Librería para la evaluación de métodos de detección de anomalías en series temporales

Pedro R. Velasco[1], Amelia Zafra[1,2], and Sebastián Ventura[1,2]

[1] Departamento de Ciencia de la Computación e Inteligencia Artificial, Universidad de Córdoba, Córdoba, España
[2] Instituto Andaluz Interuniversitario en Ciencia de Datos e Inteligencia Computacional (DaSCI)
{i12veprp,azafra,sventura}@uco.es

Abstract. Este trabajo presenta una librería Python de código abierto que proporciona un marco unificado para evaluar métodos de detección de anomalías en series temporales. Integra 34 métricas bajo una taxonomía sistemática que distingue la evaluación punto a punto de los criterios de evaluación temporal, incluyendo la cobertura parcial de intervalos, el alineamiento temporal preciso, las penalizaciones por retraso, el emparejamiento temporal y la tolerancia a desplazamientos. La librería también proporciona gestión validada de parámetros, flujos de trabajo basados en configuración, ejecución por lotes sobre múltiples conjuntos de datos, generación de informes estructurados y mecanismos de extensibilidad para incorporar nuevas métricas. Se describen la taxonomía propuesta, la arquitectura y un ejemplo de uso, destacando su contribución como herramienta para realizar evaluaciones consistentes, comparables y reproducibles en el ámbito de la detección de anomalías en series temporales.

Keywords: Series temporales · Detección de anomalías · Métricas de evaluación · Librería Python

1 Introducción

El crecimiento exponencial de los flujos de datos en tiempo real en dominios como la monitorización industrial [29], los sistemas financieros [6] y las infraestructuras críticas [2] ha incrementado de forma significativa la importancia de la detección de anomalías en series temporales [4]. La detección de anomalías en datos secuenciales es esencial para garantizar la fiabilidad operativa, prevenir fallos y mitigar riesgos económicos o sociales. A diferencia de los conjuntos de datos estáticos, las anomalías en series temporales suelen manifestarse como intervalos temporales contiguos en lugar de como puntos aislados, lo que introduce desafíos específicos en la evaluación de los modelos. En este contexto, las métricas tradicionales, diseñadas originalmente para la clasificación punto a punto, no capturan las dependencias temporales ni la estructura basada en intervalos propia de este tipo de anomalías. Medidas como la exactitud, la precisión o el F-score punto

Tabla 1. Comparación con los principales recursos existentes.

Enfoque	Métricas	Taxonomía	Recurso	API	Configuración
Revisión estructural [26]	20	Sí	Código experimental	No	No
Taxonomía orientada [30]	20	Sí	–	No	No
Benchmark RankEval [32]	12	Parcial	Pipeline	No	No
TSADmetrics	**34**	**Sí**	**Librería**	**Sí**	**Sí**

a punto evalúan cada observación de forma independiente, ignorando aspectos clave como la duración de la anomalía, la continuidad temporal, el retraso en la detección y el alineamiento entre los intervalos anómalos predichos y reales [15, 27]. En la práctica, esta falta de adecuación puede conducir a conclusiones engañosas. Por ejemplo, métricas ampliamente utilizadas como el F-score con ajuste por punto pueden no penalizar adecuadamente las anomalías de corta duración que, aunque críticas, apenas tienen impacto en la métrica global debido al reducido número de puntos que abarcan [8, 17].

