Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Ciencia de Datos e Inteligencia Artificial

Trabajo Fin de Grado

Integración y evaluación de modelos interpretables utilizando la librería interpretML

Autor: Javier Pérez Vargas
Tutor(a): Bojan Mihaljevic

Madrid, Junio 2025

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ciencia de Datos e Inteligencia Artificial

Título: Integración y evaluación de modelos interpretables utilizando la librería interpretML

Junio 2025

Autor: Javier Pérez Vargas
Tutor: Bojan Mihaljevic
Departamento de Inteligencia Artificial
Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

Resumen

En la actualidad, los sistemas de inteligencia artificial han adquirido un papel fundamental en múltiples ámbitos de la sociedad, desde asistentes personales y motores de recomendación hasta herramientas utilizadas en medicina, banca y justicia. Sin embargo, mientras que muchos modelos avanzados como XGBoost o redes neuronales profundas ofrecen altos niveles de precisión, suelen ser considerados "cajas negras" (blackbox), es decir, su proceso de decisión resulta opaco para los usuarios. Esta falta de transparencia supone una barrera en dominios donde las decisiones automatizadas deben ser justificables y confiables. En este contexto, la Inteligencia Artificial Explicable (XAI) surge como una rama esencial del aprendizaje automático, centrada en desarrollar modelos que permitan entender, analizar y confiar en sus predicciones.

La mayor parte del desarrollo de este Trabajo Fin de Grado se ha llevado a cabo en el marco de una Beca de Colaboración en Departamentos Universitarios, convocada por el Ministerio de Educación, con el objetivo de fomentar la iniciación en tareas de investigación. La resolución de concesión fue publicada en noviembre de 2024 y, desde entonces, he estado trabajando en este proyecto, profundizando en el campo de la Inteligencia Artificial Explicable (XAI) y colaborando con el departamento de IA para integrar y evaluar herramientas relacionadas con modelos transparentes de aprendizaje automático.

Este Trabajo Fin de Grado tiene como objetivo contribuir a este campo mediante la integración y evaluación de modelos interpretables utilizando la librería InterpretML, un marco reconocido en la comunidad de XAI. Se ha desarrollado una extensión de dicha librería, denominada *interpret-extension*, que amplía sus capacidades para incluir nuevos modelos que hasta ahora no eran compatibles. Entre los modelos incorporados se encuentran clasificadores probabilísticos como Naive Bayes, Tree Augmented Naive Bayes (TAN) y Análisis Discriminante Lineal (LDA), así como modelos aditivos más modernos como Neural Additive Models (NAM). Esta integración se ha diseñado respetando la estructura de InterpretML, permitiendo generar explicaciones tanto globales como locales mediante representaciones gráficas.

El desarrollo de la extensión ha incluido la creación de clases específicas para cada modelo, la implementación de métodos de visualización compatibles con el ecosistema de InterpretML y la publicación del paquete en el repositorio oficial PyPI. A través de esta herramienta, se busca facilitar el uso de modelos explicables en entornos reales, permitiendo a profesionales y desarrolladores acceder

a modelos transparentes. La validación de estas implementaciones se ha llevado a cabo mediante una evaluación en condiciones reproducibles utilizando el benchmark TabZilla, una plataforma que permite comparar modelos de machine learning en diversos conjuntos de datos.

La experimentación se ha centrado en 15 datasets de clasificación binaria, empleando métricas como precisión (accuracy), log loss, y tiempos de entrenamiento e inferencia. Los resultados obtenidos muestran que, si bien los modelos de boosting como CatBoost y XGBoost siguen liderando en términos de rendimiento bruto, modelos como EBM han alcanzado precisiones comparables, situándose incluso como el modelo más efectivo en varios datasets. Este hallazgo refuerza la idea de que es posible lograr un equilibrio entre interpretabilidad y rendimiento, especialmente en entornos donde la comprensión de las decisiones es tan importante como la precisión del modelo.

En conclusión, este trabajo no solo presenta una contribución mediante la extensión de una librería de referencia en el campo, sino que también muestra el potencial de los modelos explicables como una alternativa frente a los modelos opacos.

Abstract

Nowadays, artificial intelligence systems have taken on a key role in many areas of society, from personal assistants and recommendation engines to tools used in medicine, banking, and justice. However, although many advanced models like XGBoost or deep neural networks offer high levels of accuracy, they are often considered "blackbox", meaning their decision-making process is not clear to users. This lack of transparency becomes a challenge in fields where automated decisions need to be justified and trustworthy. In this context, Explainable Artificial Intelligence (XAI) has emerged as an essential branch of machine learning, focused on developing models that allow us to understand, analyze, and trust their predictions.

