento exportado tendrá una columna para cada sensor y eje (18 en total), es necesario revisarlo junto con la herramienta de visualización para ir asignando a cada columna el patrón que le corresponde. Esto habrá que hacerlo con cada patrón que hayamos detectado en los datos de los murales, puesto que se descargan los datos de todos los sensores a la vez. Inicialmente se guardaban en carpetas por mural/patrón, aunque más tarde se vio que no era necesario.

El proceso a seguir una vez todos los datos estaban descargados y etiquetados manualmente según el patrón que mostraban, comienza el preprocesado en python, con la herramienta Google Colab.

Primero, para poder trabajar con todos los datos de los que disponemos, lo más cómodo era tenerlos disponibles en el propio Google Drive, por lo que se guardaron ahí. De esta forma, se puede acceder a ellos en cualquiera de las ejecuciones que se realicen solo con compilar el siguiente código al inicio de la sesión:

```
from google.colab import drive
import pandas as pd

#Montar Google Drive
drive.mount('/content/drive')
carpeta_raiz = "/content/drive/MyDrive/TFG/datos por patrones"
```

En la carpeta 'datos por patrones', hay otras subcarpetas en las que están clasificados los datos según el patrón al que pertenecen y según el mural del que han sido extraídos. Cada dato tiene las dimensiones que proporciona la aplicación asociada a LabView (625 filas, una columna de tiempos y una de datos para cada sensor-eje) y una última fila más, añadida posteriormente, que proporciona el número de patrón asignado.

Gracias a que todas tienen la misma dimensión, se puede proceder a unir todos estos documentos mediante concatenación, puesto que lo que queremos es un gran documento en el que cada fila corresponda a un experimento (las columnas son datos) y la última celda sea el patrón.

```
import pandas as pd
import os

main_folder = '/content/drive/MyDrive/TFG/datos por patrones'
output_file =
'/content/drive/MyDrive/TFG/resultados/resultado_concatenado.csv'
expected_rows, expected_cols = 627, 19

valid_dfs = []

# Recorrer todos los archivos en la carpeta y subcarpetas
for root, _, files in os.walk(main_folder):
    for file in files:
        file_path = os.path.join(root, file)

        try:
            # Leer archivo (separado por espacios)
            df = pd.read_csv(file_path, sep='\s+', header=None,
encoding='latin1')

            # Verificar si tiene las dimensiones correctas
            if df.shape == (expected_rows, expected_cols):
                valid_dfs.append(df)


            else:
                   print(f"Saltando {file_path}: dimensiones {df.shape}
           != ({expected_rows}, {expected_cols})")

              except Exception as e:
                  print(f"Error leyendo {file_path}: {e}")

      # Concatenar todos los dataframes horizontalmente
          final_df = pd.concat(valid_dfs, axis=1)
          print(f"DataFrame final: {final_df.shape[0]} filas ×
      {final_df.shape[1]} columnas")
          # Guardar
          final_df.to_csv(output_file, index=False, header=False)
```

Como podemos observar, el tamaño que tenía cada uno de los datos era 627x19, lo que no es coherente con lo que necesitamos, por cada dato queremos una fila y 626 datos, por ello vamos a prescindir de toda la información extra que no aporta nada, como las marcas temporales, el número del sensor que está midiendo y, por supuesto, también hay que trasponer el documento entero, pues los algoritmos que utilizaremos trabajan fila a fila.

```
import pandas as pd

# rutas
input_file =
'/content/drive/MyDrive/TFG/resultados/resultado_concatenado.csv'
output_final =
'/content/drive/MyDrive/TFG/resultados/resultado_final.csv'

print("Leyendo archivo concatenado...")
df = pd.read_csv(input_file, header=None)

#Eliminar columnas con 'Tiempo' en la primera fila
time_columns = df.columns[df.iloc[0] == 'Tiempo']
print(f'Se eliminan {len(time_columns)} columnas con "Tiempo"')
df_clean = df.drop(columns=time_columns)

# Transponer y eliminar primera columna
df_transposed = df_clean.T.iloc[:, 1:]  # Transponer y eliminar
primera columna

print(f'Dimensiones finales: {df_transposed.shape}')
df_transposed.to_csv(output_final, index=False, header=False)

    print(f'Archivo final guardado en: {output_final}')
```

Ya tenemos el documento con los datos como lo queremos, el siguiente paso es hacer una breve comprobación para calcular cuántos datos/ejemplos tenemos correspondientes a cada patrón, para ello se ejecuta:

```
import pandas as pd

# Leer el archivo CSV
input_path =
'/content/drive/MyDrive/TFG/resultados/resultado_traspuesto.csv'

df = pd.read_csv(input_path, header=None)

columna_xb = 625 # Índice de la columna 'XB' (última columna)

# Reemplazar ',' por '.' en esa columna
df[columna_xb] = df[columna_xb].astype(str).str.replace(',', '.',
regex=False)

df[columna_xb] = pd.to_numeric(df[columna_xb], errors='coerce')

df_filtrado = df.dropna(subset=[columna_xb])

df_filtrado.loc[:, columna_xb] =
df_filtrado[columna_xb].round().astype(int)
```

```
# Contar por patrón
conteo_patrones = df_filtrado[columna_xb].value_counts().sort_index()

# Mostrar resultado
print("\n Filas por patrón:")

print(conteo_patrones)
```

El resultado obtenido tras esta ejecución es:

Patrón	Número de filas
0	6548
1	1022
2	219
3	122
4	178
5	790
6	24
7	201
8	238

Ilustración 25: datos por patrón

Este resultado puede ser útil para entender cómo clasifica ML los clústeres o para decidir si hace falta hacer algún otro tipo de tratamiento previo antes de 'enviar los datos' al algoritmo.

Podemos observar que el patrón predominante es el 0, el que corresponde al ruido, con más de la mitad de los datos. Otro valor llamativo es el patrón 6, que simplemente registra 24 datos.

El siguiente paso es eliminar la columna perteneciente a los patrones, pues cuando enviemos el csv al algoritmo, debe clasificarlo por sí mismo.

```
import pandas as pd

# Rutas de archivos
ruta_original =
'/content/drive/MyDrive/TFG/resultados/resultado_traspuesto.csv'
ruta_nueva = '/content/drive/MyDrive/TFG/resultados/sin_patron.csv'

# Leer el CSV y eliminar la última columna
```

```
df = pd.read_csv(ruta_original, header=None)
df_sin_ultima = df.iloc[:, :-1]

# Guardar el nuevo archivo
df_sin_ultima.to_csv(ruta_nueva, index=False, header=False)

print("Nuevo archivo guardado en:", ruta_nueva)
print(f"Dimensiones originales: {df.shape}")
print(f"Dimensiones finales: {df_sin_ultima.shape}")
```

Ahora sí que podemos decir que tenemos el documento que vamos a enviarle a nuestros algoritmos elegidos: K-Means y DB Scan. Se eligen estos como algoritmos de referencia para la clusterización de los patrones. La elección se debe a que a que son métodos ampliamente utilizados, bien documentados y con implementaciones estables, lo que facilita su replicabilidad y evaluación. Además, ofrecen enfoques complementarios: mientras K-Means permite evaluar la separación de los datos en un número fijo de grupos, DBSCAN no requiere definir *k* y resulta útil para detectar ruido o estructuras menos regulares, algo frecuente en las señales analizadas

2.2.2 K-Means

K-Means es uno de los algoritmos de clústering más populares en Machine Learning no supervisado, pero, presenta ciertas limitaciones: es muy sensible a la escala de las variables, por lo que es recomendable normalizar o estandarizar los datos antes de su aplicación; no puede manejar valores nulos y requiere que todos los datos sean numéricos. Además, K-Means necesita como parámetro de entrada el número de clústeres 'k'. Para hallar este valor, uno de los métodos más utilizados es el denominado Elbow Method. Este método consiste en ejecutar el algoritmo K-Means para un rango de valores de 'k' y calcular la suma de los errores cuadráticos dentro de los clústeres (Within-Cluster Sum of Squares, WCSS).

A medida que 'k' aumenta, la WCSS disminuye, ya que los puntos quedan más cercanos a los centroides de su clúster correspondiente. Sin embargo, después de cierto punto, el beneficio de aumentar 'k' se reduce significativamente, generando una especie de "codo" en la gráfica (Ilustración 18). Este punto de inflexión se interpreta como el número óptimo de clústeres, equilibrando la reducción de error y la complejidad del modelo (Kodinariya, 2013). En nuestro caso, si bien contamos con una idea previa sobre el número de patrones de vibración que deseamos identificar, el Elbow Method nos permite corroborar y justificar esa elección de manera más objetiva, asegurando que los clústeres encontrados reflejen de manera consistente el comportamiento de los datos durante las operaciones llevadas a cabo.

