Framework para la evaluación de métodos de explicabilidad en inteligencia artificial

María Teresa Alba[1], José María Luna[1,2], and Amelia Zafra[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)
{i12alrum,jmluna,azafra}@uco.es

Resumen La evaluación de métodos de Inteligencia Artificial Explicable continúa siendo un problema abierto debido a la diversidad de propiedades consideradas, la falta de uniformidad en la definición de las métricas y la dificultad para realizar comparaciones homogéneas entre técnicas de explicabilidad. Para abordar estas limitaciones, este trabajo propone un marco unificado basado en dos contribuciones principales. En primer lugar, se presenta una taxonomía que organiza las métricas de acuerdo con la propiedad de explicabilidad que evalúan. En segundo lugar, se desarrolla un framework modular que permite generar y evaluar explicaciones de forma automatizada y reproducible. La herramienta separa la configuración experimental, la generación de explicaciones, el cálculo de las métricas y la generación de informes, favoreciendo la reproducibilidad y la comparación sistemática de explicaciones obtenidas por diferentes métodos de explicabilidad.

Keywords: Inteligencia Artificial Explicable · Evaluación de Explicaciones · Taxonomía

1. Introducción

El rápido avance de las técnicas de inteligencia artificial (IA) ha transformado numerosos sectores, entre ellos la sanidad, las finanzas, la ciberseguridad y la logística. No obstante, este progreso también ha puesto de manifiesto una limitación importante: la falta de transparencia de muchos de estos modelos, especialmente de aquellos considerados sistemas de caja negra [23]. En muchos contextos reales, no solo es importante obtener predicciones precisas, sino que también resulta fundamental comprender cómo y por qué un modelo toma determinadas decisiones, especialmente cuando estas deben ser interpretables, auditables o justificables [20].

En este contexto, la Inteligencia Artificial Explicable (*Explainable Artificial Intelligence*, XAI) [7] tiene como objetivo proporcionar explicaciones comprensibles sobre el comportamiento de los modelos. Entre los distintos enfoques propuestos en la literatura, los métodos de explicación local, como SHAP [12], LIME [17], *Integrated Gradients* [19] o *BreakDown* [18], han adquirido una gran

relevancia, debido a que permiten analizar la contribución de las variables de entrada en decisiones concretas, facilitando tareas como la validación de modelos, la detección de sesgos o la mejora de la confianza en sistemas de IA.

A pesar del desarrollo significativo de técnicas XAI, la evaluación de la calidad de las explicaciones sigue siendo un problema abierto y complejo [20]. A diferencia de la evaluación del rendimiento predictivo de los modelos, donde existen métricas bien establecidas y ampliamente aceptadas, en XAI no siempre se dispone de una explicación de referencia que permita determinar de forma directa la calidad de una explicación. Por ello, se han propuesto numerosos enfoques y métricas para evaluar distintas propiedades como la fidelidad, la robustez, la sensibilidad, la interpretabilidad o la complejidad [11,22]. Esta proliferación ha generado falta de uniformidad terminológica, solapamientos entre propiedades [15] e implementaciones *ad hoc* difíciles de comparar y reproducir [3,5]. En consecuencia, resulta necesario disponer de una taxonomía común y de un marco unificado que facilite la aplicación homogénea y reproducible de las métricas.

Para abordar estas limitaciones, este trabajo presenta dos contribuciones principales. En primer lugar, se propone una taxonomía que organiza las métricas de evaluación de acuerdo con las propiedades de explicabilidad que analizan. En segundo lugar, se desarrolla una librería en Python que implementa 17 métricas organizadas según la taxonomía propuesta y proporciona un entorno reproducible para la evaluación y comparación sistemática de métodos XAI. La herramienta sigue una arquitectura modular y extensible, permite definir los experimentos mediante ficheros `YAML` y genera informes comparativos en formatos `CSV` y `JSON`. El repositorio de la librería se encuentra disponible en https://github.com/kdis-lab/XAIMetrics.

