DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL

UNIVERSIDAD DE GRANADA

PROGRAMA DE DOCTORADO EN TECNOLOGÍAS DE LA INFORMACIÓN Y LA COMUNICACIÓN

ALGORITMOS DE DETECCIÓN DE ANOMALÍAS Y MITIGACIÓN DE FALSOS POSITIVOS EN ENTORNOS BIG DATA

Memoria presentada por

DAVID LÓPEZ PRETEL

DIRECTORES

JULIÁN LUENGO MARTÍN
DIEGO JESÚS GARCÍA GIL

Granada, Febrero 2024

Editor: Universidad de Granada. Tesis Doctorales
Autor: David López Pretel
ISBN: 978-84-1195-285-9
URI: https://hdl.handle.net/10481/91107

DECLARACIÓN

El doctorando **David López Pretel** y los directores de la tesis **Julián Luengo Martín** y **Diego Jesús García Gil**

Garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Granada, Febrero 2024

El doctorando: Los directores:

Fdo: David López Fdo: Julián Luengo Fdo: Diego Jesús García

Lo que hoy ha empezado como novela de ciencia ficción, mañana será terminado como reportaje.

ARTHUR CHARLES CLARKE

AGRADECIMIENTOS

Esta sección está dedicada a todas esas personas que me han servido de guía y apoyo ya no solo para realizar esta tesis doctoral sino para disfrutar y seguir creciendo poco a poco en la tarea de aprender y expandir el conocimiento.

Especial dedicación a mis directores, Julián y Diego, por toda la ayuda tanto profesional como personal que me han brindado. También, me gustaría hacer especial mención a Salva, el cual me introdujo en el mundo de la investigación.

Quiero agradecer a Néstor, Germán, Miriam, Juanjo, Celia, Gerardo y Diego I. por todos los años de universidad y máster trabajando y creciendo juntos como ingenieros informáticos. En el ámbito profesional a Nacho, Jacinto y Diego G. con los cuales he trabajado durante los proyectos realizados durante mi etapa como investigador.

Por último, he de mandar agradecimientos a mi familia, amigos y a Mónica, ya no solo por el apoyo incondicional, sino por todos lo momentos que vivimos y compartimos que nos hacen ser quienes somos y por tanto, ser capaces de cumplir nuestras metas.

ABSTRACT

As digitization has expanded to almost every aspect of modern life, from smart devices to social networks to connected sensors, vast amounts of real data have been generated over time. The constant growth in data generation has created the necessity to develop new technologies to process this increasing amount of information. This unprecedented volume of information has far exceeded the capabilities of traditional data processing techniques, increasing the need for more advanced technologies such as distributed processing, cloud computing and machine learning. These technologies not only help manage and store large amounts of data, but also derive valuable insights from that data, leading to significant advances in areas such as artificial intelligence, data analytics and data-driven decision making. The growth of data generation continues to force the industry to develop increasingly innovative solutions to fully exploit the potential of this wealth of information.

One of the scenarios within the field of artificial intelligence and machine learning that has suffered from this increase in data is anomaly detection. The goal of anomaly detection is to identify observations that differ significantly from the majority of the data. Anomaly detection is an essential task in data analysis, especially in *Big Data* environments (environments with a huge volume of data) and time series (data with a temporal component). Two main approaches can be used to perform this task: supervised and unsupervised. In the supervised method, a model is trained using examples labeled as normal or anomalous. This method is effective when historical data have been clearly labeled, but may be limited in exceptional cases or when anomalies are unknown. On the other hand, unsupervised methods are used when there are no labels and the objective is to detect unusual patterns in the data. In time series, anomaly detection can identify atypical behavior along the time series, such as unusual peaks or trends. In *Big Data* environments, where data volumes are huge and diverse, unsupervised anomaly detection is especially useful for identifying unknown or emerging patterns. The combination of these methods is essential to ensure system integrity and security, as well as to detect hidden opportunities and threats in large data sets and time sequences. Also, anomaly detection is not faultless and can produce many false positives, i.e., normal data is labeled as anomalous. In

this context, false positive mitigation is the task of reducing the number of false positives labeled by the anomaly detector, so the two problems are closely related.

Given the growing necessity and importance of this type of scenarios, this thesis is going to focus on the one hand, on the proposal of a methodology to solve anomaly detection problems in combination with false positive mitigation in multivariate time series. On the other hand, the design and implementation of our anomaly detection algorithms in *Big Data* environments has been carried out. Specifically, the proposals are the following:

- A two-stage methodology for anomaly detection for multivariate time series and false positive mitigation, which creates the fusion of two learning models. The first stage is an anomaly detection stage. The second stage consists of training a new classifier on the false and true positives of the anomaly detector, which refines the observations labeled as anomalous by the anomaly detector to obtain more accurate and higher quality results. Experiments have been performed on two benchmark datasets, as well as a real case study, demonstrating the performance and validity of the proposal.

- Four distributed algorithms have been designed and implemented for anomaly detection problems in *Big Data* environments: HBOS_BD, LODA_BD, LSCP_BD, and XGBOD_BD. They have been designed following the distributed methodology *MapReduce* to be able to handle problems in *Big Data* environments. These algorithms have been integrated in a *Spark* package, focused on static and dynamic anomaly detection tasks in *Big Data* environments. The experiments have been performed using a real case study and have demonstrated the performance and validity of the proposals for problems in *Big Data* environments.

The proposals made provide solutions and different ways of approaching the problems of anomaly detection. On the one hand, a methodology is provided that includes a false positive mitigation stage, a scenario for which there is little literature. In addition, solutions in *Big Data* environments are proposed, which are of great help when processing and analyzing the large amounts of data generated on a daily basis.

RESUMEN

A medida que la digitalización se ha expandido a casi todos los aspectos de la vida moderna, desde los dispositivos inteligentes hasta las redes sociales y los sensores conectados, a lo largo del tiempo se han generado grandes cantidades de datos reales. El constante crecimiento en la generación de datos ha creado la necesidad de desarrollar nuevas tecnologías para procesar esta creciente cantidad de información. Este volumen de información sin precedentes ha superado con creces las capacidades de las técnicas tradicionales de procesamiento de datos, aumentando la necesidad de tecnologías más avanzadas como el procesamiento distribuido, la computación en la nube y el aprendizaje automático. Estas tecnologías no solo ayudan a gestionar y almacenar grandes cantidades de datos, sino que también obtienen información valiosa de esos datos, lo que conduce a avances significativos en áreas como la inteligencia artificial, el análisis de datos y la toma de decisiones basada en datos. El crecimiento de la generación de datos continúa obligando a la industria a desarrollar soluciones cada vez más innovadoras para explotar plenamente el potencial de esta riqueza de información.

Uno de los escenarios dentro del ámbito de la inteligencia artificial y el aprendizaje automático que más ha sufrido dicho incremento en el volumen de datos generados es la detección de anomalías. El objetivo de la detección de anomalías es identificar las observaciones que difieren significativamente de la mayoría de los datos. La detección de anomalías es una tarea esencial en el análisis de datos, especialmente en entornos *Big Data* (entornos con un volumen de datos enorme) y series temporales (datos con una componente temporal). Se pueden utilizar dos enfoques principales para realizar esta tarea: supervisado y no supervisado. En el método supervisado, se entrena un modelo utilizando ejemplos etiquetados como normales o anómalos. Este método es eficaz cuando se han etiquetado claramente los datos históricos, pero puede estar limitado en casos excepcionales o cuando se desconocen las anomalías. Por otro lado, los métodos no supervisados se utilizan cuando no hay etiquetas y el objetivo es detectar patrones inusuales en los datos. En las series temporales, la detección de anomalías puede identificar comportamientos atípicos a lo largo de la serie temporal, como picos o tendencias inusuales. En entornos *Big Data*, donde los volúmenes de datos son enormes y diversos, la detección de anomalías

no supervisadas es especialmente útil para identificar patrones desconocidos o emergentes. La combinación de estos métodos es esencial para garantizar la integridad y seguridad del sistema, así como para detectar oportunidades y amenazas ocultas en grandes conjuntos de datos y secuencias de tiempo. Asimismo, la identificación de anomalías no es perfecta y puede producir muchos falsos positivos, es decir, se etiquetan datos normales como anómalos. En este contexto, la mitigación de falsos positivos es la tarea de reducir el número de falsos positivos etiquetados por el detector de anomalías, por lo que ambos problemas están estrechamente relacionados.

Dada la creciente necesidad e importancia de este tipo de escenarios esta tesis se va a centrar por un lado, en la propuesta de una metodología para resolver problemas de detección de anomalías en combinación de mitigación de falsos positivos en series temporales multivariantes. Mientras que por otro lado, se ha realizado un diseño e implementación de algoritmos de detección de anomalías en entornos *Big Data*. Concretamente las propuestas son las siguientes:

- Una metodología en dos etapas para la detección de anomalías para series temporales multivariantes y mitigación de falsos positivos, que crea la fusión de dos modelos de aprendizaje. La primera etapa es una etapa de detección de anomalías. La segunda etapa consiste en entrenar un nuevo clasificador sobre los falsos y verdaderos positivos del detector de anomalías, que refina las observaciones etiquetadas como anómalas por el detector de anomalías para obtener resultados más precisos y de mayor calidad. Los experimentos han sido realizados con dos conjuntos de datos de referencia, así como un estudio de caso real, demostrando el rendimiento y la validez de la propuesta.

- Se han diseñado e implementado cuatros algoritmos distribuidos para problemas de detección de anomalías en entornos Big Data: HBOS_BD, LODA_BD, LSCP_BD, y XGBOD_BD. Han sido diseñados siguiendo la metodología distribuida *MapReduce* para ser capaces de manejar problemas en entornos*Big Data*. Estos algoritmos se han integrado en un paquete *Spark*, enfocado a tareas de detección de anomalías estáticas y dinámicas en entornos *Big Data*. Los experimentos han sido realizados utilizando un caso de estudio real por lo que han demostrado el rendimiento y la validez de las propuestas para problemas en entornos *Big Data*.

Las propuestas realizadas aportan soluciones y diferentes maneras de abordar os problemas de detección de anomalías. Por un lado, se aporta un metodología

que incluye una etapa de mitigación de falsos positivos, escenario del cual hay poca literatura al respecto. Además se proponen soluciones en entornos *Big Data* que son de gran ayuda a la hora de poder procesar y analizar las grandes cantidades de datos que se generan día a día.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

El aprendizaje automático [HKV19], una rama importante de la inteligencia artificial [RN10], ha cambiado radicalmente la forma en que interactuamos con la tecnología y abordamos problemas complejos en muchos campos. En esencia, el aprendizaje automático consiste en enseñar a las máquinas a aprender y mejorar a partir de los datos. Utilizando algoritmos y modelos matemáticos avanzados, las computadoras pueden analizar grandes conjuntos de información, detectar patrones, tomar decisiones y mejorar el rendimiento con el tiempo sin necesidad de programación explícita. A medida que avanzamos hacia la era de la información, el aprendizaje automático se ha convertido en una herramienta esencial para aprovechar el poder de los datos para resolver los complejos desafíos de una economía y una sociedad cada vez más digitales. La capacidad de automatizar tareas, tomar decisiones informadas y realizar predicciones precisas ha dado lugar a avances significativos en campos como la visión por computador, la detección de anomalías, el procesamiento del lenguaje natural, la medicina, la economía, la ingeniería y muchos otros campos. Es precisamente en el campo de la detección de anomalías en el que se centrará esta tesis.

La detección de anomalías se refiere al reto de encontrar observaciones que difieren significativamente de los datos restantes. Estos patrones inesperados suelen denominarse valores atípicos (*outliers*) o anomalías [CBK09; Agg16]. En la mayoría de los casos, los datos son generados por uno o varios procesos que reflejan el comportamiento de un sistema. Cuando dicho sistema se comporta de forma incorrecta, produce anomalías o valores atípicos. La identificación de tales observaciones anómalas tiene una importancia crucial para identificar el comportamiento errático del sistema [Agg16]. La detección de anomalías tiene una gran variedad de dominios, como la detección de fraudes con tarjetas de crédito [KTV+22], la detección de intrusiones [GD+20], las redes de sensores [ECN+23], las anomalías industriales [ZVF+23], o la asistencia sanitaria [FGD+21]. Por ejemplo, un comportamiento anómalo en un motor puede significar que está próximo a una avería. La detección de anomalías temprana en los sensores de los motores podría evitar dicha avería. Debido a la gran variedad de dominios en los que se puede aplicar la detección de anomalías y a la

relevancia de los beneficios que aporta, el problema de la detección de anomalías tiene una importancia creciente en la actualidad. Existen tres tipos diferentes de problemas en la detección de anomalías [CBK09]:

- Detección anomalías supervisada: El conjunto de datos se etiqueta indicando qué instancias son normales y cuáles anómalas. Se construye un modelo predictivo para separar las instancias normales de las anómalas [JAK01].
- Detección de anomalías semi-supervisada: El conjunto de entrenamiento utilizado no tiene instancias anómalas, sólo observaciones normales. Las instancias anómalas se proporcionan en el conjunto de prueba [DM02].
- Detección de anomalías no supervisada: Las instancias no están etiquetadas. Las instancias anómalas son desconocidas y el algoritmo debe ser capaz de detectarlas sin conocimiento previo [GU16b].

Debido a la automatización en la adquisición y almacenamiento de datos, la mayoría de los problemas de detección de anomalías en el mundo real pertenecen al tipo no supervisado. El escenario de detección de anomalías no supervisada es particularmente desafiante porque los enfoques de aprendizaje automático no tienen conocimiento previo sobre los datos [BH+01]. Los algoritmos de detección de anomalías sin supervisión puntúan los datos basándose únicamente en las propiedades del conjunto de datos. Esta puntuación representa el grado de "rareza" de cada caso. A continuación, utilizando un umbral o una cantidad fija, se seleccionan las anomalías. Los algoritmos no supervisados de detección de anomalías pueden agruparse en seis categorías [Agg16]:

- Basados en valores extremos [Pic+75].
- Basados en métodos estadísticos [Pev16].
- Basados en clasificación [CC19; RBL19; ERK+16; HAB18].
- Basados en vecinos más cercanos [BKN+00].
- Basados en clustering [HXD03].
- Basados en Ensembles [ZNH+19].

En el mundo de la industria y con el auge del IoT (*Internet of Things*), la importancia de las series temporales ha cogido un gran importancia ya que es muy común que los datos provengan de las mediciones de uno o varios sensores dispuestos en las máquinas que se quieran analizar. Las series temporales son datos que tienen una componente temporal, es decir, cada observación no es independiente, sino que está relacionada en el tiempo. Las series temporales pueden clasificarse en univariantes, que tienen una sola característica, y multivariantes, que tienen más de una característica. Las series temporales mostrarán valores diferentes en distintos periodos de tiempo sin indicar necesariamente una anomalía. Por ejemplo, un motor puede tener una temperatura más alta de lo normal en un instante de tiempo, pero ese sobrecalentamiento puede deberse simplemente a una mayor carga de trabajo y no a un fallo. Aprender el comportamiento de una serie temporal puede servir para analizar datos futuros y anticiparse así a un fallo y prevenir posibles daños [Zha03; PGP+21]. Dentro de una serie temporal, una anomalía suele estar determinada por varios valores anómalos consecutivos en el tiempo [CLA+21], lo que supone un gran inconveniente para los problemas tradicionales de detección de anomalías, ya que los enfoques tradicionales tratan las anomalías sin tener en cuenta la componente temporal [TLZ+18].

Teniendo en cuenta las características de los problemas de detección de anomalías descritos anteriormente, es conveniente hacer uso de un algoritmo que tenga en cuenta la componente temporal en los problemas de detección de anomalías en series temporales. Cuando nos enfrentamos a series temporales multivariantes, el estado del arte actual está poblado por redes neuronales recurrentes como las redes de Memoria Larga a Corto Plazo (LSTMs) y las redes de Unidades Recurrentes Cerradas (GRUs) [STN18; STN19; GAS+21]. Un algoritmo reciente y de buen rendimiento son las redes convolucionales temporales (TCN)[BKK18; LVR+16]. Sin embargo, aunque estos enfoques son modelos capaces de obtener resultados de calidad, no están exentos de la presencia de falsos positivos.

Un falso positivo sucede cuando el modelo entrenado indica que una observación anómala cuando realmente ésta observación es normal. En el contexto de diversas disciplinas, como en pruebas diagnósticas o sistemas de detección, los falsos positivos son perniciosos debido a que pueden llevar a diagnósticos erróneos o alertas innecesarias. Una manera de tratar de corregir esta problemática, puede ser aplicar una etapa posterior de mitigación de falsos positivos para mejorar la calidad de los resultados, tal y como se expone en [HCL+18; LCO18; SHZ+19]. No obstante, las técnicas actuales de mitigación de falsos positivos son soluciones ad hoc y están limi-

tadas a los conjuntos de datos que analizan tal y como se recoge en las 3 referencias mencionadas en la frase anterior. O también encontramos referencias que mencionan la mitigación de falsos positivos pero no realizan una etapa de mitigación como tal, simplemente mencionan que gracias a los modelos que emplean, se garantiza la mitigación de FPs [LZW+18; Yos04].

En conjunto con lo mencionado anteriormente, la mejora de las tecnologías, así como el nacimiento de otras nuevas, como los teléfonos inteligentes, las comunicaciones 5G, los sensores, la nube de internet, la realidad virtual y las aplicaciones domésticas inteligentes, da lugar a que se genere rápidamente un enorme volumen de datos. El crecimiento de la cantidad de datos generados ha dado lugar a la era del *Big Data* [TBJ+20; CCS12]. *Big Data* se refiere a datos de gran volumen, velocidad y variedad que no pueden procesarse con métodos convencionales. Esto ha creado la necesidad de desarrollar métodos específicos para los diferentes datos que puedan llegar [LGR+20; HNG+19; RFG+18]. La automatización en la adquisición de datos, la popularización de los sensores y la falta de supervisión humana que caracteriza al *Big Data* ha incrementado la necesidad de métodos eficientes de detección de anomalías [FHL14]. La naturaleza del *Big Data* hace que en la mayoría de los casos no pueda ser supervisado y etiquetado por un experto. Esa problemática genera que la mayoría de los problemas de detección de anomalías en *Big Data* en el mundo real sean no supervisados. A pesar de contar con librerías muy populares como PyOD [ZNL19] para problemas de detección de anomalías de tamaño normal, en *Big Data* sólo podemos encontrar un puñado de propuestas dedicadas a dominios específicos de este problema [HNG+19; TBJ+20; HC14; RKC+19].

Dada la complejidad y la amplitud que aportan todas estas variables al campo de la detección de anomalías, en esta tesis se pretende aportar soluciones diversas que sirvan para resolver diferentes problemas. Concretamente, en esta tesis se presenta en primera instancia una metodología que combina la detección de anomalías con la mitigación de falsos positivos y en segunda instancia cuatro algoritmos de detección de anomalías implementados en el framework *Apache Spark*, es decir, algoritmos diseñados para entornos *Big Data*.

2

FUNDAMENTOS DE LA DETECCIÓN DE ANOMALÍAS Y BIG DATA

En este capítulo se explicará en detalle el campo de la detección de anomalías: la razón de su importancia, el origen de su complejidad, diferenciación de problemas, algoritmos clásicos, estado del arte actual y métricas para determinar la calidad de los modelos. Asimismo, también se detallará el concepto *Big Data*: en qué consiste, sus ventajas, el paradigma *MapReduce* y el *Framework Spark*. En la Sección 2.1 se definen las anomalías y sus tipos. En consecuencia, en la Sección 2.2 se detalla el problema de detección de anomalías (supervisada y no supervisada), se muestran los distintos algoritmos y modelos que se pueden emplear para la resolución de los problemas de detección de anomalías y se definen las medidas de evaluación. En la Sección 2.3 se explica qué son los falsos positivos, en que consiste la mitigación de FPs y el estado del arte actual. Por último en la Sección 2.4 se explica en qué consiste el *Big Data*, el paradigma *mapReduce* y *Apache Spark*.

2.1 LAS ANOMALÍAS Y SUS TIPOS

Una anomalía es una observación que no sigue el mismo patrón que el resto de los datos. Figura 2.1 muestra una representación gráfica de las anomalías en un conjunto de datos bidimensional. Los *clusters* C1 y C2 están compuestos por observaciones normales, ya que prácticamente todos los puntos pertenecen a estas dos regiones. El *cluster* C3 contiene muy pocas observaciones, por lo que se trata de un *cluster* anómalo. Las observaciones O1, y O2 están completamente aisladas y por lo tanto son instancias anómalas [CBK09].

En la literatura, podemos encontrar tres categorizaciones diferentes de instancias anómalas [GU16b]:

- Anomalía puntual: Es el escenario más frecuente en la detección de anomalías. Las instancias anómalas están completamente aisladas del resto. En la Figura 2.1, O1 y O2 son anomalías puntuales.

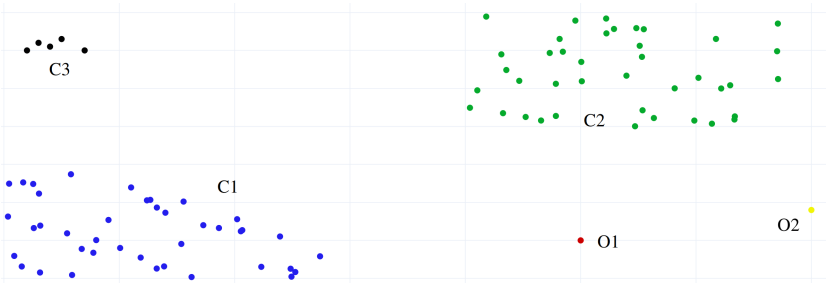


Figura 2.1: Representación de distintos tipos de anomalías en un conjunto de datos de dos dimensiones.