Para abordar estas limitaciones, trabajos recientes han propuesto numerosas métricas de evaluación sensibles al componente temporal que tienen en cuenta el solapamiento parcial, la tolerancia al desalineamiento temporal, la puntuación basada en eventos y la penalización de detecciones tardías. Sin embargo, la rápida proliferación de alternativas ha generado un panorama de evaluación fragmentado, dificultando la comparación justa de métodos y la selección de métricas que se ajusten a los objetivos y restricciones de una aplicación concreta. Diversos trabajos han tratado de sistematizar este espacio mediante la revisión de métricas y su organización en taxonomías estructurales [26], así como mediante taxonomías orientadas al problema basadas en características operativas como la oportunidad temporal, la sensibilidad a costes y la comparabilidad [30]. También se han realizado estudios basados en la formalización de propiedades deseables y en el análisis de qué métricas las satisfacen, así como benchmarks y protocolos específicos para evaluar la robustez de los rankings inducidos por las métricas [32]. Aunque estas contribuciones sistematizan el espacio de métricas mediante taxonomías, código experimental o protocolos de benchmarking, ninguna reúne estas funcionalidades en una librería multimétrica de propósito general. La Tabla 1 sintetiza las principales diferencias, considerando la cobertura de métricas, el tipo de recurso software y el soporte para una API unificada y la configuración externa de experimentos.

Frente a este escenario, TSADMetrics proporciona un marco unificado para la evaluación de métodos de detección de anomalías en series temporales. Como parte de esta contribución, se introduce una taxonomía extendida y sistemática que organiza las métricas de evaluación de acuerdo con sus propiedades fundamentales y objetivos de evaluación, conectando explícitamente consideraciones estructurales y temporales y extendiendo perspectivas taxonómicas previas [26]. Esta taxonomía está diseñada para clarificar el panorama fragmentado de evaluación y facilitar una selección informada de métricas mediante su categorización en dos grupos: aquellas que operan a nivel de punto y aquellas que incorporan contexto temporal. En concreto, distingue entre *Single-Point Based Metrics*

(SPM), que evalúan la detección de anomalías de forma independiente en cada instante temporal, y *Temporal Evaluation Metrics* (TEM), que evalúan las detecciones teniendo en cuenta el contexto temporal. La familia TEM se subdivide además en *Tolerant Partial Detection Metrics* (TPDM), *Precise Temporal Detection Metrics* (PTDM), *Delay-Penalized Metrics* (DPM), *Temporal Matching Evaluation Metrics* (TMEM) y *Temporal Shift-Tolerant Metrics* (TSTM). Esta organización jerárquica facilita una selección informada de métricas al alinear los criterios de evaluación con las características específicas de las anomalías y los objetivos de la tarea de detección.

Más allá de la organización conceptual, la librería incluye 34 métricas dentro de esta taxonomía y ofrece soporte tanto para predicciones binarias como para puntuaciones continuas de anomalía, junto con utilidades automatizadas que facilitan la ejecución de estudios experimentales y la generación de informes estructurados. En contraste con las implementaciones *ad hoc* asociadas a estudios individuales [30, 32], esta librería se diseña como un artefacto de software reutilizable, con una arquitectura modular y extensible que facilita la incorporación de nuevas métricas. De este modo, contribuye a mejorar la transparencia, la comparabilidad y la reproducibilidad en la investigación sobre detección de anomalías en series temporales.

Asimismo, la librería proporciona un marco común basado en configuración que permite la definición estructurada y explícita de conjuntos de datos, métricas y valores de parámetros mediante ficheros YAML externos. Esta capacidad permite a los investigadores diseñar y ejecutar estudios experimentales exhaustivos de manera sistemática y reproducible, reduciendo significativamente la complejidad asociada a la gestión manual de múltiples escenarios de evaluación. El uso de ficheros de configuración no solo simplifica la automatización de los estudios experimentales sistemáticos, sino que también garantiza la consistencia en la evaluación, favoreciendo la comparabilidad entre distintos modelos y configuraciones. Además, la librería elimina la necesidad de modificar el código, ya que todas las configuraciones de los experimentos pueden definirse externamente.

El resto del documento se organiza del siguiente modo: en la Sección 2 se presenta una visión general de la arquitectura y funcionalidades de la librería. En la Sección 3 se muestra un ejemplo ilustrativo. Finalmente, las conclusiones se recogen en la Sección 4.