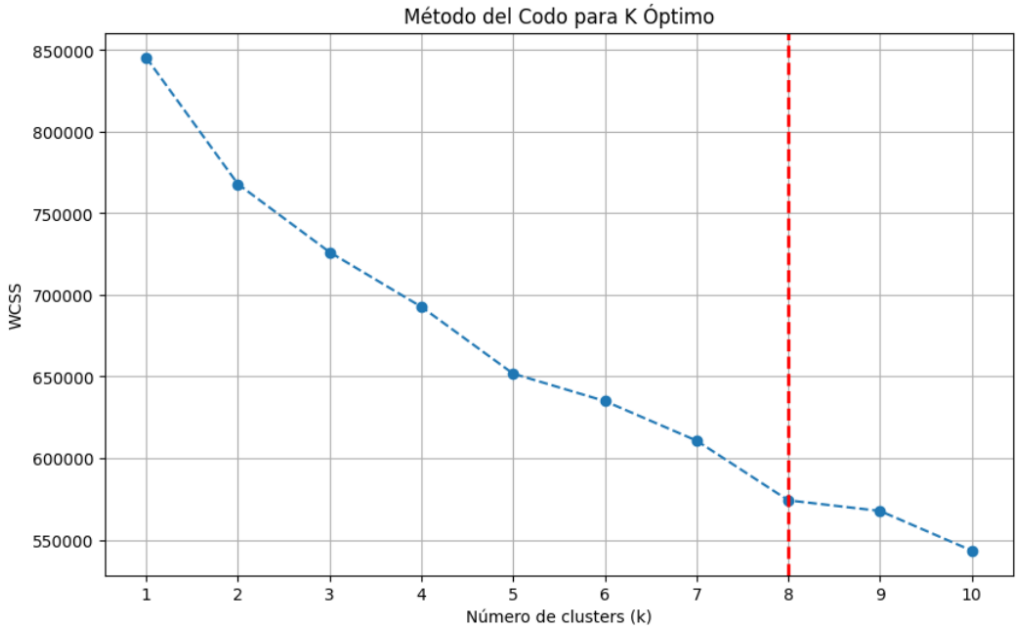

Ilustración 26: 'Elbow method'

Como se ha mencionado, es muy sensible con el formato de los datos, es por eso que hay que revisar con cuidado cómo son las filas del .csv que queremos que evalúe. En este caso, tenían la forma visible en la Ilustración 12, por lo que el primer paso es convertir la notación científica y prestar atención a la separación de los datos y los números con puntos o comas o a los valores nulos. Este paso es crucial, pues cualquier error en los datos puede hacer que el algoritmo no funcione, que clasifique todos los clústeres erróneamente, que solo detecte algunos datos, etc.

```
import pandas as pd
import numpy as np

ruta_entrada = '/content/drive/MyDrive/TFG/resultados/sin_patron.csv'

ruta_salida='/content/drive/MyDrive/TFG/resultados/procesados/_procesa
dos.csv'

df = pd.read_csv(ruta_entrada, header=None, dtype=str, decimal=',',
thousands='.')

print(f"Original: {df.shape}")


# Función de conversión simplificada
def convert_formato_numerico(value):

    if pd.isna(value) or value == '': return value
    value_str = str(value).strip()

    # Reemplazar comas por puntos en todos los casos
    if ',' in value_str:
        if '.' in value_str and value_str.find('.') <
value_str.find(','):
            value_str = value_str.replace('.', '').replace(',', '.')
        else:
            value_str = value_str.replace(',', '.')

    try:
        return float(value_str)

    except:
        return value

print("Convirtiendo formatos...")

df = df.applymap(convert_formato_numerico)
nulos_totales = df.isnull().sum().sum()

nulos_por_columna = df.isnull().sum()
nulos_con_valor = nulos_por_columna[nulos_por_columna > 0]
```

```
print(f"Valores nulos totales: {nulos_totales}")

# Eliminar nulos
if nulos_totales > 0:

    print("\nEliminando filas con nulos...")
    filas_antes = len(df)
    df = df.dropna()

    print(f"Filas eliminadas: {filas_antes - len(df)}")
    print(f"Filas restantes: {len(df)}")

# Guardar y verificar
print(f"\nTipos finales: {df.dtypes.unique()}")
df.to_csv(ruta_salida, index=False, header=False)

print(f"Archivo guardado: {ruta_salida}")
print(f"Dimensión final: {df.shape}")

try:
    df_verificacion = pd.read_csv(ruta_salida, header=None)
    print(f"Verificación exitosa: {df_verificacion.shape}")

except Exception as e:
    print(f"Error en verificación: {e}")
```

Una vez se ha verificado que los datos van a ser leídos correctamente, hay que planificar las ejecuciones del algoritmo, las pruebas que vamos a realizar. Vamos a necesitar organizar los datos en distintos documentos, concretamente 9.

La distribución inicial de los patrones identificados manualmente evidencia una elevada descompensación (desbalanceo de clases) y la presencia de una clase mayoritaria de ruido (Patrón 0), lo cual puede sesgar gravemente los resultados del algoritmo (Ilustración 17).

Ante esta distribución, se plantea una estrategia experimental con dos objetivos principales:

1. Evaluar la robustez de K-Means ante diferentes escenarios de datos (balanceados vs. no balanceados, con y sin ruido).
2. Investigar el impacto del patrón 6, una clase con muy pocos ejemplos (únicamente 24), que podría actuar como un 'outlier' multivariante y distorsionar la formación de los centroides.

Para ello, se generaron nueve conjuntos de datos derivados del original, cada uno diseñado para probar una hipótesis específica:

- 'datos_procesados.csv': Datos originales sin el patrón de ruido (0). Sirve como línea base sin la clase mayoritaria.
- 'balanceo_sinruido.csv' y 'balanceo_sinruido_no6.csv': Conjuntos balanceados donde todas las clases (patrones 1-8) tienen números parecidos de muestras, excluyendo completamente el ruido. La versión '_no6' excluye además el patrón 6 para aislar su efecto.
- 'balanceo_2000ruido.csv' y 'balanceo_200ruido.csv': Conjuntos balanceados en los patrones 1-8, pero que incorporan una cantidad controlada y limitada de muestras de ruido (2000 y 200). Esto prueba la capacidad del algoritmo para identificar los patrones estructurales incluso en presencia de un volumen manejable de ruido.
- 'balanceo_2000ruido_no6.csv' y 'balanceo_200ruido_no6.csv': Versiones de los anteriores que también excluyen el patrón 6. Permiten analizar si la presencia del patrón 6 afecta a la clusterización del ruido y los otros patrones.
- 'nobalanceo_2000ruido.csv' y 'nobalanceo_200ruido.csv': Mantienen la distribución natural y desbalanceada de los patrones 1-8, añadiendo nuevamente una cantidad controlada de ruido. Esto prueba el comportamiento de K-Means en un escenario más realista pero con el ruido mitigado.

Esta aproximación multidimensional permite no solo encontrar la mejor configuración para el clústering, sino también diagnosticar problemas específicos en los datos (como el efecto de una clase minoritaria o el ruido) y entender el comportamiento del algoritmo K-Means de una manera mucho más profunda y científica, asegurando que los resultados finales son robustos y no un artefacto de la distribución sesgada de los datos.

De esta forma se busca que no haya algún problema con las dimensiones de los datos que afecte al resultado final. Para los documentos que requieren datos balanceados, se usa el siguiente código:

```
import pandas as pd
import numpy as np

df =
pd.read_csv('/content/drive/MyDrive/TFG/resultados/resultado_traspue
sto.csv')
columna_patron = df.shape[1] - 1
df[columna_patron] = pd.to_numeric(df[columna_patron],
errors='coerce')
df[columna_patron] = df[columna_patron].round().astype(int)
```

```
# Dimensiones objetivo para el balanceo
dimensiones_objetivo = {
    0: 2000,
    1: 180,
    2: 170,
    3: 122,
    4: 150,
    5: 180,
    6: 24,
    7: 160,
    8: 170
}

df_balanceado_list = []

# Balancear cada patrón
for patron, cantidad_obj in dimensiones_objetivo.items():
    df_patron = df[df[columna_patron] == patron]

    if len(df_patron) > cantidad_obj:
        df_patron_balanceado = df_patron.sample(n=cantidad_obj,
random_state=42)

    elif len(df_patron) < cantidad_obj:
        muestras_adicionales = cantidad_obj - len(df_patron)

        df_extra = df_patron.sample(n=muestras_adicionales,
replace=True, random_state=42)

        df_patron_balanceado = pd.concat([df_patron, df_extra])

    else:
        df_patron_balanceado = df_patron
    df_balanceado_list.append(df_patron_balanceado)

df_balanceado = pd.concat(df_balanceado_list)

# Mezclar las filas para evitar agrupamientos por patrón
df_balanceado = df_balanceado.sample(frac=1,
random_state=42).reset_index(drop=True)

# Verificar las dimensiones finales
conteo_final =
df_balanceado[columna_patron].value_counts().sort_index()
print("Dimensiones finales balanceadas:")
print(conteo_final)

# Guardar datos balanceados CON la columna de patrón
output_path_balanceado_con_patron =
'/content/drive/MyDrive/TFG/resultados/….csv'
```

```
df_balanceado.to_csv(output_path_balanceado_con_patron, index=False,
header=False)
print(f"\nArchivo balanceado con patrón guardado en: {...}")
# Eliminar la columna del patrón
df_balanceado_sin_patron =
df_balanceado.drop(columns=[columna_patron])
# Guardar el archivo final SIN la columna de patrón
output_path_final =
'/content/drive/MyDrive/TFG/resultados/procesados/...csv'
df_balanceado_sin_patron.to_csv(output_path_final, index=False,
header=False)
```

Con ligeras variaciones de este código se obtienen los 9 documentos mencionados previamente. Es importante recalcar que estos balanceos deben realizarse con cuidado, pues como el código reordena todos los datos, hay que guardar un documento con los datos balanceados con la columna del patrón y otro documento sin dicha columna, de esta forma se podrá contrastar esta información con la que nos devuelva el algoritmo.

Otro problema que surgió fue que al ejecutar estos códigos, los datos se imprimían en formato correcto, pero a la hora de intentar abrir los .csv en Hoja de Cálculo, los datos se corrompían y aparecían en dimensiones cambiadas, por lo que no se podía 'confiar en ellos'.