El resto del trabajo se organiza de la siguiente manera. La Sección 2 revisa los principales *frameworks* de evaluación XAI. La Sección 3 presenta la taxonomía propuesta. La Sección 4 describe la arquitectura y funcionalidad de la librería. La Sección 5 muestra un ejemplo de uso y la Sección 6 presenta las conclusiones y líneas de trabajo futuro.

2. Trabajo relacionado

En los últimos años se han desarrollado distintas herramientas *software* con el objetivo de facilitar la evaluación sistemática de métodos XAI. Entre las más relevantes destacan `Quantus` [10], `OpenXAI` [2], `AIX360` [4] y `Xplique` [9]. Aunque todas ellas comparten el objetivo general de mejorar la evaluabilidad de los métodos XAI, difieren en su alcance, diseño y funcionalidades. La Tabla 1 resume estas diferencias según dichos criterios. `Quantus` [10] destaca por su amplia cobertura de métricas y por organizarlas en distintas categorías de evaluación, lo que la convierte en una de las herramientas más completas para analizar explicaciones basadas en atribuciones. Sin embargo, su uso se centra principalmente en el cálculo de métricas y no proporciona un flujo experimental completo con configuración declarativa e informes comparativos integrados. `OpenXAI` [2] está orientada al *benchmarking* de métodos de explicación *post-hoc*, integran-

do métricas, métodos XAI y mecanismos de comparación dentro de un mismo ecosistema. No obstante, su flexibilidad es menor cuando se desea trabajar con configuraciones experimentales heterogéneas o completamente desacopladas del código. `AIX360` [4], por su parte, ofrece una colección amplia de técnicas de explicabilidad, pero no está orientada específicamente a la evaluación sistemática de explicaciones mediante una taxonomía de métricas. Finalmente, `Xplique` [9] integra métodos de atribución, visualización y métricas, especialmente en el contexto de aprendizaje profundo, aunque no proporciona un sistema declarativo de experimentos ni informes comparativos finales.

Librería	**Cobertura de métricas**	**Taxonomía explícita**	**Métodos XAI locales y agnósticos**	**Configuración con ficheros**	**Informes comparativos**
`Quantus`	37 métricas	Sí	0 métodos	No	No integrado
`OpenXAI`	11 métricas	Sí	2 métodos	No	Sí, *Leaderboards*
`AIX360`	2 métricas	No	3 métodos	No	No integrada
`Xplique`	11 métricas	Sí	4 métodos	No	No integrado
`XAIMetrics`	17 métricas	Sí	4 métodos	Sí, `YAML`	Sí, `CSV`/`JSON`

Tabla 1. Comparación esquemática entre `XAIMetrics` y otras librerías XAI.

El análisis conjunto de estas herramientas muestra que no todas combinan una organización explícita de métricas, configuración declarativa, métodos XAI integrados y generación de informes estructurados. Estas limitaciones motivan el desarrollo de la librería propuesta, centrada en la evaluación sistemática y reproducible de métodos XAI mediante una taxonomía de propiedades, configuración mediante ficheros `YAML` e informes en formatos `CSV` y `JSON`.

3. Taxonomía de las métricas

En esta sección se propone una taxonomía para organizar las métricas de evaluación de explicaciones, basada en la distinción entre métodos centrados en humanos y métodos centrados en el sistema [11,22], y complementada con categorías utilizadas en `Quantus` [10]. Las métricas se dividen en subjetivas, que requieren la intervención de usuarios, y objetivas, que se calculan automáticamente a partir del modelo, los datos y las explicaciones. Este trabajo se centra en estas últimas, al permitir evaluaciones cuantitativas y reproducibles, organizándolas según la propiedad principal que se desea evaluar:

- **Fidelidad de la atribución (*Faithfulness*)**: evalúa hasta qué punto la explicación sigue el comportamiento predictivo del modelo en términos de contribución de las características. Las métricas implementadas en esta categoría son Consistencia (*Consistency*) [8], Estimación de la fidelidad (*Faithfulness Estimate*) [15], Fidelidad (*Faithfulness*) [14], Correlación de la monotonicidad (*Monotonicity Correlation*) [15], Métrica de monotonicidad (*Monotonicity Metric*) [15], Monotonicidad (*Monotonicity*) [14], Sensibilidad-n (*Sensitivity-n*) [15] y Suficiencia (*Sufficiency*) [15].