2 Diseño y funcionalidades de la librería

Las siguientes secciones presentan el diseño y las funcionalidades principales de la librería. La sección de arquitectura del software describe la organización estructural de la librería, destacando su diseño modular, que facilita su extensión, así como sus principales componentes. La sección de funcionalidad describe las características prácticas de la librería, detallando cómo soporta evaluaciones reproducibles, automatizadas y transparentes de métodos de detección de anomalías en series temporales mediante taxonomías unificadas, flujos de trabajo basados en configuración e informes estructurados.

2.1 Arquitectura del software

La librería está disponible en GitHub[3] bajo licencia GNU. La base de código sigue un diseño modular para garantizar flexibilidad, mantenibilidad y futuras extensiones. Está organizada en paquetes, cada uno de los cuales contiene las interfaces y clases necesarias para extender su funcionalidad:

- base: Define la clase abstracta `Metric`, que proporciona una interfaz consistente para todas las métricas, incluyendo validación de parámetros, carga de configuraciones desde ficheros YAML y la lógica de cálculo. Esta clase impone una especificación unificada para la evaluación de métricas y simplifica la integración de nuevas métricas.
- evaluation: Implementa la clase `Runner` para la evaluación automatizada por lotes sobre múltiples conjuntos de datos y métricas, y la clase `Report` para generar resúmenes estructurados en formato CSV. `Runner` soporta tanto objetos Python directos como ficheros de configuración externos, permitiendo experimentos a gran escala sin modificar el código.
- utils: Proporciona funciones auxiliares para el preprocesamiento de datos, la gestión de intervalos y las operaciones numéricas, garantizando consistencia entre las implementaciones de métricas.
- metrics: Contiene la implementación de todas las métricas de evaluación, organizadas en submódulos de acuerdo con la taxonomía propuesta. En concreto, la librería incluye 34 métricas distribuidas en dos grandes grupos: métricas basadas en puntos individuales y métricas de evaluación temporal. Estas métricas fueron identificadas mediante una revisión de la literatura que incluyó revisiones, taxonomías y estudios comparativos recientes [26, 30, 32].
  - SPM (Single-Point Based Metrics): métricas que tratan cada instante temporal de forma independiente. Incluyen:
    - `PointwiseFScore` (*pwf*).
    - `DiceCoefficient` (*dicec*) [20].
    - `PointwiseAucRoc` (*pw_auc_roc*) [9].
    - `PointwiseAucPr` (*pw_auc_pr*) [9].
    - `PrecisionAtK` (*pak*) [22].
  - TEM (Temporal Evaluation Metrics): métricas que incorporan contexto temporal y evalúan las detecciones a nivel de intervalo o secuencia. Se dividen en las siguientes subfamilias:
    - TPDM (Tolerant Partial Detection Metrics): permiten detecciones parciales dentro de intervalos anómalos. Incluyen:
      - `PointadjustedFScore` (*paf*) [28].
      - `BalancedPointadjustedFScore` (*bpaf*) [3].
      - `SegmentwiseFScore` (*swf*) [14].
      - `CompositeFScore` (*cf*) [10].
      - `PointadjustedAucPr` (*pa_auc_pr*) [7].
      - `PointadjustedAucRoc` (*pa_auc_roc*) [7].

[3] `https://github.com/kdis-lab/TSADmetrics`

  - `RangebasedFScore` (*rbf*) [27].

