eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Trabajo Fin de Máster

Máster Universitario en Ingeniería Computacional y Sistemas Inteligentes

Comparación entre modelos Shallow y Deep para la detección de anomalías

Borja Rey Perea

Dirección
Dr. Alberto Jiménez Cortadi
Unidad operativa Digital: Cores
TECNALIA
Dr. Iñaki Inza Cano
Ciencia de la computación e Inteligencia Artificial
Universidad del País Vasco (UPV/EHU)

13 de junio de 2022

Abstract

This document contains the process to complete the Master's Thesis, which consist in learning different One Class Classification models, from Shallow models, to Deep Learning models, to detect failures in a crane's axis. The data will be prepared in different ways in order to evaluate the performance of each technique: PCA, expert variables, automatic variables and images. The objective is to detect anomalies and to compare both models.

Resumen

Este documento recoge el proceso seguido para la realización del Trabajo de Fin de Máster, el cual consiste en aprender diferentes modelos de One Class Classification para la clasificación de fallos del eje de una grúa. Se emplearán tanto modelos Shallow como modelos de Deep Learning. Se trabajarán distintas formas de preparar los datos para evaluar el rendimiento de los modelos según la técnica aplicada: PCA, variables expertas, variables automáticas e imágenes. El objetivo será detectar anomalías y comparar los distintos modelos.

Laburpena

Dokumentu honek Master Tesia egiteko landu den prozesua jarraitzen du. Honen helburua, garabi baten ardatzaren akatsak sailkatzeko, One Class Classification modeloak ikastea da. Shallow modeloak zein Deep Learning modeloak erabiliko dira. Datuak prestatzeko modu desberdinak landuko dira, bakoitzaren etekina ebaluatzeko (PCA, datu jakitunak, datu automatikoak eta argazkiak). Xedea, anomaliak detektatu eta modelo ezberdinak konparatzea izango da.

Agradecimientos

Por encima de todo me gustaría agradecer a mis padres, no solo el haberme permitido realizar el máster, sino haberme impulsado a seguir formándome. Gracias por apoyarme y guiarme durante tantos años. También a mi hermana y al resto de mi familia, gracias por haber estado siempre ahí.

Por otro lado, quiero agradecer a Iñaki Inza, mi tutor en este proyecto, quien ha realizado un seguimiento total mostrando interés y ayudándome en lo que hiciera falta, siempre dispuesto. Gracias por la dedicación y cercanía.

Tampoco podía faltar Alberto Jiménez, mi supervisor en Tecnalia, quien ha tratado al becario como uno más. Gracias por ayudarme en todo lo posible, ser cercano e invitarme a ir los cafés. Nos seguiremos viendo por ahí. Gracias también al resto de compañeros con los que he pasado estos 4 meses.

Finalmente, gracias a Ainhoa, que a pesar de haber estado lejos este año me ha apoyado como siempre, y me ha permitido pasar unos largos fines de semana lejos de aquí. También tengo que agradecer a mis amigos, quienes logran que coja con ganas una semana normal, ya que sé, que al final de la misma, nos volveremos a juntar.

GRACIAS A TODOS.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

INTRODUCCIÓN

Una anomalía consiste en un cambio o desviación respecto a un comportamiento normal. La detección de anomalías o outliers, es una característica inherente al ser humano, que ha llevado a la supervivencia a lo largo de las distintas eras de maneras muy distintas; desde detectar alimentos en mal estado hasta localizar traidores o enemigos. Hoy en día sigue siendo una llamada de atención que ayuda a evitar problemas de todo tipo a diario. Es por ello que esta técnica se ha extrapolado a diferentes ámbitos y se han ido desarrollado distintas herramientas para detectar anomalías, como puede ser para detección de células tumorales, detección de fraudes bancarios o, como en el caso de estudio que se presenta, detección del deterioro para el mantenimiento de maquinaria.

La detección de outliers es una técnica que juega un papel fundamental en el mantenimiento de equipos industriales. Esto es debido a que este tipo de mantenimiento busca realizar el mantenimiento antes de que surja un fallo. Con la ayuda de distintos sensores y mediciones, se busca maximizar el tiempo entre reparaciones a la par que se minimiza el coste [14].

1.1 Motivación

Durante la jornada de presentación de Proyectos de Fin de Máster realizado en la Facultad de Informática de la UPV/EHU en Donostia, se presentaron un gran número de proyectos de diferentes empresas. En una de las presentaciones de Tecnalia, no se presentó un único proyecto, sino 5 de sus líneas de investigación.
Desde que lo vi, me resultó muy interesante la línea llamada “Deep Generative Models For Anomaly Detection”, por lo que me puse en contacto con el doctor

Alberto Jiménez para conocer algo más acerca de dicho proyecto.

Del proyecto, me gustó especialmente el hecho de trabajar en la detección de outliers, un tema que durante el transcurso del máster me pareció muy interesante. Esto sumado al hecho de poder crear modelos Deep, hizo que me decantara por este proyecto. La sorpresa fue que además, esta detección de outliers debía lograrse empleando métodos de clasificación One Class, lo cual me pareció una forma perfecta de aprender sobre dichos modelos.

Por otra parte, siendo Tecnalia uno de los centros de investigación más grandes y punteros, hizo que me sintiera seguro al escoger esta opción como Proyecto de Fin de Máster.

1.2 Tecnalia

Este Proyecto de Fin de Máster ha sido realizado en Tecnalia. Tecnalia es el mayor centro de investigación aplicada y desarrollo tecnológico de España, un referente en Europa y miembro de Basque Research and Technology Alliance (BRTA). Su misión es transformar investigación tecnológica en prosperidad; la investigación de Tecnalia tiene un impacto real en la sociedad y genera beneficios en forma de calidad de vida y progreso. Trabajan con el propósito de construir un mundo mejor a través de la investigación tecnológica y la innovación. Concretamente, el proyecto se ha desarrollado en la Unidad Operativa Digital, en el área denominada Cores.

1.3 Facultad de Informática, UPV/EHU

Asimismo, el Trabajo de Fin de Máster ha sido realizado como aplicación en un proyecto real de los conocimientos adquiridos durante el máster "Ingeniería Computacional y Sistemas Inteligentes"de la UPV/EHU, cursado en la Facultad de Informática.

Dicha facultad está localizada en Guipuzkoa y dispone de una larga trayectoria investigadora. En el año 2020 diecisiete grupos de investigación trabajaban en 95 proyectos de investigación y en ellos se presentaron 14 tesis doctorales.

1.4 Objetivos

El objetivo principal de este Trabajo de Fin de Máster, es crear un modelo aplicable al mundo real, que ayude a mejorar y facilitar el mantenimiento del eje de una grúa. Mediante dicho modelo, se podrá inferir el estado del eje, y en base al grado de anomalía predicho, detectar modos de fallo.

Para ello se han aprendido distintos modelos y comparado para comprobar cual de ellos realiza mejor la tarea de detección de outliers. Por un lado tenemos modelos tradicionales (Support Vector Machine, Isolation Forest, Local Outlier Factor), para cuyo aprendizaje se han seleccionado varias características de los datos manualmente. Por otro lado, se ha aprendido un modelo de Deep Learning en el que se introducirán directamente los datos, sin realizar ninguna extracción de características previa, esperando que el modelo realice dicha tarea de extracción de características (Feature Extraction).

En los siguientes capítulos se muestran el estado del arte, donde se explican los distintos tipos de mantenimiento y los diferentes modelos utilizados.

Capítulo 2

ESTADO DEL ARTE

En este apartado se revisa la literatura existente relativa al mantenimiento industrial y a la clasificación de fallos y sus aplicaciones mediante técnicas de Machine Learning, tanto en modelos Shallow como en modelos profundos o Deep. También se revisará documentación acerca de metodologías de One Class Classification.

2.1 Mantenimiento

El mantenimiento en ingeniería se refiere a cualquier acción técnica o administrativa, o una combinación de ambas, que permita mantener los activos de ingeniería, o devolverlos a sus estados funcionales [15]. Principalmente existen 3 tipos de mantenimiento, los cuales han ido evolucionando con el paso del tiempo a medida que la tecnología ha ido avanzando.

En los siguientes apartados se detallará cada uno de los tipos de mantenimiento en mayor profundidad, desde el mantenimiento correctivo, que consiste en arreglar los fallos según van surgiendo, hasta el mantenimiento predictivo, que consiste en predecir cuándo surgirá un fallo para evitar costes mayores, pasando por el mantenimiento preventivo, que consiste en realizar revisiones sistemáticas para tratar de evitar el fallo.

2.1.1 Mantenimiento correctivo

El mantenimiento correctivo se refiere a aquellas actividades realizadas en un sistema para devolverlo a la normalidad de su funcionamiento. Este tipo de mantenimiento se focaliza en identificar la causa de los fallos, y el mantenimiento no se realiza hasta que la máquina muestra el fallo [16]. Generalmente se realiza

en pequeños componentes cuyo recambio no supone un gran coste temporal ni económico[17]. Sin embargo, el hecho de tener que parar el sistema para realizar las reparaciones o sustituciones requeridas, aumenta los costes y produce pérdidas por inactividad, además de la posibilidad de producir otros daños en la cadena de producción.

2.1.2 Mantenimiento preventivo

El mantenimiento preventivo consiste en mantener las instalaciones y maquinarias en condiciones correctas para el funcionamiento mediante inspecciones sistemáticas para reducir la posibilidad de fallo de la maquinaria [18]. Además, al tratarse de inspecciones programadas, no se suelen realizar durante las jornadas de producción [17]. El hecho de detectar, prevenir y corregir los fallos incipientes antes de que ocurran o de que evolucionen en algo más grave supone una ventaja frente al mantenimiento correctivo y se corresponde con una disminución de los costes y aumento de la productividad [19].

Sin embargo, dichas revisiones sistemáticas suponen un coste, además de que no asegura la ausencia de fallos entre revisiones, lo que supondría un coste extra.

2.1.3 Mantenimiento predictivo

El mantenimiento predictivo consiste en determinar cuándo son necesarias las labores de mantenimiento mediante el empleo de herramientas predictivas. Se basa en la continua monitorización del proceso de una máquina, por lo que permite la temprana detección de fallos, permitiendo así actuar únicamente en caso de que sea necesario [20]. Para indicar el momento en el que se debe actuar se debe fijar un umbral en la probabilidad de fallo. La elección de dicho umbral en las situaciones reales se estimará teniendo en cuenta numerosos factores como la seguridad de los trabajadores, criticidad del fallo o factores económicos, por ejemplo [17].

Una vez la maquina ha sido monitorizada, se habrán recolectado una gran cantidad de datos. El siguiente paso consiste en preprocesar dichos datos para que se pueda trabajar con ellos. Este paso es vital ya que la correcta detección y eliminación de los datos erróneos y ruidosos mejora la calidad del dataset en su conjunto [21]. Esta calidad del dataset indica cómo de fieles son los datos a la hora de caracterizar el problema que se esté estudiando [22]. También se deben integrar de manera que los datos se encuentren en una estructura y formato intuitivos y con

los que se pueda trabajar correctamente.

Finalmente, resta implementar un sistema de toma de decisiones. En el caso del mantenimiento predictivo interesa la implementación de un sistema de prognosis, centrado en predecir los fallos antes de que se produzcan. Para ello se emplean técnicas de aprendizaje automático. Existe el sistema de diagnosis, que se centra en la detección, identificación y aislamiento de los fallos cuando se producen. El sistema de diagnosis es clave para hallar la relación entre los datos monitorizados y el estado de salud de la maquinaria [23].

2.2 Técnicas de preprocesamiento aplicadas al mantenimiento predictivo

2.2.1 t-SNE

El t-SNE, "t-distributed Stochastic Neighbor Embedding"[24] , es una novedosa técnica de visualización de datos de alta dimensionalidad. El objetivo, es que mediante transformaciones no lineales de las variables de un dataset de alta dimensionalidad, se logre crear una representación fiel de los datos en dos dimensiones. Gracias al hecho de que se realicen transformaciones no lineales, se pueden separar y agrupar datos que de otra forma no se podría.

Figura 2.1: Datos no separables linealmente [1]

Tratar de visualizar los datos mostrados en la Figura 2.1 realizando un Análisis de las Componentes Principales (PCA) carecería de sentido, ya que no son separables linealmente. Para esta tarea, y para comprender datasets de alta dimensionalidad, el t-SNE es una técnica realmente útil.

Funcionamiento

El primer paso que realiza el t-SNE es crear una distribución de probabilidad que represente la similitud entre los datos. Esta similitud es la probabilidad condicional de que el punto $\boldsymbol{x_j}$ escogiera al punto $\boldsymbol{x_i}$ como su vecino. [24]

$$p_{i|j} = \frac{exp(\frac{-||x_i - x_j||^2}{2\sigma_i^2})}{\sum_{k \neq i} exp(\frac{-||x_i - x_k||^2}{2\sigma_i^2})}$$

donde σ_i es la varianza de la distribución Gaussiana centrada en el punto x_i. Se calcula la distancia respecto a todos los puntos y se muestra en un único eje.

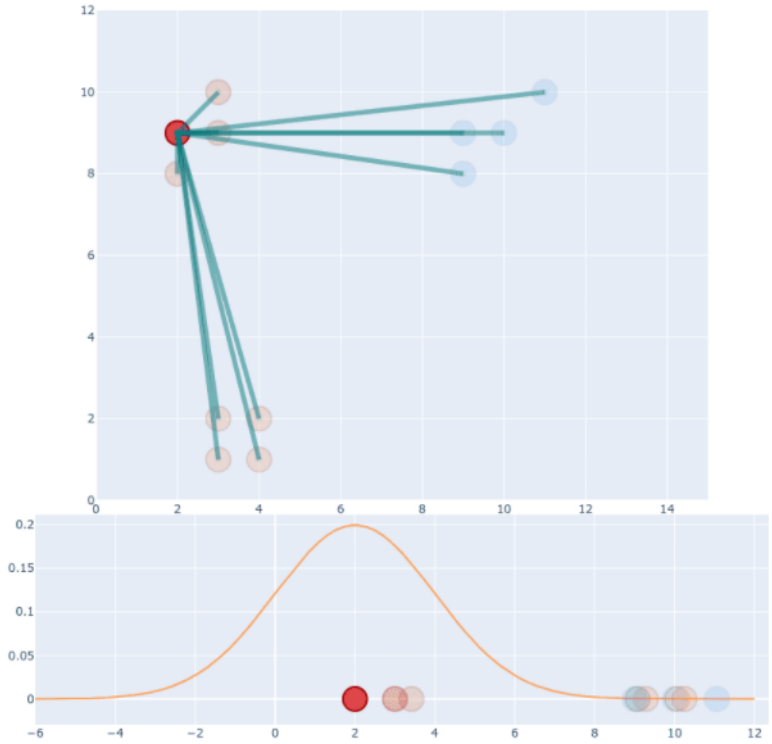

Figura 2.2: Distancia entre puntos [1]

A la hora de calcular la similitud entre dos puntos tomando como referencia el primero y después el segundo, el resultado es diferente. Esto es debido a que se calculan mediante dos distribuciones distintas, tal y como se ve en la Figura 2.3.

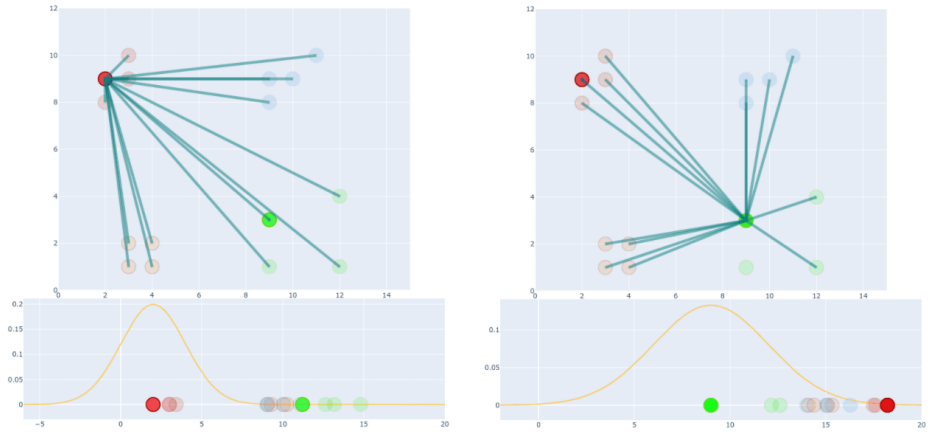

Figura 2.3: Diferencia entre similitudes [1]

Por ello, se toma como valor de similitud $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2N}$.

Perplejidad

El t-SNE es especialmente sensible a un parámetro llamado perplejidad. La perplejidad está directamente relacionada con la varianza de los valores. Básicamente, t-SNE realiza una búsqueda binaria para el valor de σ que produce una distribución de probabilidad con un valor de perplejidad especificada por el usuario. [24]

$$Perp(P_i) = 2^{-\sum p_{j|i} log_2 p_{j|i}}$$

Donde:

$-\sum p_{j|i} log_2 p_{j|i}$ es la entropía de Shannon.

Reducción de la dimensionalidad

El siguiente paso del algoritmo, consiste en reducir la dimensionalidad y tratar de mantener en dos dimensiones la distribución de probabilidad. Para ello, se utilizará la distribución *t* student. Esto es debido a que la cola de la distribución Gaussiana es demasiado corta, y puede provocar que varios puntos queden sola-

pados. La distribución *t* student es similar a la Gaussiana solo que tiene una cola más larga, lo que ayudará a solventar dicho problema.

En la Figura 2.4 podemos ver cómo se resolvería la visualización de los ejemplos mostrados en la Figura 2.1.

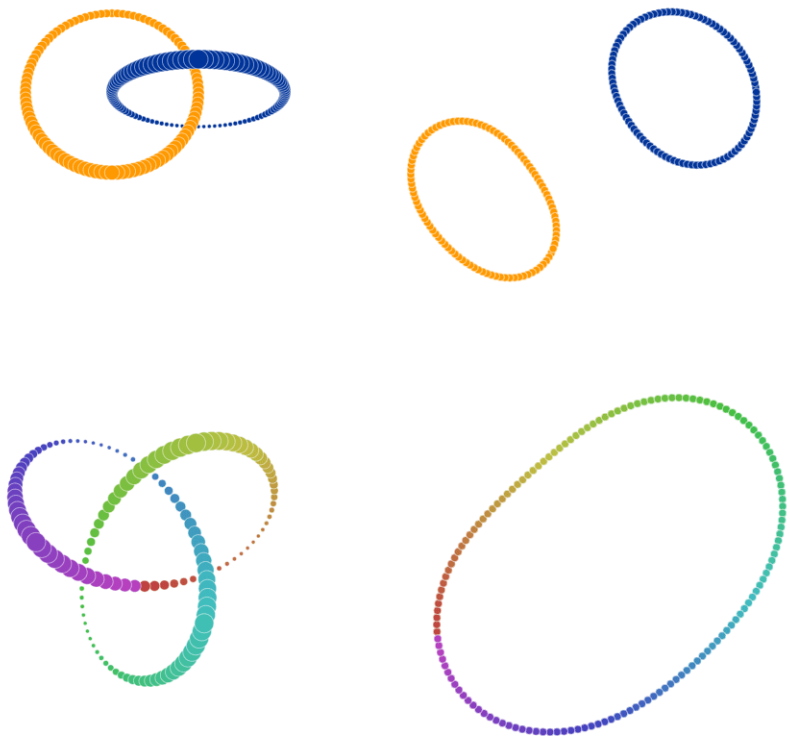

Figura 2.4: t-SNE aplicado a los ejemplos [1]

2.2.2 PCA

La PCA o Principal Component Analysis, es una técnica de reducción de la dimensionalidad. El objetivo de dicha técnica, es crear combinaciones lineales de las variables con el fin de simplificar espacios de alta dimensionalidad pero manteniendo toda la varianza posible. Cada una de estas combinaciones lineales se denomina Componente Principal, y cada uno de los Componentes Principales explica un porcentaje de la varianza explicada.

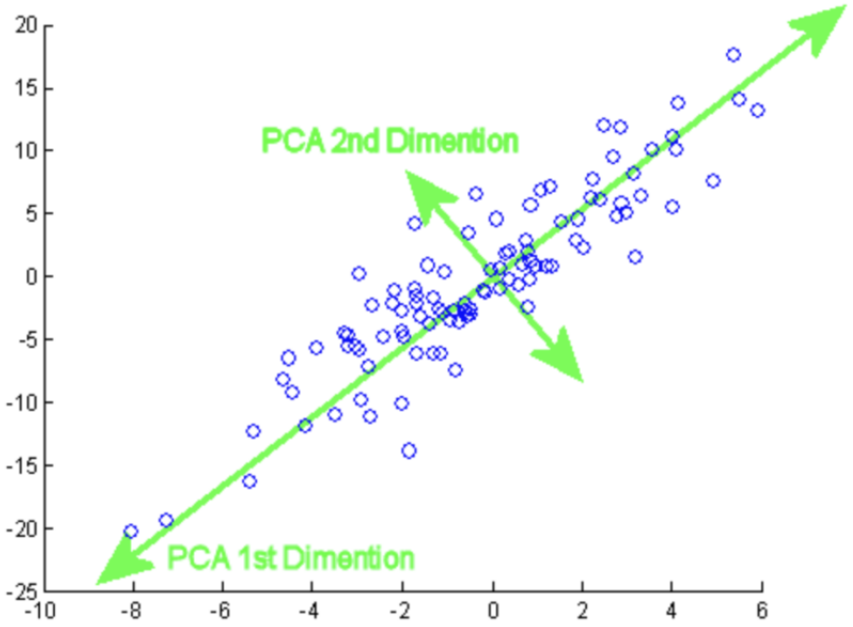

Figura 2.5: Componentes principales de un dataset

Respecto a la varianza explicada, es un índice que nos indica cómo de correctamente representada va a estar la información en cada Componente Principal, y para seleccionar el número de Componentes Principales, habitualmente se tiene en cuenta que se debe representar aproximadamente un 90 %-95 % de la varianza explicada, esto con el fin de que la información extraíble de la dimensión reducida sea lo más fidedigna posible respecto a la original.

2.3 Sistemas de toma de decisiones aplicadas al mantenimiento predictivo

2.3.1 One Class Classification

One Class Classification [25] es una técnica que consiste en la clasificación de muestras desconocidas habiendo entrenado el modelo con un único tipo de muestra, todas ellas con el mismo valor de etiqueta. Este tipo de clasificación es realmente útil especialmente en ocasiones donde los datos están tan desbalancea-

dos que se tienen muchas muestras de un tipo de datos pero extremadamente pocas de otro. Es decir, One Class Classification es realmente útil a la hora de detectar anomalías. Esto es debido a que al estar solo entrenado con muestras de un único valor de etiquetado, empleando técnicas tanto Shallow como Deep se puede obtener un valor que indica como de distintos son los valores de un nuevo caso respecto a los de muestras originales con las que se ha entrenado el modelo [26].

2.3.2 Técnicas Shallow

Este apartado ofrece un resumen y explicación de algunos de los algoritmos empleados habitualmente para la detección de anomalías.

