University of Granada

Department of Computer Science
and Artificial Intelligence

PhD Program in Information
and Communication Technologies

REDUCTION OF FALSE POSITIVES IN ONLINE OUTLIER DETECTION OVER TIME SERIES USING ENSEMBLE LEARNING

PhD STUDENT

ALAIÑE ITURRIA AGUINAGA

PhD ADVISORS

JAVIER DEL SER LORENTE

FRANCISCO HERRERA TRIGUERO

Granada, January 2023

Editor: Universidad de Granada. Tesis Doctorales
Autor: Alaiñe Iturria Aquinaga
ISBN: 978-84-1117-893-8
URI: https://hdl.handle.net/10481/82540

University of Granada

REDUCTION OF FALSE POSITIVES IN ONLINE OUTLIER DETECTION OVER TIME SERIES USING ENSEMBLE LEARNING

MEMORY SUBMITTED BY

ALAIÑE ITURRIA AGUINAGA

FOR THE DOCTORAL DEGREE

January 2023

ADVISORS

JAVIER DEL SER LORENTE

FRANCISCO HERRERA TRIGUERO

Department of Computer Science
and Artificial Intelligence

Título en Español: Reducción de falsos positivos en la detección de anomalías online en series temporales mediante técnicas de ensambles
Título en Inglés: Reduction of False Positives in Online Outlier Detection over Time Series using Ensemble Learning
Programa de doctorado: Programa de Doctorado en Tecnologías de la Información y la Comunicación
Doctorando: Alaiñe Iturria Aguinaga
Directores: Javier Del Ser Lorente y Francisco Herrera Triguero

AUTORIZACIÓN / AUTHORIZATION

El doctorando / The doctoral candidate Alaiñe Iturria Aguinaga y los directores de la tesis / and the thesis supervisor/s: Javier Del Ser Lorente and Francisco Herrera Triguero

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.

Guarantee, by signing this doctoral thesis, that the work has been done by the doctoral candidate under the direction of the thesis supervisor/s and, as far as our knowledge reaches, in the performance of the work, the rights of other authors to be cited (when their results or publications have been used) have been respected.

Granada, January 2023

Doctorando / Doctoral candidate:

Sgd.: Alaiñe Iturria Aguinaga

Director de la tesis / Thesis supervisor:

Sgd.: Javier Del Ser Lorente

Director de la tesis / Thesis supervisor:

Sgd.: Francisco Herrera Triguero

AGRADECIMIENTOS

Empecé esta tesis con mucha ilusión de aprender y retarme. Al escribir estas palabras siento que después de cuatros años, finalmente estoy cerrando esta etapa de mi vida en el que no solo he aprendido mucho a nivel académico, sino también en lo personal. Han sido cuatro años, muy duros y de muchas emociones, que no hubiera superado sin todas las personas que me han acompañado en este viaje y a los que me gustaría dar las gracias.

He de comenzar agradeciendo a mis directores Francisco Herrera y Javier Del Ser, no solo por guiarme en el camino, sino también por enseñarme a buscar el mío propio. Por instruirme en lo académico y también en la vida misma. Me gustaría hacer especial mención a Javier Del Ser, por su buen hacer, positivismo, y por contagiarme su ilusión por la investigación. De verdad espero tener la oportunidad de volver a trabajar contigo.

Quiero extender estos agradecimientos a mi supervisor y compañeros en Ikerlan. En especial a Santi Charramendieta, quien ha supervisado y guiado mi trabajo desde el máster hasta el doctorado, por confiar en mí, por su continuo apoyo, y por ayudarme a mantener la esperanza y la cordura en los momentos más difíciles. También, agradecer a Aizea Lojo y Josu Bilbao por su apoyo y ayuda a que esto saliera. Finalmente quiero dar gracias a los compañeros, que antes o después emprendieron el camino del doctorado junto a mí, Goiuri Peralta, Ane Blazquez, Josu Ircio y Jokin Labaien, por escucharme, comprenderme y ayudarme en lo técnico y en lo personal durante todos estos años.

No pueden faltar mis compañeras de piso, Fátima Fernández y Ana Larrañaga, que desde el primer día que vivimos juntas, se han convertido en mi segunda familia. Muchas gracias, por estar siempre a mi lado, aguantarme, escucharme, entenderme y hacer lo imposible por animarme y sacar lo mejor de mí. Por estar ahí en los momentos tristes, pero también por todos los días sacarme una sonrisa, celebrar las alegrías y compartir nuevas experiencias. El mayor tesoro que me llevo de este camino, muchas gracias patitos. Nunca podré agradeceros todo lo que me habéis dado y ayudado.

Por último, y por supuesto, agradecer a mi familia. A mi madre y a mi padre por su apoyo incondicional, y aunque muchos no sepan por su contribución a la obtención de algunos de los resultados de esta tesis, haciendo mil viajes escaleras arriba y abajo con el portátil mientras estaba encamada. También a mi hermano Iban, por soportarme estos años y por leer y enseñarme a mejorar en la escritura de los artículos. También agradecer al resto de mis familiares, que, aunque no todos sigan entre nosotros, han confiado en mí y me han sacado una sonrisa con solo verlos. Mila esker guztioi!

RESUMEN

Este apartado contiene un resumen en español del contenido de esta memoria de la tesis. El resumen incluye, la introducción, el contexto, la justificación, los objetivos, el desarrollo de la tesis y las conclusiones y trabajos futuros.

INTRODUCCIÓN

La detección de anomalías es un campo de interés en muchos dominios. Por ejemplo, en la industria la detección de anomalías es necesaria para la detección de fallos [1, 2, 3], en ciber-seguridad para detección de intrusos [4, 5, 6] y en entidades bancarias para detección de fraudes [7, 8, 9]. Hasta no hace mucho tiempo la detección de anomalías se hacía de forma manual, estableciendo revisiones periódicas para revisar el estado de las maquinas en el ámbito industrial o bien fijando reglas de forma manual basándose en el conocimiento de expertos en los entornos de ciber-seguridad y en entidades bancarias. Sin embargo, las empresas requieren ser cada vez más competitivas y aprovechar sus recursos al máximo, por lo que sus requisitos son cada vez más exigentes.

Por ejemplo, en la industria requieren buenas estrategias de mantenimiento que permitan detectar los fallos nada más ocurrir e incluso poder predecirlos para maximizar la vida útil de la maquinaria y minimizar los costes. En cuanto a los dominios de la ciberseguridad y las entidades bancarias, requieren técnicas de detección capaces de adaptarse rápidamente a las estrategias de ataque o fraude que evolucionan rápidamente. Para dar soporte a esta demanda, surge la necesidad de modelar e implementar métodos de detección automática capaces de analizar los datos pasados para aprender a predecir o detectar de forma eficaz las anomalías.

CONTEXTO

Teóricamente, un problema de detección de anomalías puede definirse como un problema de minería de datos (DM). La minería de datos se describe como el estudio de la recopilación, la limpieza, el procesamiento, el análisis y la obtención de información útil a partir de los datos. Las contribuciones en esta área de conocimiento intentan crear modelos matemáticos y computacionales que permitan a las maquinas construir sistemas de detección de anomalías de manera automática.

Formalmente, una anomalía se define como un dato o conjunto de datos que difieren significativamente del comportamiento esperado [10]. Dependiendo del ámbito de aplicación, la anomalía puede denominarse de diferentes maneras, como detección de valores atípicos, de fallos, de intrusión o de fraude. Hay varios aspectos a tener en cuenta a la hora de elegir o implementar un algoritmo de detección de anomalías [11]: la naturaleza de los datos de entrada, los tipos de anomalías a detectar, las etiquetas de los datos y la salida de la detección de anomalías. Por ejemplo, los datos pueden ser descritos por una única característica (univariante) o varias características (multivariante) y pueden ser etiquetados para indicar si una instancia es o no anómala. En función de las etiquetas disponibles en el conjunto de entrenamiento, existen tres tipos de aprendizaje: supervisado, semisupervisado y no supervisado. Debido a la dificultad de obtener datos etiquetados, la detección de anomalías suele realizarse en un marco no supervisado. Además, es esencial definir el tipo de anomalía que se va a detectar; pueden ser anomalías puntuales, secuenciales o contextuales. Además, la salida del detector también es determinante; mientras que las salidas de algunos detectores son binarias, otras son puntuaciones de anormalidad que aportan más información.

La detección de anomalías es un problema complejo que debe tratar con conjuntos de datos no etiquetados, conjuntos de datos parcialmente etiquetados o conjuntos de datos desbalanceados, entre otros. Además, recientemente, con la aparición de la digitalización, el Internet de las Cosas (IoT) y la generación masiva de datos, han surgido nuevas necesidades para la detección de anomalías: el procesamiento en streaming u online y el procesamiento BigData [12, 13]. Gran parte de los datos que se recogen actualmente son series temporales, es decir, se almacenan de forma ordenada y están correlacionados en el tiempo. Además, uno de los requisitos más demandados para la detección de anomalías, considerado dentro de los problemas de BigData, es la detección de anomalías en series temporales en streaming u online, donde los datos llegan muy rápido. Por un lado, cuando los datos llegan en tiempo real, el orden o el tiempo de llegada de los datos cobra vital importancia, ya que es una fuente de información extra de gran valor que al ser explotado pueden mejorar los resultados, por ejemplo, detectando patrones repetitivos en el tiempo. De ahí que la detección de anomalías en las series temporales sea de gran interés en el procesamiento online o streaming. Por otro lado, la rápida y continua generación de datos hace que su almacenamiento sea demasiado costoso y a menudo inasequible. Además, el conjunto de datos siempre estará incompleto (siempre habrá nuevos datos por llegar), y con el tiempo la distribución de los datos puede cambiar, haciendo que los modelos queden obsoletos.

JUSTIFICACIÓN

Debido a su interés y creciente demanda, en esta tesis nos centramos en la detección de anomalías online en series temporales univariantes. A pesar de que en los últimos años, han surgido progresivamente varias propuestas de algoritmos todavía hay varios problemas abiertos que abordar.

La primera es que existe muy poco software público para la detección de anomalías en series temporales online. El software público es importante desde un punto de vista aplicado, pero también es indispensable en la comunidad científica para evaluar, comparar y mejorar los algoritmos existentes. Aunque las propuestas algorítmicas son cada vez más numerosas, la mayoría de ellas han quedado relegadas al contexto teórico. Es más, en la mayoría de los casos, están mal explicados, por lo que hay muy poco software público disponible. Debido a la gran demanda, la mayoría del software es de pago, como las soluciones propuestas por diferentes empresas como AWS, Azure y Anodot. El software público existente hasta el momento consta de aproximadamente nueve librerías entre ellas [14, 15, 16, 17, 18, 19, 20, 21, 22] implementados en diversos lenguajes de programación como C++, Java, Matlab, Python y R. Es importante señalar que todos ellos implementan una único algoritmo, por lo que actualmente hay nueve algoritmos disponibles para la detección de anomalías en series temporales de streaming.

La segunda cuestión es que todas las propuestas que hay en la actualidad para la detección de anomalías en series temporales online dan una gran tasa de falsos positivos y negativos. Las razones pueden ser muchas, como los retos derivados del aprendizaje no supervisado, la correlación temporal de los datos, el procesamiento en tiempo real y las características de las series temporales. Por ello, y como está ampliamente demostrado en la literatura, no se puede determinar que un único enfoque o algoritmo sea el mejor respecto al resto. El mejor detector dependerá del problema en cuestión, de las características de la serie temporal, y de los tipos de anomalía entre otros. Por lo tanto, es esencial contar con una amplia gama de detectores de anomalías que se enfoquen en analizar los patrones de series temporales de diferentes maneras.

En relación con esto, a pesar de que no se puede determinar ningún algoritmo como el mejor de todos, muchos estudios recientes en detección de anomalías [12, 23] han demostrado que los ensembles o sistemas de clasificación múltiple están entre las líneas de investigación más prometedoras para obtener detectores de anomalías más robustos y precisos. Estas técnicas combinan varios modelos base (o detectores) para producir modelos más robustos y detectar anomalías de manera eficiente [12]. Esta técnica es útil para robustecer el rendimiento de los detectores, ya que pueden reducir la dependencia del modelo en el conjunto de datos [23] y complementar las debilidades de los detectores individuales y a su vez mejorar sus puntos fuertes. Los ensembles han sido ampliamente estudiados y es una de las

técnicas más efectivas en otras áreas del aprendizaje automatico, como la clasificación [24] o el clustering [25], estando siempre entre los mejores resultados en la mayoría de las competiciones [24]. Sin embargo, los ensembles para la detección de anomalías son todavía un dominio reciente, por lo que *la formalización, la documentación y el software disponible son escasos*. Además, debido al reciente interés en el procesamiento de streaming, actualmente no hay estudios sobre los ensembles para la detección de anomalías en streaming. Por este motivo, es fundamental revisar el estado del arte y analizarlo a fondo para identificar los métodos de combinación disponibles y que sean aplicables al procesamiento en línea.

OBJETIVOS

Esta tesis propone tres soluciones para solventar estos problemas: un software de código abierto que implementa varios algoritmos del estado del arte reciente de detección de anomalías en series temporales online, un marco de trabajo para adaptar cualquier algoritmo de predicción online a la detección de anomalías sobre series temporales, y un novedoso detector de anomalías online basado en el aprendizaje de ensembles capaz de superar a los detectores de la literatura. Para conseguir estos objetivos se han definido las siguientes líneas de investigación:

1. *Estudio del estado del arte de la detección de anomalías online en series temporales y ensembles para la detección de anomalías.*

2. *Implementación de un conjunto de algoritmos de detección de anomalías de series temporales online.*

3. *Framework para adaptar cualquier algoritmo de predicción de series temporales online a la detección de anomalías.*

4. *Una algoritmo novedoso para la detección de anomalías online en series tiemporales utilizando técnicas de ensembles.*

DESARROLLO DE LA TESIS

A continuación, se detalla un resumen de cada uno de los capítulos de la tesis. Debido a su reciente interés y creciente demanda, en esta tesis nos centramos en la detección de anomalías en series temporales univariantes. En los últimos años han surgido progresivamente varias propuestas de algoritmos de detección de anomalías de series temporales online. Sin embargo, todavía hay varios problemas abiertos que abordar: la escasez de software de código abierto y la alta tasa de falsos positivos y negativos de las propuestas del estado del arte. Esta tesis pretende profundizar en la línea de investigación

de la detección de anomalías en series temporales online y resolver estos dos problemas.

El Capítulo 2 introduce los conceptos preliminares necesarios para desarrollar y comprender esta memoria. Este capitulo se divide en dos partes. En la primera se describen los conceptos básicos de la detección de anomalías en series temporales online y en la segunda se recogen los conceptos y el estudio bibliográfico correspondiente a los ensembles de detección de anomalías.

La primera parte, el estudio del estado del arte de los métodos de detección de anomalías en series temporales online describe las diferencias entre los problemas clásicos de detección de anomalías, la detección de anomalías en series temporales y el procesamiento online. Además, también identifica las características específicas de cada uno de los problemas, los tipos de técnicas y los desafíos. Este primer análisis ya deja en evidencia de que a pesar del reciente interés y la demanda emergente, existen pocas propuestas y tipos de técnicas para la detección de anomalías en series temporales online.

En la segunda parte, se realiza una revisión bibliográfica exhaustiva de los métodos de ensemble para la detección de anomalías, prestando especial atención a las técnicas que han sido o pueden ser utilizadas en la detección de series temporales y en el procesamiento online. Aunque las técnicas de ensembles han sido ampliamente estudiadas y utilizadas para obtener algoritmos más robustos en otros campos, los ensembles para la detección de anomalías son todavía una línea de investigación reciente. A través de este estudio, hemos identificado los diferentes pasos para construir un ensemble, así como las técnicas disponibles y los retos para realizar cada uno de estos pasos en la detección de anomalías en series temporales online. Tras este estudio, observamos que los ensembles rara vez se aplican en la detección de anomalías en series temporales online. Sin embargo, concluimos que pueden ayudar a fortalecer los detectores, y es posible generar nuestros propios ensembles siguiendo los pasos y técnicas identificadas en esta tarea.

El Capítulo 3 recoge la primera aportación significativa de esta tesis. En concreto la implementación de una librería R eficiente y fácil de usar llamada *otsad*. *otsad* es el primer paquete de R que recoge un conjunto de detectores de anomalías de series temporales online. En este caso implementa seis de los algoritmos de detección más recientes de la literatura: PEWMA, SD-EWMA, TSSD-EWMA, KNN-LDCD, KNN-CAD y CAD-OSE. También implementa una nueva técnica de reducción de falsos positivos para mejorar significativamente los resultados de los detectores. Inspirada en una situación de la vida real en la que hay un lapso de tiempo entre que se dispara una alarma y hasta que se toma una acción correctiva, nuestra propuesta utiliza el número de datos procesados entre dos anomalías detectadas para reducir el número de falsos positivos. Además, *otsad* también incluye algunas funcionalidades avanzadas, como la técnica de medición del detector NAB y una función de visualización. Por último, se ha realizado un estudio

comparativo de la eficacia y la eficiencia de los métodos implementados para añadir valor al trabajo.

Se han realizado varios experimentos lazando los algoritmos sobre un amplio conjunto de datos de diferentes características y dominios. Para medir el rendimiento de los algoritmos se han evaluado dos aspectos: cual generaliza mejor y cual tiene mayor capacidad de adaptación al conjunto de datos. Por un lado, para ver cual generaliza mejor, hemos evaluado el rendimiento de los detectores implementados utilizando siempre la misma (la mejor) configuración de parámetros de entrada del algoritmo en todos los conjuntos de datos. Por otro lado, para evaluar la capacidad de adaptación, hemos evaluado el rendimiento de los detectores utilizando su mejor configuración de los parámetros de entrada en cada conjunto de datos.

Los experimentos realizados, además de revelar los mejores detectores, muestran que el buen rendimiento de los algoritmos está correlacionado con el uso de técnicas de reducción de falsos positivos y los parámetros de entrada elegidos para cada conjunto de datos, demostrando a su vez que el reductor de falsos positivos propuesto es eficaz. Además, durante el estudio comparativo, observamos que todos ellos dan una alta tasa de falsos positivos y negativos y que todos tienen una alta dependencia del conjunto de datos. Por todo ello, dedujimos que todavía hacen falta mejores estrategias para robustecer los algoritmos y que no hay un algoritmo mejor que otro, sino que este dependerá del conjunto de datos en cuestión.

El Capítulo 4 recoge la última contribución de la tesis que se divide en dos partes. En primer lugar, proponemos un marco que permite generar más algoritmos de detección de anomalías en series temporales online, facilitando la adaptación de los algoritmos de predicción de series temporales online disponibles en la literatura. En segundo lugar, para demostrar la eficacia del marco proponemos un nuevo detector de anomalías basado en ensembles.

El marco pretende permitir la extrapolación de los avances realizados en la predicción de series temporales online a la detección de anomalías y así proporcionar las herramientas para ampliar el catálogo de detectores. El marco propuesto implementa varios métodos de normalización de los datos de entrada y puntuación de anormalidad capaces de funcionar en tiempo real a medida que llegan nuevos datos. Los métodos de normalización de los datos de entrada en tiempo real son muy necesarios no solo en el campo de la detección de anomalías sino también en muchos otros como la clasificación, ya que muchos algoritmos, como por ejemplo las redes neuronales, requieren que los datos estén normalizados en un rango especifico como $[-1, 1]$ o $[0, 1]$ o estandarizados con una media y desviación fijas. Dado que en el procesamiento online la distribución de los datos puede cambiar y no disponemos del conjunto de datos completo para poder calcular el máximo y el mínimo, actualmente son muy pocos los métodos de normalización y estandarización online. Por otro lado para poder adaptar los métodos de predicción de series temporales online a la detección de anomalías, son

necesarias técnicas que dado el error de predicción, es decir la diferencia entre el valor esperado y el predicho, sean capaces de calcular una puntuación de anomalía normalizado en un rango facilmente interpretable como $[0, 1]$.

Con este fin, el marco propuesto implementa algunas de las técnicas de normalización y puntuación de anomalía ya disponibles en los modelos de última generación, así como nuevas propuestas diseñadas para mejor estas. En concreto, se proponen dos nuevos métodos de normalización: one-pass adaptive normalization (OAN) y one-pass adaptive min-max normalization (OAMN), así como dos métodos de puntuación: sigma scoring (SS) y dynamic SS (DSS).

Para demostrar la utilidad y eficacia del marco propuesto realizamos la adaptación de un algoritmo de predicción llamado online recurrent extreme learning machine OR-ELM y proponemos una variante más robusta basada en ensembles. El nuevo detector se ha denominado ensemble-based online recurrent extreme learning machine anomaly detector EORELM-AD y se creó implementando los pasos del marco propuesto sobre un conjunto de OR-ELMs. La propuesta de ensemble combina varias instancias inicializadas con diferentes configuraciones de parámetros de entrada. De este modo, se evita el problema de la selección de los parámetros de entrada, lo que reduce la dependencia del modelo al conjunto de datos. Además, EOR-ELM elimina los modelos que se desvían en cada iteración inicializando nuevos modelos, por lo que puede adaptarse rápidamente a los cambios de distribución y reducir significativamente los falsos positivos. Por lo tanto, EORELM-AD proporciona un enfoque mucho más robusto ante los cambios de distribución y las anomalías en las series temporales.

Los extensos experimentos muestran que el método de puntuación que mejor funciona para el detector propuesto es DSS propuesta en este estudio. Es rápido, no requiere parámetros de entrada, y su rendimiento no depende de la técnica de reducción de falsos positivos. Además, el rendimiento del detector EORELM-AD es competitivo frente a los del estado del arte, y se comporta mejor en varias de las categorías específicas de conjuntos de datos. Basándonos en los experimentos y en las conclusiones extraídas del estudio sobre la eficiencia temporal de EORELM-AD, podemos concluir que el marco propuesto puede servir como herramienta de referencia para que la comunidad adapte los algoritmos de predicción de series temporales online a la detección de anomalías. Además, la estructura del ensemble propuesto es eficiente y puede reducir los falsos positivos.

CONCLUSIONES Y TRABAJOS FUTUROS

En esta tesis, hemos profundizado en la línea de investigación de detección de anomalías en series temporales online. Entre otras muchas, nuestras tareas nos han permitido: adquirir los conocimientos básicos y especializarnos en el tema a través del estudio inicial del estado del arte, aprender a interpretar

artículos científicos implementando los algoritmos del paquete *otsad*, y finalmente aplicar los conocimientos adquiridos para realizar una propuesta innovadora, como es el marco para adaptar los algoritmos de series temporales online y el nuevo detector de anomalías EORELM-AD.

A pesar de que en esta tesis hemos realizado nuestras aportaciones para solucionar la escasez del software público y la alta tasa de falsos positivos y negativos, estos no han sido resueltos del todo. Todavía hay una gran necesidad de obtener algoritmos de detección más robustos y que mejoren el rendimiento de los actuales, así como que el software de estos sea libre. En concreto hemos identificado las siguientes futúras líneas de investigación:

- Investigar nuevos métodos de generación de diversidad para el procesamiento online capaces de generar nuevos subconjuntos de datos seleccionando dinámicamente instancias o características manteniendo la información de correlación temporal entre los datos.
- Investigar nuevos métodos de combinación de ensembles específicos para series temporales que tengan en cuenta la correlación temporal de los resultados a la hora de combinar.
- Generar nuevos ensembles heterogéneos combinando algoritmos de diferentes tipos, como los implementados en *otsad*, e investigar si es posible obtener un mejor detector que supere los individuales.
- Utilizar técnicas de paralelización para mejorar la eficiencia temporal de los ensembles. A pesar de que se ha demostrado que es temporalmente eficiente, destacamos la estructura inherentemente paralela del enfoque EORELM-AD propuesto que tras su programación utilizando paradigmas de paralelización como el map-reduce, debería permitir su uso sobre datos de series temporales de velocidad ultra alta.
- Proporcionar nuevos algoritmos de detección basados en otros modelos de aprendizaje incremental con métodos de entrenamiento eficientes.
- Realizar un estudio detallado de la adaptabilidad del algoritmo a series temporales con valores atípicos de diferente naturaleza y la caracterización incremental de las anomalías.
- Aplicar los conocimientos adquiridos y continuar con la investigación en series temporales multivariantes.

ABSTRACT

The identification of unusual patterns or anomalous features allows critical information to be extracted from data. With the development of the Internet of Things, real-time time series anomaly detection has become a relevant task in many domains, including fault detection in the manufacturing industry, intrusion detection in cybersecurity, and fraud detection in banks. Within this research area, online time series anomaly detection is a more challenging task compared to classical outlier detection for several reasons. First, a complete dataset is not available for training, and therefore, training must be performed incrementally. Second, every new incoming data sample must be processed once without multiple passes through the entire dataset (i.e., one-pass learning). Third, the distribution of data is non-stationary and can change over time (concept drift), requiring the inclusion of adaptation/forgetting mechanisms in outlier detection methods. These constraints hinder the design of online methods that effectively detect anomalies in time series data under such challenging conditions.

This Thesis delves into research around the detection of anomalies in online time series. In recent years, several proposals for online time series anomaly detection algorithms have gradually emerged. However, the literature study performed in this Thesis has identified two crucial shortcomings in online time series anomaly detection: the scarcity of open-source software and the high rate of false positives and negatives of state-of-the-art proposals.

To reduce false positives, the Thesis focuses on ensemble techniques. Ensemble techniques are helpful to solve this type of problem since they can reduce the dependence of the model on the data set and complement the weaknesses of single detectors while enhancing their strengths. Furthermore, several of the most recent studies on anomaly detection demonstrate that ensembles or multiple classifier systems are the most promising research line to obtain robust and accurate detectors.

In order to provide more open source software, the first significant contribution of this thesis is the implementation of an efficient and easy-to-use R library named *otsad* and the comparative study of implemented detectors. *otsad* is the first R package that collects a set of online time-series anomaly detectors: PEWMA, SD-EWMA, TSSD-EWMA, KNN-LDCD, KNN-CAD, and CAD-OSE. It also implements a new false positive reduction technique to improve detectors' results significantly. Inspired by a real-life situation where there is a time-lapse between an alarm being triggered and until corrective action is taken, our proposal uses the number of processed data points between two detected anomalies to reduce the number of false positives.

Furthermore, it also includes some advanced functionalities, such as the NAB detector measurement technique and a visualization function. Finally, a comparative study of the effectiveness and efficiency of the implemented methods was carried out to add value to the work.

The last contribution is divided into two parts. First, we propose a framework that allows the generation of more online time series outlier detection algorithms by facilitating the adaptation of available time series prediction algorithms. The framework aims to allow the extrapolation of the advances made in online time series forecasting to anomaly detection and thus provide the tools to expand the detectors catalog. The proposed framework implements several online normalization and outlier scoring methods already available in state-of-the-art models, as well as novel proposals designed to improve upon baselines. Specifically, two novel normalization methods—one-pass adaptive normalization (OAN) and one-pass adaptive min-max normalization (OAMN), as well as two scoring methods—sigma scoring (SS) and dynamic SS (DSS) are proposed.

Then, we demonstrate the usability and efficacy of the proposed framework by discussing the adaptation of a novel ensemble-based online recurrent extreme learning machine, EORELM-AD. The EORELM-AD was created by implementing the steps of the proposed framework over an ensemble of Online Recurrent Extreme Learning Machines. The ensemble proposal combines several instances initialized with different parameter settings. In this manner, the hyperparameter selection problem is circumvented, which reduces the dependency of the model's configuration on the target dataset. Furthermore, EOR-ELM removes deviating models in each iteration by initializing new models, so it can quickly adapt to distribution changes and significantly reduce false positives. Therefore, EORELM-AD provides a much more robust approach to distribution changes and anomalies in time series.

To conclude, extensive experiments on well-known benchmark datasets for time series outlier detection are presented and discussed, yielding two main conclusions. First, the performance of the proposed EORELM-AD detector is competitive in comparison to several state-of-the-art outlier detection algorithms. Second, the proposed framework is a useful tool for adapting an online time series prediction algorithm to outlier detection.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Part I

INTRODUCTION

1

INTRODUCTION

The recognition of unusual patterns or anomalous features allows extracting critical information from data, hence providing helpful information for a variety of applications in diverse activity sectors, such as industry [1, 2, 3], cyber-security [4, 5, 6], and banking [7, 8, 9], among others. Anomalies in data collected in these domains can be a valuable asset or a critical threat, depending on the event/phenomenon causing it in practice. For example, in the industry, an anomaly can represent a machine failure. In cyber-security applications, an anomaly may expose the presence of an intruder. Likewise, in banking an anomalous event can be an alarm of fraud or theft. Due to the rapid technological evolution and competitiveness, the success of today's business environment depends on detecting these opportunities and threats as quickly as possible, as close to real time as possible.

Until recently, when there were only a few metrics to be controlled, anomaly detection was carried out manually, mainly by means of domain-driven expert rules. In industrial setups, periodic reviews were scheduled to check the status of machines, while for intrusion and fraud detection, rules were manually fixed and tailored based on domain experts' knowledge. Much differently, nowadays companies collect a greater number of metrics, which are furthermore captured at significantly faster rates. Consequently, the requirements to exploit this collected data flows effectively are becoming increasingly demanding. For example, good industrial maintenance strategies are required to detect failures as soon as they occur and even predict them, maximizing machinery's remaining useful life and eventually minimizing maintenance costs. Similarly, the cybersecurity and banking domains require detection techniques that quickly adapt to rapidly evolving attacks and fraud strategies so as to avoid the damage caused by these events. To support this demand, a necessity to model and implement automatic detection methods emerges to analyze past data and to detect anomalies effectively, meeting the near real-time requirements of the aforementioned data streams.

1.1. CONTEXT

From a theoretical perspective, an anomaly detection problem can be defined as a *Data Mining* (DM) problem. DM is described as the study of collecting,

cleaning, processing, analyzing, and extracting useful insights from data [26]. Contributions in this knowledge area attempt to create mathematical and computational models that allow machines to build automatic anomaly detection systems by learning them from the available data. Formally, an anomaly is defined as data that differ significantly from the expected behavior [10]. Depending on the application domain, the anomaly can be regarded in different ways, such as an outlier, a fault, an intrusion, or a fraud event. There are several aspects to consider when designing and implementing an anomaly detection algorithm [11], such as the nature of the input data, the types of the anomaly to be detected, the *a priori* availability of annotation, and the output of the anomaly detection algorithm itself. For example, data can be described by a unique feature (univariate) or several features (multivariate), and can be labeled to indicate whether an instance is or is not anomalous. Depending on the availability of annotation in the training data, three types of learning tasks can be generally distinguished in DM, namely, supervised, semi-supervised and unsupervised learning. Due to the difficulty of obtaining labeled data for anomaly detection, this particular DM task is usually conducted within an unsupervised framework. Moreover, it is essential to define the anomaly type to be detected, which can be point, sequential or contextual anomalies. In addition, the output of the detector is also another aspect to be determined; while the output of some detectors is binary, other detectors elicit soft anomaly scores, providing more information about the anomalous nature of the input data to the end user.

Anomaly detection is a complex problem that must deal with unlabeled data sets, partially labeled data sets, or labeled yet mostly imbalanced data sets, among other practical issues. Moreover, recently, with the emergence and progressive maturity of digitization, the Internet of Things (IoT), and the massive generation of data in almost any field, new needs for anomaly detection have arisen: among them, streaming/online processing and Big Data processing [12, 13]. Much of the data currently collected by sensors and monitoring systems are time series, i.e., data instances are stored orderly and are correlated over time. Furthermore, one of the most stringent computational requirements for anomaly detection within problems that fall within the Big Data paradigm is anomaly detection over time series in streaming or online settings, where data samples are generated continuously and arrive very fast. On the one hand, when data comes in the form of streams, the order or time of arrival of the instances becomes critical for the detection of anomalies. Time is hence a domain over which data is produced and, when exploited by the algorithm, can improve the results, for example, by detecting concurrently occurring anomalous patterns over time. Particularly, the detection of anomalies in time series is of great interest in streaming processing. On the other hand, the rapid and continuous data generation makes its storage too expensive and often unaffordable. Furthermore, the data set will always be incomplete as there will always be new samples to arrive and

be fed to the algorithm. This continuous flow and the possible presence of exogenous factors affecting the statistical characteristics of the produced data streams may cause that their distribution changes over time, making the knowledge captured by the anomaly detection algorithm potentially obsolete.

1.2. MOTIVATION AND HYPOTHESIS

Due to its recent interest and growing demand, this Thesis focuses on anomaly detection over streaming univariate time series. In the last few years, several proposals have been reported in the literature to deal with this problem. However, there remain several open issues that deserve further research efforts:

- The first remarkable open problem is the *scarcity of open-source software* available for online anomaly detection over time series. The public availability of software is of great importance from an applied point of view, but it is also crucial for the scientific community to evaluate, compare, and improve existing algorithms for this modeling task. Although the spectrum of algorithmic proposals stemming from this field has grown sharply in recent times, most proposals have been relegated to the theoretical context. What is more, in most cases proposals are loosely explained and their implementation is very poorly organized and documented, so that very little public software can be found in the literature that can support future research advances solidly. Due to the high demand for these algorithms, most available software implementing them require a paid subscription, such as the solutions contained in the frameworks proposed by different companies, such as AWS, Azure, and Anodot. In detail, the public software for online anomaly detection consists of nine libraries [14, 15, 16, 17, 18, 19, 20, 21, 22] implemented in various programming languages, including C++, Java, Matlab, Python and R. It is important to note that all these libraries implement a single proposal, implying that currently there are only 9 algorithms available for anomaly detection over streaming time series.

- The second issue noted in this research area is that all up-to-date proposals for online outlier detection over time series still render *high false positives and negatives detection rates* when applied to real-world time series. Reasons for this statement can be many and diverse in nature, such as the challenges that learning from unsupervised stream data can impose, the effective exploitation of the temporal correlation among successive data instances along the stream, the algorithmic design constraints set by the low latency required by streaming processing, and the evolving features of the analyzed time

series. Due to these reasons, it has been widely acknowledged and proven in the literature that no single modeling approach can be concluded to perform best when compared to any other algorithm when the comparison is made over the totality of streaming time series that can occur in practice. Consequently, the best performing detector depends on the problem at hand, the characteristics of the time series, and the types of anomaly to be identified, among other factors. Thus, it is essential to have a wide range of anomaly detectors capable of differently analyzing and discerning anomalies from time series.

Related to this second issue, although no algorithm can be claimed to perform best in all cases, many recent studies in anomaly detection [12, 23] have shown that ensembles or multiple classification systems are among the most promising lines of research to obtain more robust and accurate anomaly detectors. Ensembles rely on different strategies to combine several base learners (or detectors) together to produce more robust models and detect anomalies efficiently [12]. These strategies can be helpful to improve the performance of anomaly detection algorithms, since they can reduce the dependence of the model on the data set [23] and complement the weaknesses featured by single detectors, while preserving their individual strengths. Ensemble learning has been widely studied in the literature [27], and is an effective approach to tackle learning problems in other areas of Machine Learning, such as classification [24] or clustering [25]. Indeed, ensembles are among the top performers in most competitions [24].

Interestingly, ensembles for anomaly detection are still a domain in its infancy, with very few contributions falling in this crossroads to date. Consequently, a *formalization* of ensemble learning models as online anomaly detection algorithms over time series is needed, supported by a thorough and informed study of the state of the art on streaming anomaly detection ensembles. This analysis is necessary to identify which ensemble-based proposals can meet the requirements of online processing. Furthermore, following up on the first issue noted above, unified and well-documented software for implementing online anomaly detection methods is also in need towards supporting advances in this area. These two statements, which are later argued in further detail, constitute the justification of the research presented in this Thesis.

1.3. CONTRIBUTION AND OBJECTIVES

The overarching contribution of this Thesis to the niches presented in the previous section are threefold: i) an open-source software that implements several avant-garde online time series anomaly detectors; ii) a framework for adapting any online prediction algorithm to outlier detection over time series; and iii) a novel online outlier detector based on ensemble learning.