The majority of this Final Degree Project has been carried out within the framework of a Collaboration Grant in University Departments, funded by the Spanish Ministry of Education, with the aim of promoting early involvement in research activities. The award resolution was published in November 2024, and since then, I have been working on this project, delving into the field of Explainable Artificial Intelligence (XAI) and collaborating with the Department of Artificial Intelligence to integrate and evaluate tools related to transparent machine learning models.

This Final Degree Project aims to contribute to this field through the integration and evaluation of interpretable models using the InterpretML library, a framework widely recognized within the XAI community. An extension of this library has been developed, called *interpret-extension*, which expands its capabilities to include new models that were previously unsupported. Among the incorporated models are probabilistic classifiers such as Naive Bayes, Tree Augmented Naive Bayes (TAN), and Linear Discriminant Analysis (LDA), as well as more modern additive models like Neural Additive Models (NAM). This integration was designed to align with InterpretML's architecture, enabling the generation of both global and local explanations through graphical representations.

The development of the extension included the creation of specific classes for each model, the implementation of visualization methods compatible with the InterpretML ecosystem, and the publication of the package in the official PyPI repository. With this tool, the goal is to make explainable models easier to use in real-world environments, allowing professionals and developers to access transparent models. The validation of these implementations was carried out through evaluation under reproducible conditions using the TabZilla bench-

mark, a platform that allows comparison of machine learning models across various datasets.

The experimentation focused on 15 binary classification datasets, using metrics such as accuracy, log loss, and training and inference times. The results show that, although boosting models like CatBoost and XGBoost still lead in terms of raw performance, models like EBM achieved comparable accuracy, even ranking as the most effective model in several datasets. This finding supports the idea that it is possible to find a balance between interpretability and performance, especially in areas where understanding the model's decisions is as important as its accuracy.

In conclusion, this project not only presents a contribution through the extension of a key library in the field, but also provides evidence of the potential of explainable models as a real alternative to traditional black-box models.

Tabla de contenidos

Capítulo 1

Introducción

1.1. Contexto

En los últimos años los sistemas de Inteligencia Artificial han pasado a formar parte de nuestras vidas en muchos aspectos: asistentes virtuales, sistemas de recomendación, traducción automática, coches autónomos... La mayoría de estas aplicaciones no requieren que la respuesta ofrecida por el sistema explique cómo ha llegado a tomar esa decisión. No obstante, existen otros usos donde esto es muy relevante: ¿Aceptaríamos el diagnóstico médico generado por una Inteligencia Artificial? ¿Confiaríamos en una IA para decidir si una persona obtiene un préstamo bancario? ¿Aceptaríamos una sentencia judicial generada por un sistema de IA? La respuesta, seguramente, sería que no.

Durante los años 70 y 80 se comenzaron a desarrollar los primeros sistemas de Inteligencia Artificial capaces de resolver tareas específicas, como MYCIN o GUIDON, que se basaban en un conjunto de reglas e incluso podían explicar cuál de estas reglas contribuía a la decisión tomada, habiendo sido muy estudiado su uso en el dominio médico [1].

No obstante, a partir de los años 90 se popularizaron las redes neuronales, opacas en cuanto a su interpretabilidad, y otros métodos eficaces en muchas tareas pero incapaces de explicar por qué realizaban una predicción concreta. Estos son los métodos conocidos como "caja negra" (*Blackbox*), por lo general bastante complejos y donde la enorme cantidad de parámetros no permite interpretar sus decisiones. Esta preocupación por la opacidad de los modelos complejos no es nueva, y ya se viene discutiendo desde hace décadas en la literatura científica [2], [3].

Es aquí donde surge la **Inteligencia Artificial Explicable** (*XAI*), un campo de investigación de la IA que abarca un conjunto de métodos que permiten al ser humano comprender por qué se ha tomado una u otra decisión. Aquí se encuentran los modelos conocidos como "transparentes" (*Glassbox*) [4], así como técnicas que permiten interpretar los modelos *Blackbox* [5]. La investigación en *XAI* ha crecido notablemente en los últimos años, dada la necesidad de tener sistemas que ofrezcan respuestas transparentes. Al mismo tiempo, también se ha