- Anomalía colectiva: La anomalía es una mezcla de varios casos anómalos. Por ejemplo, la detección de un robo de tarjeta de crédito puede implicar la detección de varios extractos de cuentas bancarias.
- Anomalía contextual: Un caso que no es anómalo podría serlo dentro de un contexto determinado. Por ejemplo, si medimos la temperatura de un motor en un intervalo de 50 a 120 grados. Una temperatura de 80 grados parece completamente normal, pero si ese valor se da cuando el motor no tiene carga de trabajo, la temperatura estimada debería ser inferior.

Las anomalías están relacionadas con el ruido, pero ambos conceptos no deben confundirse. El ruido tiene el mismo comportamiento descrito en la Figura 2.1, pero, el ruido no es de interés para el analista de datos mientras que las anomalías sí lo son. El ruido se produce por una alteración en los datos, por lo que no refleja la distribución original de los mismos. Además, el ruido daña la calidad de los datos y esas observaciones deben ser corregidas o eliminadas [LGR+20; GLG+19; GLL+19]. Las anomalías son información valiosa que debe detectarse, extraerse y analizarse.

2.2 DETECCIÓN DE ANOMALÍAS

La detección de anomalías se refiere al problema de encontrar patrones que difieren significativamente de las observaciones estándar. Existen multitud de aplicaciones en el ámbito de la detección de anomalías. La detección de intrusos consiste en la detección de actividad anómala en una red de ordenadores [GD+20]. Este problema se caracteriza por la gran cantidad de flujo de información que puede conducir a

un alto número de falsas alarmas. La detección de fraude se refiere a encontrar movimientos inusuales relacionados con delitos en aplicaciones comerciales como tarjetas de crédito, compañías telefónicas, bancos, etc [KTV+22]. Esos movimientos inusuales están relacionados con intentos de robo o usurpación de identidad por parte de un consumidor. Otras aplicaciones no relacionadas con fraudes, como las médicas sanitarias, consisten en la detección de anomalías en las medidas de los pacientes, que pueden ser producidas por alguna enfermedad del paciente, errores de instrumentación o errores de registro [FGD+21]. También, la detección de anomalías se puede aplicar en el mundo de la industria, detectando comportamientos inesperados de los motores en una cadena de montaje, errores de instrumentación de sensores de motores o algún daño en estructuras [ZVF+23]. Otras aplicaciones de interés son el procesamiento de imágenes [CC19], la detección de anomalías en datos de texto o la detección de anomalías en redes de sensores, que se refiere a la detección de anomalías en la recogida de datos que pueden significar intrusiones o errores en los sensores [ECN+23]. El gran número de dominios que involucran el problema de detección de anomalías, y el creciente número de sensores en todos los campos están haciendo que la popularidad e importancia de la detección de anomalías siga creciendo.

La detección de anomalías es una parte esencial del análisis de datos para identificar patrones inusuales o eventos atípicos. En un contexto supervisado, el modelo se entrena con datos normales y anómalos conocidos, lo que le permite aprender las características normales del conjunto de datos y luego detectar cualquier desviación significativa durante la fase de test. Por otro lado, en la detección de anomalías no supervisada, el modelo se enfrenta a datos sin etiquetar y busca identificar patrones anómalos basándose únicamente en la estructura inherente de los datos normales. Ambos enfoques son importantes en una variedad de campos, desde la monitorización industrial hasta la ciberseguridad, donde la identificación temprana de comportamientos anómalos puede ser clave para prevenir incidentes y optimizar el comportamiento de cualquier sistema.

2.2.1 *Detección de anomalías supervisada*

La característica principal de la detección de anomalías supervisada es que se entrena un algoritmo de aprendizaje automático utilizando un conjunto de datos etiquetados que contiene ejemplos de datos normales y de datos anómalos. El objetivo

es que el modelo aprenda a distinguir entre estos dos tipos de datos para que, una vez entrenado, éste sea capaz de identificar con precisión nuevas observaciones de datos anómalos [CBK09; AMH16].

Dada su naturaleza, la detección de anomalías supervisada suele ser más precisa que su contraparte no supervisada ya que el modelo entrenado dispone de información etiquetada y por tanto, es más sencillo aprender a diferenciar las observaciones normales de las anómalas. Además, también se facilita la interpretación del modelo ya que el conjunto de datos usado para entrenar está previamente definido, por lo que se facilita su comprensión.

No obstante, la necesidad de disponer de datos etiquetados es problemático ya que es una tarea costosa y tediosa, que incluso en determinadas ocasiones, puede ser inviable. Además, el modelo puede tener dificultades a la hora de detectar nuevas anomalías que no se encontraban en el conjunto de datos de entrenamiento.

La metodología de detección de anomalías en combinación con la mitigación de falsos positivos propuesta en esta tesis está diseñada para resolver problemas de detección de anomalías supervisados.

2.2.2 *Detección de anomalías no supervisada*

En contraposición con la detección de anomalías supervisada, la variante no supervisada se caracteriza por no disponer de datos etiquetados de entrenamiento. El objetivo en este caso, es identificar patrones inusuales sin la guía de ejemplos etiquetados de datos anómalos [CBK09; AMH16].

Una de sus principales características es la adaptabilidad y capacidad de aprendizaje de nuevas anomalías que eran desconocidas, lo que es bastante útil en escenarios que puedan variar con el paso del tiempo. Otra ventaja es que no es necesario etiquetar los datos, ya que, como se ha comentado anteriormente, es una tarea que puede ser bastante complicada en algunas situaciones.

Sin embargo, debido a esa falta de etiquetas, la incertidumbre que genera el modelo puede ser mayor que en los problemas supervisados ya que se puede generar un mayor número de falsos positivos (FP) o falsos negativos (FN). También se pierde explicabilidad, ya que no está bien definido con anterioridad qué es una anomalía en ese conjunto de datos.

Los algoritmos diseñados para entornos *Big Data* propuestos en esta tesis se ubican dentro de la detección de anomalías no supervisada.

2.2.3 *Planteamientos clásicos para resolver problemas de detección de anomalías*

Podemos encontrar muchas técnicas diferentes de detección de anomalías clásicas en la literatura [CBK09; Agg16]. Aunque en esta tesis nos centraremos en aplicar las técnicas *Deep Learning* que componen el estado del arte, los modelos clásicos según el problema a tratar pueden aportar un buen rendimiento, sobre todo porque, a priori, sus tiempos de cómputo son más reducidos. Además, algunas de estas técnicas serán empleadas en la etapa de mitigación de FPs. Por lo tanto, a continuación se clasifican las técnicas clásicas en función del enfoque que se utilice:

- Análisis de valores extremos: Es la forma más básica de detección de anomalías. Se basa en el análisis de datos unidimensionales. Estos métodos asumen que un valor es anómalo si es demasiado grande o demasiado pequeño [Pic+75].
- Modelos probabilísticos y estadísticos: Los datos se modelan como una distribución de probabilidad. Las instancias que se encuentran en regiones de mayor probabilidad son las normales y las instancias anómalas se encuentran en regiones de menor probabilidad [GD12; Pev16].
- Basadas en la clasificación: Estas técnicas utilizan un conjunto de datos de entrenamiento etiquetado para construir un modelo y posteriormente, el conjunto de datos de prueba se clasifica utilizando el modelo aprendido asignando una etiqueta a cada instancia. Como ejemplos de estas técnicas podemos encontrar: Deep Learning-based [CC19], Bayesian Networks-based [RBL19], Support Vector Machines-based [ERK+16] y Rule-based [HAB18] methods.
- Basados en Vecinos más Cercanos: Asumen que las instancias dentro de un vecindario de alta densidad son instancias normales, y las instancias alejadas de esos vecindarios serían anómalas. Dentro de estas técnicas existen dos enfoques principales: Los métodos basados en la distancia [RRS00] y los modelos basados en la densidad [BKN+00].
- Basados en clustering: Asume que las instancias dentro de un cluster son normales. En cambio, las instancias que están fuera de un cluster son anómalas [HXD03]. La principal diferencia con respecto a las técnicas de vecinos más cercanos es que los métodos de clustering evalúan cada instancia en función del cluster al que pertenece, mientras que las técnicas de vecinos más cercanos analizan cada instancia en su vecindad local.

- Ensembles: Consiste en la combinación de múltiples algoritmos diversos de aprendizaje base para obtener un modelo global que mejore los detectores base [Pev16; ZNH+19; ZH18].

2.2.4 *Deep Learning para resolver problemas de detección de anomalías*

Antes de la llegada del Deep Learning, los tipos de algoritmos utilizados para resolver problemas de detección de anomalías suelen incluir algoritmos clásicos de aprendizaje automático (KNN, k-Means o HBOS), algoritmos de detección de valores atípicos (Isolation Forest o LOF) o algoritmos de Minería de Datos (STOMP o PST) entre otros [SWP22].

Actualmente, el estado del arte en detección de anomalías en series temporales se centra en el uso de redes neuronales recurrentes (LSTM y GRU) y redes neuronales temporales (TCN) [STN18; STN19; GAS+21]. A continuación, describimos brevemente el funcionamiento de algunos de los métodos más recientes y de mejor rendimiento que se usarán en esta tesis:

- TCN [LVR+16]: El modelo TCN se centra en el uso de un marco de codificación-decodificación que utiliza un único conjunto de mecanismos computacionales (convoluciones 1D, pooling y normalización de canales) para capturar jerárquicamente información temporal de bajo, medio y alto nivel. Las convoluciones 1D se aplican para visualizar los cambios de las características en los niveles inferiores a lo largo del tiempo, la agrupación se utiliza para calcular patrones temporales de largo alcance de forma eficiente, y la normalización consigue una mayor robustez frente a diversas condiciones ambientales.

- WeiXiaoyan [WZZ+19]: Para crear un modelo de aprendizaje profundo espacio-temporal se combinan LSTM y una CNN. La CNN se utiliza para extraer características relevantes. Las características se descomponen en componentes secuenciales y se proporcionan a unidades LSTM repetitivas para su análisis. La salida del último paso de la LSTM se proporciona a la capa totalmente conectada para la predicción.

- YiboGao [GWL21]: El uso de una red neuronal convolucional funciona bastante bien en ciertos problemas, sin embargo, no tiene en cuenta las características temporales del problema. Para solucionarlo, se añade a la arquitectura un bloque de atención temporal basado en residuos (bloque RTA). El bloque RTA

utiliza el aprendizaje residual para generar el peso de la atención temporal, lo que permite extraer más información de las características.

Es importante recalcar que el modelo TCN está concebido para series temporales univariantes. Por lo tanto, para su uso en escenarios más generales, se considerará un modelo multivariante modificado. Esta versión multivariante de TCN funciona de forma similar a TCN, adaptando únicamente la entrada de la red para aceptar múltiples características de entrada. Para simplificar, se hará referencia a esta modificación multivariante como TCN durante el resto de la tesis.

Por último, dado que el estado del arte actual se centra en las redes neuronales, los algoritmos descritos anteriormente se considerarán los principales detectores de anomalías en este trabajo.

2.2.5 *Evaluación de los problemas de detección de anomalías*

En cuanto a la salida de un algoritmo de detección de anomalías, puede ser de dos tipos diferentes [CBK09]:

- Etiquetas (*labels*): El valor de retorno es una etiqueta binaria para cada instancia que indica qué instancias son normales y cuáles son anomalías.
- Puntuación (*score*): El valor de retorno es una puntuación de anomalía para cada instancia. La puntuación indica la probabilidad de que la instancia sea anómala, o también puede servir como medida cuantitativa del grado de anomalía de la instancia. También debe estudiarse qué puntuaciones son anómalas.

Para analizar el rendimiento de un algoritmo de detección de anomalías, se suelen utilizar cuatro métricas, que se representan en la matriz de confusión que se muestra en la Tabla 2.1. En el caso de que la salida del algoritmo de detección de anomalías sea un *score*, será necesario convertir dicho *score* a una etiqueta (anomalía o normal). La manera típica de convertir ese *score* será establecer un umbral a partir del cual, la observación se considera anómala o no. El umbral a establecer será un parámetro a optimizar durante la resolución del problema.

Volviendo a la matriz de confusión, tanto los verdaderos negativos (VN) como los verdaderos positivos (VP) son los valores más importantes y considerados, representan que el modelo está acertando en la predicción. En contraposición, los FPs y los

		Predicción	
		Negativo	Positivo
Real	Negativo	Verdadero Negativo (VN)	Falso Positivo (FP)
	Positivo	Falso Negativo (FN)	Verdadero Positivo (VP)

Tabla 2.1: Matriz de confusión

FNs son los valores que se intentará reducir en la medida de lo posible ya que indica que el modelo está fallando en la predicción.

Un número elevado de FPs o FNs se considera problemático. Hay dos medidas principales calculadas a partir de la matriz de confusión, la *sensibilidad* ($\frac{VP}{VP+FN}$), que indica la capacidad de etiquetar como positivos los VPs, y la *especificidad* ($\frac{VN}{VN+FP}$), que indica la capacidad de etiquetar como negativos los VNs. Sin embargo, existe un equilibrio entre ambas medidas, ya que el aumento de una implica la disminución de la otra, por lo que encontrar el mejor ajuste posible es una tarea difícil y dependerá del problema a tratar qué medida tendrá prioridad. Existe otro compromiso entre VPs y la tasa de FPs ($FPR = \frac{FP}{FP+VP}$). El aumento de VPs suele dar lugar a un mayor FPR porque se da más importancia a las observaciones positivas y, por lo tanto, aunque más observaciones positivas se etiqueten correctamente, las observaciones negativas también se etiquetarán como positivas.

Una característica distintiva de los problemas de detección de anomalías es que el número de observaciones normales es, por norma general, mucho mayor que el número de observaciones anómalas, por lo que intentar ajustar el modelo para que se centre en las observaciones anómalas provoca un aumento del número de FPs, generando así alarmas innecesarias. Además, es posible que una observación sea etiquetada como positiva cuando no lo es en función de su contexto (dando lugar a un FPs), lo que es común cuando se trata de un problema que implica series temporales [HCL+18; SG14]. Debido a las complicaciones descritas anteriormente, será difícil que el modelo encuentre un buen ajuste en el que el número de FPs sea bajo, por lo tanto, para corregir los modelos que generan un alto número de FPs o para mejorar la calidad de dicho modelo, es interesante aplicar un método de reducción de FPs.

Otra de las métricas más usadas para evaluar el rendimiento de los métodos de detección de anomalías es el *Área Bajo la Curva* (ROC-AUC) que se define en [HM82] como: "*Given a ranking or scoring of a set of points in order of their propensity to be outliers (with higher ranks/scores indicating greater outlierness), the ROC-AUC is equal to*

the probability that a randomly selected outlier-inlier pair is ranked correctly (or scored in the correct order)" [Dada una clasificación o puntuación de un conjunto de puntos en relación a su tendencia a ser valores anómalos (con clasificaciones/puntuaciones más altas que indican mayor tendencia a ser anómalos), el *ROC-AUC* es igual a la probabilidad de que un par de valores normal-anómalo seleccionado al azar se clasifique correctamente (o se puntúe en el orden correcto)].

Finalmente, otra de las métricas ampliamente utilizada en el ámbito de la detección de anomalías es la métrica *F1-score*. La métrica *F1-score* es una forma de combinar precisión ($\frac{VP}{VP+FP}$) y sensibilidad ($\frac{VP}{VP+FN}$) en una sola medida, su fórmula se describe en la Ecuación 2.1:

$$F1 - score = \frac{2 * precision * sensibilidad}{precision + sensibilidad} \tag{2.1}$$

El *ROC-AUC* es una métrica que evalúa la capacidad del modelo para discriminar entre clases, por lo que es útil en los problemas de detección de anomalías. Un *ROC-AUC* cercano a 1 indica que el modelo es capaz de separar claramente las instancias normales de las anómalas. Por otro lado, la métrica *F1-score* es útil cuando se busca un equilibrio entre precisión y sensibilidad. En la detección de anomalías, es importante no solo identificar anomalías correctamente (sensibilidad alta) sino también minimizar los FPs (precisión alta). En esta tesis se emplearán ambas métricas, es decir, *ROC-AUC* y *F1-score* para estimar la calidad de los modelos que se han empleado tanto para demostrar la validez de la metodología propuesta como para demostrar el rendimiento de los algoritmos diseñados para entornos *Big Data*.

2.3 MITIGACIÓN DE FALSOS POSITIVOS

Un FP se produce cuando el modelo predice una instancia como positiva cuando en realidad es negativa, lo que constituye un escenario que debe evitarse. Por ejemplo, en relación con la reciente enfermedad por coronavirus 2019 (COVID-19), es frecuente encontrar pruebas de PCR que resultaron en FPs. Esto obliga a poner en cuarentena a estas personas, impidiéndoles en muchos casos acudir al trabajo, lo que conlleva pérdidas económicas [BSH+21]. Otro ejemplo se centra en el alcance de los ciberataques en los Sistemas de Control Industrial (ICS), un elevado número de FPs se traduce en una alta tasa de falsas alarmas, en consecuencia, el sistema tiene que

ser revisado más de lo necesario lo que se traduce en un menor rendimiento. Por lo tanto, la reducción de FPs es de gran importancia [SG14].

Es muy frecuente encontrarse con la problemática de un elevado número de FPs cuando se trata de un problema de detección de anomalías [JD02; Jul03]. Normalmente, en los problemas de detección de anomalías los conjuntos de datos tienen un gran número de observaciones normales mientras que el número de observaciones anómalas es pequeño debido a la falta de datos, ya que las anomalías son eventos inusuales y ocurren en ocasiones excepcionales. Esto hace que el modelo de detección no sea lo suficientemente preciso y, por tanto, no pueda detectar las anomalías al haber pocas observaciones. Una forma de solucionarlo es dar más importancia a las observaciones anómalas o generar más instancias anómalas sintéticas. Sin embargo, esta solución conlleva un mayor número de FPs [HG09].

En relación con las cuestiones ya mencionadas, la mitigación de FPs tiene una importante aplicación en los problemas de detección de anomalías. Sin embargo, la mayor parte de la información en la literatura que se centra en tratar la mitigación de FPs no propone un procedimiento general, sino que se centra en una solución *ad-hoc* para el problema que no se puede generalizar a otras áreas. A continuación, se detallan las propuestas más relevantes en el marco en el que se desarrolla la presente tesis.

En [LZW+18] los autores argumentan que una CNN tiene una capacidad de aprendizaje limitada, por lo que es posible que no aprenda todas las características fundamentales para distinguir la estructura de un nódulo de otros que no lo son. Sin embargo, trabajar con varias CNNs permite tratar con más nódulos. La propuesta consiste en un conjunto de CNNs (E-CNNs) para permitir aprender distintos tipos de nódulos y reducir así el número de FPs.

En [KY02] los autores reducen los FPs en la detección de nódulos pulmonares en radiografías de tórax utilizando características morfológicas, el borde de la costilla y la característica de cobertura del borde del nódulo. Por otro lado, también en el área de detección de nódulos pulmonares, el autor en [Yos04] reduce los FPs utilizando la eliminación de la estructura de la costilla basada en la simetría del área pulmonar izquierda y derecha.

Como último ejemplo, en [ZG20] los autores describen una técnica para la detección de anomalías que garantiza la mitigación de los FPs mediante la poda de anomalías no relacionadas, manteniendo al mismo tiempo un umbral actualizado que se utilizará en el sistema de puntuación de anomalías.

Las propuestas anteriores tienen un buen rendimiento, sin embargo, aunque sus técnicas consiguen un bajo índice de FPs, no son extrapolables a otros problemas. Es interesante y necesario disponer de mecanismos que mejoren los resultados de los actuales métodos de detección de anomalías. Es decir, disponer de una propuesta de mitigación de FPs que complemente y sea capaz de mejorar los resultados de un modelo de detección de anomalías ya entrenado, lo que daría una gran versatilidad al estudio de anomalías.

Un enfoque para abordar el problema es el propuesto en [HCL+18], que consiste en la poda de anomalías. En el problema abordado en el citado artículo, existen una serie de secuencias de error calculadas a partir de una LSTM, mientras que la idea es podar estas secuencias para mantener un contexto de datos actual y reducir el coste de memoria y tiempo de computación. El proceso consiste en volver a etiquetar como normales aquellas secuencias anómalas cuyos valores no superen consecutivamente un umbral.

Otro enfoque descrito en [LCO18] es el uso de un componente inicial denominado "Process Action Monitoring Component", disponible en los sistemas *antimalware*. A continuación, se genera un nuevo conjunto de datos con nuevos atributos generados por el componente. Este nuevo conjunto de datos se entrena utilizando un modelo de red neuronal artificial (RNA). El entrenamiento de la red neuronal artificial corresponde a la etapa de mitigación de los FPs.

En [SHZ+19], los autores siguen la última idea descrita, esta publicación propone el uso de un modelo de red neuronal convolucional (CNN) para la detección de nódulos pulmonares. A partir del modelo CNN, se extraen una serie de atributos que serán la entrada de un nuevo clasificador de máquina de vectores soporte (SVM). Este modelo SVM consigue una tasa de FPs más baja.

Abordar el problema de la mitigación de los FPs de la misma forma que en los tres últimos artículos mencionados, aplicando una etapa de mitigación tras el modelo de detección de anomalías, permite extrapolar estas herramientas de mitigación a una gran variedad de ámbitos. Además, hay un gran número de herramientas que se pueden emplear en la etapa de mitigación que aún no se ha estudiado.