One Class Support Vector Machines (OCSVM)

El objetivo de las Support Vector Machines es el de proyectar los datos en un espacio de mayor dimensionalidad que el del dataset mediante combinaciones no lineales. De este modo, datos de diferentes clases que no podrían ser separados mediante una hiperplano en el espacio original, sí que se pueden separar mediante una recta o hiperplano en el espacio de dimensionalidad superior. Así, al volver a proyectar los datos en el espacio original, obtendremos una proyección no lineal del hiperplano que separaba los datos.

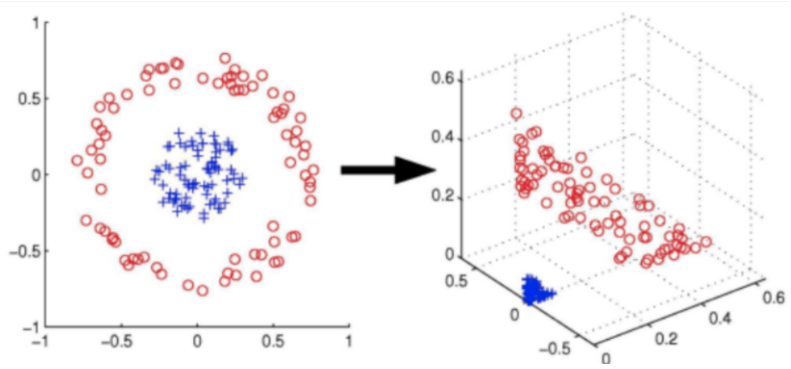

Figura 2.6: Transformación de los datos a un nuevo hiperespacio [2]

Al calcular el hiperplano que separa las clases en dos grupos, el algoritmo trata de maximizar la distancia entre los datos de las diferentes clases. Sin embargo,

a medida que las dimensiones del dataset aumentan, también lo hace la complejidad y coste computacional para el cálculo de las coordenadas de los datos en ese nuevo espacio. Es por ello que se usa el truco de Kernel o Kernel-trick. Éste nos permite operar con los datos en el espacio original sin la necesidad de calcular las coordenadas de los datos en el nuevo espacio de mayor dimensionalidad.

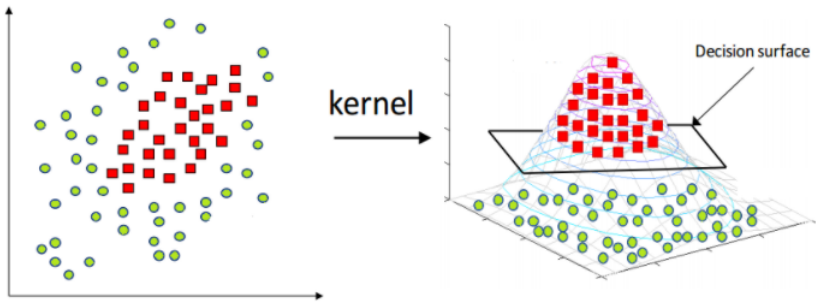

Figura 2.7: Corte del hiperplano en un nuevo hiperespacio [3]

Sin embargo, mientras que en las SVM comunes se busca un hiperplano que maximice la distancia entre las diferentes clases, en las One Class SVM esto no es posible, ya que únicamente existe un tipo de dato en el entrenamiento. Es por ello que se busca maximizar los datos respecto al punto de origen. Además, en lugar de buscar una distancia media entre ambos extremos, este hiperplano estará más cercano a los datos de entrenamiento [27].

Finalmente, al recibir nuevos datos, estos se proyectan en el hiperespacio creado, y dependiendo de si dicha proyección reside en la zona positiva del hiperplano será considerada un dato común o normal, en caso contrario será considerado un outlier [28].

Se ha probado que las SVM son buenas opciones para la clasificación One Class en diferentes situaciones. Se emplearon para comparar redes convolucionales, y aunque obtuvieron peores resultados, eran lo bastante buenos como para tenerlos en cuenta [25]. También han sido previamente empleadas para la clasificación de fallos de ejes similares a los presentados en este documento. El hecho de que obtuvieran buenos resultados empleando dicho algoritmo lo convierte en una opción a tener en cuenta [29].

Isolation Forest (IF)

Isolation Forest es un algoritmo no supervisado de detección de anomalías basado en árboles de decisión. El algoritmo consiste en aislar mediante rectas paralelas a los ejes en el plano cada dato. Una vez los datos han sido separados, se observa el número de cortes que se han realizado para aislar cada dato, siendo esto un indicador del grado de anomalía de cada dato.

Isolation Forest está compuesto por un conjunto de árboles de decisión binarios, en el que cada árbol de decisión es denominado Isolation Tree. Durante el entrenamiento, se escoge una muestra aleatoria de los datos a la que se le asigna un árbol de decisión.

La ramificación comienza seleccionando una variable aleatoria del dataset y un umbral a partir del cual aplicar la bifurcación. De este modo, los datos estarán separados en 2 zonas. Este proceso se repite hasta que cada punto ha sido aislado o hasta que se alcanza la profundidad máxima. Un ejemplo de este paso se puede observar en la Figura 2.8, donde se ven los árboles de decisión que separa cada dato. En la Figura 2.9 podemos ver las diferentes áreas que se forman. El color indica el grado de anomalía y cuanto más oscuro es, más probable es que un dato localizado en dicha área sea una anomalía.

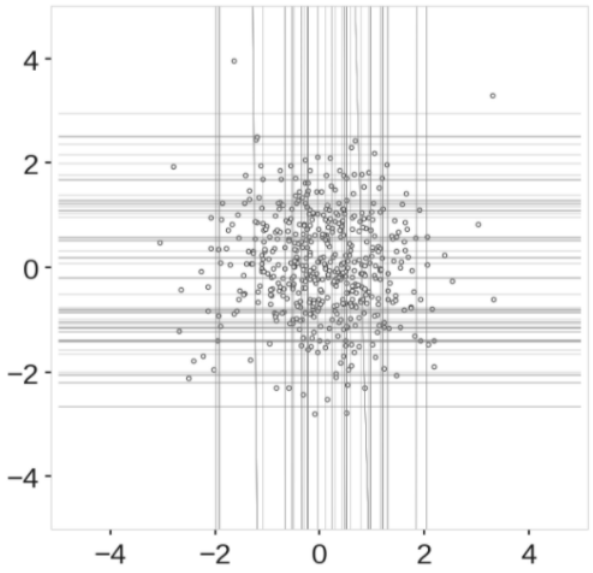

Figura 2.8: Árboles de decisión de Isolation Forest [4]

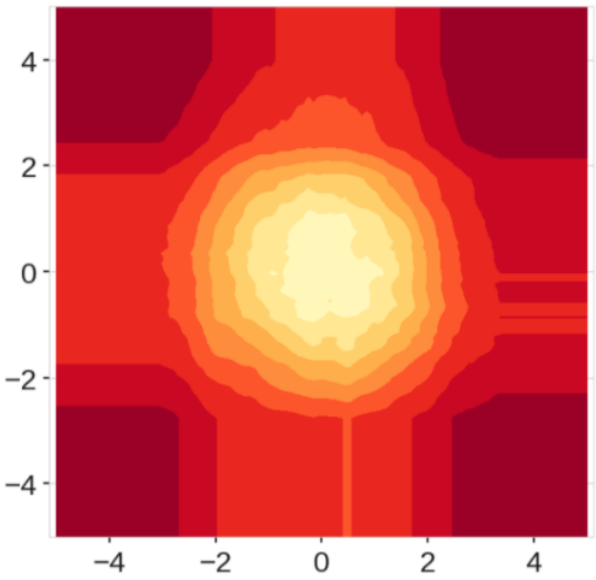

Figura 2.9: Heat map de Isolation Forest [4]

Una vez el modelo ha sido aprendido, se asigna un valor de anomalía a cada dato. Este valor, nos indicará cómo de anómalo es dicho dato, y se obtiene en función del nivel de profundidad requerido en el modelo de Isolation Forest para alcanzar el dato. El algoritmo es muy sensible al parámetro contaminación, el cuál indica el porcentaje de elementos anómalos que estimamos existen en el dataset. En la Figura 2.10 se puede observar cómo un dato anómalo requiere de muchos menos cortes en el plano que uno común [30] [4] [31].

El Isolation Forest ha sido empleado para la detección de anomalía en casos industriales reales como en procesos de fabricación de semiconductores [32] o en comparación con otros algoritmos de aislamiento [33], logrando resultados positivos.

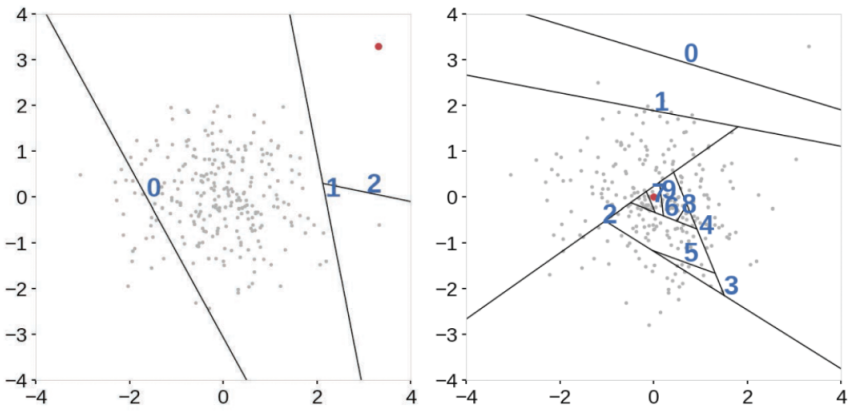

Figura 2.10: Valor de anomalía de ejemplo de Isolation Forest [4]

Kernel Density Estimation (KDE)

Kernel Density Estimation (KDE) consiste en aprender una función de densidad que represente una estimación fiel de la muestra de entrenamiento. Para ello, en la posición de cada dato se agrega una función de densidad a elegir, por ejemplo, Gaussiana, Exponencial o de Epanechnikov. Una vez se fija el kernel de densidad de cada dato, se crea una función de densidad global, como se puede observar en la Figura 2.11.

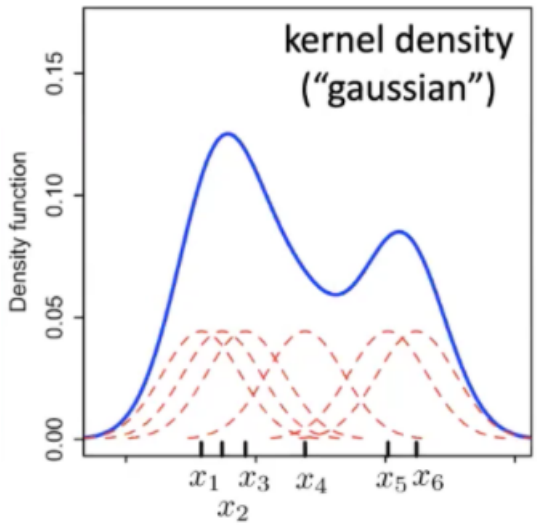

Figura 2.11: Creación de la función de densidad en KDE [5]

Para calcular una función de densidad o kernel de un dato, se aplicará la siguiente fórmula:

$K_h(x) = \frac{1}{h} K(\frac{x}{h})$

Donde:

- K indica el kernel que se haya utilizado como función de densidad.
- h es un parámetro que controla la anchura de las funciones de densidad asignadas a cada dato. La función $K(h)$ para cada $h > 0$ tiene una densidad acumulada de 1, por lo tanto, dividiremos por la anchura.

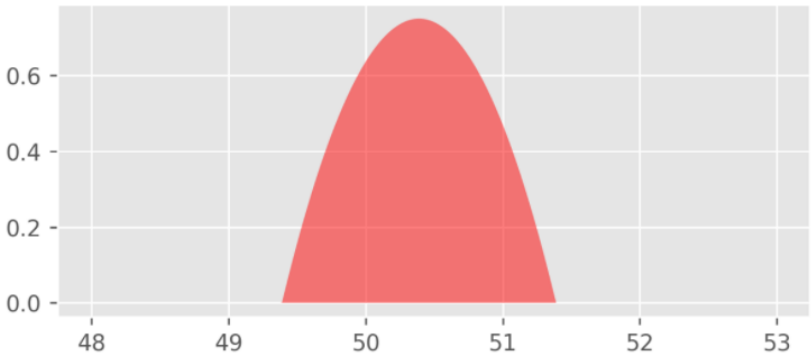

Figura 2.12: Función de densidad en KDE de un único dato [6]

Dicho valor de densidad, está centrado en 0, por lo que para poder ubicarlo en la posición del dato, se debe restar la posición espacial respecto a la del punto.

Es por ello que para el punto $\boldsymbol{x_i}$, la función de densidad se calcula:

$\frac{1}{nh} K(\frac{x - x_i}{h})$

Donde cada x_i tendrá un valor de densidad de $\frac{1}{n}$

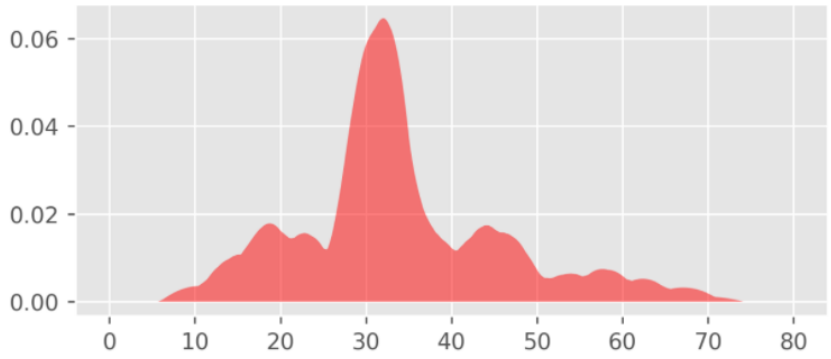

Figura 2.13: Función de densidad KDE aprendida a partir de un dataset [6]

A la hora de realizar el modelo para la detección de outliers basado en KDE, se decidirá según el valor de densidad asignado a dicho dato [34] [31].

Local Outlier Factor (LOF)

Local Outlier Factor (LOF) es un algoritmo que nos devuelve un indicador del grado de anomalía de cada dato dado un número k de vecinos. Dicho algoritmo genera para cada dato, un área cuyo centro es el dato, y cuyo extremo alcanza al k vecino más cercano. El grado de anomalía se calcula a partir de la densidad de puntos del área, siendo los datos centrados en las áreas menos pobladas considerados datos anómalos.

Primero, el algoritmo calcula la distancia a k ($dist(k)$), siendo k el número de datos que se incluirán en el área de cada dato, es decir, el número de vecinos de cada dato. Esta distancia se define como la distancia hasta el k vecino más lejano. En la Figura 2.14 se puede observar un ejemplo de dicha distancia a k. En ese caso, al ser $k = 2$ y estar en un espacio de dimensión 2, se define un área cuyo radio equivalga a la distancia a k. Al haber dos puntos a la misma distancia, ambos son incluidos.

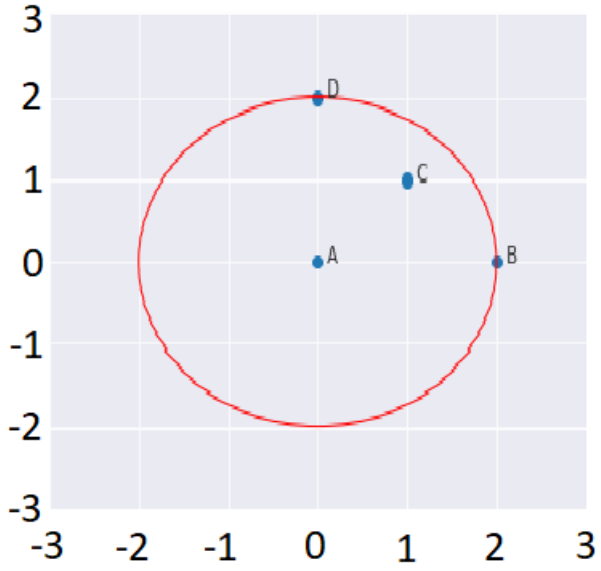

Figura 2.14: Ejemplo distancia a K para K=2 [7]

Dicha distancia a k es usada para calcular la *Distancia de Accesibilidad*. Esta *Distancia de Accesibilidad* devuelve la distancia entre los dos puntos en caso de que esté fuera del vecindario. Si no, devuelve la distancia a k.

$$DistAccesibilidad(\boldsymbol{x_i}, \boldsymbol{x_j}) = max(K - dist(\boldsymbol{x_j}), dist(\boldsymbol{x_i}, \boldsymbol{x_j}))$$

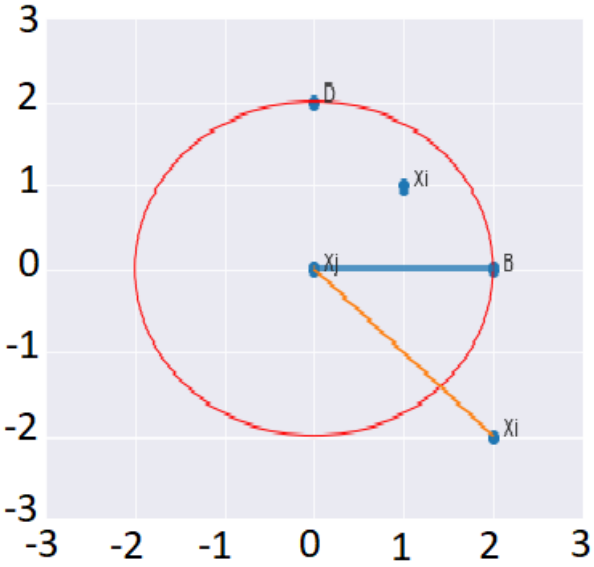

Figura 2.15: Ejemplo distancia de accesibilidad[7]

Finalmente, la *Distancia de Accesibilidad* se utiliza para calcular la *Densidad de Accesibilidad Local* (LRD). Esta se calcula realizando la inversa de la media de las *Distancias de Accesibilidad* de todos los puntos del vecindario. Este valor indica cómo de cerca está el dato del cluster más cercano. Un valor bajo indica que es un dato lejano.

$$LRD_k(A) = \frac{1}{\sum x_{j \in N_k(A)} \frac{DistAccesibilidad(A,x_j)}{||N_k(A)||}}$$

Una vez se tiene este valor, se calcula el LOF de cada punto, que consiste en el ratio entre la media de la *Densidad de Accesibilidad Local* de cada punto del vecindario de un punto, y la *Densidad de Accesibilidad Local* del propio punto. En caso de que el dato sea un dato común el valor de LOF será cercano a 1, ya que la densidad de los datos de su cluster será similar a la de sus vecinos. En caso de que el dato sea un outlier, el valor LOF será mucho más alto ya que la densidad de los vecinos será notablemente superior a la del dato.

$$LOF_k(A) = \frac{\sum x_{j \in N_k(A)} LRD_k(x_j)}{||N_k(A)||} \frac{1}{LRD_K(A)}$$

Como resumen, el indicador de anomalía de este método nos indica cómo de denso es su vecindario respecto al de sus vecinos [8] [35] [31]. Es decir, si el dato

está situado en un área con gran densidad de datos, será considerado un dato más normal que un dato que esté situado en un área alejado del resto de datos.

El algoritmo LOF ha sido empleado en situaciones reales, como para el caso de detectar anomalías en procesos químicos, obteniendo buenos resultados [36]. También en casos como detectar anomalías para detectar lavados de dinero ha demostrado ser un algoritmo eficiente [37].

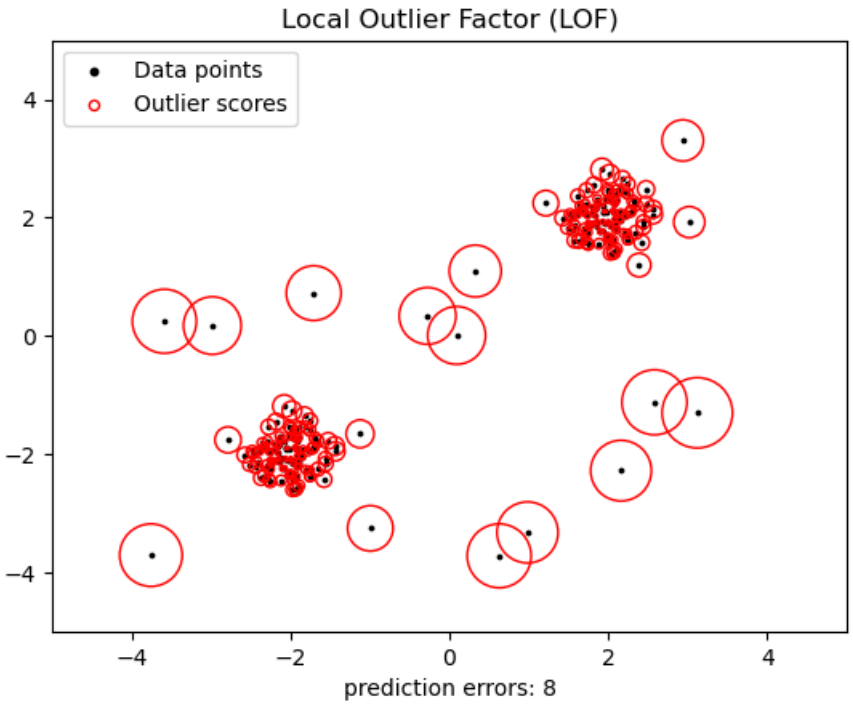

Figura 2.16: Ejemplo LOF [8]

2.3.3 Deep Learning

Autoencoder

Por otro lado, se empleará un modelo Autoencoder para la detección de anomalías. Los Autoencoders, son redes neuronales con una forma y función específica. Están compuestos por múltiples capas densas de neuronas conectadas entre sí. Se pueden dividir en dos fases, la fase de codificación, donde el número de neuronas por capa va decreciendo, y la fase de decodificación, donde el número de

neuronas por capa va aumentando progresivamente hasta llegar al tamaño original. Tanto la compresión como la posterior descompresión de los datos se realiza mediante combinaciones no lineales.

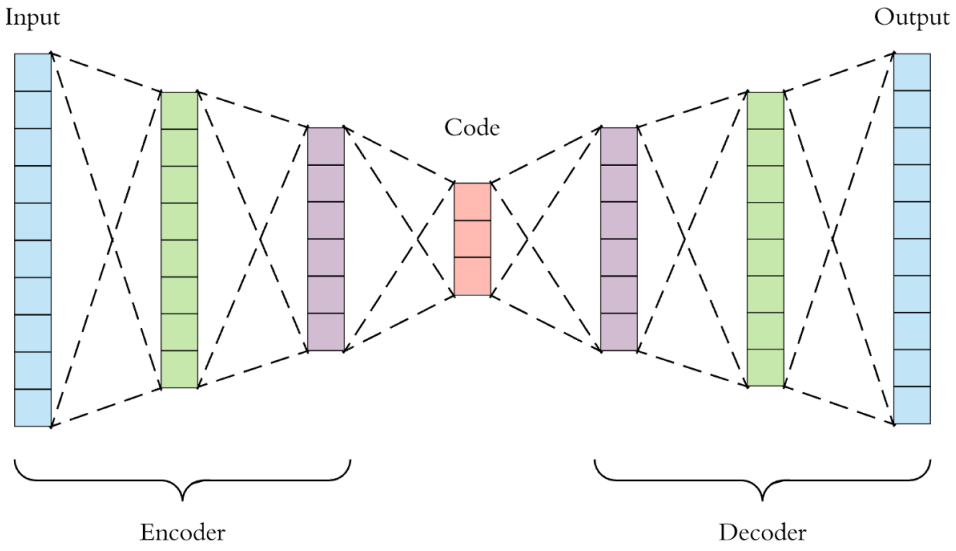

Figura 2.17: Ejemplo de Deep Autoencoder [9]

En estos casos, el dato de entrada es de un tamaño igual que el de salida, ya que el objetivo es la reconstrucción del dato [38]. A la hora de entrenar el modelo, se le indica el dato de entrada y el dato objetivo. Esto puede usarse para entrenar un modelo pasándole imágenes con ruido y que el objetivo de la salida sea una imagen limpia, para que posteriormente sea capaz de limpiarlas. En este proyecto, debido a que el objetivo es la detección de anomalías, el modelo tratará que la reconstrucción del dato sea lo más fiel posible. De ese modo, cuando el modelo sea incapaz de reconstruir un dato correctamente, se podrá deducir que se trata de un dato anómalo [39].