Specifically, these research advances break down into different objectives that must be pursued to realize them fully:

- *Study of the state of the art related to online time series anomaly detection, as well as outlier detection ensembles*: a necessary first step of the Thesis is to collect and inspect in depth the different proposals reported to date for the online detection of anomalies in time series data, so that a profound understanding of the benefits and weaknesses featured by the most competitive approaches can be gained in an informed fashion. Furthermore, analyzing the different ensemble strategies followed by the community to detect outliers from data is essential to build new ensemble detectors that improve current baselines used for this task in online settings.

- *Implementation of online time series anomaly detection algorithms*: departing from the knowledge acquired after studying the state of the art of online time series anomaly detection algorithms, the Thesis pursues to implement several selected algorithms in a public software library in response to the scarcity of public software available for the community. Beyond its inherent contribution to the research community, the implementation of competitive outlier detection methods for time series can help identify the algorithmic aspects at their core that cause the general high false detection rates observed in other studies, and outline how to overcome them by adopting ensemble techniques. For this reason, a second objective of the Thesis is to develop a public software library that implements some of the most recent online time series outlier detectors.

- *Framework for adapting any online prediction algorithm to outlier detection over time series*: no single detector can be considered better than the others over the totality of possible datasets. Therefore, having a systematic framework to produce a wide range of anomaly detectors of different natures can be very suitable to evaluate and choose an approach that best suits the online anomaly detection problem at hand. For this reason, the third objective of the Thesis is to design a framework to adapt online time series prediction techniques to anomaly detection. Online time series forecasting is a more mature research area than online outlier detection over time series, and can be used as a profitable source of modeling alternatives to be adapted for online anomaly detection. As the third objective, the Thesis aims to construct a methodological framework to allow the extrapolation of the advances made in online time series forecasting to anomaly detection, encompassing streaming data normalization, online anomaly scoring and the identification of outliers from forecasting prediction errors.

This framework provides the community with the tool chain needed to expand the current portfolio of available online anomaly detectors.

- *Exploration of ensemble learning for the online time series anomaly detection task*: finally, the Thesis seeks to explore whether the use of ensemble learning can yield competitive algorithms for the detection of anomalies in time series. To this end, the literature study (first objective), the implemented software library (second objective) and the methodological framework (third objective) support a fourth objective: the creation, implementation and performance assessment of a neural network-based bagging ensemble that has been used for online time series prediction, adapting it to detect outliers from time series data in an online fashion.

1.4. RESEARCH METHODOLOGY

The above set of objectives will be accomplished by following several scientific methodological steps, which are adapted to the specific needs of this Thesis. Such steps are summarized as follows:

1. *Problem characterization*, which includes the review and understanding of the scenarios and problems to be addressed in the Thesis, as well as their characteristics. Specifically, the Thesis covers unsupervised anomaly detection in online time series and ensembles for outlier detection, from both classical and online points of view.

2. *Hypothesis formulation*, which spans three different contributions: the design and development of an open-source software package, a framework for adapting online time series prediction algorithms to anomaly detection, and an ensemble-based anomaly detector.

3. *Experimentation*, which encompasses the implementation of the software needed to validate the formulated hypotheses and, where applicable, the evaluation of the results of the novel algorithmic proposals over public datasets used by the research community working in this area. Performance evaluation is done by complying with the standards used for the task at hand, including performance metrics, statistical significance assessment and repeated trials.

4. *Hypothesis contrasting*, which refers to an empirical comparison of the results obtained by the proposed approaches and those of state-of-the-art algorithmic alternatives. This analysis measures the capacity of the proposed algorithms to solve problems in different scenarios.

5. *Hypothesis validation or refutation*, which implies accepting or rejecting – and modifying – the stated hypothesis once the results gathered in

the different experiments have been analyzed. If the hypothesis is rejected, changes must be considered before iterating over the previous methodological steps toward eliciting a new set of experimental results.

6. *Scientific thesis*, including the extraction, writing, and acceptance of the conclusions obtained during the process. The findings and their scientific approach must be collected and presented in a thesis dissertation and peer-reviewed journal publications.

1.5. STRUCTURE OF THE THESIS

The Thesis is organized in three parts: Part I, which introduces the reader to the context, motivation and objectives of the Thesis (Chapter 1) and presents preliminary concepts necessary to develop and understand the technical contributions thereafter (Chapter 2); Part II, which exposes in details the two novel contributions of the Thesis in connection to the objectives stated previously:

- In correspondence with the second objective of the Thesis, Chapter 3 presents *otsad*, the first R package which implements a set of novel online anomaly detection algorithms for univariate time-series, along with some advanced functionalities and contents, such as a new false positive reduction algorithm, and a novel NAB detectors measurement technique. This chapter starts with a description of the implementation and functionalities of the package (Section 3.2). Then, extensive experimentation is carried out to compare the performance and time efficiency of the proposed algorithms (Section 3.3). In addition, some illustrative examples of the use of the *otsad* package are provided (Section 3.4).

- Part II ends with Chapter 4, where two contributions are presented to cover the third and fourth objectives of the Thesis. First, a framework to generate new online time series anomaly detection algorithms by adapting available time series prediction algorithms is described in Section 4.2. Second, by harnessing this designed framework, a new ensemble-based online time series anomaly detection algorithm is proposed (Section 4.3) and experimentally compared to several competitive approaches from the state of the art (Section 4.4 and Section 4.5).

Finally, the Thesis concludes with Part III, which expands a single Chapter 5 in which final remarks and lessons learned from the research results attained in the Thesis are outlined. This chapter also includes a summary of the scientific publications where the research outcomes have been presented, as well as an outlook toward future research stimulated by the findings reported in the Thesis.

2

PRELIMINARIES

This Thesis deals with two important research topics that have been widely studied in the literature: anomaly detection and ensemble methods. Moreover, it delves into less addressed issues in each of such topics: online anomaly detection in time series, and ensembles for anomaly detection. As a starting point, this chapter introduces preliminary concepts of these main topics to provide the reader with the background required to understand the contributions that follow this chapter. Section 2.1 introduces the basic concepts related to online time series anomaly detection, whereas Section 2.2 presents the theoretical framework of ensembles used for anomaly detection.

2.1. ONLINE TIME SERIES ANOMALY DETECTION

Anomaly detection is a vast field, which has been studied for decades. However, tackling anomaly detection in online time series requires first introducing some of the basic concepts of both traditional anomaly detection and anomaly detection in time series. For the sake of comprehensiveness, this section describes background knowledge about these two tasks, so that their combination can be understood in a smoother and better informed manner. Specifically, the section starts with the basic principles of traditional anomaly detection posed in Section 2.1.1. Then, fundamentals of time series outlier detection are introduced in Section 2.1.2. The chapter concludes by addressing online time series anomaly detection in Section 2.1.3.

2.1.1. Anomaly detection

Theoretically, an anomaly is defined as an observation significantly different from the rest of the observations and of interest to the analyst [28]. There are several terms which many researchers used interchangeably as synonyms of anomaly [29, 28], such as outlier, abnormality, noise, discordant observations, discords, exceptions, aberrations, surprises, peculiarities, or contaminants. Moreover, as described in the following sections, outlier detection is a complex problem that must deal with imbalanced, partially labeled, or unlabeled data sets. Moreover, anomalies may have different definitions depending on the application domain into others. Due to the diversity of practical uses for

which the identification of anomalies can be pursued, anomaly detection is an important field for many domains, such as industry [1, 2, 3], energy [4, 5, 6] and security [7, 8, 9].

The following sections describe the main aspects of anomaly detection problems (Section 2.1.1.1), systematically classify anomaly detection techniques (Section 2.1.1.2), and enumerate several challenges of outlier detection problems that remain unsolved to date (Section 2.1.1.3).

2.1.1.1. Main aspects of anomaly detection problems

There are several aspects to consider when choosing or implementing an anomaly detection algorithm [11]. The most important ones are summarized in Figure 2.1, namely, the nature of the input data, types of anomaly to be detected, availability of annotated labels, and the output sought for the anomaly detection approach.

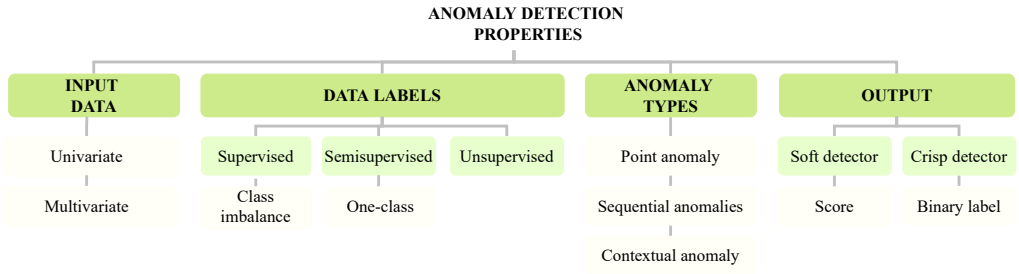


Figure 2.1: Main design aspects of anomaly detection problems.

NATURE OF THE INPUT DATA: Usually, the input data is a collection of data instances, where each instance can be described by a unique feature (univariate) or several features (multivariate). Such features can be of different types: binary, categorical, continuous, or mixed. Besides the application, the representation of the data may differ. Data can be multivariate, without any relation between successive points, can be sequential and ordered temporally, or be defined in the form of a network with arbitrary relations between the data instances.

ANOMALY TYPES: The definition of what is normal or anomalous is not an easy task. It depends on the context of each problem and domain. Even so, in the classical literature, three types of anomalies are distinguished as shown in Figure 2.2: point anomaly, collective anomalies, and contextual anomaly. A *point anomaly* occurs when a single data instance can be considered anomalous when compared to any other. On the other hand, *collective anomalies* happen when a collection of instances is anomalous with respect to the entire dataset. Finally, a *contextual anomaly* holds if and only if an

instance is deemed to be anomalous in a specific context, not the entire dataset.

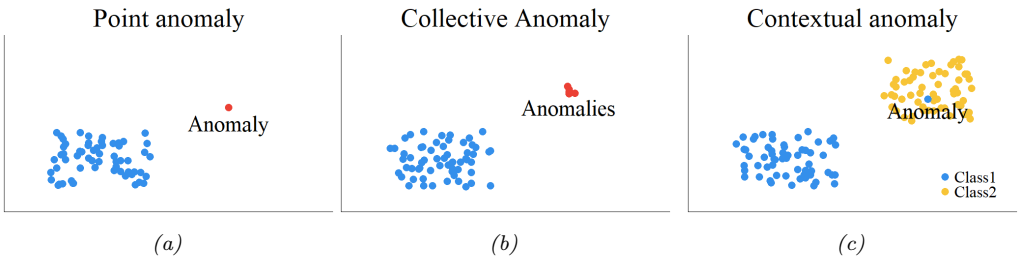


Figure 2.2: Types of anomaly that can occur, exemplified in a two-dimensional dataset. Figure (a) shows a point anomaly isolated from the rest of the points of the set. Figure (b) depicts the collective anomalies as a set of reduced points away from the rest in the data set. Figure (c) illustrates a contextual anomaly where the class gives the context. In this case, the value of the anomalous point is normal within the complete data set. However, its value is anomalous for the class to which it belongs.

DATA LABELS: In some practical scenarios, it can be possible to have a labeled training data set so that each instance has a label that indicates whether it is or it is not anomalous. Depending on the availability of these labels on the training set (*annotation*), three learning types can be defined: supervised, semi-supervised and unsupervised. *Supervised learning* techniques assume that the training data set is composed of labeled data containing both abnormal and normal instances. *Semi-supervised learning* techniques assume that only some instances are labeled as normal and/or anomalous, whereas the rest lacks any annotation. For example, only normal class instances can be available. Finally, *unsupervised learning* assumes that the entire training data set is unlabeled, and can be composed of both normal and anomalous instances.

OUTPUT OF ANOMALY DETECTION: The output resulting from the application of an anomaly detection technique can be of two different types. The first and most common option is to inform about the presence of an anomaly in terms of an anomaly score, that is, the probability or confidence of the model on the input instance to be anomalous. A detector that produces scores is called a soft detector [30, 8]. The second option is to report a binary predicted label that indicates if the input instance is or is not anomalous. When detectors produce a label instead of a score, they are referred to as crisp detectors. It should be noted that the transformation of anomaly scores into binary labels is straightforward by applying a cut-off threshold to such scores, labeling those instances that exceed the threshold as anomalous.

2.1.1.2. Anomaly detection techniques

Over the years, different categories have been proposed to classify all anomaly detection methods. Based on some of the older and most recent studies [11, 31, 12], these are the most relevant categories:

- *Statistical-based techniques:* These techniques calculate the difference between a point and the model or statistical distribution assumed for the data, and determine that a point is anomalous if this difference is greater than a threshold. These methods distinguish two categories: parametric-based anomaly detection algorithms [32, 33] and non-parametric anomaly detection algorithms [34, 35].

- *Nearest neighbor-based techniques:* these approaches analyze the distance between a test instance and its nearest neighbors. They assume that the distances from an anomalous point to its neighbors are much larger than the distances between a normal point and its neighbors. In the literature a manifold of proposals can be found within this category [36, 37, 38].

- *Clustering-based techniques:* Techniques falling within this category group normal data into a cluster space, and label as anomalous data those instances that do not fit into the clusters within that space, i.e., anomalies that belong to very small groups or that are far from the centroids of the clusters that are representative of normal data. Some clustering based proposals are reported in [39], [40] and [41], to cite an exemplifying few.

- *Spectral-based approaches:* These methods attempt to approximate normal data by using features with lower dimensionality that capture most of the data variability. It assumes that data can be represented in a lower-dimensional subspace, where normal and anomalous instances can look significantly different from each other [42].

- *Information-theoretic-based methods:* these algorithms rely on different information-theoretic measures (e.g., entropy or relative entropy) to analyze the information content in the data at hand. They further assume that the anomalies in the data induce irregularities in the information content. Some examples are those proposed in [43], [44] and [45].

- *Ensemble-based methods:* schemes inside this last category combine several diverse models to detect outliers more reliably. To this end, such diverse models explore different views (subspaces) of the input data, so that their outputs are combined to yield a consensus detection on the abnormal nature of the input data. Several proposals have

been done in recent years that leverage the use of ensembles for the detection of outliers [46, 47, 48].

2.1.1.3. Challenges of anomaly detection problems

Despite its profitable past of research achievements, anomaly detection is a complex problem with many challenges to be faced, which makes each practical scenario in which this task is accomplished unique and difficult to solve on its own. A representative fraction of usual difficulties that anomaly detection problems encounter in practice is offered below:

1. Definition of the normal region is a challenging task. The boundary between normal and abnormal is not always clear as per the information and indications provided by domain experts towards the definition of the task.

2. Related to the previous point, when anomalies result from malicious acts, attackers tailor their attacks to make them appear as normal as possible. Thus, distinguishing normal instances from anomalies is even more difficult in such adversarial settings. This is the reason for which the recent past has witnessed the emergence of a new field (*adversarial machine learning* [49]), focused on the development of adversarial attacks for Machine Learning models, as well as defenses to counteract them effectively.

3. The notion of anomaly is different for each application domain. For example, in the medical domain, small fluctuations (i.e., body temperature) can be an anomaly, while a similar fluctuation in the stock market domain can be normal.

4. The noise often mimics the behavior of anomalies in the data space, so it is difficult to discern and eliminate anomalous events based on their effect on the input data.

5. In multivariate datasets, anomalies can be hidden and visible or detectable for only a subset of the features. Therefore, as the dimensionality of the dataset increases, it becomes unfeasible to explore all feature subsets and reliably detect eventual anomalies that reflect only on some of them.

6. The cost, difficulty, or even impossibility of obtaining annotation is already a problem in itself, so having annotated anomalous data for all types of anomalies under target is often an unfeasible assumption to approach the problem from a supervised learning framework.

7. The lack of annotation is a problem for measuring the detector's performance, therefore affecting the performance-driven choice of the best

model, its hyperparameter tuning and comparison to other counterparts.

2.1.2. Time series anomaly detection

A time series is a time or chronological-ordered sequence of observations that are correlated in time [28]. Due to continuous data generation and recollection, time series is one of the most used data types in our life. For example, credit, personal, financial, judicial, medical, or web usage data are temporal. In this context, time series outlier detection aims to analyze anomalous behaviors over time series data, becoming arguably one of the main tasks that can be tackled in time series data mining. It has been studied in various application domains, such as credit card fraud detection, fault detection in industry, and intrusion detection in cybersecurity. Surprisingly, only two literature reviews can be found out related to this research area: the first one was introduced in [50], whereas the second one was recently contributed in [28].

The following sections introduce time series anomaly detection: in Section 2.1.2.1, essential time series properties are presented. Next, Section 2.1.2.2 summarizes different techniques used to detect anomalies in this particular type of data, while challenges to be addressed are discussed in Section 2.1.2.3.

2.1.2.1. Properties of time series

As in traditional anomaly detection problems, time series data have their own properties, such as input data, stationarity, and anomaly types, which are summarized in Figure 2.3. Such properties are decisive to choose and eventually implement an anomaly detection algorithm for time series data.

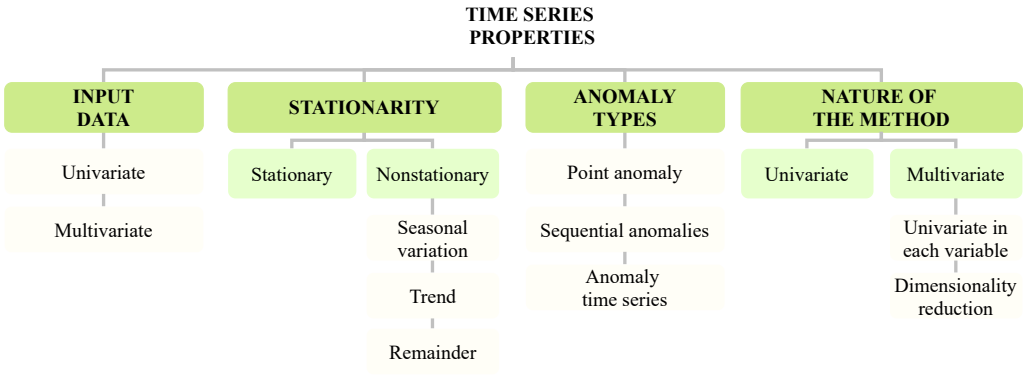


Figure 2.3: Properties of time series data that are relevant for the design of anomaly detection algorithms over them.

INPUT DATA: Time series can be univariate or multivariate. A univariate time series is an ordered set of real-valued observations, where each observation is recorded at a specific time. In contrast, a multivariate time series is described by ordered k-dimensional vectors, where for each specific time, k real-valued observations are collected.

STATIONARITY: Time series can be stationary or non-stationary. Stationary time series are those in which the mean and variance are constant over time. An example of stationary time series is depicted in Figure 2.4. In contrast, as shown in Figure 2.5, in non-stationary time series the mean or variance may change over time.

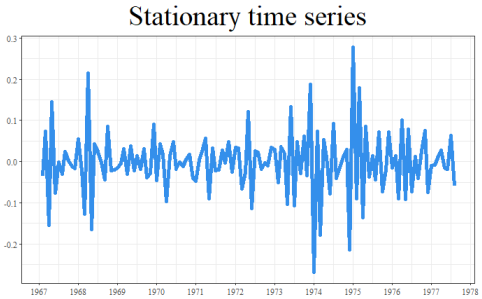


Figure 2.4: Example of a stationary time series.

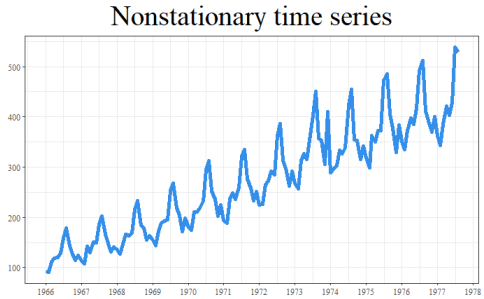


Figure 2.5: Example of a non-stationary time series.

Nonstationary time series can be represented as the sum of three components: trend, seasonal variation, and noise. An example of a time series decomposition is shown in Figure 2.6. *Trend* reflects the direction in which the series aims at in the long term. *Seasonal variation* reflects the oscillations that occur around the trend, repetitively and in short-term periods. Finally, *noise* or *remainder* are the residual values that are not explained by the trend or the stationary components of the time series, and may or may not have a random statistical behavior.

Most traditional statistical algorithms in the literature are based on stationary time series because of its several advantages. First, as the mean is constant, it is easily estimated to predict a new observation. Furthermore, assuming that the data follow a known distribution, such as a normal, it is also possible to estimate prediction (or confidence) intervals, assuming that the data follow a known distribution, such as a normal. However, although stationary time series have significant advantages, in most cases, the time series are nonstationary. For that reason, many statistical techniques try to convert nonstationary time series into stationary ones, such as [22], by employing different strategies to eliminate these three components of the time series.

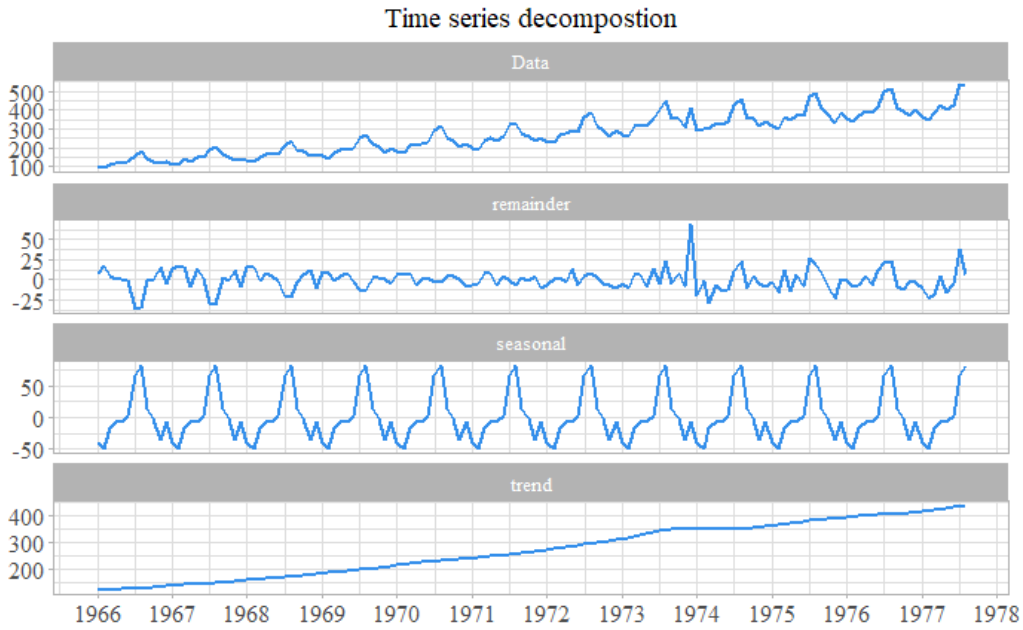


Figure 2.6: Example of a time series decomposition.

ANOMALY TYPES: The recent study in [28] distinguishes among three anomaly types in time series: point anomalies, sequential anomalies, and anomaly time series. *Point anomaly* is a data instance considered anomalous concerning others in a specific instant of time. Such anomalies may arise in univariate and multivariate time series, and can be global or local. An anomaly is called *global* when a data instance is considered anomalous concerning the time series, and *local* when it is considered anomalous when compared to its neighboring points. Instead, *sequential anomalies* are consecutive instances in time whose behavior is unusual. As in point anomalies, sequential anomalies can also be global or local. To conclude, *anomaly time series* refers to an anomalous entry time series. This kind of anomaly can only be detected when the input data is a multivariate time series.

NATURE OF THE METHOD: Anomaly detection methods can be univariate or multivariate. Univariate methods only consider a single time-dependent variable, whereas multivariate can work with more than one time-dependent variable. There are two techniques to deal with multivariate time series. The first one consists of employing univariate techniques over each time-dependent variable. Some of these methods do not consider variable correlation. Instead, other approaches use dimensionality reduction techniques to find a new set of uncorrelated variables, and then a univariate detection method is applied to each of them. The second method is to employ fully multivariate techniques. This method can only be applied to multivariate time series, and they cannot be used in univariate time flows.

2.1.2.2. Time series anomaly detection techniques

According to the aforementioned studies in [50, 28], anomaly detection techniques for time series can be classified into three categories: model-based methods, distance-based methods, and histogramming:

- *Model-based or statistical methods*: these are the most common approaches in the literature. They determine if a point at a time x_t as anomalous if the distance to its expected value x'_t is higher than a predefined threshold. These methods distinguish two sub-categories: estimation model-based anomaly detectors and prediction-based anomaly detectors. Estimation models [51, 52, 22] need previous and subsequent observations to the current data to compute the expected value, while prediction-based techniques [53, 54, 53] only require past data to calculate the expected value.

- *Dissimilarity-based methods*: approaches in this category measure the previous dissimilarity between multivariate points or representation, and they do not need to fit a model. An anomaly is detected if the dissimilarity between the current multivariate point and its expected value is higher than a predefined threshold. Some dissimilarity-based anomaly detectors are [55] and [56].

- *Histogramming*: These techniques compare the error given by the original histogram generated from all the data and the histogram obtained by removing the point from the data. The point is detected as an anomaly if the histogram representation when removing the point yields a smaller error than the original one. An anomaly detection algorithm hinging on this principle is the one proposed in [57].

2.1.2.3. Challenges in anomaly detection problems over time series data

As a result of the temporal correlation between data points in time series, additional challenges emerge besides those already present in traditional anomaly detection problems:

- The statistical distribution underlying the data points in a time series may change over time as a consequence of the effect of exogenous factors (*concept drift*), causing that the consideration of what an anomaly is by the anomaly detection algorithm may become obsolete. Hence, models that adapt suitably to non-stationary time series data must be able to detect distribution changes and adapt to them when they occur.

- Input data may be collected in regular or irregular time intervals, and may contain missing values. It is essential to analyze such missing values to declare them as an anomaly and impute them before model

training. In anomaly detection over multivariate time series, data corresponding to different time-dependent variables must be collected within the same time intervals.

- Time continuity plays an essential role, since time series data values are highly correlated at successive points in time. In multivariate time series, time correlation may exist between different variables. Thus, some anomalies can only be observed in specific subspaces. Moreover, when several time-dependent variables are correlated with each other, outliers in one variable can cause new outliers to appear in the correlated variables.

- Temporal context is determinant to detect anomalies, since it is possible that a point is a normal value, but not at the specific time instant at which it is observed. For example, 25 degrees Celsius can be a normal temperature in summer, but not in winter.

2.1.3. Online time series anomaly detection

With the emergence of digitization, the Internet of Things (IoT), and massive data generation, online time series outlier detection has become a relevant task in many domains, such as fault detection (prognosis) in manufacturing industries [58, 59], intrusion detection in cybersecurity [60], and fraud detection in banks [61]. Due to the speed at which data are generated, massive data storage is not always possible, and its distribution may change over time. Thus, new challenges have emerged: i) the entire dataset is not available for learning, and hence learning must be performed incrementally; ii) every newly incoming data must be processed once, without multiple passes through the entire data set (namely, *one-pass* learning); and iii) the distribution of data is not stationary and can change over time (concept drift), requiring the inclusion of adaptation/forgetting mechanisms in outlier detection methods [50, 19]. These constraints hinder even further the design of time series anomaly detection methods that effectively detect anomalies in time series data under such challenging circumstances.

Most contributions reported to date in online times series outlier detection [19, 62] consist of adapting traditional offline learning algorithms to stream processing. Recent research focuses on two different methodologies [28]: retrain the model each time a new data arrives or learn the model in an incremental manner by adjusting the learning algorithm itself. These approaches are often combined with techniques to handle drifting data distributions over the stream, leading to several of the most renowned learning algorithms for non-stationary data streams known in the area [63].

Sections that follow hereafter introduce some of the main aspects related to online time series anomaly detection. To begin with, Section 2.1.3.1 describes the properties of streaming time series. Then, different techniques

for outlier detection in online time series are categorized and summarized in Section 2.1.3.2. Finally, similarly to previous section, challenges of anomaly detection resulting from the requirements of online processing are discussed in Section 2.1.3.3.

2.1.3.1. Properties of online time series anomaly detection

In addition to those characterizing traditional anomaly detection and offline time series anomaly detection, online time series anomaly detection problems have their particular properties that are determinant when choosing or implementing an online anomaly detection algorithm for time series. These main characteristics are summarized in Figure 2.7, and consist of the amount of arrived data each time, types of online learning, and types of anomalies in online processing.

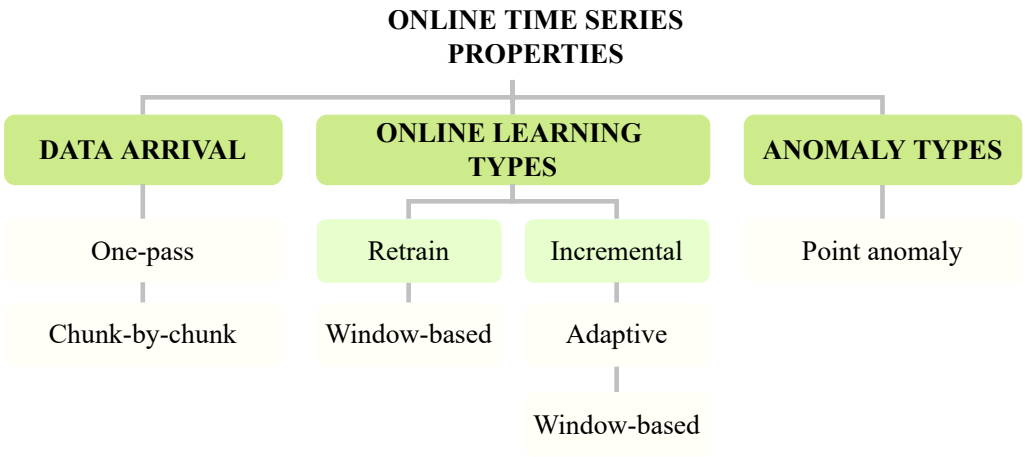


Figure 2.7: Properties of online time series anomaly detection problems.

DATA ARRIVAL: One of the main characteristics to consider when designing or choosing a model is how the data is received. Data may come in the form of samples, one at a time, named one-pass processing, or chunk-by-chunk, also referred to as batch processing. In both cases, models must process and detect anomalies in time whenever new data arrive, either as samples in isolation or in chunks.

ONLINE LEARNING TYPES: There are several ways to adapt to the evolution of time series. The most intuitive one is to *retrain the model* within the past data every time new data arrive. This is usually done by resorting to a sliding window of past data, which is used as the data for retraining the model. However, this strategy may be computationally expensive depending on the model, and past information relevant for the discrimination of what is

normal and what is anomalous can be lost depending on its temporal depth. A generally better approach is based on *incremental learning*, where the model is not rebuilt from scratch every time new data arrive, but is instead updated incrementally by using only the new information received. Two techniques can be distinguished to realize incremental learning: adaptive methods and window-based methods. *Adaptive methods* update the statistics whenever a new value arrives, without any need for retaining historical values. These methods require little memory and are very fast. On the other hand, *window-based methods* use sliding windows to maintain and update parameters on the most recent data [28]. These techniques are especially useful for training neural networks and distance-based methods, as sliding windows reduce the set of observations to be considered and maintain the most recent subset of data that may influence the current data.

TYPE OF ANOMALIES: Online time series anomaly types should be the same as those defined in traditional time series anomaly detection, i.e., point anomalies, sequential anomalies, and anomaly time series. However, currently all streaming processing works are only focused on point anomaly detection.

2.1.3.2. Online time series anomaly detection techniques

As already mentioned, point anomaly detection for univariate time series is closely related to online processing. Hence, as can be concluded from most recent studies, many proposals are for real-time and fall within three general categories [28]: model-based, density-based, and histogramming.

- *Model-based approaches*: as explained for offline time-series anomaly detection, two model-based strategies can be noted: estimation-based [64, 62] and prediction model-based [65, 66, 19].
- *Density-based methods*, such as [67, 20, 68], analyze the number of neighbors within a distance. They build upon the intuitive assumption that anomaly points have fewer neighbors, less than τ within a distance threshold.
- *Dissimilarity-based approaches*: As introduced in offline time series detection techniques, these methods measure the pairwise dissimilarity between the current multivariate point and its expected value, and detect it as an anomaly if the dissimilarity is higher than a predefined threshold. A representative dissimilarity-based detector is proposed by Li et al. [69].
- *Histogramming*: as before, histogramming methods consider that a point represents an anomaly if the new histogram generated by removing the query point gives a smaller error than the histogram using the

entire available data. Muthukrishnan, Shah, and Vitter [57] introduced a proposal based on histogramming.

2.1.3.3. Challenges of online time series anomaly detection problems

In addition to the traditional anomaly detection and offline time series anomaly detection, the requirements of online processing jeopardizes even further the detection of anomalies in a number of aspects:

- As in traditional anomaly detection over time series, in online processing time series can also be stationary or non-stationary, the latter being the most common circumstance in stream settings. Consequently, models must be able to adapt to distribution changes over time, so as to preserve steady detection figures in varying data contexts.
- In steam processing, the training data continuously change, and can be incomplete since data is constantly collected, and models must be updated with recent data. Furthermore, as a result of this variability and incompleteness, the scale of the data, mean, and variance are not static, and are difficult to reliably compute or estimate for a significantly long time, hindering even further the process of identifying abnormalities over the flowing data.
- In streaming environments, normal behavior may evolve and not be sufficiently represented in the future.
- The high dimensionality of the data, especially in Big Data problems, can insert redundant, irrelevant, or correlated characteristics that can damage the definition of normality and the model itself.
- The arrival speed is also critical. Data can be produced very fast, e.g., each second, or more sparsely, e.g., every ten minutes or each hour. When data flows fast, storing a massive amount of data points is computationally costly, even unaffordable by any technological means. Moreover, according to the arrival time interval, some models may be limited due to their high time computational cost and their inability to incrementally learn within the imposed latency and/or memory constraints.
- To handle multiple large data streams, computing systems must be able to distribute the workload in parallel, assigning processing tasks to more than one physical processor. This distribution requirement and the need for exploiting the time correlation pose a challenge for the design and parallel implementation of anomaly detection algorithms.

2.2. ENSEMBLES

Ensemble learning, also referred to as *committee-based learning* or *learning multiple classifier systems*, are methods used in data mining that combine the results of multiple base learners (or detectors) to enhance the overall reliability and accuracy of the model when undertaking a given modeling task [70]. By combining diverse learners, ensembles can reduce the dependence of the model on the specific data set, and overcome the weak performances resulting from the lack of ground truth. Furthermore, they alleviate the model selection problem, and have in general better generalization abilities than a single method.

Ensembles are extensively studied in different data mining contexts, such as classification or clustering. However, their application to anomaly detection tasks is more limited. Investigations conducted on anomaly detection ensembles have shown that the theoretical foundations of ensembles in this field are not very different from those known for classification ensembles. As a result, much of the research is focused on adapting these ensemble techniques to anomaly detection problems. Nevertheless, certain limitations arise from the natural absence of ground truth that makes this line of research more challenging.

The construction of an ensemble is defined by two phases: base learner generation and combination method selection (see Figure 2.8 and Figure 2.9). Each phase can be broken down into several sub-steps. On the one hand, base learner generation can be divided into two steps: algorithm(s) selection and diversity generation. On the other hand, the selection of the combination method comprises another two sub-steps: component selection and combination method selection. Component selection is optional, but it can be used to improve the overall performance of the ensemble even further.

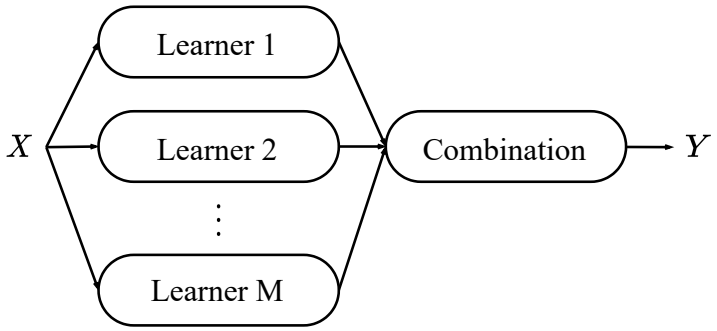


Figure 2.8: Common ensemble schema.