Las propuestas de los tres últimos artículos mencionados se limitan a un dominio y unos conjuntos de datos específicos; además, en [LCO18; SHZ+19] los conjuntos de datos propuestos no tienen un componente temporal. La metodología propuesta en esta tesis está diseñada de tal manera que puede aplicarse a series temporales. Además, la etapa de mitigación de FPs propuesta en la metodología de esta tesis, puede aplicarse a cualquier conjunto de datos. En [SA22], los autores afirman que los

modelos de aprendizaje profundo no siempre son mejores en datos tabulares, lo que fomenta el uso de diferentes métodos (SVMs, Random Forest, ANNs, y ensembles) para analizar su comportamiento en diferentes tipos de problemas,y por tanto, se usarán en en la parte de experimentación de esta tesis.

2.4 BIG DATA

Con la explosión de datos, la popularización de los sensores y la automatización en la adquisición y almacenamiento de datos, el problema de la detección de anomalías se ha convertido en un problema *Big Data*. Ciertos ámbitos, como la detección de intrusiones en la red o la prevención de fallos, deben abordarse rápidamente para evitar problemas mayores. Además, con este crecimiento exponencial de los datos, los algoritmos clásicos no pueden procesarlos en un tiempo razonable [LGR+20; HNG+19].

El término *Big Data* no se limita a la cantidad de datos sino que también está relacionado con la variedad y velocidad en la generación de los mismos. En general, el *Big Data* se caracteriza por las famosas *tres Vs*: Volumen, Variedad y Velocidad. No obstante, además de estas tres características básicas, se han introducido otras más para definir con más exactitud el concepto *Big Data*: visión (un propósito), verificación (los datos procesados se ajustan a unas especificaciones), validación (el propósito se cumple), valor (se puede extraer información relevante de los datos), complejidad (es difícil organizar y analizar los datos debido a la evolución de las relaciones entre los datos) e inmutabilidad (los datos recopilados y almacenados pueden ser permanentes si se gestionan bien) [OBL+18]. A continuación se definen con más detalle las tres características básicas:

1. Volumen: el volumen se refiere a la enorme cantidad de datos que se generan y acumulan continuamente. Esto puede incluir datos estructurados (por ejemplo, bases de datos) y datos no estructurados (texto, imágenes, videos, etc.). La escala de datos es un aspecto importante del *Big Data*, ya que las organizaciones cantidades ingentes de datos, llegando a poseer incluso petabytes de información.

2. Variedad: se refiere a la variedad de fuentes y formatos de datos. *Big Data* incluye datos digitales, texto, registros de sensores, datos de redes sociales, etc. Esta diversidad requiere de herramientas y técnicas especializadas para su análisis y procesamiento.

3. Velocidad: consiste en la velocidad a la que se crean y procesan los datos. En muchos situaciones, los datos deben obtenerse y analizarse en tiempo real para tomar decisiones rápidas. Esto es especialmente importante en aplicaciones como la detección de fraudes o la supervisión de sistemas en tiempo real.

Los beneficios del *Big Data* son notorios y variados ya que permite una toma de decisiones más informada y estratégica, identificando patrones ocultos, personalizando productos y servicios para mejorar la satisfacción del cliente, optimizando las operaciones internas para reducir costos y la capacidad de impulsar la innovación en todo el mundo en una amplia variedad de industrias. Además, el *Big Data* puede contribuir al avance de la ciencia, la medicina y la investigación al brindar acceso a enormes conjuntos de datos que pueden revelar información de interés y que de otra manera no se podría procesar ni analizar. Por lo tanto, el *Big Data* se ha convertido en una herramienta esencial para la prosperidad, la eficiencia y el desarrollo de la era digital actual [HNG+19; LGR+20].

2.4.1 *Paradigma MapReduce*

MapReduce es el paradigma más popular y ampliamente utilizado para el procesamiento ene entornos *Big Data* en la actualidad. El paradigma *MapReduce* puede procesar y generar grandes cantidades de datos de forma distribuida y eficiente, además minimiza el acceso a disco y el uso de red [DG04].

Consta de dos fases: la fase *map* y la fase *reduce*. En primer lugar, el nodo maestro particiona los datos y los distribuye por el *cluster*. La función *map* aplica una operación de transformación a los pares clave-valor locales de cada nodo de computación. En otras palabras, cada nodo procesa una parte del algoritmo para un subconjunto de datos. Tras la fase *map*, todos los pares con la misma clave se redistribuyen. Cuando todos los pares con la misma clave están en el mismo nodo de cálculo, comienza la fase de *reduce*. La fase *reduce* es una operación de resumen que une todos los resultados de las diferentes fases de mapa en los valores finales. En la Figura 2.2 se muestra visualmente el proceso descrito.

2.4.2 *Apache Spark*

Apache Spark [KKW+15] es un framework de código abierto para entornos *Big Data*, centrado en la velocidad, la facilidad de uso y la analítica sofisticada. *Spark*

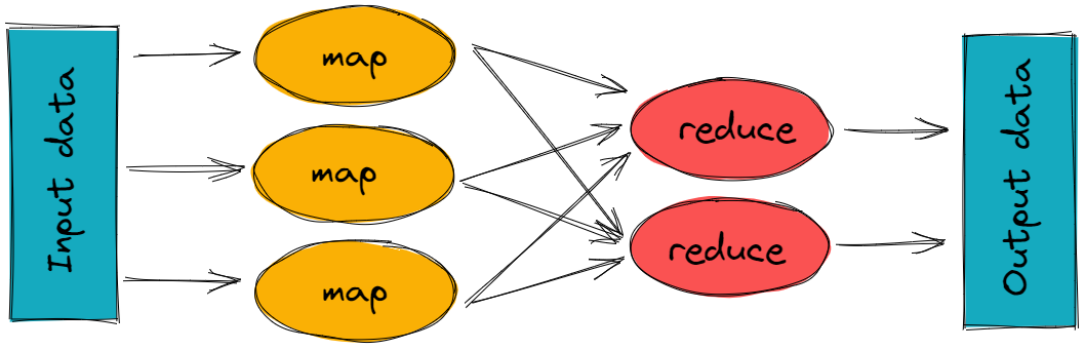


Figura 2.2: Representación gráfica del paradigma MapReduce

permite persistir los datos en memoria para su procesamiento consecutivo o iterativo, mejorando enormemente el rendimiento. *Spark* está construido sobre *Resilient Distributed Datasets* (RDDs) [ZCD+12], que no están ordenados y son inmutables por naturaleza. Permiten a los programadores persistirlos en memoria, además, los RDDs se encuentran distribuidos en los nodos del cluster para que cada partición pueda volver a calcularse en caso de fallo. Los *RDD* admiten dos tipos de operaciones: transformaciones, que se evalúan perezosamente (solo se ejecutan cuando una acción lo indica) y producen un nuevo *RDD*, y acciones, que activan todas las transformaciones anteriores y devuelven un valor.

2.4.2.1 *Primitivas de Spark*

En este apartado se describen algunas de las primitivas de *Spark* disponibles en la *API* de *Spark*. Estas primitivas extienden las funcionalidades del paradigma *MapReduce* permitiendo realizar operaciones más complejas sobre los datos. A continuación se comentan las empleadas en la implementación de los algoritmos de detección de anomalías[1]:

- *map*: Realiza una función definida por el usuario sobre cada elemento de un conjunto distribuido. Tras esta transformación se crea un nuevo conjunto distribuido.

1 Para una descripción completa de las operaciones de *Spark*, consultar la *API* de *Spark*: https://spark.apache.org/docs/latest/api/scala/org/apache/spark/index.html

- *histogram*: Calcula un histograma del conjunto distribuido utilizando un número especificado de contenedores espaciados uniformemente entre el mínimo y el máximo del conjunto de datos.
- *count*: Devuelve el número de elementos de un conjunto distribuido.
- *sort*: Devuelve un conjunto distribuido ordenado por orden ascendente.
- *join*: Fusiona dos conjuntos de datos distribuidos por instancias y devuelve un nuevo conjunto de datos.
- *mapPartitions*: De forma similar a la función *map*, realiza una función definida por el usuario sobre cada partición de un conjunto distribuido. Tras esta transformación se crea un nuevo conjunto distribuido.
- *repartitionByRange*: Devuelve un nuevo conjunto distribuido particionado por las columnas dadas.
- *MulticlassMetrics*: Devuelve un conjunto de métricas para tareas de clasificación como la precisión y la matriz de confusión.
- $xgbClassifier$: Aprende un modelo *XGBoost* distribuido [CG16].

3

JUSTIFICACIÓN

En este capítulo se define la problemática que justifica la elaboración de esta tesis.

En los capítulos anteriores se ha introducido el problema de detección de anomalías y todas las variantes y distintos enfoques que pueden tener los problemas dentro de este campo. Por un lado, la complejidad a la hora de identificar las anomalías, sobre todo en problemas de series temporales y que da lugar a una posible alta presencia de FPs y por otro lado el amplio volumen de datos que se generan continuamente. Por lo tanto, se plantean dos lineas de trabajo en esta tesis para resolver los distintos casos de uso que se puedan dar:

1. Resolver problemas de detección de anomalías es una tarea compleja debido al gran desajuste en el número de observaciones normales frente a las anómalas. Además, para problemas de series temporales, la complejidad es mayor y al ser un nicho más concreto, hay menos soluciones, especialmente en series temporales multivariantes [STN18; STN19; GAS+21]. Asimismo, los resultados de los modelos que resuelven problemas de detección de anomalías en series temporales multivariantes, que componen el estado del arte actual, aún pueden ser mejorados. Una forma de mejorar los resultados de estos modelos es mitigando el número de falsos positivos obtenidos por el modelo. No obstante, aunque en la literatura hay planteamientos para realizar dicha mitigación, las soluciones están sujetas a problemas concretos y no se pueden generalizar [LZW+18; KY02; ZG20; HCL+18; LCO18; SHZ+19]. Ser capaz de plantear una solución que sea genérica y que por tanto, se pueda aplicar a cualquier problema es de gran interés y valor, ya que se puede aplicar en una mayor amplitud de casos de uso.

2. Aunque existen algunas propuestas en la literatura para la detección de anomalías en entornos *Big Data*, la mayoría de ellas siguen un enfoque de *clustering* similar [HNG+19; TBJ+20; HC14; RKC+19]. Otros estudios como el propuesto en [RAP16], realiza diferentes pasos para la selección de características y posterior etiquetado de datos, utilizando clasificadores como *SVM*, *Naïve Bayes* o *Random Forest*. El método propuesto en [LIN+16] sigue un enfoque estadístico supervisado para la detección de anomalías en dominios de consumo energético.

El método desarrollado en [DRO15], denominado *UNADA*, se basa en aplicar el algoritmo *DBSCAN* a varios subconjuntos de características. En [CDZ+16], los autores concluyen que hay poca variedad en la literatura actual. Todos los algoritmos disponibles son demasiado básicos o están demasiado centrados en un ámbito específico. Así, se requieren algoritmos para procesar esas grandes cantidades de datos de forma eficiente. Sin embargo, no existen métodos generales ni marcos de trabajo que ayuden en los problemas de detección de anomalías en entornos *Big Data*.

4

OBJETIVOS

En este capítulo se presentan los objetivos desarrollados a raíz de los problemas descritos en el campo de la detección de anomalías. Los objetivos surgen a partir del estudio inicial del problema de detección de anomalías y la observación de las distintas problemáticas que poseen tanto en la calidad de los resultados de los modelos como en el manejo de grandes volúmenes de datos. Por lo tanto, los objetivos presentes en esta tesis son:

1. La propuesta de una metodología de detección de anomalías en series temporales multivariantes en combinación de mitigación de FPs:
   a) El estudio del estado del arte en lo referente a detección de anomalías en series temporales.
   b) El estudio del estado del arte en cuanto a la mitigación de FPs.
   c) El diseño de la metodología propuesta.
   d) La experimentación que compone una comparación con resultados de otros estudios en conjuntos de datos de *benchmark* y su evaluación con un conjunto de datos real.
2. La propuesta de un conjunto de algoritmos distribuidos con diferentes mecanismos de trabajo para los problemas de detección de anomalías en entornos *Big Data*:
   a) El estudio de los algoritmos que detección de anomalías que componen el estado del arte actual del *Big Data*.
   b) El estudio de algoritmos de detección de anomalías que son interesantes para llevar a entornos *Big Data*.
   c) El diseño e implementación de los algoritmos elegidos.
   d) Experimentación que supone una comparación de la versión *Big Data* frente a la versión clásica de los algoritmos. Además de un caso de estudio con datos reales.

5

METODOLOGÍA PARA PROBLEMAS DE DETECCIÓN DE ANOMALÍAS EN SERIES TEMPORALES MULTIVARIANTES EN COMBINACIÓN DE MITIGACIÓN DE FALSOS POSITIVOS

En este capitulo se explicara en profundidad la metodología propuesta para problemas de detección de anomalías en series temporales multivariantes en combinación de mitigación de falsos positivos. También se detallará la experimentación realizada compuesta por una comparación con resultados de otros estudios en conjuntos de datos de *benchmark* y su evaluación con un conjunto de datos real. Para finalizar se detallarán las conclusiones extraídas. En la Sección 5.1 de detalla a fondo la metodología propuesta. Mientras que en la Sección 5.2 se detallan los conjuntos de datos usados, algoritmos y resultados obtenidos.

5.1 PROPUESTA

En esta sección, describimos en profundidad la metodología para resolver problemas de detección de anomalías en series temporales multivariantes. La Figura 5.1 representa el diagrama de flujo de la propuesta. La propuesta consta de dos etapas: En la primera etapa, una porción de la partición de entrenamiento se usa como entrada de un algoritmo de detección de anomalías. En la segunda etapa, los FPs se mitigan utilizando un clasificador entrenado con los VPs y FPs generados por el modelo de detección de anomalías. Así, para una serie de pruebas determinada, el resultado final estará compuesto por los resultados del detector de anomalías y del clasificador de mitigación de FPs.

Dado que en la metodología se hace uso de dos modelos, el detector de anomalías y el clasificador de FPs, también se realizan dos pasos de partición: uno para el modelo de detección de anomalías propiamente dicho y otro para el clasificador (etapa de mitigación de FPs). La partición completa generada por las dos etapas se representa en la Figura 5.2, nombrando además cada partición con su propia nomenclatura que se introducirá en los apartados posteriores.

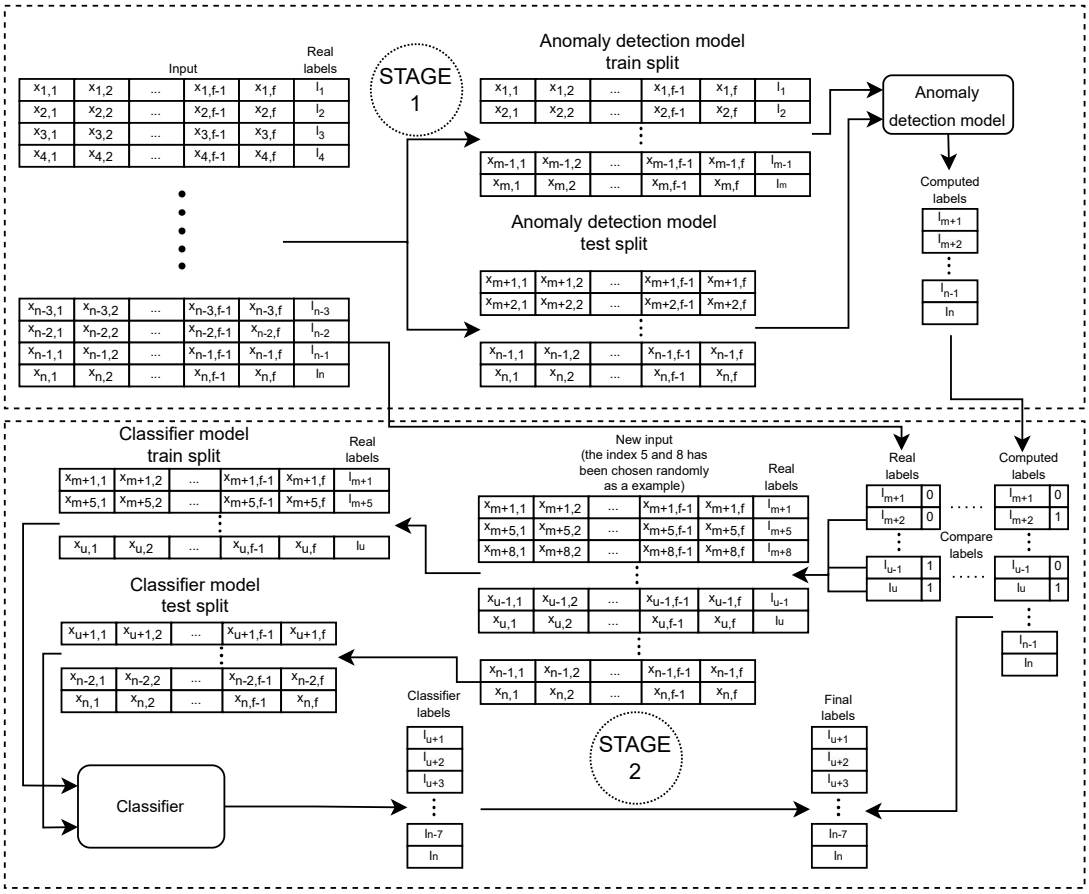


Figura 5.1: Diagrama de flujo de la metodología.

Las dos etapas implicadas en la metodología se detallan en sus secciones correspondientes: La Sección 5.1.1 describe la aplicación del algoritmo de detección de anomalías en la primera etapa, mientras que la Sección 5.1.2 se centra en la etapa de mitigación de FPs. Por último, la Sección 5.1.3 describe cómo crear el conjunto de validación para aplicar un clasificador de series temporales si es aconsejable mantener la temporalidad en la etapa de mitigación.

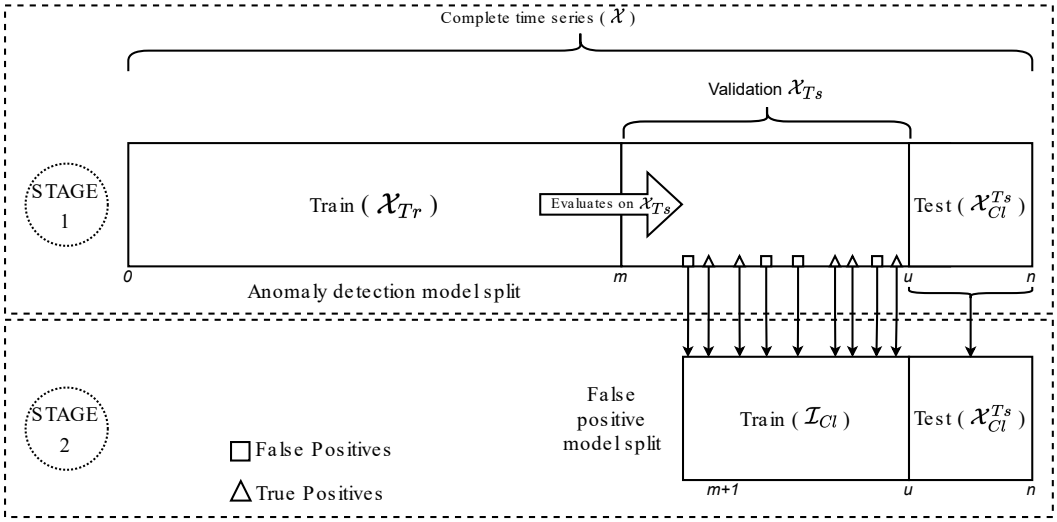


Figura 5.2: Diagrama de como se realiza la partición de los datos

5.1.1 *Etapa de detección de anomalías*

Como se ha mencionado anteriormente, la etapa de detección de anomalías consiste en entrenar un modelo base para clasificar los datos, que también proporcionará FPs que se utilizarán para entrenar el modelo para la etapa de mitigación de FPs. Para ello, el conjunto de datos se divide en tres porciones, *train* de 0 a m, *validation* de m a u y *test* de u a n, donde n es el tamaño del conjunto de datos completo. El modelo de detección de anomalías se entrena con la parte de entrenamiento y se calculan las etiquetas de las instancias de m a n, es decir, validación y prueba. Las etiquetas calculadas de la parte de validación se comparan con las etiquetas reales para obtener los FPs junto con los VPs, que se utilizarán para construir el nuevo conjunto de datos que se utilizará en la etapa de mitigación de los FPs. Las etiquetas calculadas como positivas para la parte de prueba se sustituirán en función de los resultados del modelo de mitigación de falsos positivos.

Es interesante mencionar que, dado que la metodología se centra en el tratamiento de series temporales, el problema *concept drift* puede tener un impacto negativo en el rendimiento. Por lo tanto, sería conveniente re-entrenar el modelo de detección de anomalías cada cierto tiempo para intentar evitarlo. Sin embargo, dependerá del conjunto de datos que se esté tratando.

El resto de esta sección describe la etapa de detección de anomalías, que se representa en la Figura 5.1 por el término *Stage 1*.