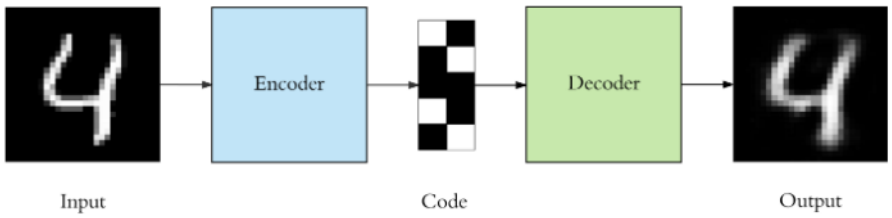

Figura 2.18: Ejemplo de reconstrucción del dato [9]

Autoencoder convolucional

La principal diferencia entre los Autoencoders y los Autoencoders convolucionales es que en lugar de emplear capas totalmente conectadas entre sí, se utilizan capas convolucionales; pero el funcionamiento y objetivo es el mismo.

Una capa convolucional es el resultado de aplicar un filtro a una imagen, y en cada capa se puede aplicar más de un filtro. Cada filtro se compone de distintas variables.

Cuadro 2.1: Variables usadas en los filtros de la convolución

Variable	Descripción
W	Forma del dato de entrada
F	Tamaño del filtro
S	Salto que se empleará la hora de desplazar el filtro
P	Padding que se empleará para los bordes de las imágenes

Teniendo en cuenta las variables descritas en la Tabla 2.1, el tamaño del dato de salida será de $(W - F + 2P)/S + 1$.

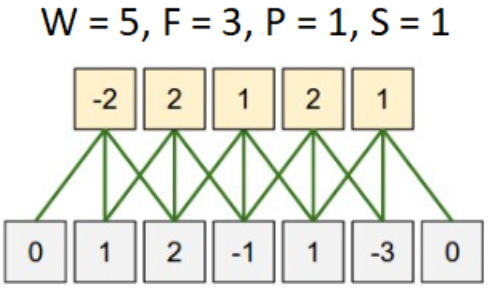

Figura 2.19: Ejemplo de filtro 1 [10]

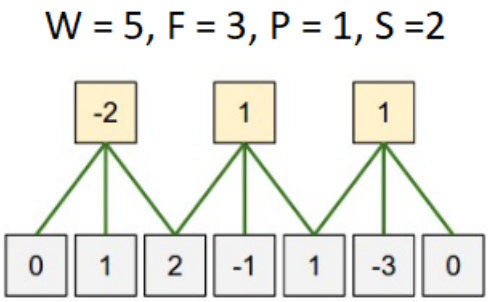

Figura 2.20: Ejemplo de filtro 2 [10]

En la Figura 2.21 se puede observar cómo al aplicar una convolución a una imagen de 32 píxeles de alto por 32 píxeles de ancho (con una profundida de 3 debido a los canales de color RGB), se aumenta la profundidad de la imagen, ya que se tendrán 5 capas, manteniendo el tamaño original gracias al padding.

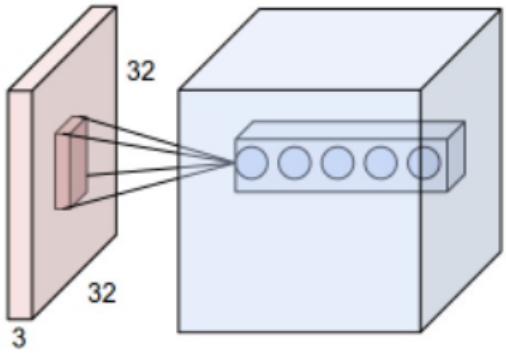

Figura 2.21: Capa convolucional [10]

Cada capa dispone además de una función de activación, la cual devuelve un valor en función del valor obtenido a partir del filtro. Por ejemplo, una de las más populares es Relu, la cual devuelve 0 en caso de que el output del filtro sea negativo, y el mismo valor en caso de que sea positivo.

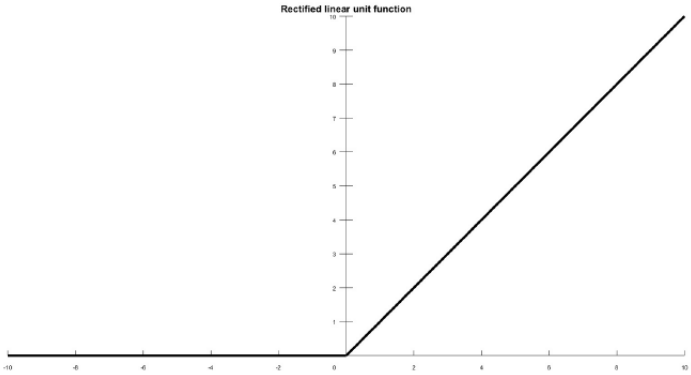

Figura 2.22: Función de activación Relu

Debido a que el tamaño aumenta de manera rápida, se aplica una técnica llamada pooling, que consiste en reducir el tamaño de la imagen. Aunque pueda parecer que esto conlleva una pérdida de información, es común usar un max pooling, que consiste en seleccionar el mayor valor para la reducción. Es decir, se logran representaciones más compactas de la imagen con un alto valor informativo [40].

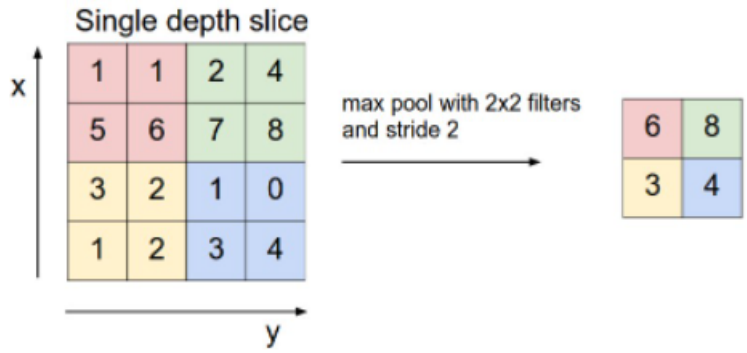

Figura 2.23: Max pooling [10]

Esto implica que la imagen reducida contiene la información más relevante que el filtro ha sido capaz de extraer. Gracias a esto, se logran disminuir los costes computacionales a la par que se agrupa la información relevante.

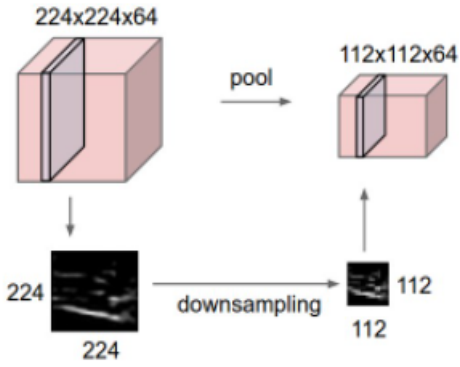

Figura 2.24: Max pooling ejemplo [10]

Este proceso de capa convolucional y pooling va formando una disminución en altura y anchura de las imágenes y un aumento en su profundidad, ya que se irán aumentando el número de filtros. El momento en el que los datos son pequeños y alargados, se entiende que contienen las características extraídas de las imágenes. Posteriormente, comienza un proceso de reconstrucción, en el que se combinan los filtros y se realizan progresivos upsamplings, que consisten en aumentar el tamaño de la imagen, para lograr recuperar el tamaño original expandiendo los valores que contiene.

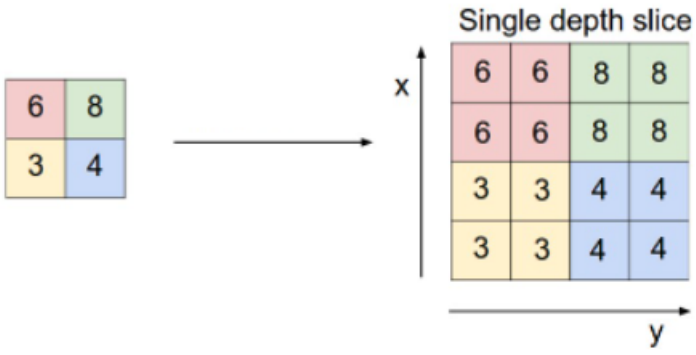

Figura 2.25: Upsampling

Una vez se ha creado la estructura, se debe entrenar el modelo durante un número determinado de iteraciones, en las que, utilizando un valor que nos indique la calidad de la reconstrucción, se actualizan los pesos mediante backpropagation del error. Se emplea un optimizador con un ratio de aprendizaje.

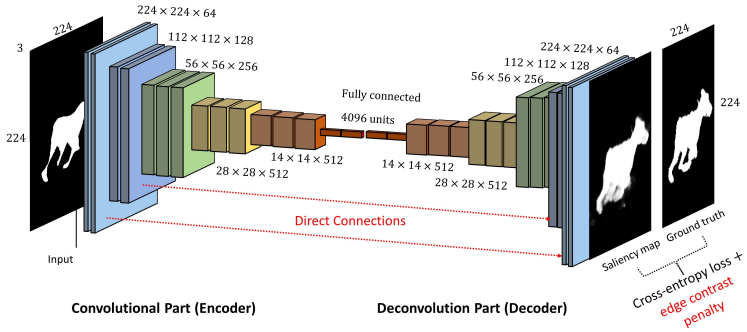

Figura 2.26: Arquitectura de autoencoder convolucional [11]

Finalmente, se pueden observar las diferencias entre el dato de entrada y el dato de salida.

Capítulo 3

CASO DE ESTUDIO

3.1 Descripción del problema

El comercio por mar supone el 80 % del comercio internacional y las terminales de contenedores tienen requisitos exigentes de frecuencia y estabilidad de servicio.

Figura 3.1: Grúa en el Astillero

En este área, los incidentes de las grúas de muelle componen un 24 % de los costes del puerto de los cuales, hasta un 25 % de los incidentes puede ser causado por un mantenimiento incorrecto de la estructura. Además de los costes, un mantenimiento incorrecto puede llegar a suponer periodos de inactividad más largos, o incluso permanentes, y hasta ser un riesgo para la salud de las personas.

Para ello, se desea monitorizar el giro de un eje de una grúa STS (Ship To Shore), localizada en el astillero de Bilbao, con el fin de validar su correcto funcio-

namiento. Una grúa STS es una grúa pórtico montada sobre carriles para realizar labores de carga y descarga de contenedores de barco a muelle y viceversa. Dicha grúa se muestra en la Figura 3.2.

Figura 3.2: Grúa STS

Debido a que el eje gira a una velocidad muy lenta, la sensórica tradicional basada en acelerómetros no es la más adecuada para recabar datos útiles. Por ello se han colocado dos sensores de desplazamiento. Ambos colocados perpendicularmente al eje vertical con una diferencia de 90º entre ellos. Un esquema de dichos sensores se muestran en la Figura 3.3.

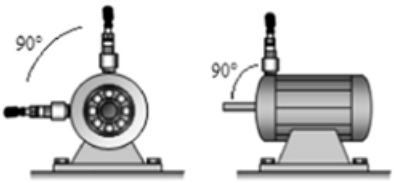

Figura 3.3: Posicionamiento de los sensores de desplazamiento [12]

Estos sensores miden la distancia respecto al cilindro que gira perpendicular a ellos, concretamente respecto a una placa de metal ubicada en el cilindro, por lo que las señales capturadas tienen un desfase de 90º. Debido a que ningún proceso

es perfecto, esta distancia varía en el tiempo en forma de sinusoide con cierto ruido. Este efecto se produce por la imposibilidad de crear un cilindro perfecto, ya que al estar levemente desalineado, el cilindro se acerca y se aleja periódicamente del sensor, obteniendo así una señal sinusoidal. Como se puede ver en la Figura 3.4, en el eje vertical se encuentra la distancia en milímetros y en el eje horizontal el tiempo en milisegundos [13].

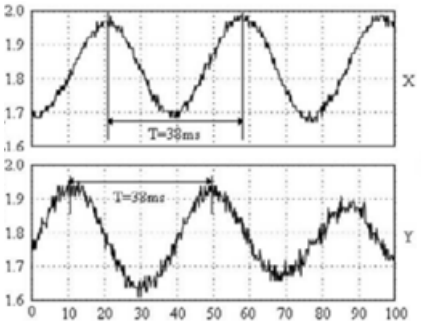

Figura 3.4: Ondas sinusoidales recogidas [13]

Al graficar la representación polar de ambas distancias, se observa que se genera un círculo con dichas señales, tal y como se ve representado en la Figura 3.5.

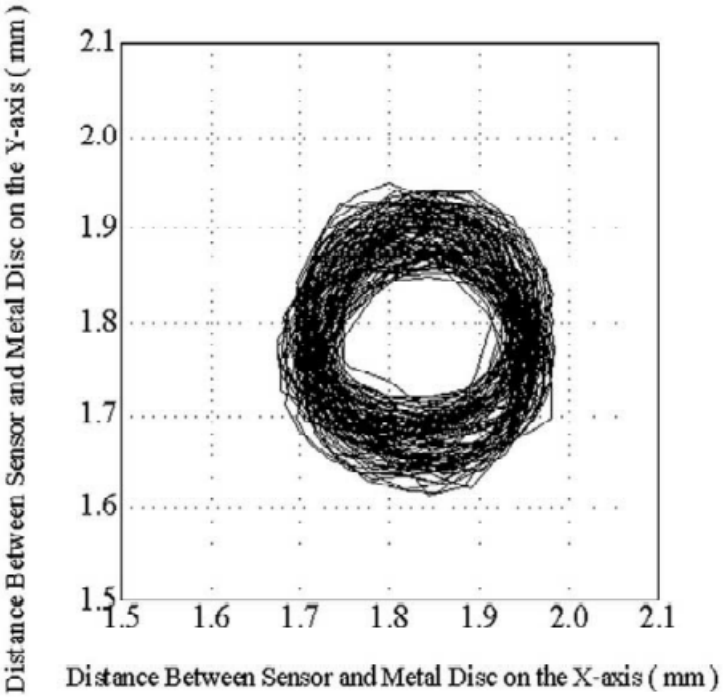

Figura 3.5: Representación polar de los sensores de desplazamiento [13]

Estas figuras serán los datos con los que se trabajará en el proyecto. En la Figura 3.5 se puede observar que se forma un círculo casi perfecto. Idealmente, ese será el output que devolverán los sensores de desplazamiento. Sin embargo, existen problemas en el eje que pueden desfigurar la señal.

El primero de los fallos es el desbalanceo del motor, en el cual la distribución de la masa en el centro de rotación del motor se desequilibra. Por otro lado, es posible que el eje esté desalineado. Esto provoca una vibración más alta de lo normal, haciendo que se acelere el deterioro de la máquina [12]. Dichos errores se observan en la Figura 3.6, en la que se ve cual es el caso correcto y los casos de desbalanceo, desequilibrio y desequilibrio extremo.

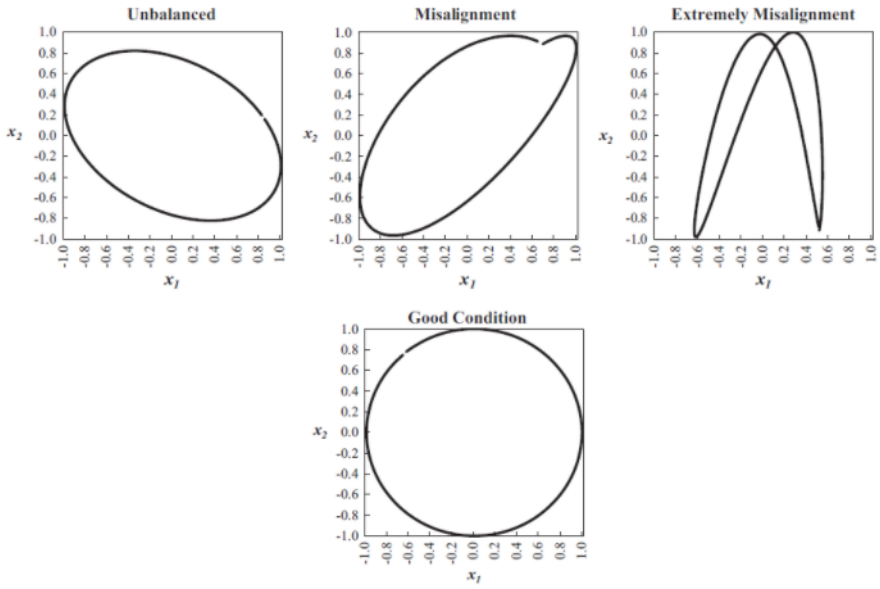

Figura 3.6: Tipos de outputs [12]

3.2 Generación de los datos

Debido a la imposibilidad de obtener datos reales, se ha obtenido la fórmula matemática con la que simular una generación de datos cercana a la realidad.

Los datos que se generan son dos ondas sinoidales con un desfase de 90º, a partir de las cuáles se obtiene la representación polar. En la Tabla 3.1 se muestran las variables empleadas en la generación de datos.

Cuadro 3.1: Variables usadas en la generación de datos

Variable	Descripción
$a1, b1$	Valores de amplitud
$x1, x2$	Valores de fase de onda
w	Velocidad angular
t	Puntos de muestreo

Las fórmulas empleadas para los datos correctos muestran dos ondas sinusoidales con las mismas variables con un desfase de 90º.

$$axis_1 = a1 * np.sin(w * t)$$

$$axis_2 = a1 * np.cos(w * t)$$

Los datos de desbalanceo sin embargo, muestran que cada onda dispone de una amplitud distinta.

$$axis_1 = a1 * np.sin(w * t + x1)$$

$$axis_2 = b1 * np.cos(w * t + x1)$$

Finalmente, los datos de desalineación muestran como una de las ondas varía más que la otra, dando lugar a un tipo de error más grave.

$$axis_1 = a1 * np.sin(w * t + x1) + a2 * np.sin(2 * w * t + x2)$$

$$axis_2 = b1 * np.cos(w * t + x1)$$

3.3 Esquema de experimentación

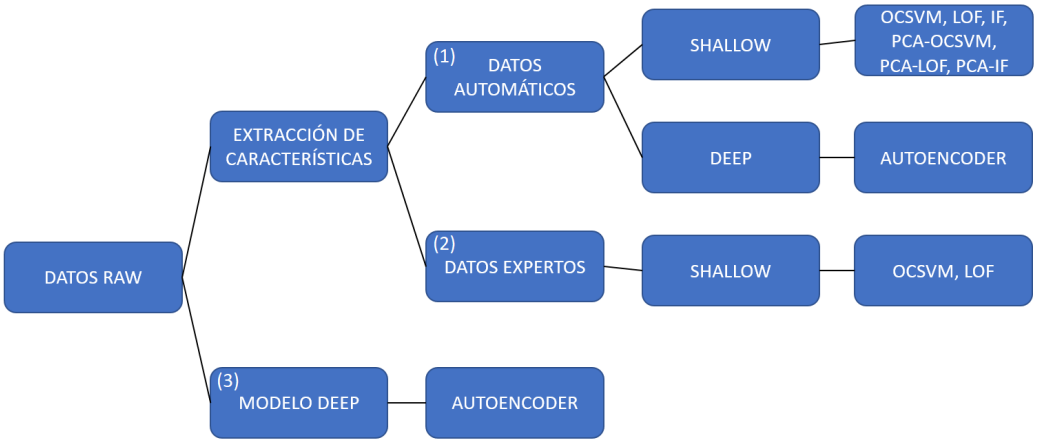

Figura 3.7: Esquema de experimentación

En el proyecto se pueden diferenciar diferentes ramas de actuación, por un lado (1) una extracción de características automáticas, por otro lado (2) una extracción de características expertas, y finalmente (3) el empleo de un algoritmo profundo sin extracción de características y trabajando con las propias imágenes.

Primeramente se propuso realizar una extracción de características, por lo que

se realizó una extracción automática mediante la librería Tsfresh [41] con la que se extrajeron las características listadas en la sección 3.4: “Extracción de características automáticas”. Con estas variables automáticas se aprendieron modelos Shallow (One Class Support Vector Machines (OCSVM), Local Outlier Factor (LOF) e Isolation Forest(IF)) tanto con todas las variables como aplicando un PCA. Finalmente se aprendió un modelo Deep, siendo este un autoencoder, con todas las variables, ya que a pesar de que se ha realizado un extracción de características, esta es muy amplia, por lo que se espera que el autoencoder sea capaz de extraer características a partir de las variables.

Debido a las razones explicadas en la sección 4.3: “Explicabilidad de los datos automáticos”, se llevó a cabo una extracción de variables expertas o de alto nivel, con las que se aprendieron modelos Shallow (One Class Support Vector Machines (OCSVM) y Local Outlier Factor (LOF)).

Por último, se implementó un modelo Deep empleando un autoencoder con los datos raw o de bajo nivel, es decir, utilizando como datos de entradas las propias imágenes.

3.4 Extracción de características automáticas

En esta sección se mostrarán en la Tabla 3.3 las características extraídas para el aprendizaje del primer modelo. Se añadirá una breve descripción acompañada de la fórmula matemática a partir de la cual se extrae la característica concreta. Además, en la Tabla 3.2 se mostrarán las variables que usaremos para describir las ondas.