The first step, algorithm(s) selection, consists of choosing the most suitable algorithm(s) that will constitute the ensemble components. With this aim, the main aspects of specific anomaly detection problems, and the characteristics, advantages, and drawbacks of each technique, described in Section 2.1,

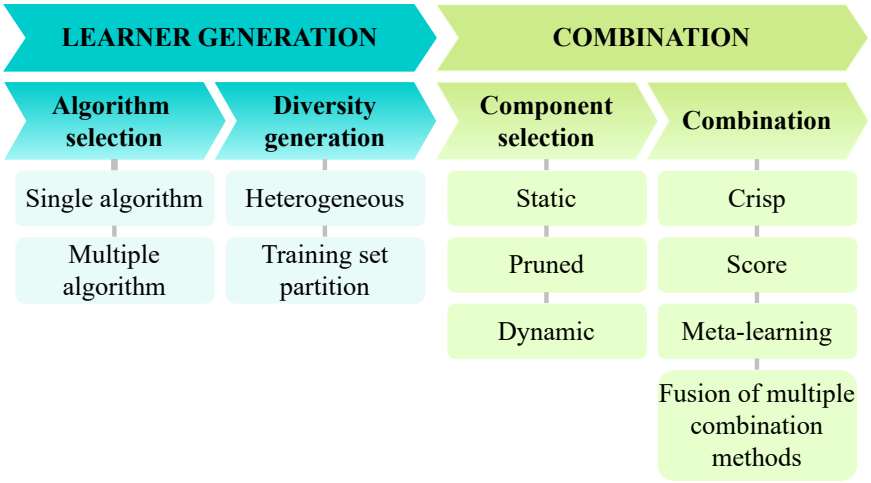


Figure 2.9: Design steps of an ensemble anomaly detection algorithm.

must be considered. As the ensemble comprises several base learners, it is possible to choose a single algorithm (homogeneous base learners) or combine several different techniques (heterogeneous base learners), to complement the weaknesses of single detectors while enhancing their strengths.

The rationale for the rest of steps and a summary of the methods available to implement them in practice are described in the following sections. In particular, Section 2.2.1, Section 2.2.2, and Section 2.2.3 correspond to the analysis of diversity induction, combination, and selection, respectively. Then, Section 2.2.4 reviews shortly different types of ensemble strategies. To conclude, Section 2.2.5 analyzes the challenges of this specific flavor of anomaly detection algorithms.

2.2.1. Diversity induction techniques

The diversity among the individual base learners is a key aspect to ensure the effectiveness of an ensemble when solving a given learning task. That is, the results of base learners when queried with a given input data instance should be as different as possible, yet coherent with respect to the data distribution to be modeled. It is intuitive to think that to harness the combination of the learners' outputs, the results of the base learners should be different; otherwise, there will be no improvement in performance. Hence, in general it is convenient to increase the number of learners that will take part in the ensemble, provided that they are built differently from the available data.

When approached as an unsupervised learning problem, the absence of ground truth in outlier detection problems makes it difficult to effectively measure and guarantee a degree of diversity between the base learners in the ensemble. However, heuristic mechanisms for diversity generation exist to construct outlier detection ensembles. This section examines in depth such diversity generation methods, categorizing them in data-based diversity

induction methods (Section 2.2.1.1), model-based diversity induction (Section 2.2.1.2), and different strategies to combine several diversity induction techniques (Section 2.2.1.3).

2.2.1.1. Data-based diversity induction

Data-based diversity induction methods are based on the idea that each part/view of the dataset provides a specific insight about its underlying distribution. Therefore, using an ensemble over different portions/views of the data allows modeling its distribution diversely, yet coherently. These techniques aim to induce diversity training the base learners on different training subsets generated by distinct parts, samples, projections, or functions of the original data. Different training subsamples can be drawn by selecting a sample of data instances, or by choosing a relevant subspace of the feature space. At this point it is necessary to distinguish between instance-based and feature-based diversity generation methods, depending on the domain over which such diversity is produced:

- **Instance-based diversity generation**: these techniques generate different training subsamples by choosing a fraction samples of the original data. This sampling is performed by following a given criteria, such as random sampling. Diversity generation techniques utilized in recent contributions include:
  - *Bagging*: this method is one of the most popular instance-based diversity generation methods in classification ensembles. Bagging applies bootstrap sampling to generate different base learners. Given a training data set of n training samples, bagging uses sampling with replacement to generate different subsamples and train a base learner on each subsample. This process implies that some original instances may appear more than once in the samples drawn from the available, while others will not be present. As a result of this bootstrapping process, a variation named wagging has been recently analyzed by [23]. As duplicated data points can be seen as a kind of weight that adds diversity, in wagging all the points are retained, choosing their weights explicitly from a particular distribution.
  - *Subsampling*: some outlier detection algorithms (e.g., Local Outlier Factor, LOF [36]) are not robust to the presence of repeated points, making the bagging approach an unsuitable choice for creating diversity in outlier detection ensemble. To overcome this, given a training data set of n training samples, a training subsample can be drawn by sampling without replacement. Over the years, some variants of subsampling have emerged, including variable subsampling [71] to reduce the bias induced by the data size, and geometric subsampling [23].

- *Boosting/Data set pruning*: boosting is very popular in data classification, having coined well-established ensemble models for this specific task (e.g., AdaBoost [72]). Specifically, boosting is a sequential ensemble technique that uses weak base learners to classify instances into normal or anomalous. At each iteration, misclassified samples are given an increasingly higher relevance. By proceeding over several iterations, base learners constructed along the boosting chain strive to classify difficult examples correctly. Ultimately, a set of diverse learners is produced and their output is combined. Due to the absence of ground truth, it is hard to identify anomalous samples and thus, weight them correctly along the boosting chain. Consequently, very few unsupervised sequential boosting ensembles can be found in the literature, and most of them aim to refine the training data set, removing possible anomalous samples instead of weighting them [73].
- *Other approaches*: alternative diversity induction methods operating on the data themselves have been designed based on the properties of the selected base models. Some examples are projected clustering [47], adaptive sampling [74], stream mini-batch learning [75] and noise injection into the training dataset [5], among others.

- **Feature-based diversity generation:** differently to data-based approaches, feature-based diversity generation techniques generate different training subsets by choosing or extracting features from the original data. This is done by following any of the criteria described in what follows:
  - *Feature bagging or random subspace*: this method exploits the data-locality effect in detecting high-dimensional anomalies. During several iterations, it selects some dimensions uniformly at randomly (without repetition) from the original dataset, and creates dimensional projections or training sub-datasets. There are few works, such as [76] (supervised), [3, 77] (semi-supervised) and [48] (unsupervised), whose proposal for inducing diversity in their utilized ensembles rely on this feature bagging approach. Recently, a few variations have emerged with the aim to reduce the existing correlations among detectors, such as rotated bagging [71] and non-redundant feature bagging [78].
  - *Random projection*: Recently contributed by Khan and Ahmad [77], this method maps several points to a high-dimensional space, where the Euclidean distance of any two points in the original feature space is approximately preserved in the projection.
  - *Randomized feature weighting*: This method was introduced by Aggarwal and Sathe [23], and can be viewed as a soft version of feature bagging. Distinct to other feature bagging methods, in randomized feature weighting no dimension is dropped. Instead, features

are weighted with the use of a specific distribution to produce the weights.

- *Other approaches*: other strategies have been adopted to create different training sets by operating on the features of the original data. For example, some works [79, 58, 80] train one or more models per feature, data source or data source type. Other methods use different feature extraction techniques to create new training datasets [80, 5, 81]. To conclude, Wang, Mao, and Huang [82] uses Kernel Principal Component Analysis (KPCA) to create new diverse feature sets.

2.2.1.2. Model-based diversity induction

Model-based diversity induction methods hinge on the idea that differently generated base models can learn from the same data in different ways. Thus, when used for anomaly detection, these techniques aim to induce diversity by creating diverse base learners by using various anomaly detection algorithms, initializing them with different hyperparameters, or by adding randomness to their learning process. Specifically, these model-based diversity induction strategies give rise to three different subcategories, namely, heterogeneous base learners, parameter tuning, and model randomization, which are further described below:

- **Heterogeneous base learners**: an effective way to achieve diversity between models is by selecting different anomaly detection techniques as base learners. These methods model anomalies differently from each other, based on processing and analyzing diverse aspects of the data towards detecting anomalies in them. Using distinct detection techniques in constructing an ensemble makes it possible to generalize unique base models that analyze the data from different points of view. Due to its simplicity and effectiveness, this is one of the most used techniques in recent contributions to this area, not only when tackling anomaly detection as a supervised learning problem [2, 7, 9, 46, 83, 84, 85, 86], but also for semi-supervised [6, 8, 87, 88, 89, 90, 91, 92] and unsupervised learning [93, 94, 95, 96, 97].

- **Parameter tuning**: This technique consists of initializing the same anomaly detection algorithm by using different hyperparameter sets. Hyperparameter selection can be made manually based on previous domain knowledge or, as it occurs in most cases, randomly over reasonable value ranges for every hyperparameter. This technique is very common in semi-supervised [98, 99, 100, 101] and especially in unsupervised [102, 30, 103] formulations of the anomaly detection problem. The absence of ground truth when formulated as an unsupervised learning problem makes it difficult to choose the most appropriate hyperparameter values for each

practical circumstance in which anomalies are to be detected. Moreover, as most outlier detection algorithms have different associated parameters, this technique circumvents the problem of selecting the best hyperparameters and hence, allows generating more robust anomaly detection ensembles. Nevertheless, it is important to note that varying the hyperparameter values may not necessarily lead to very dissimilar base learners, hence producing non-diverse results within the ensemble. Thus, hyper-parameter tuning is used as a complementary diversity generation method in many unsupervised works published in the area [104, 105, 82, 106, 107, 108].

- **Model randomization**: another alternative way to induce diversity among models in an ensemble is to introduce randomness at different steps of the learning process of the base detector. Randomness can be generated in many ways. For example, a common method is the one used in [74], which introduces random connections by randomly dropping connections in a neural network or in this particular case, autoencoders. This method is also used to build well-known random forests [109] for classification or isolation forests [13] for anomaly detection. In this case, a random feature selection is performed within the learning process rather than before learning, as it is done when using feature bagging.

2.2.1.3. Combining multiple diversity induction methods

Except in those cases where heterogeneous base learners are used (and even in some of these cases), it is common to combine several diversity generation techniques to ensure the effectiveness of the ensemble for the modeling task at hand. A plethora of works combining several diversity induction techniques can be found in the literature for supervised [110, 1], semi-supervised [87, 91, 90, 92, 3, 111, 76, 112, 5, 80, 113] and unsupervised anomaly detection [108, 104, 82, 106, 105, 107, 74]. From the unsupervised point of view, almost all the works classified in this section use parameter tuning as one of the complementary diversity techniques. As mentioned previously, this is done because in unsupervised problems it is not straightforward to select the best hyper-parameter values for the model in the absence of annotation that can drive this choice.

2.2.2. Combination techniques

As in other modeling tasks, the combination phase of an anomaly detection ensemble is an essential part of the design process. Combining different models can reduce both variance and bias, providing more accurate and robust results. Therefore, this is a crucial phase to get strong generalization skills. In the literature, there are several combination methodologies and variants. In this section, a new taxonomy is proposed to categorize them according to the type of base model outputs. As such, a distinction is done

between crisp-wise combination, score-wise combination, meta-learning-wise combination and the fusion of multiple combination methods. Figure 2.10 depicts this taxonomy proposed to classify the detection methods.

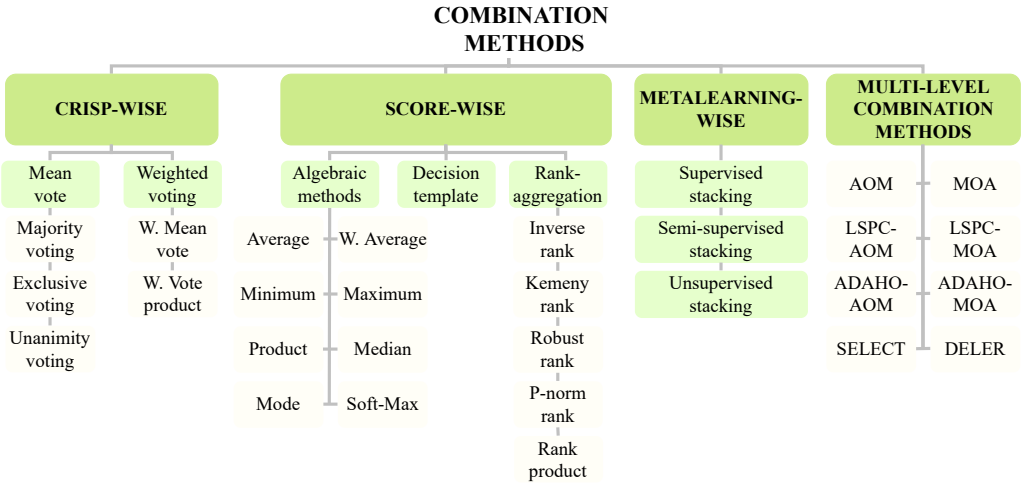


Figure 2.10: Classification of the combination methods that can be used to aggregate the outputs of the base detection algorithms in an anomaly detection ensemble.

2.2.2.1. *Crisp-wise combination*

Crisp-wise combination techniques combine the binary detection labels obtained by the different base detectors. A usual strategy to combine anomaly scores that are not comparable to each other is to convert them into binary outputs by applying a threshold. However, it is essential to note that this can lead to relevant information loss. In the following, we distinguish two popular crisp-wise combination methods: voting and weighted voting.

- **Voting**: The voting method is the most popular and fundamental method of the literature related to ensemble learning. Voting-based ensembles resemble an electoral system in which each base learner votes for one of the classes. Based on the agreement percentage, or the number of votes for each class, three voting variants can be distinguished:
  - *Majority voting*: This is the most popular voting strategy. Here each base model votes for one class label, and the final output class label is the one that receives more than half of the votes. Majority voting is the most used crisp combination method for supervised [83, 109, 114, 84], semi-supervised [75, 79, 4, 77, 112] and unsupervised formulations of the anomaly detection problem [96, 108].
  - *Exclusive voting*: This is is the least restrictive variant of the *voting* combination method. Here, an instance is labeled as an anomaly if at least one detector detects it as such. This combination method is

useful for reducing the rate of false negatives of the ensemble. A use case of this combination method is proposed by Diao, Naqvi, and Pecht [95].

- *Unanimity voting:* Unanimity voting is the most restrictive voting variant. In this case, an instance is labeled as an anomaly if all the detectors detect it as such. It is useful for reducing the rate of false positives of the ensemble. An interesting use of this method is proposed by Liu et al. [47].

- **Weighted voting**. Weighted voting is the most powerful crisp combination method. It can be used when the base detectors perform unequally to each other. Intuitively it makes sense to grant a higher importance to the detection results provided by the most accurate base detector. This importance is modeled by defining different weights per base model. Usually, such weights are normalized in such way that their sum equals 1. The weight assignment is usually allocated based on the base learners' performance. That is why it is more common to note the use of weighted voting in anomaly detection problems formulated over supervised data [2, 114].

2.2.2.2. Score-wise combination

Scoring-wise combination techniques fuse the anomaly scores obtained by the different base learners. Combination techniques within this category can be further divided into three subcategories: ranking-based methods, algebraic-based techniques and decision template-based techniques.

- **Algebraic Methods**: in the literature, there are several score-wise combination methods. These methods require anomaly scores to be comparable to each other, having to be normalized or standardized if their value ranges differ from each other. The most popular score-wise combination methods are the average and its weighted version. However, there are other approaches that have been less frequently adopted, but that they are essential in the ensembles' literature tackling other modeling tasks: maximum, minimum, median, mode, sum and product. In what follows all algebraic combination methods are described in further detail:

  - *Average*: This is the most popular score-wise combination method. There are few recent works for semi-supervised [5, 80, 91, 92] and for unsupervised learning [13, 106, 73] that use this combination method to construct their ensembles. This method consists of averaging the anomaly scores given by the base detectors. In addition to the traditional average over the years, some variations such as *soft voting* [86] and *dumped average* [115] have been explored.

- *Weighted average*: The weighted average is a powerful variant of the average combination method. In this case, as in weighted majority voting, each base model has a different weight according to its performance or diversity. Despite the absence of ground truth in unsupervised learning settings, this combination method has been very popular in the last years, being used in several semi-supervised [99, 98, 100, 89, 101, 76] and unsupervised ensembles proposals [105, 82, 107]. Other variations of the weighted average combination method have been proposed in recent times, including the so-called *exponential induced ordered weighted averaging (EIOWA)* approach presented in [116].
- *Maximum*: This method computes the maximum of anomaly scores achieved by each detector. It can be considered as a scoring version analogous to exclusive voting. This technique can obtain considerably better results than those obtained by averaging, especially in complex datasets (i.e., with high dimensionality where anomalies are well hidden).
- *Minimum*: This method calculates the minimum of anomaly scores achieved by each detector. It can be considered a scoring version analogous to unanimity voting. This technique can be useful to reduce high false positive rates. However, it usually leads to an increased rate of false negatives, and hence this method is hardly used.
- *Median*: this combination method reports the median score over the base learners. It gives a more stable central representation of scores than the average, and is not greatly affected by unusual deviations. This method has been used in several unsupervised anomaly detection works [74, 117]. The major disadvantage of this combination strategy is that part of the underlying diversity between the base detectors can be lost.
- *Mode*: this method computes the output as the most repeated anomaly score between all detectors. This method is rarely used in outlier ensembles, but it can be useful to construct outlier detection based on forecasting ensembles, as done in [118].
- *Product*: This technique calculates the product of all the anomaly scores. It is especially appropriate for anomaly scores that are interpreted as a probability. In statistics, under the assumption that all detectors are independent from each other, the product rule allows finding the probability that an instance is detected as an anomaly by all detectors simultaneously.
- *SoftMax*: this combination method is used by Chakraborty, Narayanan, and Ghosh [110], and is similar to the average and soft voting methods. It uses the softmax function to combine normalized anomaly

scores. Although this combination method is proposed for a supervised multi-class classification task, it is worth noting that it can be extrapolated for semi-supervised and unsupervised learning problems, as well as for binary detection problems.

- **Rank aggregation**: in some cases, the anomaly scores obtained by the different base detectors are not comparable to each other, so the threshold selection or score normalization can be complex to perform. Rank aggregation is an effective unsupervised alternative that allows combining the scores of different detectors without the need for normalization or the use of a specific threshold. Due to its inherent advantages, rank aggregation is a popular method among recent unsupervised anomaly detection ensembles [119, 120, 94]. These methods transform the anomaly scores obtained by each detector into a rank list ordering the scores in descending order. On the negative note, the main drawback featured by this method is that it dismisses information concerning the relative difference between anomaly scores. Nevertheless, rank aggregation has a rich history in which a variety of solutions have been proposed. Next, rank aggregation methods used in recent works are summarized:
  - *Kemeny Young rank aggregation.* Recently used by Rayana and Akoglu [94] and proposed by Kemeny [121], Kemeny Young rank aggregation is a voting strategy that combines several rank lists using preferential ballot and pair-wise comparison counts, in which the detectors are treated as voters and the points as candidates to vote for.
  - *Inverse rank aggregation.* This method was recently used in [94]. It re-scores the anomaly scores of each instance based on the inverse of its ranking position $1/r_{pos}$. Then the average of these scores across detectors is computed.
  - *Robust rank aggregation.* This method was also recently used in [94]. It can identify the detectors whose ranking is consistently better than expected under the null hypothesis of uncorrelated inputs and assign a significance score to each detector.
  - *P-norm rank aggregation.* Do, Tran, and Venkatesh [120] introduce a near-optimal rank aggregation to minimize the disagreement with all ranks. Optimal minimization requires searching through a size $n!$ permutation space for n instances, which is unfeasible. Thus, this method looks to a small portion of data at the top to search through a smaller permutation space.
  - *Rank product aggregation.* It is a non-parametric statistical technique biologically motivated to detect differentially expressed genes in replicated microarray experiments. This method computes the aggregated anomaly score by the product of all rank values. After

ranking the observations by their rank product, the significance level is usually computed. For example, in Lopes et al. [119] the corresponding $p - values$ are computed under the null hypothesis that each ranking is uniformly distributed.

- **Decision template**: Decision template [122] is a robust supervised combination method that fuses the outputs of different models compared to a characteristic template for each class. Although it is a supervised method, the recent works by Wang, Mao, and Huang [90] and Wang and Mao [87] suggest a solution to use this method deriving pseudo anomaly labels from the decision profile assuming that the training set could have a 10% error.

2.2.2.3. *Meta-learning-wise combination*

Meta-learning-wise combination, also known as stacking, is a special combination method that uses a meta-learner (a new learner) to learn how to combine the first-level learners (base learners) [24]. The general idea of stacking (see Figure 2.11) is to use the original training data set to train the base learners and then to create a new data set (meta-dataset) using these outputs as new features.

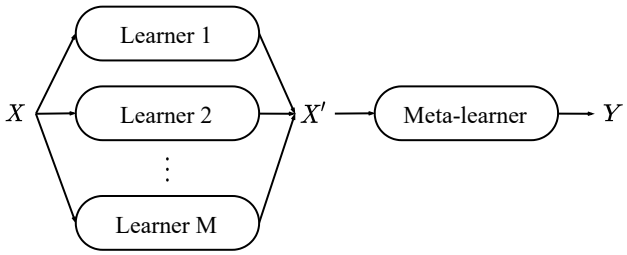


Figure 2.11: Schematic diagram of a stacking ensemble.

Three different strategies can be followed to create the meta-dataset considering some old and recent stacking ensembles. One option is to regard the outputs of base learners as the input features of the new training dataset, and if the meta-learner requires so, to consider the original labels as the annotation of the newly furnished training dataset [7, 93, 46, 9]. Another option is to extend the original features regarding the outputs of base learners as additional features [1, 46, 85]. The last option is to extract new features from the base learning outputs and then to regard these new ones to create the meta-dataset [6]. In addition, based on the learning paradigm used to construct first-level learners and meta-learner, three stacking ensembles types can be distinguished:

- **Supervised stacking**: this method uses supervised learning models as first-level base learners and the meta-learner. This is the most common stacking structure and one of the best ensemble techniques in classification

tasks. However, as we already mentioned, obtaining a labeled data set in anomaly detection tasks is hard to accomplish in practical scenarios. Therefore, very few works have been proposed in this direction in recent years Vanerio and Casas [7], Tama et al. [9], and Ouyang et al. [1].

- **Semi-supervised stacking**: in this second case, stacking uses unsupervised or semi-supervised first-level learners and a supervised meta-learner. It is helpful to use unsupervised base learners to detect never-seen-before anomalies, and to train a supervised meta-learner to combine the outputs and ultimately reducing the rate of false positives. Recent use cases adopting semi-supervised stacking ensembles can be found in [46] and [85].
- **Unsupervised stacking**: this third stacking method relies on unsupervised or semi-supervised learners for the implementation of first-level learners and the meta-learner. Due to the complexity of obtaining labeled data, it is intricate to use unsupervised algorithms as meta-learners to create an unsupervised stacking ensemble. However, despite its complexity, Zhou et al. [93] introduce a completely unsupervised stacking ensemble by proposing a novel meta-learner referred to as *Maximum Likelihood Estimation* (MLE).

2.2.2.4. Fusion of multiple combination methods

It is possible and even desirable to use multiple combination methods within a multilevel ensemble. One of the main advantages of merging several combination methods is that it can better balance the bias/variance trade-off by leveraging the strengths of each combination method [71]. Furthermore, this technique eases the combination of heterogeneous detectors learned from different task formulations (supervised, semi-supervised, unsupervised) or with different outputs (crisp, or anomaly scores in diverging scales). From the unsupervised perspective, there are some well-known combination methods that hinge on fusing multiple ensemble decisions:

- *AOM/MOA*: Introduced by Aggarwal and Sathe [71], *Average of maximum* (AOM) and *Maximum of average* (MOA) are two well-known bi-level ensemble fusion methods. In the first stage of AOM, the M base detectors of the ensemble are distributed into M/q buckets of q components, and each bucket is combined by the maximum (and by average in the case of MOA). In the second step, the M/q resultant outputs are fused by averaging (with the maximum in MOA). Recently, two new variants based on dynamic ensembles have been presented: *LSPC_AOM/LSPC_MOA* [103] and *ADAHO_AOM/ADAHO_MOA* [104].
- *SELECT*: Rayana and Akoglu [94] suggest an alternative two-level ensemble that learns heterogeneous base learners for which the best

are selected and combined by seven different combination strategies. Then, the selection process is carried again to select the outputs of the best combination methods. Finally, inverse rank aggregation is used to fuse the selected results, returning a unified rank list ordered from most to least anomalous.

- *DELR*: Zhang et al. [48] propose a bi-level ensemble, named DELR, for unsupervised anomaly detection. In the first level, for each detector, inverse rank is used to compute new anomaly scores, whereas in the second stage, a weighted average is applied. Weights are computed based on the intermediate scores and rank information.

Note that the multilevel fusion of combination methods allows generating an infinite number of different ensemble structures, such as those proposed in supervised [1] and semi-supervised formulations of the anomaly detection problem [113, 111, 81, 3].

2.2.2.5. The best base learner

Choosing the best base learner is not a combination method by itself. Nevertheless, there are cases, as in some sequential ensembles or dynamic ensembles, where it makes sense to select a single base learner. In some sequential ensembles, as in Zhang, Li, and Chen [73], the best base learner(s) is trained in the last iteration (i.e., on the pruned data set). Besides, in dynamic ensembles, the anomaly scores associated to some instances can be computed by a unique base learner. At this point, two different cases should be noted. The first case is when for each instance, only one base learner (the best one) is selected [88]. In the classification domain, this strategy is also known as dynamic classifier selection. The second case, adopted in recent dynamic ensembles [87, 90, 92, 91], is to determine, for each instance, which is the best option, either by select the best detector or by choosing a set of detectors. In the latter case, one of the combination techniques described above is often implemented to combine the outputs of the selected base models.

2.2.3. Learner selection techniques

Selection techniques are an optional but very effective step to increase the performance and robustness of ensembles. The goal of selection techniques is to choose the optimal subset of learners by selecting the best performing learners while preserving the ensemble's diversity. Moreover, it should be noted that some combination techniques, particularly those based on weighting, are closely related to selection methods. All of them are based on the same criteria: to assign each learner a higher or lower weight. When a learner is assigned a very low weight, it effectively reduces to a deletion

of the base learner from the ensemble (pruning). This section overviews different categories in which learner selection techniques can be organized, starting by inspecting the literature related to pruned ensembles, and ending with a review of dynamic ensembles. Figure 2.12 illustrates the taxonomy of learner selection methods that is followed throughout this literature analysis.

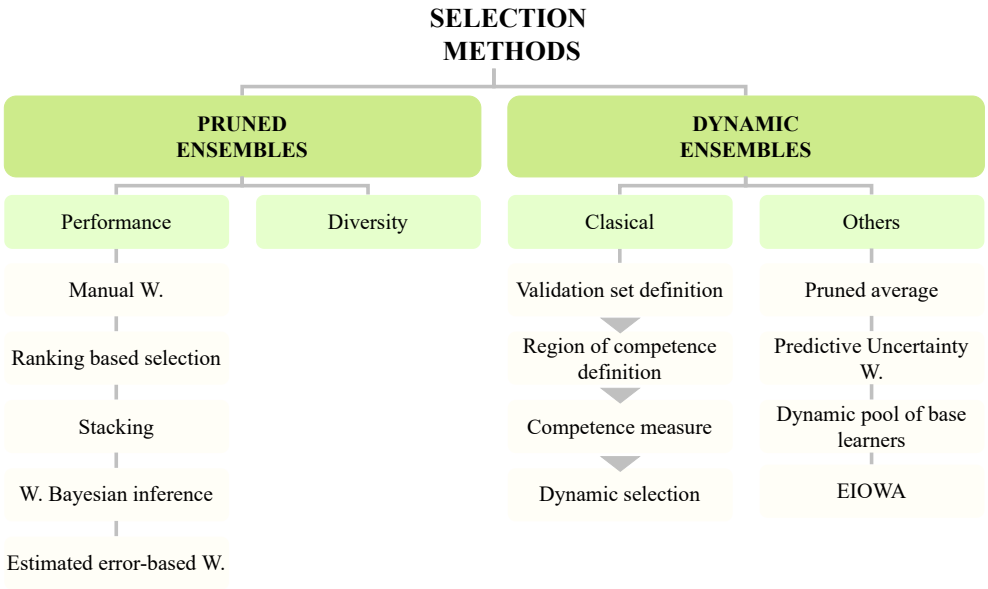


Figure 2.12: Taxonomy of learner selection methods for ensemble learning.

2.2.3.1. *Pruned ensembles*

Usually, pruned ensembles aim to maximize either performance or group diversity. However, due to the absence of ground truth, all analyzed works are focused on optimizing the performance of the pool of base learners. Specifically, we can distinguish them into three different weighting and selection techniques: manual weighting, ranking-based selection, stacking and weighted Bayesian inference.

- *Manual weighting* is the oldest and most straightforward option. However, it can be also regarded as the most complex approach in practice, as it implies establishing the weights of the base learners manually based on domain knowledge. However, not many practical cases afford domain knowledge that can be used for this purpose. To facilitate the analysis process and create efficient anomaly ensembles, Xu et al. [107] develops a complete toolkit for ensemble construction and visualization.
- *Ranking based selection*: Rayana and Akoglu [94] present SELECT approach to build selective anomaly ensembles. SELECT assumes

that there are inaccurate detectors in the ensemble, and provides two unsupervised orthogonal selection strategies – vertical and horizontal selection – to discard inaccurate detectors. The vertical selection strategy exploits correlation among the results, removing those detectors that do not agree with the pseudo ground truth. At the same time, the horizontal approach uses order statistics to filter out far-off results, eliminating the detectors that provide inaccurate ranking (compared to the others) of the target anomalies.

- *Stacking*: The use of another machine learning algorithm (meta-learner) to learn to combine diverse base learners' outputs is a common approach to effectively weigh and select the most accurate base learners. As already introduced in Section 2.2.2.3, several stacking-based proposals have been proposed for supervised and semi-supervised tasks, being only one work reported in [93] where this approach has been explored for unsupervised problems due to the lack of ground truth.

- *Weighted Bayesian inference*: most ensembles proposed for process monitoring [100, 99, 98, 89, 101, 76] use a weighted average approach as their combination method. These works combine the control limits instead of the anomaly scores, so the resulting output of the fusion is a threshold instead of an anomaly score. Although the application of this combination and weighting method has been so far specific of process monitoring applications, its use can be extrapolated to any other domain.

- *Estimated errors-based weighting*: Rayana, Zhong, and Akoglu [105], in their ensemble proposal, introduce a novel unsupervised weighted aggregation method based on estimated errors of the base detectors. In concrete, they examine unsupervised agreement rates for all possible pairs of base detectors to estimate their error rates and thereby, compute weights for their aggregation.

2.2.3.2. Dynamic ensembles

As opposed to pruned ensembles, dynamic ensembles select the best subset of base learners on the fly for each new test instance. The term *dynamic* highlights the adaptability of the ensemble structure to process each incoming instance, selecting the best subset of base learners for each in contrast to static ensembles and pruned ensembles, where an identical subset of the base learners within the ensemble are used for every new query instance.

- **Classical dynamic ensembles**, also known as dynamic classifier selection or dynamic ensemble selection, are increasingly popular in the last years, having been applied to a variety of supervised [2, 114], semi-supervised [87, 90, 91, 92, 88] and unsupervised anomaly detection problems[103,

104]. As shown in Figure 2.13, the construction of these ensembles consists of four fundamental steps: i) the definition of a labeled validation set; ii) the determination of the region of competence; iii) the establishment of a competence measure; and iv) the choice of the selection criterion.

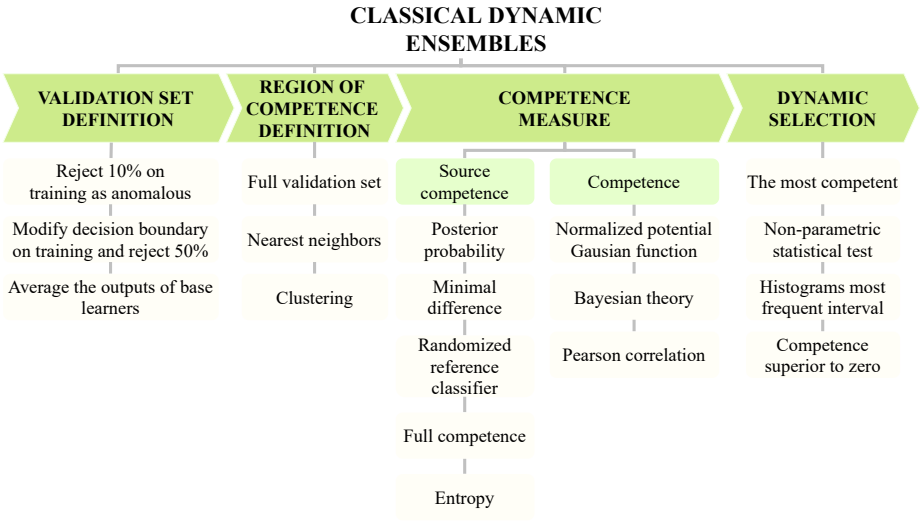


Figure 2.13: Steps of classical dynamic ensembles.

i) *Definition of a labeled validation set*: dynamic ensembles require a labeled validation set VS to compute the competence of the learners. Some related works [88, 114, 2, 90] distinguish explicitly between training dataset and validation dataset. In contrast, in [91, 92, 87, 103, 104], the training dataset is used as the validation dataset. Due to the absence of labels, most semi-supervised [87, 90, 91, 92] (excluding [88]) and unsupervised [103, 104] proposals suggest different techniques to generate the pseudo labels: a rejection rate of 10% of the training data to be declared as anomalous, the modification of the decision boundary learned after training, the rejection of 50% of the training samples, or an averaging of the outputs of the base learners. These pseudo labels are only used to select the best base learners, but are not part of the final result.

ii) *Determination of the region of competence*: the competence of a detector defines how effective or good it is at detecting or classifying instances. Then, the competence region RC^{test} concerning a test instance can be defined as a subset of the validation set VS which is used to evaluate the competency of the detectors, by classifying the test instance. This step identifies the similarity between the known instances in the validation set and the given observation. Methods for determining the region of competence proposed by the most recent publications can be classified into three groups: the use of the entry

validation set [88, 2], nearest neighbors-based [91, 114, 92, 103, 104] and clustering-based criteria [87, 90].

iii) *Competence measure*: this step defines the criterion to measure each model's performance or competence. Alves Ribeiro et al. [114] proposes to measure the competence of a detector as the true positive rate and true negative rate obtained by the model in the region of competence. This is a supervised dynamic selection approach and for this reason, there is no uncertainty of the correctness of the labels of the region of competence. To deal with this uncertainty and make model selection more reliable, the remaining supervised [2] and semi-supervised works [88, 87, 90] compute the detector's region of competence. This is done by first calculating the detector's local competence using assorted techniques, such as the posterior probability [91, 92, 90], minimal difference measure [88], randomized reference classifier [87], full competence measure [88] or the entropy measure [88]. Secondly, they calculate the global competence by the normalized potential Gaussian function [2, 87, 90, 123] or by the Pearson correlation [103, 104].

iv) *Selection*: Once the measure of competence of each base learner has been computed, it is necessary to establish the selection criterion. Several strategies have been reported for this purpose, from the selection of the most competent detector [114, 88, 104], the use of non-parametric statistical tests [87, 90, 91, 92], histograms [103] or the retention of learners whose competence is above a threshold [2].