Dada una serie temporal $\mathcal{X} \subseteq \mathbb{R}$ (que representa las características de la serie temporal a analizar) con sus correspondientes etiquetas Y donde $y \in Y$ está contenido en $\{0, 1\}$ (que representa las etiquetas del conjunto de datos a analizar), se construyen dos subconjuntos denominados $\mathcal{X}_{Tr}$ (características del subconjunto de *train*) y $\mathcal{X}_{Ts}$ (características del subconjunto de *test*) con sus correspondientes subconjuntos de etiquetas Y_{Tr} (etiquetas del subconjunto de *train*) y Y_{Ts} (etiquetas del subconjunto de *test*). $\mathcal{X}_{Tr}$ corresponde a las primeras m instancias del conjunto de datos, donde m lo define el analista. $\mathcal{X}_{Ts}$ corresponde a las instancias de m a n (tamaño del conjunto de datos completo):

$$\mathcal{X}, Y \rightarrow \begin{cases} \mathcal{X}_{Tr}^{i}, Y_{Tr}^{i} : i = 0, ..., m \\ \mathcal{X}_{Ts}^{i}, Y_{Ts}^{i} : i = m+1, ..., n \end{cases} . \tag{5.1}$$

Se entrena una función f mediante $\mathcal{X}_{Tr}$ y se aplica al subconjunto $\mathcal{X}_{Ts}$, del que se obtienen puntuaciones que posteriormente se transforman en etiquetas. Sea S_{T_s} el resultado de aplicar la función de puntuación al subconjunto de prueba. Definimos L como un transformador de puntuaciones a etiquetas, por tanto una función de números reales al conjunto conformado por 0 y 1. El proceso se describe en la Ecuación 5.2:

$$f(\mathcal{X}_{Ts}) = \underbrace{S_{Ts}}_{\subseteq \mathbb{R}} \rightarrow L(S_{Ts}) = L(f(\mathcal{X}_{Ts})) = A_{Ts} : \forall a \in A_{T_s}\ a \in \{0, 1\}. \tag{5.2}$$

Tras estos pasos, se obtiene una primera clasificación del conjunto de datos (en algunos casos puede ser suficiente porque el número de falsos positivos obtenidos es muy bajo o inexistente), sin embargo, la metodología implica la aplicación de una etapa de mitigación de FPs para mejorar la calidad de estos resultados iniciales. Dicha etapa requiere una parte del conjunto de entrenamiento original para entrenar el modelo de mitigación, es decir, el subconjunto de *validation*, en el que se evalúa la técnica de detección de anomalías y se anotan y extraen los VPs y FPs en un nuevo conjunto de entrenamiento.

Los detalles de la etapa de mitigación se describen en la siguiente sección.

5.1.2 *Etapa de mitigación de falsos positivos*

Como ya se ha comentado en capítulos anteriores, en los problemas de detección de anomalías es habitual alcanzar un elevado número de FPs debido al gran desequilibrio existente en los datos. Por ello, aplicar técnicas que reduzcan el número de FPs obtenidos por el modelo de detección de anomalías ayuda a conseguir un mejor rendimiento en la solución del problema. Un mecanismo que da buenos resultados es el utilizado en [LCO18] y en [SHZ+19], que consiste en aplicar técnicas de clasificación con los resultados del modelo de detección de anomalías. De esta forma, el modelo de anomalías aprende las características más importantes y el modelo de clasificación (etapa de mitigación) refina los resultados.

Sin embargo, las propuestas realizadas en [LCO18; SHZ+19] se limitan a las características de los conjuntos de datos analizados y al algoritmo utilizado antes de la etapa de mitigación, ya que este algoritmo extrae características específicas de interés para el conjunto de datos que se está tratando. Por lo tanto, es interesante considerar un mecanismo de mitigación de FPs que no dependa del conjunto de datos y del algoritmo empleado.

La etapa de mitigación analiza los FPs obtenidos en la etapa de detección de anomalías, utilizando un modelo de clasificación para datos tabulares. El modelo se entrena con las instancias originales de la parte *validation*, pero sólo con las correspondientes a FPs y VPs. Una vez entrenado para detectar FPs, el modelo de mitigación de FPs calcula las etiquetas de la parte de *test* y se comparan las etiquetas calculadas por el modelo de detección de anomalías. Las etiquetas predichas corresponden a las etiquetas del modelo de detección de anomalías, con la excepción de que las que fueron etiquetadas como positivas por el modelo de detección de anomalías, se sustituyen por las etiquetas calculadas por el modelo de clasificación (modelo de mitigación de FPs).

A continuación se detalla la etapa de mitigación de FPs (ver *Stage 2* en la Figura 5.1) como una continuación de la etapa de detección de anomalías. Las primeras u instancias de Y_{Ts} (etiquetas del subconjunto de *test*) y A_{Ts} (etiquetas calculadas del subconjunto de prueba) se comparan entre sí para obtener qué instancias corresponden a los FPs y a las anomalías reales. Sea $\mathcal{I}_{Cl}$ el conjunto de índices de FPs y anomalías reales,

$$\mathcal{I}_{Cl} = \{i : i \in [0, u) : \begin{cases} y^i_{Ts} = 0 \text{ y } a^i_{Ts} = 1 \\ y^i_{Ts} = 1 \end{cases} \}. \tag{5.3}$$

A partir de los índices $\mathcal{I}_{Cl}$, se construye un nuevo conjunto de datos $\mathcal{X}^{Tr}_{Cl}$ (que representan las características del nuevo conjunto de datos utilizado para entrenar al clasificador) con sus correspondientes etiquetas $\mathcal{Y}^{Tr}_{Cl}$ (que representan las etiquetas del nuevo conjunto de datos utilizado para entrenar al clasificador). El subconjunto de *test* para el clasificador compuesto por $\mathcal{X}^{Ts}_{Cl}$ (que representa las características del subconjunto de *test* para el clasificador) y $\mathcal{Y}^{Ts}_{Cl}$ (que representa las etiquetas del subconjunto de *test* para el clasificador) son las instancias restantes de u a n. Por lo tanto, la definición de los conjuntos será:

$$\mathcal{X}^{Tr}_{Cl} = \{x^i_{Ts} \;\; i \in \mathcal{I}_{Cl}\}, \tag{5.4}$$
$$\mathcal{Y}^{Tr}_{Cl} = \{y^i_{Ts} \;\; i \in \mathcal{I}_{Cl}\}, \tag{5.5}$$
$$\mathcal{X}^{Ts}_{Cl} = \{x^i_{Ts} \;\; i \in u, ..., n\} \; y \tag{5.6}$$
$$\mathcal{Y}^{Ts}_{Cl} = \{y^i_{Ts} \;\; i \in u, ..., n\}. \tag{5.7}$$

El último paso será entrenar una función g con el nuevo conjunto $\mathcal{X}^{Tr}_{Cl}$ y aplicarla al nuevo conjunto $\mathcal{X}^{Ts}_{Cl}$. El resultado de la primera será un conjunto de etiquetas que se compararán con las etiquetas obtenidas por la función anterior f. Por lo tanto, la definición de los conjuntos será:

$$\mathcal{Y}_F = \begin{cases} y^i_{Rf} \text{ si } a^i = 1 \\ a^i \text{ si } a^i = 0 \end{cases} \quad i = u, ..., n$$
$$y^i_{Rf} \in \mathcal{Y}_{Rf}, \; a^i \in A_{Ts}. \tag{5.8}$$

El nuevo conjunto $\mathcal{Y}_F$ representa la salida de la metodología, es decir, las etiquetas que corresponden a la última porción de la serie temporal. Estas etiquetas son las calculadas por el modelo de detección de anomalías, pero las etiquetadas como positivas son re-etiquetadas por la salida del modelo de mitigación de FPs.

$$Y_F = \begin{cases} y^i_{Rf} \text{ si } a^i = 1 \\ a^i \text{ si } a^i = 0 \end{cases} \quad i = u, ..., n$$
$$y^i_{Rf} \in Y_{Rf},\ a^i \in A_{Ts}. \tag{5.9}$$

De este modo, las instancias potencialmente más problemáticas son re-etiquetadas por un modelo que se ha especializado en ese tipo de observaciones. Por lo tanto, el número de FPs es capaz de reducirse, ya que esas observaciones son re-clasificadas por un modelo que sólo se ha entrenado con esas observaciones, además de las verdaderamente anómalas, que deben ser diferentes de las observaciones etiquetadas como FPs, ya que en realidad son normales.

5.1.3 *Mantener la temporalidad en la etapa de mitigación de falsos positivos*

Dado que nos centramos en un problema de detección de anomalías en series temporales, la aplicación de la técnica de mitigación de FPs basada en datos tabulares, descrita en la sección anterior tendrá como consecuencia la pérdida de la temporalidad de los datos, ya que lo más probable es que los FPs obtenidos en el primer modelo (etapa de detección de anomalías) no sean consecutivos.

Esto no tiene por qué ser un problema, sin embargo, se proponen dos enfoques diferentes teniendo en cuenta esta característica. El primero sólo toma los FPs y VPs para construir $\mathcal{I}_{Cl}$ como se explica en la Figura 5.2. El segundo enfoque pretende mantener la temporalidad en los datos, permitiendo utilizar un clasificador de series temporales como clasificador de FPs, en lugar de un clasificador tabular como se describe en la sección anterior. Para ello, consideramos las instancias s previas a un FP o VP, construyendo así un conjunto de datos de secuencias temporales (Figura 5.3) con el objetivo de explotar mejor la dimensión temporal original de los datos. Ambas versiones se analizan en la sección 5.2.3.

Es importante señalar que aunque la metodología se centra en series temporales, la etapa de mitigación basada en datos tabulares puede aplicarse tanto a problemas de series temporales como no temporales debido a la ausencia de temporalidad en el entrenamiento del mitigador de FPs.

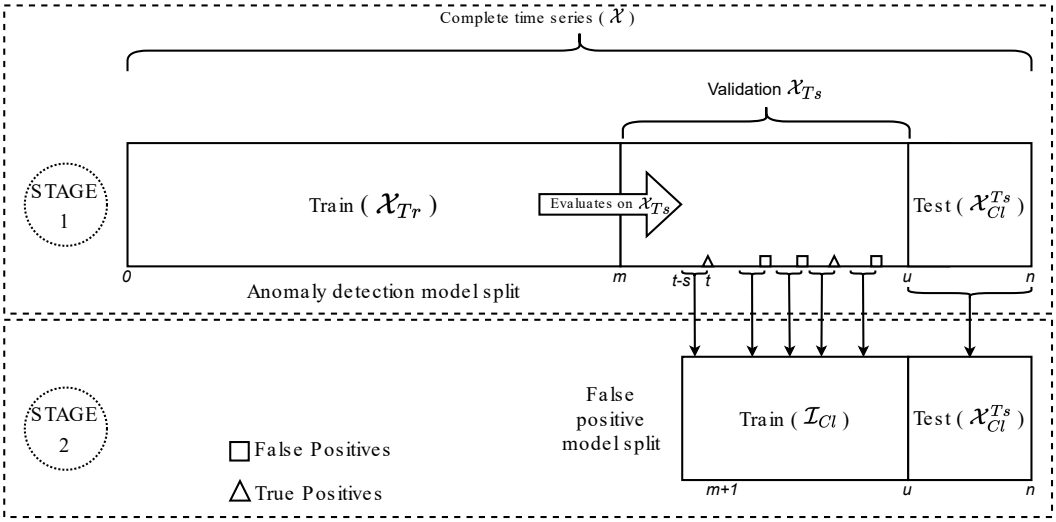


Figura 5.3: Si se quiere mantener la temporalidad al entrenar al mitigador de FPs, se debe tomar una ventana de instancias que precedan a un FP o VP al construir $\mathcal{I}_{Cl}$.

5.2 EXPERIMENTACIÓN

En esta sección se presentan los conjuntos de datos utilizados en la experimentación. También se realizará un análisis del funcionamiento de distintos algoritmos a la hora de usarlos en la metodología.

5.2.1 *Conjuntos de datos analizados*

CONJUNTO DE DATOS DE *benchmark* Para la comparación con otras técnicas se han utilizado dos conjuntos de datos analizados previamente en la literatura. El primero es un conjunto de datos construido a partir de los conjuntos de datos Skoltech Anomaly Benchmark (SKAB) diseñados para evaluar los algoritmos de detección de anomalías [KK20]. SKAB contiene hasta 35 conjuntos de datos en formato *csv*, se han fusionado los conjuntos de datos de la sección valve1 que contiene un gran número de mediciones para un solo día. Por lo tanto, el conjunto de datos final se obtiene con **18.163 observaciones** y **10 características**.

El segundo conjunto de datos se ha extraído de *kaggle* [Sca20]. No hay información en cuanto a lo que el conjunto de datos representa, sin embargo, es un conjunto de datos multivariante en el que hay un gran número de observaciones y cuyas muestras están bastante desequilibradas, ya que sólo el 0,09 % de las observaciones son anómalas. Por lo tanto, es un conjunto de datos ideal con el que probar la eficacia de la metodología propuesta, además de tener la posibilidad de comparar con estudios ya realizados con este conjunto de datos. El conjunto de datos contiene **509.633 observaciones** y **11 características**. Ambos conjuntos de datos han sido preprocesados, escalándolos al rango cero-uno. No se ha eliminado ninguna característica.

CASO DE ESTUDIO REAL: ARCELORMITTAL Los datos de uno de los mantenimientos predictivos de la maquinaria de ArcelorMittal [1] han sido facilitados directamente por la empresa. Estas maquinarias sufren averías con frecuencia, debido al entorno hostil en el que se despliegan estos activos. El principal problema es que algunas averías implican que la maquinaria esté parada durante varios días. Ese periodo en el que la maquinaria está parada supone grandes pérdidas para la empresa. Los datos proporcionados se componen de la información de los sensores de una de sus máquinas de producción, así como de información relacionada, como información contextual, averías, etc. Las características del conjunto de datos se construyen a partir de la información de los sensores individuales. Estas variables, modelan diferentes propiedades de la maquinaria, siendo la mayoría de ellas de naturaleza real. El conjunto de datos tiene 40 millones de instancias correspondientes a casi 2 años de mediciones, y cada instancia tiene más de 100 variables. El objetivo es detectar estos fallos con la suficiente antelación para minimizar los periodos de reparación de estas máquinas. Se ha trabajado con subconjuntos de un mes, ya que es suficiente para representar el comportamiento del conjunto de datos completo. Además, entrenar el modelo cada mes evita problemas como el *concept drift*, ya que dependiendo del periodo del año el rango de valores de las variables es diferente, por ejemplo, en verano valores como la temperatura o la vibración serán más altos que en invierno. El conjunto de datos se ha preprocesado, escalándolo al rango cero-uno. Además, se han eliminado 6 características del total de 112, porque tienen un valor constante. Así pues, el conjunto de datos utilizado tiene **168.956 observaciones** y **106 características**.

1 El conjunto de datos completo y anonimizado, con una breve explicación de su estructura, está disponible en https://github.com/ari-dasci/OD-TINA

5.2.2 *Algoritmos usados en la experimentación*

Además del modelo TCN, hemos seleccionado dos novedosos modelos de redes neuronales recurrentes para la etapa de detección de anomalías: *YiboGao* [GWL21] y *WeiXiaoyan* [WZZ+19]. Sin embargo, como ya se ha mencionado, cualquier método de detección de anomalías puede aplicarse a la metodología. Lo mismo es aplicable a la etapa de mitigación de FPs, por lo que se han seleccionado varios algoritmos clásicos además de algoritmos más recientes para probar diferentes enfoques.

En la Tabla 5.1 podemos observar la lista completa de los parámetros optimizados para todos los algoritmos usados. Los parámetros para la metodología son los parámetros necesarios para un modelo de detección de anomalías y para un algoritmo de clasificación, estos parámetros serán diferentes según el algoritmo de mitigación a utilizar. Los problemas de detección de anomalías dependen de los datos. Debido a este hecho, los parámetros difícilmente pueden generalizarse a otro problema. Por lo tanto, para encontrar el rendimiento óptimo, hay que calcular la combinación correcta de parámetros. Para esta tarea, se ha elegido un marco de optimización de hiper-parámetros llamado *Optuna* [ASY+19] para realizar una optimización de hiper-parámetros. Este marco tiene como objetivo construir el espacio de búsqueda de parámetros para los hiper-parámetros de forma dinámica. Se ha elegido la medida *F1-score* como valor objetivo a optimizar.

Algoritmo	Parámetros
Detección de anomalías	
TCN	batch_size, dropout, epochs, kernel_size, levels, learning_rate, optimizer, number_of_hidden_units
YiboGao	batch_size, epochs
WeiXiaoyan	batch_size, epochs
Clasificación	
KNN	n_neighbors, weights, algorithm, leaf_size, p (power parameter for the Minkowski metric)
SVM	C (regularization parameter), kernel, tol (tolerance for stopping criterion), gamma
RANDOM FOREST	min_samples_split, criterion, min_samples_leaf, min_weight_fraction_leaf, max_depth, n_estimators
XGBOD	learning_rate, min_child_weights, max_delta_step, subsample, subsample_bytree, gamma, max_depth, n_estimators
XGBOOST	booster, eta (learning_rate), min_child_weight, max_delta_step, sampling_method, reg_lambda, alpha, num_round, threshold, gamma, depth
TABNET	n_d (dimension of the prediction layer), n_a (dimension of the attention layer), n_steps, learning_rate, gamma
NODE	num_layers, num_trees, dropout, threshold_init_beta, learning_rate, gamma, depth
1D-CNN	batch_size, epochs

Tabla 5.1: Lista completa de todos los parámetros que han sido optimizados para todos los algoritmos durante la experimentación

5.2.3 *Análisis de los conjuntos de datos de benchmark*

En esta sección se analizarán los dos conjuntos de datos de *benchmark*, por un lado se hará un análisis de las métricas de evaluación y por otro de la mitigación de

falsos positivos. Además se analizará cómo afecta la mitigación de FPs a la métrica. También se hará un estudio manteniendo la temporalidad en la etapa de mitigación.

5.2.3.1 *Análisis de la métrica de evaluación*

La Tabla muestra la métrica *F1-score* de los modelos de detección de anomalías, los resultados de la literatura y los distintos modelos de clasificación que se han evaluado para la etapa de mitigación de FPs. Las dos primeras filas corresponden a loa resultados de la literatura y a los modelos de detección de anomalías, los cuales no contienen una etapa de mitigación de FPs. Las otras filas se refieren a los modelos de clasificación para la mitigación de FPs.

Los resultados obtenidos, que pueden observarse en la Tabla 5.2, se explican a continuación:

- Para el conjunto de datos *SKAB* y los resultados del modelo de *TCN*, la etapa de mitigación mantiene los mismos resultados a excepción de los modelos *SVM* y *TABNET*, que mejora los resultados de modelo de *TCN*.
- Este comportamiento se debe a que los resultados del modelo de *TCN* contienen sólo 3 FPs (ver Tabla 5.3), por lo que la mejora que se puede obtener es muy baja.
- Los resultados de WeiXiaoyan sí contienen un número notable de FPs, por lo que más modelos consiguen mejorar la métrica *F1-score* obtenida y en mayor cantidad.
- En el caso del conjunto de datos de *Kaggle*, en el que todos los modelos obtienen un *F1-score* casi perfecta, pero el número de FPs es elevado (15.164), la etapa de mitigación siempre consigue mejorar los resultados.

Los algoritmos de detección de anomalías por sí solos ya son capaces de mejorar los resultados de estudios anteriores. Además, la etapa de mitigación consigue mejorar aún más el *F1-score*. Es importante destacar el hecho de que si el número de FPs obtenidos por el modelo de detección de anomalías es muy bajo, será más difícil que la etapa de mitigación consiga mejores resultados. Sin embargo, dependiendo del algoritmo utilizado, es posible mejorar los resultados anteriores. Como afirman los autores en [SA22], los modelos de *deep learning* no obtienen necesariamente los mejores resultados en problemas de clasificación.

		F1-Score					
	Conjunto de datos	SKAB			Kaggle		
	Resultados del Paper Original	0.79			0.985		
No hay mitigación	Algoritmos de Detección de Anomalías	TCN	Yibogao	WeiXiaoyan	TCN	Yibogao	WeiXiaoyan
		0.9982	0.9515	0.9298	0.9921	0.9914	0.9896
Mitigación	KNN	0.9982	0.9496	0.9333	0.9997	0.9988	0.9987
	SVM	**0.9989**	0.9518	0.9296	**0.9999**	**0.9999**	0.9997
	RANDOM FOREST	0.9982	0.9515	0.9390	0.9987	0.9985	0.9987
	XGBOD	0.9982	0.9503	**0.9409**	**0.9999**	**0.9999**	**0.9999**
	XGBOOST	0.9982	0.9416	0.9260	**0.9999**	**0.9999**	**0.9999**
	TABNET	**0.9989**	**0.9525**	0.9296	**0.9999**	0.9988	0.9994
	NODE	0.9982	0.9515	0.9387	0.9987	**0.9999**	0.9994
	1D-CNN	0.9982	0.9508	0.9278	0.9987	0.9988	0.9987

Tabla 5.2: *F1-score* para los conjuntos de datos *SKAB* y *Kaggle*. Las celdas en gris indican un peor resultado con respecto al modelo base de detección de anomalías sin mitigación, mientras que los números en negrita indican el mejor resultado para ese conjunto de datos.

5.2.3.2 *Análisis de la mitigación de falsos positivos*

La Tabla 5.3 muestra el número de FPs obtenidos para los modelos de detección de anomalías y para los diferentes modelos de clasificación. La primera fila corresponde a los modelos de detección de anomalías sin etapa de mitigación de FPs. Las demás filas se refieren a los modelos de clasificación con mitigación de FPs.

Como demuestran los resultados, la etapa de mitigación de FPs reduce el número de FPs obtenidos por el modelo de detección de anomalías en la mayoría de los casos. Como se comentó en la sección anterior, el número de FPs obtenidos por el modelo de detección de anomalías influye en el rendimiento del clasificador. El número de FPs mitigados en el conjunto de datos *Kaggle* es mayor debido a ésta razón.