Cuadro 3.2: Variables usadas

Variable	Descripción
X	Onda sinusoidal
X_i	Valor del eje vertical de la onda en el punto *i*
$X1, X2$	Primera y segunda onda respectivamente
$X1_i, X2_i$	Valor del eje vertical de las ondas $X1$ y $X2$ en el punto *i*
n	Tamaño
{i}th	El i-ésimo elemento

Cuadro 3.3: Características extraídas

Característica	Descripción	Fórmula
Valor máximo	El valor máximo que cada onda alcanza en el eje vertical	$max(X)$
Valor mínimo	El valor mínimo que cada onda alcanza en el eje vertical	$min(X)$
Media	El valor medio del eje vertical para cada onda	$\mu = \frac{1}{n}\sum_{i=1}^{n} X_i$
Diferencia entre máximos en el eje vertical	La diferencia entre el máximo de la primera onda y la segunda	$diff_Y = max(X1) - max(X2)$
Diferencia entre máximos en el eje horizontal	La diferencia entre la primera localización del máximo entre las dos ondas. Esto puede resultar realmente útil ya que para que las ondas generen datos correctos deben tener un desfase de 90º	$diff_X = posX(max(X1)) - posX(max(X2))$

Curtosis	La curtosis nos aporta información acerca de la forma de distribución. Por un lado tenemos las distribuciones Leptocúrticas, que son las distribuciones con un alto valor de curtosis. Una curtosis alta implica mayor valor en la moda que en las colas de la distribución. Por otro lado tenemos las distribuciones Platicúrticas con un valor de curtosis negativo. Una curtosis negativa implica que la moda será menos picada y habrá unos valores más uniformemente distribuidos en las colas y en las modas. Finalmente, las distribuciones Mesocúrticas son las que tienen un valor de curtosis cercano a 0, es decir, más equilibradas, como podría ser la distribución normal. Estos casos se muestran en la Figura 3.8, donde podemos observar cada tipo de curtosis con su valor asociado.	$\beta_2 = \frac{\mu_4}{\sigma^4}$ Donde: $\mu_k = E[(X - E[X])^k]$

Skewness	La función de asimetría estadística o skewness nos indica cómo de simétrica es la onda. Un valor cercano a 0 nos indica que es simétrica, mientras que si el valor es cercano a 1, la señal tendrá valores superiores en el lado izquierdo. En caso de tener un valor cercano a -1, los valores altos estarán localizados en la derecha	$\tilde{\mu_3} = \frac{\mu_3}{\sigma^3}$ Donde: $\mu_k = E[(X - E[X])^k]$
Desviación estándar	Nos indica cuánto se desvía cada onda de la media.	$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \mu)^2}{n}}$
Complejidad	Para cada onda se devuelve un valor de complejidad. Un alto nivel de complejidad indica que la onda tendrá más picos y valles, mientras que un bajo nivel de complejidad implica una onda más llana. Esta característica puede resultar útil en casos de desalineado grave, ya que pueden surgir más picos que en una onda sinoidal común.	$complexity = \sqrt{\sum_{i=1}^{n-1}(X_i - X_{i-1})^2}$
Mediana	El valor de la mediana de cada onda.	$mediana = \{(n+1)2\}th$
Root Mean Square	Nos devuelve el valor de RMS para cada onda. La media cuadrática es un tipo de media estadística en el que se toma la media de los valores elevados al cuadrado y se le aplica una raíz cuadrada.	$RMS = \sqrt{\frac{\sum_{i=1}^{n} X_i^2}{n}}$

Root Mean Square Error	A pesar de que se suele utilizar para comprobar diferencias entre valores predichos y observados, nosotros lo usaremos para ver la diferencia entre las dos ondas. Cuanto más cercano sea el valor a 0, más similares serán las ondas.	$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(X_i - Y_i)^2}{n}}$
Número de veces que cruza el eje de abscisa	Nos indica el número de veces que la onda cruza el eje de abscisas.	$f(X) = 0$
Número de picos	Nos indica el número de picos de cada onda, basándonos en un parámetro de 5 vecinos a tomar en cuenta.	X_i cuando: $X_i > X_{i-2}, X_{i-1}, X_{i+1}, X_{i+2}$
Energía	Nos indica el valor absoluto de la energía de cada onda. La energía está directamente relacionada con la amplitud y la frecuencia de la onda. A mayor amplitud y frecuencia, mayor energía.	$E = \sum_{i=1}^{n} X_i^2$
Diferencia entre energías	La diferencia ente las energías de las ondas.	$diff_E = \sum_{i=1}^{n} X1_i^2 - \sum_{i=1}^{n} X2_i^2$
Autocorrelación	La autocorrelación, es la correlación entre la onda y la misma onda a la que se le aplica un delay temporal. Función empleada para la detección de patrones cuando hay ruido, por lo que puede ser útil si introducimos ruido en las funciones.	$Autocorrelacion = \frac{1}{(n-l)\sigma^2} \sum_{t=1}^{n-l}(X_t - \mu)(X_{t+l} - \mu)$ Donde l representa el lag que tiene una onda respecto a la otra
Correlación entre dos ondas	La correlación entre dos ondas nos sirve para comprobar el grado de similitud entre dos ondas.	$z[k] = (X * Y)(k - N + 1) = \sum_{i=0}^{\|\|X\|\|-1} X_i Y_{i-k+N-1}$

Sample entropy	Indica el valor de entropía para cada onda.	$Entropia = -log\frac{A}{B}$ Donde: $A = d[X_{m+1}(i), X_{m+1}(j)] < r$ $B = d[X_m(i), X_m(j)] < r$ $d[X_m(i), X_m(j)]$ es la distancia de Chebyshev m representa la longitud de un nuevo vector r representa la tolerancia

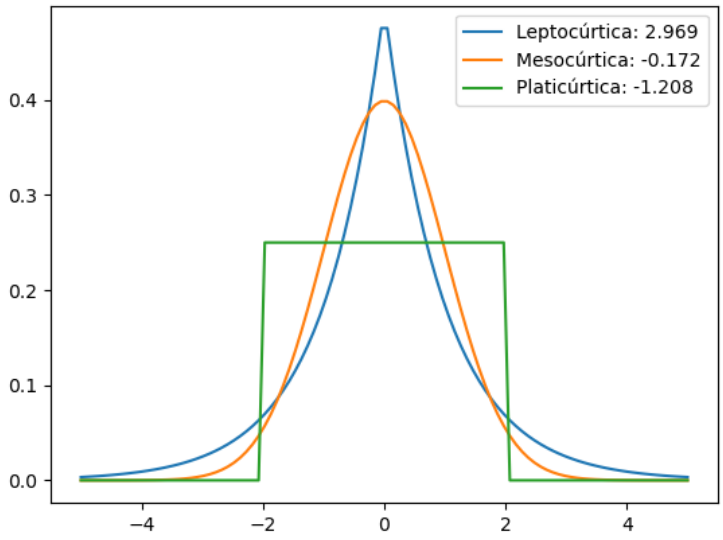

Figura 3.8: Diferentes tipos de curtosis

3.5 Modelos

Esta sección recogerá los modelos aprendidos para la clasificación de los datos con sus respectivos parámetros.

3.5.1 One Class Classification

Para la clasificación de los datos del proyecto, se escogieron los algoritmos One Class Support Vector Machine, Isolation Forest, Local Outlier Factor y Autoencoders.

One Class Support Vector Machine (OCSVM)

Los hiperparámetros recogidos en la Tabla 3.4 serán los seleccionados para implementar modelos de One Class Support Vector Machine.

Cuadro 3.4: Parámetros del SVM

Parámetro	Descripción	Valor
Kernel	El tipo de kernel que se empleará para pasar al espacio de dimensionalidad mayor	Polinomial
Gamma	El coeficiente del kernel	0.007
Nu	El límite para el error en el entrenamiento	0.007

Isolation Forest (IF)

Los hiperparámetros recogidos en la Tabla 3.5 serán los seleccionados para implementar un modelo de Isolation Forest.

Cuadro 3.5: Parámetros del IF

Parámetro	Descripción	Valor
Número de estimadores	El número de árboles que se generarán en el forest	250
Número de muestras máximas	El número de muestras con las que entrenar cada estimación	Automático
Contaminación	El porcentaje de outliers que hay en el dataset	0.1
Número de características máximas	El número de características con las que entrenar cada uno de los árboles	8

Local Outlier Factor (LOF)

Los hiperparámetros recogidos en la Tabla 3.6 serán los seleccionados para implementar modelos de Local Outlier Factor.

Cuadro 3.6: Parámetros del Local Outlier Factor

Parámetro	Descripción	Valor
Número de vecinos	El número de datos a tener en cuenta a la hora de calcular la densidad del dato	30
Algoritmo	El algoritmo a emplear para computar los vecinos más cercanos	kd_tree
Leaf size	Profundidad del árbol del kd_tree	30
Metric	El tipo de distancia con la que medir la distancia entre puntos	Euclídea (Distancia de Minkowski con p=2)

3.5.2 Deep Learning

Autoencoder con características extraídas manualmente

La red neuronal se compone de 1 capa de entrada, 1 capa de salida y 5 capas ocultas. Debido a que el tamaño tanto de entrada como de salida es de 37, las 3 primeras capas ocultas comprimirán la información teniendo 16, 8 y 4 neuronas respectivamente. Para la descompresión, las 2 capas ocultas se compondrán de 8 y de 16 neuronas, respectivamente, tal y como se puede observar en la Figura 3.9.

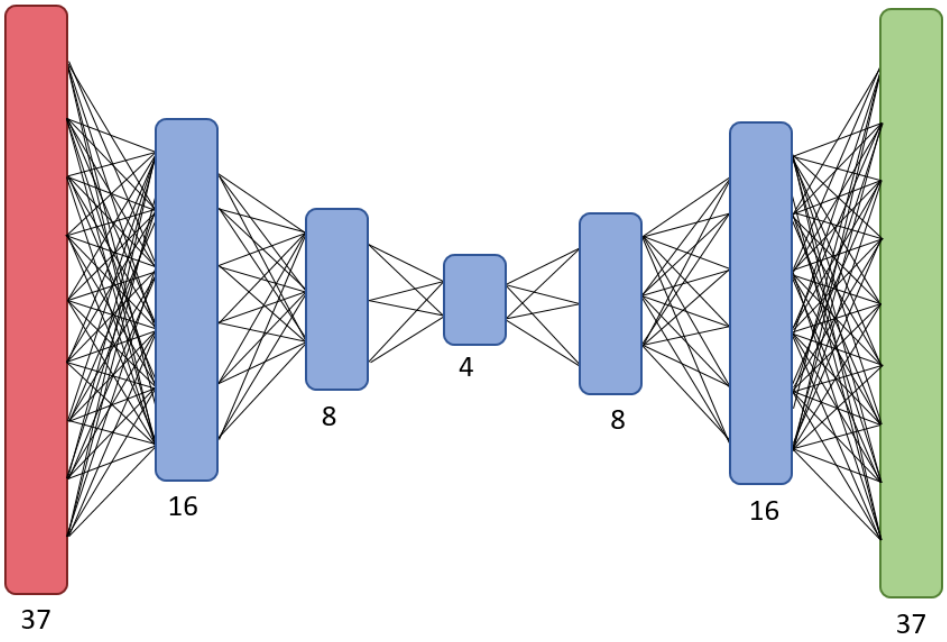

Figura 3.9: Arquitectura del autoencoder

La función de activación de la capa de salida es sigmoidal mientras que las del resto son Relu. El ratio de aprendizaje es constante con un valor de 0.001. Finalmente, el optimizador empleado es Adam y la función de pérdida es el Error Cuadrático Medio.

El entrenamiento se compondrá de 250 iteraciones con un tamaño de batch de 8.

Autoencoder convolucional

Durante la experimentación con el autoencoder convolucional, se observó que a medida que se aumentaba la profundidad del modelo, disminuía drásticamente la eficacia de la reconstrucción. Por ello, se decidió no usar un autoencoder muy profundo, en concreto, se creó un modelo con la estructura resumida en la Tabla 3.7 y representada en la Figura 3.10.

Cuadro 3.7: Estructura autoencoder convolucional

Capa	Forma
Entrada	224x224x1
Convolucional	224x224x16
Max pooling	112x112x16
Convolucional	112x112x8
Upsampling	224x224x8
Convolucional	224x224x16
Salida	224x224x1

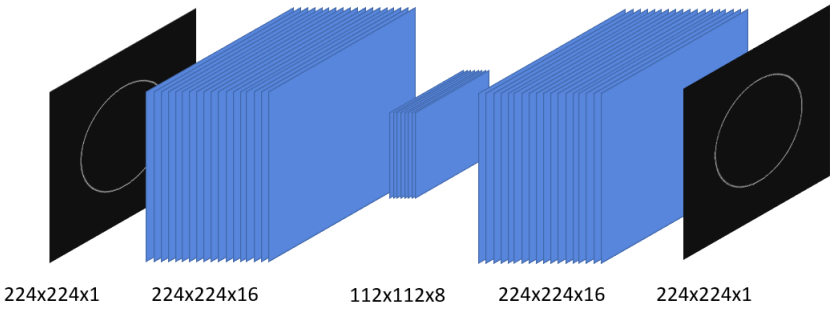

Figura 3.10: Arquitectura del autoencoder convolucional

Inicialmente se le aplica una capa convolucional en la que se le añaden 16 filtros. Una vez la convolución ha sido realizada, se disminuye el tamaño mediante un max pooling, y a esa reducción se le aplican 8 filtros. Después, se aumenta el tamaño mediante un upsampling y se le vuelven a aplicar 16 filtros. Finalmente, esos filtros se agrupan y se deja aislada la capa de salida.

Las imágenes con las que se ha trabajado tienen unas proporciones de 3000 píxeles por 3000 píxeles, una cantidad demasiado grande como para trabajar con ella. Es por ello que el primer paso que se llevó a cabo fue una redimensionalización de las mismas. Las imágenes se redujeron a un tamaño de 224 píxeles por 224 píxeles, disminuyendo en gran medida el tamaño a aproximadamente un 7.46 % del tamaño original y manteniendo la información de la imagen.

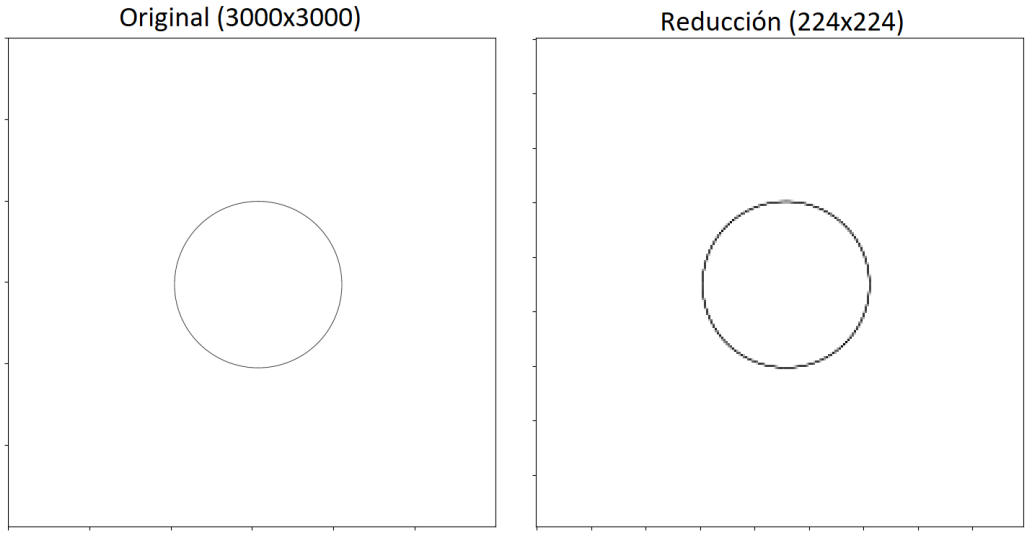

Figura 3.11: Imagen original y reducida

Cabe destacar además que los datos originales consistían en círculos negros sobre fondo blanco. Sin embargo, se han invertido dichos colores para que muestren círculos blancos sobre fondo negro. Esto es debido a que durante la experimentación se obtuvieron mejores resultados aplicando los mismos modelos teniendo un fondo negro.

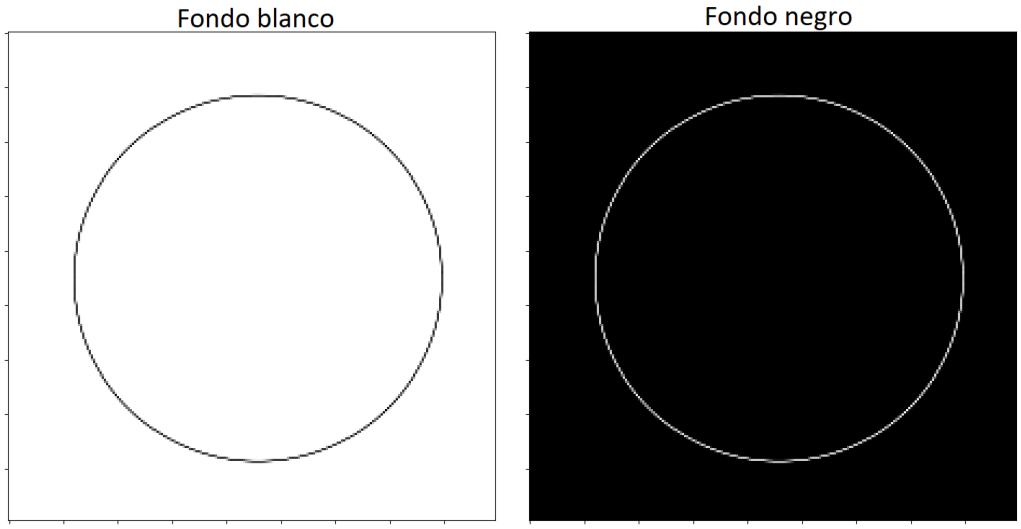

Figura 3.12: Fondo blanco y negro

Como optimizador se ha empleado Adam con un ratio de aprendizaje de 0.01. Como función de pérdida se ha utilizado el Error Cuadrático Medio, para comparar la similitud entre la imagen y su reconstrucción, calculando la diferencia entre cada píxel y sumando dichas diferencias al cuadrado.

Capítulo 4

EXPERIMENTOS Y RESULTADOS

4.1 Preparación de los datos

4.1.1 División en sets de entrenamiento, testeo y validación

Para el entrenamiento de los modelos, se han realizado 3 particiones de los datos. La primera partición corresponde al conjunto de datos de entrenamiento y representa el 80 % de los datos. Al estar trabajando con clasificación One Class, los 1000 elementos de este conjunto están etiquetados como correctos. La segunda y tercera partición corresponden al conjunto de validación y de test, y cada uno representa el 10 % de los datos. En estos conjuntos de datos de 125 elementos cada uno, se encuentran muestras de tipos de fallo y correctas, 42 muestras de desalineación (el primer tipo de fallo), 42 de desbalanaceo (el segundo tipo de fallo) y 42 correctos.

Cuadro 4.1: Partición del conjunto de datos

Conjunto	**Número de datos**	**Porcentaje del conjunto global**	**Tipo de datos**
Total	1252	100 %	Muestras correctas, desbalanceadas y desalineadas
Entrenamiento	1000	80 %	Muestras correctas

Validación	126	10 %	Muestras correctas, desbalanceadas y desalineadas
Testeo	126	10 %	Muestras correctas, desbalanceadas y desalineadas

4.1.2 Datos

Empleando el método explicado en la sección 3.2: “Generación de los datos”, se han creado los datos con las proporciones explicadas previamente. Para la simulación se seleccionaron 10 revoluciones por minuto y 1 hercio de frecuencia como parámetros para la generación de las ondas.

De este modo se obtuvieron los datasets necesarios para el entrenamiento, testeo y validación. En la Figura 4.1 se observa una muestra de 9 datos correctos. Como se puede observar, son todos círculos perfectos. Si bien algunos son de tamaño mayor o menor, todos tienen cada punto de su circunferencia a la misma distancia del centro.

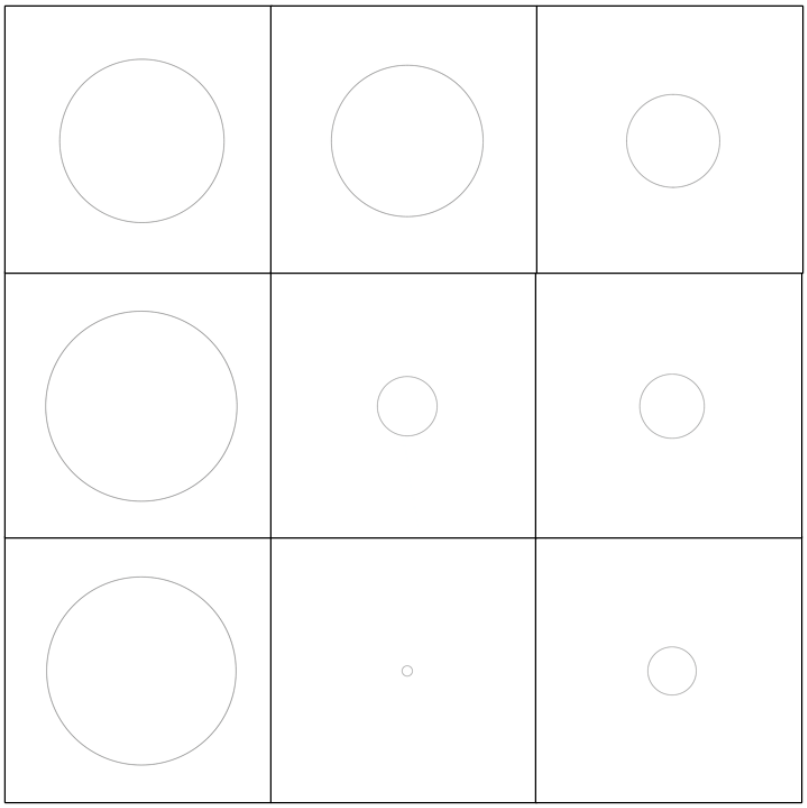

Figura 4.1: Muestra de datos correctos

En la Figura 4.2 se observa una muestra de 9 datos desalineados. Se observa en este caso que los datos tienden a la elipsoicidad, y en los casos más extremos apenas se distingue una elipse de una recta. Cabe destacar que tienden a tener un tamaño menor.

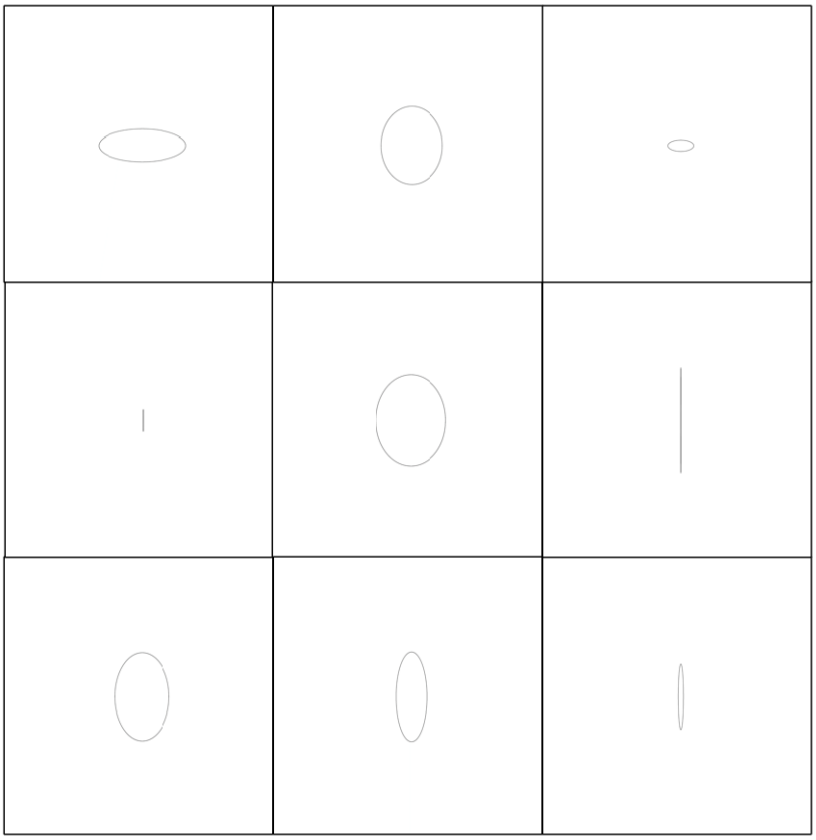

Figura 4.2: Muestra de datos desalineados

Finalmente, en la Figura 4.3 se observa una muestra de 9 datos desbalanceados. Al igual que en los casos de desalineación, tienden a la elipsoicidad pero tienen un tamaño mayor por lo general.