* PTDM (Precise Temporal Detection Metrics): evalúan la cobertura, precisión y alineamiento de las detecciones respecto a los intervalos anómalos reales. Incluyen:
  - `AverageDetectionCount` (*adc*) [11].
  - `TotalDetectedInRange` (*tdir*) [18].
  - `DetectionAccuracyInRange` (*dair*) [18].
  - `WeightedDetectionDifference` (*wdd*) [18].
  - `PointadjustedAtKFScore` (*pakf*) [24].
  - `PointadjustedAtKLFScore` (*paklf*) [24].
  - `TimeseriesAwareFScore` (*taf*) [15].
* DPM (Delay-Penalized Metrics): penalizan detecciones tardías y favorecen la identificación temprana de anomalías. Incluyen:
  - `DelayThresholdedPointadjustedFScore` (*dtpaf*) [23].
  - `EarlyDetectionScore` (*eds*) [5].
  - `LatencySparsityawareFScore` (*lsaf*) [1].
  - `MeanTimeToDetect` (*mttd*) [31].
  - `NabScore` (*nab_score*) [19].
* TMEM (Temporal Matching Evaluation Metrics): evalúan el grado de correspondencia entre intervalos anómalos reales y predichos. Incluyen:
  - `AbsoluteDetectionDistance` (*add*) [11].
  - `EnhancedTimeseriesAwareFScore` (*etaf*) [16].
  - `TemporalDistance` (*ed*) [18].
* TSTM (Temporal Shift-Tolerant Metrics): introducen tolerancia frente a pequeños desalineamientos temporales y a imprecisiones en los límites de los intervalos. Incluyen:
  - `AffiliationbasedFScore` (*aff_f*) [13].
  - `NormalizedAffiliationbasedFScore` (*naff_f*) [33].
  - `PateFScore` (*pate_f1*) [12].
  - `Pate` (*pate*) [12].
  - `TimeTolerantFScore` (*ttf*) [25].
  - `VusRoc` (*vus_roc*) [21].
  - `VusPr` (*vus_pr*) [21].

La arquitectura incluye un `Registry` central que registra automáticamente todas las clases de métricas, haciéndolas disponibles para los flujos de evaluación y los experimentos basados en configuración. Los usuarios pueden definir conjuntos de datos, métricas y valores de parámetros íntegramente mediante ficheros de configuración YAML, lo que permite realizar estudios experimentales exhaustivos sin modificar el código subyacente. Este diseño promueve la reproducibilidad, la transparencia y la escalabilidad en los flujos de trabajo de investigación.

Además, la distribución proporciona un manual de usuario extenso con una descripción completa de la librería[4], incluyendo:

- Guía de instalación e inicio rápido: instrucciones para descargar la librería desde GitHub, instalar dependencias mediante `pip` y ejecutar los primeros ejemplos de evaluación.
- Descripción detallada de cada métrica y sus parámetros: cubre las 34 métricas implementadas, organizadas por taxonomía (SPM, TPDM, PTDM, DPM, TMEM, TSTM), incluyendo definiciones matemáticas, esquemas de parámetros y notas de uso.
- Ejemplos de evaluación directa y automatizada: muestra dos modos de ejecución: cálculo manual mediante clases individuales de métricas y evaluación a gran escala mediante la clase `Runner` con ficheros YAML para conjuntos de datos y métricas.
- Directrices para añadir nuevas métricas y extender la librería: proporciona un proceso paso a paso para implementar nuevas métricas mediante la herencia de la clase base `Metric`, la definición de esquemas de parámetros y el uso del sistema de registro automático. Incluye ejemplos y recomendaciones para pruebas unitarias.
- Generación de informes y reproducibilidad: explica cómo producir informes CSV estructurados que resumen resultados de métricas y valores de parámetros, garantizando trazabilidad y comparabilidad entre experimentos.
- Referencia de la API: ofrece documentación completa en formatos HTML y PDF, detallando cada clase definida en la librería.

Asimismo, la documentación en línea completa de la librería está disponible en `https://tsadmetrics.readthedocs.io/`.

2.2 Funcionalidades del software

La librería proporciona un conjunto completo de funcionalidades para soportar evaluaciones *reproducibles*, *escalables* y *transparentes* de métodos de detección de anomalías en series temporales:

- Taxonomía unificada de métricas (34 métricas). Consolida las métricas bajo una clasificación sistemática para adecuarse a objetivos de evaluación como la detección temprana, la cobertura, el alineamiento y la tolerancia temporal [26]. Esto ayuda a evitar problemas conocidos de las métricas clásicas, como puntuaciones excesivamente optimistas en métricas con ajuste por punto [8, 17].
- Canalización de evaluación automatizada. La clase `Runner` ejecuta múltiples métricas sobre uno o varios conjuntos de datos por lotes, soportando tanto predicciones binarias como puntuaciones continuas de anomalía [21].