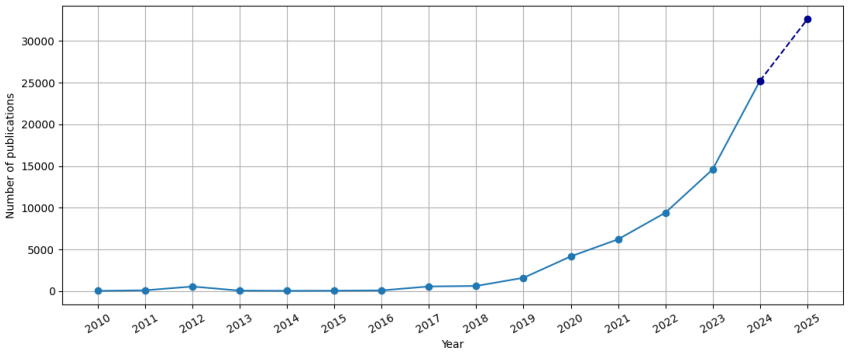

Figura 1.1: Publicaciones relacionadas con *XAI* en los últimos años.

abierto un debate en la comunidad científica sobre los límites, retos y oportunidades reales de estas técnicas, tanto en términos técnicos como sociales [6], [7]. Este debate ha sido clave para entender no solo lo que XAI puede ofrecer, sino también en qué contextos sus explicaciones son realmente útiles o necesarias.

Una de las iniciativas que trabajan en ello es **InterpretML** [8], un paquete de código abierto (desarrollado por Microsoft) que incorpora las técnicas de interpretabilidad de machine learning que conforman el estado del arte. Esta librería ofrece un marco para la interpretabilidad de modelos de machine learning, permitiendo aplicar, comparar y visualizar explicaciones de modelos. InterpretML incorpora una selección representativa de métodos de interpretación tanto intrínsecos como pos-hoc, incluyendo modelos totalmente interpretables como los Explainable Boosting Machines (EBMs) y técnicas de explicación local como LIME o SHAP. Así, se ha consolidado como una herramienta de referencia en el desarrollo de soluciones de aprendizaje automático explicables.

En el ámbito del aprendizaje automático aplicado a datos tabulares, existe una amplia variedad de modelos interpretables, que van desde enfoques probabilísticos como Naive Bayes hasta modelos más avanzados como Explainable Boosting Machine (EBM), una variante de los Generalized Additive Models (GAMs). Sin embargo, los modelos que actualmente representan el estado del arte en términos de rendimiento, como XGBoost o las redes neuronales profundas, suelen carecer de interpretabilidad. Por ello, es importante desarrollar algoritmos que sean explicables y al mismo tiempo alcancen un rendimiento similar o cercano a los modelos mencionados, como ya se ha estudiado estos últimos años [9] [10].

El desarrollo de este trabajo consistirá en la integración de modelos interpretables en el framework InterpretML, llevando a cabo también una evaluación exhaustiva de comparación entre diferentes modelos para determinar en qué medida los modelos interpretables pueden aproximarse al rendimiento de las soluciones más avanzadas [11], asegurando un equilibrio entre precisión y explicabilidad.

1.2. Objetivos

Los objetivos que se plantean en este proyecto son los siguientes:

- O1: Integración de clasificadores basados en redes bayesianas en la librería interpretML.
- O2: Adaptación de las funcionalidades de explicación de interpretML a los nuevos modelos integrados.
- O3: Aplicación de los modelos integrados en conjuntos de datos reales y simulados.
- O4: Integración de modelos aditivos generalizados basados en redes neuronales en la librería interpretML.
- O5: Comparativa sistemática de distintos modelos interpretables en términos de su capacidad predictiva y su interpretabilidad.

Capítulo 2

Marco Teórico

2.1. Modelos Probabilísticos

Los métodos probabilísticos forman una de las ramas principales del Machine Learning, representando la base tradicional de este campo. Se basan en la aplicación de la estadística al análisis de datos, proporcionando predicciones en forma de probabilidades. No se consideran a menudo entre los modelos con mayor rendimiento (aunque, dependiendo del contexto, pueden llegar a tener resultados similares), pero muchos de ellos tienen la propiedad de interpretabilidad, por lo que serán objeto de estudio en este trabajo.

Más concretamente, y definiendo $\mathbf{X} = \{X_1, ..., X_n\}$ como el vector de características y C como la clase, los modelos que toman la forma $P(\mathbf{X}, C) = P(C)P(\mathbf{X}|C)$ se conocen como **modelos generativos**. Estos modelos estiman la distribución de probabilidad conjunta (JPD), lo que les permite crear muestras x de cada clase C. Esto los distingue de los **modelos discriminativos**, que 'simplemente' permiten discriminar el valor de la variable C dado X. Los modelos discriminantes suelen tener un entrenamiento más complejo, basado en métodos iterativos para resolver un problema de optimización; sin embargo, los modelos generativos suelen entrenarse de una sola pasada.

Durante esta sección estudiaremos diferentes modelos generativos.

2.1.1. Naive Bayes

Los clasificadores bayesianos, como Naive Bayes, asignan la clase más probable a un ejemplo dado descrito por su vector de características. Es un tipo de modelo generativo, es decir, que busca modelar la distribución de los datos de entrada de una clase o categoría.