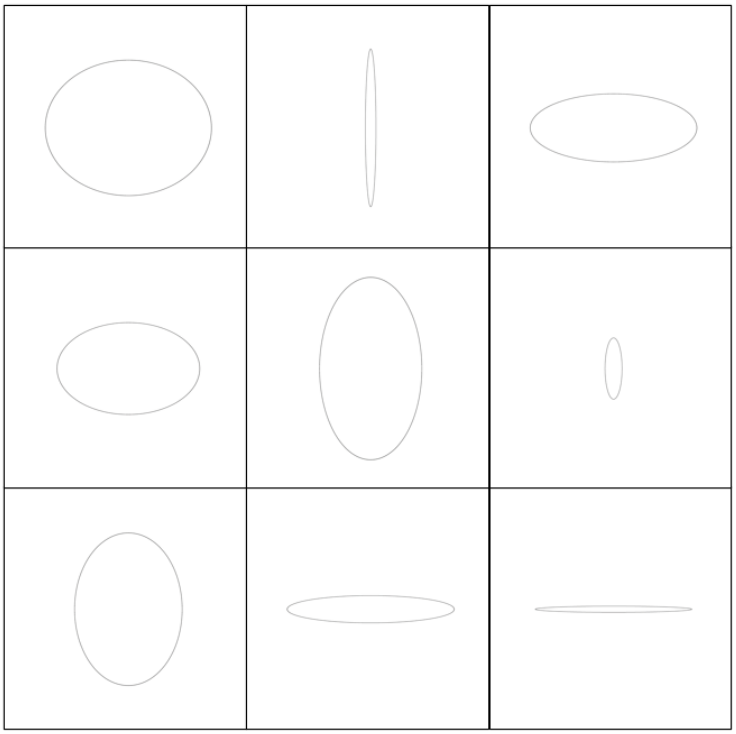

Figura 4.3: Muestra de datos desbalanceados

4.1.3 Aplicación del t-SNE: Visualización en 2D

En nuestro caso, se trabajará con un dataset de 37 variables, por lo que resultaría demasiado complejo representar dicho dataset en un espacio de dimensionalidad 2. Por ello, se va a aplicar la técnica t-SNE para poder visualizar los datos. Se empleará una perplejidad de 75 durante 10000 iteraciones.

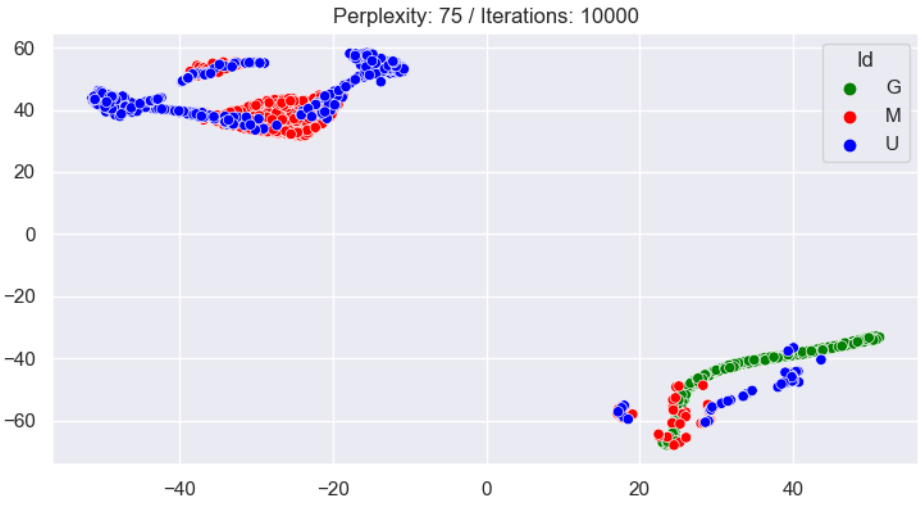

Figura 4.4: t-SNE aplicado a nuestros datos

Como se puede observar en la Figura 4.4, se muestran dos grupos claramente diferenciados. Por un lado, en el de abajo a la derecha se encuentran todos los datos correctos (mostrados en verde y etiquetados con la letra “G” indicando “Good”). Sin embargo, se observan también algunos datos de desalineación (mostrados en rojo y etiquetados con la letra “M” indicando “Misalignment”), muy cercanos a los datos correctos. Finalmente también se pueden distinguir datos de desbalanceo (mostrados en azul y etiquetados con la letra “U” indicando “Unbalance”), algo más separados de los correctos.

Por otro lado, podemos percibir que en el grupo de arriba a la izquierda hay un cluster con datos de fallo, aunque se podría defender que hay dos grupos en vez de uno.

Los datos que aparecen junto a los correctos se corresponden con datos de desalineación o desbalanceo que aún no presentan daños muy grandes por lo que su forma es aún bastante circular, pese a estar algo “achatados”.

4.1.4 Aprendizaje de modelos tras PCA

Para realizar el análisis PCA, primero se analizará la correlación entre los pares de variables.

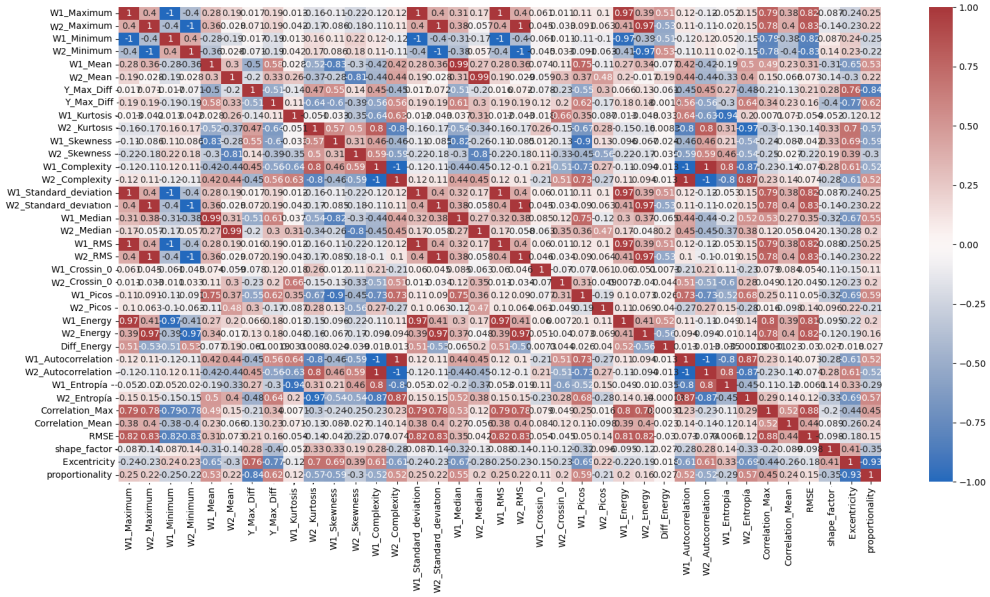

Figura 4.5: Matriz de correlación entre pares de variables

Como se puede observar en la Figura 4.5, hay una gran cantidad de variables que están correlacionadas de manera notable entre sí. Esto tiene mucho sentido, debido a que, observando por ejemplo el máximo valor en el eje vertical y la energía de la primera onda, se muestra que tienen una correlación total, porque, por definición, la energía depende de la amplitud. Como este caso hay varios, pero en vez de eliminar dichas variables, se realizará un análisis por PCA para observar qué resultados se obtienen y en qué medida se puede reducir la dimensionalidad [42].

Mediante una técnica llamada “biplot”, se puede representar en un gráfico de dos dimensiones una proyección de los datos, en nuestro caso las dos primeras Componentes Principales. Mediante este gráfico se puede observar que datos son parecidos entre sí además de la influencia que ejerce cada variable [43].

Para que esta representación se pueda considerar fiel a la realidad, las dos componentes empleadas para representar los datos deberían recoger al menos el

90 % de la varianza total. Sin embargo, nuestras dos Componentes Principales solo recogen un 56 %, tal y como podemos comprobar en la Figura 4.7. El gráfico biplot mostrado en la Figura 4.6, aún no siendo capaz de recoger gran parte de la variante explicada, sirve para visualizar los datos de manera superficial.

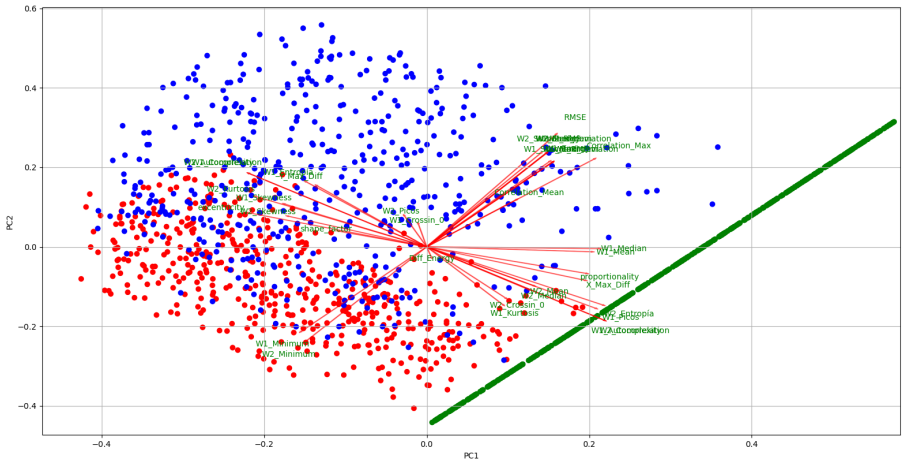

Figura 4.6: Biplot

En dicho gráfico, se puede observar la influencia que ejerce cada variable. Cuando dos variables son ortogonales, significa que su correlación es 0, mientras que si apuntan a una dirección similar u opuesta, implica una positiva o negativa correlación, respectivamente.

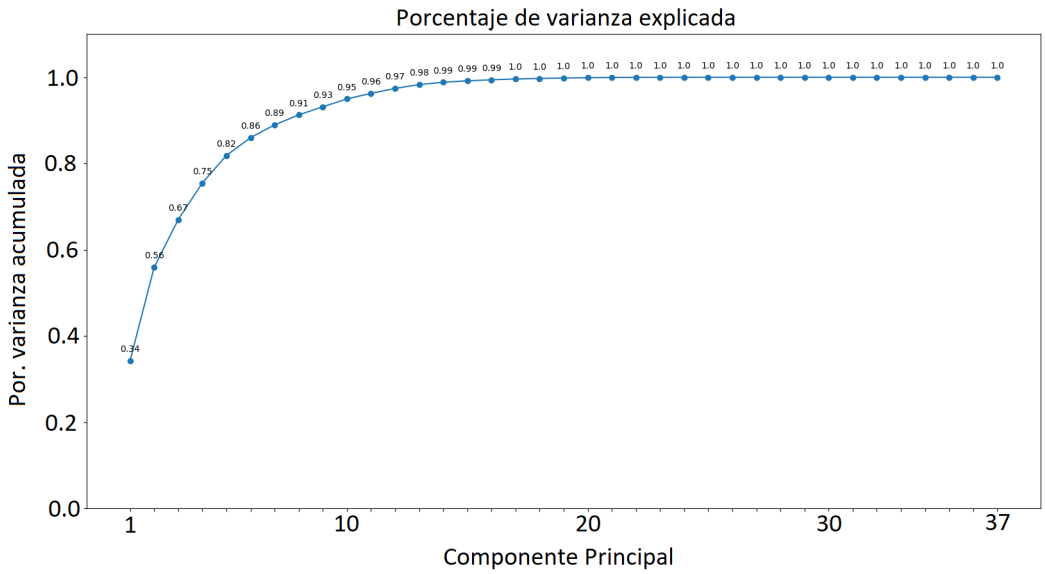

Figura 4.7: Varianza acumulada

Tal y como acabamos de explicar, para que la proyección sea representativa mediante PCA de los datos originales, se debe alcanzar alrededor de un 90 % de varianza explicada, por lo que elegiremos quedarnos con las 8 primeras Componentes Principales.

4.2 Experimentación con los datos automáticos

4.2.1 One Class Classification

Respecto a la clasificación de los datos, se han creado dos tipos de modelos por cada algoritmo: uno con las variables originales, y otro con las componentes principales. Cabe destacar que el set de validación se ha empleado para el ajuste de los parámetros del modelo, mientras que el set de testeo únicamente ha sido usado para probar el modelo y obtener las métricas de rendimiento finales. De este modo, el umbral se ha elegido únicamente teniendo en cuenta los datos de validación, más concretamente, eligiendo el valor del outlier menos anómalo. Los resultados de precisión y matrices de confusión son el resultado de aplicar dicho umbral en el set de testeo.

Respecto a las matrices de confusión, se considerarán positivos los datos con comportamiento normal, a los que se les asignará un “1”. Los datos negativos serán los datos que muestran desalineación o desbalanaceo, y se les asignará un “0”.

One Class SVM

Aprendiendo el modelo con los parámetros mostrados en la Tabla 3.4, se entrenó el modelo con los datos de entrenamiento. Los datos de validación se utilizaron para ajustar dichos parámetros. Finalmente, observamos los resultados en la matriz de confusión mostrada en la Figura 4.8. Se puede observar que los 42 datos de comportamiento normal los ha clasificado correctamente, mientras que 4 de los outliers los ha clasificado como correctos.

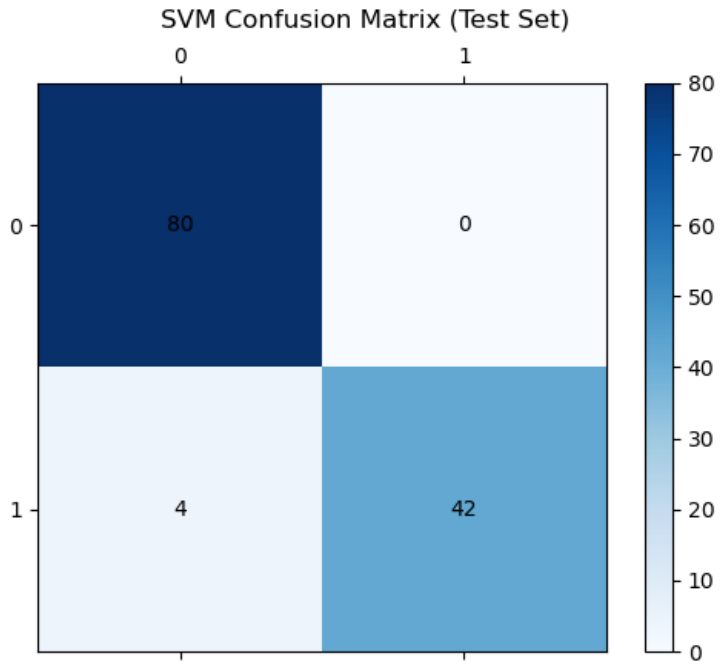

Figura 4.8: Matriz de confusión de los datos de Testeo (SVM)

En la Figura 4.9 se observa el valor de anomalía de cada dato, basado en la distancia al plano que separa los datos en el espacio de dimensionalidad superior respecto a los datos de entrenamiento. Los datos de color verde muestran los datos correctos, mientras que los rojos muestran las anomalías. La línea horizontal roja

muestra el umbral a partir del cual se considera un dato outlier o no. Además, el eje x muestra un identificador para cada dato, es decir, para poder localizar cada dato.

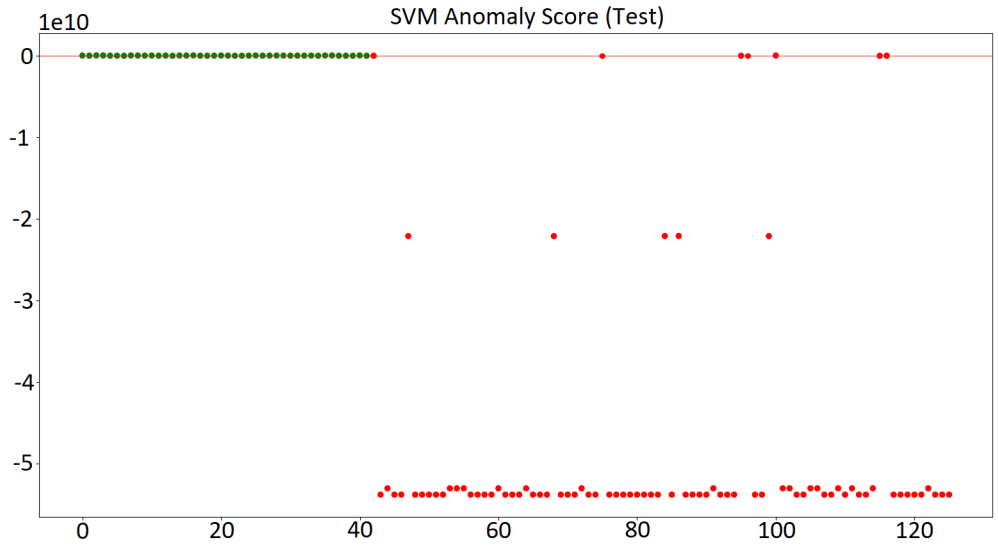

Figura 4.9: Valor de anomalía de los datos de Testeo (One Class SVM)

One Class SVM PCA

En este apartado se muestran los resultados obtenidos a partir de las variables a las que se les ha aplicado un PCA. El modelo aprendido ha sido el One Class SVM.

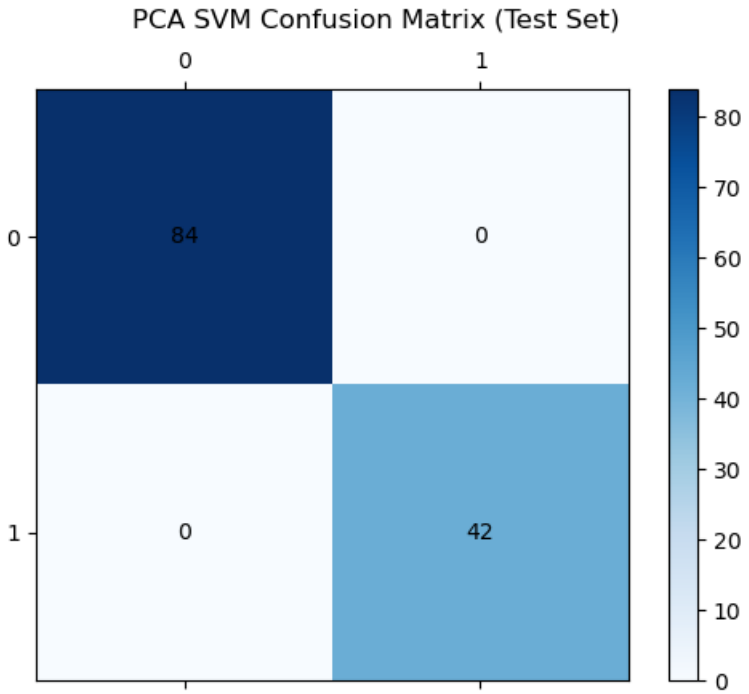

Figura 4.10: Matriz de confusión de los datos de Testeo (PCA SVM)

En la Figura 4.11 se observa el valor de anomalía de cada dato, basado en la distancia al plano que separa los datos en el espacio de dimensionalidad superior respecto a los datos de entrenamiento. Los datos de color verde muestran los datos correctos, mientras que los rojos muestran las anomalías. La línea horizontal roja muestra el umbral a partir del cual se considera un dato outlier o no.

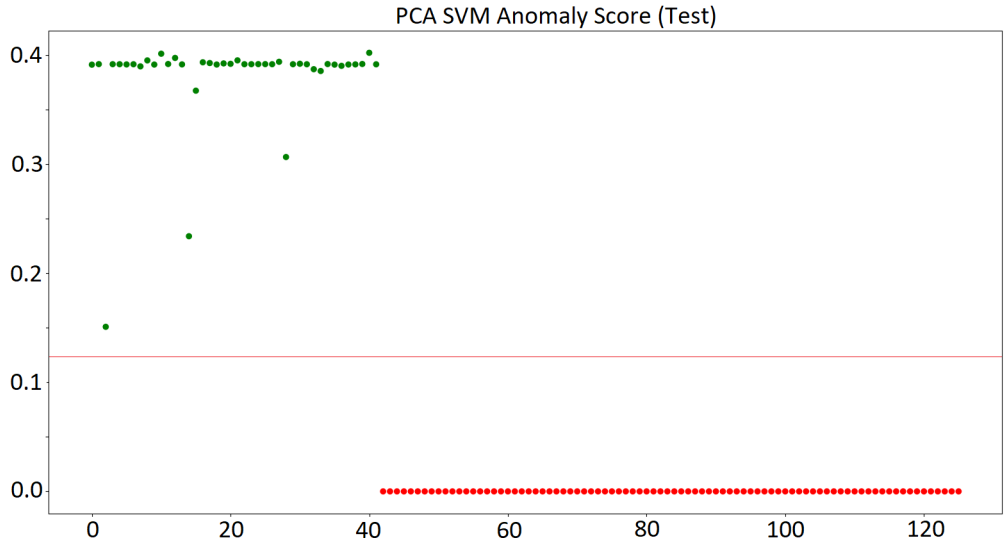

Figura 4.11: Valor de anomalía de los datos de Testeo (PCA SVM)

Isolation Forest

Aprendiendo el modelo con los parámetros mostrados en la Tabla 3.4, se entrenó el modelo con los datos de entrenamiento. Los datos de validación se utilizaron para ajustar dichos parámetros. Finalmente, vemos los resultados en la matriz de confusión mostrada en la Figura 4.12. Se puede observar que de los 42 datos correctos, 3 los ha clasificado como outliers, mientras que 2 de los outliers los ha clasificado como correctos.

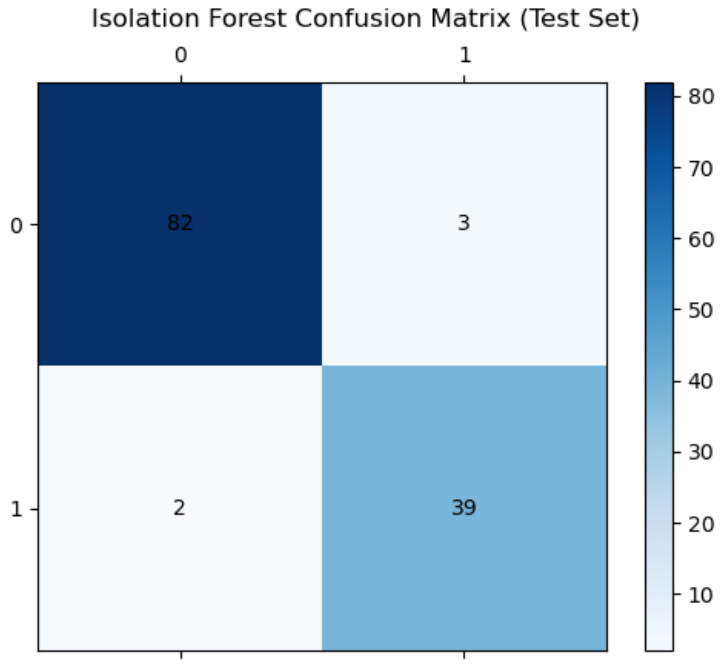

Figura 4.12: Matriz de confusión de los datos de Testeo (Isolation Forest)

En la Figura 4.13 se observa el valor de anomalía de cada dato, basado en el número de cortes necesarios para aislar dicho dato respecto a los datos de entrenamiento. Los datos de color verde muestran los datos correctos, mientras que los rojos muestran las anomalías.