- **Localización (*Localization*)**: mide si la evidencia explicativa se concentra en una región de interés [10]. Es una categoría especialmente habitual en problemas de visión por computador, donde se desea comprobar si la explicación resalta la zona del objeto o patrón realmente relevante.
- **Interpretabilidad (*Interpretability*)**: hace referencia al grado en que una explicación puede ser comprendida por una persona. Se divide en:
  - **Simplicidad (*Simplicity*)**: indica que la explicación debe presentarse de forma compacta, evitando complejidad innecesaria. En explicaciones basadas en reglas, por ejemplo, puede evaluarse mediante el número de reglas, la longitud media de las reglas o el número de predicados.
  - **Claridad (*Clarity*)**: evalúa si la explicación es directa, no ambigua y consistente para instancias similares. Una explicación clara debería evitar solapamientos innecesarios, contradicciones o cambios bruscos ante ejemplos equivalentes.
- **Sensibilidad (*Sensitivity*)**: analiza cuánto varía una explicación cuando se producen cambios en la entrada, en los parámetros del modelo o en el proceso de generación de la explicación. Esta categoría está estrechamente relacionada con la idea de que una explicación no debería cambiar de manera arbitraria ante perturbaciones pequeñas o irrelevantes. La métrica implementada en esta categoría es Sensibilidad media (*Avg-Sensitivity*) [21].
- **Robustez (*Robustness*)**: mide la estabilidad de las explicaciones frente a pequeñas perturbaciones de la entrada, asumiendo que la predicción del modelo permanece aproximadamente constante [10]. Una explicación robusta debería producir resultados similares para instancias muy parecidas. Las métricas implementadas son Estimación local de Lipschitz (*Local Lipschitz Estimate*) [3], Sensibilidad máxima (*Max-Sensitivity*) [21], Estabilidad relativa de la entrada (*Relative Input Stability*) [15] y Estabilidad relativa de la salida (*Relative Output Stability*) [1].
- **Complejidad (*Complexity*)**: captura hasta qué punto una explicación es concisa, es decir, si utiliza pocas características o pocos elementos para justificar la predicción del modelo [10]. Las métricas implementadas son Complejidad (*Complexity*) [17] y Dispersión (*Sparseness*) [6].
- **Fidelidad respecto al modelo (*Fidelity*)**: mide hasta qué punto la explicación representa correctamente el comportamiento del modelo de aprendizaje automático [15]. Se divide en:
  - **Completitud (*Completeness*)**: evalúa si la explicación contiene información suficiente para reflejar la dinámica del modelo o para cubrir los elementos relevantes de su proceso de decisión. La métrica implementada en esta subcategoría es Completitud (*Completeness*) [15].
  - **Solidez (*Soundness*)**: mide si la explicación es correcta y veraz respecto al comportamiento real del modelo. La métrica implementada es Insensibilidad (*Non-Sensitivity*) [13].

Las métricas enumeradas corresponden a las implementadas en la librería, orientada a explicaciones locales basadas en atribuciones sobre datos tabulares. Por ello, no incluye métricas de localización ni de interpretabilidad basadas en

modelos simbólicos, aunque estas categorías se mantienen en la taxonomía para ofrecer una visión completa y facilitar futuras ampliaciones.

4. Arquitectura y funcionalidad del framework

El *framework* se diseña como un entorno unificado para la generación y evaluación de explicaciones XAI, permitiendo analizar distintos métodos sobre múltiples modelos y conjuntos de datos bajo condiciones homogéneas. Para ello, integra una taxonomía de métricas, un módulo de explicadores, un sistema de configuración mediante `YAML` y un proceso de ejecución automatizado que abarca la generación de atribuciones, el cálculo de métricas y la creación de informes.