Naive Bayes se basa en el Teorema de Bayes, y la suposición que realiza este modelo, y a la que debe su nombre, es que asume que las características son condicionalmente independientes, es decir, son independientes dentro de la misma clase. Esto permite crear un modelo más eficiente y manejable a nivel computacional. Su fórmula viene dada como:

$$P(C \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid C)\ P(C)}{P(\mathbf{X})} = \frac{\prod_{i=1}^{n} P(X_i \mid C)\ P(C)}{P(\mathbf{X})} \tag{2.1}$$

A pesar de esta simplificación, Naive Bayes ha demostrado un gran rendimiento en problemas como la clasificación de textos (spam/no spam). Suele demostrar mejores resultados que otros modelos en problemas de alta dimensionalidad.

Existen diferentes variantes de Naive Bayes, dependiendo de la manera en que se calcule $P(X_i|C)$; es decir, en la asunción de distribución que se haga sobre los datos con los que se está trabajando:

- **Naive Bayes Gaussiano**: Como su nombre indica, asume una distribución normal de los datos. Por tanto, es más adecuado para datos continuos.
- **Naive Bayes Categórico**: Es la variante adecuada para trabajar con variables discretas.
- **Naive Bayes Bernouilli**: Se utiliza con variables booleanas, es decir, aquellas que solo pueden tener 2 valores distintos.
- **Naive Bayes Multinomial**: Esta variante asume que los datos provienen de distribuciones multinomiales, y es útil cuando se trabaja, por ejemplo, con conteos de frecuencias.

2.1.2. Tree Augmented Naive Bayes (TAN)

El modelo Tree Augmented Naive Bayes (TAN) constituye una variación del modelo Naive Bayes, que consiste en relajar la condición de independencia entre las diferentes variables. Para ello utiliza una estructura de árbol, permitiendo que cada característica dependa de otra característica además de la clase.

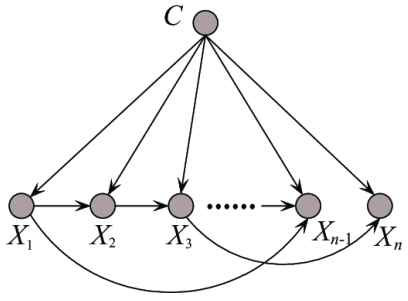

Figura 2.1: Ejemplo del modelo TAN. Imagen de [12]

Estas nuevas conexiones se corresponden con dependencias adicionales entre las características, que se establecen formando un árbol de máxima ponderación. De esta manera, la fórmula sufre un pequeño cambio, pues debemos de tener en cuenta la dependencia entre características. Si definimos T como la estructura del árbol que captura estas dependencias, la probabilidad de una clase dada una observación X se expresa como:

$$P(C \mid \mathbf{X}) = \frac{\prod_{i=1}^{n} P(X_i \mid X_{\pi(i)}, C)\ P(C)}{P(\mathbf{X})} \tag{2.2}$$

donde $\pi(i)$ representa la característica padre de X_i en el árbol T. Esta modificación permite capturar relaciones entre características sin perder la eficiencia computacional de Naive Bayes, lo que puede mejorar la precisión en muchos problemas de clasificación.

Para elegir qué características deben estar conectadas por estos arcos adicionales se suele aplicar el método de Chow-Liu, que se basa en utilizar la información mutua de todas las combinaciones de variables y conectar aquellas que tengan un valor más alto.

La manera en que se calcula la probabilidad condicional, como en Naive Bayes, también depende de la asunción de distribución que se haga sobre los datos. Es decir, existen las mismas variantes de TAN que las mencionadas en Naive Bayes.

2.1.3. Redes Bayesianas

Las redes bayesianas son modelos que representan una probabilidad de distribución conjunta a partir de los datos, y se definen por:

- Un grafo acíclico dirigido G, con vértices V y aristas E, de manera que cada $v \in V$ se corresponde con una variable X_i en $\mathbf{X}$
- Los parámetros θ, donde $\Theta_i \subset \theta$ especifica la distribución de probabilidad condicional de X_i dados sus padres en G.