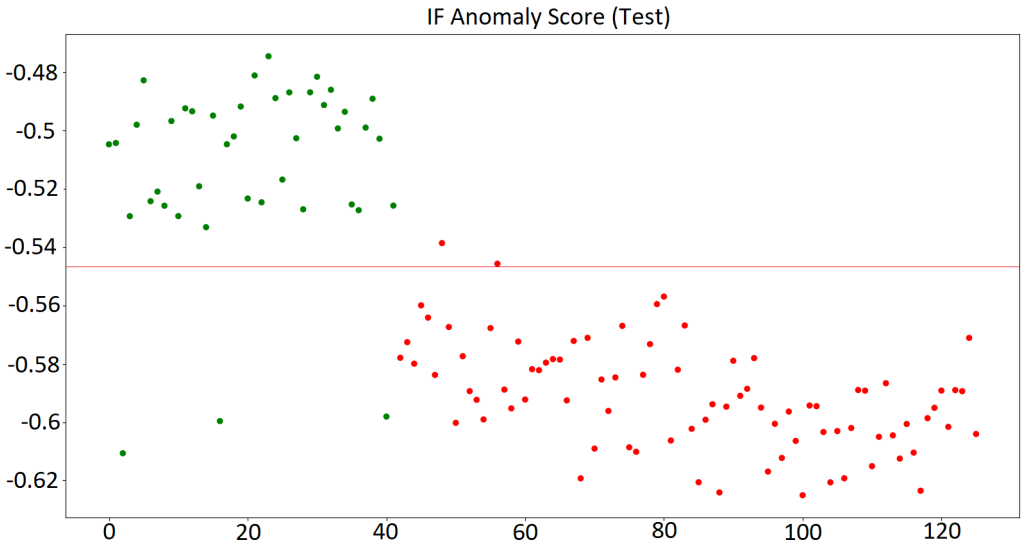

Figura 4.13: Valor de anomalía de los datos de Testeo (Isolation Forest)

Isolation Forest PCA

En este apartado se muestran los resultados obtenidos a partir de las variables a las que se les ha aplicado un PCA. El modelo aprendido ha sido el Isolation Forest.

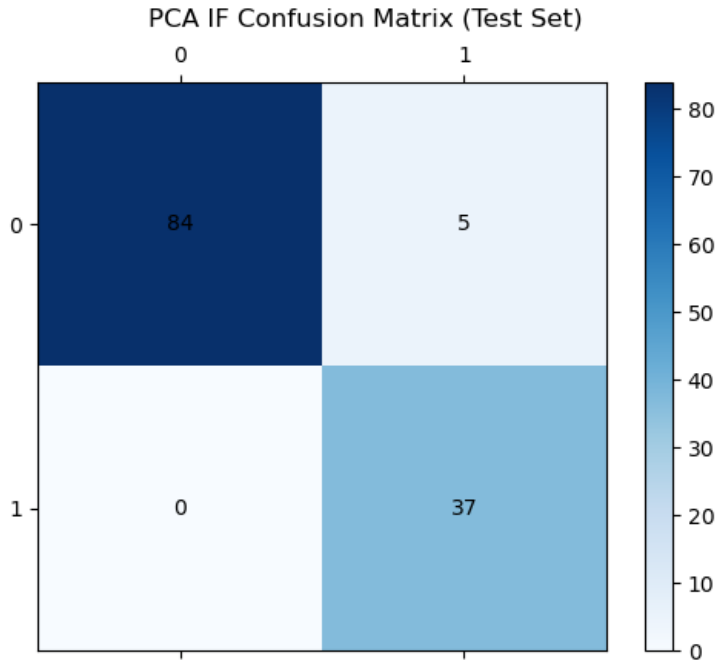

Figura 4.14: Matriz de confusión de los datos de Testeo (Isolation Forest PCA)

En la Figura 4.15 se observa el valor de anomalía de cada dato, basado en el número de cortes necesarios para aislar dicho dato respecto a los datos de entrenamiento. Los datos de color verde muestran los datos correctos, mientras que los rojos muestran las anomalías.

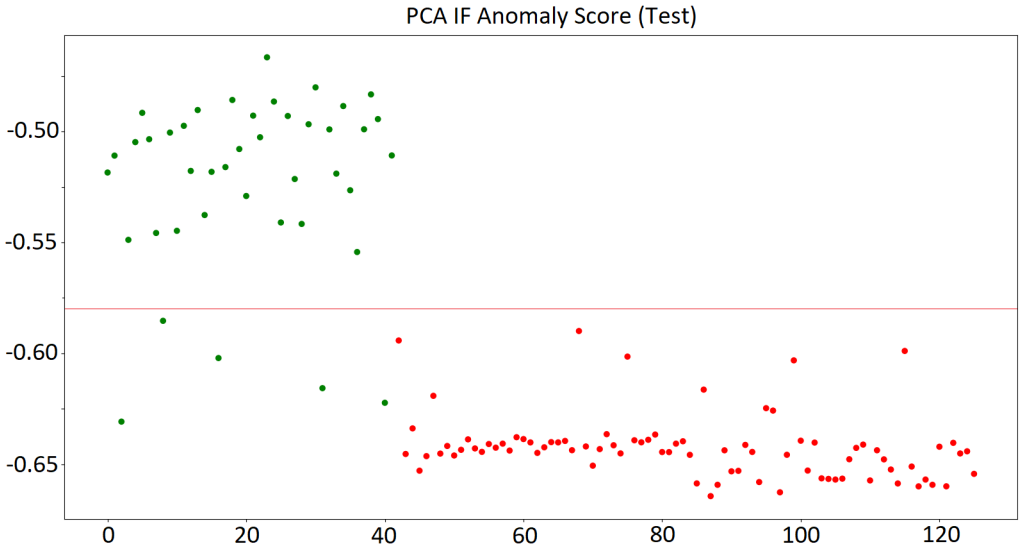

Figura 4.15: Valor de anomalía de los datos de Testeo (Isolation Forest PCA)

Local Outlier Factor

Aprendiendo el modelo con los parámetros mostrados en la Tabla 3.6, se entrenó el modelo con los datos de entrenamiento. Los datos de validación se utilizaron para ajustar dichos parámetros. Finalmente, vemos los resultados en la matriz de confusión mostrada en la Figura 4.16. Se puede observar como todos los datos son clasificados correctamente, tanto los 84 outliers como los 42 inliers.

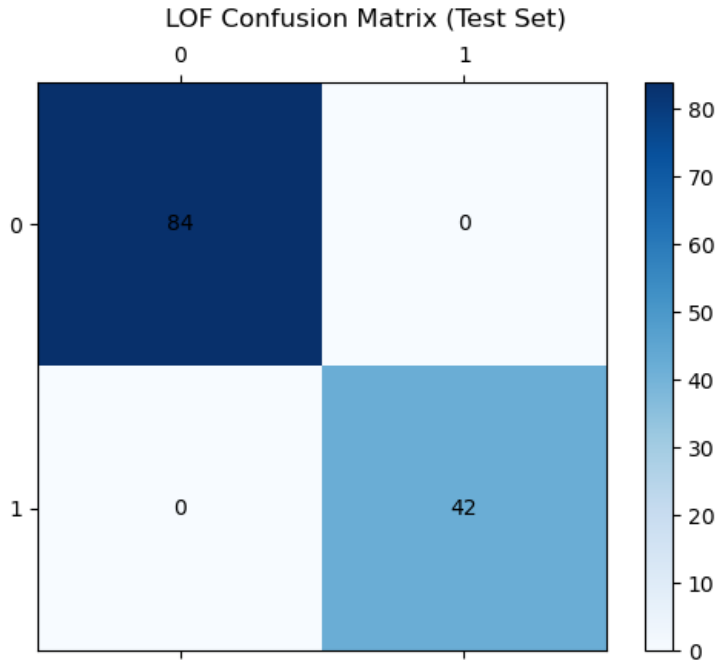

Figura 4.16: Matriz de confusión de los datos de Testeo (LOF)

En la Figura 4.17 se observa el valor de anomalía de cada dato, basado en la densidad de cada dato según su vecindario. Los datos de color verde muestran los correctos, mientras que los rojos muestran las anomalías.

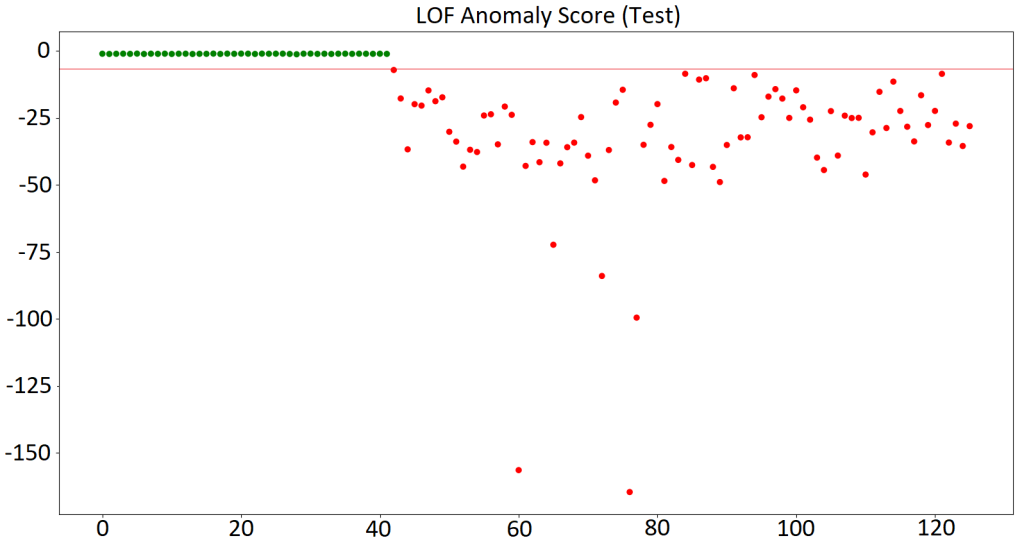

Figura 4.17: Valor de anomalía de los datos de Testeo (LOF)

PCA Local Outlier Factor

En este apartado se muestran los resultados obtenidos a partir de las variables a las que se les ha aplicado un PCA. El modelo aprendido ha sido el Local Outlier Factor.

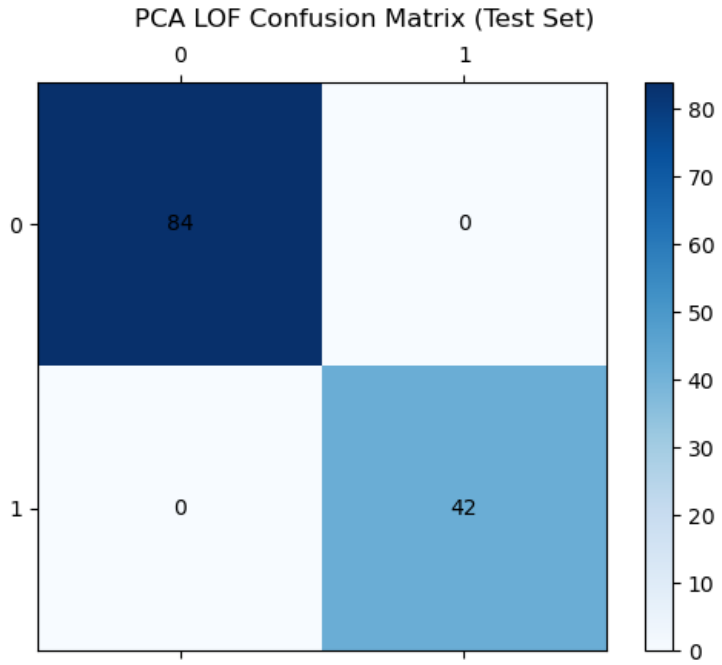

Figura 4.18: Matriz de confusión de los datos de Testeo (LOF PCA)

En la Figura 4.19 se observa el valor de anomalía de cada dato, basado en la densidad de cada dato según su vecindario respecto a los datos de entrenamiento. Los datos de color verde muestran los datos correctos, mientras que los rojos muestran las anomalías.

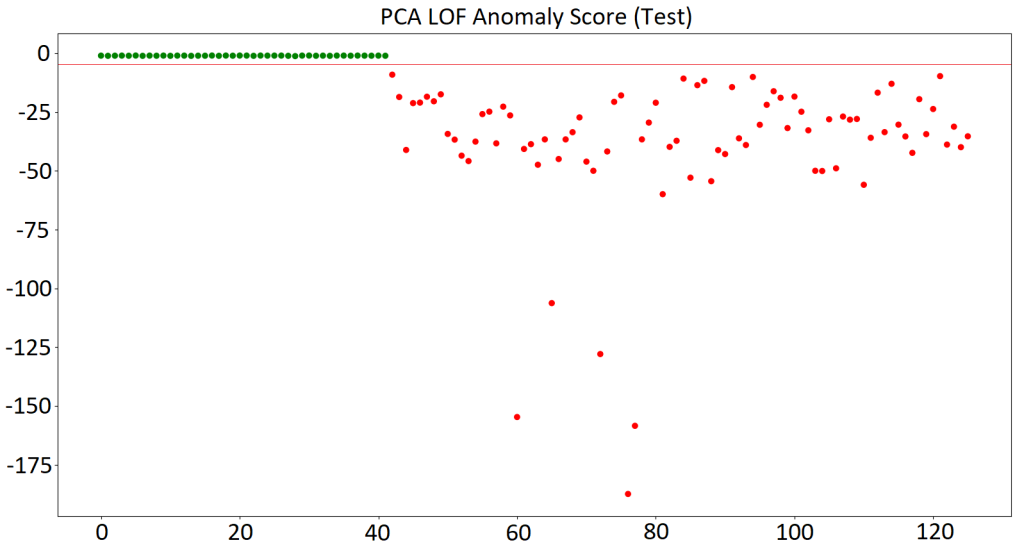

Figura 4.19: Valor de anomalía de los datos de Testeo (LOF PCA)

4.2.2 Deep Learning

Se aprendieron dos modelos distintos empleando Autoencoders. Por un lado se encuentran los Autoencoders de capas densas que recibe como input los datos detallados en la sección 3.4: “Extracción de características automáticas”. Por otro lado, se encuentra un modelo de Autoencoder que recibe como input las imágenes de los datos directamente.

Autoencoder con características automáticas

Una vez entrenado el modelo, obtuvimos una gráfica en la que se mostraba la función de pérdida tanto en el set de entrenamiento como en el de validación. Dicha gráfica se puede observar en la Figura 4.20. El hecho de que la función de pérdida en el set de validación no llegue a bajar tanto como en el de entrenamiento, puede indicar un grado de over-fitting en entrenamiento.

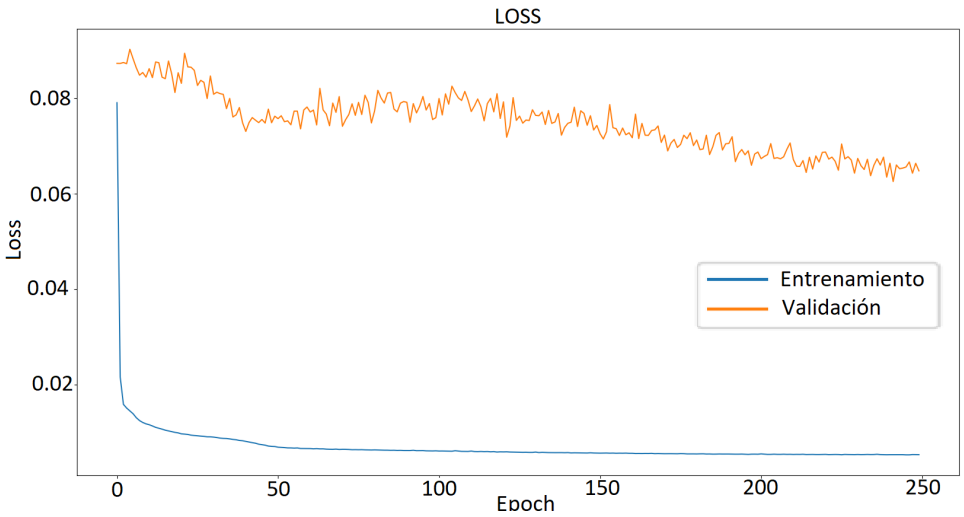

Figura 4.20: Función de pérdida del autoencoder

A la hora de la reconstrucción de los datos, en la Figura 4.21 se puede observar que algunos datos incorrectos son reconstruidos mejor que algunos correctos. Se estableció el umbral de outlierness en 2, como se puede observar en la Figura 4.21.

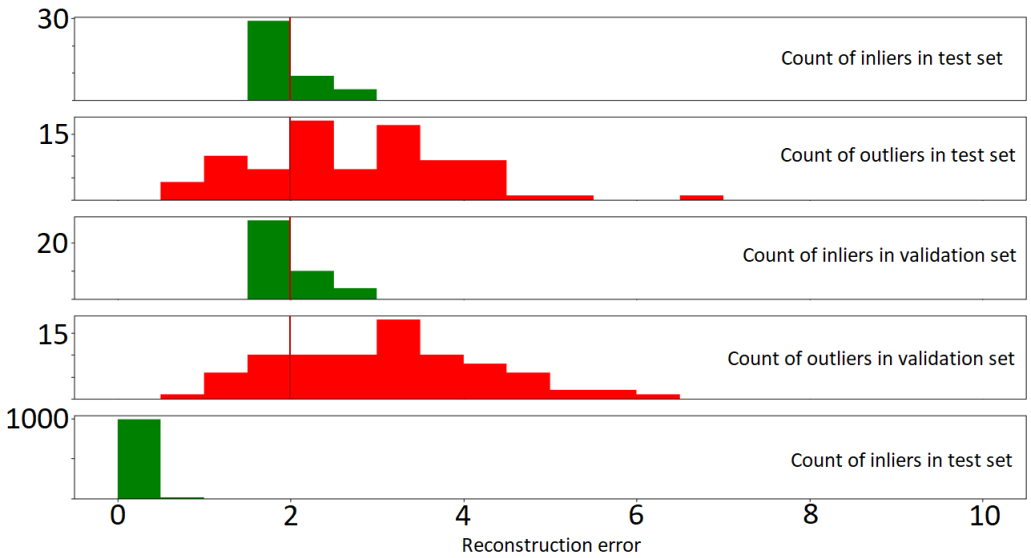

Figura 4.21: Error de reconstrucción de los datos

Finalmente, en la Figura 4.22,se puede observar que el modelo obtiene unos resultados claramente peores que los anteriores modelos.

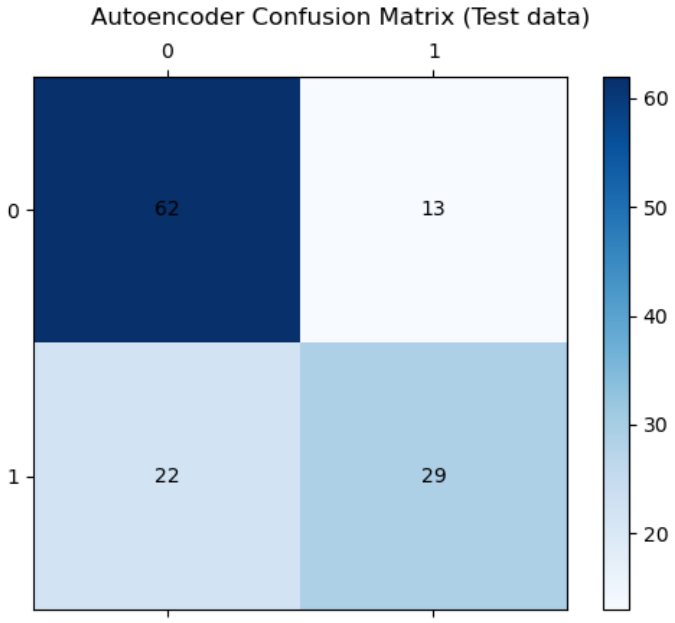

Figura 4.22: Matriz de confusión del Autoencoder

4.3 Explicabilidad de los datos automáticos

Una vez obtenidos los resultados de todos los modelos, se realizó un análisis de los datos, para comprobar su intuitividad y explicabilidad. La idea prevista de los resultados, era que los datos de desalineo o desbalanceo con un valor de anomalía bajo, es decir, los que según el modelo son cercanos a los datos correctos, serían datos practicamente circulares, mientras que los datos más lejanos, serían casos de desalineado extremo.

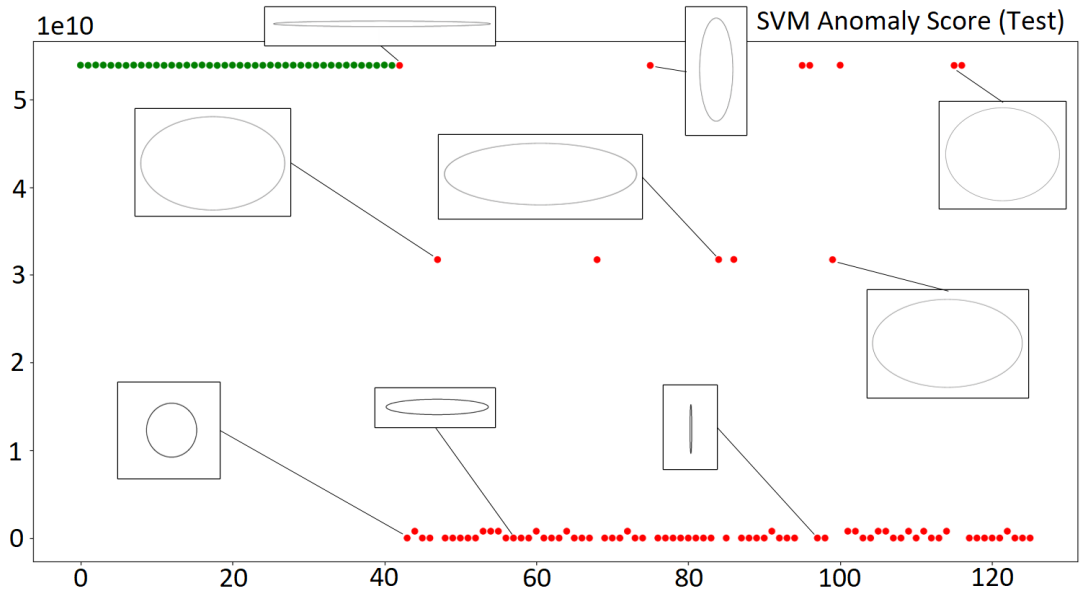

Figura 4.23: Umbral de anomalía de los datos automáticos de testeo (SVM)

Sin embargo, como se puede observar en la Figura 4.23, esto no resultó así, ya que los modelos se estaban fijando en alguna de las características propias de las ondas individuales en vez de en la relación. Los datos más circulares y los más deformes estaban indistintamente situados en la gráfica, y por ello, se contactó con expertos en el área que indicaron cuales serían las variables expertas.