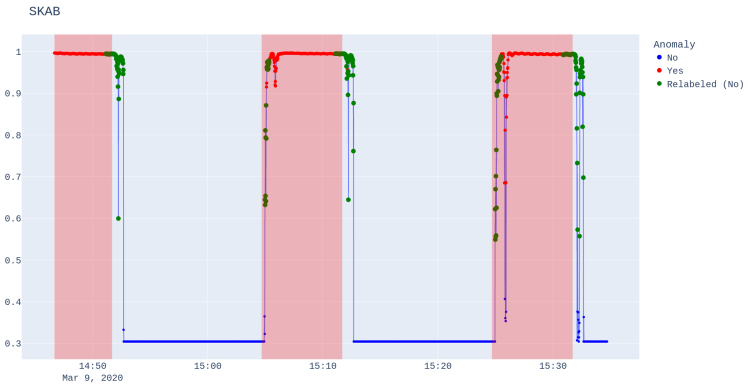
En la Figura 5.4, mostramos una representación gráfica de cómo la etapa de mitigación de FPs reduce los FPs en el conjunto de datos *SKAB*. La Figura 5.4a muestra cómo el algoritmo *XGBOOST* es capaz de reducir completamente las observaciones fuera del periodo anómalo, es decir, de mitigar los FPs. Sin embargo, también reclasifica los puntos realmente anómalos como normales, por lo que el *F1-score* final obtenido es inferior al obtenido antes de la etapa de mitigación. Por otro lado, en la Figura 5.4b el algoritmo *XGBOD*, que sí consigue mejorar el *F1-score*, no está

	Número de Falsos Positivos					
	Conjunto de datos	SKAB			Kaggle		
No hay mitigación	Algoritmos de	TCN	Yibogao	WeiXiaoyan	TCN	Yibogao	WeiXiaoyan
	Detección de Anomalías	3	16	158	15,164	15,276	15,452
Mitigación	KNN	3	14	58	1	5	5
	SVM	**1**	14	38	**0**	**0**	3
	RANDOM FOREST	3	16	101	6	8	6
	XGBOD	3	13	71	**0**	**0**	**0**
	XGBOOST	3	**7**	**4**	**0**	**0**	**0**
	TABNET	**1**	11	15	**0**	6	2
	NODE	3	16	119	8	**0**	5
	1D-CNN	3	15	47	7	6	7

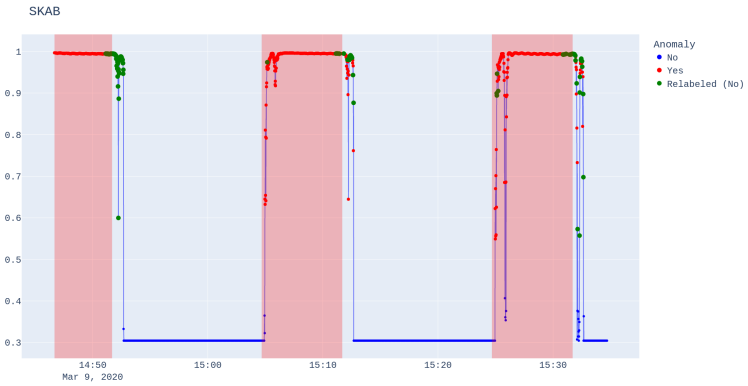
Tabla 5.3: Cantidad de FPs obtenidos para los conjuntos de datos *SKAB* y *Kaggle*. Las celdas en gris indican un peor resultado con respecto al modelo base de detección de anomalías sin mitigación, mientras que los números en negrita indican el mejor resultado para ese conjunto de datos.

mitigando todos los FPs tras los dos últimos periodos anómalos, pero no está dejando de reclasificar las observaciones VPs. La Figura 5.4c muestra cómo el algoritmo *TABNET*, que parte de un modelo previo más robusto, reduce los pocos FPs sin encontrar problemas para reclasificar los puntos realmente anómalos.

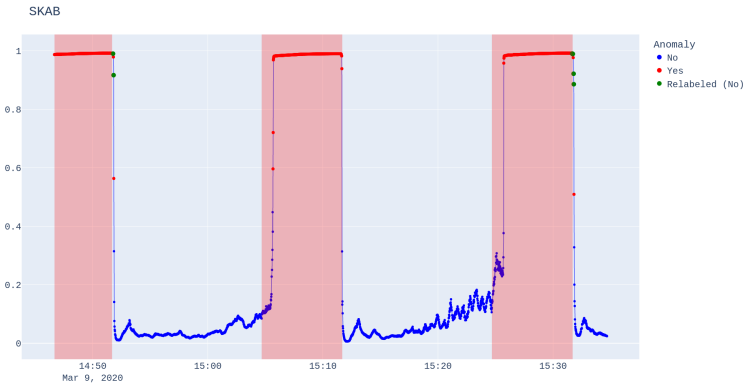
Por lo tanto, la etapa de mitigación es capaz de mejorar el rendimiento de los algoritmos de detección de anomalías. La calidad de la mejora vendrá determinada por el rendimiento del algoritmo de detección de anomalías previo, ya que depende del número de FPs que genere. Además, es interesante aplicar un algoritmo u otro en función del grado de mitigación a aplicar.



(a) Gráfico de los resultados de aplicar el algoritmo XGBOOST en la etapa de mitigación de FPs a partir de los resultados de WeiXiaoyan.



(b) Gráfico de los resultados de aplicar el algoritmo XGBOD en la etapa de mitigación de FPs a partir de los resultados de WeiXiaoyan.



(c) Gráfico de los resultados de aplicar el algoritmo TABNET en la etapa de mitigación de FPs a partir de los resultados de WeiXiaoyan.

Figura 5.4: Gráficos que muestran el comportamiento de los algoritmos de la etapa de mitigación. Los puntos rojos representan las observaciones detectadas como anómalas, los azules las observaciones representadas como normales y los verdes las observaciones que han sido re-etiquetadas. Las bandas rojas representan la zona anómala real del conjunto de datos.

5.2.3.3 *Comportamiento de la métrica de evaluación a la hora de mitigar falsos positivos*

Los resultados de las Tablas 5.2, 5.3, 5.5 y 5.6 muestran que dependiendo del algoritmo utilizado, el número de FPs y la mejora en el *F1-score* es diferente. Además, en algunos casos, el número de FPs se reduce, pero el *F1-score* obtenido es de peor calidad.

El *F1-score* está relacionado con la precisión y la sensibilidad. Por lo tanto, el *F1-score* puede mejorar aunque el número de FPs sea menor, ya que el número de FNs aumenta. Un buen ejemplo del equilibrio entre FPs y FNs puede observarse en el comportamiento del modelo *KNN* para el conjunto de datos *SKAB* y para los resultados de *YiboGao*. Al aplicar este modelo a estos conjuntos de datos, los FPs se reducen en casi un tercio, mientras que el *F1-score* sólo varía en una décima parte. Como ya se ha mencionado, si se mantiene el *F1-score*, una disminución de los FPs implica un aumento de los FNs. Sin embargo, algunos algoritmos como *TABNET* para el conjunto de datos de *Arcelor* y para los resultados del modelo *TCN* consiguen mitigar completamente los FPs sin aumentar los FNs. La ligera mejora en *F1-score* se debe a que el número de FPs mitigados es bajo.

Además, dependiendo del tipo de problema a tratar, puede ser interesante reducir el número de FPs a pesar de una pequeña reducción en la medida de precisión. La razón es que cuando se trata de series temporales, las anomalías suelen estar agrupadas en una ventana temporal, de modo que un pequeño aumento del número de FNs puede significar una detección ligeramente posterior (si se produce al principio del periodo anómalo) o incluso carecer de importancia si se produce al final del periodo anómalo. Sin embargo, mitigar los FPs puede suponer la eliminación de falsas alarmas, evitando así paradas innecesarias del sistema.

5.2.3.4 *Considerando la temporalidad en la etapa de mitigación de FPs: Caso de estudio de SKAB*

Como se indicó en la Sección 5.1.2, el clasificador encargado de aprender a mitigar los FPs puede contextualizarse mejor si se le proporcionan los ejemplos previos a un FP en la serie temporal. En esta sección, nuestro objetivo es analizar si estas instancias añadidas, que aumentan el tiempo de entrenamiento y la complejidad en la etapa de mitigación de FPs, se traducen en una mejora del rendimiento.

Para ello, probamos el comportamiento de un nuevo conjunto de datos construido a partir de las instancias que preceden a los FPs. Hemos seleccionado los resultados

del modelo *WeiXiaoyan* para el conjunto de datos *SKAB* para la etapa de detección de anomalías.

Para explotar mejor la temporalidad de la ventana de instancias que preceden a un FP, se ha entrenado un modelo *TCN* para la etapa de mitigación de FPs en lugar de un clasificador tabular, ya que este último no tendría en cuenta la temporalidad de los datos. La Figura 5.3 muestra cómo funciona esta construcción. Se ha elegido el conjunto de datos *SKAB*, variando el número de instancias anteriores a un FP incluidas en la ventana.

Tamaño del intervalo (s)	10	20	30	40
F1-score	0.9314	0.9346	0.9352	**0.9387**
FP	**17**	29	41	68

Tabla 5.4: Resultados obtenidos por el modelo de mitigación *TCN* aplicando diferentes periodos de tiempo antes de cada FP. La experimentación se ha realizado con el conjunto de datos *SKAB* y para los resultados de *WeiXiaoyan*. La frecuencia de muestreo del conjunto de datos *SKAB* es de 1 segundi.

Los resultados de la Tabla 5.4 indican que a medida que aumenta el intervalo, es decir, que hay más observaciones normales en el conjunto de datos, se reduce la mitigación de los FP. Sin embargo, el *F1-score* aumenta. Con un intervalo más pequeño, la proporción de observaciones anómalas es mayor, lo que permite al modelo clasificar mejor las primeras.

Al aumentar el intervalo y, por tanto, el número de observaciones negativas, el modelo se centra más en éstas, logrando un *F1-score* más alta, ya que mejora la precisión de las instancias normales a costa de clasificar erróneamente más observaciones anómalas. Este comportamiento puede ser útil dependiendo del conjunto de datos y de los requisitos del problema a resolver.

El uso de la temporalidad en la etapa de mitigación sigue mejorando los resultados sin mitigación, pero su comportamiento es diferente al de los demás clasificadores tabulares para la mitigación de FPs considerados en esta investigación. El *F1-score* obtenido utilizando la temporalidad es muy similar al *F1-score* de los mejores modelos de clasificación, pero a costa de un número significativamente mayor de FPs. La explicación de este comportamiento puede deberse a dos factores. El primer factor es que el nuevo conjunto de datos construido a partir de intervalos de observaciones no mantiene totalmente la temporalidad, ya que los intervalos pueden estar separados.

El otro factor es que las anomalías detectadas por ambos métodos (con y sin temporalidad) sean diferentes, en cuyo caso ambos métodos podrían complementarse para obtener un posible mejor rendimiento. El segundo hecho factor se analizará como línea de trabajo futuro.

5.2.4 *Análisis del conjunto de datos real: ArcelorMittal*

En esta sección se analizará el conjunto de datos real de *ArcelorMittal*, por un lado se hará un análisis de las métricas de evaluación y por otro de la mitigación de falsos positivos.

5.2.4.1 *Análisis de la métrica de evaluación*

La Tabla 5.5 muestra el *F1-score* para los modelos de detección de anomalías y los diferentes modelos de clasificación que han sido evaluados para la etapa de mitigación FPs. La primera fila corresponde a los modelos de detección de anomalías que no contienen una etapa de mitigación de FPs. Las otras filas se refieren a los modelos de clasificación para la mitigación de FPS.

Como puede observarse en la Tabla 5.5, de forma similar al comportamiento en los conjuntos de datos de referencia, el modelo *TCN* es el modelo que obtiene mejores resultados antes de aplicar la etapa de mitigación de FPs. Sin embargo, para este conjunto de datos, el valor del *F1-score* para los 3 algoritmos muestra que los detectores de anomalías no consiguen un buen rendimiento. Al igual que con los otros conjuntos de datos, el número de FPs obtenidos por el modelo *TCN* es bajo (52), por lo que incluso si algunos algoritmos consiguen eliminarlos por completo, la mejora del *F1-score* es muy pobre. En cambio, cuando se utilizan *YiboGao* y *WeiXiaoyan* como modelos de detección de anomalías, que tienen un *F1-score* mucho más bajo y un mayor número de FPs, la etapa de mitigación consigue mejorar mucho los resultados obtenidos. De hecho, se obtienen mejores resultados que los obtenidos utilizando el modelo de *TCN* como modelo de detección de anomalías a pesar de obtener resultados iniciales significativamente mejores. Por tanto, se demuestra que para la etapa de mitigación es deseable que el modelo de la etapa de detección de anomalías obtenga un número significativo de FPs para que el modelo de la etapa de mitigación sea capaz de entrenarse correctamente.

		F1-Score		
	Conjunto de datos	Arcelor		
No hay mitigación	Algoritmos de Detección de Anomalías	TCN	Yibogao	WeiXiaoyan
		0.5939	0.4348	0.3508
Mitigación	KNN	0.5899	0.6378	0.6103
	SVM	0.5969	0.6320	**0.7122**
	RANDOM FOREST	0.5975	0.5899	0.5830
	XGBOD	0.5971	0.6522	0.7074
	XGBOOST	0.5896	0.6410	0.5830
	TABNET	**0.5988**	**0.6771**	0.6989
	NODE	**0.5988**	0.5972	0.5922
	1D-CNN	0.5945	0.5908	0.5912

Tabla 5.5: *F1-score* para el conjunto de datos del mundo real (*Arcelor*). Las celdas en gris indican un peor resultado con respecto al modelo base de detección de anomalías sin mitigación, mientras que los números en negrita indican el mejor resultado para cada algoritmo de detección de anomalías.

5.2.4.2 *Análisis de la mitigación de falsos positivos*

La Tabla 5.6 muestra el número de FPs obtenidos para los modelos de detección de anomalías y para los diferentes modelos de clasificación que han sido evaluados para la etapa de mitigación de FPs. La primera fila corresponde a los modelos de detección de anomalías, sin etapa de mitigación de FPs. Las otras filas se refieren a los modelos de clasificación para la mitigación de FPs.

Los resultados son muy similares a los obtenidos para los conjuntos de datos de referencia; de hecho, para este conjunto de datos, todos los algoritmos reducen el número de FPs obtenidos por el algoritmo de detección de anomalías. Independientemente del algoritmo de detección de anomalías utilizado, el número de FPs se reduce a 0 dependiendo del algoritmo de clasificación utilizado. Al igual que en los conjuntos de datos de referencia, un menor número de FPs no implica una mejor medida de la precisión. El mejor *F1-score* obtenido corresponde al algoritmo de detección de anomalías *Yibogao* tras aplicar el algoritmo *TABNET* en la etapa de

mitigación. Sin embargo, el número de FPs obtenido es de 208, muy lejos del valor 0 alcanzado por otros algoritmos de clasificación. Asimismo, la reducción del número de FPs obtenidos en comparación con el algoritmo de detección de anomalías es bastante notable (208 frente a 2.479).

Número de Falsos Positivos

	Conjunto de datos	Arcelor		
No hay mitigación	Algoritmos de Detección de Anomalías	TCN	Yibogao	WeiXiaoyan
		52	2,479	2,926
Mitigación	KNN	32	488	512
	SVM	20	235	395
	RANDOM FOREST	14	**0**	**0**
	XGBOD	18	356	488
	XGBOOST	16	142	**0**
	TABNET	**0**	208	301
	NODE	**0**	82	18
	1D-CNN	23	142	165

Tabla 5.6: Cantidad de FPs obtenidos para el conjunto de datos del mundo real (*Arcelor*). Los números en negrita indican el mejor resultado para cada algoritmo de detección de anomalías.

5.3 CONCLUSIONES

Este propuesta presenta una metodología novedosa para tratar los FPs en problemas de detección de anomalías en series temporales multivariantes, que refina y mejora la calidad de los resultados. Se divide en dos etapas. En primer lugar, se entrena un modelo de detección de anomalías en una fracción de la partición de entrenamiento de la serie temporal y clasifica la parte de validación restante. La segunda etapa construye un modelo de clasificación (mitigador de FPs) a partir de una parte de los FPs y VPs obtenidos por el modelo anterior en la porción de validación. Ante nuevas

observaciones, el mitigador de FPs revisa las predicciones positivas del modelo detector de anomalías, corrigiendo el FP cuando es necesario.

Los resultados obtenidos muestran que los modelos de detección de anomalías elegidos en la metodología son capaces de obtener resultados de calidad para conjuntos de datos de series temporales, además, la etapa de mitigación consigue una reducción de la tasa de FPs. Asimismo, se ha demostrado que la estrategia de mitigación de FPs mediante la aplicación de un clasificador funciona para una amplia variedad de algoritmos. De hecho, el algoritmo de elección para la etapa de mitigación será importante en función del resultado final que se desee obtener (reducir el mayor número de FPs, mejorar la medida de precisión o un equilibrio entre ambos). El número de FPs obtenidos por el modelo de detección de anomalías ha resultado ser relevante, ya que un número bajo de FPs limita el rendimiento de la etapa de mitigación. Por lo tanto, se demuestra la fiabilidad de la metodología, así como que permite la posibilidad de continuar el estudio de la mitigación de la FPs también en conjuntos de datos no temporales.

6

ALGORITMOS DE DETECCIÓN DE ANOMALÍAS PARA ENTORNOS *BIG DATA*

En este capítulo se proponen cuatro algoritmos distribuidos de detección de anomalías en entornos *Big Data*. Estos algoritmos han sido diseñados siguiendo el paradigma *MapReduce*, donde todos los procesos se realizan de forma distribuida. Se detallará el diseño e implementación de los cuatro algoritmos. Además se incluye una sección de experimentación en la que se probará la calidad y la aplicabilidad de los algoritmos. Por último, se detallarán las conclusiones extraídas de la propuesta. En la Sección **??** se describen las primitivas de *Spark* empleadas para en el diseño e implementación de los algoritmos. La Sección 6.1 recoge el diseño en profundidad de todos los algoritmos propuestos. Por último, en la Sección 6.2 se desarrolla el análisis comparativo de la experimentación realizada para probar la eficacia de los algoritmos propuestos.

6.1 ALGORITMOS DE DETECCIÓN DE ANOMALÍAS PARA BIG DATA

En esta sección se detallará el diseño distribuido elegido para estos algoritmos clásicos a entornos *Big Data*. Se realizará tanto una explicación detallada del proceso además de mostrar un pseudo-código explicativo.

6.1.1 *Histogram-Based Outlier Score For Big Data: HBOS_BD*

Histogram-Based Outlier Score (*HBOS*) [GD12] es un método de detección de anomalías basado en histogramas. *HBOS* puede procesar conjuntos de datos multivariantes y tiene una baja complejidad en tiempos de ejecución. *HBOS* construye un histograma para cada característica de los datos, cada histograma se divide en intervalos y el *score* de cada instancia (p) se calcula en función de la altura del intervalo en el que se encuentra.

La Ecuación 6.1 muestra cómo se calcula este *score*:

$$f(x) = \sum_{i=0}^{k} \log \frac{1}{hist_i(p)} \tag{6.1}$$

HBOS propone dos alternativas para procesar las características numéricas:

- Estático: Calcula un histograma estándar utilizando *k bins* de igual anchura. La altura de los intervalos es una estimación basada en la cantidad relativa de muestras dentro de cada intervalo.
- Dinámico: Los valores se ordenan y un número fijo de $\frac{N}{k}$ valores consecutivos se agrupan en un solo *bin*. k representa el número de contenedores y N la longitud de los datos. El área de cada *bin* representa el número de instancias en cada *bin*, que es el mismo para todos los *bins*. La anchura viene determinada por el primer y el último valor de la bandeja. Por lo tanto, la altura de cada *bin* se define por la Ecuación 6.2:

$$\frac{\frac{N}{k}}{last - first} \tag{6.2}$$

A pesar de ser un método muy eficiente con una baja complejidad computacional, la naturaleza iterativa de *HBOS* lo hace inadecuado para abordar problemas en entornos *Big Data*. *HBOS_BD* es una implementación distribuida que utiliza primitivas de *Spark*, como el cálculo distribuido de histogramas, lo que permite aplicar *HBOS* a grandes conjuntos de datos con rapidez. En *HBOS_BD* se han implementado las dos variantes del método *HBOS* original (estática y dinámica), siguiendo el mismo comportamiento que las originales.

El Algoritmo 1 describe el proceso de detección de anomalías estáticas *HBOS_BD*:

- El primer paso es el cálculo del histograma de los datos. Se itera por cada característica unidimensional y se dibuja sobre ella un histograma con intervalos *n_bins*, obteniendo los límites y el número de elementos. Por último, se normaliza el histograma, dividiendo cada valor por la suma total de sus valores (líneas 5-10).

- El siguiente paso es el cálculo de los *scores* de los datos mediante una función *map* distribuida (líneas 12-19). En primer lugar, es necesario averiguar en qué ubicación se encuentra la característica de la instancia seleccionada. La ubicación se calcula mediante la siguiente ecuación: $bin \leftarrow \frac{(feature-min)\cdot n\_bins}{max-min}$ donde $feature$ es el valor de la instancia para esa característica, max es el valor máximo del *bin*, min es el valor mínimo del *bin*, y n_bins el número de *bins* (línea 15). Una vez encontrado el *bin*, el *score* de una sola instancia se calcula utilizando la Ecuación 6.1 (línea 16), realizando la suma al final del proceso (línea 19).

- Finalmente, se devuelve el conjunto distribuido resultante que contiene el *scores* de anomalía de cada instancia (línea 20).