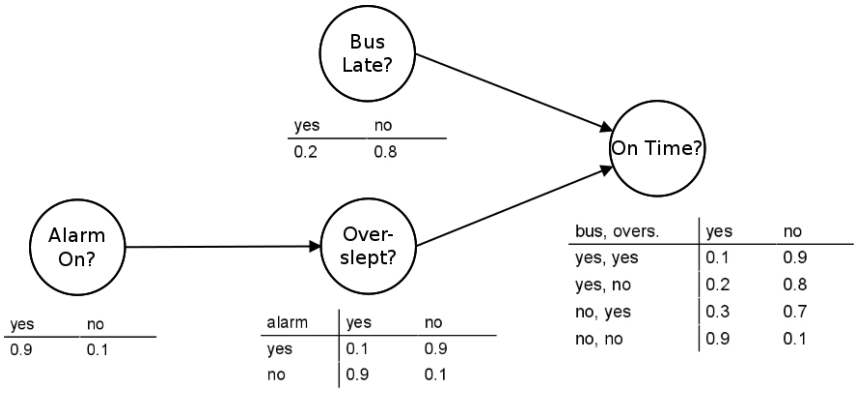

Figura 2.2: Ejemplo de Red Bayesiana. Imagen de [13]

2.1.4. Análisis Discriminante Lineal (LDA)

LDA es un clasificador que permite encontrar una combinación lineal de las características $\mathbf{X}$ capaz de separar los datos en las diferentes clases C. Para ello, cada clase es ajustada mediante una densidad gaussiana, asumiendo que todas las clases *comparten la matriz de covarianza*. A pesar de funcionar como

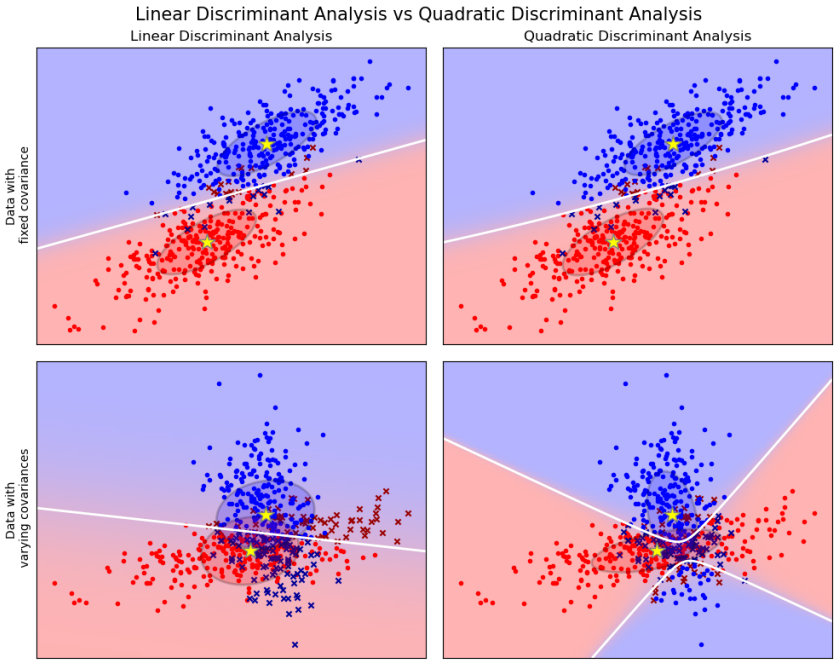

Figura 2.3: Diferencia entre LDA y QDA: LDA asume que las clases comparten una matriz de covarianza, lo que resulta en fronteras lineales. Por otro lado, QDA permite que cada clase tenga su propia matriz de covarianza, generando fronteras de decisión cuadráticas. Imagen de [14]

clasificador, también puede ser utilizado para reducir la dimensionalidad de los datos en otros problemas.

No obstante, como su propio nombre indica, la frontera de decisión que obtendremos con LDA es lineal, lo cual no se ajusta a la realidad de muchos casos. La razón de ello es que emplea la misma matriz de covarianza ($\Sigma_0 = \Sigma_1 = \Sigma$) para todas las clases, asumiendo que la varianza no cambia al variar C.

Para generar la frontera de decisión debemos ver dónde se igualan las probabilidades de pertenecer a ambas clases (desarrollo completo en [15]):

$$P(C_0|X) = P(C_1|\mathbf{X}) \tag{2.3}$$

Aplicando el Teorema de Bayes:

$$\frac{P(\mathbf{X}|C_0)P(C_0)}{P(\mathbf{X})} = \frac{P(\mathbf{X}|C_1)P(C_1)}{P(\mathbf{X})} \tag{2.4}$$

LDA asume distribuciones gaussianas, por lo que, ignorando el denominador, pasamos a la siguiente igualdad. Recordemos que ambas clases tienen la misma matriz de covarianza Σ:

$$\frac{1}{\sqrt{2\pi\,|\Sigma|}} \exp\left(-\frac{(\mathbf{X}-\mu_0)^\top \Sigma^{-1}(\mathbf{X}-\mu_0)}{2}\right)\pi_0 = \frac{1}{\sqrt{2\pi\,|\Sigma|}} \exp\left(-\frac{(\mathbf{X}-\mu_1)^\top \Sigma^{-1}(\mathbf{X}-\mu_1)}{2}\right)\pi_1, \tag{2.5}$$