Una vez los documentos están organizados, se procede a ejecutar todas las pruebas. En total 58, siguiendo el siguiente criterio:

Con el documento no balanceado sin ruido: datos_procesados.csv:

prueba 1: kmeans sin ruido, no balanceado, 8 clústres

prueba 2:kmeans sin ruido, no balanceado 7 clústeres

prueba 3:kmeans sin ruido, no balanceado 6 clústeres

prueba 4:kmeans sin ruido, no balanceado 5 clústeres

prueba 5:kmeans sin ruido, no balanceado 4 clústeres

prueba 6:kmeans sin ruido, no balanceado 3 clústeres

prueba 7:kmeans sin ruido, no balanceado 2 clústeres

Con el documento balanceado, sin ruido: balanceo_sinruido.csv:

prueba 8:kmeans sin ruido, balanceado 8 clústeres (con el 6)

prueba 10:kmeans sin ruido, balanceado 7 clústeres (con el 6)

prueba 12:kmeans sin ruido, balanceado 6 clústeres (con el 6)

prueba 14:kmeans sin ruido, balanceado 5 clústeres (con el 6)
prueba 16:kmeans sin ruido, balanceado 4 clústeres (con el 6)
prueba 18:kmeans sin ruido, balanceado 3 clústeres (con el 6)

Con el documento balanceado, sin ruido, sin el 6:balanceo_sinruido_no6.csv:
prueba 9:kmeans sin ruido, balanceado 7 clústeres (sin el 6)
prueba 11:kmeans sin ruido, balanceado 6 clústeres (sin el 6)
prueba 13:kmeans sin ruido, balanceado 5 clústeres (sin el 6)
prueba 15:kmeans sin ruido, balanceado 4 clústeres (sin el 6)
prueba 17:kmeans sin ruido, balanceado 3 clústeres (sin el 6)

Con el documento no balanceado 2000 ruido : nobalanceo_2000ruido.csv:
prueba 19: kmeans con ruido (2000 datos), no balanceado, 9 clústeres
prueba 20: kmeans con ruido (2000 datos), no balanceado 8 clústeres
prueba 21: kmeans con ruido (2000 datos), no balanceado 7 clústeres
prueba 22: kmeans con ruido (2000 datos), no balanceado 6 clústeres
prueba 23: kmeans con ruido (2000 datos), no balanceado 5 clústeres
prueba 24: kmeans con ruido (2000 datos), no balanceado 4 clústeres
prueba 25: kmeans con ruido (2000 datos), no balanceado 3 clústeres

Con el documento balanceado y con 2000 datos de ruido: balanceo_2000ruido.csv:
prueba 26: kmeans con ruido (2000 datos), balanceado 9 clústeres (con el 6)
prueba 28: kmeans con ruido (2000 datos), balanceado 8 clústeres (con el 6)
prueba 30: kmeans con ruido (2000 datos), balanceado 7 clústeres (con el 6)
prueba 32: kmeans con ruido (2000 datos), balanceado 6 clústeres (con el 6)
prueba 34: kmeans con ruido (2000 datos), balanceado 5 clústeres (con el 6)
prueba 36: kmeans con ruido (2000 datos), balanceado 4 clústeres (con el 6)
prueba 38: kmeans con ruido (2000 datos), balanceado 3 clústeres (con el 6)

Con el documento balanceado y con 2000 datos de ruido, sin el 6: balanceo_2000ruido_no6.csv:

prueba 27: kmeans con ruido (2000 datos), balanceado 8 clústeres (sin el 6)

prueba 29: kmeans con ruido (2000 datos), balanceado 7 clústeres (sin el 6)

prueba 31: kmeans con ruido (2000 datos), balanceado 6 clústeres (sin el 6)

prueba 33: kmeans con ruido (2000 datos), balanceado 5 clústeres (sin el 6)

prueba 35: kmeans con ruido (2000 datos), balanceado 4 clústeres (sin el 6)

prueba 37: kmeans con ruido (2000 datos), balanceado 3 clústeres (sin el 6)

Con el documento no balanceado 200 ruido: nobalanceo_200ruido.csv:

prueba 39: kmeans con ruido (200 datos), no balanceado, 9 clústeres

prueba 40: kmeans con ruido (200 datos), no balanceado 8 clústeres

prueba 41: kmeans con ruido (200 datos), no balanceado 7 clústeres

prueba 42: kmeans con ruido (200 datos), no balanceado 6 clústeres

prueba 43: kmeans con ruido (200 datos), no balanceado 5 clústeres

prueba 44: kmeans con ruido (200 datos), no balanceado 4 clústeres

prueba 45: kmeans con ruido (200 datos), no balanceado 3 clústeres

Con el documento balanceo y 200 datos de ruido: balanceo_200ruido.csv:

prueba 46: kmeans con ruido (200 datos), balanceado 9 clústeres (con el 6)

prueba 48: kmeans con ruido (200 datos), balanceado 8 clústeres (con el 6)

prueba 50: kmeans con ruido (200 datos), balanceado 7 clústeres (con el 6)

prueba 52: kmeans con ruido (200 datos), balanceado 6 clústeres (con el 6)

prueba 54: kmeans con ruido (200 datos), balanceado 5 clústeres (con el 6)

prueba 56: kmeans con ruido (200 datos), balanceado 4 clústeres (con el 6)

prueba 58: kmeans con ruido (200 datos), balanceado 3 clústeres (con el 6)

Con el documento balanceo y 200 datos de ruido, sin el 6: balanceo_200ruido_no6.csv:

prueba 47: kmeans con ruido (200 datos), balanceado 8 clústeres (sin el 6)

prueba 49: kmeans con ruido (200 datos), balanceado 7 clústeres (sin el 6)

prueba 51: kmeans con ruido (200 datos), balanceado 6 clústeres (sin el 6)

prueba 53: kmeans con ruido (200 datos), balanceado 5 clústeres (sin el 6)

prueba 55: kmeans con ruido (200 datos), balanceado 4 clústeres (sin el 6)

prueba 57: kmeans con ruido (200 datos), balanceado 3 clústeres (sin el 6)

Para ello, se ejecuta el siguiente código, variando en cada ejecución el nombre del archivo del que se extraen los datos, al nombre del archivo donde se guardan y también el número de clústeres que queremos.

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

df =
pd.read_csv('/content/drive/MyDrive/TFG/resultados/procesados/nobalanc
eo_200ruido.csv', header=None)

X = df.values
X_normalized = StandardScaler().fit_transform(X.T).T

# Aplicar K-Means con X clusters
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)

clusters = kmeans.fit_predict(X_normalized)

df['cluster'] = clusters #etiquetas de cluster al DF original

cluster_counts = df['cluster'].value_counts().sort_index()

print("Datos por cluster:")
print(cluster_counts)

df.to_csv('/content/drive/MyDrive/TFG/resultados/procesados/pruebas/cl
uster_prueba45.csv', index=False, header=False)
```

A la hora de comprobar los resultados, hay tener en cuenta que algunas funciones, como la de balancear, han mezclado los números, no puedo comparar todos los resultados con el mismo .csv origen, para ello se ha guardado en un paso intermedio posterior al balanceo, un documento extra con los patrones originales ordenados de la nueva manera.

Como ejemplo de salida obtenida, en la ejecución de la prueba 1, es decir datos no balanceados, sin ruido, K-Means nos devuelve:

```
Datos por clúster:
Clúster

0    584
1     74
2    105
3     96
4    516
5     57
6    730
7     631
```

2.2.3 DBSCAN

A diferencia de los métodos que se basan en centroides, DBSCAN forma clústeres a partir de la densidad de los puntos de datos. Este algoritmo no necesita conocer como parámetro de entrada número de clústeres; en su lugar, utiliza dos parámetros fundamentales: eps (ε), que define el radio de vecindad de cada punto, y min_samples, que determina la cantidad mínima de puntos necesarios para considerar una región como densa.

Gracias a este enfoque, DBSCAN puede detectar clústeres de cualquier forma y también identificar valores atípicos. Sin embargo, depende en gran medida de elegir correctamente los parámetros eps y min_samples, y puede presentar problemas cuando los clústeres tienen densidades muy distintas.

Una estrategia habitual para seleccionar estos parámetros es el conocido método del codo (Elbow method). Este consiste en calcular la distancia de cada punto a su k-ésimo vecino más cercano (siendo k igual a min_samples), ordenar estas distancias de forma descendente y buscar un "codo" en la gráfica (ilustración 26). Este punto indica el valor de eps donde la distancia empieza a aumentar de manera significativa, marcando la transición entre zonas densas y menos densas.

Al igual que en K-Means, la preparación de los datos es muy importante. DBSCAN es sensible a la escala de las variables, por lo que normalizar o estandarizar los datos es necesario para evitar que variables con rangos mayores dominen el cálculo de distancias. Además, tampoco admite valores nulos y solo va a leer entradas en formato numérico. En nuestro caso, la estructura de los datos ya cumple con estos requisitos tras el procesamiento y la estandarización de formatos numéricos.

Para aplicar este nuevo algoritmo, el primer paso consiste en utilizar el Análisis de Componentes Principales (PCA). Los componentes principales representan las direcciones en las que los datos muestran mayor variabilidad, y cada uno captura una parte de la información total. Esta técnica proyecta los datos originales en un espacio de menor dimensión, reduciendo así la cantidad de características correlacionadas, pero manteniendo la mayor parte de la varianza (Sharma, 2024). Gracias a ello, el modelo trabaja con variables no correlacionadas que reflejan distintos aspectos de la variabilidad del conjunto

Tras la normalización se aplica PCA con n_components=0.95 para conservar el 90% de la varianza de nuestros datos originales. Se imprime también la dimensión resultante, para comprobar que es suficientemente grande como para poder representar luego los resultados.