Algoritmo 1 Algoritmo *HBOS_BD* Estático

1: **Input:** *data* el conjunto de datos con el formato Dataset[“features”]
2: **Input:** *n_bins* El número de *bins* para el histograma
3: **Output:** Un RDD[Double] con los *scores* de cada instancia del conjunto de datos
4: $limits, histograms \leftarrow \emptyset$
5: **for** $i \leftarrow 0$ until $n\_attributes$ **do**
6: $\quad hist \leftarrow histogram(select(data, i), n\_bins)$
7: $\quad append(limits(i), getLimits(hist))$
8: $\quad hist\_sum \leftarrow sum(getHistogram(hist))$
9: $\quad append(histograms(i), getHistogram(hist).map(l => l/hist\_sum))$
10: **end for**
11: $scores \leftarrow$
12: **map** $instance \in data$
13: $\quad values \leftarrow \emptyset$
14: $\quad$ **for** $i \leftarrow 0$ until $n\_attributes$ **do**
15: $\quad\quad index \leftarrow computeIndex(instance(i), limits(i))$
16: $\quad\quad values(i) \leftarrow computeScores(histogram(i), index)$
17: $\quad$ **end for**
18: $\quad sum(values)/n\_bins$
19: **end map**
20: $return(scores)$

El Algoritmo 2 describe el proceso de detección dinámica de anomalías *HBOS_BD*:

- Para calcular el histograma, primero hay que ordenar los datos. Las instancias de interés son los que están en los límites de cada *bin*. La longitud de cada *bin* se define como: $\frac{N}{n\_bins}$ valores, por lo que el índice de cada característica se puede calcular previamente (líneas 8-9). Una vez que conocemos los límites de cada *bin*, la altura se calcula utilizando la Ecuación 6.2 (líneas 10-12).

- El siguiente paso es el cálculo de los *scores* de los datos utilizando una función *map* distribuido (líneas 15-22). En primer lugar, es necesario averiguar en qué ubicación se encuentra la característica de la instancia seleccionada. La ubicación se calcula comparando el valor de la característica entre el primer y el último valor de cada casilla, para determinar si el valor se encuentra entre los dos (línea 18). Una vez encontrada la ubicación, se calcula el *scores* de una única instancia mediante la Ecuación 6.1 (línea 19), realizando la suma al final del proceso (línea 21).

- Finalmente, se devuelve el conjunto distribuido resultante que contiene el *score* de anomalía de cada instancia (línea 23).

Algoritmo 2 Algoritmo *HBOS_BD* Dinámico

```
1: Input: data el conjunto de datos con el formato Dataset["features"]
2: Input: n_bins El número de bins para el histograma
3: Output: Un RDD[Double] con los scores de cada instancia del conjunto de datos
4: limits, histograms ← ∅
5: inc ← count(data) / n_bins
6: for i ← 0 until n_attributes do
7:     sortedValues ← sort(select(data, i))
8:     limits(i)._1 ← select(sortedValues(0 until count(data) − inc by inc))
9:     limits(i)._2 ← select(sortedValues(inc until count(data) by inc))
10:    hist ← computeHistogram(selected(data, i), limits(i))
11:    hist_sum ← sum(getHistogram(hist))
12:    append(histograms(i), getHistogram(hist).map(l => l/hist_sum))
13: end for
14: scores ←
15: map instance ∈ data
16:    values ← ∅
17:    for i ← 0 until n_attributes do
18:        index ← computeIndexDynamic(instance(i), limits(i))
19:        values(i) ← computeScores(histogram(i), index)
20:    end for
21:    sum(values)/k
22: end map
23: return(scores)
```

Como parámetros de entrada para ambos métodos se requieren: el conjunto de datos (*data*), y el número de *bins* para los histogramas (*n_bins*).

6.1.2 *Lightweight On-line Detector of Anomalies For Big Data: LODA_BD*

Un conjunto de clasificadores débiles es capaz de producir un clasificador robusto. Esta es la premisa de Lightweight On-line Detector of Anomalies (*LODA*) [Pev16], en la que una colección de detectores muy débiles puede dar lugar a un detector de anomalías robusto. *LODA* se basa en el cálculo de una colección de histogramas unidimensionales, cada uno de ellos extraído de la proyección de los datos originales sobre un vector generado aleatoriamente. El *score* de *LODA*, $f(x)$, para una muestra dada, x, es la media del logaritmo de las probabilidades de cada vector de proyección individual. Siendo p_i la probabilidad estimada por el *ith* histograma, w_i el *ith* vector

de proyección, y k el número de proyecciones aleatorias, el *score* de $LODA$ se calcula como se muestra en la Ecuación 6.3:

$$f(x) = -\frac{1}{k}\sum_{i=1}^{k} \log p_i(x^T w_i) \tag{6.3}$$

A pesar de ser un detector de anomalías rápido y con una baja complejidad temporal y espacial, $LODA$ no está adaptado al paradigma *Big Data* ni tiene un enfoque distribuido, ya que presenta una naturaleza iterativa. La proyección de los datos sobre un vector unidimensional y el cálculo del histograma se realiza de forma iterativa. Este proceso suele realizarse varios cientos de veces, lo que, en un entorno *Big Data*, dará lugar a la programación y ejecución de varios cientos de tareas.

$LODA\_BD$ resuelve los problemas anteriores transformando las proyecciones iterativas de los datos, en una operación matricial distribuida. Se crea una matriz que contiene todas las proyecciones deseadas y, mediante una multiplicación matricial distribuida con los datos originales, se calculan las proyecciones de una sola vez. A continuación, $LODA\_BD$ calcula los diferentes histogramas sobre los datos proyectados y calcula el *score* de anomalía de cada instancia.

El Algoritmo 3 describe el proceso de detección de anomalías $LODA\_BD$:

- En primer lugar, los datos originales se transforman del formato *Dataset* de *Spark* a un formato de matriz distribuida (*RowMatrix*) (línea 5).
- En la línea 6 se calcula la matriz de proyección aleatoria. Esta matriz estará formada por valores muestreados a partir de una distribución gaussiana de media 0 y varianza 1. Se compone de $\sqrt{d}$ componentes no nulos seleccionados al azar (siendo d el número de características). El resultado es la matriz local $projections^{d \times k}$.
- Una vez que los datos y los vectores de proyección están en formato matricial, se realiza una multiplicación matricial distribuida (línea 7). Este proceso multiplica cada partición de los datos distribuidos (en formato matricial) por la matriz de proyección local.
- El siguiente paso es el cálculo del histograma de los datos proyectados. Consiste en iterar por cada característica unidimensional y dibujar sobre ella un histograma con intervalos *n_bins*, obteniendo los límites y el número de elementos. Por

último, se normaliza el histograma, dividiendo cada valor por la suma total de sus valores (líneas 9-14).

- En el último paso, se calcula el *score* de anomalía de cada instancia mediante una función *map* distribuida (líneas 15-22). Para cada elemento, se itera a través de las k proyecciones, y se calcula el $\log p_i(datos\ proyectados)$ (línea 19). Por último, todos los *scores* de cada instancia se normalizan (línea 21), y el conjunto distribuido resultante que contiene el *score* de anomalía de cada instancia se devuelve.

Como parámetros de entrada se requieren: el conjunto de datos (*data*), el número de *bins* para los histogramas (*n_bins*) y el número de proyecciones (k).

Algoritmo 3 Algoritmo *LODA_BD*

```
1:  Input: data el conjunto de datos con el formato Dataset["features"]
2:  Input: n_bins El número de bins para el histograma
3:  Input: k El número de proyecciones
4:  Output: Un RDD[Double] con los scores de cada instancia del conjunto de datos
5:  dataAsMatrix ← RowMatrix(data)
6:  projections ← createRandomProjections(size(data), k)
7:  projectedData ← multiply(dataAsMatrix, projections)
8:  limits, histograms ← ∅
9:  for i ← 0 until k do
10:     hist ← histogram(select(projectedData, i), n_bins)
11:     append(limits(i), getLimits(hist))
12:     hist_sum ← sum(getHistogram(hist))
13:     append(histograms(i), getHistogram(hist).map(l => l/hist_sum))
14: end for
15: scores ←
16: map instance ∈ projectedData
17:     values ← ∅
18:     for i ← 0 until k do
19:         values(i) ← calculateScores(limits(i), histograms(i))
20:     end for
21:     sum(values)/k
22: end map
23: return(scores)
```

6.1.3 *Locally Selective Combination in Parallel Outlier Ensembles For Big Data: LSCP_BD*

Locally Selective Combination in Parallel Outlier Ensembles (*LSCP*) [ZNH+19] es un método de *ensembles* que aborda el problema de no tener un *ground truth* (información correcta o verdadera sobre las salidas o etiquetas asociadas a un conjunto de datos) en la detección no supervisada de anomalías. Define una pseudo *ground truth* base utilizando un número de detectores base y seleccionando la media o el máximo de ellos. A continuación, define una región local alrededor de cada instancia y utiliza los vecinos más cercanos en sub-espacios de características seleccionados aleatoriamente. Se seleccionan los detectores base con mejor rendimiento en dichas regiones locales y se combinan como dando el resultado del *ensemble*.

Aunque los *ensembles* han demostrado ser capaces de lograr un gran rendimiento y adaptarse a entornos *Big Data* de manera efectiva [GRG+18], los métodos con alta complejidad computacional presentan un problema en tales dominios. *LSCP* define la región local de cada instancia extrayendo los *k-Nearest Neighbors* de una serie de grupos de características seleccionadas aleatoriamente, y luego tomando los ejemplos más seleccionados por $k - NN$. Este proceso implica el cálculo de varios métodos $k - NN$. En dominios *Big Data*, esto implicará el cálculo de miles de millones de distancias, lo que puede ser prohibitivo en términos de tiempo de computación.

LSCP_BD alivia esta alta complejidad computacional de la definición de la región local calculando una aproximación de dichos *k-Nearest Neighbors*. *LSCP_BD* aprovecha el concepto de datos particionados y distribuidos para crear particiones en las que todas las instancias tendrán un alto grado de similitud. Se sustituye el uso de $k - NN$ para *clustering* por un método más rápido para la definición de vecindarios. Concretamente, se emplea la implementación distribuida de *Spark* de $k - Means$ [Mac67], y *Bisecting k-Means* [SKKoo] para medir la similitud entre instancias. Esto puede dar lugar a grandes conglomerados de datos normales y pequeños conglomerados de *outliers* o anomalías. Por último se procesan estos conglomerados individualmente, dividiendo los grandes y uniendo los pequeños. El resultado es un conjunto de datos en el que cada partición contiene la región local de esas instancias.

El Algoritmo 4 describe el método de detección de anomalías *LSCP_BD*:

- *LSCP_BD* comienza aprendiendo un número de detectores base y uniendo sus *scores* de anomalía usando la función distribuida *join* de *Spark* (líneas 8-11). Estos detectores base pueden ser cualquier método distribuido de detección de anomalías.

- A continuación, el pseudo *ground truth* para cada ejemplo se establece utilizando una función distribuida *map* sobre los *scores* de los detectores base unidos. El *score* máximo (o medio) de anomalía de cada detector para cada instancia dada será seleccionada como pseudo *ground truth*, de acuerdo con *pgt_strategy* (líneas 12-16).

- Una vez establecido el pseudo *ground truth*, la función *clusterPartitioning* propuesta se aplica a los datos con el método de *clustering* seleccionado (*clus_method*) y el número de *clusters* (*n_clus*). Este proceso se describe en profundidad en el Algoritmo 5. El resultado es un conjunto de datos en el que cada partición contiene una región local (línea 17).

- A continuación, el proceso de selección y combinación de modelos se aplica a cada partición utilizando una función *mapPartitions* distribuida (líneas 18-34). En primer lugar, se calcula la correlación de cada detector base con el pseudo *ground truth* mediante la correlación de *pearson* (líneas 20-23). A continuación, *LSCP_BD* elegirá la estrategia de selección y combinación en función del porcentaje de detectores base seleccionados para la selección dinámica de conjuntos de valores anómalos (*dcs*). Un *dcs* de 1 implicará utilizar todos los detectores base y seleccionar el más correlacionado como decisión del conjunto (líneas 24-25). Si $dcs < 1$, se seleccionarán los *dcs* detectores base más correlacionados y, atendiendo a *pgt_strategy*, se seleccionará la media o el máximo de ellos (Media del Máximo o Máximo de la Media) (líneas 26-34).

- Finalmente, se devuelve el conjunto distribuido con la decisión del *ensemble* para cada instancia (línea 35).

Como parámetros de entrada se requieren los siguientes: el conjunto de datos (*data*), el número de detectores de base (*n_base_detectors*), la estrategia para la generación del pseudo *ground truth* (*pgt_strategy*), el método de agrupación para el cálculo de la región local (*clus_method*), el número de conglomerados para el cálculo de la región local (*n_clus*), y el porcentaje de detectores base seleccionados para la selección dinámica de valores anómalos del *ensemble* (*dcs*).

Algoritmo 4 Algoritmo *LSCP_BD*

Input: *data* el conjunto de datos con el formato Dataset["features"]
Input: *n_base_detectors* El número de detectores base
Input: *pgt_strategy* La estrategia para la generación del pseudo *ground truth* ("avg", "max")
Input: *clus_method* Método de *clustering* para el cálculo de la región local ("kmeans", "bisec")
Input: *n_clus* Número de *clusters* para el cálculo de la región local
Input: *dcs* Porcentaje de detectores base seleccionados para la selección dinámica de valores anómalos del *ensemble*
Output: Un RDD[Double] con los *scores* de cada instancia del conjunto de datos
$detectors \leftarrow data$
for $i \leftarrow 0$ until $n\_base\_detectors$ **do**
    $detectors \leftarrow join(detectors, learnBaseDetector(data))$
end for
$pseudo\_gt \leftarrow$
map $score \in detectors$
    if $pgt\_strategy = "avg"$ **then** $append(score, avg(score))$ **end if**
    if $pgt\_strategy = "max"$ **then** $append(score, max(score))$ **end if**
end map
$gt\_clustered \leftarrow clusterPartitioning(pseudo\_gt, clus\_method, n\_clus)$
$scores \leftarrow$
mapPartitions $partition \in gt\_clustered$
    $correlations \leftarrow \emptyset$
    for $i \leftarrow 0$ until $n\_base\_detectors$ **do**
        $correlations(i) \leftarrow pearson(partition(i), partition(size) - 1)$
    end for
    if $dcs = 1$ **then**
        $max(partition(correlations))$
    else
        $mostCorrelated \leftarrow top(correlations, dcs)$
        if $pgt\_strategy = "avg"$ **then**
            $max(partition(mostCorrelated))$
        else
            $avg(partition(mostCorrelated))$
        end if
    end if
end mapPartitions
$return(scores)$

El Algoritmo 5 describe el proceso de partición de los datos por similitud mediante un método de *clustering*.

- Esta función comienza aplicando un método de *clustering* distribuido de la *MLlib* de *Spark* ($k-Means$ o *Bisecting k-Means*), con n_clus *clusters* diferentes. Los datos predichos tendrán un índice de *cluster* asignado a cada instancia (líneas 8-9).
- A continuación, los datos distribuidos se repartirán de acuerdo con los *clusters* asignados utilizando la función *repartitionByRange* de *Spark* (línea 10).
- El siguiente paso es comprobar cada partición utilizando una función distribuida *mapPartition*, para evaluar si el número de instancias es óptimo ($size < max\_size \wedge size > min\_size$) (líneas 12-21). Si el tamaño de la partición es mayor que el umbral ($tamao_m x$), se seleccionarán trozos de tamaño $tamao_m x$ y se asignarán a un $sub-cluster$ (por ejemplo, el *cluster* 1 se dividirá en los $sub-clusters$ 1.1, 1.2, ...). Por otro lado, las particiones más pequeñas que el umbral min_size se unirán en un nuevo *cluster*.
- Finalmente, los datos resultantes con los nuevos índices de *cluster* se reparticionan utilizando la función *repartitionByRange* (línea 22).

Como parámetros de entrada se requieren: el conjunto de datos (*datos*), el método de *clustering* para el cálculo de la región local (*método_clus*), el número de *clusters* para el cálculo de la región local (n_clus), el número máximo de elementos por partición *tamaño_max*, y el número mínimo de elementos por partición *tamaño_min*.

Algoritmo 5 Función *clusterPartitioning*

Input: *data* el conjunto de datos con el formato Dataset[“features”]
Input: *clus_method* Método de *clustering* para el cálculo de la región local (“kmeans”, “bisec”)
Input: *n_clus* Número de *clusters* para el cálculo de la región local
Input: *max_size* Máximo de elementos por partición (por defecto = 10,000)
Input: *min_size* Mínimo de elementos por partición (por defecto = 1,000)
Output: Un RDD[Double] con los *scores* de cada instancia del conjunto de datos
$clustering \leftarrow \emptyset$
if $clus\_method = \text{“}kmeans\text{”}$ **then** $clustering \leftarrow KMeans(data, n\_clus)$ **end if**
if $clus\_method = \text{“}bisec\text{”}$ **then** $clustering \leftarrow BisectingKMeans(data, n\_clus)$ **end if**
$repartitionedData \leftarrow repartitionByRange(clustering, \text{“}cluster\text{”})$
$balancedData \leftarrow$
mapPartitions $partition \in repartitionedData$
    $clusterSize \leftarrow size(partition)$
    if $clusterSize > max\_size$ **then**
        for $i \leftarrow 0$ until $clusterSize$ by max_size **do**
            $partition(i).cluster \leftarrow partition(i).cluster + 0.1$
        end for
    else if $clusterSize < min\_size$ **then**
        $partition(i).cluster \leftarrow \infty$
    end if
end mapPartitions
$return(repartitionByRange(balancedData, \text{“}cluster\text{”}))$

6.1.4 *Extreme Gradient Boosting Outlier Detection for Big Data: XGBOD_BD*

Extreme Gradient Boosting Outlier Detection (*XGBOD*) [ZH18] es un método conjunto semi-supervisado propuesto recientemente para la detección de anomalías. Este método combina el potencial de los enfoques supervisados y no supervisados para la detección de anomalías. *XGBOD* utiliza un conjunto de algoritmos no supervisados de detección de anomalías para extraer valiosas representaciones de la estructura de los datos que aumentan las capacidades predictivas de un clasificador supervisado. El conjunto original de características se amplía con un conjunto de *scores* de valores anómalos transformados (*TOS*) seleccionados y, a continuación, se aprende un clasificador *XGBoost* sobre los datos ampliados.

A pesar de ser un método de *ensemble* eficiente, *XGBOD* hace uso de varios métodos base no supervisados de detección de anomalías para seleccionar las diferentes *TOS*. Como ya hemos comentado, en entornos con grandes cantidades de datos, los métodos no supervisados clásicos no son capaces de alcanzar el rendimiento deseado. Además, *XGBOD* aplica un clasificador iterativo (*XGBoost*) a los datos ampliados. Esto provoca que *XGBOD* sea incapaz de manejar conjuntos de datos *Big Data*.

XGBOD_BD aborda estos problemas empleando un método distribuido de detección de anomalías no supervisado para el aprendizaje de *TOS*. También hace uso de la implementación distribuida de *Spark* de *XGBoost*. *XGBOD_BD* implementa dos estrategias diferentes para la selección de *TOS*: aleatoria y basada en la precisión. La estrategia aleatoria selecciona aleatoriamente y sin reemplazo *n_selected_detectors* detectores. La estrategia basada en la precisión selecciona los mejores *n_selected_detectors* detectores basándose en la medida de precisión. Para establecer la precisión de las distintas *TOS*, se aplica un umbral. Con este umbral, se etiquetan los *TOS* y se calcula la precisión. Las *TOS* seleccionadas se añaden al conjunto original de características y se aprende un modelo *XGBoost* distribuido.

El Algoritmo 6 describe el método de detección de anomalías *XGBOD_BD*:

- *XGBOD_BD* comienza aprendiendo *n_TOS* y uniendo los *scores* de anomalía usando la función distribuida *join* de *Spark* (líneas 7-10). Estos detectores de anomalías no supervisados de base pueden ser cualquier método de detección de anomalías distribuido.
- A continuación, se aplica el proceso de selección de *TOS* (líneas 11-26). Si la estrategia seleccionada es *random*, *XGBOD_BD* elegirá *n_selected_TOS* índices aleatoriamente y sin reemplazo (líneas 12-13). Por otro lado, si la estrategia seleccionada es *precisin*, cada *TOS* se etiqueta utilizando una función *map* distribuida y el *umbral* proporcionado (líneas 15-20). A continuación, se calcula la precisión de cada *TOS* etiquetada utilizando la métrica distribuida *MulticlassMetrics* de *Spark*, comparando la etiqueta original con la *TOS* etiquetada (líneas 21-24). Se seleccionan los índices *n_selected_TOS TOS* con mayor precisión (línea 25).
- Una vez seleccionados los *n_selected_TOS* índices *TOS*, se combinan las características originales y las *TOS* seleccionadas (líneas 27-30).
- Por último, un modelo distribuido *XGBoost* se aprende en los datos combinados, y sus predicciones se devuelven como la decisión del método.

Se requieren los siguientes parámetros de entrada: el conjunto de datos (*data*), el número de *TOS* (*n_TOS*), el número de *TOS* seleccionados (*n_selected_TOS*), la estrategia para la selección de *TOS* (*TOS_strategy*), y el umbral para la estrategia de precisión (*threshold*).