[4] `https://github.com/kdis-lab/TSADMetrics/blob/main/documentation/TSADmetrics-UserManual.pdf`

– Entradas flexibles y ficheros de configuración. Las evaluaciones aceptan arrays de datos directos o ficheros CSV (con información de los valores reales y las predicciones) y pueden gestionarse íntegramente mediante configuración externa (ficheros YAML) para declarar conjuntos de datos, métricas y valores de parámetros. Esto permite una experimentación exhaustiva y sistemática.
– Configuración de los experimentos sin modificar código. El uso de ficheros de configuración no solo simplifica la automatización de experimentos a gran escala, sino que también impone ajustes de evaluación consistentes entre ejecuciones y modelos, sin requerir modificaciones en el código subyacente. Los investigadores pueden ajustar conjuntos de datos, métricas, umbrales/tolerancias y otros parámetros de forma externa, mejorando la comparabilidad y la reproducibilidad.
– Informes estructurados. La librería produce informes CSV estructurados que resumen los resultados de las métricas y los valores de parámetros, facilitando la comparación y el benchmarking. La Figura 1 resume este flujo de trabajo, desde la definición de los datos y la configuración de métricas hasta la ejecución mediante `Runner` y la generación de resultados estructurados.
– Extensibilidad por diseño. Se pueden añadir nuevas métricas fácilmente mediante la interfaz `Metric` y el `Registry`. Las métricas específicas de dominio, como variantes multivariantes o sensibles a costes, pueden integrarse sin modificar la canalización.
– Distribución de código abierto y validación. Disponible a través de GitHub, con documentación detallada, ejemplos de uso y pruebas unitarias para garantizar la fiabilidad.

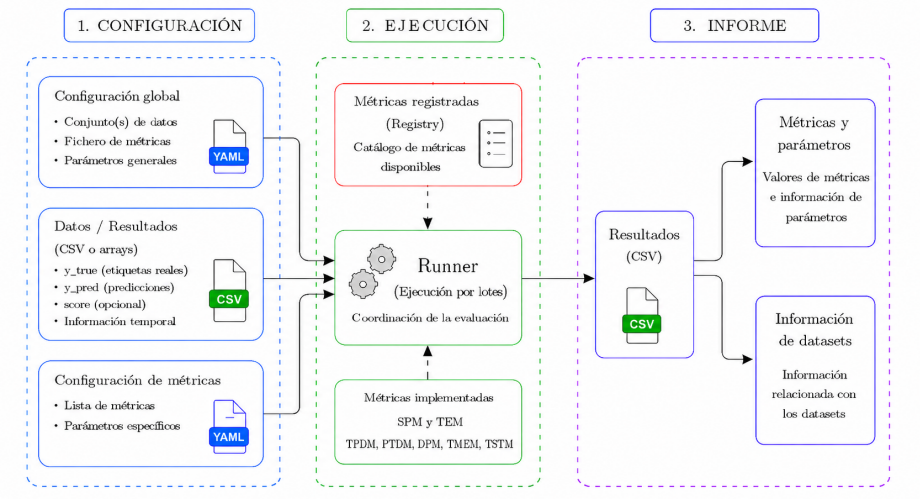


Fig. 1. Flujo de trabajo de la librería.

3 Un ejemplo ilustrativo

Esta sección resume un flujo de evaluación basado en configuración, mostrando cómo la librería separa la definición experimental de la ejecución del código. La librería trabaja con ficheros CSV que contienen las etiquetas de referencia y las predicciones de los detectores, y con ficheros YAML donde se especifican los conjuntos de datos, las métricas y sus parámetros.