4.4 Aprendizaje empleando las variables expertas

Los expertos en el área recomendaron 3 variables que contenían información del círculo que se forma al juntar ambas ondas, en lugar de las ondas por separado. Dichas variables son las resumidas en la Tabla 4.2.

Cuadro 4.2: Datos expertos

Característica	Descripción	Fórmula
Shape factor	Llamamos Shape Factor a la relación entre el valor mínimo y el valor máximo de "c", siendo "c", la suma de los valores al cuadrado de ambas ondas	$SF = min(c)/max(c)$ donde: $c = x1^2 + x2^2$
Excentricidad	La excentricidad es la razón entre la semidistancia focal con el semieje mayor	$\sqrt{\lvert\frac{(max(x1)^2 - max(x2)^2)}{max(x2)^2}\rvert}$
Proporcionalidad	La proporcionalidad nos indica la elipsoidicidad del dato: cuanto más cercano a 1 sea, más circular será	$Prop = max(x2)/max(x1)$

Tras cambiar las variables, se propuso repetir el trabajo realizado previamente expuesto, esta vez con las variables expertas. Por cuestiones relativas al reducido número de variables expertas, no se pudieron llevar a cabo todos los modelos empleados con las variables automáticas. Los autoencoders dejaron de tener sentido, ya que la información ya estaba lo suficientemente comprimida. El Isolation Forest no fue capaz de generar buenos resultados, por lo que se descartó su uso. Asimismo, la técnica PCA también carecía de utilidad ya que solo se disponía de 3 variables, y una disminución de la dimensionalidad no aportaría nada.

En la Figura 4.24, se puede observar que existe una diferencia notable entre los datos correctos e incorrectos, aunque no entre los datos desalineados y los desbalanceados. Aunque pueda parecer que se trata de una separación clara, esto puede deberse al hecho de que sean datos sintéticos, por lo que a pesar de que pueda parecer fácil establecer un umbral, los expertos decidieron no fijarlo, ya que los datos podrían variar al pasar a la realidad.

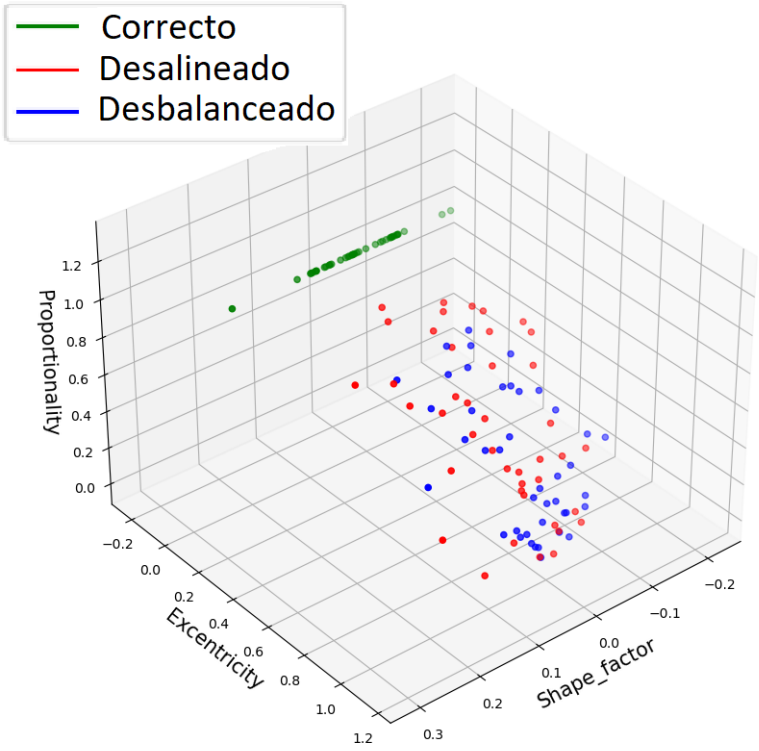

Figura 4.24: Datos caracterizados según sus variables expertas

En la sección 4.5: “Experimentación con los datos expertos”, se analizarán los resultados obtenidos con las variables automáticas.

4.5 Experimentación con los datos expertos

4.5.1 One Class SVM

El modelo de Suport Vector Machine se entrenó con el dataset de entrenamiento y se ajustó con el dataset de validación. En este caso, observando los datos de validación, se estableció un umbral a partir del cuál se considerará un valor anómalo o no. Mostrando los valores de testeo en la Figura 4.25, se observa que encajan perfectamente.

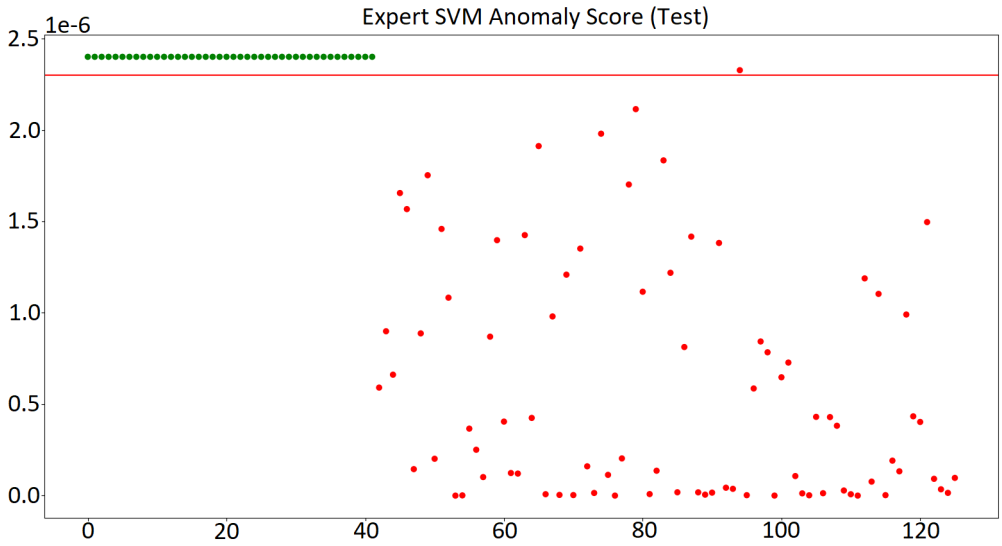

Figura 4.25: Umbral de anomalía de los datos expertos de testeo (SVM)

Mostrando la Matriz de Confusión de estos resultados en la Figura 4.26, se puede observar que los resultados son muy buenos, únicamente hay un dato que se clasifica incorrectamente.

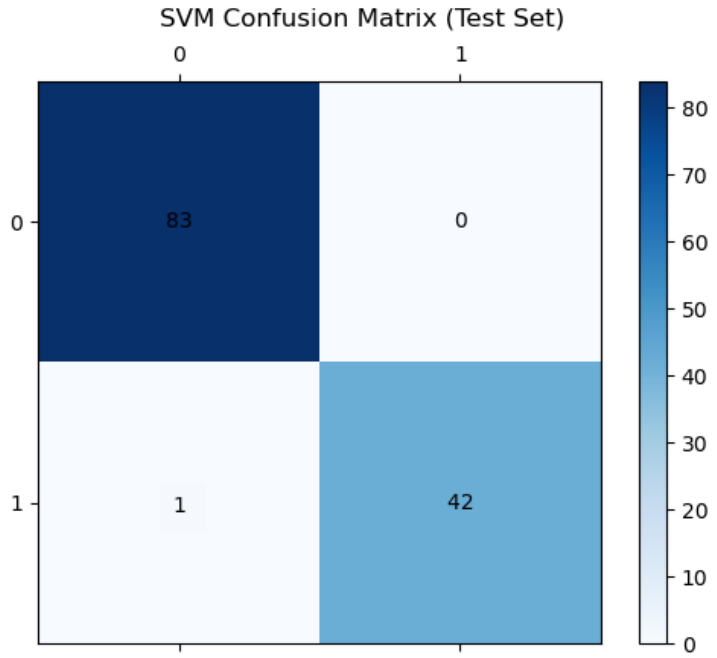

Figura 4.26: Matriz de confusión de los datos expertos de Testeo (SVM)

4.5.2 Local Outlier Factor (LOF)

De nuevo con el modelo LOF, se estableció un umbral a partir del cuál se consideraría un valor anómalo o no. Mostrando los valores de testeo en la Figura 4.27, se observa que encajan perfectamente.

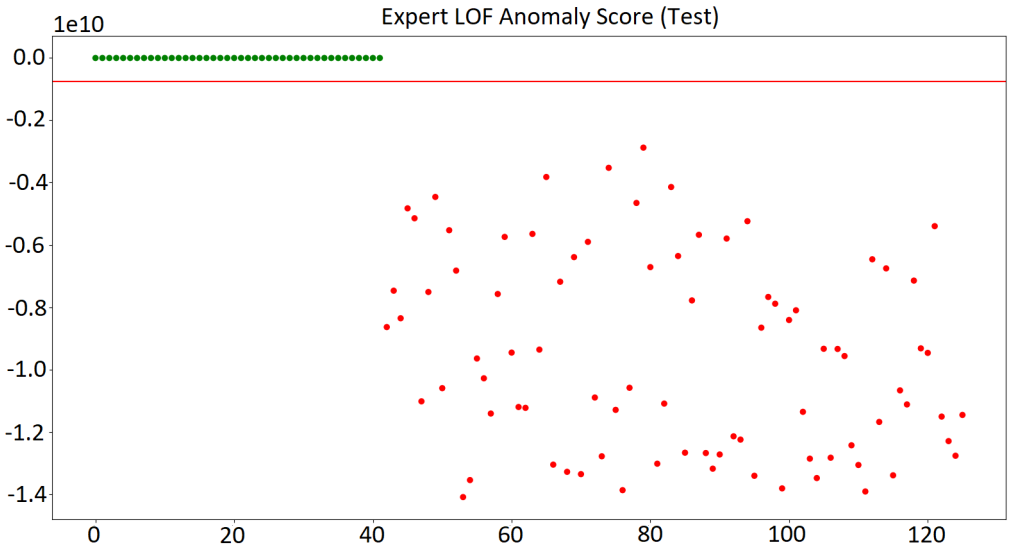

Figura 4.27: Umbral de anomalía de los datos expertos de Testeo (LOF)

Mostrando la Matriz de Confusión de estos resultados en la Figura 4.28, se puede observar que los resultados son óptimos.

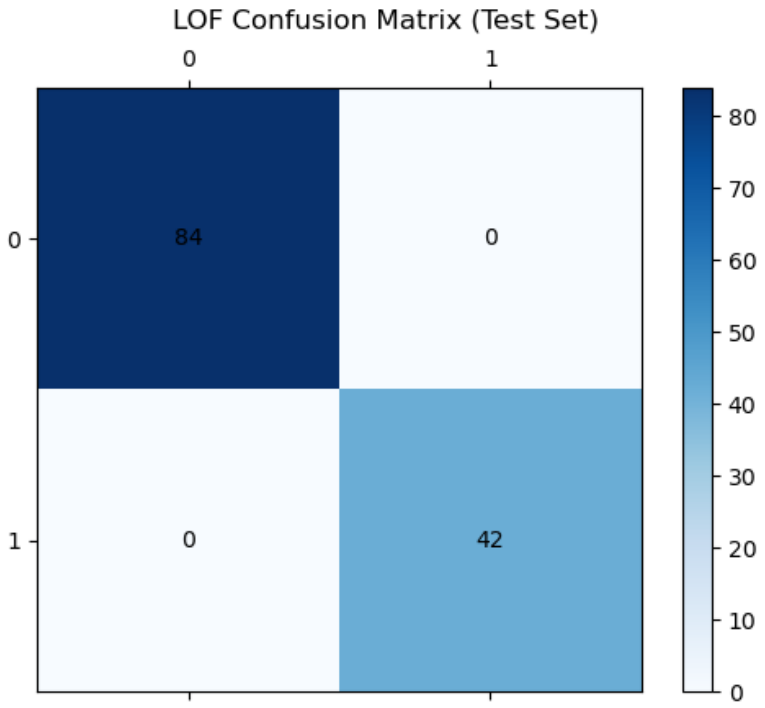

Figura 4.28: Matriz de confusión de los datos expertos de Testeo (LOF)

4.6 Explicabilidad de los datos expertos

En este caso los resultados obtenidos sí que resultaron ser intuitivos. Como se puede observar en la Figura 4.29, los datos cercanos a los valores correctos, son valores que a simple vista se diría que son prácticamente correctos, mientras que los más lejanos son claramente casos extremos.

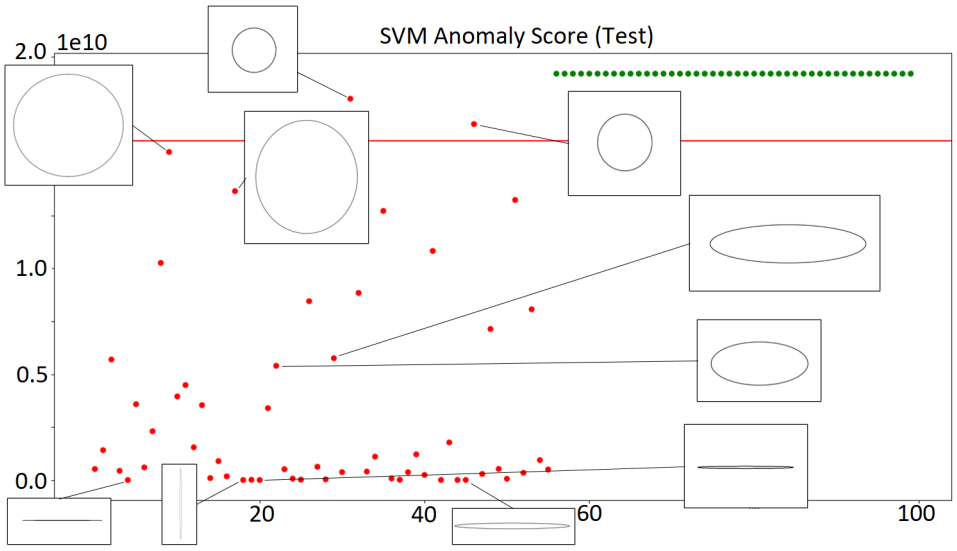

Figura 4.29: Valores de anomalía de los datos (SVM)

Éste es un hecho realmente importante, ya que teniendo en cuenta los resultados de las variables automáticas, se podía indicar si el estado del eje era correcto o incorrecto, mientras que empleando los resultados obtenidos mediante las variables expertas, además de si el estado del eje es correcto o incorrecto, se puede indicar en qué medida está fallando, añadiendo así una capa extra de información y explicabilidad.

Por ello, para la validación de los modelos realizada en la sección 4.8: "Metodología para la obtención de resultados", se utilizarán exclusivamente las variables expertas.

4.7 Autoencoder convolucional

Para el aprendizaje del Autoencoder Convolucional, se seleccionó un dataset de 800 datos de entrenamiento, 100 de validación y 100 de testeo, y se entrenó el Autoencoder explicado en la sección 3.5.2: "Autoencoder convolucional". Como se puede observar en la Figura 4.30, tanto el valor de pérdida del set de entrenamiento como del de validación decrecen de manera similar.

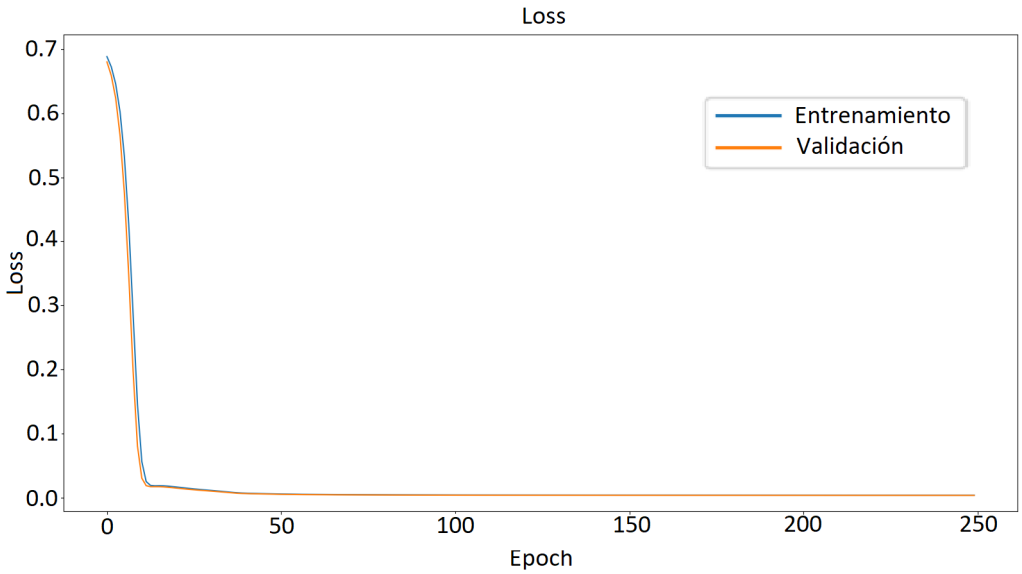

Figura 4.30: Función de pérdida del Autoencoder Convolucional

Cada capa oculta extrae ciertas características, resultado de aplicar un kernel a la imagen. En este caso tendremos una primera capa oculta de 16 filtros y un tamaño de 224x224. Un ejemplo de la extracción de características de uno de los datos correctos se puede observar en la Figura 4.31.

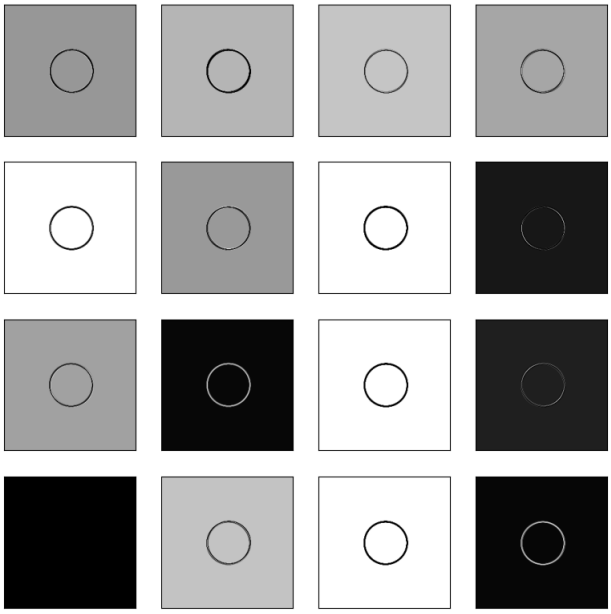

Figura 4.31: Primera capa convolucional

La segunda capa oculta se compone de 8 filtros y un tamaño de 112x112. Un ejemplo se puede observar en la Figura 4.32.

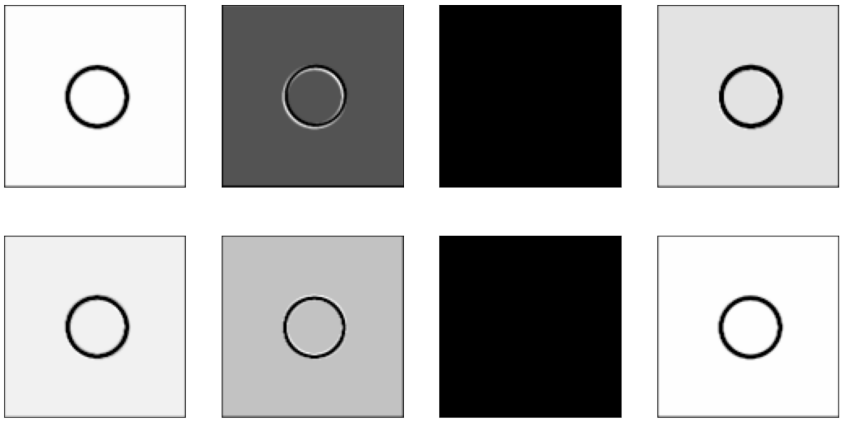

Figura 4.32: Segunda capa convolucional

Finalmente, la tercera capa oculta se compone de 16 filtros y un tamaño de 224x224. Un ejemplo se puede observar en la Figura 4.33.

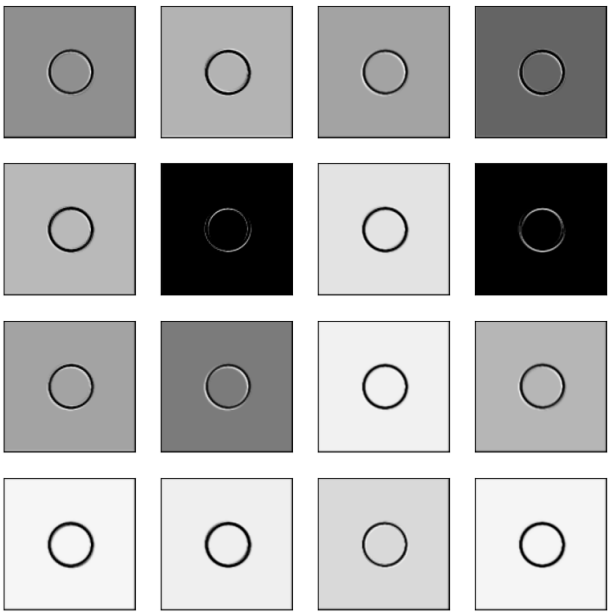

Figura 4.33: Tercera capa convolucional

Una vez se ha entrenado el modelo y las capas convolucionales son capaces de extraer características, el modelo es capaz de reconstruir las imágenes, tal y como se puede ver en la Figura 4.34 para uno de los datos correctos y en la Figura 4.35 para uno de los datos incorrectos

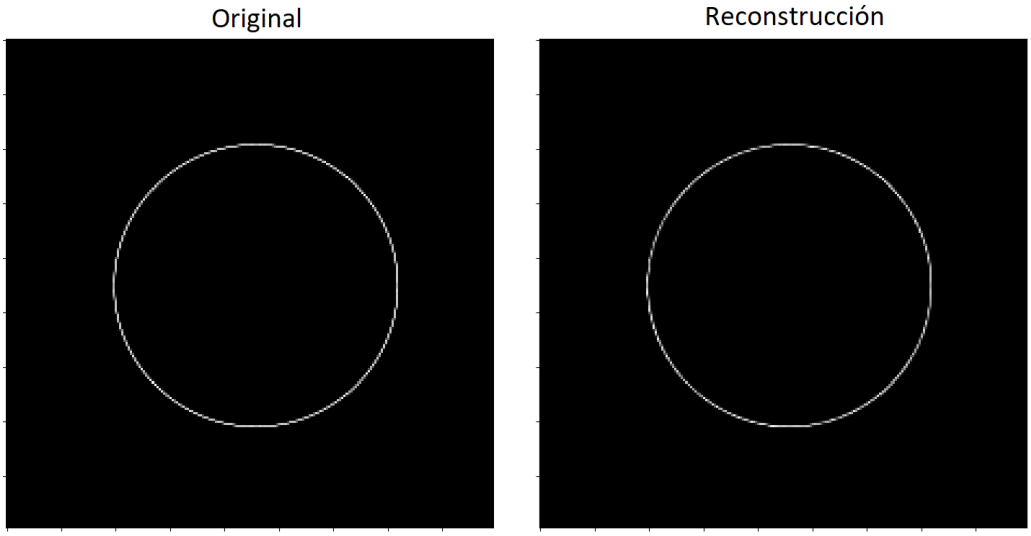

Figura 4.34: Reconstrucción de un dato correcto

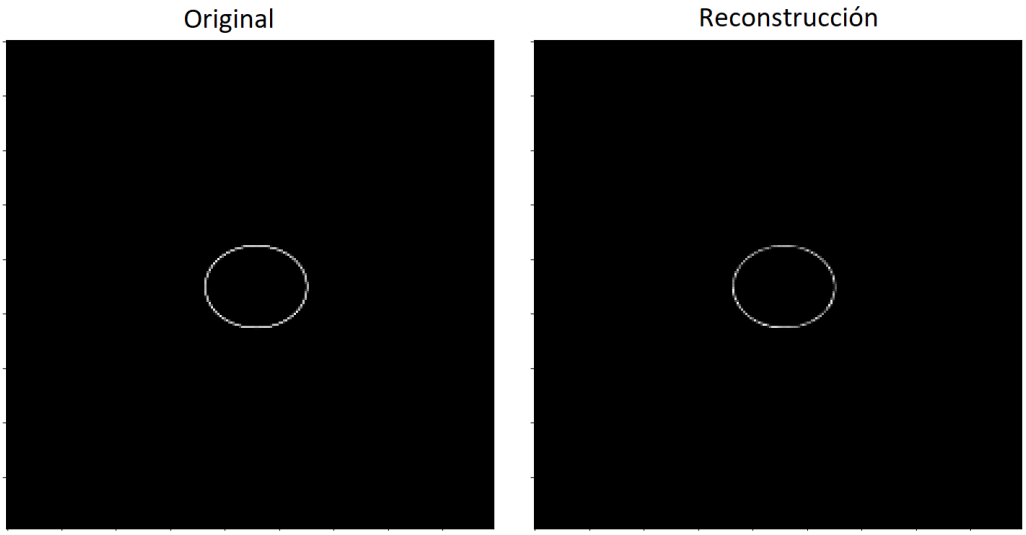

Figura 4.35: Reconstrucción de un dato incorrecto

Para comprobar el nivel de fidelidad del dato original respecto a la predicción se empleó el método de MSE. En la Figura 4.36 se puede observar el valor de reconstrucción de cada dato. Se puede observar que no hay una clara manera de separar ambos grupos, ya que la calidad de las reconstrucciones de todos los tipos de datos son similares.