Algoritmo 6 Algoritmo *XGBOD_BD*

Input: *data* el conjunto de datos con el formato Dataset[“features”, “label”]
Input: *n_TOS* el número de *TOS*
Input: *n_selected_TOS* el número de *TOS* seleccionado
Input: *TOS_strategy* La estrategia para la selección de *TOS* (“rnd”, “acc”)
Input: *threshold* El umbral para la precisión de la estrategia
Output: Un RDD[Double] con la etiqueta de cada instancia del conjunto de datos
$TOS \leftarrow data$
for $i \leftarrow 0$ until n_TOS **do**
    $TOS \leftarrow join(TOS, learnBaseDetector(data))$
end for
$selected\_TOS \leftarrow \emptyset$
if $TOS\_strategy = \text{“}rnd\text{”}$ **then**
    $selected\_TOS \leftarrow randomList(0, n\_TOS, n\_selected\_TOS)$
else
    $labeled\_TOS \leftarrow$
    map $(instance, scores) \in TOS$
        for $i \leftarrow 0$ until n_TOS **do**
            if $scores(i) >= threshold$ **then** $(instance, 1)$ **else** $(instance, 0)$
        end for
    end map
    $accuracy \leftarrow \emptyset$
    for $i \leftarrow 0$ until n_TOS **do**
        $accuracy(i) \leftarrow MulticlassMetrics(labeled\_TOS, \text{“}accuracy\text{”})$
    end for
    $selected\_TOS \leftarrow indices(sort(accuracy))$
end if
$features\_combined \leftarrow$
map $(instance, scores) \in TOS$
    $select(selected\_TOS, scores)$
end map
$predictions \leftarrow xgbClassifier(features\_combined)$
$return(predictions)$

6.2 COMPARATIVA EXPERIMENTAL DE LAS PROPUESTAS

En esta sección se describen los detalles experimentales y el análisis realizado para evaluar el rendimiento de los detectores de anomalías distribuidos propuestos sobre un caso real de estudio de *Big Data*. Se detalla el conjunto de datos del caso real de estudio, así como el marco experimental. También se analiza el rendimiento de los algoritmos propuestos. Finalmente, se mostrarán los tiempos de computación y la escalabilidad de las propuestas.

6.2.1 *Caso de estudio Big Data utilizado: Descripción de los datos de ArcelorMittal*

Los datos de una de las máquinas de ArcelorMittal han sido proporcionados directamente por la empresa. Debido al entorno hostil en el que trabajan estos activos, sufren averías con relativa frecuencia. Algunas de estas averías pueden arreglarse "in situ", pero otras mantienen la maquinaria parada durante varios días. Estas averías suponen un grave retraso en su cadena de producción.

Los datos proporcionados están compuestos por la información de los sensores de una de sus máquinas de producción, así como información relacionada, como averías, información contextual, etc. Cada uno de estos sensores corresponde a una de las características que componen el conjunto de datos. Estas variables, la mayoría de carácter real, modelan distintas propiedades de la maquinaria.

Estos datos abarcan un periodo de dos años, con casi 40 millones de observaciones, cada una de ellas con más de 100 variables. El objetivo es detectar estos fallos con la suficiente antelación para minimizar los periodos de reparación de estas máquinas.

6.2.2 *Configuración experimental*

El conjunto de datos se ha preprocesado, escalándolo al rango cero-uno. Se han eliminado seis características del total de 112 porque tienen un valor constante.

Los métodos *ensemble* (*LSCP_BD* y *XGBOD_BD*) requieren un detector base diverso. Para nuestros experimentos, hemos utilizado *LODA_BD* como detector base ya que tiene más diversidad que *HBOS_BD* debido al uso de proyecciones aleatorias. Dado que *XGBOD_BD* es un método semi-supervisado, los datos de entrenamiento necesitan ser etiquetados. Para esta tarea, seleccionamos un intervalo de [1, 2, 4, 6]

horas antes de un fallo y lo etiquetamos como clase 1 (anomalía). El resto se etiqueta como clase 0 (normal).

El rendimiento se evalúa mediante la métrica $ROC - AUC$. Esta métrica ya comentada en el Capítulo 2 ha sido ampliamente empleada en la investigación de anomalías [Agg16; CBK09; CZS+16; GU16a]. Dado que nos encontramos en un escenario no supervisado, los algoritmos se han evaluado utilizando un *ground truth* desconocido para ellos. El criterio seguido para evaluar el rendimiento de los métodos en términos de $ROC - AUC$ ha sido detectar anomalías en un periodo fijo de 1 a 48 horas antes de un fallo.

Algoritmo	Parametros
HBOS_BD	n_bins = 100, strategy = "static"
LODA_BD	n_bins = 100, k = 100
LSCP_BD	detector = "LODA_BD", n_base_detectors = 10, pgt_strategy = "avg", clus_method = "kmeans", n_clus = 19, max_size = 10,000, min_size = 1,000, dcs = 0.5
XGBOD_BD	detector = "LODA_BD", n_TOS = 10, n_selected_TOS = 5, TOS_strategy = "acc", threshold = 0.1

Tabla 6.1: Parámetros por defecto para los detectores de anomalías

En la Tabla 6.1 podemos observar la lista completa de parámetros por defecto para los algoritmos distribuidos de detección de anomalías propuestos. Los problemas de detección de anomalías dependen mucho de los datos. Por este motivo, un conjunto de parámetros fijos difícilmente puede extrapolarse a otro problema. Para evaluar el rendimiento de cada uno de los métodos propuestos, es necesario encontrar la combinación adecuada de parámetros. Para esta tarea se ha realizado una optimización de hiper-parámetros utilizando *Optuna*, ya comentada en la Sección 5.2.2 del capitulo anterior.

Los experimentos se han llevado a cabo en un *cluster* compuesto por 15 nodos de computación y un nodo maestro. Los nodos de computación tienen las siguientes especificaciones de hardware: 2 x Intel Xeon CPU E5-645, 6 núcleos por procesador (12 hilos), 2,40 GHz, 4 TB HDD, 64 GB RAM, e Infiniband 40 Gb/s. En cuanto al software, hemos utilizado la siguiente configuración: Apache Hadoop 2.6.0, Apache

Spark 3.0.1 con 360 núcleos (24 núcleos/nodo) y 780 GB de RAM (52 GB/nodo) disponibles.

6.2.3 *Resultados y Análisis*

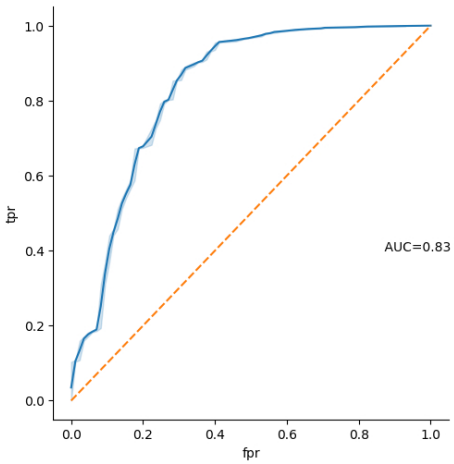
En esta sección, se muestran los resultados de los algoritmos sobre el caso de estudio real. Estos resultados se han evaluado utilizando un *ground truth* desconocido para los algoritmos. Se han seleccionado varios tamaños de ventana fijos, que van desde 1 hora hasta 48 horas antes de un fallo. Dado que es un escenario no supervisado, cabe esperar un valor $ROC - AUC$ bajo.

Horas previas	HBOS_BD		LODA_BD	LSCP_BD	XGBOD_BD
	Estático	Dinámico			
1	0,6322	**0,6483**	0,6206	0,6367	0,4981
2	0,6352	**0,6571**	0,6194	0,6370	0,5166
3	0,6366	**0,6614**	0,6214	0,6382	0,4931
4	0,6411	**0,6635**	0,6267	0,6413	0,5040
5	0,6438	**0,6636**	0,6268	0,6406	0,5054
6	0,6482	**0,6707**	0,6287	0,6428	0,4947
12	0,6772	**0,6914**	0,6525	0,6672	0,4712
18	0,7010	**0,7159**	0,6607	0,6813	0,4860
24	0,7314	**0,7442**	0,7001	0,7224	0,4940
36	0,7772	**0,7869**	0,7275	0,7522	0,4481
48	0,8325	**0,8547**	0,8038	0,8291	0,4440

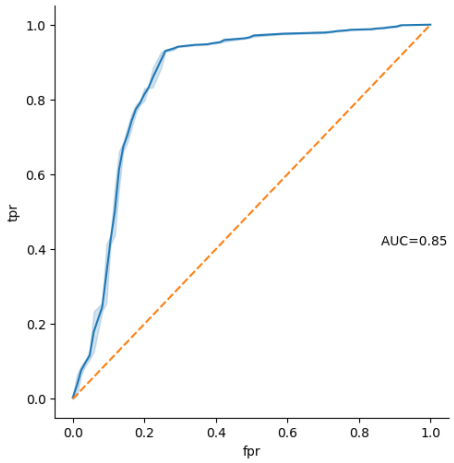
Tabla 6.2: Valor $ROC - AUC$ para cada algoritmo y número de horas previas a un fallo. El valor $ROC - AUC$ más alto por hora se destaca en negrita.

En la Tabla 6.2 se muestran los resultados en términos del valor $ROC - AUC$ para los cuatro métodos propuestos. Como se puede observar, excepto $XGBOD\_BD$, todos los métodos obtienen mejores resultados que una predicción aleatoria, y sus resultados son cada vez mejores a medida que aumenta el número de horas previas a un fallo. Como cabe esperar, el escenario más restrictivo es detectar un fallo

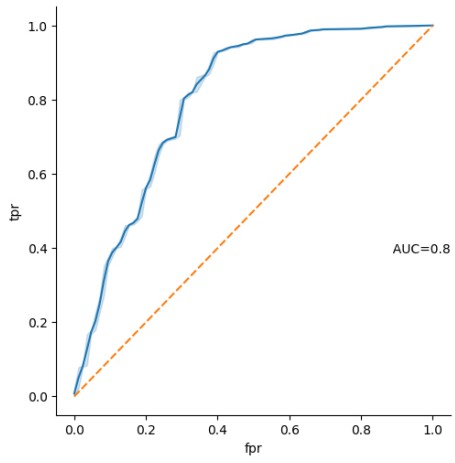
justo una hora antes de que se produzca. Dado que los fallos pueden deberse a un desgaste continuo a lo largo del tiempo, aumentar el número de horas previas a un fallo también mejora el rendimiento de los métodos. La versión dinámica de *HBOS_BD* obtiene los mejores resultados, seguida de su enfoque estático y de *LSCP_BD*. *LSCP_BD* muestra el comportamiento perfecto de un método *ensemble*, siendo capaz de mejorar eficazmente el rendimiento del detector base (*LODA_BD*). *XGBOD_BD* no es capaz de detectar eficazmente las anomalías en este problema en particular. Esto puede explicarse porque la estrategia de etiquetado añade ruido al conjunto de datos. Es decir, *XGBOD_BD* hace uso de algoritmos de detección de anomalías no supervisados para extraer representaciones de la estructura de los datos, dicha representación se añade a las características originales para dar más información al clasificador final. Por tanto, esas representaciones pueden no ser correctas y por tanto, se estaría añadiendo ruido a los datos, dando lugar a que $XGBOD_DB$ obtenga un peor rendimiento que los demás algoritmos. En la Figura 6.1 mostramos los gráficos del valor $ROC - AUC$ para el mejor resultado de cada método.



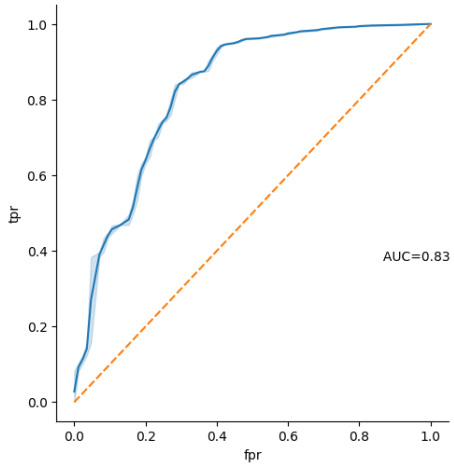
(a) HBOS_BD (static)



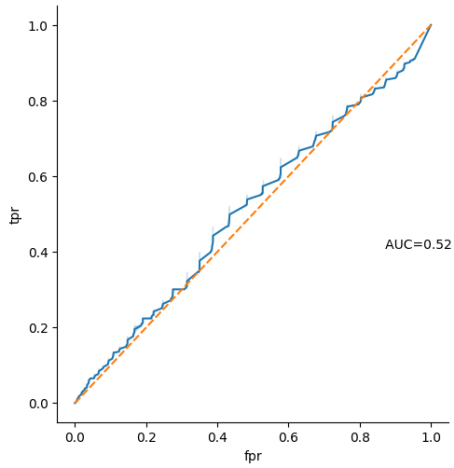
(b) HBOS_BD (dynamic)



(c) LODA_BD



(d) LSCP_BD



(e) XGBOD_BD

Figura 6.1: Gráfico de valores $ROC - AUC$ para *HBOS_BD*, *LODA_BD*, *LSCP_BD* y *XGBOD_BD* con los mejores resultados obtenidos.

Para verificar el rendimiento de la propuesta, se ha comparado *HBOS_BD*, *LODA_BD*, *LSCP_BD*, y *XGBOD_BD* con las versiones originales de los algoritmos *HBOS*, *LODA*, *LSCP* y *XGBOD*. Para esta comparación, se han utilizado 3 *benchmarks* públicos de detección de anomalías de tamaño normal de *ODDS*[1]. La Tabla 6.3 recoge dichos resultados en términos de la métrica $ROC - AUC$. A partir de estos resultados se puede concluir que los algoritmos de detección de anomalías propuestos logran resultados similares o mejores que los métodos originales.

Conjunto de datos			Breast	Shuttle	Satellite
Clásico	HBOS		0,9910	0,9855	0,7581
	LODA		0,9866	0,9920	0,6114
	LSCP		0,7845	0,5551	0,6106
	XGBOD		0,9916	**1,0**	**0,9685**
Big Data	HBOS_BD	Estático	0,9853	0,9922	0,8040
		Dinámico	0,9640	0,5807	0,7043
	LODA_BD		0,9879	0,9906	0,7291
	LSCP_BD		0,9831	0,9891	0,7420
	XGBOD_BD		**1,0**	**1,0**	0,9394

Tabla 6.3: Valor $ROC - AUC$ para *HBOS* clásico, *LODA*, *LSCP* y *XGBOD* frente a cada algoritmo *Big Data* propuesto. El valor ROC-AUC más alto se resalta en negrita.

6.2.4 *Análisis de Eficiencia*

En la sección anterior se ha mostrado la idoneidad de *HBOS_BD*, *LODA_BD*, *LSCP_BD*, y *XGBOD_BD* en términos de valor $ROC - AUC$. Para constituir una propuesta válida en entornos *Big Data*, estas propuestas tienen que ser también eficientes y escalables. En esta sección, se presentan los tiempos de computación y la escalabilidad en términos de tamaño de los datos, número de hilos y *workers* para los diferentes métodos que componen la propuesta utilizando el conjunto de datos

1 ODDS Library: http://odds.cs.stonybrook.edu/

Big Data de *ArcelorMittal*. Para esta comparación se han seleccionado parámetros por defecto.

En la Tabla 6.4 se muestran los tiempos de ejecución para los cuatro métodos propuestos. Con el fin de mostrar la escalabilidad de los algoritmos propuestos, también mostramos los tiempos de computación utilizando diferentes muestras del conjunto de datos, que van desde un 10 % a un 100 % del tamaño del conjunto de datos original. Como se puede observar, todos los métodos muestran un incremento lineal del tiempo con el tamaño de los datos. *LODA_BD* es el método con mejor rendimiento, capaz de procesar un conjunto de datos *Big Data* en menos de 93 segundos, seguido de la versión estática de *HBOS_BD*. Se espera que los métodos *ensemble*, así como la versión dinámica de *HBOS_BD*, tengan un rendimiento más lento, ya que son métodos más costosos computacionalmente.

Muestras(%)	HBOS_BD		LODA_BD	LSCP_BD	XGBOD_BD
	Estático	Dinámico			
10	76,54	1.358,49	47,42	508,82	2.033,77
20	81,65	1.773,09	52,98	613,75	2.098,80
30	97,53	2.290,66	63,45	752,22	2.191,52
40	111,13	2.661,60	68,25	962,91	2.226,92
50	120,37	2.854,53	74,12	1.086,29	2.353,04
60	144,39	3.133,64	72,32	1.151,85	2.454,87
70	161,58	3.457,43	76,68	1.269,82	2.598,17
80	175,49	3.712,69	85,12	1.340,68	2.680,75
90	187,66	4.074,33	87,17	1.455,42	2.787,53
Original	198,30	4.464,62	92,94	1.667,13	2.860,13

Tabla 6.4: Tiempos de cálculo con diferentes tamaños de datos para *HBOS_BD*, *LODA_BD*, *LSCP_BD*, y *XGBOD_BD* en segundos.

Para medir la escalabilidad de los algoritmos propuestos, se han ejecutado los diferentes métodos utilizando un número creciente de hilos. En la Tabla 6.5 se muestran los tiempos de ejecución de *HBOS_BD*, *LODA_BD*, *LSCP_BD* y *XGBOD_BD* utilizando 16, 32, 64, 128 y 256 hilos del total de 360 hilos disponibles. Como puede

apreciarse, de 16 a 64 hilos los tiempos de ejecución mejoran casi linealmente, siendo 2 veces más rápidos cada incremento de hilos. A partir de 64 hilos, la ganancia de tiempo empieza a disminuir. Esto puede explicarse por la diferencia en términos de núcleos físicos e hilos disponibles (2 x núcleos).

Hilos	HBOS_BD		LODA_BD	LSCP_BD	XGBOD_BD
	Estático	Dinámico			
16	836,96	23.896,53	679,13	8.212,83	14.195,39
32	474,50	12.694,02	337,39	4.249,22	7.072,06
64	317,72	7.614.48	201,01	2.397,62	3.883,32
128	222,52	5.073,10	123,03	1.601,19	2.779,84
256	198,76	4.451,82	105,80	1.332,67	2.663,07

Tabla 6.5: Tiempos de computación con diferente número de hilos para *HBOS_BD*, *LODA_BD*, *LSCP_BD*, y *XGBOD_BD* en segundos.

Además del número de hilos, se ha medido la escalabilidad en términos de *workers* disponibles. En la Tabla 6.6 se recogen los resultados de los algoritmos propuestos utilizando un número creciente de *workers*. De forma similar al número de hilos, el rendimiento aumenta a un ritmo lineal hasta 8 *workers*. A partir de 10 *workers*, la ganancia de tiempo empieza a ser menos notable. Añadir más *workers* no sólo aporta beneficios en términos de potencia de cálculo, sino que también tiene el inconveniente de aumentar exponencialmente el número de comunicaciones entre los *workers*.

A la vista de estos resultados se puede concluir que:

- *HBOS_BD*, *LODA_BD*, y *LSCP_BD* han sido capaces de detectar eficazmente anomalías en un caso real de estudio, así como en conjuntos de datos de referencia de tamaño normal. *XGBOD_BD* no ha sido capaz de obtener buenos resultados en el conjunto de datos real debido a sus particularidades, pero logró resultados notables en los conjuntos de datos de referencia de tamaño normal, mostrando su rendimiento superior en ciertos escenarios.

- La versión dinámica de *HBOS_BD* se ha establecido como la de mejor rendimiento para el conjunto de datos reales en términos de valor $ROC - AUC$,

Workers	HBOS_BD		LODA_BD	LSCP_BD	XGBOD_BD
	Estático	Dinámico			
4	383,60	13.418,41	273,28	4.864,79	6.525,12
6	315,04	9.229,28	200,83	2.325,94	4.705,57
8	265,03	7.222,37	157,12	2.060,46	3.366,33
10	223,89	6.113,41	138,32	1.651,43	3.175,52
12	208,80	5.370,53	117,99	1.448,69	3.064,11
14	202,80	4.741,01	109,98	1.350,99	2.960,13

Tabla 6.6: Tiempos de cómputo con diferente número de *workers* para *HBOS_BD*, *LODA_BD*, *LSCP_BD*, y *XGBOD_BD* en segundos.

mientras que la versión estática logra excelentes resultados en los conjuntos de datos de referencia de tamaño normal.

- *LODA_BD* ha demostrado ser el método más eficiente en términos de tiempo de computación, siendo capaz de procesar un conjunto de datos *Big Data* en menos de 93 segundos.
- En el *cluster* utilizado, los algoritmos escalan linealmente hasta 8 *workers* y 64 hilos. A partir de ahí, el escalado de reducción de tiempo se hace menos notable.
- Como era de esperar, los métodos ensemble (*LSCP_BD* y *XGBOD_BD*) y la versión dinámica de *HBOS_BD* son los métodos más caros computacionalmente, pero son capaces de lograr un mejor rendimiento que sus algoritmos base.

6.3 CONCLUSIONES

Este capítulo presenta los primeros diseños distribuidos adecuados para la detección de anomalías en entornos *Big Data*, donde el tamaño de los datos disponibles y los problemas de alta dimensionalidad, plantean nuevos retos a los propuestas tradicionales de detección de anomalías. Además, actualmente, hay una ausencia de

propuestas de detección de anomalías en entornos *Big Data*. Por ello, Se han propuesto cuatro algoritmos distribuidos diferentes siguiendo el paradigma *MapReduce*: *HBOS_BD*, *LODA_BD*, *LSCP_BD*, y *XGBOD_BD*.

Los resultados experimentales utilizando un caso de estudio real han demostrado la validez de la propuesta. Todos los métodos han sido capaces de abordar eficazmente un conjunto de datos *Big Data*, alcanzando excelentes valores $ROC - AUC$ en tiempos de cómputo razonables. Estos resultados han sido validados por un experimento adicional utilizando un *benchmark* de conjuntos de datos de tamaño normal para comparar los algoritmos propuestos frente a las versiones clásicas de *HBOS*, *LODA*, *LSCP* y *XGBOD*.