Donde x es el dato, μ_i el vector de medias de la clase i, Σ_i la matriz de covarianza de la clase i y π_i la probabilidad a priori de la clase i. Simplificando, obtenemos lo siguiente:

$$2\left(\Sigma^{-1}(\mu_2-\mu_1)\right)^\top \mathbf{X} + (\mu_1-\mu_2)^\top\Sigma^{-1}(\mu_1-\mu_2) + 2\ln\left(\frac{\pi_2}{\pi_1}\right) = 0, \tag{2.6}$$

Que es la ecuación de una recta en la forma $\mathbf{a}^T\mathbf{X} + b = 0$, que es lineal, donde a es un vector de coeficientes y b es un término independiente.

2.1.5. Análisis Discriminante Cuadrático (QDA)

Al igual que LDA, QDA es un clasificador que genera una frontera de decisión para separar las clases. No obstante, mientras que la frontera de LDA es lineal, la de QDA es cuadrática. Esto se debe a que QDA alivia la restricción de homocedasticidad (es decir, no asume que las covarianzas de cada clase son iguales), lo que ofrece una frontera con términos cuadráticos.

Vamos a realizar el mismo análisis que con LDA para observarlo:

$$P(C_0|\mathbf{X}) = P(C_1|\mathbf{X}) \tag{2.7}$$

QDA también asume que ambas clases siguen distribuciones gaussianas, pero en este caso con diferentes varianzas ($\Sigma_0 \neq \Sigma_1$), por lo que la fórmula es más compleja:

$$\frac{1}{\sqrt{2\pi\,|\Sigma_0|}}\exp\left(-\frac{(\mathbf{X}-\mu_0)^\top\Sigma_0^{-1}(\mathbf{X}-\mu_0)}{2}\right)\pi_0 = \frac{1}{\sqrt{2\pi\,|\Sigma_1|}}\exp\left(-\frac{(\mathbf{X}-\mu_1)^\top\Sigma_1^{-1}(\mathbf{X}-\mu_1)}{2}\right)\pi_1, \tag{2.8}$$

Simplificando y eliminando términos, podemos llegar a la siguiente fórmula:

$$\mathbf{X}^\top(\Sigma_1-\Sigma_2)^{-1}\mathbf{X} + 2\left(\Sigma_2^{-1}\mu_2-\Sigma_1^{-1}\mu_1\right)^\top\mathbf{X} + (\mu_1^\top\Sigma_1^{-1}\mu_1-\mu_2^\top\Sigma_2^{-1}\mu_2) + \ln\left(\frac{|\Sigma_1|}{|\Sigma_2|}\right) + 2\ln\left(\frac{\pi_2}{\pi_1}\right) = 0,$$

Que es una función cuadrática de la forma $\mathbf{X}^T\mathbf{a}\mathbf{X}+\mathbf{b}\mathbf{X}+c=0$. La frontera cuadrática permite más flexibilidad a la hora de separar las clases; sin embargo, para nuestro objetivo de interpretabilidad, una frontera cuadrática complica la comprensión del modelo. Las interacciones cuadráticas entre variables dificultan la explicación directa del impacto de cada característica sobre la predicción.

De esta manera, QDA puede resolver problemas más complejos, siendo más efectivos en aquellos donde las clases tengan varianzas distintas. No obstante, en otros problemas más simples este método puede sufrir de sobreajuste, ya que introduce mayor flexibilidad a costa de una mayor varianza en la estimación.

2.2. Generalized Additive Models (GAM)

Antes de abordar los modelos aditivos generalizados (GAM), es fundamental introducir los modelos previos de los que derivan para comprender su evolución.

En primer lugar, debemos hablar de los **modelos lineales**. El modelo de regresión lineal clásica ha sido la base del análisis estadístico desde hace décadas, y establece una relación entre una variable respuesta Y y un conjunto $\mathbf{X} = \{X_1, ..., X_n\}$ de predictores:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n \tag{2.9}$$

Destaca por su simplicidad y su más que evidente interpretación, pues cada variable está acompañada de un coeficiente β_i. Sin embargo, impone dos restricciones fundamentales: asume una relación estrictamente lineal entre $\mathbf{X}$ e Y y considera que Y sigue una distribución normal, lo que rara vez se cumple en datos reales.