```
pca = PCA(n_components=0.95)
X_pca = pca.fit_transform(X_scaled)

print("Dimensión original:", X_scaled.shape)
print("Dimensión tras PCA:", X_pca.shape)
```

Para el valor de los parámetros, el procedimiento es el ya introducido, 'Elbow method', se ejecuta el siguiente código:

```
print("Datos cargados:", X.shape)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
#(gráfico k-distance)
k = 4
neighbors = NearestNeighbors(n_neighbors=k)
neighbors_fit = neighbors.fit(X_scaled)
distances, indices = neighbors_fit.kneighbors(X_scaled)

# Ordenamos distancias de los k vecinos más cercanos
distances = np.sort(distances[:, k-1])

plt.figure(figsize=(8,5))
plt.plot(distances)
plt.title("Curva k-distancia (elige el 'codo' como épsilon)")
plt.xlabel("Muestras ordenadas")
plt.ylabel(f"Distancia al {k}-ésimo vecino más cercano")
plt.grid(True)
plt.show()
```

Tras esta ejecución se obtiene la siguiente gráfica de k-distancia (ilustración 27):

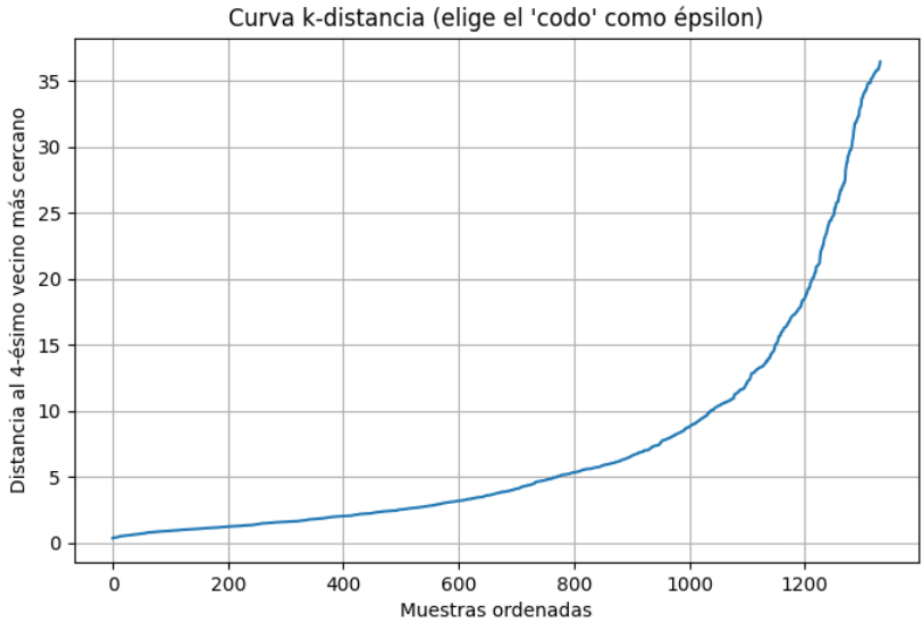

Ilustración 27: gráfica elección Épsilon

Con este resultado en la pantalla, la mejor elección de épsilon se sitúa entre 5 y 7, eps=6.

En vez de 58 pruebas, en este caso bastaría con 9, una por cada documento, pues las pruebas obtenidas antes eran el resultado de ir variando el número de clústeres en los que queríamos que nos dividiese los patrones. Sin embargo, como este algoritmo no requiere especificar el número de clústeres, realizar una prueba por documento es suficiente.

- no balanceado sin ruido: datos_procesados.csv
- balanceado y con 2000 datos de ruido: balanceo_2000ruido.csv
- balanceo y 200 datos de ruido: balanceo_200ruido.csv
- balanceado y con 2000 datos de ruido, sin el 6: balanceo_2000ruido_no6.csv
- balanceo y 200 datos de ruido, sin el 6: balanceo_200ruido_no6.csv
- balanceado, sin ruido: balanceo_sinruido.csv
- balanceado, sin ruido, sin el 6:balanceo_sinruido_no6.csv
- no balanceado 2000 ruido : nobalanceo_2000ruido.csv
- no balanceado 200 ruido : nobalanceo_200ruido.csv

Con los parámetros ya elegidos, el código a ejecutar sería el siguiente:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score,
davies_bouldin_score

rutas = [
    "/content/drive/MyDrive/TFG/resultados/procesados/reducidos/b
alanceo_200ruido_no6_pca.csv",
]

datos = [pd.read_csv(r, header=0).values for r in rutas]
X = np.vstack(datos)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

pca = PCA(n_components=0.95)
X_pca = pca.fit_transform(X_scaled)
print("Dimensión original:", X_scaled.shape)
```

```
print("Dimensión tras PCA:", X_pca.shape)

# Aplicar DBSCAN
eps = 6
min_samples = 3
dbscan = DBSCAN(eps=eps, min_samples=min_samples)
clusters = dbscan.fit_predict(X_scaled)

n_clusters = len(set(clusters)) - (1 if -1 in clusters else 0)
n_noise = list(clusters).count(-1)

print(f"Número de clusters: {n_clusters}")
print(f"Número de puntos ruido: {n_noise}")
```

Obteniendo tras la ejecución:

```
Número de clusters: 10
Número de puntos ruido: 422
```

Resultados

3.1 K-Means

Con el objetivo de averiguar qué tan buenos son los resultados obtenidos tras la aplicación de K-Means, dado que se dispone de una clasificación manual previa, habría que comparar dichas clasificaciones.

Se van a evaluar dos métricas: Adjusted Rand Index (ARI) es una métrica que cuantifica la similitud entre dos particiones de un mismo conjunto de datos, comparando la clasificación obtenida mediante un algoritmo de clústering con una clasificación de referencia (etiquetas reales). Esta medida se basa en el recuento de pares de elementos que coinciden en su asignación (ya sea en el mismo grupo o en grupos distintos, por lo que no nos afecta que el algoritmo haya nombrado los clústeres diferente a los patrones hallados) en ambas particiones, y ajusta este valor corrigiendo el acuerdo esperado por azar (Vinh, 2010). El ARI varía en un rango de -1 a 1, donde 1 indica una coincidencia perfecta, 0 refleja una asignación equivalente al azar y valores negativos sugieren un acuerdo inferior al esperado aleatoriamente. Por otro lado, la Normalized Mutual Information (NMI) mide la información compartida entre las dos particiones. Normalizada entre 0 y 1, alcanza el valor 1 cuando las particiones son idénticas y 0 cuando no existe relación alguna. La NMI resulta útil en escenarios donde el número de clústeres difiere entre las particiones, pues se centra en la dependencia estadística entre ambas.

```
from sklearn.metrics import adjusted_rand_score, normalized_mutual_in
fo_score
from scipy.optimize import linear_sum_assignment
from sklearn.metrics import confusion_matrix
import numpy as np

def calcular_metricas_clustering(etiquetas_reales, etiquetas_clusters
):
    Calcula las métricas ARI y NMI para evaluar el clustering
    # Calcular ARI
    ari = adjusted_rand_score(etiqueta_real, etiqueta_cluster)
    # Calcular NMI
    nmi = normalized_mutua_info_score(etiqueta_real,etiqueta_cluster)

    return ari, nmi
```

Tras la ejecución anterior se obtiene el resultado:

```
Adjusted Rand Index (ARI): 0.1541
Normalized Mutual Information (NMI): 0.2401
```

Los resultados, Adjusted Rand Index (ARI) de 0.1541 y Normalized Mutual Information (NMI) de 0.2401 indican que la división por clústeres que ha hecho el algoritmo K-Means no coincide bien con los patrones existentes. El valor de ARI cerca de 0 sugiere que la clasificación es casi aleatoria, y el NMI bajo confirma que hay poca información compartida entre los clústeres del algoritmo y los patrones.

Por otro lado, la clasificación de la que se dispone es casi en su totalidad visual, por lo que parece lógico que un componente de la evaluación se centre también en una representación gráfica. Es por eso que se lleva a cabo una impresión de varios datos de cada clúster, para evaluar sus relaciones.

Tras la ejecución de el siguiente código:

```
def plot_cluster_samples(data, clusters, n_samples=5, title="Muestras
por Cluster"):

    unique_clusters = np.unique(clusters)

    n_clusters = len(unique_clusters)

    n_cols = min(n_samples, 5)
    n_rows = int(np.ceil(n_clusters))

    fig, axes = plt.subplots(n_rows, n_cols, figsize=(15, 3*n_rows))

    if n_rows == 1 and n_cols == 1:
        axes = np.array([[axes]])
    elif n_rows == 1:
        axes = axes.reshape(1, -1)
    elif n_cols == 1:
        axes = axes.reshape(-1, 1)

    axes_flat = axes.flatten()

    current_axis = 0

    for cluster_id in unique_clusters:

        cluster_indices = np.where(clusters == cluster_id)[0]
```

```
        if len(cluster_indices) > 0:
            sample_indices = np.random.choice(cluster_indices,
                in(n_samples, len(cluster_indices)), replace=False)


            for j, idx in enumerate(sample_indices):

                if current_axis < len(axes_flat):
                    ax = axes_flat[current_axis]
                    ax.plot(data[idx], linewidth=2, alpha=0.8,
color=f'C{cluster_id % 10}')

                    ax.set_title(f'Cluster {cluster_id}\nMuestra
{idx}', fontsize=10)
                    ax.set_ylim(-3, 3)
                    ax.grid(True, alpha=0.3)
                    current_axis += 1

    for j in range(current_axis, len(axes_flat)):
        axes_flat[j].set_visible(False)

    plt.suptitle(title, fontsize=16, y=0.95)
    plt.tight_layout()
    return fig
```