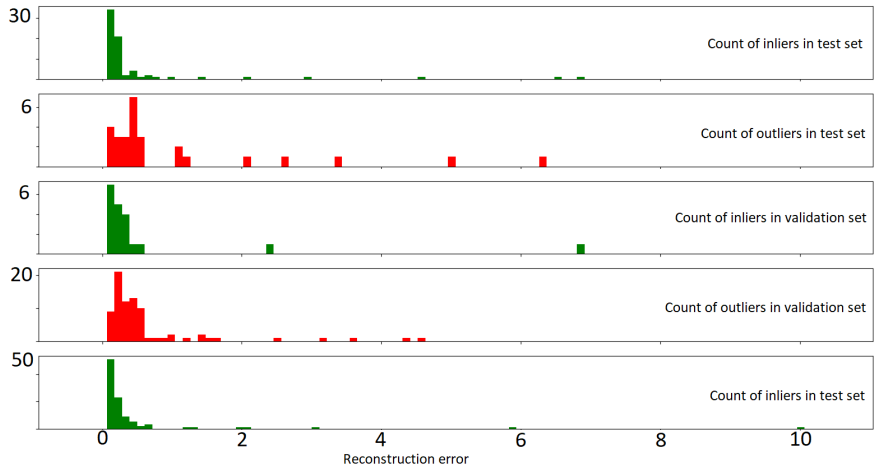

Figura 4.36: Reconstrucción de los datasets

En el set de validación, los datos peor reconstruidos son los mostrados en la Figura 4.37, mientras que los mejores son los mostrados en la Figura 4.38.

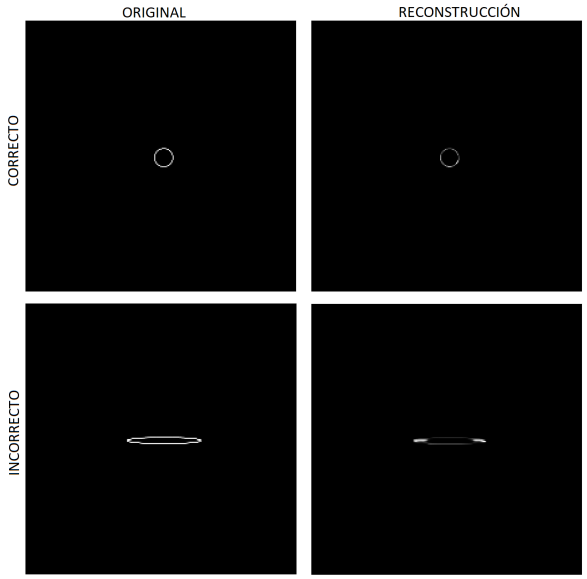

Figura 4.37: Datos peor reconstruidos

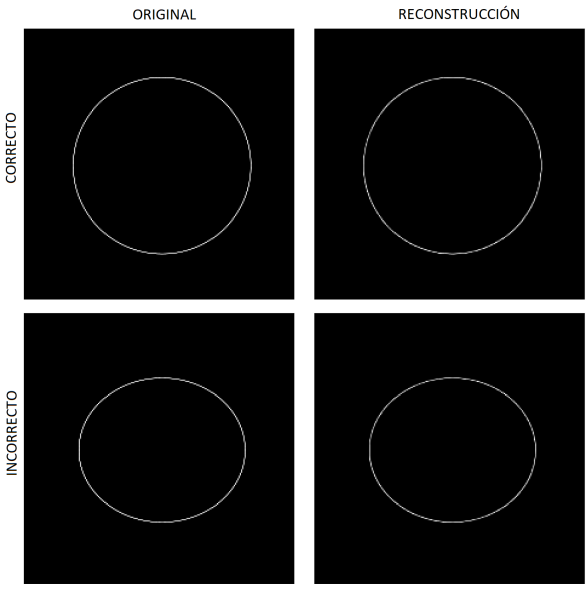

Figura 4.38: Datos mejor reconstruidos

El principal problema observado es que no ha sido posible la reconstrucción correcta de los círculos e incompleta de los semicírculos, esto es debido a que variando tanto el número de capas a emplear como el número de filtros, o bien ambos datos se reconstruían correctamente o bien ambos se reconstruían erróneamente. Es importante destacar que los tiempos de ejecución de este modelo han sido más elevados respecto a los modelos anteriores, lo cuál puede ser un factor importante a tener en cuenta.

4.8 Metodología para la obtención de resultados

Para obtener unos valores más fiables, se ejecutaron 50 veces los algoritmos con diferentes muestras y proporciones. Para ello, se crearon dos bolsas con datos. La primera de un tamaño de 10000 datos, todos ellos correctos. La segunda, de 1000, la mitad siendo datos desbalanceados y la mitad restante datos desalineados. Estos tamaños tan grandes han servido para aumentar la variabilidad en los muestras empleadas durante las iteraciones.

En cada iteración, se escogen aleatoriamente 800 de los 10000 datos correc-

tos para crear el set de entrenamiento en formato One-Class. Posteriormente se escogen dos números aleatorios entre 5 y 95, con los que se eligen el número de datos incorrectos que habrá por un lado en el set de validación, y por otro lado en el set de entrenamiento. Finalmente, ya que cada uno de estos sets debe tener un tamaño de 100 (para mantener la proporción 80 %, 10 %, 10 %), la diferencia se llena con datos aleatorios de la bolsa de los correctos. De este modo, existe una gran variabilidad entre iteraciones.

Una vez se generan los datasets en cada iteración, se ejecutan los algoritmos y se calcula la precisión, el ratio de falsos positivos, el ratio de falsos negativos y el umbral de outlierness que se establece en cada iteración. El umbral se establece automáticamente en el set de validación, estableciéndolo en el dato anómalo con un valor de anomalía más bajo. El dataset de testeo se emplea únicamente para calcular la precisión, el ratio de falsos positivos y el ratio de falsos negativos, pero nunca para establecer el umbral. En la Figura 4.39 se observa cómo el valor menos anómalo se establece como umbral, mientras que en la Figura 4.40 se observa el resultado de aplicar dicho umbral en el set de test. Para el caso de los Autoencoders, al estar los datos entremezclados, se seleccionó el umbral como la moda de los datos correctos en el set de validación.

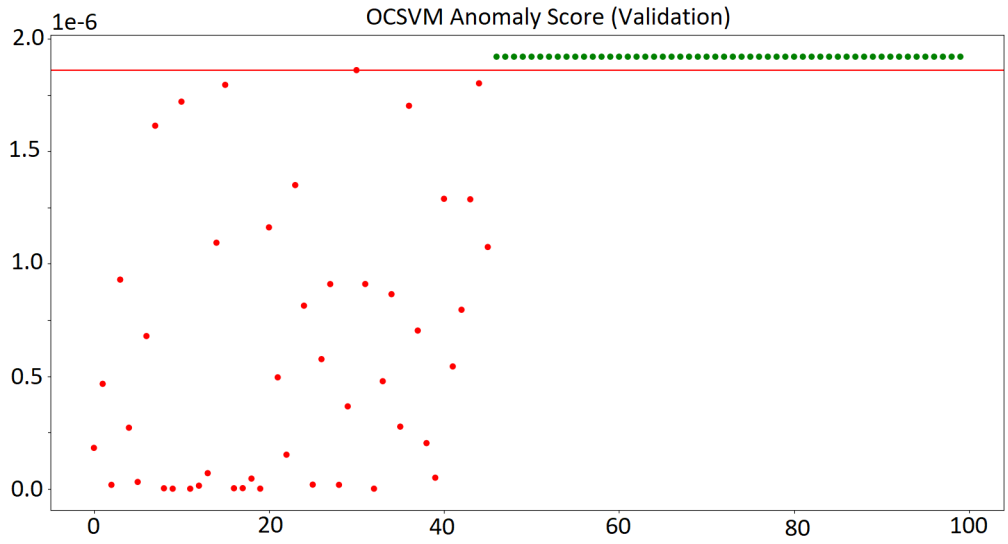

Figura 4.39: Valores de anomalía en el set de validación

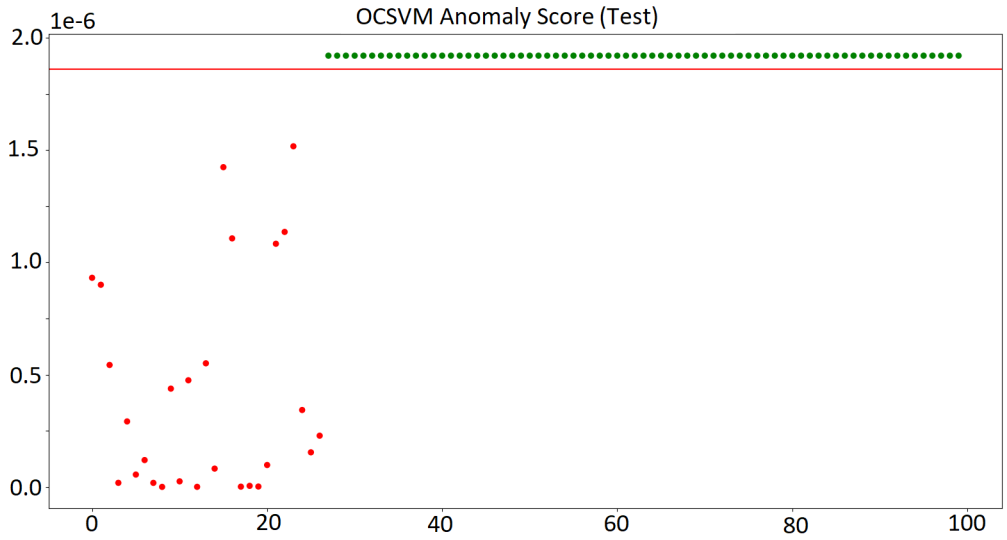

Figura 4.40: Valores de anomalía en el set de testeo

4.9 Resultados

Una vez realizadas dichas iteraciones, se calculó la media de las métricas que calculábamos. Se observa que debido al cambio en la selección del umbral para este apartado, sumado a la variabilidad de emplear distintos datasets en 50 iteraciones distintas, varios de los resultados son peores.

Cuadro 4.3: Resultados de los modelos

Modelo	Precisión	FP	FN	Umbral	Desviación estándar del umbral	Desviación estándar de la precisión
OCSVM (Variables automáticas)	81 %	1.32 %	17.68 %	4.11e10	7.22e09	0.1906
LOF (Variables automáticas)	96.74 %	1.26 %	2 %	-3.654	3.189	0.05557
IF (Variables automáticas)	73.96 %	23.54 %	2.5 %	-0.586	0.023	0.2035
PCA - OCSVM (Variables automáticas)	63.16 %	1.9 %	34.94 %	1.11e09	2.51e09	0.2916
PCA - LOF (Variables automáticas)	96.92 %	1.28 %	1.8 %	-3.973	3.181	0.0560
PCA - IF (Variables automáticas)	87.64 %	10.2 %	2.16 %	-0.597	0.030	0.1521
OCSVM (Variables Expertas)	98.18 %	1.82 %	0 %	1.66e-06	3.07e-07	0.0398
LOF (Variables Expertas)	98.26 %	1.74 %	0 %	-2.71e09	1.79e08	0.0392
Autoencoder (Variables automáticas)	66.46 %	14.18 %	19.36 %	5.775	0.572	0.0957
Autoencoder convolucional	56.44 %	41.86 %	0.7 %	-0.9996	0.01	0.2232

4.10 Criterio

A la hora de comparar los datos, es importante aclarar qué métrica es más importante. La precisión es una gran forma de saber a rasgos generales qué método es más certero a la hora de clasificar los datos. Sin embargo, cometer falsos positivos (FP) es más grave en este caso que cometer falsos negativos (FN). Esto es debido a que en el caso de un falso negativo, se ordenaría realizar un mantenimiento y se comprobaría que todo funcionaba correctamente. Mas, en el caso de un falso positivo, la maquinaria continuaría funcionando sin saber que un fallo puede estar causando daños.

Si bien esto puede ser cierto, también puede ser preferible ser peor en las métricas, siempre y cuando se puedan explicar los resultados.

4.11 Conclusiones

Para concluir, tomando en cuenta la explicabilidad de los datos y no solo la precisión, se ha escogido el Local Outlier Factor (LOF) con las variables expertas como el modelo que mejor lleva a cabo la tarea de clasificación de fallos mediante One Class Classification. Por una parte, es el modelo que mayor precisión alcanza, teniendo además un valor bajo de desviación estándar en las precisiones. Por otra parte, a pesar de que los fallos que produce son falsos positivos, esto carece de la importancia que tendría con las variables automáticas, ya que tal y como se ha explicado en la sección 4.5: "Experimentación con los datos expertos", los datos que son incorrectos clasificados erróneamente son datos muy parecidos a los correctos. En ese caso, indicar que el dato es correcto cuando la etiqueta indica lo contrario no es una situación conflictiva en la realidad, ya que se puede indicar cómo de correcto o incorrecto es el dato.

Además de este modelo, sería muy seguro emplear el One Class Support Vector Machine (OCSVM) por las mismas razones, ya que su precisión es ligeramente más baja, una diferencia apenas perceptible.

Respecto al resto de modelos, en estrictos términos de clasificación, podría emplearse con cierto grado de fiabilidad el LOF con variables Automáticas o el PCA-LOF, sin embargo, para un caso real, la falta de explicabilidad de los resultados hace que se pierda parte de la información. Esto causa que la al clasificar un dato incorrecto como correcto no se conozca el grado de deformidad del círculo.

Finalmente, no se recomienda el uso del resto de modelos, ya que se han obte-

nido resultados altamente inferiores no siendo capaces de ofrecer ninguna ventaja frente al resto.

Bibliografía

[1] Kemal Erdem, Towards Data Science. t-SNE clearly explained, 2020. https://towardsdatascience.com/t-sne-clearly-explained-d84c537f53a#:~:text=t%2DSNE%20is%20mostly%20used,when%20dealing%20with%20CNN%20networks.

[2] Zach Bedell, Medium. Support Vector Machines explained. https://medium.com/@zachary.bedell/support-vector-machines-explained-73f4ec363f13.

[3] Grace Zhang, Medium. What is the kernel trick? why is it important?, 2018. https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d.

[4] Akshara, Analytics Vidhya. Anomaly detection using isolation forest – a complete guide. https://www.analyticsvidhya.com/blog/2021/07/anomaly-detection-using-isolation-forest-a-complete-guide/#:~:text=In%20an%20Isolation%20Forest%2C%20randomly,more%20cuts%20to%20isolate%20them.

[5] Wikipedia. Kernel density estimation. https://en.wikipedia.org/wiki/Kernel_density_estimation.

[6] Julian Wergieluk, Towards Data Science. Histograms vs. kdes explained, 2020. https://towardsdatascience.com/histograms-vs-kdes-explained-ed62e7753f12.

[7] Vaibhav Jayaswal, Towards Data Science. Local outlier factor (lof) — algorithm for outlier identification, 2020. https://towardsdatascience.com/local-outlier-factor-lof-algorithm-for-outlier-identification-8efb887d9843.

[8] Scikit-learn, a Python library for Machine Learning. Outlier detection with local outlier factor (lof). https://scikit-learn.org/stable/auto_examples/neighbors/plot_lof_outlier_detection.html.

[9] Arden Dertat, Towards Data Science. Applied deep learning - part 3: Autoencoders. https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798.

[10] Github. Convolutional neural networks (cnns / convnets). https://cs231n.github.io/convolutional-networks/.

[11] Arthurmeyer, Github. Saliency detection convolutional autoencoder. https://github.com/arthurmeyer/Saliency_Detection_Convolutional_Autoencoder.

[12] José Juan Carbajal-Hernández, Luis P. Sánchez-Fernández, Ignacio Hernández-Bautista, José de J. Medel-Juárez, and Luis A. Sánchez-Pérez. Classification of unbalance and misalignment in induction motors using orbital analysis and associative memories. *Neurocomputing*, 175:838–850, 2016.

[13] Toshihiko Yamaguchi, Yoshiro Iwai, Shigeru Inagaki, and Masahiro Ueda. A method for detecting bearing wear in a drain pump utilizing an eddy-current displacement sensor. *Measurement*, 33(3):205–211, 2003.

[14] R. Keith Mobley. 6 - predictive maintenance techniques. In R. Keith Mobley, editor, *An Introduction to Predictive Maintenance (Second Edition)*, Plant Engineering, pages 99–113. Butterworth-Heinemann, Burlington, second edition edition, 2002.

[15] Great Britain. Quality Management, Statistics Standards Committee, and British Standards Institution. *British Standard Glossary of Maintenance Management Terms in Terotechnology*. British Standards Institution, 1984.

[16] Yuanhang Wang, Chao Deng, Jun Wu, Yingchun Wang, and Yao Xiong. A corrective maintenance scheme for engineering equipment. *Engineering Failure Analysis*, 36:269–283, 2014.

[17] Diego Galar and Uday Kumar. *EMaintenance: Essential Electronic Tools for Efficiency*. Academic Press, 2017.

[18] Krishna B. Misra. *Maintenance Engineering and Maintainability: An Introduction*, pages 755–772. Springer London, London, 2008.

[19] Przemyslaw Moczko Zhaklina Stamboliska, Eugeniusz Rusiński. *Proactive Condition Monitoring of Low-Speed Machines*. Springer International, 2015.

[20] Thyago P. Carvalho, Fabrízzio A. A. M. N. Soares, Roberto Vita, Roberto da P. Francisco, João P. Basto, and Symone G. S. Alcalá. A systematic literature review of machine learning methods applied to predictive maintenance. *Computers &amp Industrial Engineering*, 137:106024, nov 2019.

[21] Maurizio Lenzerini. Data integration: a theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '02, pages 233–246, New York, NY, USA, 2002. ACM.

[22] Hans-Walter Bandemer. *Mathematics of Uncertainty: Ideas, Methods, Application Problems*, volume 189. Springer, 01 2006.

[23] Yaguo Lei. *Intelligent fault diagnosis and remaining useful life prediction of rotating machinery*. Butterworth-Heinemann, 2016.

[24] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[25] Poojan Oza and Vishal M. Patel. One-class convolutional neural network. *CoRR*, abs/1901.08688, 2019.

[26] Makoto Takamatsu, Medium. Deep one class classification, 2020. `https://medium.com/analytics-vidhya/paper-summary-deep-one-class-classification-doc-adc4368af75c`.

[27] Roemer's Blog. Introduction to one-class support vector machines. `http://rvlasveld.github.io/blog/2013/07/12/introduction-to-one-class-support-vector-machines/#:~:text=One%2DClass%20SVM%20according%20to%20Sch%C3%B6lkopf&text=basically%20separates%20all%20the%20data,density%20of%20the%20data%20lives.`

[28] Bernhard Schölkopf, Robert Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, page 582–588, Cambridge, MA, USA, 1999. MIT Press.

[29] Vinicius Augusto Diniz Silva and Robson Pederiva. Fault detection in induction motors based on artificial intelligence. 2013.

[30] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[31] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[32] Gian Antonio Susto, Alessandro Beghi, and Seán McLoone. Anomaly detection through on-line isolation forest: An application to plasma etching. In *2017 28th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, pages 89–94, 2017.

[33] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-based anomaly detection. *ACM Trans. Knowl. Discov. Data*, 6(1), mar 2012.

[34] Longin Jan Latecki, Aleksandar Lazarevic, and Dragoljub Pokrajac. Outlier detection with kernel density functions. In Petra Perner, editor, *Machine Learning and Data Mining in Pattern Recognition*, pages 61–75, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[35] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM, 2000.

[36] Hehe Ma, Yi Hu, and Hongbo Shi. Fault detection and identification based on the neighborhood standardized local outlier factor method. *Industrial & Engineering Chemistry Research*, 52:2389–2402, 01 2013.

[37] Zengan Gao. Application of cluster-based local outlier factor algorithm in anti-money laundering. *2009 International Conference on Management and Service Science*, pages 1–4, 2009.

[38] Chong Zhou and Randy C. Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 665–674, New York, NY, USA, 2017. Association for Computing Machinery.

[39] Benjamin, Medium. Anomaly detection using pytorch autoencoder and mnist, 2022. `https://benjoe.medium.com/anomaly-detection-using-pytorch-autoencoder-and-mnist-31c5c2186329`.

[40] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular Object Retrieval With Integral Max-Pooling of CNN Activations. In *ICLR 2016 - International Conference on Learning Representations*, International Conference on Learning Representations, pages 1–12, San Juan, Puerto Rico, May 2016.

[41] Maximilian Christ. TSfresh, a Python library for systematic feature extraction from time-series data, 2016. `https://tsfresh.readthedocs.io/en/latest/index.html`.

[42] Sanskar wagavkar, Medium. Correlation matrix. https://medium.com/analytics-vidhya/correlation-matrix-5e764bcee34.

[43] Steven M Holland. Principal components analysis (pca). *Department of Geology, University of Georgia, Athens, GA*, pages 30602–2501, 2008.