Como evolución a los modelos lineales aparecen los **modelos lineales generalizados (GLM)**. Los GLM amplían la regresión lineal clásica permitiendo que la variable respuesta pueda seguir otra distribución además de la normal. Su estructura es la siguiente:

$$g(E[Y]) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n \tag{2.10}$$

Donde se introduce una función g, denominada función de enlace que transforma el valor esperado de Y de manera que la relación entre este valor transformado y las variables predictoras sea lineal. Esto permite que los GLM sean aplicables a situaciones en las que la variable respuesta sigue distribuciones como la binomial, Poisson, gamma, entre otras, además de la distribución normal.

Finalmente, llegamos a los **modelos aditivos generalizados (GAM)**, que representan una extensión de los modelos lineales generalizados (GLM) y ofrecen mayor flexibilidad al permitir capturar patrones más complejos sin perder interpretabilidad. Para ello, reemplaza los términos lineales $\beta_i X_i$ por funciones $f_i(X_i)$, permitiendo modelar relaciones no lineales de manera automática.

$$g(E[Y]) = \beta_0 + f_1(X_1) + f_2(X_2) + \cdots + f_n(X_n) = \beta_0 + \sum_{i=1}^{n} f_i(X_i) \tag{2.11}$$

Este planteamiento resuelve el problema de la no linealidad al no asumir una relación lineal entre las variables predictoras y la respuesta, lo que permite adaptarse mejor a patrones complejos en los datos.

Pero, ¿qué tipo de funciones se utilizan para modelar estas relaciones no lineales y cómo se calculan? A continuación, se explorarán diferentes tipos de funciones empleadas en los GAM, así como los métodos para su estimación.

2.2.1. Explainable Boosting Machine

La Explainable Boosting Machine (EBM) es un modelo transparente diseñado para tener una precisión cercana a los modelos que conforman el estado del arte en conjuntos de datos tabulares. La primera implementación de esta arquitectura fue presentada por la librería con la que estamos trabajando, *InterpretML*.

En este modelo, las funciones $f(X_i)$ se calculan utilizando árboles de decisión en un proceso de bagging y boosting, diseñado para garantizar tanto precisión como interpretabilidad. El entrenamiento se realiza de manera iterativa en un esquema round-robin, donde el modelo ajusta una característica a la vez, sin considerar las demás en ese momento. Este enfoque evita que el orden de las variables afecte el aprendizaje y mitiga problemas de colinealidad.

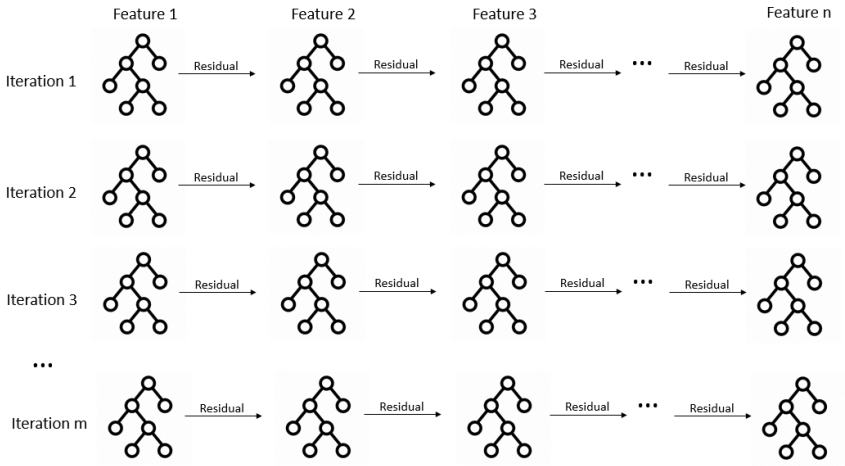

Figura 2.4: Entrenamiento de un EBM. Imagen de [16]

Además, el modelo EBM es capaz de detectar e incluir términos de interacción entre pares de variables [17], de manera que añaden un término a la fórmula anterior:

$$g(E[Y]) = \beta_0 + \sum_{i=1}^{n} f_i(X_i) + \sum_{(i,j)\in\mathcal{I}} f_{ij}(X_i, X_j) \tag{2.12}$$

Donde $\mathcal{I}$ se corresponde al conjunto de interacciones detectadas por el modelo.

2.2.2. Neural Additive Models

Es bien conocido que, a pesar de su increíble poder predictivo en numerosos problemas, incluso más allá de conjuntos de datos estructurados, las redes neuronales se caracterizan por su incapacidad para ser interpretadas. Es un modelo que agruparíamos en el conjunto de modelos "caja negra" (*Blackbox*). No obstante, las Neural Additive Models (NAM) permiten integrar el gran rendimiento de las redes neuronales con la interpretabilidad que ofrecen las GAM.