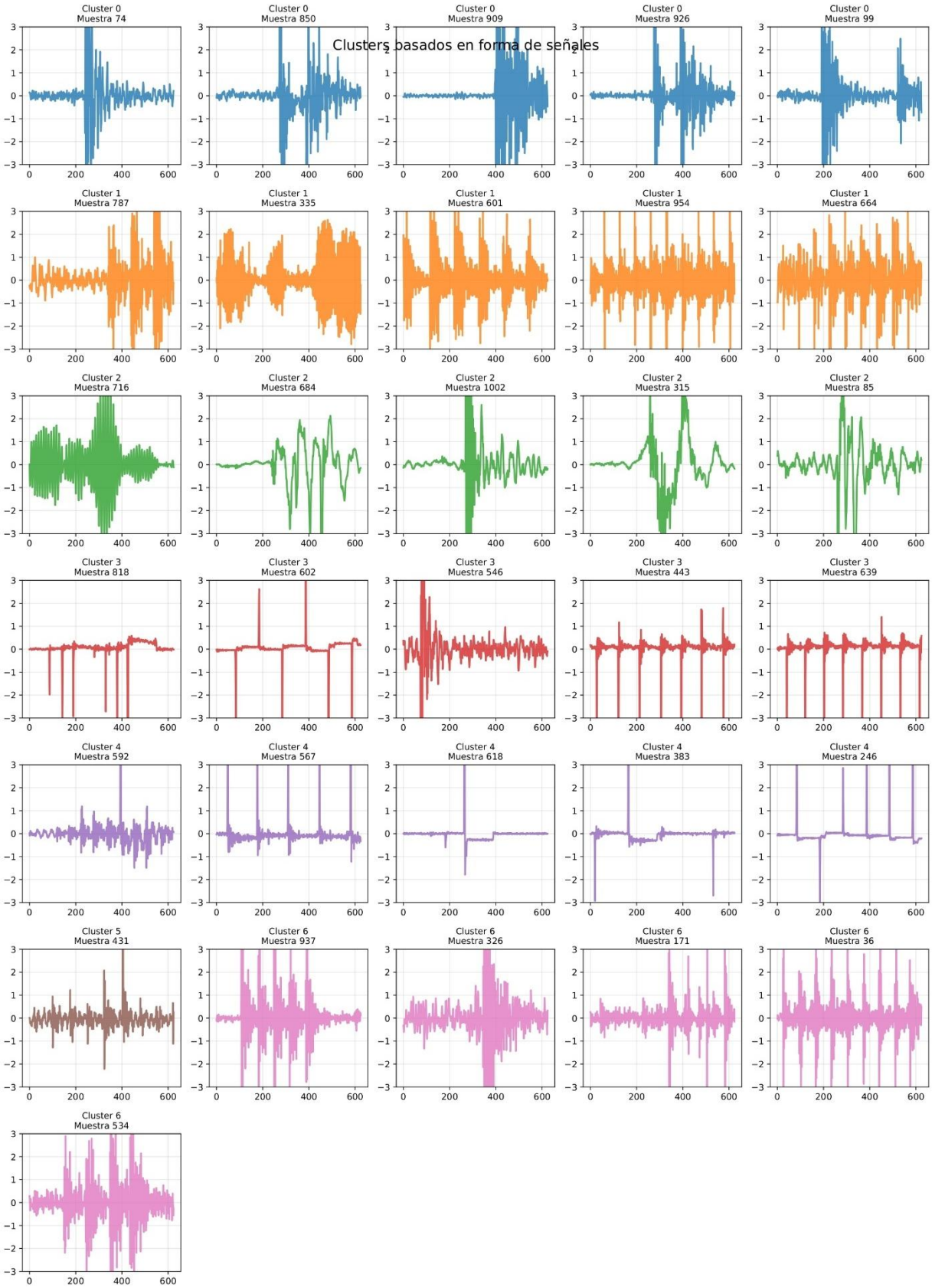

Ilustración 28: Resultado de la ejecución

Los resultados visuales parecen menos malos, se intuye algún criterio parecido al empleado en los patrones para dividir los clústeres, los picos negativos en el clúster 3 o la forma de onda irregular en el clúster 2. El algoritmo ha agrupado señales que se parecen visualmente, pero que pueden pertenecer a patrones. El Clúster 0 parece agrupar el Patrón A (picos regulares) con otros patrones de picos, ignorando la periodicidad. El Clúster 1 contiene señales con una forma de "campana", lo que podría ser el Patrón B, pero también incluye otras señales que no encajan del todo. De manera similar, los Clústeres 4 y 5, parecen agrupar patrones de picos negativos o amortiguados, pero sin diferenciar la periodicidad o el decrecimiento de la amplitud que definiste en los Patrones G y H.

Se realizaron diversas modificaciones en el preprocesamiento de las señales con el objetivo de mejorar la capacidad del algoritmo KMeans para recuperar los patrones reales.

Inicialmente, se aplicó una normalización z-score por fila, inmediatamente después, una normalización de rango (0,1) y un centrado de cada fila para intentar aproximar la correlación entre señales. También, se implementó un suavizado mediante media móvil intentando reducir el ruido y conservando la forma general de las señales. Estas estrategias pretenden que la medida utilizada por KMeans reflejara la similitud en la forma de las señales, y no solo la magnitud absoluta.

```
# 1. Normalización z-score por fila
X_zscore = (X - X.mean(axis=1, keepdims=True)) / X.std(axis=1,
keepdims=True)

# 2. Normalización al rango [0,1]
scaler = MinMaxScaler()

X_0_1 = np.array([scaler.fit_transform(row.reshape(-1,1)).flatten()
for row in X])

# 3. Suavizado con media móvil
X_smooth = np.array([pd.Series(row).rolling(window=3,
min_periods=1, center=True).mean().values for row in X])

# 4. Centrado de cada fila para aproximar correlación
X_centered = X_0_1 - X_0_1.mean(axis=1, keepdims=True)
```

Además, se estimaron algunos cambios en el algoritmo de agrupamiento: variando el número de grupos y ajustando el valor n_init para una inicialización más estable.

También se incluye MDS (escalamiento multidimensional), un método de reducción dimensional que puede manejar métricas basadas en correlaciones.

Estos puntajes de ARI y NMI se mantuvieron bajos, a pesar de estos ajustes. Debido a la complejidad, K-Means no logra capturar los patrones reales presentes en las señales.

```
# variación de n_clusters y n_init

for n_clusters in range(5, 11):
    kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init=50)

    y_pred = kmeans.fit_predict(X_preprocesado)

    print(f"n_clusters={n_clusters}")

# MDS para aproximar métricas

cor_matrix = np.corrcoef(X_preprocesado)

dist_matrix = 1 - cor_matrix  # distancia basada en correlación

mds = MDS(n_components=X_preprocesado.shape[1],
dissimilarity='precomputed', random_state=42)

X_mds = mds.fit_transform(dist_matrix)

# KMeans sobre representación MDS

kmeans_mds = KMeans(n_clusters=7, random_state=42, n_init=50)

y_pred_mds = kmeans_mds.fit_predict(X_mds)
```