3.1 Formatos de entrada y salida

La librería admite predicciones binarias, puntuaciones continuas de anomalía o ambas de forma simultánea. El formato combinado permite evaluar métricas dependientes e independientes de umbral dentro de un mismo flujo:

```
y_true;y_pred_binary;y_pred_continuous
1;1;0.7
1;1;0.6
0;0;0.1
```

La configuración de la evaluación se define mediante YAML. En el siguiente ejemplo se indican dos conjuntos de datos, una métrica puntual con parámetros por defecto y dos parametrizaciones de una métrica temporal:

```yaml
datasets:
  dataset1: ejemplo_entrada/resultado1.csv
  dataset2: ejemplo_entrada/resultado2.csv
metrics:
  - pwf
  - swf:
      beta: 0.7
  - swf:
      beta: 0.4
```

Como salida, la librería devuelve un `pandas.DataFrame` con una fila por conjunto de datos y una columna por instancia de métrica. Opcionalmente, también genera un informe CSV que conserva tanto los valores calculados como la parametrización empleada:

```
dataset;pwf;swf;swf
params;;{'beta': 0.7};{'beta': 0.4}
dataset1;0.6667;0.7525;0.8788
dataset2;0.2222;1.0000;1.0000
```

Este formato facilita la agregación, visualización y comparación posterior de resultados, manteniendo explícita la configuración necesaria para reproducir la evaluación.

3.2 Ejemplo de evaluación mediante configuración en fichero

Una vez preparados los ficheros de entrada, la evaluación se ejecuta mediante la clase `Runner`, que carga la configuración, instancia las métricas seleccionadas, aplica sus parámetros y centraliza la generación de resultados:

```
from tsadmetrics.evaluation.Runner import Runner

runner = Runner("ejemplo_entrada/configuracion_evaluacion.yaml")
results = runner.run(
    generate_report=True,
    report_file="ejemplo_salida/informe_evaluacion.csv"
)
print(results)
```

El método `run()` devuelve los resultados en memoria y, cuando `generate_report=True`, exporta la misma información a CSV. De este modo, el usuario puede modificar conjuntos de datos, métricas, umbrales o parámetros de tolerancia temporal sin alterar el código fuente de la librería.

3.3 Modos alternativos de uso

Además del uso basado en ficheros, la librería permite realizar evaluaciones directamente a partir de arrays en memoria, definir listas de métricas mediante código e instanciar manualmente métricas individuales mediante sus clases en Python. Estos modos complementarios facilitan tanto la experimentación como la integración de la librería en flujos de análisis desarrollados íntegramente en Python.

4 Conclusiones

Este trabajo presenta una librería desarrollada en Python y de código abierto que permite la evaluación de métodos de detección de anomalías en series temporales mediante un conjunto completo de 34 métricas, organizadas sistemáticamente de acuerdo con una taxonomía bien definida y fundamentada introducida en este trabajo. Esta taxonomía estructura las métricas de evaluación en función de sus objetivos principales y características temporales, cubriendo diversas dimensiones de evaluación como la exactitud punto a punto, la cobertura temporal, el alineamiento, la tolerancia a desplazamientos y las penalizaciones por retraso. Construida sobre una arquitectura modular, la librería es extensible y sencilla de instalar, configurar y utilizar. La librería se acompaña de un manual de usuario detallado que incluye ejemplos de uso, un tutorial paso a paso para realizar evaluaciones exhaustivas de detección de anomalías, una visión completa de la arquitectura y directrices para extender el software incorporando nuevas métricas. Consideramos que la librería puede facilitar de forma significativa la evaluación sistemática de métodos de detección de anomalías de acuerdo con las características específicas de las anomalías estudiadas, contribuyendo a futuros avances en el ámbito de la detección de anomalías en series temporales.

Agradecimientos

Esta investigación ha sido financiada parcialmente por los proyectos PID2023-148396NB-I00 del Ministerio de Ciencia e Innovación y el Fondo Europeo de Desarrollo Regional y por el proyecto ProyExcel-0069 de la Consejería de Universidad, Investigación e Innovación de la Junta de Andalucía.