4.1. Arquitectura de la librería

La librería sigue una arquitectura modular y extensible que separa las distintas fases del proceso de evaluación, permitiendo incorporar nuevas métricas y nuevos métodos de explicación sin alterar el flujo de ejecución. Se organiza en seis módulos principales: `base`, `config`, `explainers`, `metrics`, `reporting` y `runner`. Estos módulos desacoplan la generación de explicaciones, la definición de las métricas, la configuración, la ejecución y la generación de resultados (Figura 1).

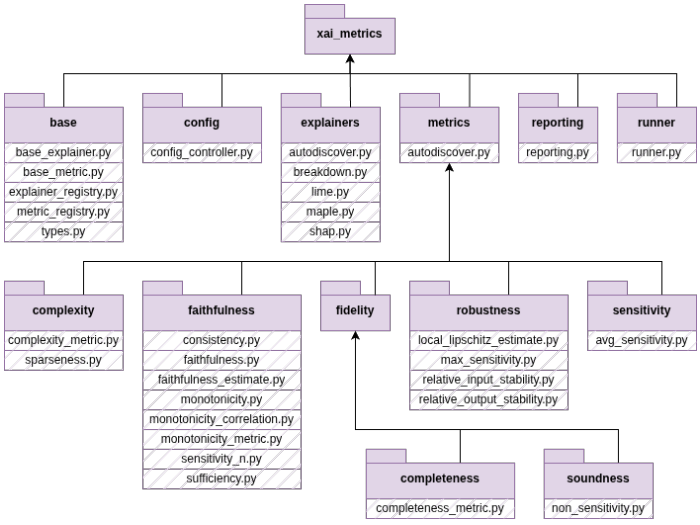


Figura 1. Diagrama de la arquitectura de la librería

El módulo `base` constituye el núcleo común de la librería. Su responsabilidad principal es definir las abstracciones y utilidades compartidas por el resto de

módulos. Incluye las clases `BaseMetric` y `BaseExplainer`, de las que heredan todas las métricas y modelos de explicabilidad, respectivamente. También define `MetricContext` y `ExplainerContext`, encargados de agrupar la información necesaria para evaluar y generar explicaciones como el modelo, los datos, las etiquetas, las atribuciones, las observaciones seleccionadas, los datos de referencia y el dispositivo de ejecución. Además, incorpora el sistema de registro automático `register_metric` y `register_explainer`, lo que permite añadir nuevos componentes sin modificar el flujo principal.

El módulo `config` se encarga de gestionar la configuración experimental. Su responsabilidad es cargar, interpretar y validar la información necesaria para definir los flujos de generación y evaluación. A partir de un fichero `YAML` o de una configuración proporcionada en tiempo de ejecución, este módulo identifica las rutas de datos, modelos y atribuciones, el dispositivo de ejecución, las métricas y los explicadores seleccionados, así como los parámetros asociados a cada componente. De este modo, los experimentos pueden modificarse y reutilizarse sin modificar el código de la librería y pueden abarcar múltiples conjuntos de datos, modelos y métodos XAI.

El módulo `explainers` contiene los métodos de explicación implementados en la librería. Su responsabilidad es generar atribuciones a partir de un modelo, las observaciones que se desean explicar y, cuando sea necesario, un conjunto de referencia y sus etiquetas. Cada explicador se implementa como una clase que hereda de `BaseExplainer`, recibe un `ExplainerContext` y define su comportamiento mediante `explain()`. La versión actual integra LIME [17], SHAP [12], BreakDown [18] y MAPLE [16]. Además, el método `as_explain_func()` permite convertir cada explicador en una función compatible con las métricas que necesitan recalcular atribuciones sobre entradas modificadas.