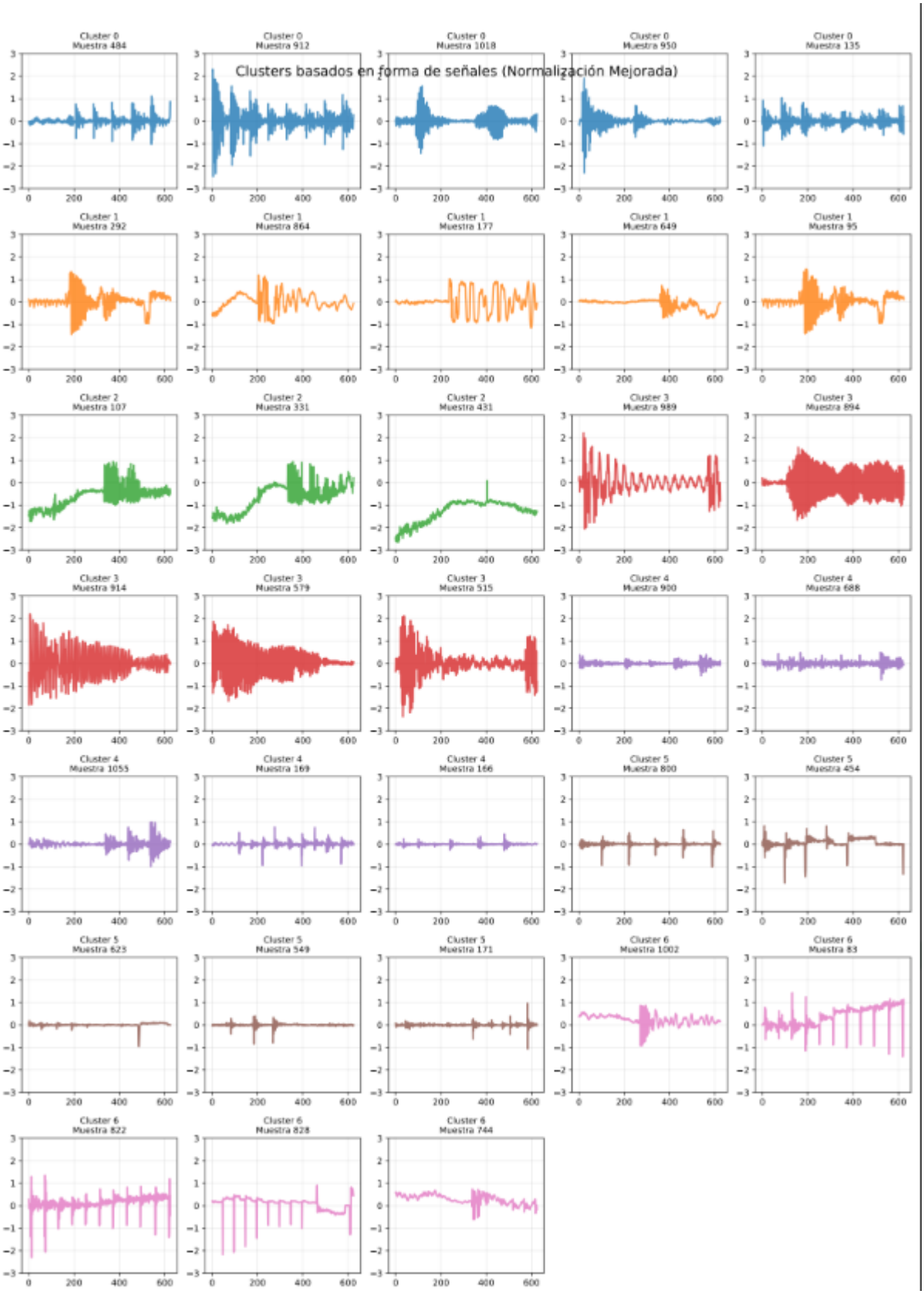

Ilustración 29:Nuevos resultados

A pesar de la aplicación de lo que pretendían ser diversas mejoras en el preprocesamiento de las señales y en la configuración del algoritmo KMeans (incluyendo normalización z-score, escalado de rango, suavizado mediante media, variación del número de clústeres…), los resultados obtenidos no muestran una mejora significativa. Los valores de ARI (0.1104) y NMI (0.1701) permanecen bajos, indicando que la clasificación generada por el algoritmo continúa siendo apenas superior al azar comparando con los patrones reales. Visualmente, los clústeres muestran cierta agrupación por similitud de forma de onda; por ejemplo, el Clúster 0 agrupa señales con picos relativamente regulares, lo que podría aproximarse al Patrón A, mientras que el Clúster 3 contiene formas de "campana" que recuerdan al Patrón B. Sin embargo, la correspondencia es incompleta: los Clústeres 2 y 1 incluyen señales con picos crecientes o decrecientes (Patrones C y D) junto con otras formas que no encajan claramente, y los Clústeres 4 y 5 muestran agrupaciones de picos negativos o amortiguados, similares a los Patrones G y H, pero sin distinguir correctamente la periodicidad ni la dinámica de amplitud. Estos resultados confirman que, la clasificación automática no logra reproducir con fidelidad los patrones definidos manualmente.

Dado que K-Means no logra capturar adecuadamente los patrones originales, otra posible causa es la alta dimensionalidad de las señales y la presencia de características redundantes o ruido que dificultan la detección de estructuras por parte del algoritmo. Por ello, el siguiente paso será aplicar Análisis de Componentes Principales (PCA) para reducir la dimensionalidad de los datos, eliminando información menos relevante.

La maldición de la dimensionalidad (Awan, 2023) se refiere a los problemas que surgen cuando se trabaja con datos de alta dimensión. A medida que el número de dimensiones aumenta, los puntos de datos se vuelven muy dispersos y la distancia euclidiana pierde sentido como medida de similitud, haciendo que algoritmos basados en distancia, como K-Means, se comporten peor.

Para intentar mitigar estos inconvenientes, el siguiente paso es aplicar técnicas de reducción de dimensionalidad, entre las cuales el PCA es uno de los más empleados. Este método transforma los datos originales en un nuevo conjunto de variables ortogonales, denominadas componentes principales, que concentran la mayor parte de la varianza en menos dimensiones. De este modo, se conserva la información más relevante y se eliminan características redundantes o ruido, lo que facilita el trabajo de los algoritmos de clústering y mejora la interpretación de los datos (Sharma, 2024).

En términos generales, el uso de PCA ayuda a minimizar los efectos de la maldición de la dimensionalidad, optimiza la eficiencia y puede incrementar la calidad del clústering. Resulta especialmente útil cuando los datos presentan muchas variables correlacionadas o con alto contenido de ruido, pues permite concentrar la información importante en menos dimensiones.

Se ejecuta el siguiente código:

```
from sklearn.decomposition import PCA

def aplicar_pca(X, n_componentes=10):

    """Aplica PCA para reducción de dimensionalidad"""
    pca = PCA(n_components=n_componentes, random_state=42)

    X_pca = pca.fit_transform(X)

    print(f"Varianza explicada acumulada por {n_componentes}
componentes: {np.sum(pca.explained_variance_ratio_):.4f}")

    return X_pca

# Reducir dimensionalidad con PCA

n_componentes = 10  # Ajustable según tus datos

X_pca = aplicar_pca(X_norm, n_componentes=n_componentes)

# KMeans sobre datos reducidos

n_clusters = 7
kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init=20)
y_pred_pca = kmeans.fit_predict(X_pca)
```

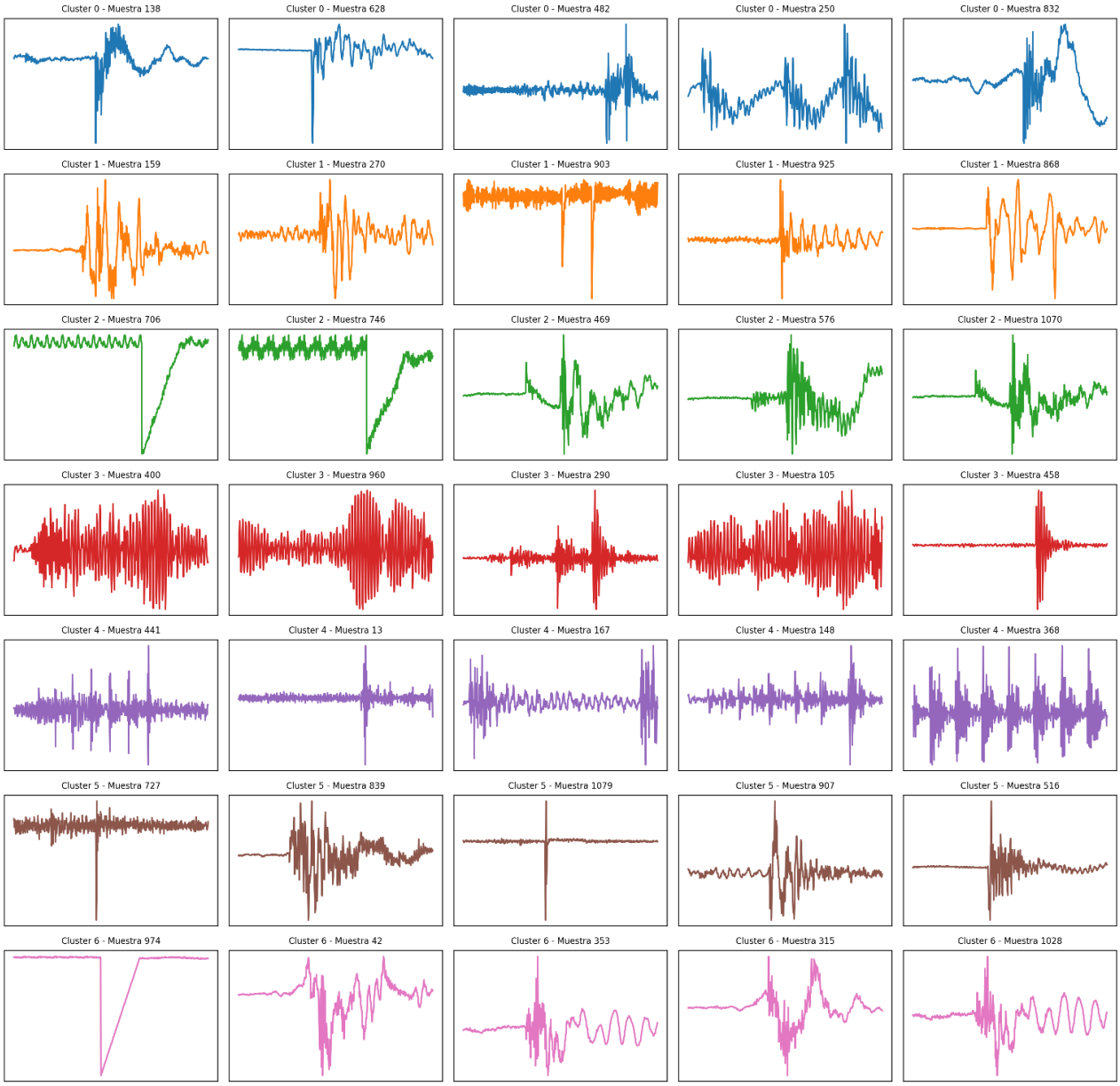

Ilustración 30:Resultados tras PCA

Tras aplicar PCA y realizar el clústering sobre el espacio reducido, se observa (Ilustración 29) que los grupos formados tienden a diferenciarse principalmente por la amplitud de las señales. El Clúster 0 recoge muestras con amplitudes moderadas, aproximándose al Patrón E, mientras que el Clúster 4 presenta cierta similitud con el Patrón A. En el resto de clústeres resulta difícil encontrar una equivalencia con los patrones designados de partida, agrupan señales diversas y en su interior podrían reconocerse subgrupos. En cualquier caso, la correspondencia es parcial y los patrones definidos manualmente no aparecen claramente diferenciados.