- **Other dynamic ensembles**: in the literature, some proposals meet the definition of a dynamic ensemble, but are not referred to as such. The following describes the most recent proposals that could also be classified as dynamic ensembles:
  - *Pruned average*: This pruning method, introduced by Aggarwal [115], can be considered as a simple but effective dynamic ensemble, where the base learners with low anomaly scores are removed before the combination phase. Pruning can be done, for example, by using an absolute threshold imposed to the anomaly score, or by choosing the k best models for each data point.
  - *Predictive uncertainty-based weighting*: in this self-adapting weighting strategy proposed by Wang, Mao, and Huang [82], the base learners of the ensemble are predictors, whose outputs are weighted and combined before the overall detection of the anomaly is produced. The weighting procedure is based on predictive uncertainty, in which base learners' influence is automatically adjusted by their predictive variance.

- *A dynamic pool of base learners*: Bose et al. [112] propose an ensemble that dynamically adds or removes base learners based on the majority voting results. If an instance is detected as an anomaly, to reduce possible false positives, two new base learners are added to the ensemble. If the instance is still detected as an anomaly, an anomaly is reported. Nevertheless, if an inlier instance is detected, the two base learners are removed from the ensemble.
- *Exponential induced ordered weighted averaging (EIOWA)*: this weighting strategy was introduced by Yager and Filev [124] and used for the first time in an outlier ensemble by Parhizkar and Abadi [116]. It can be seen as a variation of weighted average and as a dynamic selection strategy. The weights change according to the order-inducing values of their associated base classifiers. The order-inducing values may be defined differently, e.g., based on detector performance or diversity. However, in these two works, the order is governed by the anomaly score produced by the base learners inside the ensemble.

2.2.4. Types of ensembles

Several other criteria can be used to classify ensembles according to their components [23, 91], usually related to the methods selected to perform the first two steps of the ensemble construction process. A summary of these classification criteria is shown in Figure 2.14, whereas a more detailed explanation of each type of ensemble is given next:

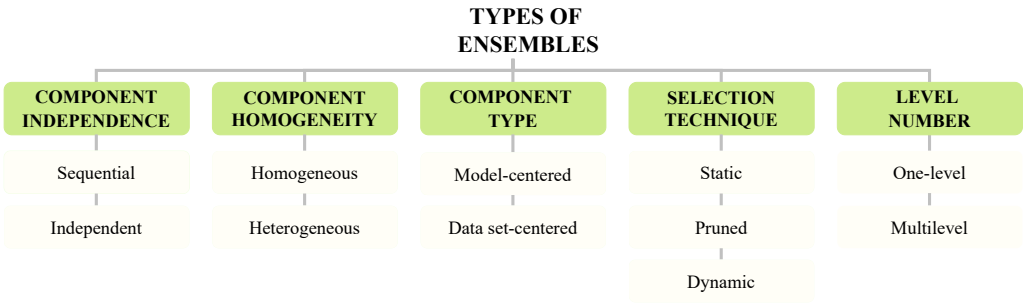


Figure 2.14: Taxonomy of the different types of ensembles.

CATEGORIZATION BY COMPONENT INDEPENDENCE: This is the most common and it consists of classifying the ensembles according to the dependence or independence of the base learners [23]. In this case, there are two types of ensembles: sequential ensembles and independent ensembles. Sequential ensembles are those that apply one or more base learner(s) sequentially to all or part of the data, so that at each iteration, either the dataset or the model changes. In other words, the execution of each model depends

on the result from previous executions. Most of these ensembles require calculating the error made in intermediate steps. Because the absence of the ground truth and the need for computing the intermediate error, sequential ensembles are not usually applied to the detection of anomalies. Even so recently several successful proposals have been made such as [73], [104] and [81] into others. On the contrary, independent ensembles learn independently or in parallel a set of different base learners, and then combine them to obtain a more robust result. Diverse models provide different perspectives on various aspects of the data. Hence, combining these perspectives can provide more robust results that do not depend on specific parameters of the algorithm or the dataset itself.

CATEGORIZATION BY COMPONENT HOMOGENEITY: An alternative way for ensemble categorization is based on the homogeneity of the base learners, differentiating between homogeneous and heterogeneous ensembles [23]. Homogeneous ensembles use a single learning algorithm. In most cases, except for some sequential ensembles, it is necessary to use an additional technique to increase the number of base learners and their diversity. Heterogeneous ensembles use multiple learning algorithms. These ensembles allow detecting anomalies using different strategies and providing higher diversity among base components. Additionally, they can be complemented with other techniques to further increase their degree of diversity.

CATEGORIZATION BY COMPONENT TYPE: Another way of classification is that presented by Aggarwal and Sathe [23]. In this case, they classify the ensembles according to the type of components, discriminating between model-centered and dataset-centered ensembles. Model-centered ensembles are those that combine different models trained from the same dataset. Particular cases are the combination of homogeneous models instantiated with different hyperparameter values or with distinct seeds. Data set-centered ensembles are those which use models learned from base algorithms trained with different subsets of data. Such subsets can be created from different samples of instances or different attribute subsets.

CATEGORIZATION BY SELECTION TECHNIQUES: With the emergence of dynamic ensembles, Wang and Mao [91] proposed a new ensembles categorization according to the selection strategy, yielding three types of ensembles: static, pruned and dynamic ensembles. Static ensembles are traditional ensembles where no base learner selection strategy is applied, i.e. all base models are combined. Instead, pruned ensembles perform the selection of the most competent algorithms only once, usually during the training phase. After selecting the most competent subset of learners, the components of the ensemble remain fixed. Finally, in dynamic ensembles the structure of the ensemble varies for each new incoming query instance.

These flexible ensemble structures allow efficiently choosing the appropriate subset of base learners for each incoming example.

CATEGORIZATION BY LEVEL NUMBER: another alternative criterion can be proposed to categorize ensembles used in the literature according to the number of levels of the ensemble. Based on this criterion, we distinguish two types of ensembles: one-level ensembles and multilevel ensembles. To begin with, one-level ensembles follow traditional the ensemble architecture shown in Figure 2.8. In this case, the ensemble comprises a set of base models, a single combination method, and optionally a selection step. Multilevel ensembles, however, combine the base learners in several phases using the same combination method in each level or, instead, by using different combination methods. This spans a myriad of ensemble architectures that can be created; among them, the most popular multilevel approach is to construct an ensemble that blends together other ensembles featuring different combinations and/or diversity induction methods.

2.2.5. Challenges of anomaly detection ensembles

As it occurs with the detection of anomalies, the absence of ground truth and the computational challenges derived from stream processing jeopardize the construction of anomaly ensembles in several phases reviewed heretofore, which are further discussed below:

1. *Challenges in the algorithm(s) selection*: since this first step of the ensemble construction process aims to choose the most appropriate anomaly detection algorithm for each problem, this step undergoes the same challenges of anomaly detection already introduced in Section 2.1.

2. *Challenges in base learner augmentation and diversity generation*: The main challenge when inducing diversity in an anomaly detection ensemble is the complexity of measuring the performance and the diversity among base learners in the absence of ground truth. Additional difficulties arise in real-time processing scenarios, such as the complexity of generating different subsets of instances (especially when data points are correlated through time) or the constraints in creating subsets of features, since some variables may be useless or even disappear over time, while new predictors may eventually emerge from the processed data streams.

3. *Challenges in component(s) selection*: Selection techniques used in classification ensembles mainly rely on available ground truth to measure the classification performance and/or the diversity of the base learners. Therefore, the absence of the ground truth that frequently prevails in

real-world anomaly detection problems handicaps the adaptation of these methods to address the detection of anomalies.

4. *Challenges in the selection of the combination method*: Some of the classical combination methods used in classification, such as mean, maximum, or majority, do not need any knowledge about the ground truth of the data over which the anomaly detection task is formulated. Therefore, they can be used in unsupervised anomaly ensembles without any modification. However, other effective combination techniques in classification ensembles, such as weighted mean, weighted median, weighted voting and stacking, usually require ground truth and finite datasets, so their adaptation to unsupervised anomaly detection and online processing remain unsolved in the current literature.

Part II

CONTRIBUTIONS

This part presents the original contributions associated with this Thesis. The first chapter covers the advances made towards solving the shortage of public software. The second chapter of this second part has two objectives: on the one hand, the proposal of a methodology to generate new anomaly detectors from online time series prediction algorithms, which is proposed to stimulate the creation of new online time series outlier detection proposals. On the other hand, the second chapter presents a new ensemble-based anomaly detector that results from the application of the aforementioned methodology, which is shown to improve the performance of current detectors by reducing the rates of false positives and negatives with respect to other modern outlier detection alternatives.

3

OTSAD: A PACKAGE FOR ONLINE TIME-SERIES ANOMALY DETECTORS

Online *time series outlier detection* (TSOD) is a young research line which is gaining interest within many domains in recent years. Compared to classical anomaly detection, online time series outlier detection is a more challenging task for many reasons: these problems do not have entire data set available, they must be able to consider the arrival time, and they have to deal with stationarity or non-stationarity of the data. Due to these constraints, there is still a long road ahead plenty of challenges and research opportunities for the research community working on TSOD. Indeed, the number of proposals has increased over the years. However, most tools and algorithms for TSOD are currently available via paid subscription, with very few free software proposals. The absence of open-source implementations hinders future research and proposals since it is difficult to compare and evaluate the performance and features of the literature detectors with new ones.

To cover this research niche, this chapter presents the *otsad*[1] R software package available at the Comprehensive R Archive Network (CRAN). In contrast to other packages used for online TSOD, *otsad* is easier to use and includes six powerful online TSOD algorithms for univariate time-series that cover different current needs, such as online processing and the ability to work in both stationary and non-stationary environments. It also provides a new original false positive reduction algorithm, called `ReduceAnomalies` capable to be used with all the algorithms in *otsad*. Finally, it also includes some advanced functionalities, such as NAB [125] detector measurement technique and a visualization function.

The rest of the chapter is organized as follows. The next section reviews the state of the art in online TSOD and the available open-source packages. Then Section 3.2 provides detail about the implementation and functionalities of the developed package. Next, a benchmark is performed in Section 3.3. Section 3.4 describes an example of the use of *otsad*. Finally, Section 3.5

1 https://CRAN.R-project.org/package=otsad

concludes the chapter and sketches some future lines of research inspired by the package herein presented.

3.1. BACKGROUND

Online TSOD is a novel research line and challenging for several reasons: the entire dataset is not available; time correlations must be considered, and stationarity or non-stationarity of the data must be handled. Hence, despite classic anomaly detection methods, which have been widely studied and have extensive open-source software available, there is little documentation [50, 28] and open-source software available for this purpose. In the following sections we provide some background about the state of the art in online TSOD (Section 3.1.1) and open-source libraries (Section 3.1.2).

3.1.1. Online time-series outlier detection

Two learning paradigms have emerged for online TSOD in recent years: model retraining and incremental learning. Model retraining retrains the model over a window of the most recent data each time new data arrives. On the other hand, incremental learning updates the model and its parameters each time new data comes.

3.1.1.1. Model retraining

Perhaps the most intuitive way to apply an outlier detection algorithm in streaming is retraining it by time windows each time new data arrives. However, this may not be the best option depending on the selected technique. It may lead to high computational costs and being unavailable to respond timely. In addition, by training the model over the most recent data window, relevant information from the past can be lost. Because of this, not many works embrace this strategy. Nonetheless, we describe the most representative ones that follow a retraining strategy.

To begin with, Basu and Meckesheimer [126] proposed two methods for time series outlier detection and removal. Both are prediction model-based detectors. The first one, *two-sided median method for cleaning noisy data*, determines whether a particular data point is an outlier by computing the median value of a neighborhood of data points over a non-overlapping sliding window considering past and future data. This method has an overlay of a few samples, as it needs to wait for future data. The second method, named *one-sided median method for cleaning noisy data*, is designed for fully-stream processing. In this case, they only look at the first difference of the observed series to compute the median value of a neighborhood of data points over an overlapping window. The window width and threshold values are determined arbitrarily (for example, the percent deviation from the mean) and may

depend on the signal. However, they suggest that it is not always possible when dealing with a non-stationary process, so it seems more appropriate for stationary data. Moreover, although all examples have been done in univariate time series, the proposed techniques can be also used to detect anomalies over multivariate streaming time series.

More recently, Siffer et al. [127] introduce two prediction model-based approaches to detect outliers in streaming univariate time series using past data and retraining. Both are based on *Extreme Value Theory*, do not require handset thresholds, and do not make an assumption on the distribution. The two algorithms are SPOT for streaming data having any stationary distribution and DSPOT for streaming data that can be subject to concept drift, i.e., nonstationary.

Finally, Zhou et al. [128] propose retraining an ARIMA model within a sliding window to compute the prediction interval and detect anomalies. In this manner, the parameters are refitted each time that the window moves a step forward. As it is well known, ARIMA is designed to work in stationary time series, and it can be applied in univariate as well as in multivariate time series. Moreover, this method can be classified as prediction model-based.

3.1.1.2. Incremental

Two types of incremental algorithms can be distinguished: i) adaptive algorithms, which do not require historical data; and ii) window-based algorithms, which store the history of the most recent data to update the model:

- *Adaptive methods* update the model's statistics whenever a new value arrives over the stream, without retaining any historical value. These methods require little memory and perform very fast. However, there are few adaptive online TSOD methods due to their design complexity. The first work is the *Probabilistic Exponentially Weighted Moving Average* (PEWMA) proposed by Carter and Streilein [64]. This algorithm is a probabilistic method that hinges on EWMA at its core, and dynamically adjusts its parameters based on the probability of the given observation. It is an estimation-based outlier detector for univariate TSOD, and it only performs reliably over stationary time series.

  Later, also based on EWMA, Raza, Prasad, and Li [62] introduced *Shift-Detection based on EWMA* (SD-EWMA) and *Two-Stage Shift-Detection based on EWMA* (TSSD-EWMA). SD-EWMA is an adaptive method that uses an *Exponentially Weighted Moving Average* (EWMA) model-based control chart to detect covariate shift-point in univariate or multivariate stationary time-series. In addition to SD-EWMA, they also proposed TSSD-EWMA, which can work over non-stationary time series. The algorithm works in two phases. In the first phase, it applies SD-EWMA, while in the second phase, it checks the authenticity of

the anomalies using a Kolmogorov-Smirnov test to reduce the rate of false alarms. It should be noted that this method has an overlay of a few samples, as it needs to wait for future data to perform the false positive reduction.

- *Window-based methods* embody the most used strategy to learn from data incrementally. These methods use a sliding window to maintain and update the model over the most recent data. Several works have been proposed based on this online learning approach. Xu, Kersting, and Ritter [65] present a prediction model-based non-parametric Bayesian method (ATOS) to detect anomalies in stationary univariate time series data. This method is based on the student-t process to learn the dynamics of the time series with submodular optimization-based kernel selection and thus, identify potential anomalous behavior. Similarly, Xu, Kersting, and Ritter [66] proposed OLAD, a prediction model-based online non-parametric Bayesian method based on the student-t process and stochastic gradient descent to learn the underlining dynamics of the time series and detect outliers in univariate stationary time series. Further along this line, Angiulli and Fassetti [67] introduced the Stream Outlier Miner (STORM) for univariate nonstationary time series. This nearest neighbors method is based on density and detects outliers that are abnormal concerning data within the current window. Likewise, Burnaev and Ishimtsev [20] proposed *Conformal KNN Anomaly Detector* (KNN-CAD), a density-based model-free anomaly detection method for univariate time series. This method adapts itself to non-stationary data, and provides probabilistic abnormality scores based on the conformal prediction paradigm. Based on a similar approach, Ishimtsev et al. [68] presented the so-called *KNN Lazy Drifting Conformal Detector* (KNN-LDCD), a variation of KNN-CAD that also hinges on density and which is suited for univariate non-stationary TSOD. Both methods use different measures for dissimilarity and conformity calculation. In recent years, Ahmad et al. [19] introduced Numenta HTM, a prediction model-based online outlier detection algorithm for non-stationary univariate time series based on an online *Hierarchical Temporal Memory* (HTM). Differently from the above, Muthukrishnan, Shah, and Vitter [57] presented a histogramming-based algorithm for outlier detection for both univariate and multivariate non-stationary time series.

3.1.2. Open source software for online TSOD

The existing public software tools for online TSOD consists of nine libraries implemented in different programming languages, including C++, Java, Matlab, Python and R. Specifically, such publicly available packages are

libspot [18], STORM [15], SCREEN [16], SCR [17], spirit [14], NumentaTM [19], KNN-CAD [20], CAD-OSE [21], and AnomalyDetection [22]. These available software packages and their source code repositories are summarized in Table 3.1.

Table 3.1: Summary of the publicly available open-source software.

Library	Lenguage	Code
libspot [18]	C++	https://github.com/asiffer/libspot
STORM [15]	Java	https://github.com/Waikato/moa/tree/master/moa/src/main/java/moa/clusterers/outliers/Angiulli
SCREEN[16]	Java	https://github.com/zaqthss/sigmod15-screen
SCR [17]	Java	https://github.com/zaqthss/sigmod16-scr
spirit [14]	Matlab	http://www.cs.cmu.edu/afs/cs/project/spirit-1/www/
NumentaTM [19]	Python	https://github.com/numenta/NAB/tree/master/nab/detectors/numenta
KNN-CAD [20]	Python	https://github.com/numenta/NAB/tree/master/nab/detectors/knncad
CAD-OSE [21]	Python	https://github.com/smirmik/CAD/
AnomalyDetection [22]	R	https://github.com/twitter/AnomalyDetection

The proposed *otsad* library is implemented in R. R is a language and free software environment for statistical computing and graphics. It is one of the most popular languages used among statisticians and data miners to develop statistical and data analysis software. CRAN *"Comprehensive R Archive Network"* is a network of FTP and web servers around the world that store identical, up-to-date versions of code and documentation for R.

Nowadays CRAN servers include a lot of influential packages for outlier detection domain which are very closed to the central learning task of this Thesis. *outliers* [129], *SMLoutliers* [130], *univOutl* [131] and *OutliersO3* [132] are some of the packages available for anomaly detection. There are few more for outlier detection in time-series i.e. *tsouliers* [133], *qicharts* [134] and *washeR* [135]. We also found few complete packages for imbalance problems i.e. *smotefamily* [136], *imbalance* [137], *ebmc* [138], and multi-imbalance [139]. There are also multiple packages for change-point detection i.e. *ocp* [140], *ecp* [141], *bulletcp* [142] and *MFT* [143]. However, only one R package deals with online TSOD, namely, AnomalyDetection [22], and it is no longer available in CRAN.

Based on this short study, it is fair to conclude that very few packages are available for online TSOD regardless of the programming language. Moreover, is important to note that all of them implement a single proposal, so there are currently only nine algorithms available for online TSOD. Therefore, it is essential to provide more algorithms and software tools to implement and evaluate them systematically.

3.2. OTSAD: IMPLEMENTATION AND FUNCTIONALITIES

Otsad is the first R package that collects a set of online TSOD. This package provides the first open source implementations of a set of up-to-date and powerful detectors (from Table 3.2 detectors 1-4). Moreover, to increase the

diversity of the initial set of detectors and ease its use to R developers, *Otsad* also includes two of the best detectors in NAB competition (detectors 5-6 from Table 3.2). It is also worth noting that all functionalities with the exception of CAD-OSE are developed in the R language. There is no documentation for the CAD-OSE algorithm, and due to the sensibility of some python methods used in its original code, it was kept in python and adapted to python3 and its use in R. All developed functions and functionalities are thoroughly documented and bundled alongside the package, giving the user detailed guidelines, such as recommended parameters values and running examples.

This package's main functionalities and implementation details are described in the following sections. The implemented TSOD algorithms are introduced in Section 3.2.1. Then, the detector measurement technique is described in Section 3.2.2. Section 3.2.3 describes the proposed false positive reduction technique. Finally the datasets and the graphic tool are introduced in Section 3.2.4 and Section 3.2.5, respectively.

3.2.1. Anomaly detection algorithms

Implementing algorithms based on their description in the article where they are first proposed is not always an easy task. Frequently the formulation is complex yet poorly explained. Sometimes even part of their algorithmic steps are missing. We have chosen four of the algorithms reviewed in the previous section to create this library: PEWMA [64], SD-EWMA [62], TSSD-EWMA [62], and KNN-LDCD [68]. This selection has been made based on their properties and interpretability. All these selected algorithms are incremental since, concerning model retraining, they generally require less computational resources and retain temporal information better than other counterparts. We choose heterogeneous algorithms covering different types of learning, adaptive and window-based, as well as the ability to process stationary and non-stationary time series. In addition, to increase the diversity of the initial set of detectors and facilitate their use to R developers, we also include two of the detectors that have performed best in the NAB competition: KNN-CAD [20] and CAD-OSE [21].

In particular, this package implements and documents the set of detectors listed in Table 3.2. On the one hand, the first three detectors are adaptive, while the next three are window-based. On the other hand, the first two algorithms can be used with stationary data. In contrast, the other four are suitable for use with non-stationary data. Each of these algorithms were implemented to work in two different scenarios: i) *classical processing*, used when the complete data set (train and test) is available, and ii) *incremental processing*, used when the complete dataset is not available but produced continuously over time. In addition, it should be noted that given their algorithmic similarity, the KNN-LDCD and KNN-CAD detectors have

been implemented in the same function `CpKnnCad` and distinguished by the `ncm.type` parameter.

Table 3.2: Features and functions of the anomaly detectors available in the `otsad` package.

Online Anomaly Detectors	Incremental	Stationarity	Functions	
			Classic processing	Online Processing
PEWMA [64]	Adaptive	Stationary	`CpPewma`	`IpPewma`
SD-EWMA [62]	Adaptive	Stationary	`CpSdEwma`	`IpSdEwma`
TSSD-EWMA [62]	Adaptive	Non-stationary	`CpTsSdEwma`	`IpTsSdEwma`
KNN-LDCD [68]	Window	Non-stationary	`CpKnnCad(ncm.type= "LDCD")`	`IpKnnCad(ncm.type= "LDCD")`
KNN-CAD [20]	Window	Non-stationary	`CpKnnCad(ncm.type= "ICAD")`	`IpKnnCad(ncm.type= "ICAD")`
CAD-OSE [21]	Window	Non-stationary	`ContextualAnomalyDetector`	`ContextualAnomalyDetector`

3.2.2. Detector measurement technique

In recent years, Lavin and Ahmad [125] introduces a novel scoring method (NAB) for measuring the performance of online TSOD algorithms. This measure is specifically designed for the evaluation of online TSOD. Besides traditional detection statistics, such as precision, recall or f1-score, the NAB score also accounts for the time elapsed until the anomalies are detected.

Specifically, the NAB scoring method works as follows: first, a time window is centered on each ground truth anomaly in order to determine the corresponding labels of detected outliers. A detected anomaly is labeled as true positive (TP) when it falls inside the window, and as false positive (FP) when it falls outside. A false negative (FN) label is considered when there are no detected anomalies inside the window. The window size is calculated as the ratio between 10% of the number of observations and the number of ground truth anomalies in the time series. Then, to account for the elapsed time until detection, this measurement technique resorts to different weights (W_{TP}, W_{FP}, W_{FN} for TP, FP, and FN, respectively). As such, FN and FP will be assigned a negative score, whereas the early detected positives will get a higher score than late ones. In this regard, the NAB scores establish three profiles, each with different weights summarized in Table 3.3: standard ($W_{FP} = W_{FN}$), reward low FP rate ($W_{FP} > W_{FN}$) and reward low FN rate ($W_{FP} < W_{FN}$). Finally, the total score is calculated as the sum of the scores assigned to each detected anomaly and the cumulative scores of missed anomalies. This score is then scaled as:

$$Score_{final} = 100\frac{Score_{detector} - Score_{null}}{Score_{perfect} - Score_{null}}, \tag{3.1}$$

where $Score_{perfect}$ is the score of a perfect detector, i.e., detecting all anomalies taking the maximum score. The $Score_{null}$ value is the score of a null detector, i.e., one that does not detect any anomaly (and no FP). It must be noted that the theoretical range of the NAB score is between $[-\infty, 100]$,

where value 100 corresponds to perfect detection. To conclude, a visual example of all procedures is shown in Figure 3.1.

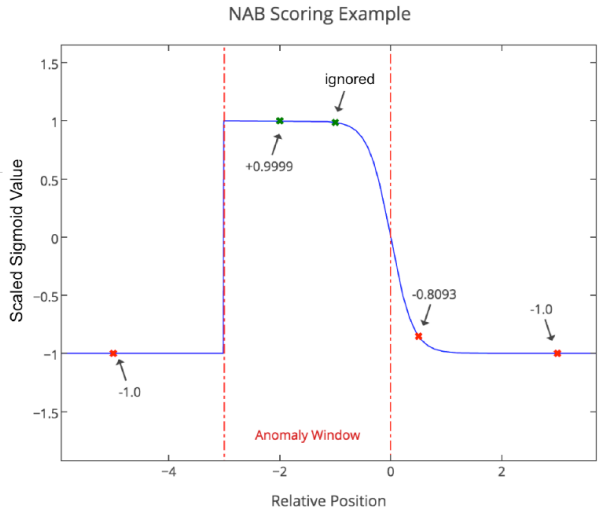


Figure 3.1: Scoring example for a sample anomaly window, taken from [125], which exemplifies how the scores of detected anomalies vary according to their relative position within the window.

Table 3.3: Label weights per profile.

	Standard	Reward low FP rate	Reward low FN rate
W_{TP}	1.0	1.0	1.0
W_{FP}	-0.11	-0.22	-0.11
W_{TN}	1.0	1.0	1.0
W_{FN}	-1.0	-1.0	-2.0

The *otsad* package implements this metric to measure the detectors' performance. For these objectives, there are three main functions implemented: `GetDetectorScore`, which calculates the detector score without normalization; `GetNullAndPerfectScores`, used to obtain the scores of the perfect and null detectors for the dataset; and `NormalizeScore`, which allows normalizing detector scores. Besides, the package incorporates an additional function (`GetNumTrainingValues`) to allow the user to obtain the number of instances used as a training set in NAB. The training set size is computed as 15% of the dataset size, and is fixed to 750 for datasets accounting for less than 5,000 data points.

3.2.3. False positive reduction technique

Some algorithms already included techniques to reduce false positives. These can be only applied to the above algorithms. For this reason, a novel algorithm

is implemented, referred to as `ReduceAnomalies`, aimed to reduce false positives, ensuring that it can be applied to all the detection algorithms developed inside the package.

Figure 3.2: Explanatory example of false positive reduction by `ReduceAnomalies` algorithm. Unfilled circles are inline points and colored ones are anomalies detected by a detector. The blue ones are those ones detected as true positives by `ReduceAnomalies` and the green ones, as they are inside the windows `ReduceAnomalies` detects them as false positives.

The `ReduceAnomalies` algorithm is inspired by the real-life situation where a time lapse exists between the time an alarm is triggered and the instant at which a corrective action is performed. In other words, the minimum time lapse between the first and the second alarm. The algorithm uses the number of processed data points between two detected anomalies to reduce the number of false positives. When the first anomaly x_t is detected, a new window of length w is created, with x_{t+1} as the starting point, and x_{t+w} as the ending point. For each newly detected anomaly, its relative position compared to the window is evaluated. Detected anomalies inside the window are excluded. If a detected anomaly is outside the window, it is then considered as a real anomaly and a new window is calculated in the same way as mentioned above. A visual example showing how `ReduceAnomalies` operates is shown in Figure 3.2.

3.2.4. Datasets

Otsad includes 51 of the 58 labeled one-dimensional time-series from different fields available in the NAB [125] repository, which are gathered in a number of groups or families:

- `artificialWithAnomaly`, comprising synthetically generated time series data with varying types of anomalies;
- `realAdExchange`, composed by online advertisement clicking rates, where the metrics are cost-per-click and cost per thousand impressions;
- `realAWSColudwatch`, built upon AWS server metrics;
- `realKnownCauses`, such as hourly registered taxi schedules in New York or CPU utilization;
- `realTraffic`: Real time traffic data from the Twin Cities Metro area of Minnesota (USA), with occupancy, speed, and travel time data from specific sensors;

- `realTweets`: A collection of Twitter mentions of large publicly-traded companies such as Google and IBM;

Each of the datasets included in *otsad* is composed of these three columns: `timestamp`, `value` and `is.real.anomaly`, the latter containing the labeled ground truth anomalies.

3.2.5. Visualization tool

The *otsad* package also implements two functions to visualize the obtained results. The first function, named `PlotDetections`, displays the detection results in an interactive graph. In addition to the detected anomalies, the `print.real.anomaly` and `print.time.window` parameters allow displaying the ground truth and the validation windows as defined in NAB. Also, other basic elements can be configured and customized, including the title, x-label, or y-label of the displayed axis. Moreover, if further customization of the chart is needed, it is possible to obtain an easily customizable ggplot object using the parameter `return.ggplot`. A live example can be found at this[2] GitHub repository.

The second graph functionality is integrated with the `GetDetectorScore` function. Through the `print` and `title` parameters, you can activate and define the title of an interactive graph that will show the scores assigned to each instance by the NAB metric. In addition to this information, you can also see the validation window, along with the false positives, true positives, and false negatives, distinguished by different colored dots. A live example can be found at this[3] GitHub repository address.

3.3. PERFORMANCE AND TIME EFFICIENCY BENCHMARKS

In order to choose a suitable algorithm for online TSOD, there are three main characteristics that need to be evaluated for each proposal: processing time, performance, and simplicity in the parameters' configuration. For this reason, several benchmarks have been performed to test the capabilities of the algorithms included in the *otsad* package. Results of these benchmarks are discussed in the following subsections:

3.3.1. Performance benchmark

A first benchmark is performed to determine which algorithm performs best over the datasets included in the developed package. To this end, two different aspects are analyzed. First, we explore which algorithm generalizes better across the considered datasets, for which only the best general

2 https://alaineiturria.github.io/otsad/KNN-CAD.html
3 https://alaineiturria.github.io/otsad/KNN-CAD-numenta.html

hyperparameter configuration of every detector is used. This general hyperparameter setting is very useful when the domain knowledge is poor. Second, the adjustment capability of the algorithms is evaluated. To this end, the best parameters settings for each dataset are considered. This case is useful when there is enough knowledge to adjust the detector to the dataset at hand.

As a first step, the detectors are applied to each dataset considering different parameter configurations (see Table 3.4). At this point, false positive reduction techniques are also applied. In the case of PEWMA, SD-EWMA and TSSD-EWMA, our algorithm with a one-day window size is used. For KNN-CAD, KNN-LDCD and CAD-OSE the reduction techniques proposed by their authors are applied. After that, the final score of each detector is calculated as the sum of the scores obtained over each dataset $Score^{d}_{Detector}$. Finally, the detector's score is normalized using max-min normalization. In the following Table 3.6, Table 3.7 and Figure 3.3, the two benchmark results are shown. The obtained best general parameters are summarized in Table 3.5.

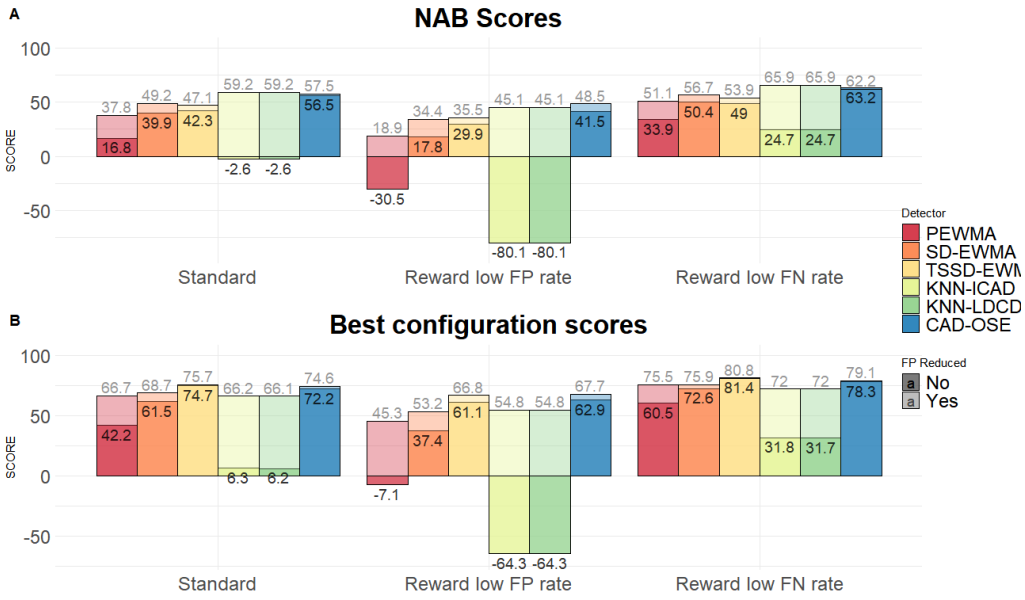


Figure 3.3: Performance benchmark. A) using the NAB scoring method, i.e. a single hyperparameter setting for all data sets. B) Using the best parameter setting for each dataset.

Table 3.6 and Figure 3.3-A evince that the top performing algorithms when configured with general parameters are KNN-CAD and KNN-LDCD, followed by CAD-OSE. Moreover, TSSD-EWMA performs worse than SD-EWMA with general parameters. In Table 3.7 and Figure 3.3-B, it can be observed that the algorithm with the highest adaptability is TSSD-EWMA, followed by CAD-OSE. Finally, when comparing the results obtained in Figure 3.3-A and

Table 3.4: Benchmark parameter settings.

Detector	Parameters	Value	Description
PEWMA	***alpha0***	{0.1, 0.2, ..., 1}	Maximal weighting parameter
	beta	{0.1, 0.2,..., 1}	Weight placed on the probability of the given observation
	l	{2, 3, ..., 8}	Control limit multiplier
SD-EWMA	***threshold***	{0.01, 0.02, ..., 0.1}	Error smoothing constant
	l	{2, 3, ..., 8}	Control limit multiplier
TSSD-EWMA	***threshold***	{0.01, 0.02, ..., 0.1}	Error smoothing constant
	l	{2, 3, ..., 8}	Control limit multiplier
	m	{100, 200, ..., 700}	Length of the subsequences for applying the Kolmogorov-Smirnov test
KNN-ICAD AND KNN-LDCD	***threshold***	{0.9, 0.95, 1}	Threshold used to determine if the current instance is an anomaly
	l	{10, 19, 30}	Window length
	k	{5, 21, 27, 31}	Number of neighbours to take into account
CAD-OSE	***max.left. semicontexts***	{3, 5, 15}	Number of semicontexts that should be maintained in memory
	max.active. neurons	{5, 15, 25}	Number of neurons of the model
	num.norm. value.bits	{3, 5, 9}	Granularity of the transformation into discrete values
	base.threshold	{0.70, 0.75, 0.76, 0.8}	Threshold used to determine if the current instance is an anomaly

Table 3.5: Best general parameters

Detector	Parameters	Standard	Low FP	Low FN	Reduced Standard	Reduced Low FP	Reduced Low FN
PEWMA	***alpha0***	0.9	0.9	0.9	0.8	0.8	0.8
	beta	0.1	0.1	0.1	0.6	0.1	0.3
	l	8	8	8	8	8	4
SD-EWMA	***threshold***	0.01	0.01	0.01	0.01	0.01	0.01
	l	7	8	7	6	7	6
TSSD-EWMA	***threshold***	0.01	0.01	0.03	0.01	0.01	0.01
	l	7	8	6	6	7	4
	m	300	300	600	600	300	600
KNN-ICAD AND KNN-LDCD	***threshold***	1	1	1	1	1	1
	l	10	10	10	19	19	19
	k	5	5	5	27	27	27
CAD-OSE	***max.left.semicontexts***	7	7	7	7	7	7
	max.active.neurons	15	15	15	15	15	15
	num.norm.value.bits	3	3	3	3	3	3
	base.threshold	0.7	0.7	0.7	0.7	0.7	0.7

Table 3.6: NAB scores obtained by detectors in the *otsad* package over the Numenta Anomaly Benchmark, using global-best hyper-parameter values for all datasets. The highest scores are highlighted in bold.