References

1. Abdulaal, A., Liu, Z., Lancewicki, T.: Practical approach to asynchronous multivariate time series anomaly detection and localization. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining. pp. 2485—-2494. Association for Computing Machinery, New York, NY, USA (2021)
2. Aghazadeh Ardebili, A., Hasidi, O., Bendaouia, A., Khalil, A., Khalil, S., Luceri, D., Longo, A., Abdelwahed, E.H., Qassimi, S., Ficarella, A.: Enhancing resilience in complex energy systems through real-time anomaly detection: a systematic literature review. Energy Informatics **7**(1), 96 (2024)
3. Bhattacharya, D., Mukherjee, S., Kamanchi, C., Ekambaram, V., Jati, A., Dayama, P.: Towards unbiased evaluation of time-series anomaly detector. In: ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 1–5. IEEE (2025)
4. Blázquez-García, A., Conde, A., Mori, U., Lozano, J.A.: A review on outlier/anomaly detection in time series data. ACM computing surveys (CSUR) **54**(3), 1–33 (2021)
5. Buda, T.S., Assem, H., Xu, L.: ADE: An ensemble approach for early anomaly detection. In: IFIP/IEEE Symposium on Integrated Network and Service Management. pp. 442–448 (2017)
6. Chen, Y., Zhao, C., Xu, Y., Nie, C., Zhang, Y.: Deep learning in financial fraud detection: Innovations, challenges, and applications. Data Science and Management (2025)
7. Dai, L., Lin, T., Liu, C., Jiang, B., Liu, Y., Xu, Z., Zhang, Z.L.: SDFVAE: Static and dynamic factorized vae for anomaly detection of multivariate cdn kpis. In: Proceedings of the Web Conference. pp. 3076—3086. WWW '21, Association for Computing Machinery, New York, NY, USA (2021)
8. Doshi, K., Abudalou, S., Yilmaz, Y.: Reward once, penalize once: Rectifying time series anomaly detection. In: International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2022)
9. Fawcett, T.: An introduction to roc analysis. Pattern Recognition Letters **27**(8), 861–874 (2006)
10. Garg, A., Zhang, W., Samaran, J., Savitha, R., Foo, C.: An evaluation of anomaly detection and diagnosis in multivariate time series. IEEE Transactions on Neural Networks Learning System **33**(6), 2508–2517 (2022)
11. Gensler, A., Sick, B.: Novel criteria to measure performance of time series segmentation techniques. In: Proceedings of the LWA Workshops. CEUR Workshop Proceedings, vol. 1226, pp. 193–204. Aachen, Germany (2014)
12. Ghorbani, R., Reinders, M.J.T., Tax, D.M.J.: Pate: Proximity-aware time series anomaly evaluation. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Association for Computing Machinery, New York, NY, USA (Aug 2024)