Las limitaciones son evidentes: los patrones de amplitud creciente o decreciente (C y D) parecen mezclarse, y aspectos más sutiles como la periodicidad regular (A) o la forma de campana (B) no logran separarse adecuadamente. En consecuencia, aunque PCA debería facilita cierta organización y mejorar la eficiencia, el clustering resultante sigue agrupando principalmente por nivel de amplitud y pierde información temporal relevante, lo que se ve reflejado en las métricas calculadas anteriormente, Adjusted Rand Index (ARI) y Normalized Mutual Information (NMI), que en este caso son incluso peores que en las ejecuciones anteriores:

Adjusted Rand Index (ARI): 0.0268

Normalized Mutual Information (NMI): 0.1497

En los resultados presentados, se muestra únicamente la ejecución de K-Means con k = 7, correspondiente al número de patrones definidos en el archivo balanceo_200ruido_no6.csv. Esta elección se justifica porque, tras evaluar de manera preliminar múltiples ejecuciones y pruebas (un total de 58 pruebas), las métricas obtenidas resultaron en todos los casos muy deficientes. Se seleccionó esta ejecución porque se consideró que era la configuración con mayor probabilidad de ofrecer los mejores resultados dentro de las pruebas realizadas. Por ello, y para optimizar el tiempo de análisis, no se realizaron evaluaciones visuales detalladas de cada prueba adicional, centrándose únicamente en la configuración más representativa.

3.2 DBScan

Para evaluar la calidad de los resultados obtenidos mediante DBSCAN, se recurre también a la comparación con la clasificación manual disponible. Se utilizan las mismas métricas que en el caso de K-Means: el Adjusted Rand Index (ARI) y la Normalized Mutual Information (NMI). Estas medidas permiten cuantificar la similitud entre los clústeres identificados por DBSCAN y los patrones de referencia, teniendo en cuenta que DBSCAN no requiere especificar el número de clústeres y puede generar puntos clasificados como ruido. Un ARI cercano a 1 indicaría una coincidencia perfecta, mientras que valores próximos a 0 reflejan un acuerdo cercano al azar; de forma similar, un NMI elevado mostraría alta información compartida entre ambas particiones.

```
# Ajuste Rand Index y NMI
ari = adjusted_rand_score(y_true, clusters)
nmi = normalized_mutual_info_score(y_true, clusters)

print(f"Adjusted Rand Index: {ari:.3f}")
print(f"Normalized Mutual Information: {nmi:.3f}")
```

Obteniendo como salida:

```
Adjusted Rand Index: 0.031
Normalized Mutual Information: 0.089
```

Los resultados obtenidos, ARI y NMI, son todavía inferiores a los obtenidos con K-Means. Estos valores cercanos a cero muestran que la partición generada por DBSCAN no coincide con los patrones de referencia en absoluto. En comparación con K-Means, cuya correspondencia con los patrones era baja pero ligeramente superior, DBSCAN no logra reproducir de manera significativa la clasificación manual existente, demostrando que la estructura de densidades diseñada por el algoritmo no coincide en absoluto con los patrones de partida. Como en las ejecuciones anteriores, se generan representaciones gráficas de varias muestras por clúster, lo que facilita una evaluación visual de la cohesión y separación de los grupos detectados por el algoritmo.

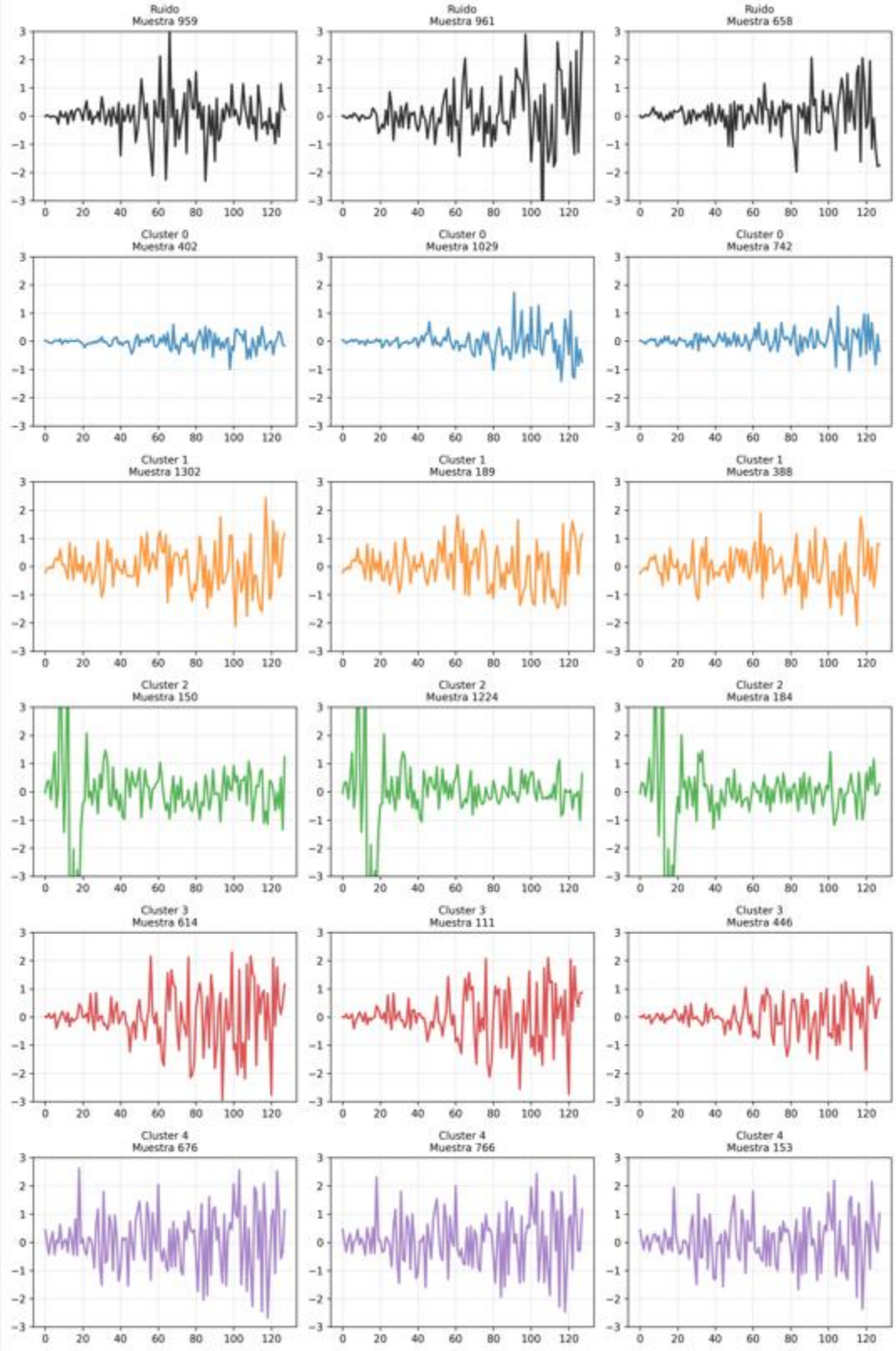

Ilustración 31:Resultados DBScan (1)

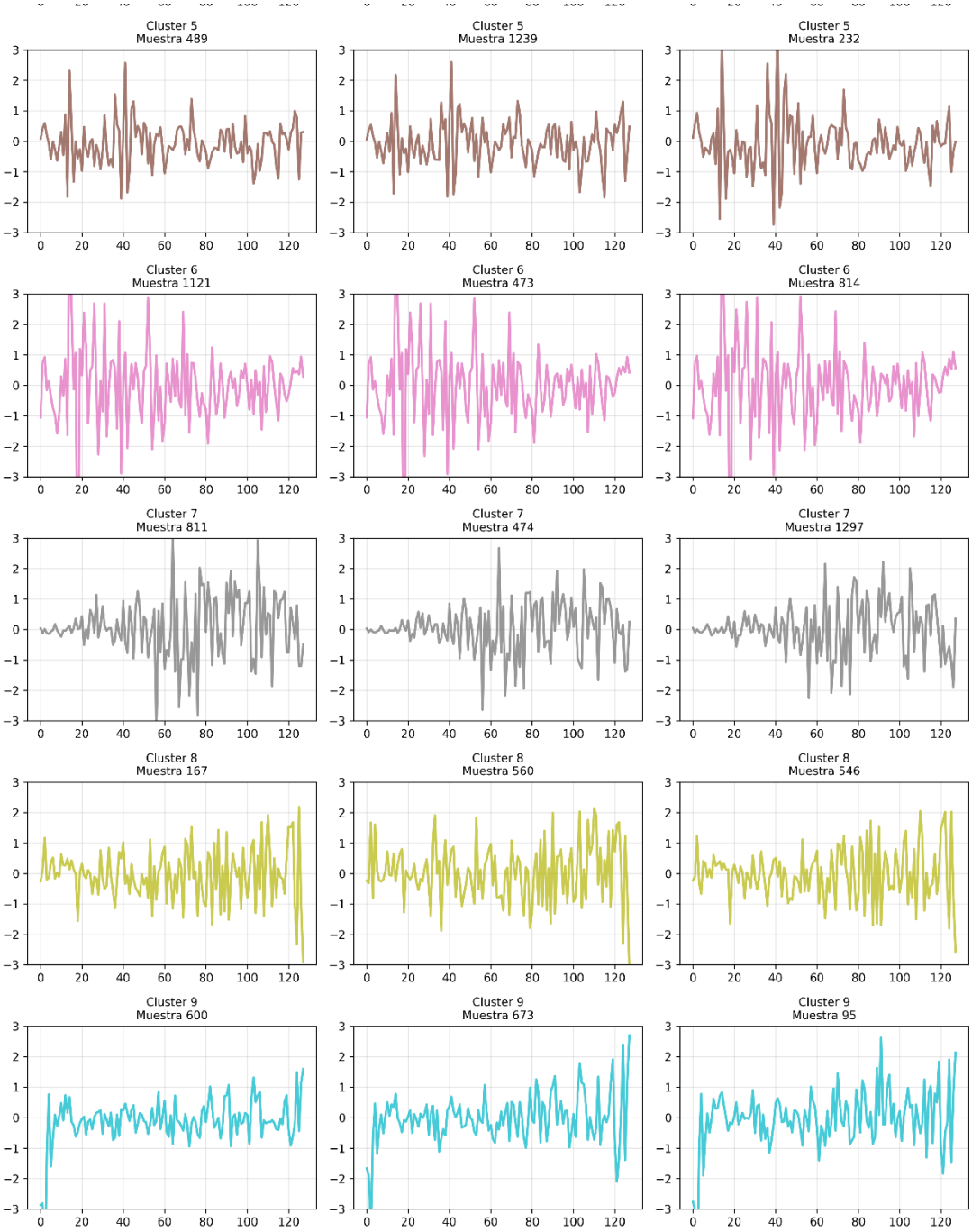

Ilustración 32: Resultados DBScan (2)