El módulo `metrics` contiene las métricas implementadas en la librería. Su responsabilidad es agrupar las clases concretas que calculan cada medida de evaluación. Siguiendo la taxonomía definida en la Sección 3, las métricas se organizan en submódulos según la propiedad principal que evalúan. Cada métrica se implementa como una clase que hereda de `BaseMetric`, recibe un `MetricContext` y define su cálculo específico en el método `run()`. Esta organización separa la lógica de cada métrica y facilita la incorporación de nuevas medidas siguiendo una misma estructura.

El módulo `runner` centraliza los dos flujos principales de la librería. Su responsabilidad es coordinar el resto de componentes. El primer flujo descubre los explicadores registrados, construye los objetos `ExplainerContext`, ejecuta los métodos XAI configurados y almacena las atribuciones generadas. El segundo flujo descubre las métricas registradas, construye los objetos `MetricContext` y ejecuta cada métrica sobre las combinaciones disponibles de conjunto de datos, modelo y método XAI. El módulo también permite inyectar dependencias externas, como cargadores personalizados de modelos o funciones de explicación. De este modo, actúa como punto de conexión entre la configuración, los explicadores, las métricas y los informes.

El módulo `reporting` se encarga de organizar y exportar los resultados obtenidos. Su responsabilidad es agregar los valores individuales calculados por las métricas y reunir en informes comparativos los conjuntos de datos, los modelos, las métricas, sus parámetros y los resultados de cada método XAI. Los informes pueden exportarse en formatos `CSV` y `JSON`. Además, las atribuciones producidas por los explicadores pueden guardarse como ficheros `CSV`, lo que facilita su inspección, reutilización y evaluación posterior sin necesidad de volver a generarlas.

4.2. Funcionalidades de la librería

La librería proporciona un flujo estructurado y automatizado para la generación y evaluación de explicaciones, diseñado para facilitar la realización de estudios experimentales sistemáticos y reproducibles. Este flujo se organiza en las siguientes fases:

1. Configuración del experimento mediante fichero `YAML`, donde se definen modelos, conjunto de datos, explicaciones y métricas a evaluar. De este modo, los experimentos pueden modificarse y reutilizarse sin necesidad de alterar el código de la librería.
2. Construcción de contextos de evaluación, que integran todos los elementos necesarios para el cálculo de métricas.
3. Generación automática de explicaciones que permite ejecutar los métodos XAI configurados sobre cada combinación de conjunto de datos y modelo. La librería permite utilizar atribuciones previamente almacenadas o ejecutar automáticamente los métodos XAI implementados. Las atribuciones generadas pueden guardarse para su análisis o reutilización posterior.
4. Ejecución automática que permite evaluar todas las métricas sobre cada combinación de modelo, *dataset* y método XAI de forma automatizada. Estas métricas se organizan de acuerdo con la taxonomía propuesta, lo que facilita la interpretación de las propiedades de explicabilidad evaluadas.
5. Agregación y análisis de resultados, los resultados se procesan y organizan para su posterior análisis. La librería conserva tanto los resultados agregados como valores asociados a observaciones individuales cuando la métrica lo permite.
6. Generación de informes que permiten comparar directamente distintos métodos XAI.

Este flujo garantiza que las explicaciones y las métricas se obtengan bajo condiciones homogéneas, favoreciendo la comparabilidad entre métodos XAI y la reproducibilidad de los experimentos. De este modo, el *framework* integra de forma coherente la configuración, la generación de explicaciones, la evaluación y la creación de informes dentro de un mismo entorno experimental.

```yaml
context:
  device: "cpu"
  datasets_dir: "data"
  models_dir: "models"
  attributions_dir: "attributions"

explainers:
  - name: LIME
    params:
      num_samples: 5000
  - name: SHAP
    params:
      max_background_samples: 100
      abs: true

metrics:
  - name: Complexity
    params:
      normalise: true
  - name: Sparseness
    params:
      normalise: true
  - name: Faithfulness
    params:
      base_strategy: mean
  - name: LocalLipschitzEstimate
    params:
      nr_samples: 200
      normalise: true
```

Listado 1: Configuración de experimento (`config.yaml`).