Detector	Standard	Low FP	Low FN	Reduced Standard	Reduced Low FP	Reduced Low FN
PEWMA	16.80	-30.50	33.90	37.80	18.90	51.10
SD-EWMA	39.90	17.80	50.40	49.20	34.40	56.70
TSSD-EWMA	42.30	29.90	49.00	47.10	35.50	53.90
KNN-ICAD	-2.60	-80.10	24.70	59.20	45.10	65.90
KNN-LDCD	-2.60	-80.10	24.70	**59.20**	45.10	**65.90**
CAD-OSE	**56.50**	**41.50**	**63.20**	57.50	**48.50**	62.20

Figure 3.3-B, it is straightforward to conclude that the performance of every algorithm is highly related to the parameter configuration of the detector and the false positive reduction techniques. Furthermore, it also exposes that the false positive reduction approach is effective towards improving the detection results. However, it is worth mentioning that all these algorithms, even the best ones, are far from obtaining perfect results, so better false positive reduction strategies and, thus, more robust detectors are still in need for prospective research efforts.

Table 3.7: NAB scores obtained by detectors in the *otsad* package over the Numenta Anomaly Benchmark, using the best hyper-parameter values found for each dataset. The highest scores are highlighted in bold.

Detector	Standard	Low FP	Low FN	Reduced Standard	Reduced Low FP	Reduced Low FN
PEWMA	42.20	-7.10	60.50	66.70	45.30	75.50
SD-EWMA	61.50	37.40	72.60	68.70	53.20	75.90
TSSD-EWMA	**74.70**	**61.10**	**81.40**	**75.70**	66.80	**80.80**
KNN-CAD	6.30	-64.30	31.80	66.20	54.80	72.00
KNN-LDCD	6.20	-64.30	31.70	66.10	54.80	72.00
CAD-OSE	72.20	62.90	78.30	74.60	**67.70**	79.10

3.3.2. Time efficiency benchmarks

In this section, the time efficiency of each detector is evaluated by two different benchmarks. On one hand, a first time efficiency benchmark analyses the execution time of the detection algorithms over the full length of every dataset, i.e. batch processing. For this study, synthetic datasets of different

lengths ($\{1000, 3000, 5000, \ldots, 29000\}$) are used. The number of training points is calculated using `GetNumTrainingValues` function, whereas all other parameters are configured using the best general parameters obtained in the previous performance benchmark. The outcomes of this first time efficiency benchmark is shown in Figure 3.4. On the other hand, the second time efficiency benchmark considers the online or incremental processing operation of the algorithms, measuring the execution time needed by each detector to process a unique value. For this aim, a synthetic dataset of 1000 points is used and the time needed to process each data instance is collected. The rest of parameters are configured as in the classical processing time efficiency benchmark. A summary of the results of this second time efficiency benchmark is shown in Table 3.8.

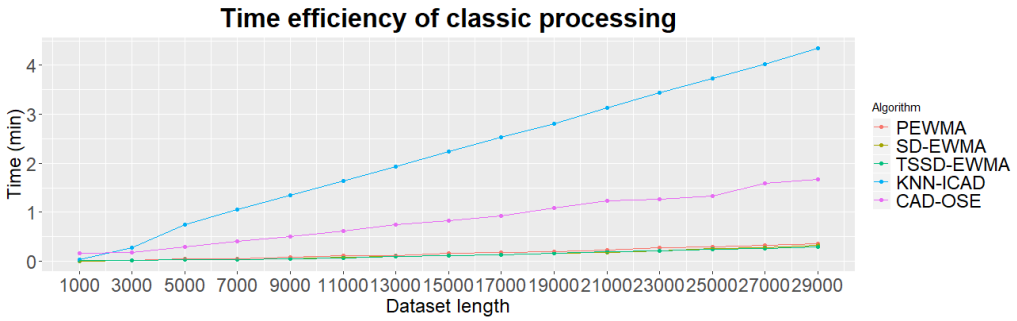


Figure 3.4: Time efficiency of detection algorithms when processing the entire dataset at once (batch processing).

Table 3.8: Time efficiency over one point processing using incremental processing algorithms.

Detector	Minimum	Mean	Maximum
PEWMA	0	0.00256	0.08
SD-EWMA	0	0.00048	0.02
TSSD-EWMA	0	0.00980	0.02
KNN-ICAD	0	0.00177	0.25
CAD-OSE	1.64	1.87000	2.41

Figure 3.4 reveal that PEWMA, SD-EWMA and TSSD-EWMA are the most time-efficient approaches, with running times falling below a half of a second for datasets comprising 29,000 data points. By contrast, the least efficient algorithm is KNN-ICAD, which requires more than four minutes to process the same dataset. Even so, it could be a reasonable time for in-laboratory experimentation. When it comes to the results of the online processing benchmark shown in Table 3.8, PEWMA, SD-EWMA, TSSD-EWMA and KNN-ICAD can operate with a collection period of less than a

half of second. CAD-OSE is the least time efficient, largely because in this implementation, it queries Python code from R for each processed point, but it still operates robustly with collection period of less than 3 seconds. Finally, we can conclude that all these algorithms are time-efficient for online processing, as the smallest recollection period prescribed in the NAB datasets is 5 seconds.

3.4. ILLUSTRATIVE EXAMPLE

To further exemplify the use of the developed package for online TSOD, this section details how to find and plot anomalies in the `Speed_7578` dataset using the KNN-ICAD detector, as well as how to evaluate the detector's performance and plot the results.

1. First of all, it is necessary to install and load the *otsad* library. As it is available in CRAN, the installation and loading are done in the usual way by using the following code:

```
# Install and load the otsad package
install.packages('otsad')
library(otsad)
```

2. The next step is to choose the dataset. A list of all available datasets can be retrieved by performing the following command:

```
# Get a list of the data sets in otsad
data(package = "otsad")

## Data sets in package otsad:
## TravelTime_387              TravelTime_387.
## TravelTime_451              TravelTime_451.
## Twitter_volume_AAPL         Twitter_volume_AAPL.
## Twitter_volume_AMZN         Twitter_volume_AMZN.
## Twitter_volume_CRM          Twitter_volume_CRM.
## Twitter_volume_CVS          Twitter_volume_CVS.
## ...
```

   In this use case, the `Speed_7578` dataset is chosen by simply typing its name as follows:

```
# Get a list of the data sets in otsad
speed_7578

##             timestamp  value  is.real.anomaly
## 1 2015-09-08 11:39:00    73                0
## 2 2015-09-08 11:44:00    62                0
## 3 2015-09-08 11:59:00    66                0
## 4 2015-09-08 12:19:00    69                0
## 5 2015-09-08 12:24:00    65                0
## ...
```

3. Now it is time to train the KNN-CAD model on the given data set. To do this, we first use the function `GetNumTrainingValues` to obtain the number of instances of the dataset. At this time, the KNN-ICAD can be trained by `CpKnnCad` for offline or by `IpKnnCad` for incremental processing:

```
# Get the number of instances to use as training set
n.train <- GetNumTrainingValues(nrow(speed_7578))

# KNN-CAD by classic processing
result <- CpKnnCad(
    speed_7578$value,
    n.train,
    threshold = 1,
    l = 19,
    k = 27,
    ncm.type = 'ICAD',
    reducefp = TRUE
)

# KNN-CAD by incremental processing
# Initialize parameters for the loop
last.res <- NULL
result <- data.frame()

## Calculate anomalies
for(i in 1:nrow(speed_7578)) {
  # calculate if it's an anomaly
  last.res <- IpKnnCad(
    data = speed_7578[i,"value"],
    n.train = n.train,
    threshold = 1,
    l = 19,
    k = 27,
    ncm.type = "ICAD",
    reducefp = TRUE,
    to.next.iteration = last.res$to.next.iteration
  )

  result <- rbind(result, c(last.res$anomaly.score, last.res$is.anomaly))
}

##       anomaly.score is.anomaly
## ...
## 285   0.960000000      FALSE
## 286   0.953333333      FALSE
## 287   0.966666667      FALSE
## 288   1.000000000       TRUE
## 289   0.500000000      FALSE
## ...
```

It is important to note that all information regarding the function, such as, description, parameters, recommendations, and use examples, are documented and accessible by `?CpKnnCad` or `help(CpKnnCad)`. The output is shown in Figure 3.5.

4. Now the results are visualized in an interactive graph. The output of the `PlotDetections` function is shown in Figure 3.6.

```
# Plot Results
```

CpKnnCad {otsad} R Documentation

Classic processing KNN based Conformal Anomaly Detector (KNN-CAD)

Description

`CpKnnCad` calculates the anomalies of a dataset using classical processing based on the KNN-CAD algorithm. KNN-CAD is a model-free anomaly detection method for univariate time-series which adapts itself to non-stationarity in the data stream and provides probabilistic abnormality scores based on the conformal prediction paradigm.

Usage

```
CpKnnCad(data, n.train, threshold = 1, l = 19, k = 27,
  ncm.type = "ICAD", reducefp = TRUE)
```

Arguments

`data`	Numerical vector with training and test dataset.
`n.train`	Number of points of the dataset that correspond to the training set.
`threshold`	Anomaly threshold.
`l`	Window length.
`k`	Number of neighbours to take into account.
`ncm.type`	Non Conformity Measure to use "ICAD" or "LDCD"
`reducefp`	If TRUE reduces false positives.

Details

`data` must be a numerical vector without NA values. `threshold` must be a numeric value between 0 and 1. If the anomaly score obtained for an observation is greater than the `threshold`, the observation will be considered abnormal. `l` must be a numerical value between 1 and 1/`n`; `n` being the length of the training data. Take into account that the value of l has a direct impact on the computational cost, so very high values will make the execution time longer. `k` parameter must be a numerical value less than the `n.train` value. `ncm.type` determines the non-conformity measurement to be used. ICAD calculates dissimilarity as the sum of the distances of the nearest k neighbours and LDCD as the average.

Value

dataset conformed by the following columns:

`is.anomaly`	1 if the value is anomalous, 0 otherwise.
`anomaly.score`	Probability of anomaly.

References

V. Ishimtsev, I. Nazarov, A. Bernstein and E. Burnaev. Conformal k-NN Anomaly Detector for Univariate Data Streams. ArXiv e-prints, jun. 2017.

Examples

```
## Generate data
set.seed(100)
n <- 350
x <- sample(1:100, n, replace = TRUE)
x[70:90] <- sample(110:115, 21, replace = TRUE)
x[25] <- 200
x[320] <- 170
df <- data.frame(timestamp = 1:n, value = x)
```

Figure 3.5: Extract from `CpKnnCad` detector documentation page.

```
myData <- cbind(speed_7578, result)
PlotDetections(myData, title = 'KNN-CAD_ANOMALY_DETECTOR')
```

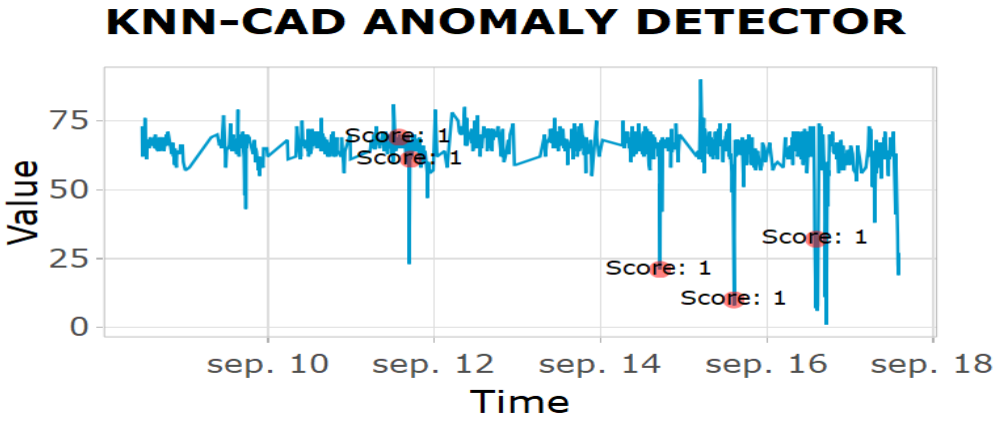


Figure 3.6: Anomaly detection results obtained by the KNN-ICAD detector.

5. To conclude, the performance of the detector is computed in terms of the NAB metric by using the code below:

```
# Get detector score
score <- GetDetectorScore(myData)
# Normalize standard result
null.perfect <- GetNullAndPerfectScores(myData)
NormalizeScore(
    score$standard,
    perfect.score = null.perfect[1, 'perfect.score'],
    null.score = null.perfect[1, 'null.score']
)
## standar.s
## 65.63917
```

The `GetDetectorScore` function has two additional parameters that allow visualizing the results of the scores in an interactive graph. The output is depicted in Figure 3.7.

```
score <- GetDetectorScore(
    myData,
    print = TRUE,
    title = "speed_7578_results_using_KNN-ICAD_detector"
)
```

3.5. CONCLUSIONS

This chapter has addressed the lack of publicly available software for online TSOD by presenting a new package developed to undertake this task. The package, called *otsad*, is an efficient and easy-to-use R package that fully

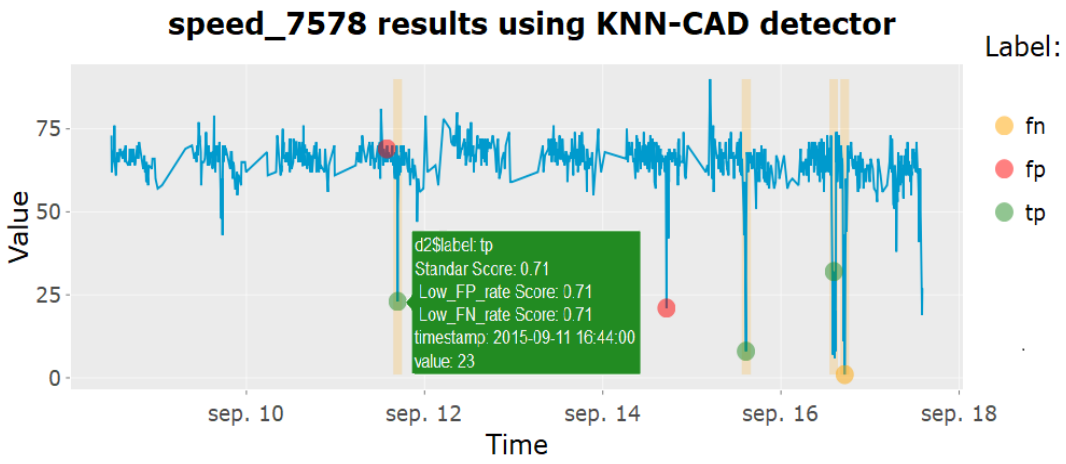


Figure 3.7: Performance scores obtained by the KNN-CAD detector using the Numenta metric.

meets the demand for anomaly detection algorithms over univariate time series in online environments. It also implements a new false positive reduction technique to improve the detection statistics of the detectors included in the package. Furthermore, the novel NAB [125] detector measurement technique is also included in the package, allowing users to compare the performance of anomaly detectors easily and more realistically.

Complimentary to the description of the package itself, the experimental results elicited by several benchmarks have been run to ascertain the time efficiency and detection performance of the algorithms implemented in the package. This experimentation has shown that all proposed algorithms are time efficient. However, in terms of performance, the results attained by the algorithms are far from the perfect detection scores. The outcomes have shown that the performance of the detectors is stringently subject to the use case, the hyperparameter values chosen for the algorithms, and the technique used for reducing the amount of false positives. This highlights the need for deriving more efficient algorithms capable of being applied to real streaming environments without the need for expert knowledge to adjust its parameters. To stimulate this search for new algorithmic proposals, additional functionalities will be added to the *otsad* package in the future, increasing its portfolio of anomaly detection algorithms and false positive reduction techniques.

4

A FRAMEWORK FOR ADAPTING ONLINE PREDICTION ALGORITHMS TO OUTLIER DETECTION OVER TIME SERIES

As concluded in the previous chapter, more and better algorithms are of utmost need for online TSOD. Most contributions reported thus far for online TSOD [19, 62] consist of adapting traditional offline learning algorithms to stream processing. However, online time series anomaly detection's constraints hinder the design of TSOD methods that effectively detect anomalies in time series data under such challenging conditions.

In the search for more and better algorithms for online TSOD, we found that several works [29, 28] have suggested that prediction-based algorithms are suitable for adaptation and use as online TSOD approaches because only past data are required during the training phase. This suitability is rooted in the maturity of real-time time series forecasting, which has been developed over decades to reach a relatively high degree of maturity compared to TSOD. However, the continuously changing behaviors of training data, their potentially non-stationary nature, and the unavailability of a complete dataset to accommodate this variability in the training phase make the adaptation of online prediction algorithms to outlier detection even more complicated, as evidenced by the scarcity of research in this domain [19, 66].

This chapter attempts to fill this research gap by proposing a novel framework that allows for the easy adaptation of any online prediction algorithm to online TSOD. To this end, the proposed framework addresses two important factors and establishes specific methodological steps for transforming an online prediction model into an outlier detection model for time series data. First, online data normalization or standardization is performed, followed by online anomaly scoring based on prediction errors. The proposed framework implements several online normalization and outlier scoring methods that are already available in state-of-the-art models, as well as novel proposals designed to improve upon baselines. Specifically, two novel normalization methods—one-pass adaptive normalization (OAN) and

one-pass adaptive min-max normalization (OAMN), as well as two scoring methods—sigma scoring (SS) and dynamic SS (DSS) are proposed.

As a second contribution of the chapter and in response to one of the main goals of the Thesis, the usability and efficacy of the proposed framework is shown by discussing the adaptation of a novel and more robust ensemble variant of an online recurrent extreme learning machine (OR-ELM) [144]. OR-ELM is an online time series forecasting algorithm capable of efficiently training a single-hidden-layer feed-forward NN in an online fashion. The variant of this model presented in this study, which is called the ensemble OR-ELM (EOR-ELM), is adapted to TSOD using the proposed framework, yielding a novel outlier detector called the EOR-ELM anomaly detector (EORELM-AD). This new detector exemplifies the potential of the proposed framework for developing novel online TSOD algorithms. Three experimental studies are conducted to provide informed answers to two different research questions (RQs) related to this novel TSOD algorithm:

- RQ1: Which streaming normalization and outlier scoring methods perform best when used within the proposed EORELM-AD approach?
- RQ2: Does EORELM-AD perform competitively compared to other TSOD methods from the literature?

The results discussed later in this chapter in response to these two RQs reveal that the proposed framework can effectively support the adaptation of online prediction algorithms to online TSOD, yielding novel approaches that can perform at par with other methods in the recent literature. Figure 4.1 conceptually summarizes the main contributions of this chapter, illustrating that the proposed framework facilitates the development of novel approaches for online TSOD using time series prediction models.

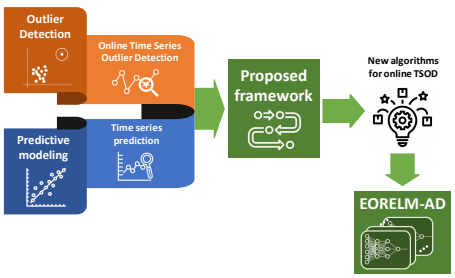


Figure 4.1: Diagram showing that two research lines in the literature (time series prediction and online TSOD) that have evolved separately over time are connected by the proposed framework, whose application yields a novel algorithm that performs competitively with respect to the state of the art.

The rest of this chapter is organized as follows. Section 4.1 describes the related work and introduces concepts necessary for a complete understanding of the proposed framework. The proposed framework and its adaptation

to EORELM-AD are described in Sections Section 4.2 and Section 4.3, respectively. Section 4.4 presents an experimental setup designed to answer the RQs formulated above. Section 4.5 presents and discusses the obtained results and compares different normalization, scoring, and TSOD algorithms. Finally, the conclusions and future research directions are discussed in Section 5.1.

4.1. BACKGROUND

Before proceeding further, it is needed to give a brief update on the state of the art (Section 4.1.1) and pause on the basic concepts and works related to two processing steps of online TSOD approaches that rely on anomaly scores: online normalization (Section 4.1.2), and online anomaly scoring (Section 4.1.3). This brief review is intended to provide the reader with the background knowledge required to understand the design of the proposed framework.

4.1.1. Online TSOD

As mentioned above, online TSOD is an emerging field of research whose interest is growing rapidly. In the time elapsed since the development of the OTSAD framework, new proposals have emerged from the literature related to online TSOD. Numenta [19] has ranked first in all related competitions for several years. Since its inception, various other algorithms have been proposed to improve the baseline detection performance set by the HTM Numenta anomaly detector. DeepAnt [97] was one of the first alternatives to outperform the baseline. In essence, DeepAnt is a deep learning method that combines convolutional and long short-term memory NNs. However, it should be noted that although DeepAnt requires very little training data, its training phase is performed offline. Recently, the online evolving spiking NNs for unsupervised anomaly detection (OeSNN-UAD) framework [145] has outperformed both the Numenta anomaly detector and DeepAnt, becoming one of the main benchmark algorithms for prospective studies. OeSNN-UAD runs entirely online and leverages eSNNs to learn to label input values in a time series as inliers or outliers.

From the perspective of prediction-based online TSOD, both of the referential studies published in [29] and [28] suggest that online prediction algorithms for time series can be suitably adapted to online TSOD. Although research on real-time time series forecasting has a relatively high degree of maturity compared to TSOD, very few proposals based on online time series forecasting can be found in the literature based on the inherent complexity of online processing. The most representative approaches based on time series prediction algorithms are the online non-parametric Bayesian

method (OLAD) [66] and Numenta [19] anomaly detector mentioned above Section 3.1.1.

In this study, we stepped beyond the principles behind the design of the Numenta anomaly detector by designing a framework that implements different methodologies for online data normalization and online outlier scoring. The effectiveness of this framework was proven by developing a novel detector called EORELM-AD based on the OR-ELM prediction algorithm. The performance analysis presented in this study demonstrates that the performance of EORELM-AD is very competitive compared to that of most recent state-of-the-art methods for online TSOD [97, 145].

4.1.2. Online normalization

In offline learning, data can be normalized in several ways. One of the most widely used approaches for data normalization is min-max normalization, where values are mapped to a predefined range $\mathbb{R}[low, high]$ (typically set to $[0, 1]$ or $[-1, 1]$). The normalized value x' for a given feature or variable x is calculated as follows:

$$x' = (high - low) \cdot \frac{x - x_{min}}{x_{max} - x_{min}} + low, \tag{4.1}$$

where $x_{min} = \min x$ (corr. x_{max}). Another normalization strategy for recurrent use is z-score standardization, where the values of a feature are normalized according to its mean and standard deviation as follows:

$$x' = \frac{x - \mu}{\sigma}, \tag{4.2}$$

where μ denotes the mean and σ denotes the standard deviation.

However, naive min-max normalization and z-score standardization methods are not appropriate for normalizing streaming data. Min-max normalization requires the maximum x_{max} and minimum x_{min} values of a variable to be computed over an entire dataset, which implicitly assumes the availability of all data samples to learn such values. This assumption clashes with the requirements of a streaming setup because data instances are generated continuously, meaning a complete dataset is never available for normalization. Z-score standardization is the most commonly used normalization methodology for stationary time series because the mean μ and standard deviation σ are constant under the premise of stationarity. Regardless, in most real-world streaming time series, the statistical properties of the time series may evolve over time, meaning this assumption is no longer valid.

To overcome these issues, several alternative normalization methods have been proposed to modify these popular normalization techniques to make them suitable for streaming time series. The two main design strategies are as follows.

- Adaptive methods update normalization statistics whenever a new value arrives over a stream without the need for retaining any historical values. These methods require little memory and operate very quickly. However, because of the complexity of their design, there have been very few adaptive normalization proposals. One of these methods is the adaptive z-score normalization proposed in [146], in which classical z-score standardization is applied using dynamically calculated means and deviations. To avoid confusion with other methods, we will hereafter refer to this method as dynamic z-score normalization (DN).

- Window-based methods use sliding windows to maintain and update parameters based on the most recent data. One traditional approach called window-based normalization (WN) divides a time series into sliding windows and normalizes incoming data samples according to statistical properties (μ_t and σ_t) inferred from the current window W_t [147, 148]. Another more sophisticated window-based approach was introduced in [149]. In this work, AN was proposed to normalize non-stationary heteroscedastic (non-uniform volatility) time series. This method was specifically designed for NN training. Furthermore, it is important to note that this method is not fully incremental and it cannot be applied to one-pass online learning because it is used to train different NNs in an offline manner. However, it possesses some incremental properties that can be exploited to adapt it to an online environment. A distributed approach to adaptive min-max normalization (AMN) for big data stream learning was recently proposed in [150]. This method is designed for online chunk-by-chunk (batches of data) processing, where disjoint (non-overlapping) fixed-size sliding windows are used. Although this method is incremental, it is designed for chunk-by-chunk processing and is not suitable for one-pass online learning.

As discussed above, several proposals for online data normalization are available in the literature. Among them, AN [149] and AMN [150] require adaptation to online one-pass processing. For this reason, in this chapter a novel approach is proposed to adapt these methods to online one-pass processing, as detailed in Section 4.2.1.

4.1.3. Streaming anomaly scoring

Regarding the estimation of the anomaly scores of instances (points) arriving in a data stream, prediction-based TSOD relies on the assumption that an outlier is a point that significantly deviates from its predicted value. Therefore, a point arriving at time t (i.e., x_t) can be considered as an outlier if the distance to its predicted value ($\widehat{x}_t$) is greater than a predefined threshold θ, i.e.:

$$\text{If } ||x_t - \widehat{x}_t|| > \theta \text{ then an } \texttt{OUTLIER} \text{ occurs at time } t. \tag{4.3}$$

Here, $||\cdot||$ denotes a vectorial norm (e.g., absolute value in the case of univariate time series or Euclidean norm when dealing with multivariate time series). With this definition in mind, online prediction-based TSOD methods must perform two tasks: computing the predicted value $\widehat{x}_t$, and computing the predefined threshold θ.

- To compute the expected value, several online time series prediction algorithms are available in the literature. This research field has been extensively investigated and comprehensively examined in recent surveys on the topic [151, 152, 153]. We refer interested readers to these overviews for additional details regarding the upsurge of algorithms proposed in this field.

- Incrementally computing the threshold θ is not an easy task because not all previous data can be stored in memory. To compute the predefined threshold, two different strategies can be adopted. First, we can calculate confidence or prediction intervals. Second, we can use historic prediction errors to calculate the degree of outlierness or predefined threshold. Based on the first strategy, the OLAD for outlier detection was introduced in [66]. This work developed an online Student-t process method to learn the underlying dynamics of time series for prediction and to compute prediction intervals. In this method, the current value is considered to be an outlier if it falls outside the prediction interval. The major drawback of this threshold-based strategy is that the calculation of the prediction intervals depends on parameters learned by the prediction model. This makes it challenging to generalize the threshold calculation method to other prediction algorithms.

Since it is independent of the prediction algorithm and its parameters, we focus on the second strategy, namely the use of historical prediction errors to compute the degree of outlierness or predefined threshold. There are two different approaches to compute normalized anomaly scores and thresholds based on prediction errors [19, 154]. The anomaly likelihood (AL) introduced in [19] is a novel incremental threshold used alongside the HTM prediction algorithm, giving rise to the so-called HTM Numenta anomaly detector. Specifically, the AL scoring approach is a general method designed in a completely independent fashion relative to the prediction model in use and it only requires prediction errors for computing score values. Recently, the authors of [154] proposed DeepAD, which is an anomaly detector based on an ensemble of different prediction algorithms. The prediction algorithms are trained in batches and periodically. Additionally, a dynamic threshold (DT) method is proposed to detect anomalies in real-time.

Applying a threshold to unbounded raw prediction errors is the easiest solution in practice. However, choosing a threshold value is not straightforward and can lead to many false positives (FPs) if the value is not tailored to the target dataset. This weakness was the main motivation for the proposal of two new streaming scoring methods, which are described in Section 4.2.2.

4.2. PROPOSED FRAMEWORK FOR ADAPTING ONLINE PREDICTION MODELS TO OUTLIER DETECTION

The proposed framework is composed of two main components: 1) online data normalization and 2) streaming anomaly scoring based on prediction errors. The procedure for adapting an online prediction model for outlier detection using this framework is illustrated in Figure 4.2. First, if required by the prediction model itself, incoming data points are normalized incrementally. Then, normalized data points are used for training and to predict the next value in the time series based on the chosen prediction model. To compute the degree of outlierness, the prediction error is then calculated and input to the outlier scoring function. Finally, a predefined threshold is used to determine whether the current sample is an outlier.

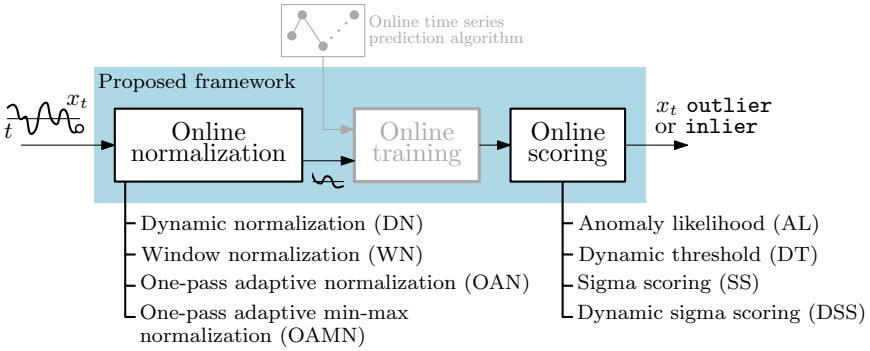


Figure 4.2: Procedure to adapt online prediction models for outlier detection using the proposed framework.

Figure 4.2 presents the normalization and scoring functions embedded in each component, along with the procedure for transforming a prediction algorithm into an anomaly detection algorithm.

4.2.1. Online normalization component

As described in Section 4.1.2, there are several incremental normalization techniques. Some of them are purely incremental (i.e., DN [146]), whereas others are window-based (e.g., WN [147, 148], AN [149], and AMN [150]). However, it is worth noting that the adaptive normalization methods (AN and AMN) are not specifically designed for one-pass data processing. Therefore, one-pass variations of both methods are proposed, called OAN and OAMN, to overcome this limitation. Furthermore, because each method has its advantages and disadvantages, all these normalization methods are utilized in the normalization component of the proposed framework. In the following subsections, OAN and OAMN are detailed.

4.2.1.1. *One-pass adaptive normalization for online learning*

AN [149] normalizes non-stationary heteroscedastic (non-uniform volatility) time series and is specifically designed for NN training. Although it features certain functionalities that can be implemented incrementally, it is not fully incremental, which is why it is mostly used in offline setups. Therefore, based on this approach, the OAN method for online learning is proposed. This novel method is described below. Similar to AN, OAN is divided into three stages.

1. *Data transformation:* OAN uses a sliding window $X_t = \{x_{t-n}, \ldots, x_t\}$ of fixed size n to facilitate one-pass online processing, where the window is updated whenever a new instance arrives.

   The normalization procedure begins by transforming the non-stationary time series X_t into a new stationary sequence R_t composed of disjoint sliding windows. To this end, given a time series $X_t = \{x_{t-n}, \ldots, x_t\}$ for each sample $X_t[i]$, its moving average of order k is calculated. Two different moving average techniques can be used. The simple moving average computes a sequence of non-weighted averages as follows:

   $$S_s^{(k)}[i] = \frac{1}{k}\sum_{j=1}^{k} X[j] \quad i = 1, \ldots, n-k+1. \tag{4.4}$$

   The exponential moving average weights averaged terms by a constant smoothing factor $0 \leq \alpha \leq 1$, which is typically expressed in terms of k (order, e.g., $\alpha = 2/(k+1)$) as follows:

   $$S_e^{(k)}[1] = \frac{1}{k}\sum_{j=1}^{k} X[j] \tag{4.5}$$

   $$S_e^{(k)}[i] = (1-\alpha)S_e^{(k)}[i-1] + \alpha X[i+k-1], \quad i = 2, \ldots, n-k+1. \tag{4.6}$$

   Then, given the current time series X_t, its k-moving average $S^{(k)}$, and the length ω of a disjoint sliding window, the new stationary sequence $R \in \mathbb{R}_t^{\phi \times \omega}$ is computed as:

   $$R^{(k)}[i,j] = \frac{X[\text{i+j-1}]}{S^{(k)}[i]},\ i = 1, \ldots, \phi,\ j = 1, \ldots, w. \tag{4.7}$$

   Therefore, R is divided into $\phi = n - \omega + 1$ disjoint sliding windows of size ω, as represented in Figure 4.3. The first $\phi - 1$ rows are used as training instances, whereas the last row is used as a testing instance. Because only the last instance is required for one-pass online model training, the window size n is fixed to $n = w + 1$. In this manner, the computational cost and initialization time are significantly reduced.

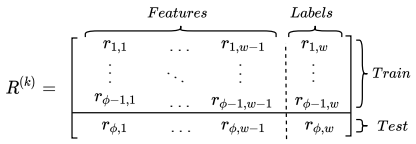


Figure 4.3: Structure of a new sequence R.

It is worth noting that the moving average method and order k used for data normalization vary in accordance with the target time series. Therefore, the adjustment levels for all combinations of different moving averages and orders k are computed to select the best moving average type and order k as:

$$\delta(S^{(k)}, R^{(k)}[i]) = \frac{1}{\omega} \sum_{j=i}^{i+w-1} (X[j] - S^{(k)})^2, \quad i = 1, \ldots, \phi - 1, \tag{4.8}$$

$$\delta(S^{(k)}, R^{(k)}) = \frac{1}{\phi} \sum \delta(S^{(k)}, R[i]), \tag{4.9}$$

where the configuration that yields the lowest adjustment level is selected.

2. *Outlier removal:* Outliers that occur at the extreme boundaries of a time series must be removed because they can lead to non-realistic minimum and maximum values. To this end, a statistical criterion is employed. Given the first quartile ($Q1$), third quartile ($Q3$), and inter-quartile range $IQR = Q3 - Q1$, all values that do not fall in the range $[Q1 - 3 \cdot IQR, Q3 + 3 \cdot IQR]$ are considered as outliers. Based on the changes made in the first step, there may be little training data available for detecting outliers, potentially leading to many FPs. To overcome this issue, instead of removing any training row in sequence R that contains at least one outlier (as is done in AN), in the proposed method, values lower than $Q1 - 3 \cdot IQR$ and higher than $Q1 + 3 \cdot IQR$ are mapped to -1 and 1, respectively.

3. *Data normalization:* R is normalized to $\mathbb{R}[-1, 1]$ using min-max normalization. Figure 4.4 presents the structure of a normalized sequence.

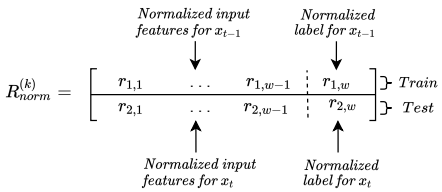


Figure 4.4: Structure of a normalized sequence.

Before proceeding further with the remaining normalization methods, it is important to note that the OAN approach requires the current value of x_t to normalize its input features. Therefore, this method cannot be used in prediction problems over a time series, where the goal is to predict a future value x_{t+1} at time t. Regardless, we underscore the fact that this is not a problem for outlier detection because the sliding time series x_t can be used to generate normalized training and testing instances. The normalized training and testing instances can then be utilized to predict the current data value x_t instead of the next value x_{t+1}. Finally, the error between the predicted value $\widehat{x}_t$ and current value x_t can be computed.

4.2.1.2. *One-pass adaptive min-max normalization for online learning*

AMN [150] is a normalization approach designed for online chunk-by-chunk learning. Therefore, it uses disjoint sliding windows. To adapt this method for one-pass online learning, OAMN is introduced.