13. Huet, A., Navarro, J.M., Rossi, D.: Local evaluation of time series anomaly detection algorithms. In: Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 635—-645. Association for Computing Machinery, New York, NY, USA (2022)
14. Hundman, K., Constantinou, V., Laporte, C., Colwell, I., Söderström, T.: Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining. pp. 387–395. ACM (2018)
15. Hwang, W.S., Yun, J.H., Kim, J., Kim, H.C.: Time-series aware precision and recall for anomaly detection: considering variety of detection result and addressing ambiguous labeling. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management. pp. 2241–2244 (2019)
16. Hwang, W.S., Yun, J.H., Kim, J., Min, B.G.: Do you know existing accuracy metrics overrate time-series anomaly detections? In: Proceedings of the ACM/SIGAPP Symposium on Applied Computing. pp. 403—412. Association for Computing Machinery, New York, NY, USA (2022)
17. Kim, S., Choi, K., Choi, H.S., Lee, B., Yoon, S.: Towards a rigorous evaluation of time-series anomaly detection. In: Proceedings of the AAAI conference on artificial intelligence. vol. 36, pp. 7194–7201 (2022)
18. Kovács, G., Sebestyen, G., Hangan, A.: Evaluation metrics for anomaly detection algorithms in time-series. Acta Universitatis Sapientiae, Informatica **11**, 113–130 (2019)
19. Lavin, A., Ahmad, S.: Evaluating real-time anomaly detection algorithms - the numenta anomaly benchmark. In: IEEE International Conference on Machine Learning and Applications, Miami, FL, USA, December 9-11. pp. 38–44 (2015)
20. Nalepa, J., Myller, M., Andrzejewski, J., Benecki, P., Piechaczek, S., Kostrzewa, D.: Evaluating algorithms for anomaly detection in satellite telemetry data. Acta Astronautica **198**, 689–701 (2022)
21. Paparrizos, J., Boniol, P., Palpanas, T., Tsay, R., Elmore, A., Franklin, M.: Volume under the surface: A new accuracy evaluation measure for time-series anomaly detection. Proceedings of the VLDB Endowment **15**, 2774–2787 (09 2022)
22. Paparrizos, J., Kang, Y., Boniol, P., Tsay, R.S., Palpanas, T., Franklin, M.J.: TSB-UAD: an end-to-end benchmark suite for univariate time-series anomaly detection. Proceedings of the VLDB Endowment **15**(8), 1697—-1711 (Apr 2022)
23. Ren, H., Xu, B., Wang, Y., Yi, C., Huang, C., Kou, X., Xing, T., Yang, M., Tong, J., Zhang, Q.: Time-series anomaly detection service at microsoft. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining. pp. 3009—3017. Association for Computing Machinery, New York, NY, USA (2019)
24. Rigas, S., Tzouveli, P., Kollias, S.: An end-to-end deep learning framework for fault detection in marine machinery. Sensors, 2024, **24**(16) (2024)
25. Scharwächter, E., Müller, E.: Statistical Evaluation of Anomaly Detectors for Sequences. In: Proceedings of the ACM SIGKDD Workshop on Mining and Learning from Time Series. Association for Computing Machinery, San Diego, CA, USA (2020)
26. Sørbø, S., Ruocco, M.: Navigating the metric maze: a taxonomy of evaluation metrics for anomaly detection in time series. Data Mining and Knowledge Discovery **38**, 1–42 (11 2023)
27. Tatbul, N., Lee, T.J., Zdonik, S., Alam, M., Gottschlich, J.: Precision and recall for time series. In: Advances in Neural Information Processing Systems 31. pp. 1924–1934 (2018)

28. Xu, H., Chen, W., Zhao, N., Li, Z., Bu, J., Li, Z., Liu, Y., Zhao, Y., Pei, D., Feng, Y., Chen, J., Wang, Z., Qiao, H.: Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In: Proceedings of the International World Wide Web Conference. pp. 187—196. Republic and Canton of Geneva, CHE (2018)
29. Yan, P., Abdulkadir, A., Luley, P.P., Rosenthal, M., Schatte, G.A., Grewe, B.F., Stadelmann, T.: A comprehensive survey of deep transfer learning for anomaly detection in industrial time series: Methods, applications, and directions. IEEE Access **12**, 3768–3789 (2024)
30. Yang, K., Liu, J., Song, Y., Yang, S., Zhou, Y.: A problem-oriented taxonomy of evaluation metrics for time series anomaly detection (2025)
31. Zamanzadeh Darban, Z., Webb, G.I., Pan, S., Aggarwal, C., Salehi, M.: Deep learning for time series anomaly detection: A survey. ACM Computing Surveys, 2024, **57**(1) (Oct 2024)
32. Zhong, Z., Yu, Z., Cheung, Y.m., Yang, K.: Cce: Confidence-consistency evaluation for time series anomaly detection. arXiv preprint arXiv:2509.01098 (2025)
33. Zhong, Z., Yu, Z., Xi, X., Xu, Y., Cao, W., Yang, Y., Yang, K., You, J.: SimAD: A simple dissimilarity-based approach for time-series anomaly detection. IEEE Transactions on Neural Networks and Learning Systems pp. 1–12 (2025)