5. Ejemplo de uso

En esta sección se muestra un ejemplo de uso cuyo objetivo es mostrar cómo definir, ejecutar y analizar un experimento de evaluación de métodos XAI de forma automatizada a partir de una configuración inicial.

5.1. Configuración del experimento

La configuración del experimento se define mediante un fichero `YAML`, en el que se especifican el contexto, las métricas a evaluar y los métodos de explicación que se deben ejecutar. En el ejemplo considerado, el contexto se define mediante los campos `datasets_dir`, donde se encuentran los datos del conjunto *hydraulic*; `models_dir`, donde se almacena el modelo `IForest`; y `attributions_dir`, que actúa como directorio de entrada o salida para las atribuciones. El bloque `metrics` especifica las métricas que se van a evaluar mediante el campo `name` y sus parámetros mediante `params`. En el ejemplo, se analizan Complejidad (*Complexity*), Dispersión (*Sparseness*), Fidelidad (*Faithfulness*) y Estimación local de Lipschitz (*Local Lipschitz Estimate*). De forma análoga, el bloque `explainers` especifica los métodos de explicación que se van a ejecutar mediante el campo `name` y sus parámetros mediante `params`. En el ejemplo, se utilizan LIME y SHAP para generar las atribuciones que se evalúan posteriormente mediante las métricas configuradas. El formato de los ficheros del contexto y los parámetros de cada métrica y explicador están detallados en el manual de usuario de la librería (https://github.com/kdis-lab/XAIMetrics). El fichero de configuración para este experimento se muestra en el Listado 1 y también está disponible en el repositorio de la librería para facilitar su uso.

5.2. Ejecución del experimento

Antes de la ejecución de las métricas, las atribuciones se generan mediante `run_explanation()`, que actúa como punto de entrada al flujo de explicación.

La función carga las combinaciones de datos y modelos, ejecuta los explicadores configurados y almacena sus salidas en el directorio indicado, organizadas por conjunto de datos, modelos y método XAI.

```
from xai_metrics.runner import run_explanation

explanation = run_explanation(config="experiments/config.yaml",
↪  attribution_output_dir="attributions")
```

Listado 2: Ejecución de la función `run_explanation()`.

La salida incluye los contextos de explicación, las matrices de atribuciones generadas, los parámetros empleados por cada explicador y las rutas de los ficheros `CSV`. De esta forma, los explicadores pueden evaluarse inmediatamente o reutilizarse en ejecuciones posteriores.

Una vez generadas las atribuciones, la evaluación se realiza mediante la función `run_evaluation()`, que lleva a cabo el proceso completo de evaluación (Listado 3). El parámetro `config` indica el fichero `YAML` y `report_output_dir` especifica el directorio de informes. La función construye los contextos de evaluación, ejecuta las métricas y devuelve los resultados individuales, los informes comparativos y las rutas de los ficheros exportados.

```
from xai_metrics.runner import run_evaluation

evaluation = run_evaluation(config="experiments/config.yaml",
↪  report_output_dir="results/reports")
```

Listado 3: Ejecución de la función `run_evaluation()`

5.3. Resultados y análisis

La ejecución del experimento da como salida un diccionario con cuatro campos principales: `contexts`, `reports`, `observation_reports` y `report_paths`. De forma simplificada, su estructura puede representarse como se muestra en el Listado 4. El campo `contexts` contiene los resultados individuales de cada combinación de configuración, conjunto de datos, modelo y método XAI. El campo `reports` agrupa estos resultados en tablas comparativas, mientras que `observation_reports` conserva los valores detallados por observación para cada métrica. Finalmente, `report_paths` recoge las rutas de todos los ficheros `CSV` y `JSON` generados.