Los resultados visuales obtenidos con DBSCAN confirman lo que mostraban las métricas: el algoritmo no logra reproducir los distintos patrones definidos al inicio. A diferencia de K-Means, no se puede establecer una relación clara con los patrones originales, puesto que han sido reducidos con PCA, esa comparación ya no tiene mucho sentido. Además, DBSCAN genera un número de clústeres distinto al de los patrones, lo que impide un mapeo directo. Mientras que, con K-Means era posible reconocer algunos patrones dentro de los clústeres, en esta nueva prueba resulta más difícil, los datos se han vuelto más confusos y la forma de las gráficas no es tan suave. Esto evidencia que DBSCAN no funciona bien con este tipo de datos cuando se busca encontrar correspondencias entre los clústeres y clasificaciones previas.

Conclusiones y líneas futuras

4.1 Conclusiones

Los resultados obtenidos después de ejecutar K-Means con los datos, muestran un bajo grado de concordancia con los patrones reales de las señales, como evidencian los valores reducidos de ARI y NMI. Como intento de mejora, se aplican estrategias de preprocesamiento, normalización por fila, escalado al rango, centrado de cada señal y suavizado mediante media móvil, pero los clústeres generados siguen sin reflejar adecuadamente las categorías originales. K-Means presenta limitaciones que no se pueden evitar cuando se aplica a series temporales o señales con formas de onda complejas, donde las agrupaciones se definen más por la estructura de la señal que por las distancias euclidianas.

También se realizaron comprobaciones adicionales variando el número de clústeres, incrementando el parámetro de inicializaciones (n_init) y reduciendo la dimensionalidad mediante PCA. Ninguna de estas modificaciones resultó en mejoras en la capacidad del algoritmo para recuperar los patrones reales. Estas observaciones sugieren que el bajo desempeño tampoco se debe a problemas de configuración o preprocesamiento, sino a la incapacidad fundamental de K-Means para capturar similitudes basadas en la forma de las señales.

En paralelo, los resultados obtenidos mediante DBSCAN reflejan aún mayores dificultades para reproducir los patrones originales. Las métricas externas, con valores de ARI y NMI cercanos a cero, indican una casi nula correspondencia entre los clústeres detectados y las categorías de referencia, mientras que las gráficas de muestras por clúster confirman la ausencia de una estructura interna a distinguir de manera visual. Además, el hecho de que DBSCAN genere un número de clústeres distinto al de los patrones y clasifique un elevado número de muestras como ruido dificulta la interpretación y la comparación. Estos resultados sugieren que, al igual que K-Means, DBSCAN también presenta limitaciones para este tipo de señales, particularmente cuando las características relevantes se basan en la forma de la señal.

4.2 Líneas futuras

Como posibles líneas de trabajo futuro, se propone aplicar nuevos algoritmos al conjunto de señales analizado, para evaluar si pueden mejorar los valores de ARI y NMI en comparación con K-Means y DBScan. Además, sería interesante combinar estas técnicas con métodos de

reducción de dimensionalidad, como PCA, para mantener la eficiencia cuando se trabaja con conjuntos de datos grandes.

Por las características de los datos, algoritmos diseñados específicamente para series temporales o formas de onda podrían obtener mejores resultados. Entre ellos, mencionar KShape y DTW-KMeans**,** que utilizan medidas de similitud basadas en la forma y la dinámica temporal de las señales, como la distancia dinámica temporal (Dynamic Time Warping, DTW), en lugar de la distancia euclidiana estándar (Paparrizos, 2015). Estos métodos permiten agrupar señales que tienen patrones similares, aunque estén desplazadas o escaladas, ofreciendo una correspondencia más fiel con las categorías reales. La implementación de estos algoritmos representa una línea de trabajo futura que podría mejorar de manera significativa la capacidad de clustering sobre este tipo de datos.

Otra línea de investigación importante sería extracción de características basadas en el dominio de la señal, que podrían mejorar la representación de la forma de la señal y, por tanto, la calidad del clústering. Por otro lado, la forma de evaluar los resultados podría también ser una línea futura, quizás otro tipo de gráficas o sobreposición de ondas del mismo clúster.

4.3 Objetivos de Desarrollo Sostenible

Los Objetivos de Desarrollo Sostenible (ODS) (ONU, 2025), son un conjunto de 17 metas globales adoptadas por todos los Estados miembros de las Naciones Unidas en 2015 como parte de la Agenda 2030 para el Desarrollo Sostenible. Estos objetivos constituyen un llamado universal a la acción para erradicar la pobreza, proteger el planeta y garantizar que todas las personas gocen de paz y prosperidad para el año 2030.

Estos objetivos están interconectados y reconocen que la acción en un área afectará los resultados en otras, por lo que el desarrollo debe equilibrar la sostenibilidad social, económica y ambiental. La Agenda 2030 representa un compromiso global para mejorar la vida de las personas y proteger el planeta, asegurando que nadie se quede atrás en el camino hacia un futuro sostenible.
El trabajo desarrollado puede asociarse de manera directa con varios Objetivos de Desarrollo Sostenible (ODS), al situarse en la intersección entre la innovación tecnológica, la preservación del patrimonio y la sostenibilidad urbana. En particular, destacan los siguientes:

- ODS 9: Industria, innovación e infraestructura. Este proyecto fomenta el uso de tecnologías innovadoras como el aprendizaje automático y los algoritmos de clústering aplicados a datos de vibración, lo que contribuye a construir infraestructuras más resilientes. La aplicación de estas técnicas no solo impulsa la investigación en el ámbito de la ingeniería civil y la conservación patrimonial, sino que también pretende

introducir sistemas de monitorización más precisos, capaces de detectar riesgos de forma temprana.

- ODS 11: Ciudades y comunidades sostenibles. La conservación de los mosaicos del centro SCOP supone un ejemplo concreto de cómo la tecnología puede contribuir a preservar el patrimonio cultural en entornos urbanos. Prolongar la vida útil de estructuras históricas mediante monitorización inteligente se alinea con un desarrollo urbano más sostenible, en contraste con estrategias de demolición y sustitución, que generan un mayor consumo de recursos y residuos.

- ODS 13: Acción por el clima. Una infraestructura patrimonial vigilada con tecnología resulta más resistente frente a los efectos del cambio climático y los eventos extremos (maquinaria en entornos urbanos). Asegurar la integridad estructural mediante sistemas inteligentes de detección y análisis constituye, por tanto, una medida de adaptación clave.

En conjunto, este trabajo muestra cómo la integración de la inteligencia artificial en la ingeniería y la conservación patrimonial puede aportar soluciones innovadoras que apoyen un desarrollo más sostenible y respetuoso con el entorno.

Ilustración 33: ODS

Referencias

Awan, A. A. (2023). *The Curse of Dimensionality in Machine Learning: Challenges, Impacts, and Solutions*.

Bisong, E. (2019). Google Colaboratory. En Building Machine Learning and Deep Learning Models on Google Cloud Platform. (pp. 59–64).

Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow.

Google. (2023). *Welcome to Colaboratory*. *Colab Research*. *https://colab.research.google.com/notebooks/intro.ipynb*.

Kareem Eltouny, M. G. (2023). Unsu- pervised Learning Methods for Data-Driven Vibration-Based Structural Health Monitoring: A Review.

Kavlakoglu, E. (2024). *What is k-means clustering? https://www.ibm.com/think/topics/k-means-clustering?*

Keita, Z. (2024). *Classification in Machine Learning: An Introduction*. Obtenido de DataCamp.

Kluyver, T. R.-K. (2016). Jupyter Notebooks – a publishing format for reproducible computational workflows. In Positioning and Power in Academic Publishing: Players, Agents and Agendas. págs. (pp. 87–90).

Kodinariya, T. M. (2013). Review on determining number of Cluster in K-Means Clustering. International Journal of Advance Research in Computer Science and Management Studies. 90-95.

Kumar, R. (2024). *https://www.datacamp.com/tutorial/dbscan-clustering-algorithm*.

Lorenzana, A., Villacorta, J. A., Val, L., & Izquierdo, A. (2024). Design of an Instant Vibration-Based Warning System and Its Operation during Relocation Works of Historic Facades. *Buildings*.

Michell, T. (1997). *Machine Learning*.

ONU. (2025). *Transformar nuestro mundo: La agenda 2030 para el desarrollo sostenible*.

Paparrizos, J. G. (2015). *K-shape: Efficient and Accurate Clustering of Time Series*.

Samuel, A. (1959). Some Studies in Machine Learning Using the Game of Checkers. *Research and Development*.

Sharma, A. (2024). *How to Do Principal Component Analysis (PCA) in Python*. Obtenido de DataCamp.

Vinh, N. E. (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. .

Wang, Z. C.-J. (2022). Unsupervised machine and deep Learning methods for structural damage detection: A comparative study.

Xia, Y. C. (2012). Temperature effect on vibration properties of civil structures: a literature review and case studies. J Civil Struct Health Monit 2. págs. 29-46.