To perform one-pass processing, a sliding window W of the past n data values is used. First, the algorithm waits for the first n data points to fill up the first window of samples $W_{t-1} = \{x_{t-n}, \ldots, x_{t-1}\}$. When this window is full, its statistics (i.e., mean μ_{t-1}, minimum $\min_{t-1}$, and maximum $\max_{t-1}$) are computed and used to initialize the global minimum $globalMin = \min_{t-1}$ and global maximum $globalMax = \max_{t-1}$. After these global statistics are obtained, min-max normalization (see Equation (4.1)) is used to normalize the first window W_{t-1}. Then, whenever a new data point x_t arrives, the current window W_t is updated as $W_t = \{x_{t-n+1}, \ldots, x_t\}$. Subsequently, its mean μ_t is obtained and used to compute the relative percentage change (i.e., $\delta = |\mu_t - \mu_{t-1}|/\mu_{t-1}$). Then, if the value of δ exceeds a predefined threshold θ [%], the minimum $\min_t$ and maximum $\max_t$ values of the current window W_t are computed and used to update the global minimum $globalMin = \min_t$ and global maximum $globalMax = \max_t$. Otherwise, $globalMin$ and $globalMax$ remain unchanged. Finally, using these global minimum and maximum values, the current window W_t is normalized.

4.2.2. Online outlier scoring

As described in Section 4.1.3, there have been very few studies on online anomaly scoring methods based on prediction error. Therefore, both AL [19] and DT [154] are included in the online outlier scoring component of the proposed framework. Furthermore, two novel streaming scoring methods called SS and DSS are proposed and explained below.

4.2.2.1. *Sigma scoring*

The first proposed online anomaly scoring method, namely SS, is based on three-sigma control charts and the scoring method proposed in [12]. Similar to

other online anomaly scoring methods, this method relies on a sliding window $W_t = \{e_{t-n+1}, \ldots, e_t\}$ of the most recent n prediction errors to operate online. Specifically, a three-sigma control chart is a statistical process control tool for determining if a manufacturing or business process is in a state of control. This method defines upper (UCL) and lower (LCL) control limits considering three standard deviations (i.e., 3σ) from the mean μ. Samples that lie outside these limits are considered to be anomalous. However, the major drawback of this method is that it does not provide any information regarding the extent to which a sample is anomalous (outlierness), which is important in real-world settings for assessing the severity of phenomena.

Therefore, to provide a measure of outlierness based on three-sigma control charts, we adopt the scoring rule proposed in [12], which was originally used to score anomalies detected by one-class learning-based methods. This rule is defined as follows:

$$score(x_t) = \exp\left(-\frac{\ln 2}{R^2}|x_t - o|^2\right),\ R = 3\sigma_t,\ o = \mu_t. \tag{4.10}$$

It is inspired by the Bayesian theorem and meets the following four desirable criteria:

(1) $0 < score(x) < 1$,

(2) $score(x) = 0.5$ for samples on the boundary,

(3) $score(x) < 0.5$ for samples inside the boundary, and

(4) $score(x) > 0.5$ for samples outside the boundary.

To adapt this scoring method to online three-sigma control limits, the radius is defined as $R = 3\sigma_t$, and the center is defined as $o = \mu_t$, where σ_t and μ_t are the standard deviation and mean of the current window W_t, respectively.

4.2.2.2. Dynamic sigma scoring

The second novel scoring method adopted in the proposed framework is a dynamic version of the SS method. In this case, the mean and the standard deviation are computed dynamically whenever a new prediction error is obtained instead of using a sliding window. Accordingly, the dynamic mean μ_t and dynamic standard deviation σ_t are computed as discussed in [146] for dynamic data normalization, namely:

$$\mu_t = \mu_{t-1} + \frac{e_t - \mu_{t-1}}{t}, \tag{4.11}$$

$$s_t = s_{t-1} + (e_t - \mu_{t-1})(e_t - \mu_t),$$

$$\sigma_t = \sqrt{s_t/(t-1)}. \tag{4.12}$$

DSS then computes an anomaly score similar to SS (see Equation (4.10)), where the radius R is equal to three times the current standard deviation ($R = 3\sigma_t$) and the center is equal to the current mean (corr. $o = \mu_t$).

4.3. ENSEMBLE-BASED OR-ELM AND THE EOR-ELM ANOMALY DETECTOR

To demonstrate the usability of the proposed framework, this section discusses the adaptation of a novel robust ensemble variant of OR-ELM, a randomization-based neural network used in the literature to predict time series in an online fashion. This section begins with an introduction to OR-ELM in Section 4.3.1. Then, Section 4.3.2 details the novel ensemble version of this mode, coined as EOR-ELM. Finally, Section 4.3.3 demonstrates how the proposed framework is used to construct a novel anomaly detection algorithm called EORELM-AD using the framework to adapt the EOR-ELM prediction algorithm described previously.

4.3.1. Online recurrent extreme learning machines

OR-ELM is an online NN that can be applied to several machine learning problems, including online time series forecasting [144]. OR-ELM is an improved variation of the online sequential ELM (OS-ELM) [155]. Both OR-ELM and OS-ELM utilize single-layer feed-forward NNs that are trained online on a one-by-one or chunk-by-chunk basis, where the latter may have a fixed or varying chunk size. In contrast to OS-ELM, which randomly assigns weights to the neural component of the model, OR-ELM utilizes autoencoders (AEs) combined with a normalization layer to learn and update the input and hidden weights. In this chapter OR-ELM is adopted as a base predictor for the proposed anomaly detector because experimental results reported in the literature demonstrate that it outperforms other online sequential learning algorithms, including online long short-term memory [156] and HTM [157], which is used in the popular HTM Numenta anomaly detector [19].

OR-ELM consists of three networks: a recurrent NN (RNN), which is the main network used for prediction, and two single-layer feed-forward networks (SLFNs), which are auxiliary ELM-AE networks [158] used for learning the RNN's input and hidden weights. Let $f_{\tilde{N}}$ be an SLFN with $\tilde{N}$ hidden nodes, $\mathbf{x} \in \mathbb{R}^n$ be an n-sized input data vector, and $T \in \mathbb{R}^m$ be the target vector. The output of the SLFN is given by:

$$f_{\tilde{N}}(\mathbf{x}) = \sum_{i=1}^{\tilde{N}} \beta_i \cdot G\left(\mathbf{a}_i, b_i, \mathbf{x}\right), \tag{4.13}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{a}_i \in \mathbb{R}^n$, a_i, and b_i are the training parameters for the hidden nodes and β_i is the output weight that connects the nodes of the hidden layer to the i-th output node. $G(\mathbf{a}_i, b_i, \mathbf{x})$ can be defined differently according to the selected type of SLFN node. For example, for additive SLFN nodes, $G(\mathbf{a_i}, b_i, \mathbf{x})$ is defined as:

$$G(\mathbf{a}_i, b_i, \mathbf{x}) = g(\mathbf{a}_i \cdot \mathbf{x} + b_i), \quad b_i \in \mathbb{R} \tag{4.14}$$

where $g(\cdot) : \mathbb{R} \to \mathbb{R}$ is the activation function (e.g., sigmoid). Alternatively, for radial basis function SLFN nodes, $G(\mathbf{a}_i, b_i, \mathbf{x})$ yields:

$$G\left(\mathbf{a}_i, b_i, \mathbf{x}\right) = g\left(b_i \left\|\mathbf{x} - \mathbf{a}_i\right\|\right), \quad b_i \in \mathbb{R}^+, \tag{4.15}$$

where the activation function $g(\cdot)$ is set to be Gaussian.

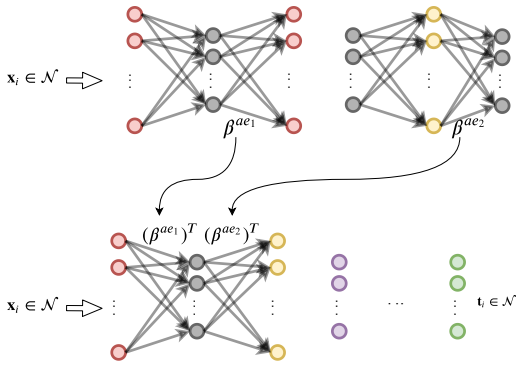


Figure 4.5: Weight learning process followed by ELM-AE. Each color represents a different layer. The hidden weights $(\beta^{ae})^\top$ of multiple ELM layers are calculated using ELM-AE, in which input and output targets are the same. The output weight β^{ae} of ELM-AE is computed by the conventional ELM method.

For each input, the OR-ELM has three stages, called the input, hidden, and output stages. Each stage has its own hidden layer output matrix $\mathbf{H}$ and output weights $\boldsymbol{\beta}$. In the first two stages, the hidden layer output matrix weights are learned using ELM-AE [158] (this process is illustrated in Figure 4.5), which is an ELM in which the input data sample $\mathbf{x} \in \mathbb{R}^n$ is also used as a target $\mathbf{x} = \mathbf{t}$. In ELM-AE, the input weight $\mathbf{a} \in \mathbb{R}^{n \times \tilde{N}}$ and bias values $b \in \mathbb{R}^{\tilde{N}}$ are assigned randomly and then orthogonalized as:

$$\mathbf{a}^\top \mathbf{a} = \mathbf{I}, \quad b^\top b = 1, \tag{4.16}$$

which helps enhance the generalization performance of the model. The output weights $\boldsymbol{\beta} \in \mathbb{R}^{\tilde{N} \times n}$ of ELM-AE are computed similarly to a basic ELM as:

$$\boldsymbol{\beta} = \mathbf{H}^\dagger \mathbf{T}, \quad \mathbf{H}^\dagger = \left(\mathbf{H}^\top \mathbf{H} + \frac{\mathbf{I}}{C}\right)^{-1} \mathbf{H}^T, \tag{4.17}$$

where $\mathbf{H}^\dagger = (\mathbf{H}^\top \mathbf{H})^{-1} \mathbf{H}^\top$ is the Moore-Penrose generalized inverse of the hidden layer output matrix, which is typically calculated using the singular

value decomposition method, and C is a regularization constant added to prevent $\mathbf{H}^T\mathbf{H}$ from being a singular matrix. Because $\boldsymbol{\beta}^T$ is responsible for the transformation from the input data to the hidden feature space, it can be used as an input weight for the ELM to extract better hidden features according to several studies [158, 159, 160].

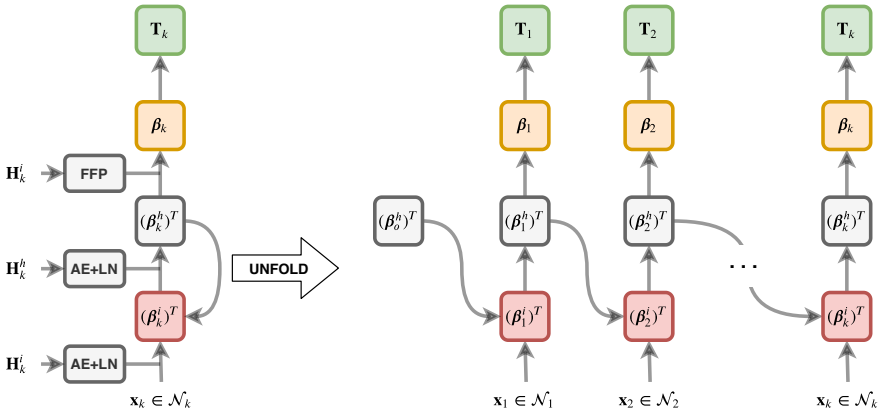


Figure 4.6: Diagram showing the architecture of OR-ELM. The structure is the same as that of a typical RNN, except for the normalization component. The right side is an unfolded representation for the sake of visual simplicity. The red and white frames respectively represent the normalization and hidden layers, which are trained recurrently using the hidden weights as the next input weights. These weights are learned using ELM-AE-IW and ELM-AE-HW, respectively. The output stage is represented in yellow, whereas the prediction is represented in green.

Regarding OR-ELM, we recall that it has three stages, namely the input, hidden, and output stages. Each of these stages has its own hidden layer output and regular output (see Figure 4.6):

- Input stage: $\mathbf{H}^i \in \mathbb{R}^{n\times\tilde{N}}$ hidden layer output matrix and $\boldsymbol{\beta}^i \in \mathbb{R}^{n\times\tilde{N}}$ output weight.
- Hidden stage: $\mathbf{H}^h \in \mathbb{R}^{\tilde{N}\times\tilde{N}}$ hidden layer output matrix and $\boldsymbol{\beta}^i \in \mathbb{R}^{\tilde{N}\times\tilde{N}}$ output weight.
- Output stage: $\mathbf{H} \in \mathbb{R}^{\tilde{N}\times m}$ hidden layer output matrix and $\boldsymbol{\beta}^i \in \mathbb{R}^{\tilde{N}\times m}$ output weight.

First, let ELM-AE-IW and ELM-AE-HW denote the ELM-AEs used for learning the weights of the hidden layer output matrices of the input and hidden states, respectively. OR-ELM consists of two phases, namely the initialization phase and online sequential learning phase, which are described as follows:

1. *Initialization phase:* The initial output weights $\boldsymbol{\beta}_0^i$, and $\boldsymbol{\beta}_0^h$, $\boldsymbol{\beta}_0$ and the auxiliary matrices $\mathbf{P}_0^i$, $\mathbf{P}_0^h$ and $\mathbf{P}_0$ are defined as:

$$\boldsymbol{\beta}_0 = 0, \quad \mathbf{P}_0 = \left(\frac{\mathbf{I}}{C}\right)^{-1}. \tag{4.18}$$

The hidden layer output matrix $\mathbf{H}_0$ and input weights for ELM-AE-IW and ELM-AE-HW are drawn randomly following a normal distribution with a mean of zero and standard deviation of one.

2. *Online sequential learning phase:* OR-ELM is able to learn sequentially each time a new chunk of data arrives. Let $\mathcal{N}_{k+1}$ be a new chunk of data consisting of N_{K+1} samples or points. For mathematical simplicity, we assume that $N_{K+1} = 1$. Therefore, if $\mathbf{x}_{k+1} \in \mathcal{N}_{k+1}$ is the input to the OR-ELM, then three online sequential learning steps are performed:

(Step A) *Input stage weights learning:* The weights $\mathbf{W}^i_{k+1}$ of the input hidden layer output matrix $\mathbf{H}^i_{k+1}$ are assigned randomly in ELM-AE-IW. Then, a normalization layer is applied to the activations obtained using these weights. The resulting hidden layer output matrix $\mathbf{H}^i_{k+1}$ is calculated as:

$$\mathbf{H}^i_{k+1} = g\left(norm\left(\mathbf{W}^i_{k+1}\mathbf{x}_{k+1}\right)\right), \tag{4.19}$$

where $norm(x)$ is defined as:

$$norm(x) = \frac{x - \mu^i}{\sqrt{\sigma^{i^2} + \epsilon}}, \tag{4.20}$$

with $\mu^i = \frac{1}{\tilde{N}}\sum_{j=1}^{\tilde{N}} x_j$, $\sigma^i = \frac{1}{\tilde{N}}\sum_{j=1}^{\tilde{N}} (x_j - \mu^i)^2$. ϵ is a small additive constant used to avoid zero division. The output weights $\boldsymbol{\beta}^i_{k+1}$ of ELM-AE-IW are calculated by setting $\mathbf{T}_{k+1} = \mathbf{x}_{k+1}$ in the recursive-least-squares operation as:

$$\boldsymbol{\beta}^i_{k+1} = \boldsymbol{\beta}^i_k + \mathbf{P}^i_{k+1}\mathbf{H}^{i\top}_{k+1}\left(\mathbf{x}_{k+1} - \mathbf{H}^i_{k+1}\boldsymbol{\beta}^i_k\right), , \tag{4.21}$$

$$\mathbf{P}^i_{k+1} = \frac{1}{\lambda}\mathbf{P}^i_k, - \mathbf{P}^i_k\mathbf{H}^{i\top}_{k+1}\left(\lambda^2 + \lambda\mathbf{H}^i_{k+1}\mathbf{P}^i_k\mathbf{H}^{i\top}_{k+1}\right)^{-1}\mathbf{H}^i_{k+1}\mathbf{P}^i_k, \tag{4.22}$$

where $\lambda \in (0, 1]$ is a forgetting factor that facilitates forgetting outdated input data and reducing their impact on subsequently learned chunks.

Once the output weights $\boldsymbol{\beta}^i_{k+1}$ have been computed, the transpose of these weights is used as the input weight vector $\mathbf{W}_{k+1}$ for OR-ELM, yielding:

$$\mathbf{W}_{k+1} = \boldsymbol{\beta}^{i\top}_{k+1}. \tag{4.23}$$

(Step B) *Hidden stage weights learning:* The weights for this stage are learned in the same manner as in the first stage, but by using ELM-AE-HW. The hidden layer output $\mathbf{H}_{k+1} \in \mathbb{R}^{\tilde{N}}$ is propagated to the corresponding hidden layer so that the hidden-layer output matrix $\mathbf{H}^h_{k+1}$ can be computed as:

$$\mathbf{H}^h_{k+1} = g\left(norm\left(\mathbf{W}^h_{k+1}\mathbf{H}_k\right)\right), \tag{4.24}$$

The output weights $\boldsymbol{\beta}^h_{k+1}$ of ELM-AE-HW are calculated using recursive-least-squares (see Equation (4.21)) as:

$$\boldsymbol{\beta}^h_{k+1} = \boldsymbol{\beta}^h_k + \mathbf{P}^h_{k+1}\mathbf{H}^{h\top}_{k+1}\left(\mathbf{H}_k - \mathbf{H}^h_{k+1}\boldsymbol{\beta}^h_k\right), \tag{4.25}$$

$$\mathbf{P}^h_{k+1} = \frac{1}{\lambda}\mathbf{P}^h_k, - \mathbf{P}^h_k\mathbf{H}^{h\top}_{k+1}\left(\lambda^2 + \lambda\mathbf{H}^h_{k+1}\mathbf{P}^h_k\mathbf{H}^{h\top}_{k+1}\right)^{-1}\mathbf{H}^h_{k+1}\mathbf{P}^h_k, \tag{4.26}$$

where the transpose of $\boldsymbol{\beta}^i_{k+1}$ is used as the hidden weight vector $\mathbf{V}_{k+1}$ for OR-ELM as:

$$\mathbf{V}_{k+1} = \boldsymbol{\beta}^{h\top}_{k+1}. \tag{4.27}$$

(Step C) *Output stage weights learning:* In the third step of the online sequential learning procedure of OR-ELM, the hidden layer output matrix $\mathbf{H}_{k+1}$ is calculated from the previous weights $\mathbf{W}_{k+1}$ and $\mathbf{V}_{k+1}$ as:

$$\mathbf{H}_{k+1} = g\left(\text{norm}\left(\mathbf{W}_{k+1}\mathbf{x}_{k+1} + \mathbf{V}_{k+1}\mathbf{H}_k\right)\right). \tag{4.28}$$

Finally, the output weights $\boldsymbol{\beta}_{k+1}$ are calculated as:

$$\mathbf{P}_{k+1} = \mathbf{P}_k, - \mathbf{P}_k\mathbf{H}^{\top}_{k+1}\left(\mathbf{I} + \mathbf{H}_{k+1}\mathbf{P}_k\mathbf{H}^{\top}_{k+1}\right)^{-1}\mathbf{H}_{k+1}\mathbf{P}_k, \tag{4.29}$$

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \mathbf{P}_{k+1}\mathbf{H}^{\top}_{k+1}\left(\mathbf{T}_{k+1} - \mathbf{H}_{k+1}\boldsymbol{\beta}^{(k)}\right). \tag{4.30}$$

When applying OR-ELM to anomaly detection, two fundamental factors must be considered, namely the online normalization of data and sensitivity of the forgetting factor λ.

4.3.1.1. Online data normalization

OR-ELM and OS-ELM are generic NNs that can be used in various machine learning problems such as classification or forecasting. However, when used in offline environments, both approaches normalize data prior to training. To understand how the normalization methods introduced in Section 4.2 can be used in conjunction with OR-ELM or OS-ELM, it is necessary to explain how input data are transformed into training and testing data to train a NN.

To train OR-ELM for one-pass time series forecasting, a window $W = \{x_{t-n}, \ldots, x_{t-1}\}$ of the most recent n values is maintained. To predict the next value x_{t+1} given the sliding window W and current value x_t, new training and testing instances must be generated. As shown in Figure 4.7, all n values in the window W are used as input features for the training phase, whereas the current value x_t is used as an output or label. Once OR-ELM is

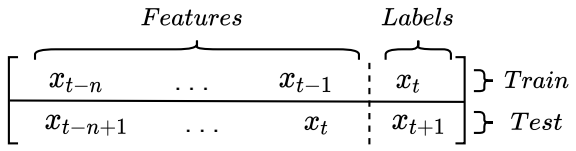


Figure 4.7: Structure of OR-ELM train and test instances.

trained, the next n values $\{x_{t-n+1}, \ldots, x_t\}$ are used as features to predict the future value x_{t+1}. This means that the input dimension for OR-ELM is n (i.e., the size of the window of recent data instances) and the output dimension, assuming a univariate time series, is one.

4.3.1.2. Sensitivity of the forgetting factor

As stated in [144], both the OR-ELM and OS-ELM algorithms are very sensitive to the value of the forgetting factor λ. This parameter allows these algorithms to forget past input data in their learning procedures so that the effects of outdated data on the knowledge learned by a trained model is minimized. When $\lambda = 1$, a model does not forget anything. The smaller the value of λ, the more the information is forgotten and the faster the error decreases. However, as shown in [144] and verified experimentally in preliminary experiments performed during the Thesis, if $\lambda < 1$ and the number of hidden nodes in a model is high, then the predictions of an NN begin to deviate until the error explodes and renders the model useless.

For online processing, it is important to tune the capacity of forgetting outdated input data to account for possible non-stationarities and undetected anomalies in a data stream. Regardless, in unsupervised online problems such as TSOD, it is difficult to find a suitable value of λ and number of hidden neurons that ensure error-free operation of an algorithm and good adaptability to the characteristics of the target dataset. This is the main problem targeted by the proposed EOR-ELM approach, which is an ensemble that uses several instances of OR-ELM with different parameters. The design rationale of EOR-ELM is detailed in the following section.

4.3.2. Ensemble-based OR-ELM

As described in Section 4.3.1.2, both the OR-ELM and OS-ELM algorithms are very sensitive to the value of the forgetting factor λ. Additionally, based on a lack of ground-truth labels and the potential for concept drift, choosing the optimal hyperparameter configuration is difficult and the optimal configuration can change over time. To tackle this problem, the EOR-ELM approach is proposed. By combining diverse learners, ensembles can reduce the dependence of a model on a specific dataset and overcome the weak performance stemming from a lack of ground-truth labels [70]. Additionally,

they alleviate the model selection problem and have better generalization capabilities than a single method. These factors make ensembles a suitable solution for outlier detection [108] and online TSOD.

The EOR-ELM proposal combines several instances initialized with different parameter settings. In this manner, the hyperparameter selection problem is circumvented, which reduces the dependency of the model's configuration on the target dataset. Furthermore, EOR-ELM removes deviating models in each iteration by initializing new models, so it can easily adapt to distribution changes and significantly reduce FPs. Therefore, EOR-ELM not only overcomes the problems associated with the forgetting factor, but also provides a much more robust approach to distribution changes and anomalies in time series that can make the knowledge learned by a model obsolete.

For the sake of clarity, Section 4.3.2.1 explains the mode of operation of the proposed ensemble, and then in Section 4.3.2.2, the structure of the proposed ensemble is analyzed in detail.

4.3.2.1. *Operating mode of the EOR-ELM*

In the following we explain how the Ensemble-based OR-ELM works. EOR-ELM is divided into three stages: initialization, training, and testing. A visual workflow of the proposed EOR-ELM is presented in Figure 4.8.

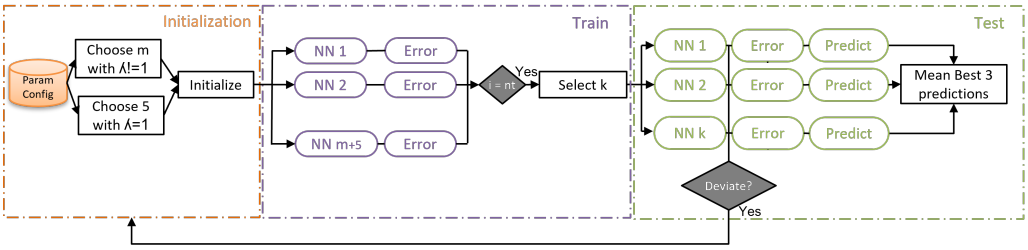


Figure 4.8: Schematic diagram showing the proposed EOR-ELM workflow. All combinations of different forgetting factors and the number of hidden neuron values are computed in the initialization step. From this set, m different parameter settings with a forgetting factor lower than one and five different parameter settings with a forgetting factor equal to one are selected to initialize $m + 5$ EOR-ELM networks. In the training step, the models are trained and used to predict training data. Once the training step is completed, the best k predictors are selected and used to learn and predict new incoming data. If the performance of any of the predictors degrades, new OR-ELMs are trained. The final prediction is the mean of the best three forecasted values.

1. *Initialization:* First, all combinations of different forgetting factors and the number of hidden neuron values are computed. Then, from this set, m different parameter settings with a forgetting factor lower than one are selected uniformly at random and used as the parameters for m OR-ELMs. To ensure that we have at least one valid predictor when the error of all

m previous models increases without bound (particularly when handling neural architectures consisting of a large number of hidden neurons), five additional OR-ELMs are initialized, all of which are configured with random parameter settings and a fixed forgetting factor value equal to one.

2. *Training:* During the training phase, $m + 5$ models are trained and used to predict training data. Additionally, for each predictor, the mean absolute percentage error and normalized root-mean-squared error are computed based on the last prediction errors. Then, the deviating models are removed. Once the training phase is completed, the best k predictors are selected to reduce the computational load (memory and speed) of the model. Within the subset of the best predictors, at least one with a forgetting factor equal to one is included to ensure that the ensemble retains a non-forgetting model. All k predictors are trained and used to produce the corresponding predictions for each current instance. Then, the mean of the best three predictions is computed and output as the final prediction.

3. *Test:* The testing phase begins by reproducing the same steps as the training phase, but with k learners. Then, if the performance of any of the predictors has degraded, new OR-ELMs are initialized and trained according to the training process described above.

4.3.2.2. EOR-ELM ensemble structure

The structure of the proposed ensemble is detailed below, describing the decisions taken at each step and their justification.

- *Learner selection*: As one of the objectives of this chapter and the Thesis itself is to show the usefulness of the framework, a prediction-based algorithms has been selected as the base learner, specifically the OR-ELM algorithm.

- *Diversity*: The major problem of the OR-ELM algorithm for unsupervised learning is choosing the correct values of the forgetting factor and the number of hidden neurons. Hence, to overcome this issue and increase the diversity, a random hyperparameter tuning strategy was used. Moreover, the algorithm has a randomization process on its own to generate its initial weights.

- *Base learner selection*: The selection is an optional step, but it can further improve the results. In the case of this particular ensemble, it is imperative to detect the corrupted base learners and eliminate them as soon as possible. Therefore, in order to avoid deviating base learners and additionally reduce the computational cost and select the

best base learners for fusion, three different selection strategies have been employed:

1. *Pruned selection*: a performance-based selection is made in the training phase to reduce the computational cost and remove deviated base learners. Specifically, the k best performing models are selected based on two prediction evaluation metrics (MAPE and NRMSE).
2. *Dynamic selection 1*: A first dynamic selection strategy is adopted to avoid harmful models. Each iteration checks if any model has deviated and new ones are retrained (with their corresponding selection).
3. *Dynamic selection 2*: A second dynamic selection method is used to avoid overfitting and those models that, for a particular instance, perform poorly. Before the combination stage, at each iteration, the three best models are selected, i.e., those leading to the lowest prediction error.

- *Combination*: The anomaly scores are normalized, so that the available combination options include score-wise, meta-learning, the fusion of multiple combination methods, and the best learner. On the one hand, meta-learning and fusion of multiple combination methods would increase the computational cost, already expensive because of the need for training several OR-ELM models, compromising the speed requirement needed for online processing. On the other hand, choosing the best learner does not work very well since it suffers from overfitting. When the models predict the data too well, even if it is anomalous, it makes the anomaly go unnoticed along the stream, ultimately increasing the rate of false negatives in the detection of anomalies. Ranking methods are left aside as they cannot be applied in an online fashion (and relevant information would be lost). In conclusion, the most suitable combination method is found out to be a mean. In this way, the overfitting problem is avoided and the overall computational efficiency of the detector is not compromised.

4.3.3. EOR-ELM based anomaly detector

To demonstrate the usability of the framework described in Section 4.2, we now describe the procedure to create an EORELM-AD. This procedure, which is illustrated in Figure 4.9, is divided into three steps: (A) online normalization and the creation of training and testing instances, (B) prediction of current data and calculation of prediction error, and (C) computation of anomaly scores and anomaly detection. Before diving into the details of these steps, it is important to note a critical difference between the structure

of the training and testing instances obtained by the OAN approach (see Figure 4.4) and those used by the OR-ELM NN (see Figure 4.7). Since the OAN method requires the current value of x_t to normalize its input features, EORELM-AD predicts the current value x_t instead of x_{t+1}; therefore, the prediction error is computed as usual between the predicted value $\widehat{x}_t$ and the current value x_t. In this manner, the same predictive strategy can be used for all normalization methods.

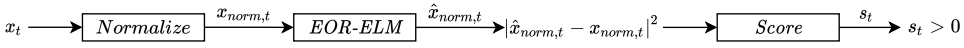


Figure 4.9: Procedure showing how EOR-ELM is adapted to construct an EORELM-AD. First, the current data are normalized and passed to EOR-ELM for training and obtaining a predicted value. Then, the prediction error is computed and the outlier score is calculated. Finally, an anomaly is detected when the outlier score is higher than a given outlier threshold θ.

A. *Normalization and creation of training and testing instances*

Similar to other NNs, the EOR-ELM algorithm requires normalized or standardized inputs. To determine the best normalization method for this particular algorithm, all methods included in the framework are tested and compared to each other.

As mentioned previously, EOR-ELM requires the previous n data values to generate a prediction. Therefore, to normalize the current data and generate the necessary training and testing instances, a normalized sliding window $X_{norm} = \{x_{norm,t-n-1}, \ldots, x_{norm,t}\}$ of size $n+2$ is retained, where n is the input dimension of the network. Given n and X_{norm}, training and testing instances are generated as shown in Figure 4.10.

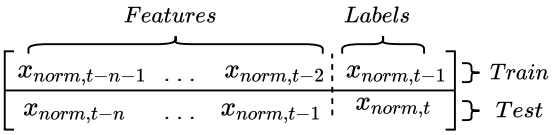


Figure 4.10: Structure of OR-ELM training and testing instances.

We now elaborate on the normalization methods used to generate the normalized sliding window X_{norm} or, in the case of OAN, to directly generate normalized training and testing instances:

- Dynamic z-score Normalization (DN): This method takes the current value x_t as an input and returns its normalized value $x_{norm,t}$. Therefore, every time a new data point arrives, it is standardized and then added to X_{norm}.

- Window Normalization (WN): This is a window-based method that departs from the current value x_t and uses a sliding window of size n. This function stores a sliding window X, which is updated with the arrival of each input sample x_t. Upon arrival, X is standardized to return X_{norm}. Therefore, in this case, the sizes of X and X_{norm} must be the same.
- One-pass Adaptive Normalization (OAN): This method also hinges on the use of sliding windows. As inputs, it takes 1) the current value x_t, 2) size of the disjoint sliding window w (in this case, $w = n + 1$), and 3) size of the sliding time series $n = w + 1$ (in this case, $n + 2$). This normalization method is specific for NNs and directly returns normalized training and testing instances.
- One-pass Adaptive Min-max Normalization (OAMN): This function is also based on a sliding window. As inputs, it takes the current value x_t and window size $n + 2$ to return the normalized window X_{norm}. Additionally, it also takes a threshold θ as input. The optimal value of θ depends on the problem and time series under consideration.

B. *Prediction of current data and calculation of prediction errors*

Given normalized training and testing instances, the EOR-ELM algorithm is trained and used to predict the current sample $x_{norm,t}$, as indicated in Section 4.3.2. After obtaining the predicted value $\widehat{x}_{norm,t}$, the squared prediction error is computed as $|\widehat{x}_{norm,t} - x_{norm,t}|^2$.

C. *Outlier scoring and detection*

Given the prediction error and outlier threshold θ, an outlier score s_t is calculated and used to declare whether an outlier has occurred (i.e., if $s_t > \theta$, then the current data instance is an outlier). All scoring methods take the current prediction error as an input. Window-based methods have two additional input parameters, namely the minimum window size to start calculating anomalies and the maximum window size. The more the available data, the more accurate the detection result, but the greater the computational and memory resources required for calculation.

4.4. EXPERIMENTAL SETUP

A comprehensive experimental setup was developed using several benchmark datasets to provide informed answers to the RQs posed in the introduction of this chapter, as listed below.

- RQ1: Which streaming normalization and outlier scoring methods perform best when used within the proposed EORELM-AD approach?

- RQ2: Does EORELM-AD perform competitively when compared to other TSOD methods from the literature?

The goal of the first experiment is to determine the normalization and scoring methods that work best in conjunction with EOR-ELM. Next, a second experiment is conducted to compare the performance of EORELM-AD (using the best normalization and scoring methods) to that of the detectors included in the *otsad* package described in Chapter 3, as well as some state-of-the-art algorithms reported in the literature related to TSOD. Additionally, a third experiment is conducted to evaluate the time efficiency of EORELM-AD (i.e., time required to train the model and determine whether an input data point is an outlier). This section presents the details of the experimental setup that are common to the experiments later discussed in Section 4.5.

4.4.1. Datasets

To design experiments capable of providing informed answers to the above RQs, 52 labeled one-dimensional time series from different fields available in the OTSAD R package introduced in Chapter 3 are considered. These time series correspond to those contained in the following categories of the Numenta Anomaly Benchmark (NAB) repository [125]. The reader is referred to Section 3.2 for further details. Specifically:

- `artificialWithAnomaly`: Synthetically generated time series data with varying types of anomalies.
- `realAdExchange`: Composed of online advertisement clicking rates, where the metrics are cost per click and cost per thousand impressions.
- `realAWSColudwatch`: Built upon AWS server metrics.
- `realKnownCauses`: Data such as hourly registered taxi schedules in New York or CPU utilization.
- `realTraffic`: Real-time traffic data from the Twin Cities Metro area of Minnesota (USA) with occupancy, speed, and travel time data from specific sensors.
- `realTweets`: A collection of Twitter mentions of large publicly traded companies such as Google and IBM.

4.4.2. Evaluation metrics

To measure the performance of the detectors, classical measures (precision, recall, and F-measure) and the NAB scoring method [125] introduced in Section 3.2.2 are used. Classical measures are widely used in imbalanced

classification or anomaly detection problems. Precision reflects the ratio of detected anomalies that are labeled as anomalous within a dataset. Recall quantifies the proportion of true cases among the anomalies detected by every algorithm. Finally, F-measure is defined as the harmonic mean of precision and recall. These measures can be computed as:

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{4.31}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{4.32}$$

$$\text{F-measure} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{4.33}$$

where true positive (TP) refers to labeled anomalies that are detected as such by the detector, FP refers to the number of non-anomalous data points that are incorrectly detected as anomalies by the detector, and false negative (FN) is the number of labeled anomalies that are not identified as such by the detector.

4.4.3. Implementation details

All the source code for implementing the proposed framework and EORELM-AD was developed in R and included in the *otsad* package. This package is publicly available in a GitHub repository at `https://github.com/alaineiturria/otsad`. The user manual can be found at `https://github.com/alaineiturria/otsad/blob/master/vignettes/otsad.pdf`.

4.5. RESULTS AND DISCUSSION

We now present and discuss the results obtained from the experiments. For the sake of clarity and correspondence with the formulated RQs, this section is divided into comparisons between normalization and scoring methods (Section 4.5.1. RQ1), a performance evaluation of the proposed EORELM-AD compared to the detectors included in the *otsad* library (Section 4.5.2. RQ2, part 1), and a performance comparison of EORELM-AD to some state-of-the-art TSOD algorithms retrieved from the literature (Section 4.5.3. RQ2, part 2). Finally, the results of the study on the time efficiency of EORELM-AD are presented in Section 4.5.4.