El Listado 5 muestra el informe generado para el conjunto de datos *hydraulic*, el modelo `IForest` y los métodos LIME y SHAP. Las columnas `dataset_name` y

```
{"contexts": [{"metadata": {"dataset_name": "hydraulic",
      "model_name": "IForest",
      "xai_method_name": "LIME"},
  "observations": [...],
  "results": [...],
  "skipped": []},
  {"metadata": {"dataset_name": "hydraulic",
      "model_name": "IForest",
      "xai_method_name": "SHAP"},
   "observations": [...],
   "results": [...],
   "skipped": []],
 "reports": {"hydraulic": {"IForest": "<DataFrame agregado>"}},
 "observation_reports": {"Complexity": {...}},
   "Sparseness": {...},
   "Faithfulness": {...},
   "LocalLipschitzEstimate": {...}},
 "report_paths": {"csv": "results/reports/xai_metrics_report.csv",
   "json": "results/reports/xai_metrics_report.json",
   "observation_csv": {"Complexity": "Complexity/Complexity.csv",
     "Sparseness": ...,
     "Faithfulness": ...,
     "LocalLipschitzEstimate": ...}}}
```


Listado 4: Resultado abreviado de la evaluación.

`model_name` identifican el contexto evaluado, mientras que las columnas `metric` y `metric_params` indican la métrica ejecutada y los parámetros utilizados. Las columnas `LIME` y `SHAP` contienen los resultados agregados de ambos métodos de explicación. Esta estructura facilita la consulta, el filtrado y la comparación de las distintas métricas y métodos XAI, y permite distinguir ejecuciones de una misma métrica realizadas con parámetros diferentes.


```
dataset_name,model_name,metric,metric_params,LIME,SHAP
hydraulic,IForest,Complexity,"{""normalise"": true}",1.218711,1.498500
hydraulic,IForest,Faithfulness,...,-0.481378,0.407578
hydraulic,IForest,LocalLipschitzEstimate,...,19.436659,7.889390
hydraulic,IForest,Sparseness,...,0.532481,0.385798
```


Listado 5: Informe resultado de la evaluación.

6. Conclusiones

En este trabajo se ha avanzado hacia un enfoque unificado para la evaluación de explicaciones en XAI mediante la propuesta de una taxonomía que organiza

las métricas según la propiedad evaluada y el desarrollo de un *framework* modular y extensible. La librería implementa 17 métricas de evaluación e incorpora un módulo de explicadores que integra métodos locales basados en atribuciones, permitiendo generar y almacenar explicaciones dentro del mismo entorno en el que se evalúan. Además, permite definir mediante ficheros `YAML` las combinaciones de datos, modelos, métricas y explicadores, ejecutar automáticamente ambos flujos y generar informes comparativos en formatos `CSV` y `JSON`. De este modo, se proporciona un entorno estructurado que favorece la reproducibilidad, la homogeneidad de las evaluaciones y la incorporación progresiva de nuevas métricas y métodos XAI. Como trabajo futuro, se plantea ampliar la cobertura de métricas, incorporar nuevos explicadores y extender la librería a otros tipos de explicaciones y modalidades de datos. También resulta relevante mejorar la compatibilidad de los métodos XAI con modelos heterogéneos y estudiar el coste computacional asociado a la generación y regeneración de atribuciones durante la evaluación.

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.