La detección de anomalías es un problema crucial para muchas aplicaciones diferentes del mundo real. Con esta propuesta, se permite a los profesionales detectar anomalías de manera eficiente y eficaz en conjuntos de datos *Big Data*, donde la detección temprana de una anomalía puede conducir a una decisión adecuada y oportuna tanto para las empresas como para el mundo académico.

7

CONCLUSIONES Y TRABAJOS FUTUROS

7.1 CONCLUSIONES

Durante la elaboración de esta tesis se han definido distintos tipos de problemas dentro del marco de la detección de anomalías y, en consecuencia, se han definido dos objetivos para proponer soluciones que ayuden dentro del campo de la detección de anomalías.

El primer objetivo ha sido la elaboración de una metodología de trabajo supervisada para problemas de detección de anomalías en series temporales multivariantes en combinación de mitigación de falsos positivos. Dicha propuesta viene motivada por la escasez de modelos actuales que resuelvan problemas de detección de anomalías en series temporales multivariantes con un mecanismo de control de FPs. Además, existe una falta de propuestas universales en la literatura para la mitigación de FPs. Por ello se han diseñado una metodología en dos etapas, una de detección de anomalías y otra de mitigación de FPs. Los resultados demuestran que la metodología propuesta es capaz de mejorar los actuales resultados presentes en la literatura y que abre un abanico de posibilidad en este marco. Los resultados han mostrado que en función de los resultados de los modelos de detección de anomalías, el comportamiento en la etapa de mitigación de FPs será distinto. Asimismo, es interesante variar los modelos empleados tanto en la etapa de detección de anomalías como en la etapa de mitigación de FPs en función de los objetivos que se tengan para el problema, enfocándolo de manera que se puedan reducir en mayor cantidad los FPs sacrificando TNs o manteniendo una mayor cantidad de TNs en pos de una menor reducción en el número de FPs. Independientemente del problema, se ha demostrado gracias a la experimentación que es posible mejorar la métrica y reducir el número de FPs aplicando la etapa de mitigación. Además, la etapa de mitigación de FPs puede extrapolarse a otros entornos, aportando así más alternativas a la escasez de propuestas de mitigación de FPS universales.

El segundo objetivo ha sido proporcionar nuevos algoritmos a los ya implementados en entornos *Big Data*. En la literatura, los autores ven necesario el diseño e implementación de nuevos modelos ya que los que hay actualmente son muy básicos

o están centrados en problemas concretos. Es por ello que empleando el paradigma *MapReduce* y el framework *Spark* se ha diseñado e implementado una versión distribuida de 4 algoritmos de detección de anomalías clásicos muy exitosos: HBOS, LODA, LSCP y XGBOD. En los resultados se puede comprobar que se igualar o mejorar los resultados de los algoritmos clásicos en términos de la métrica *AUC-ROC*. Además, los tiempos de ejecución en algoritmos como LODA son muy pequeños, 93 segundos para un conjunto de datos de 40 millones de observaciones y 106 atributos. Por lo tanto, se ha conseguido mantener la calidad de los resultados de estos algoritmos bien colocados en la literatura consiguiendo reducir enormemente el tiempo de ejecución de los mismos. Por lo tanto, queda demostrada la utilidad y aplicabilidad de la propuesta para problemas de detección de anomalías en entornos *Big Data*.

7.2 PUBLICACIONES ASOCIADAS A LA TESIS

- López, D., Aguilera-Martos, I., García-Barzana, M., Herrera, F., García-Gil, D., & Luengo, J. (2023). Fusing anomaly detection with false positive mitigation methodology for predictive maintenance under multivariate time series. Information Fusion, 100, 101957. DOI: https://doi.org/10.1016/j.inffus.2023.101957.
  - Estado: Publicado
  - Factor de impacto: (JCR 2022): 18.6
  - Categoría: Computer Science, Theory & Methods (Q1)
    - Posición: 2/111
    - Cuartil: Q1
  - Categoría: Computer Science, Artificial Intelligence
    - Posición: 4/145
    - Cuartil: Q1
- García-Gil, D., López, D., Argüelles-Martino, D., Carrasco, J., Aguilera-Martos, M., Luengo, J., & Herrera, F. Developing Big Data Anomaly Dynamic and Static Detection Algorithms: AnomalyDSD Spark Package.
  - Estado: Sometido

7.3 TRABAJOS FUTUROS

Como se ha analizado en la Sección 5.1.3, es interesante estudiar por qué la alternativa de la metodología que propone mantener la temporalidad para la etapa de mitigación de FPs tiene los resultados obtenidos en la experimentación. Por lo que el principal objetivo será el de estudiar como se podrían gestionar las particiones de manera que se mantenga mejor la temporalidad y ser capaces de analizar de mejor manera si es posible obtener una mejora sustancial en esa etapa.

También sería interesante tratar de llevar al entorno *Big Data* otros algoritmos distintos que permitan dar aún mayor variedad a la batería de algoritmos que hay disponible para su uso en este tipo de problemas.

BIBLIOGRAFÍA

[Agg16] C. C. Aggarwal, *Outlier Analysis*, 2nd. Springer Publishing Company, Incorporated, 2016.

[AMH16] M. Ahmed, A. N. Mahmood y J. Hu, «A survey of network anomaly detection techniques», *Journal of Network and Computer Applications*, vol. 60, págs. 19-31, 2016.

[ASY+19] T. Akiba, S. Sano, T. Yanase, T. Ohta y M. Koyama, «Optuna: A Next-generation Hyperparameter Optimization Framework», en *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[BKK18] S. Bai, J. Z. Kolter y V. Koltun, «An empirical evaluation of generic convolutional and recurrent networks for sequence modeling», *arXiv preprint arXiv:1803.01271*, 2018.

[BH+01] R. J. Bolton, D. J. Hand et al., «Unsupervised profiling methods for fraud detection», *Credit scoring and credit control VII*, págs. 235-255, 2001.

[BSH+21] G. D. Braunstein, L. Schwartz, P. Hymel y J. Fielding, «False positive results with SARS-CoV-2 RT-PCR tests and how to evaluate a RT-PCR-positive test for the possibility of a false positive result», *Journal of Occupational and Environmental Medicine*, vol. 63, n.º 3, e159, 2021.

[BKN+00] M. M. Breunig, H.-P. Kriegel, R. T. Ng y J. Sander, «LOF: identifying density-based local outliers», en *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, págs. 93-104.

[CZS+16] G. O. Campos, A. Zimek, J. Sander, R. J. G. B. Campello, B. Micenková, E. Schubert, I. Assent y M. E. Houle, «On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study», *Data Mining and Knowledge Discovery*, vol. 30, n.º 4, págs. 891-927, jul. de 2016.

[CLA+21] J. Carrasco, D. López, I. Aguilera-Martos, D. García-Gil, I. Markova, M. García-Barzana, M. Arias-Rodil, J. Luengo y F. Herrera, «Anomaly detection in predictive maintenance: A new evaluation framework for temporal unsupervised anomaly detection algorithms», *Neurocomputing*, vol. 462, págs. 440-452, 2021.

[CDZ+16] P. Casas, A. D'Alconzo, T. Zseby y M. Mellia, «Big-DAMA: big data analytics for network traffic monitoring and analysis», en *Proceedings of the 2016 workshop on Fostering Latin-American Research in Data Communication Networks*, 2016, págs. 1-3.

[CC19] R. Chalapathy y S. Chawla, «Deep learning for anomaly detection: A survey», *arXiv preprint arXiv:1901.03407*, 2019.

[CBK09] V. Chandola, A. Banerjee y V. Kumar, «Anomaly detection: A survey», *ACM computing surveys (CSUR)*, vol. 41, n.º 3, págs. 1-58, 2009.

[CCS12] H. Chen, R. H. Chiang y V. C. Storey, «Business intelligence and analytics: From big data to big impact», *MIS quarterly*, págs. 1165-1188, 2012.

[CG16] T. Chen y C. Guestrin, «XGBoost: A Scalable Tree Boosting System», en *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ép. KDD '16, San Francisco, California, USA: ACM, 2016, págs. 785-794.

[DM02] D. Dasgupta y N. S. Majumdar, «Anomaly detection in multidimensional data using negative selection algorithm», en *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, IEEE, vol. 2, 2002, págs. 1039-1044.

[DG04] J. Dean y S. Ghemawat, «MapReduce: simplified data processing on large clusters», en *OSDI'04: Proceedings of the 6th Conference on Symposium on Operating Systems Design and Implementation*, USENIX Association, 2004.

[DRO15] J. Dromard, G. Roudière y P. Owezarski, «Unsupervised network anomaly detection in real-time on big data», en *East European Conference on Advances in Databases and Information Systems*, Springer, 2015, págs. 197-206.

[ERK+16] S. M. Erfani, S. Rajasegarar, S. Karunasekera y C. Leckie, «High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning», *Pattern Recognition*, vol. 58, págs. 121-134, 2016.

[ECN+23] F. Esmaeili, E. Cassie, H. P. T. Nguyen, N. O. Plank, C. P. Unsworth y A. Wang, «Anomaly Detection for Sensor Signals Utilizing Deep Learning Autoencoder-Based Neural Networks», *Bioengineering*, vol. 10, n.º 4, pág. 405, 2023.

[FHL14] J. Fan, F. Han y H. Liu, «Challenges of big data analysis», *National science review*, vol. 1, n.º 2, págs. 293-314, 2014.

[FGD+21] T. Fernando, H. Gammulle, S. Denman, S. Sridharan y C. Fookes, «Deep learning for medical anomaly detection–a survey», *ACM Computing Surveys (CSUR)*, vol. 54, n.º 7, págs. 1-37, 2021.

[GD+20] R. Ganeshan, T. Daniya et al., «A systematic review on anomaly based intrusion detection system», en *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 981, 2020, pág. 022 010.

[GWL21] Y. Gao, H. Wang y Z. Liu, «An end-to-end atrial fibrillation detection by a novel residual-based temporal attention convolutional neural network with exponential nonlinearity loss», *Knowledge-Based Systems*, vol. 212, pág. 106 589, 2021.

[GLG+19] D. García-Gil, J. Luengo, S. García y F. Herrera, «Enabling Smart Data: Noise filtering in Big Data classification», *Information Sciences*, vol. 479, págs. 135-152, 2019.

[GLL+19] D. García-Gil, F. Luque-Sánchez, J. Luengo, S. García y F. Herrera, «From Big to Smart Data: Iterative ensemble filter for noise filtering in Big Data classification», *International Journal of Intelligent Systems*, vol. 34, n.º 12, págs. 3260-3274, 2019.

[GRG+18] D. García-Gil, S. Ramírez-Gallego, S. García y F. Herrera, «Principal Components Analysis Random Discretization Ensemble for Big Data», *Knowledge-Based Systems*, vol. 150, págs. 166-174, 2018.

[GD12] M. Goldstein y A. Dengel, «Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm», *KI-2012: Poster and Demo Track*, págs. 59-63, 2012.

[GU16a] M. Goldstein y S. Uchida, «A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data», en, *PLOS ONE*, vol. 11, e0152173, abr. de 2016, Publisher: Public Library of Science.

[GU16b] M. Goldstein y S. Uchida, «A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data», *PloS one*, vol. 11, n.º 4, e0152173, 2016.

[GAS+21] S. Gopali, F. Abri, S. Siami-Namini y A. S. Namin, «A Comparison of TCN and LSTM Models in Detecting Anomalies in Time Series Data», en *2021 IEEE International Conference on Big Data (Big Data)*, IEEE, 2021, págs. 2415-2420.

[HNG+19] R. A. A. Habeeb, F. Nasaruddin, A. Gani, I. A. T. Hashem, E. Ahmed y M. Imran, «Real-time big data processing for anomaly detection: A survey», *International Journal of Information Management*, vol. 45, págs. 289-307, 2019.

[HM82] J. A. Hanley y B. J. McNeil, «The meaning and use of the area under a receiver operating characteristic (ROC) curve.», *Radiology*, vol. 143, n.º 1, págs. 29-36, 1982.

[HC14] M. A. Hayes y M. A. Capretz, «Contextual anomaly detection in big sensor data», en *2014 IEEE International Congress on Big Data*, IEEE, 2014, págs. 64-71.

[HG09] H. He y E. A. Garcia, «Learning from imbalanced data», *IEEE Transactions on knowledge and data engineering*, vol. 21, n.º 9, págs. 1263-1284, 2009.

[HXD03] Z. He, X. Xu y S. Deng, «Discovering cluster-based local outliers», *Pattern Recognition Letters*, vol. 24, n.º 9-10, págs. 1641-1650, 2003.

[HAB18] S. Hela, B. Amel y R. Badran, «Early anomaly detection in smart home: A causal association rule-based approach», *Artificial intelligence in medicine*, vol. 91, págs. 57-71, 2018.

[HCL+18] K. Hundman, V. Constantinou, C. Laporte, I. Colwell y T. Soderstrom, «Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding», en *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, págs. 387-395.

[HKV19] F. Hutter, L. Kotthoff y J. Vanschoren, *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[JAK01] M. V. Joshi, R. C. Agarwal y V. Kumar, «Mining needle in a haystack: classifying rare classes via two-phase rule induction», en *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, 2001, págs. 91-102.

[Jul03] K. Julisch, «Clustering intrusion detection alarms to support root cause analysis», *ACM transactions on information and system security (TISSEC)*, vol. 6, n.º 4, págs. 443-471, 2003.

[JD02] K. Julisch y M. Dacier, «Mining intrusion detection alarms for actionable knowledge», en *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, págs. 366-375.

[KKW+15] H. Karau, A. Konwinski, P. Wendell y M. Zaharia, *Learning spark: lightning-fast big data analysis*. O'Reilly Media, Inc., 2015.

[KK20] I. D. Katser y V. O. Kozitsin, *Skoltech Anomaly Benchmark (SKAB)*, https://www.kaggle.com/dsv/1693952, 2020.

[KY02] B. Keserci y H. Yoshida, «Computerized detection of pulmonary nodules in chest radiographs based on morphological features and wavelet snake model», *Medical Image Analysis*, vol. 6, n.º 4, págs. 431-447, 2002.

[KTV+22] G. Ketepalli, S. Tata, S. Vaheed e Y. M. Srikanth, «Anomaly Detection in Credit Card Transaction using Deep Learning Techniques», en *2022 7th International Conference on Communication and Electronics Systems (ICCES)*, IEEE, 2022, págs. 1207-1214.

[LVR+16] C. Lea, R. Vidal, A. Reiter y G. D. Hager, «Temporal convolutional networks: A unified approach to action segmentation», en *European Conference on Computer Vision*, Springer, 2016, págs. 47-54.

[LZW+18] C. Li, G. Zhu, X. Wu e Y. Wang, «False-positive reduction on lung nodules detection in chest radiographs by ensemble of convolutional neural networks», *IEEE Access*, vol. 6, págs. 16 060-16 067, 2018.

[LIN+16] X. Liu, N. Iftikhar, P. S. Nielsen y A. Heller, «Online anomaly energy consumption detection using lambda architecture», en *International Conference on Big Data Analytics and Knowledge Discovery*, Springer, 2016, págs. 193-209.

[LGR+20] J. Luengo, D. García-Gil, S. Ramírez-Gallego, S. García y F. Herrera, *Big Data Preprocessing - Enabling Smart Data*. Springer, 2020.

[LCO18] A. M. Lungana-Niculescu, A. Colesa y C. Oprisa, «False positive mitigation in behavioral malware detection using deep learning», en *2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP)*, IEEE, 2018, págs. 197-203.

[Mac67] J. MacQueen, «Some methods for classification and analysis of multivariate observations», en *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, Berkeley, Calif.: University of California Press, 1967, págs. 281-297.

[OBL+18] A. Oussous, F.-Z. Benjelloun, A. A. Lahcen y S. Belfkih, «Big Data technologies: A survey», *Journal of King Saud University-Computer and Information Sciences*, vol. 30, n.º 4, págs. 431-448, 2018.

[Pev16] T. Pevnỳ, «Loda: Lightweight on-line detector of anomalies», *Machine Learning*, vol. 102, n.º 2, págs. 275-304, 2016.

[PGP+21] F. Piccialli, F. Giampaolo, E. Prezioso, D. Camacho y G. Acampora, «Artificial intelligence and healthcare: Forecasting of medical bookings through multi-source time-series fusion», *Information Fusion*, vol. 74, págs. 1-16, 2021.

[Pic+75] J. Pickands III et al., «Statistical inference using extreme order statistics», *the Annals of Statistics*, vol. 3, n.º 1, págs. 119-131, 1975.

[RRS00] S. Ramaswamy, R. Rastogi y K. Shim, «Efficient algorithms for mining outliers from large data sets», en *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, págs. 427-438.

[RFG+18] S. Ramírez-Gallego, A. Fernández, S. García, M. Chen y F. Herrera, «Big Data: Tutorial and guidelines on information and process fusion for analytics algorithms with MapReduce», *Information Fusion*, vol. 42, págs. 51-61, 2018.

[RAP16] M. M. Rathore, A. Ahmad y A. Paul, «Real time intrusion detection system for ultra-high-speed big data environments», *The Journal of Supercomputing*, vol. 72, n.º 9, págs. 3489-3510, 2016.

[RKC+19] L. Rettig, M. Khayati, P. Cudré-Mauroux y M. Piórkowski, «Online anomaly detection over big data streams», en *Applied Data Science*, Springer, 2019, págs. 289-312.

[RBL19] E. Roberts, B. A. Bassett y M. Lochner, «Bayesian anomaly detection and classification», *arXiv preprint arXiv:1902.08627*, 2019.

[RN10] S. J. Russell y P. Norvig, *Artificial intelligence a modern approach*. London, 2010.

[Sca20] A. Scarlat, *Anomaly detection in multivariate time series*, https://www.kaggle.com/drscarlat/anomaly-detection-in-multivariate-time-series, 2020.

[SWP22] S. Schmidl, P. Wenig y T. Papenbrock, «Anomaly detection in time series: a comprehensive evaluation», *Proceedings of the VLDB Endowment*, vol. 15, n.º 9, págs. 1779-1797, 2022.

[SHZ+19] Z. Shi, H. Hao, M. Zhao, Y. Feng, L. He, Y. Wang y K. Suzuki, «A deep CNN based transfer learning method for false positive reduction», *Multimedia Tools and Applications*, vol. 78, n.º 1, págs. 1017-1033, 2019.

[SA22] R. Shwartz-Ziv y A. Armon, «Tabular data: Deep learning is not all you need», *Information Fusion*, vol. 81, págs. 84-90, 2022.

[STN18] S. Siami-Namini, N. Tavakoli y A. S. Namin, «A comparison of ARIMA and LSTM in forecasting time series», en *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, IEEE, 2018, págs. 1394-1401.

[STN19] S. Siami-Namini, N. Tavakoli y A. S. Namin, «The performance of LSTM and BiLSTM in forecasting time series», en *2019 IEEE International Conference on Big Data (Big Data)*, IEEE, 2019, págs. 3285-3292.

[SG14] S. Sridhar y M. Govindarasu, «Model-based attack detection and mitigation for automatic generation control», *IEEE Transactions on Smart Grid*, vol. 5, n.º 2, págs. 580-591, 2014.

[SKK00] M. Steinbach, G. Karypis y V. Kumar, «A comparison of document clustering techniques», en *In KDD Workshop on Text Mining*, 2000.

[TLZ+18] N. Tatbul, T. J. Lee, S. B. Zdonik, M. Alam y J. E. Gottschlich, «Precision and Recall for Time Series», en *Advances in Neural Information Processing Systems*, vol. 31, Curran Associates, Inc., 2018.

[TBJ+20] S. Thudumu, P. Branch, J. Jin y J. J. Singh, «A comprehensive survey of anomaly detection techniques for high dimensional big data», *Journal of Big Data*, vol. 7, n.º 1, págs. 1-30, 2020.

[WZZ+19] X. Wei, L. Zhou, Z. Zhang, Z. Chen e Y. Zhou, «Early prediction of epileptic seizures using a long-term recurrent convolutional network», *Journal of neuroscience methods*, vol. 327, pág. 108 395, 2019.

[Yos04] H. Yoshida, «Local contralateral subtraction based on bilateral symmetry of lung for reduction of false positives in computerized detection of pulmonary nodules», *IEEE Transactions on Biomedical Engineering*, vol. 51, n.º 5, págs. 778-789, 2004.

[ZCD+12] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker e I. Stoica, «Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing», en *Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*, 2012, págs. 15-28.

[Zha03] G. P. Zhang, «Time series forecasting using a hybrid ARIMA and neural network model», *Neurocomputing*, vol. 50, págs. 159-175, 2003.

[ZH18] Y. Zhao y M. K. Hryniewicki, «XGBOD: improving supervised outlier detection with unsupervised representation learning», en *2018 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2018, págs. 1-8.

[ZNH+19] Y. Zhao, Z. Nasrullah, M. K. Hryniewicki y Z. Li, «LSCP: Locally Selective Combination in Parallel Outlier Ensembles», en *Proceedings of the 2019 SIAM International Conference on Data Mining, SDM 2019*, SIAM, Calgary, Canada, mayo de 2019, págs. 585-593.

[ZNL19] Y. Zhao, Z. Nasrullah y Z. Li, «PyOD: A Python Toolbox for Scalable Outlier Detection», *Journal of Machine Learning Research*, vol. 20, n.º 96, págs. 1-7, 2019.

[ZVF+23] J. Zipfel, F. Verworner, M. Fischer, U. Wieland, M. Kraus y P. Zschech, «Anomaly detection for industrial quality assurance: A comparative evaluation of unsupervised deep learning models», *Computers & Industrial Engineering*, vol. 177, pág. 109 045, 2023.

[ZG20] Z. Zohrevand y U. Glässer, «Dynamic attack scoring using distributed local detectors», en *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, págs. 2892-2896.