4.5.1. RQ1: Which streaming normalization and outlier scoring methods perform best when used within the proposed EORELM-AD?

One of the most important steps in constructing an anomaly detector is to determine which normalization and scoring methods are the most suitable

for the selected prediction algorithm. With this goal, different detectors initialized with different normalization and scoring functions available in the proposed framework were compared. The results of standardizing all datasets before using EORELM-AD were also included and are labeled as "none" in the tables below.

In this first benchmark aimed at answering RQ1, fixed values were defined for all hyperparameters, except for the number of previous samples n and anomaly threshold θ, for each detector. Therefore, each detector had only two hyperparameter values to be tuned. All hyperparameter values in use are listed in Table 4.1.

Table 4.1: Hyperparameter values adopted for the proposed EOREM-AD

Parameters	Fixed	Value	Description
Normalization			
normMethod	no	{None, DN, WN, OAN, OAMN}	Streaming normalization method
Ensemble			
m	yes	30	Number of ORELMs with a forgetting factor lower than 1 to train
k	yes	6	Number of best ORELMs to keep into the test phase
$\tilde{N}$	yes	{20, 25, ..., 45, 50}	Possible values of the number of hidden neurons that can be used to initialize different ORELMs in the ensemble
λ	yes	{0.9, 0.91, ..., 1}	Possible output forgetting factor values that can be used to initialize different ORELMs in the ensemble
n	no	{50, 100, 150}	Number of past instances used to predict the current instance value
Anomaly detection			
scoringMethod	no	{AL, DT, SS, DSS}	Streaming anomaly scoring method
wnMax	yes	2000	Maximum window size to compute the anomaly scores (only for window based scoring methods)
wnMin	yes	100	Minimum window size to start to compute the anomaly scores (only for window based scoring methods)
θ	no	{0.50, 0.55, ..., 0.95, 0.99, 0.999, 0.9999, 0.99999, 1}	Threshold used to determine if the current instance is an anomaly. If the anomaly score is higher than the threshold an anomaly is reported

To determine which normalization and scoring functions perform best, two different factors were analyzed. First, we examined which one of the algorithms performed best. Comparisons were performed in the same manner as some recent works [68, 145] and a grid search was performed to find the values of the hyperparameters that worked best for all test cases. Then, the same n and θ values were used for all time series datasets. This experimental methodology is appropriate when there is insufficient domain knowledge to infer which specific configuration (particularly in terms of n and θ) will best fit the characteristics and dynamics of the time series under consideration in

advance. Second, we evaluated the performances of all possible EROELM-AD configurations (see Table 4.1) and recorded the best-performing configuration for each dataset.

In the first strategy, all detectors were tested using different n and θ values. Then, the hyperparameter values that performed best for all datasets were selected and used for comparisons. Similar to [161], we compared the results before and after FP reduction (using the method available in *otsad* with a one-day window size). Based on the random initialization of weights in EOR-ELM, the results may vary slightly. Therefore, we independently conducted the same experiment 10 times and recorded the average score values computed over all trials. Only the values of the NAB measure are reported to improve the clarity and simplicity of discussion. The obtained results are summarized in Table 4.2 and presented as a chart in Figure 4.11 (only positive scores are presented for ease of interpretability and comparison).

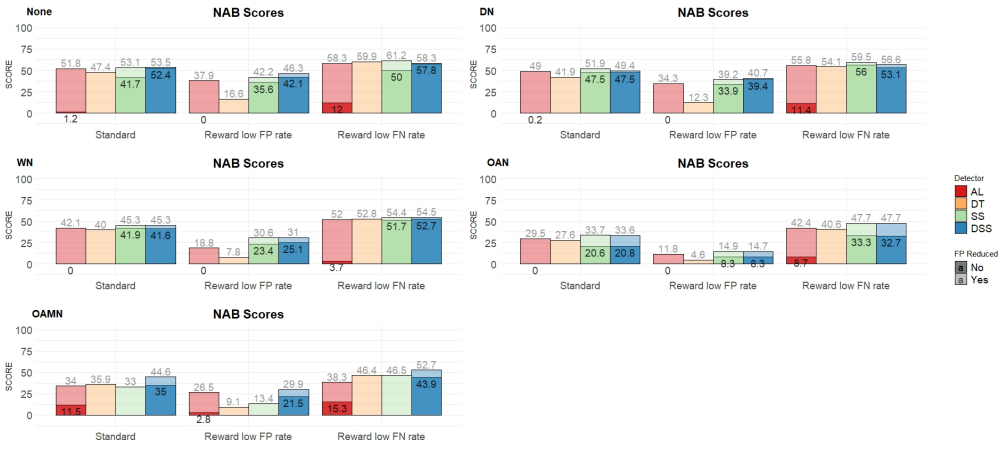


Figure 4.11: General performance benchmark of EORELM-AD using the NAB scoring method corresponding to the global-best-performing hyper-parameter setting over all datasets. Translucent colors and numbers underlying the plots represent the scores obtained after the FP reduction method is applied. In contrast, opaque colors correspond to the scores obtained without FP reduction.

Table 4.2: Comparison of NAB scores obtained over the NAB using different normalization and scoring methods for the EORELM-AD with the optimal hyperparameter configuration for all time series datasets. The best scores are highlighted in bold and the scores of the best-scoring method for each normalization technique are underlined.

Norm.	Scoring	Standard	Low FP	Low FN	Reduced Standard	Reduced Low FP	Reduced Low FN
None	AL	1.25	0.00	11.96	51.84	37.85	58.27
None	DT	-624.18	-1338.22	-384.91	47.39	16.59	59.92
None	SS	41.68	35.62	50.03	53.13	42.17	**61.19**
None	DSS	**52.38**	**42.12**	**57.83**	**53.49**	**46.34**	58.26
DN	AL	0.18	0.00	11.41	49.05	34.32	55.81
DN	DT	-114.96	-315.41	-46.87	41.93	12.28	54.12
DN	SS	47.52	33.89	55.98	51.86	39.16	59.54
DN	DSS	47.53	39.41	53.09	49.38	40.73	56.63
WN	AL	0.00	0.00	3.74	42.09	18.82	51.99
WN	DT	-78.51	-243.52	-22.02	40.05	7.85	52.76
WN	SS	41.91	23.40	51.67	45.34	30.60	54.41
WN	DSS	41.64	25.07	52.70	45.30	30.98	54.50
OAN	AL	0.00	0.00	4.03	32.41	11.06	44.38
OAN	DT	-17.40	-72.56	1.93	26.08	8.75	38.94
OAN	SS	23.45	8.39	34.97	34.48	15.59	46.93
OAN	DSS	22.49	8.17	34.29	35.35	16.21	48.21
OAMN	AL	11.54	2.79	15.28	34.01	26.47	38.32
OAMN	DT	-88.82	-252.53	-32.88	35.87	9.13	46.39
OAMN	SS	-497.14	-1005.78	-327.32	33.01	13.43	46.53
OAMN	DSS	35.03	21.47	43.93	44.59	29.92	52.74

Regarding the second strategy, a similar procedure was followed to determine which detector best adapts to the characteristics of each time series dataset. In this case, the best hyperparameter values were considered for each dataset. The results are summarized in Table 4.3 and presented in Figure 4.12 according to the same visualization criteria used in Figure 4.11.

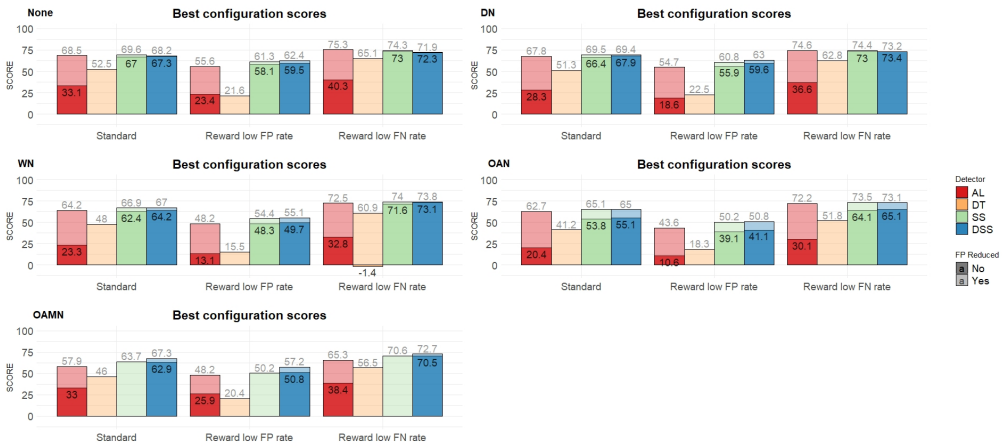


Figure 4.12: Best performance benchmark of EORELM-AD using the NAB scoring method, where the optimal hyper-parameter setting for each dataset is used. The interpretation of colors and numbers follows the same convention as in Figure 4.11.

We first focus our discussion on the relevance of the normalization methods. Figure 4.11 reveals that performance is closely related to the selected normalization function. DN yields results similar to those obtained by normalizing all datasets before applying the detector. In Figure 4.12, one can see that the results of detectors that utilize window-based normalization are generally significantly better than those with general parameters. Therefore, we can conclude that the poor detection results of detectors using window-based normalization can be attributed to the window size used for normalization, which depends on the parameter n. According to the results of these two benchmarks, we arrive at the conclusion that DN is the best normalization technique for EORELM-AD based on its independence of the parameter n, as well as its low computational requirements.

We follow the discussion by commenting on the comparison between different streaming outlier scoring methods. To this end, Figure 4.11 and Figure 4.12 reveal three methods that stand out from the rest: AL, SS, and DSS. AL achieves slightly worse results than the other two. Furthermore, its performance depends significantly on the technique used for reducing FPs. In contrast, SS and DSS exhibit similar performance, where SS is slightly better under the standard and reward false negative NAB scoring profiles. However, DSS depends less on the FP reduction method, in addition to being faster and non-parametric.

Table 4.3: Comparison of NAB scores obtained by EORELM-AD configured with different normalization and scoring methods for the NAB using the optimal hyperparameters for each dataset. The best detector scores are highlighted in bold and the scores of the best-scoring method for each normalization technique are underlined. The best scores for each category are underlined and the best scores between no FP reduction and FP reduction are highlighted in bold.

Norm.	Scoring	Standard	Low FP	Low FN	Reduced Standard	Reduced Low FP	Reduced Low FN
None	AL	33.12	23.45	40.30	68.47	55.60	75.35
None	DT	-594.08	-1277.81	-364.73	52.51	21.63	65.11
None	SS	67.03	58.14	72.99	**69.60**	61.34	74.27
None	DSS	67.30	59.50	72.29	68.23	62.37	71.86
DN	AL	28.30	18.65	36.61	67.76	54.68	74.58
DN	DT	-97.80	-279.98	-35.60	51.30	22.53	62.83
DN	SS	66.39	55.93	73.02	69.48	60.75	**74.40**
DN	DSS	**67.93**	**59.62**	**73.41**	69.38	**62.95**	73.19
WN	AL	23.28	13.12	32.76	64.16	48.21	72.47
WN	DT	-48.12	-183.46	-1.35	47.96	15.46	60.88
WN	SS	62.42	48.31	71.59	66.91	54.38	74.04
WN	DSS	64.25	49.72	73.12	67.02	55.12	73.78
OAN	AL	21.84	12.64	31.23	60.53	43.09	69.80
OAN	DT	3.06	-38.89	19.95	41.68	22.30	51.20
OAN	SS	53.17	38.53	63.89	64.20	48.66	73.26
OAN	DSS	54.54	39.15	65.34	64.79	49.66	73.66
OAMN	AL	33.05	25.86	38.41	57.85	48.19	65.27
OAMN	DT	-61.63	-199.85	-13.19	46.02	20.36	56.50
OAMN	SS	-450.30	-969.50	-273.83	63.72	50.24	70.55
OAMN	DSS	62.85	50.80	70.49	67.27	57.19	72.72

To support the conclusions of this first set of experiments, an additional comparison was conducted to determine which of these three scoring methods performs best. The classic F-measure was adopted to produce the results summarized in the two tables below. Table 4.4 presents the F-measure results when using general hyperparameter values, whereas Table 4.5 presents the results obtained by using the optimal hyperparameter values for each dataset. Since our prior analysis demonstrated that DN performs the best as a normalization method, only the corresponding results are presented in the tables below for simplicity.

By analyzing Table 4.4 and Table 4.5, one can see that the results obtained without using FP reduction are better in the cases of SS and DSS. However, this could be because the adopted FP reduction method prioritizes the

Table 4.4: Average F-measure values obtained by DN and the top-three EORELM-AD detectors for the NAB using the general hyperparameter settings for all datasets. The best scores for each category are underlined and the best scores between no FP reduction and FP reduction are highlighted in bold.

Norm.	Scoring	Precision	Recall	F-measure	Reduced Precision	Reduced Recall	Reduced F-measure
DN	AL	0.015	**0.483**	0.027	0.112	0.200	0.114
DN	SS	0.261	0.284	0.229	0.270	0.223	0.218
DN	DSS	**0.279**	0.274	**0.238**	0.270	0.186	0.195

Table 4.5: Comparison of average F-measure values obtained by DN and the top-three EORELM-AD detectors for the NAB using the optimal hyperparameters for each dataset.

Norm.	Scor.	Precision	Recall	F-measure	Reduced Precision	Reduced Recall	Reduced F-measure
DN	AL	0.020	**0.767**	0.035	0.141	0.303	0.154
DN	SS	0.338	0.561	0.342	0.337	0.406	0.326
DN	DSS	**0.372**	0.579	**0.369**	0.371	0.403	0.339

first anomaly detected (similar to the NAB measure) and labels subsequent detected anomalies within a short time period as FPs. Overall, these results corroborate the idea that DSS and SS perform better than AL. Furthermore, DSS provides better generalization and adaptation capabilities compared to SS, as indicated by the reported experimental outcomes, leading to the overall conclusion that the DSS scoring technique proposed in this study has the best performance among the scoring techniques considered in the proposed framework.

4.5.2. RQ2, part 1: Does EORELM-AD perform competitively when compared to TSOD methods from the *otsad* library?

In this section, the experimentation from a previous work Section 3.3 [161] is extended by comparing the performance of the best EORELM-AD detector to the following detectors included in the *otsad* package.

The procedure for designing this experiment was the same as the previous experiments. EORELM-AD with DN and DSS is selected as the best anomaly detector. The global optimal hyperparameter values are summarized in Table 4.6. The hyperparameter values used for the *otsad* detectors are reported in [161], which were also found by means of a grid search strategy. Following the same comparison methodology, each detector's detection and specific problem adaptation skills were evaluated. In this comparative study,

only the NAB measure was considered for the sake of a focused and clear discussion.

Table 4.6: NAB scores obtained by detectors in the otsad package and the proposed EORELM-AD approach on the NAB using global optimal hyperparameter values for all datasets. The highest scores are highlighted in bold.

Detector	Standard	Low FP	Low FN	Reduced Standard	Reduced Low FP	Reduced Low FN
PEWMA	16.80	-30.50	33.90	37.80	18.90	51.10
SD-EWMA	39.90	17.80	50.40	49.20	34.40	56.70
TSSD-EWMA	42.30	29.90	49.00	47.10	35.50	53.90
KNN-ICAD	-2.60	-80.10	24.70	59.20	45.10	65.90
KNN-LDCD	-2.60	-80.10	24.70	**59.20**	45.10	**65.90**
CAD-OSE	**56.50**	**41.50**	**63.20**	57.50	**48.50**	62.20
EORELM-AD	47.53	39.41	53.09	49.38	40.73	56.63

Table 4.7: NAB scores obtained by detectors in the *otsad* package and the proposed EORELM-AD over the NAB using the optimal hyperparameter values found for each dataset. The highest scores are highlighted in bold.

Detector	Standard	Low FP	Low FN	Reduced Standard	Reduced Low FP	Reduced Low FN
PEWMA	42.20	-7.10	60.50	66.70	45.30	75.50
SD-EWMA	61.50	37.40	72.60	68.70	53.20	75.90
TSSD-EWMA	**74.70**	**61.10**	**81.40**	**75.70**	66.80	**80.80**
KNN-CAD	6.30	-64.30	31.80	66.20	54.80	72.00
KNN-LDCD	6.20	-64.30	31.70	66.10	54.80	72.00
CAD-OSE	72.20	62.90	78.30	74.60	**67.70**	79.10
EORELM-AD	67.93	59.62	73.41	69.38	62.95	73.19

The results are summarized in Table 4.6 and Table 4.7, and visualized in Figure 4.13.A (generalization across different datasets) and 4.13.B (adaptability to a given dataset). Figure 4.13.A indicates that EORELM-AD outperforms the PEWMA, SD-EWMA, and TSSD-EWMA detectors, and achieves competitive results compared to KNN-CAD, KNN-LDCD, and CAD-OSE. Figure 4.13.B indicates that EORELM-AD with only two hyperparameters (n and θ) outperforms PEWMA, SD-EWMA, and KNN-based detectors. Additionally, it is noteworthy that TSSD-EWMA is not a fully online anomaly detector because it incorporates a lag of several points in its detection procedure to reduce FPs. Therefore, it is fair to say that EORELM-AD is the second-most flexible algorithm in this first benchmark.

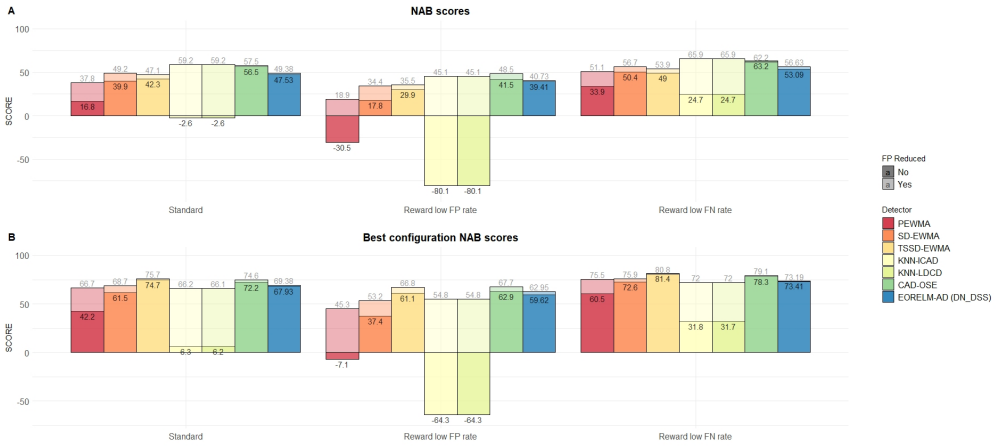


Figure 4.13: Performance benchmark of EOR-ELM-AD and otsad anomaly detectors. Transparent colors and numbers represent the scores obtained after the FP reduction method is applied. In contrast, opaque colors correspond to the scores obtained without FP reduction.

4.5.3. RQ2, part 2: How does EORELM-AD perform compared to other state-of-the-art TSOD methods?

In this subsection, we focus on the comparison of EORELM-AD to state-of-the-art TSOD methods reported in the recent literature. To this end, we extend the benchmarks presented in [97, 145] to demonstrate the capabilities of the proposed framework and algorithm.

Table 4.8: Comparison of average F-measure values obtained for unsupervised anomaly detection of streaming time series data and the proposed EORELM-AD for the NAB (the results for methods marked with * are given in [145]). The highest scores for each dataset category are highlighted in bold.

Dataset category	*Bayesian Change-point**	*Context OSE**	*EXPoSE**	*HTM JAVA**	*KNN CAD**	*Numenta**	*NumentaTM**	*Relative Entropy**	*Skyline**	*Twitter ADVec**	*Windowed Gaussian**	*DeepAnT**	*OeSNN-UAD**	*b.EORELM-AD (proposed)*	*g.EORELM-AD (proposed)*
Artificial no Anomaly	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Artificial with Anomaly	0.009	0.004	0.004	0.017	0.003	0.012	0.017	0.021	0.043	0.017	0.013	0.156	**0.427**	0.186	0.161
Real ad Exchange	0.018	0.022	0.005	0.034	0.024	0.040	0.035	0.024	0.005	0.018	0.026	0.132	0.234	**0.564**	0.237
Real AWS Cloud	0.006	0.007	0.015	0.018	0.006	0.017	0.018	0.018	0.053	0.013	0.06	0.146	0.369	**0.472**	0.346
Real Known Cause	0.007	0.005	0.005	0.013	0.008	0.015	0.012	0.013	0.008	0.017	0.006	0.2	**0.324**	0.102	0.095
Real Traffic	0.012	0.02	0.011	0.032	0.013	0.033	0.036	0.033	0.091	0.020	0.045	0.223	0.340	**0.497**	0.252
Real Tweets	0.003	0.003	0.003	0.010	0.004	0.009	0.010	0.006	0.035	0.018	0.026	0.075	0.310	**0.391**	0.337

The results of this comparison are summarized in Table 4.8, where the performance of EORELM-AD with DN and DSS for the NAB is compared to that of the other unsupervised anomaly detection methods and algorithms from the literature. Similar to the benchmark results presented recently in [97] and [145], the mean F-measure obtained for each category of the Numenta data files is reported. The quantitative scores in both [97] and [145] were obtained using the optimal hyperparameter settings for each dataset.

Because it is important to determine the generalization and adaptation skills of the algorithms, in this table, we consider the two hyperparametric configurations of the EORELM-AD detector explained previously, namely the best hyperparameter setting for each dataset (identified as b.EORELM-AD, where b. is the prefix used to refer to best setting for each dataset) and the global optimal hyperparameter settings for all datasets ($n = 150$, $\theta = 1$), identified as g.EORELM-AD, where g. stands for global best).

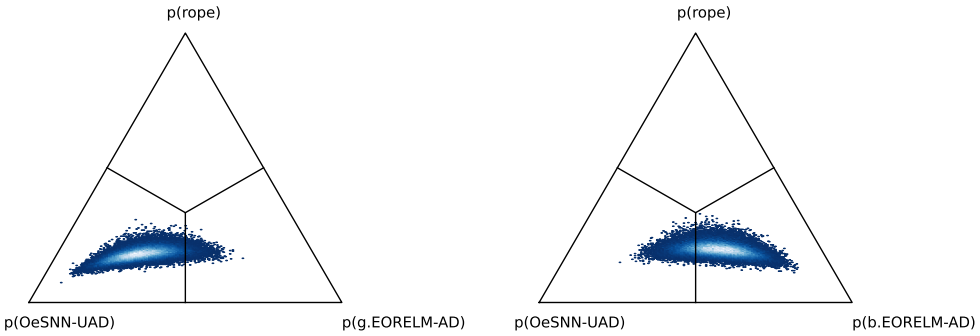
When analyzing these performance scores, one can see that when considering the best hyperparameter values for every dataset (b.EORELM-AD), EORELM-AD significantly outperforms the other approaches for most of the categories in the Numenta benchmark, ranking first in four of the categories, and second and third in the other two. Furthermore, when considering the global optimal hyperparameter settings (g.EORELM-AD), EORELM-AD is the second-best method in all categories except for the `realKnownCause` category, where it is the third-best method.

To assess the statistical significance of the performance gaps reported in Table 4.8, we focused on the two configurations of the proposed EORELM-AD approach (namely b.EORELM-AD and g.EORELM-AD) and OeSNN-UAD, which is the model performing most similarly to EORELM-AD according to the scores reported in the above table. We first analyze the output of a Wilcoxon signed-rank test applied to the F-measure values obtained by every pairing of these methods. Based on a confidence value α, this non-parametric hypothesis test determines whether the median difference between paired F-measures is zero. Table 4.9 indicates that except for the case where OeSNN-UAD and g.EORELM are compared to each other, the null hypothesis can be rejected for a significance level equal to $\alpha = 0.05$. Therefore, we can conclude that the performance gaps are statistically significant for OeSNN-UAD and b.EORELM-AD when they are compared to g.EORELM-AD. However, when comparing OeSNN-UAD and b.EORELM-AD, a closer inspection of performance differences is required.

Table 4.9: Results of the Wilcoxon signed-rank test applied to the results of the best models reported in Table 4.8.

	g.EORELM-AD	b.EORELM-AD	OeSNN-UAD
g.EORELM-AD	-	$< 10^{-7}$	0.040
b.EORELM-AD	-	-	0.330
OeSNN-UAD	-	-	-

Following the guidelines in [162], we proceeded by performing Bayesian analysis on the differences in F-measure between OeSNN-UAD and both b.EORELM-AD and g.EORELM-AD. This analysis yields an adjusted posterior probability that according to the given performance measurements, one approach performs better than the other (or vice versa) or that such per-



(a) OeSNN-UAD vs g.EORELM-AD *(b)* OeSNN-UAD vs b.EORELM-AD

Figure 4.14: Sampled posterior distributions (10^5 points) of (a) OeSNN-UAD vs. g.EORELM-AD and (b) OeSNN-UAD vs. b.EORELM-AD.

formance measurements fall within a so-called region of practical equivalence. This region is delimited by a threshold (denoted as *rope*) that imposes the minimum difference in terms of F-measure for the compared models to be declared equivalent. Figure 4.14 presents the sampled posterior distribution (10^5 points) of OeSNN-UAD versus g.EORELM-AD (a) and b.EORELM-AD (b) corresponding to a rope of 0.05. One can see that OeSNN-UAD is the clear winner when compared to g.EORELM-AD with region-based accumulated probabilities equal to P(OeSNN-UAD $>$ g.EORELM) $=$ 0.97368 and P(g.EORELM $>$ OeSNN-UAD) $=$ 0.02632. However, when examining the comparison between OeSNN-UAD and b.EORELM-AD, the posterior distribution is skewed toward b.EORELM-AD with an imbalanced distribution of P(OeSNN-UAD $>$ b.EORELM) $=$ 0.1395 versus P(b.EORELM $>$ OeSNN-UAD) $=$ 0.8605. This indicates that on average across different datasets, b.EORELM-AD performs better than OeSNN-UAD. No cases of practical equivalence were identified (P(*rope*) $=$ 0.0).

These results demonstrate the competitive performance of the developed method, as well as the potential of the proposed framework for easing the discovery of novel online TSOD approaches based on prediction models.

4.5.4. Time efficiency of the EORELM-AD algorithm

We conclude our analysis by inspecting the time required by different approaches to process data in an online or incremental fashion. Specifically, the execution time required for processing a unique value is studied. To this end, a synthetic dataset of $1,000$ points is considered and the time required to process each dataset is reported. All other parameters are configured according to the best general configuration obtained in the previous experiments. The results of this benchmark are summarized in Table 4.10.

As shown in Table 4.10, although EORELM-AD is slightly slower than PEWMA, SD-EWMA, TSSD-EWMA, and KNN-ICAD, all models are able to process data with a collection period of less than half of a second.

Table 4.10: Time efficiency over one point processing using incremental processing algorithms for online TSOD.

	Minimum	Mean	Maximum
PEWMA	0	0.00256	0.08
SD-EWMA	0	0.00048	0.02
TSSD-EWMA	0	0.00980	0.02
KNN-ICAD	0	0.00177	0.25
CAD-OSE	1.64	1.87000	2.41
EORELM-AD	0	0.01000	0.12

Therefore, we conclude that EORELM-AD is efficient for online processing because the smallest recollection period among the NAB datasets is 5 s. Additionally, based on its ensemble structure, EORELM-AD can be easily parallelized to reduce processing time further.

4.6. CONCLUDING REMARKS

This chapter has presented presented a novel outlier detection framework that facilitates the adaptation of any online time series prediction algorithm to construct an online outlier detection algorithm for time series covering both streaming data normalization and online anomaly scoring. The proposed framework not only defines the methodological steps required to realize this adaptation, but also provides different algorithmic options to implement them in practice. In addition to the framework itself, its application yielded a novel anomaly detection algorithm called EORELM-AD, which is a more robust ensemble-based variant of the OR-ELM prediction algorithm. Extensive experiments have been conducted to answer two different RQs:

- RQ1) Which online normalization and online scoring methods perform best when used with the proposed EORELM-AD?

  Our first round of experiments demonstrated that the normalization method that works best for the proposed detector is DN, which is parameter-free and requires very little computational resources. Regarding the outlier scoring method, the DSS technique proposed in this study provides the best performance. It is fast and parameter-free, and its performance does not depend on the FP reduction technique.

- RQ2) How does EORELM-AD perform when compared to state-of-the-art TSOD methods?

  Our second round of experiments revealed that the performance of the EORELM-AD detector is competitive with that of some state-of-

the-art TSOD algorithms, and that it outperforms them on certain dataset categories.

Based on our experiments and insights drawn from our study on the time efficiency of EORELM-AD, we can conclude that the proposed framework can serve as a referential tool for the community to adapt online time series prediction algorithms to outlier detection.

Several future research lines can be outlined based on the findings reported in this chapter. Among them, it is worth highlighting the inherently parallel structure of the proposed EORELM-AD approach, which should enable its application to ultra-high-rate time series data. Additionally, other recurrent learning models with efficient training methods (e.g., recurrent broad learning systems or echo state networks) could be also considered to develop novel outlier detection algorithms over time series that are compliant with the computational requirements of online settings.

Part III

FINAL REMARKS

5

CONCLUSIONS, PUBLICATIONS, AND FUTURE RESEARCH DIRECTIONS

5.1. CONCLUSIONS

Due to its recent interest and growing demand, this Thesis has focused on anomaly detection over streaming univariate time series. Several algorithmic proposals have been proposed in the last few years to tackle this learning task from different formulations and perspectives. However, a thorough study of the state of the art in this research area reveals several issues that remain open to date: the scarcity of open-source software to support further knowledge advances, and the high false positives and false negatives rates observed for state-of-the-art proposals in benchmarks and studies reported in the literature. This Thesis has gravitated around these noted issues, proposing several original contributions aimed to overcome these two problems.

To this end, a first study of the state of the art related to online time series anomaly detection methods has been performed and summarized in Chapter 2. On the one hand, this initial study has permitted to set clear grounds in what refers to the modeling implications of classical outlier detection problems, time series outlier detection, and online processing. On the other hand, the critical examination of the literature has identified specific characteristics of these problems, taxonomies of different aspects of the techniques proposed so far to tackle them efficiently, and a prospect of challenges and research opportunities. The analysis unveils that despite the emerging demand of anomaly detection approaches for online time series, scarce proposals and techniques can be currently found within the community.

To advance over this noted shortage of proposals, the Thesis has hypothesized the use of ensemble techniques. Although they have been widely studied and used for this purpose in other fields, ensembles for anomaly detection are still a recent line of research. For this reason, the Thesis has carried out a literature review of ensemble methods for anomaly detection, paying particular attention to the techniques that have been – or can be – used in time series

detection and online processing. This literature study has identified different methodological steps to build an ensemble, the different techniques available for this purpose, and the challenges posed by the implementation of each of these steps under the computational constraints imposed by online time series anomaly detection. After this study, ensembles have been found to be rarely applied to online outlier time series detection. However, ensembles can help strengthen detectors, and it is possible to generate new ensemble detection approaches by following the steps and techniques identified in this literature review.

Once all the bibliographic study was done and departing from the conclusions drawn therefrom, the first significant contribution of this Thesis (Chapter 3) has been the implementation of an efficient and easy-to-use R library named *otsad* and the comparative study of the implemented online anomaly detectors for time series implemented in this software package. *otsad* is the first R package that collects a set of online time-series anomaly detectors: PEWMA, SD-EWMA, TSSD-EWMA, KNN-LDCD, KNN-CAD, and CAD-OSE. It also implements a new false positive reduction technique to improve detectors' results significantly. Inspired by a real-life situation where there is a time lapse between an alarm being triggered and until corrective action is taken, our proposal uses the number of processed data points between two detected anomalies to reduce the number of false positives. Furthermore, it also includes some advanced functionalities, such as the NAB detector measurement technique and a visualization function. Finally, a comparative study of the effectiveness and efficiency of the implemented methods has been carried out to showcase the inherent utility of the developed package to support and stimulate further research in online TSOD. Besides revealing the best performing detectors among those included in *otsad*, the conducted experiments have revealed that the performance of the algorithms is linked to the use of false positive reduction techniques and the hyperparameter values chosen to process each dataset, proving that the proposed false positive reducer is effective for this purpose. Furthermore, the comparative study has verified that all detection algorithms for TSOD give high false positive and negative rates, and that all of them have a high dataset dependency. Therefore, we conclude that better strategies are still in urgent need to surpass the relatively poor performance of current algorithmic proposals, and that there is no algorithm better than another when considering many diverse datasets: the winning proposal in a performance benchmark depends stringently on the dataset in question.

These last two observations have paved the way towards the last contribution of the Thesis (Chapter 4), which breaks down into two achievements. First, a framework has been proposed in Section 4.2 to allow the generation of new online time series outlier detection algorithms by adapting available algorithms for time series prediction to the online detection of anomalies. The devised framework aims to extrapolate advances made in online time

series forecasting to anomaly detection, and thus embodies an useful tool to quickly and systematically expand the number and diversity of online anomaly detectors for streaming time series. The proposed framework implements several online normalization and outlier scoring methods already available in state-of-the-art models, as well as novel proposals designed to improve upon baselines. Specifically, two novel normalization methods have been proposed – one-pass adaptive normalization (OAN) and one-pass adaptive min-max normalization (OAMN) – as well as two scoring methods, namely, sigma scoring (SS) and dynamic SS (DSS).

Then, Section 4.3 and subsections thereafter have shown the usability and efficacy of the proposed framework by adapting a novel ensemble-based online recurrent extreme learning machine, EORELM-AD, to deal with the online detection of anomalies in streaming time series. EORELM-AD has been created by implementing the steps of the proposed framework over an ensemble of Online Recurrent Extreme Learning Machines. The proposed ensemble combines several instances initialized with different parameter settings. In this manner, the hyperparameter selection problem is circumvented, reducing the dependency of the model's configuration on the target dataset. Furthermore, EOR-ELM removes deviating models at each iteration by initializing new models, so it can quickly adapt to distribution changes and can significantly reduce false positives. Therefore, EORELM-AD provides a much more robust approach to evolving time series data and the presence of different anomalies in time series data. Experiments performed to evaluate the performance of EORELM-AD have verified that the performance of the EORELM-AD detector is competitive with respect to some state-of-the-art online time series outlier detection algorithms, performing best on specific dataset categories. Based on the obtained experimental results and the insights drawn from the study on the time efficiency of EORELM-AD, it is fair to conclude that the proposed framework can serve as a referential tool for the community to adapt online time series prediction algorithms to outlier detection, and that ensembles can be efficient online anomaly detectors that can effectively reduce false positives and perform on par with other avant-garde methods.

On a summarizing note, this Thesis has delved into the anomaly detection problem when formulated over streaming time series data, contributing with several achievements to the general knowledge in this research area: i) a thorough study of the state of the art in different subareas intersecting with this topic; ii) a solid understanding of the requirements imposed by online processing; iii) a detailed exposition of the methodological steps and algorithmic components involved in an algorithmic proposal devised to tackle this problem; iv) the proposal of a public software package that collects and unifies the implementation of approaches for online anomaly detection over time series data; and finally, v) the application of the knowledge acquired during the Thesis to realize a framework to produce innovative proposals

based on time series prediction algorithms, and a novel ensemble anomaly detection that results from the application of the aforementioned framework. These contributions comprise a Thesis that aims to ultimately provide evidence about the challenging nature of online anomaly detection as a data-based modeling task, and to stimulate its readership towards continued research efforts in this exciting and rapidly evolving field.