Referencias

1. Agarwal, C., Johnson, N., Pawelczyk, M., et al.: Rethinking stability for attribution-based explanations. In: ICLR 2022 Workshop on PAIR2Struct (2022), https://openreview.net/forum?id=BfxZAuWOg9
2. Agarwal, C., Krishna, S., Saxena, E., et al.: OpenXAI: Towards a transparent evaluation of model explanations. In: Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (2022), https://openreview.net/forum?id=MU2495w47rz
3. Alvarez Melis, D., Jaakkola, T.: Towards robust interpretability with self-explaining neural networks. Advances in neural information processing systems **31** (2018)
4. Arya, V., Bellamy, R.K.E., Chen, P.Y., et al.: Ai explainability 360: An extensible toolkit for understanding data and machine learning models. Journal of Machine Learning Research **21**(130), 1–6 (2020), http://jmlr.org/papers/v21/19-1035.html
5. Bhatt, U., Weller, A., Moura, J.M.F.: Evaluating and aggregating feature-based model explanations. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20). pp. 3016–3022. International Joint Conferences on Artificial Intelligence Organization (2020). https://doi.org/10.24963/ijcai.2020/417
6. Chalasani, P., Chen, J., Chowdhury, A.R., et al.: Concise explanations of neural networks using adversarial training. In: International conference on machine learning. pp. 1383–1391. PMLR (2020)

7. Cummins, L., Sommers, A., Ramezani, S.B., et al.: Explainable predictive maintenance: A survey of current methods, challenges and opportunities. IEEE Access **12**, 57574–57597 (2024). https://doi.org/10.1109/ACCESS.2024.3391130
8. Dasgupta, S., Frost, N., Moshkovitz, M.: Framework for evaluating faithfulness of local explanations. In: International Conference on Machine Learning. pp. 4794–4815. PMLR (2022)
9. Fel, T., Hervier, L., Vigouroux, D., Pothers: Xplique: A deep learning explainability toolbox. Workshop on Explainable Artificial Intelligence for Computer Vision (CVPR) (2022)
10. Hedström, A., Weber, L., Krakowczyk, D., et al.: Quantus: An explainable ai toolkit for responsible evaluation of neural network explanations and beyond. Journal of Machine Learning Research **24**(34), 1–11 (2023)
11. Lopes, P., Silva, E., Braga, C., Oliveira, T., Rosado, L.: Xai systems evaluation: A review of human and computer-centred methods (2022). https://doi.org/10.3390/app12199423
12. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Advances in Neural Information Processing Systems 30, pp. 4765–4774 (2017), http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf
13. Nguyen, A.p., Martínez, M.R.: On quantitative aspects of model interpretability. arXiv preprint arXiv:2007.07584 (2020)
14. Oblizanov, A., Shevskaya, N., Kazak, A., et al.: Evaluation metrics research for explainable artificial intelligence global methods using synthetic data. Applied System Innovation **6**(1) (2023). https://doi.org/10.3390/asi6010026
15. Pawlicki, M., Pawlicka, A., Uccello, F., et al.: Evaluating the necessity of the multiple metrics for assessing explainable ai: A critical examination. Neurocomputing **602** (2024). https://doi.org/10.1016/j.neucom.2024.128282
16. Plumb, G., Molitor, D., Talwalkar, A.S.: Model agnostic supervised local explanations. Advances in neural information processing systems **31** (2018)
17. Ribeiro, M.T., Singh, S., Guestrin, C.: "why should I trust you?": Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016. pp. 1135–1144 (2016)
18. Staniak, M., Biecek, P.: Explanations of model predictions with live and breakdown packages. The R Journal **10**(2), 395–409 (2018). https://doi.org/10.32614/RJ-2018-072
19. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: International conference on machine learning. pp. 3319–3328. PMLR (2017)
20. Wiratsin, I.O., Ragkhitwetsagul, C.: Effectiveness of explainable artificial intelligence (xai) techniques for improving human trust in machine learning models: A systematic literature review. IEEE Access (2025)
21. Yeh, C.K., Hsieh, C.Y., Suggala, A., et al.: On the (in) fidelity and sensitivity of explanations. Advances in neural information processing systems **32** (2019)
22. Zhou, J., Gandomi, A.H., Chen, F., Holzinger, A.: Evaluating the quality of machine learning explanations: A survey on methods and metrics (2021). https://doi.org/10.3390/electronics10050593
23. ŞAHiN, E., Arslan, N.N., Özdemir, D.: Unlocking the black box: an in-depth review on interpretability, explainability, and reliability in deep learning (2025). https://doi.org/10.1007/s00521-024-10437-2