5.2. PUBLICATIONS

The main results stemming from the investigations pursued in the Thesis have given rise to several publications in top JCR-indexed journals related to the wide fields of Machine Learning and Artificial Intelligence. Details of the two journal articles presenting the technical contributions of the Thesis are next given:

- Alaiñe Iturria, Jacinto Carrasco, Santi Charramendieta, Angel Conde, and Francisco Herrera. «otsad: A package for online time-series anomaly detectors.» In: Neurocomputing 374 (2020), pp. 49–53.
  - Status: **Published**
  - Impact factor (JCR 2019): 4,438
  - Subject Category: Computer Science, Artificial Intelligence. Ranking 28/137 (**Q1**)
- Alaiñe Iturria, Jokin Labaien, Santi Charramendieta, Aizea Lojo, Javier Del Ser, and Francisco Herrera. «A framework for adapting online prediction algorithms to outlier detection over time series.» In: Knowledge-Based Systems 256 (2022), p. 109823.
  - Status: **Published**
  - Impact Factor (JCR 2021): 8,139
  - Subject Category: Computer Science, Artificial Intelligence. Ranking 24/145 (**Q1**)

5.3. FUTURE RESEARCH DIRECTIONS

The research of this dissertation can be continued in many directions. The most important topics that can be referred to as having great potential for further research are shortly discussed below:

- *Study of online data-based diversity generation methods*: most ensembles proposed for online processing are based on static diversity generation methods, performing feature selection or model initialization only once before the model's training process. As explained throughout the

Thesis, the training dataset may change continuously with the arrival of new data in online processing. However, most related works train the base models on the same dataset, being [75] the only contribution proposing to learn an ensemble on dynamically generated subsets of training data with the arrival of new instances. Inspired by this observation, an interesting line of research could be to find a way to generate new subsets of data dynamically, by selecting instances or features while maintaining the temporal correlation between the stream time series data. This aspect is not only interesting for the generation of ensembles, but is also essential for the parallel implementation and processing of Big Data time series.

- *Investigation of ensemble combination methods that take into account the temporal correlation between the data points of the time series*: in the study about outlier detection ensembles, it was observed that no combination technique has considered so far the temporal correlation inherently existing in time series data. A plausible research hypothesis is that combination methods for time series should consider the temporal correlation of the results when combining them to yield a final decision about the outlier nature of arriving data. This being stated, it should be possible to use prediction algorithms to estimate the current point value x_t, and produce a prediction for the following n points, $x_{t+1}, \ldots, x_{t+n}$. In this way, when the data to be predicted is x_{t+n}, n predictions from the previous steps will be available, which can be combined by averaging or weighted averaging and thereby, the impact of anomalies when learning earlier instances can be minimized.

- *Derivation of new algorithms based on heterogeneous ensembles*: as has been showcased in the Thesis, new anomaly detectors can be easily generated by varying the structure of ensembles. In this context, the Thesis has proposed a homogeneous EORELM-AD algorithm whose inter-learner diversity stems mainly from different hyperparameter sets of OR-ELM base detection algorithm. However, it would also be interesting to combine algorithms of different types, such as those implemented in *otsad*, and investigate if it is possible to obtain better detection ensembles that outperforms their individual base learners.

- *Use of parallelization techniques to improve the time efficiency of ensembles*: although ensembles can provide a higher level of robustness, they also require a higher computational cost. Fortunately, the implementation of the learning algorithm of some ensembles can be run efficiently over multiple processors due to their inherently parallel structure. Related to the last contribution of the Thesis, the proposed EORELM-AD detector falls within the subset of ensembles whose structure is appropriate for its parallelization. Further efforts can be

invested towards realizing this implementation and evaluating its use over ultra-high rate time series data.

- *Proposal of new detection algorithms based on other recurrent learning models with efficient training methods*: the framework described in Section 4.2 allows for the use of algorithms from the area of online time series prediction to their use for anomaly detection. Besides OR-ELM, other recurrent learning models with efficient training methods (e.g., recurrent Broad Learning Systems or Echo State Networks) could also be studied to yield novel outlier detection algorithms over time series compliant with the computational requirements of online settings.

- *Study on the adaptability of detection algorithms to time series with outliers of different nature and the incremental characterization of anomalies*: the practical use of anomaly detection algorithms could be complemented in practice with additional mechanisms to consolidate a recurrently appearing anomaly as a learnable pattern so that, when assimilated inside the anomaly detection algorithm, the detection of the consolidated anomaly could be done more reliably in subsequent instants of time along the time series stream.

- *Application of the acquired knowledge and research in online anomaly detection over multivariate time series*: a starting point of this Thesis has been an intended focus on anomaly detection over univariate time series. However, extrapolating the acquired knowledge to tackle online anomaly detection in multivariate time series is a mandatory follow-up research direction to be pursued in the future. Interestingly, some of the algorithms investigated in this Thesis (e.g., SD-EWMA, TSSD-EWMA, and EORELM-AD) already have a variant for multivariate data. Implications of handling multiple variables over the time series in the methodological steps of the proposed framework and the construction of ensemble detectors will be actively investigated, opening up the possibility to create new algorithmic approaches capable of performing competitively over multivariate time series.

BIBLIOGRAPHY

[1] Zhiyou Ouyang et al. «Multi-View Stacking Ensemble for Power Consumption Anomaly Detection in the Context of Industrial Internet of Things.» In: *IEEE Access* 6 (2018), pp. 9623–9631.

[2] Yueyi Yang et al. «Anomaly Detection for Controller Area Network in Braking Control System with Dynamic Ensemble Selection.» In: *IEEE Access* 7 (2019), pp. 95418–95429.

[3] Biao Wang and Zhizhong Mao. «Detecting outliers in industrial systems using a hybrid ensemble scheme.» In: *Neural Computing and Applications* 32.12 (2020), pp. 8047–8063.

[4] Daniel B. Araya et al. «An ensemble learning framework for anomaly detection in building energy consumption.» In: *Energy and Buildings* 144 (2017), pp. 191–206.

[5] Cheng Fan et al. «Analytical investigation of autoencoder-based methods for unsupervised anomaly detection in building energy data.» In: *Applied Energy* 211 (2018), pp. 1123–1135.

[6] Yu Weng, Ning Zhang, and Chunlei Xia. «Multi-Agent-Based Unsupervised Detection of Energy Consumption Anomalies on Smart Campus.» In: *IEEE Access* 7 (2019), pp. 2169–2178.

[7] Juan Vanerio and Pedro Casas. «Ensemble-learning approaches for network security and anomaly detection.» In: *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*. Vol. 6. 2017, pp. 1–6.

[8] Wael Khreich et al. «Combining heterogeneous anomaly detectors for improved software security.» In: *Journal of Systems and Software* 137 (2018), pp. 415–429.

[9] Bayu Adhi Tama et al. «An enhanced anomaly detection in web traffic using a stack of classifier ensemble.» In: *IEEE Access* 8 (2020), pp. 24120–24134.

[10] Victoria J. Hodge and Jim Austin. «A survey of outlier detection methodologies.» In: *Artificial Intelligence Review* 22.2 (2004), pp. 85–126.

[11] Varun Chandola, Arindam Banerjee, and Vipin Kumar. «Anomaly detection: a survey.» In: *ACM Computing Surveys* 41.3 (2009), pp. 1–58.

[12] Hongzhi Wang, Mohamed Jaward Bah, and Mohamed Hammad. «Progress in outlier detection techniques: a survey.» In: *IEEE Access* 7 (2019), pp. 107964–108000.

[13] Xuyun Zhang et al. «LSHiForest: A generic framework for fast tree isolation based ensemble anomaly analysis.» In: *Proceedings - International Conference on Data Engineering*. 2017, pp. 983–994.

[14] Faloutsos Christos et al. *SPIRIT: Streaming Pattern Discovery*. Matlab package. 2006. URL: http://www.cs.cmu.edu/afs/cs/project/spirit-1/www/.

[15] Albert Bifet et al. «MOA: Massive Online Analysis.» In: *J. Mach. Learn. Res.* 11 (2010), pp. 1601–1604.

[16] Shaoxu Song et al. «SCREEN: Stream Data Cleaning under Speed Constraints.» In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 2015, pp. 827–841.

[17] Aoqian Zhang, Shaoxu Song, and Jianmin Wang. «Sequential Data Cleaning: A Statistical Approach.» In: *Proceedings of the 2016 International Conference on Management of Data SIGMOD*. 2016, pp. 909–924.

[18] A. Siffer, P. A. Fouque, and C. Termier A. amd Largouet. «Anomaly Detection in Streams with Extreme Value Theory.» In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017, pp. 1067–1075.

[19] Subutai Ahmad et al. «Unsupervised real-time anomaly detection for streaming data.» In: *Neurocomputing* 262 (2017), pp. 134–147.

[20] Evgeny Burnaev and Vladislav Ishimtsev. «Conformalized density- and distance-based anomaly detection in time-series data.» In: *ArXiv* abs/1608.04585 (2016).

[21] Mikhail Smirnov. *Contextual Anomaly Detector*. Python package. 2016. URL: https://github.com/smirmik/CAD/.

[22] Jordan Hochenbaum, Owen Vallis, and Arun Kejariwal. «Automatic Anomaly Detection in the Cloud Via Statistical Learning.» In: *ArXiv* abs/1704.07706 (2017).

[23] Charu Aggarwal and Saket Sathe. *Outlier ensembles*. 2. Springer Publishing Company, Incorporated, 2017, pp. XVI, 276.

[24] Zhi Hua Zhou. *Ensemble methods: Foundations and algorithms*. Chapman & Hall/CRC, 2012, pp. 1–218.

[25] Tahani Alqurashi and Wenjia Wang. «Clustering ensemble method.» In: *International Journal of Machine Learning and Cybernetics* 10.6 (2019), pp. 1227–1246.

[26] Aggarwal, Charu C. Springer Publishing Company, Incorporated, 2015.

[27] Sergio González et al. «A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities.» In: *Information Fusion* 64 (2020), pp. 205–237.

[28] Ane Blázquez-García et al. «A Review on Outlier/Anomaly Detection in Time Series Data.» In: *ACM Computing Surveys* 54.3 (2021).

[29] Charu C Aggarwal. 2nd edition. Springer, Cham, 2017.

[30] Md Shariful Islam, Wael Khreich, and Abdelwahab Hamou-Lhadj. «Anomaly Detection Techniques Based on Kappa-Pruned Ensembles.» In: *IEEE Transactions on Reliability* 67.1 (2018), pp. 212–229.

[31] Yang Zhang, Nirvana Meratnia, and Paul Havinga. «Outlier Detection Techniques for Wireless Sensor Networks: A Survey.» In: *IEEE Communications Surveys Tutorials* 12.2 (2010), pp. 159–170.

[32] Xingwei Yang, Longin Jan Latecki, and Dragoljub Pokrajac. «Outlier detection with globally optimal exemplar-based GMM.» In: *Proceedings of the 2009 SIAM International Conference on Data Mining*. SIAM. 2009, pp. 145–154.

[33] Xiu-Ming Tang, Rong-Xiang Yuan, and Jun Chen. «Outlier Detection in Energy Disaggregation Using Subspace Learning and Gaussian Mixture Model.» In: *International Journal of Control and Automation* 8.8 (2015), pp. 161–170.

[34] M. Pavlidou and G. Zioutas. «Kernel density outlier detector.» In: *Springer Proceedings in Mathematics and Statistics*. Vol. 74. 2014, pp. 241–250.

[35] V. S. Kumar Samparthi and Harsh K. Verma. «Outlier Detection of Data in Wireless Sensor Networks Using Kernel Density Estimation.» In: *International Journal of Computer Applications* 5.6 (2010), pp. 28–32.

[36] Markus M. Breuniq et al. «LOF: Identifying density-based local outliers.» In: *SIGMOD Record (ACM Special Interest Group on Management of Data)* 29.2 (2000), pp. 93–104.

[37] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. «Efficient algorithms for mining outliers from large data sets.» In: *ACM SIGMOD Record*. 2000, pp. 427–438.

[38] Amol Ghoting, Srinivasan Parthasarathy, and Matthew Eric Otey. «Fast mining of distance-based outliers in high-dimensional datasets.» In: *Data Mining and Knowledge Discovery* 16.3 (2008), pp. 349–364.

[39] James others MacQueen. «Some methods for classification and analysis of multivariate observations.» In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* 1.14 (1967), pp. 281–297.

[40] Martin Ester et al. «A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.» In: *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*. 1996, pp. 226–231.

[41] Jianhua Yin and Jianyong Wang. «A model-based approach for text clustering with outlier detection.» In: *2016 IEEE 32nd International Conference on Data Engineering, ICDE* (2016), pp. 625–636.

[42] Atiq ur Rehman and Samir Brahim Belhaouari. «Unsupervised outlier detection in multidimensional data.» In: *Journal of Big Data* 8.1 (2021), pp. 1–27.

[43] Guilan Feng et al. «Entropy-based outlier detection using spark.» In: *Cluster Computing* 23.2 (2019), pp. 409–419.

[44] Feng Jiang, Yuefei Sui, and Cungen Cao. «An information entropy-based approach to outlier detection in rough sets.» In: *Expert Systems with Applications* 37.9 (2010), pp. 6338–6344.

[45] Zhong Yuan et al. «Fuzzy information entropy-based adaptive approach for hybrid feature outlier detection.» In: *Fuzzy Sets and Systems* 421 (2021), pp. 1–28.

[46] Fabrizio Carcillo et al. «Combining unsupervised and supervised learning in credit card fraud detection.» In: *Information Sciences* (2019), pp. 1–15.

[47] Jiachen Liu et al. «Modular ensembles for one-class classification based on density analysis.» In: *Neurocomputing* 171 (2016), pp. 262–276.

[48] Jia Zhang et al. «DELR: A double-level ensemble learning method for unsupervised anomaly detection.» In: *Knowledge-Based Systems* 181 (2019), p. 104783.

[49] Battista Biggio and Fabio Roli. «Wild patterns: Ten years after the rise of adversarial machine learning.» In: *Pattern Recognition* 84 (2018), pp. 317–331.

[50] Manish Gupta et al. «Outlier Detection for Temporal Data: A Survey.» In: *Ieee Transactions on Knowledge and Data Engineering* 25.1 (2013), pp. 1–20.

[51] Saeed Mehrang et al. «Outlier detection in weight time series of connected scales.» In: *Proceedings - 2015 IEEE International Conference on Bioinformatics and Biomedicine*. 2015, pp. 1489–1496.

[52] Aarthi Reddy et al. «Using Gaussian mixture models to detect outliers in seasonal univariate network traffic.» In: *Proceedings - 2017 IEEE Symposium on Security and Privacy Workshops*. 2017, pp. 229–234.

[53] David J. Hill and Barbara S. Minsker. «Anomaly detection in streaming environmental sensor data: A data-driven modeling approach.» In: *Environmental Modelling and Software* 25.9 (2010), pp. 1014–1022.

[54] Y. Zhang et al. «Statistics-based outlier detection for wireless sensor networks.» In: *International Journal of Geographical Information Science* 26.8 (2012), pp. 1373–1392.

[55] Haibin Cheng et al. «A Robust Graph-Based Algorithm for Detection and Characterization of Anomalies in Noisy Multivariate Time Series.» In: *2008 IEEE International Conference on Data Mining Workshops*. 2008, pp. 349–358.

[56] Haibin Cheng et al. «Detection and Characterization of Anomalies in Multivariate Time Series.» In: *Proceedings of the 2009 SIAM International Conference on Data Mining*. 2009, pp. 413–424.

[57] S. Muthukrishnan, R. Shah, and J.S. Vitter. «Mining deviants in time series data streams.» In: *Proceedings. 16th International Conference on Scientific and Statistical Database Management*. Vol. 16. 2004, pp. 41–50.

[58] Zhong Min Wang, Guo Hao Song, and Cong Gao. «An isolation-based distributed outlier detection framework using nearest neighbor ensembles for wireless sensor networks.» In: *IEEE Access* 7 (2019), pp. 96319–96333.

[59] Alberto Diez-Olivan et al. «Data fusion and machine learning for industrial prognosis: Trends and perspectives towards Industry 4.0.» In: *Information Fusion* 50 (2019), pp. 92–111.

[60] Fangyu Li et al. «System Statistics Learning-Based IoT Security: Feasibility and Suitability.» In: *IEEE Internet of Things Journal* 6.4 (2019), pp. 6396–6403.

[61] Monika Arya and Hanumat Sastry G. «DEAL–Deep Ensemble ALgorithm Framework for Credit Card Fraud Detection in Real-Time Data Stream with Google TensorFlow.» In: *Smart Science* 8.2 (2020), pp. 71–83.

[62] Haider Raza, Girijesh Prasad, and Yuhua Li. «EWMA model based shift-detection methods for detecting covariate shifts in non-stationary environments.» In: *Pattern Recognition* 48.3 (2015), pp. 659–669.

[63] Heitor M Gomes et al. «Adaptive random forests for evolving data stream classification.» In: *Machine Learning* 106.9 (2017), pp. 1469–1495.

[64] Kevin M. Carter and William W. Streilein. «Probabilistic reasoning for streaming anomaly detection.» In: *2012 IEEE Statistical Signal Processing Workshop*. Vol. 1. 2012, pp. 377–380.

[65] Zhao Xu, Kristian Kersting, and L. von Ritter. «Adaptive Streaming Anomaly Analysis.» In: *Proceedings of NIPS 2016 Workshop on Artificial Intelligence for Data Science*. 2016.

[66] Zhao Xu, Kristian Kersting, and Lorenzo Von Ritter. «Stochastic online anomaly analysis for streaming time series.» In: *IJCAI International Joint Conference on Artificial Intelligence*. 2017, pp. 3189–3195.

[67] Fabrizio Angiulli and Fabio Fassetti. «Distance-based outlier queries in data streams: The novel task and algorithms.» In: *Data Mining and Knowledge Discovery* 20.2 (2010), pp. 290–324.

[68] Vladislav Ishimtsev et al. «Conformal k-NN Anomaly Detector for Univariate Data Streams.» In: *Proceedings of the Sixth Workshop on Conformal and Probabilistic Prediction and Applications* 60 (2017), pp. 213–227.

[69] Xiaolei Li et al. «Temporal Outlier Detection in Vehicle Traffic Data.» In: *2009 IEEE 25th International Conference on Data Engineering*. 2009, pp. 1319–1322.

[70] Bartosz Krawczyk et al. «Ensemble learning for data stream analysis: A survey.» In: *Information Fusion* 37 (2017), pp. 132–156.

[71] Charu C. Aggarwal and Saket Sathe. «Theoretical Foundations and Algorithms for Outlier Ensembles.» In: *ACM SIGKDD Explorations Newsletter* 17.1 (2015), pp. 24–47.

[72] Yoav Freund and Robert E. Schapire. «A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting.» In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139.

[73] Jia Zhang, Zhiyong Li, and Shaomiao Chen. «Diversity Aware-Based Sequential Ensemble Learning for Robust Anomaly Detection.» In: *IEEE Access* 8 (2020), pp. 42349–42363.

[74] Jinghui Chen et al. «Outlier detection with autoencoder ensembles.» In: *Proceedings of the 17th SIAM International Conference on Data Mining*. 2017, pp. 90–98.

[75] Yue Dong and Nathalie Japkowicz. «Threaded ensembles of supervised and unsupervised neural networks for stream learning.» In: *Advances in Artificial Intelligence*. 2016, pp. 304–315.

[76] Huifen Hong et al. «Concurrent Monitoring Strategy for Static and Dynamic Deviations Based on Selective Ensemble Learning Using Slow Feature Analysis.» In: *Industrial and Engineering Chemistry Research* 59.10 (2020), pp. 4620–4635.

[77] Shehroz S. Khan and Amir Ahmad. «Relationship between Variants of One-Class Nearest Neighbors and Creating Their Accurate Ensembles.» In: *IEEE Transactions on Knowledge and Data Engineering* 30.9 (2018), pp. 1796–1809.

[78] Yanjie Fu et al. «REMIX: Automated exploration for interactive outlier detection.» In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017, pp. 827–835.

[79] Shehroz S. Khan and Babak Taati. «Detecting unseen falls from wearable devices using channel-wise ensemble of autoencoders.» In: *Expert Systems with Applications* 87 (2017), pp. 280–290.

[80] Liu Liu et al. «Anomaly-based insider threat detection using deep autoencoders.» In: *IEEE International Conference on Data Mining Workshops*. 2019, pp. 39–48.

[81] Junfeng Wu et al. «Combining OC-SVMs with LSTM for Detecting Anomalies in Telemetry Data with Irregular Intervals.» In: *IEEE Access* 8 (2020), pp. 106648–106659.

[82] Biao Wang, Zhizhong Mao, and Keke Huang. «A prediction and outlier detection scheme of molten steel temperature in ladle furnace.» In: *Chemical Engineering Research and Design* 138 (2018), pp. 229–247.

[83] Zhao Xu et al. «Data-Driven Inter-Turn Short Circuit Fault Detection in Induction Machines.» In: *IEEE Access* 5 (2017), pp. 25055–25068.

[84] Andreas Theissler. «Detecting known and unknown faults in automotive systems using ensemble-based anomaly detection.» In: *Knowledge-Based Systems* 123 (2017), pp. 163–173.

[85] Daniel T. Ramotsoela, Gerhard P. Hancke, and Adnan M. Abu-Mahfouz. «Attack detection in water distribution systems using machine learning.» In: *Human-centric Computing and Information Sciences* 9.1 (2019), pp. 1–22.

[86] Fadi Salo, Ali Bou Nassif, and Aleksander Essex. «Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection.» In: *Computer Networks* 148 (2019), pp. 164–175.

[87] Biao Wang and Zhizhong Mao. «Outlier detection based on a dynamic ensemble model: Applied to process monitoring.» In: *Information Fusion* 51 (2019), pp. 244–258.

[88] Bartosz Krawczyk and Michał Woźniak. «Dynamic classifier selection for one-class classification.» In: *Knowledge-Based Systems* 107 (2016), pp. 43–53.

[89] Likang Shi et al. «Statistical process monitoring based on ensemble structure analysis.» In: *IEEE/CAA Journal of Automatica Sinica* (2018), pp. 1–8.

[90] Biao Wang, Zhizhong Mao, and Keke Huang. «Detecting outliers for complex nonlinear systems with dynamic ensemble learning.» In: *Chaos, Solitons and Fractals* 121 (2019), pp. 98–107.

[91] Biao Wang and Zhizhong Mao. «A dynamic ensemble outlier detection model based on an adaptive k-nearest neighbor rule.» In: *Information Fusion* 63 (2020), pp. 30–40.

[92] Ping Yuan, Biao Wang, and Zhizhong Mao. «Using multiple classifier behavior to develop a dynamic outlier ensemble.» In: *International Journal of Machine Learning and Cybernetics* (2020), pp. 1–13.

[93] Mengze Zhou et al. «Ensemble-Based Algorithm for Synchrophasor Data Anomaly Detection.» In: *IEEE Transactions on Smart Grid* 10.3 (2019), pp. 2979–2988.

[94] Shebuti Rayana and Leman Akoglu. «Less is more: Building selective anomaly ensembles.» In: *ACM Transactions on Knowledge Discovery from Data* 10.4 (2016).

[95] Weiping Diao, Ijaz Haider Naqvi, and Michael Pecht. «Early detection of anomalous degradation behavior in lithium-ion batteries.» In: *Journal of Energy Storage* 32 (2020), p. 101710.

[96] Bo Li et al. «Incorporating URL embedding into ensemble clustering to detect web anomalies.» In: *Future Generation Computer Systems* 96 (2019), pp. 176–184.

[97] Mohsin Munir et al. «DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series.» In: *IEEE Access* 7 (2019), pp. 1991–2005.

[98] Chudong Tong, Ting Lan, and Xuhua Shi. «Double-layer ensemble monitoring of non-gaussian processes using modified independent component analysis.» In: *ISA Transactions* 68 (2017), pp. 181–188.

[99] Chudong Tong, Ting Lan, and Xuhua Shi. «Ensemble modified independent component analysis for enhanced non-Gaussian process monitoring.» In: *Control Engineering Practice* 58 (2017), pp. 34–41.

[100] Chengjun Zhan, Shuanghong Li, and Yupu Yang. «Enhanced Fault Detection Based on Ensemble Global-Local Preserving Projections with Quantitative Global-Local Structure Analysis.» In: *Industrial and Engineering Chemistry Research* 56.38 (2017), pp. 10743–10755.

[101] Ping Cui, Chengjun Zhan, and Yupu Yang. «Improved nonlinear process monitoring based on ensemble KPCA with local structure analysis.» In: *Chemical Engineering Research and Design* 142 (2019), pp. 355–368.

[102] Alexey Tsymbal, Mykola Pechenizkiy, and Pádraig Cunningham. «Diversity in search strategies for ensemble feature selection.» In: *Information Fusion* 6.1 (2005), pp. 83–98.

[103] Yue Zhao et al. «LSCP: Locally selective combination in parallel outlier ensembles.» In: *SIAM International Conference on Data Mining*. 2019, pp. 585–593.

[104] Joash Kiprotich Bii, Richard Rimiru, and Ronald Waweru Mwangi. «Adaptive boosting in ensembles for outlier detection: Base learner selection and fusion via local domain competence.» In: *ETRI Journal* (2020), pp. 1–13.

[105] Shebuti Rayana, Wen Zhong, and Leman Akoglu. «Sequential Ensemble Learning for Outlier Detection: A Bias-Variance Perspective.» In: *2016 IEEE 16th International Conference on Data Mining*. 2016, pp. 1167–1172.

[106] Saket Sathe and Charu C. Aggarwal. «Subspace histograms for outlier detection in linear time.» In: *Knowledge and Information Systems* 56.3 (2018), pp. 691–715.

[107] Ke Xu et al. «EnsembleLens: Ensemble-based Visual Exploration of Anomaly Detection Algorithms with Multidimensional Data.» In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (2019), pp. 109–119.

[108] Alican Dogan and Derya Birant. «A Two-Level Approach based on Integration of Bagging and Voting for Outlier Detection.» In: *Journal of Data and Information Science* 5.2 (2020), pp. 111–135.

[109] Shamsul Huda et al. «An Ensemble Oversampling Model for Class Imbalance Problem in Software Defect Prediction.» In: *IEEE Access* 6 (2018), pp. 24184–24195.

[110] Debasrita Chakraborty, Vaasudev Narayanan, and Ashish Ghosh. «Integration of deep feature extraction and ensemble learning for outlier detection.» In: *Pattern Recognition* 89 (2019), pp. 161–171.

[111] Salisu Wada Yahaya, A. Lotfi, and M. Mahmud. «A Consensus Novelty Detection Ensemble Approach for Anomaly Detection in Activities of Daily Living.» In: *Applied Soft Computing Journal* 83 (2019), p. 105613.

[112] Sumon Kumar Bose et al. «ADEPOS: A Novel Approximate Computing Framework for Anomaly Detection Systems and its Implementation in 65-nm CMOS.» In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 67.3 (2020), pp. 913–926.

[113] Manuel Roveri and Francesco Trovò. «An Ensemble Approach for Cognitive Fault Detection and Isolation in Sensor Networks.» In: *International Journal of Neural Systems* 27.3 (2017), p. 1650047.

[114] Victor Henrique Alves Ribeiro et al. «A novel dynamic multi-criteria ensemble selection mechanism applied to drinking water quality anomaly detection.» In: *Science of the Total Environment* 749 (2020), p. 142368.

[115] Charu C. Aggarwal. «Outlier Ensembles: Position Paper.» In: *ACM SIGKDD Explorations Newsletter* 14.2 (2013), pp. 49–58.

[116] Elham Parhizkar and Mahdi Abadi. «BeeOWA: A novel approach based on ABC algorithm and induced OWA operators for constructing one-class classifier ensembles.» In: *Neurocomputing* 166 (2015), pp. 367–381.

[117] Tung Kieu et al. «Outlier detection for time series with recurrent autoencoder ensembles.» In: *IJCAI International Joint Conference on Artificial Intelligence*. 2019, pp. 2725–2732.

[118] Jina Jeong et al. «Identifying outliers of non-Gaussian groundwater state data based on ensemble estimation for long-term trends.» In: *Journal of Hydrology* 548 (2017), pp. 135–144.

[119] Marta B. Lopes et al. «Ensemble outlier detection and gene selection in triple-negative breast cancer data.» In: *BMC Bioinformatics* 19.1 (2018), p. 168.

[120] Kien Do, Truyen Tran, and Svetha Venkatesh. «Energy-based anomaly detection for mixed data.» In: *Knowledge and Information Systems* 57.2 (2018), pp. 413–435.

[121] John G. Kemeny. «Mathematics without Numbers.» In: *Daedalus* 88.4 (1959), pp. 577–591.

[122] Ludmila I. Kuncheva, James C. Bezdek, and Robert P.W. Duin. «Decision templates for multiple classifier fusion: an experimental comparison.» In: *Pattern Recognition* 34.2 (2001), pp. 299–314.

[123] Bartosz Krawczyk. «One-class classifier ensemble pruning and weighting with firefly algorithm.» In: *Neurocomputing* 150 (2015), pp. 490–500.

[124] Ronald R. Yager and Dimitar P. Filev. «Induced Ordered Weighted Averaging operators.» In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 29.2 (1999), pp. 141–150.

[125] A. Lavin and S. Ahmad. «Evaluating Real-Time Anomaly Detection Algorithms – The Numenta Anomaly Benchmark.» In: *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. 2015, pp. 38–44.

[126] Sabyasachi Basu and Martin Meckesheimer. «Automatic outlier detection for time series: An application to sensor data.» In: *Knowledge and Information Systems* 11.2 (2007), pp. 137–154.

[127] Alban Siffer et al. «Anomaly detection in streams with extreme value theory.» In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Vol. Part F1296. 2017, pp. 1067–1075.

[128] Yusheng Zhou et al. «A data quality control method for seafloor observatories: The application of observed time series data in the east China sea.» In: *Sensors* 18.8 (2018).

[129] Lukasz Komsta. *outliers: Tests for outliers*. R package version 0.14. 2011. URL: https://CRAN.R-project.org/package=outliers.

[130] Siddharth Jain and Prabhanjan Tattar. *SMLoutliers: Outlier Detection Using Statistical and Machine Learning Methods*. R package version 0.1. 2017. URL: https://CRAN.R-project.org/package=SMLoutliers.

[131] Marcello D'Orazio. *univOutl: Detection of Univariate Outliers*. R package version 0.1-4. 2018. URL: https://CRAN.R-project.org/package=univOutl.

[132] Antony Unwin. *OutliersO3: Draws Overview of Outliers (O3) Plots*. R package version 0.6.2. 2019. URL: https://CRAN.R-project.org/package=OutliersO3.

[133] Javier López de Lacalle. *tsoutliers: Detection of Outliers in Time Series*. R package version 0.6-8. 2019. URL: https://CRAN.R-project.org/package=tsoutliers.

[134] Jacob Anhoej. *qicharts: Quality Improvement Charts*. R package version 0.5.5. 2017. URL: https://CRAN.R-project.org/package=qicharts.

[135] Andrea Venturini. *washeR: Time Series Outlier Detection*. R package version 0.1.2. 2018. URL: https://CRAN.R-project.org/package=washeR.

[136] Wacharasak Siriseriwan. *smotefamily: A Collection of Oversampling Techniques for Class Imbalance Problem Based on SMOTE*. R package version 1.3. 2018. URL: https://CRAN.R-project.org/package=smotefamily.

[137] Ignacio Cordón et al. «Imbalance: Oversampling algorithms for imbalanced classification in R.» In: *Knowledge-Based Systems* 161 (2018), pp. 329 –341.

[138] Hsiang Hao and Chen. *ebmc: Ensemble-Based Methods for Class Imbalance Problem*. R package version 1.0.0. 2017. URL: https://CRAN.R-project.org/package=ebmc.

[139] Chongsheng Zhang et al. «Multi-Imbalance: An Open-Source Software for Multi-Class Imbalance Learning.» In: *Knowledge-Based Systems* (2019).

[140] Andrea Pagotto. *ocp: Bayesian Online Changepoint Detection*. R package version 0.1.0. 2018. URL: https://CRAN.R-project.org/package=ocp.

[141] Nicholas A. James and David S. Matteson. «ecp: An R Package for Nonparametric Multiple Change Point Analysis of Multivariate Data.» In: *Journal of Statistical Software* 62.7 (2014), pp. 1–25.

[142] Nathaniel Garton. *bulletcp: Automatic Groove Identification via Bayesian Changepoint Detection*. R package version 1.0.0. 2019. URL: https://CRAN.R-project.org/package=bulletcp.

[143] Michael Messer et al. *MFT: The Multiple Filter Test for Change Point Detection*. R package version 2.0. 2019. URL: https://CRAN.R-project.org/package=MFT.

[144] Jin Man Park and Jong Hwan Kim. «Online recurrent extreme learning machine and its application to time-series prediction.» In: *Proceedings of the International Joint Conference on Neural Networks*. 2017, pp. 1983–1990.

[145] Piotr S. Maciąg et al. «Unsupervised Anomaly Detection in Stream Data with Online Evolving Spiking Neural Networks.» In: *Neural Networks* 139 (2021), pp. 118–139.

[146] Danushka Bollegala. «Dynamic feature scaling for online learning of binary classifiers.» In: *Knowledge-Based Systems* 129 (2017), pp. 97–105.

[147] Jessica Lin and Eamonn Keogh. «Finding or not finding rules in time series.» In: *Advances in Econometrics*. Vol. 19. 2004, pp. 175–201.

[148] Simon S. Haykin. *Neural networks and learning machines*. 3rd ed. Pearson Education, 2009.

[149] Eduardo Ogasawara et al. «Adaptive Normalization: A novel data normalization approach for non-stationary time series.» In: *Proceedings of the International Joint Conference on Neural Networks*. 2010, pp. 1–8.

[150] Vibhuti Gupta and Rattikorn Hewett. «Adaptive Normalization in Streaming Data.» In: *Proceedings of the 2019 3rd International Conference on Big Data Research*. 2019, pp. 12–17.

[151] Bryan Lim and Stefan Zohren. «Time-series forecasting with deep learning: a survey.» In: *Philosophical Transactions of the Royal Society A* 379.2194 (2021), p. 20200209.

[152] Ricardo P Masini, Marcelo C Medeiros, and Eduardo F Mendes. «Machine learning advances for time series forecasting.» In: *Journal of Economic Surveys* (2021).

[153] Nesreen K Ahmed et al. «An empirical comparison of machine learning models for time series forecasting.» In: *Econometric Reviews* 29.5-6 (2010), pp. 594–621.

[154] Teodora Sandra Buda, Bora Caglayan, and Haytham Assem. «Deepad: A generic framework based on deep learning for time series anomaly detection.» In: *Pacific-Asia conference on knowledge discovery and data mining*. Springer. 2018, pp. 577–588.

[155] Nan Ying Liang et al. «A fast and accurate online sequential learning algorithm for feedforward networks.» In: *IEEE Transactions on Neural Networks* 17.6 (2006), pp. 1411–1423.

[156] Yuwei Cui et al. «A comparative study of HTM and other neural network models for online sequence learning with streaming data.» In: *Proceedings of the International Joint Conference on Neural Networks*. 2016, pp. 1530–1538.

[157] Yuwei Cui, Subutai Ahmad, and Jeff Hawkins. «The HTM spatial pooler—a neocortical algorithm for online sparse distributed coding.» In: *Frontiers in Computational Neuroscience* 11 (2017).

[158] Liyanaarachchi Lekamalage Chamara Kasun et al. «Representational learning with extreme learning machine for big data.» In: *IEEE intelligent systems* 28.6 (2013), pp. 31–34.

[159] Jiexiong Tang, Chenwei Deng, and Guang-Bin Huang. «Extreme learning machine for multilayer perceptron.» In: *IEEE transactions on neural networks and learning systems* 27.4 (2015), pp. 809–821.

[160] Jonathan Tapson, Philip De Chazal, and André van Schaik. «Explicit computation of input weights in extreme learning machines.» In: *Proceedings of ELM-2014*. Springer, 2015, pp. 41–49.

[161] Alaiñe Iturria et al. «otsad: A package for online time-series anomaly detectors.» In: *Neurocomputing* 374 (2020), pp. 49–53.

[162] Alessio Benavoli et al. «Time for a Change: a Tutorial for Comparing Multiple Classifiers Through Bayesian Analysis.» In: *Journal of Machine Learning Research* 18.77 (2017), pp. 1–36.