Durante el entrenamiento, las NAM aprenden una combinación lineal de redes neuronales donde cada una atiende individualmente a una característica concreta; es decir, cada función $f(X_i)$ se modela como una red neuronal. Además, utilizan una función de activación especial: ExU (*exp-centered*). Específicamente, para una entrada escalar x, cada unidad oculta con una función de activación f calcula $h(x)$ como:

$$h(x) = f\left(e^{w} \cdot (x - b)\right) \tag{2.13}$$

donde w y b son los parámetros de peso y sesgo, respectivamente.

La razón de usar esta función de activación y no otras más comunes, como ReLU, es que en muchos casos del mundo real las funciones que queremos modelar pueden presentar saltos abruptos. Las redes neuronales tradicionales tienden a suavizar en exceso estos cambios, perdiendo detalles importantes, mientras que ExU permite capturar mejor dichas discontinuidades sin comprometer el ajuste global al modelar los pesos en el espacio logarítmico y desplazar las entradas mediante un sesgo.

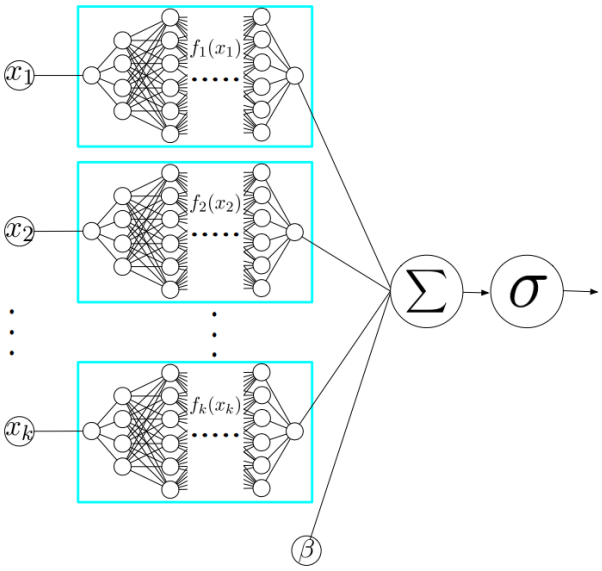

Figura 2.5: Arquitectura NAM. Imagen de [18]

Como se observa, cada variable es modelada mediante una red neuronal, por lo que la interpretación de una NAM es evidente: el impacto de una característica en una predicción es independiente de las demás características y se puede visualizar graficando la función $f(X_i)$ frente a X_i.

2.3. Otros Modelos

2.3.1. Random Forest

El algoritmo Random Forest se basa en el uso de múltiples Árboles de Decisión, lo que permite reducir la varianza del modelo original. Se trata de un modelo *en-*

semble basado en *bagging*, lo cual significa que entrena múltiples árboles sobre distintos subconjuntos de datos, y luego combina sus predicciones, normalmente mediante votación mayoritaria (en clasificación), para obtener un modelo más robusto y generalizable.

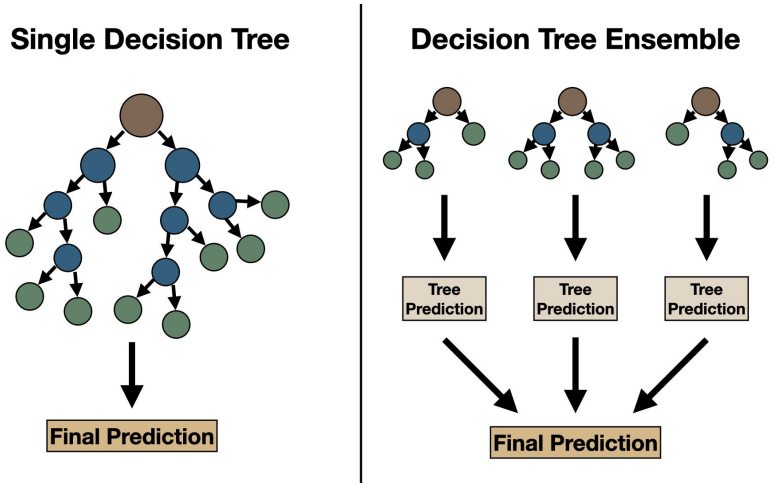

Figura 2.6: Diferencia entre Árbol de Decisión y Random Forest. Imagen de [19]

No obstante, a pesar de que una gran ventaja de los Árboles de Decisión era su interpretabilidad, no ocurre lo mismo con Random Forest. Puede parecer incoherente, ya que se basa en modelos interpretables; de hecho, uno podría interpretar los Árboles de Decisión que conforman el modelo individualmente. El problema viene cuando queremos entender como el modelo, colectivamente, ha tomado una decisión, especialmente cuando hablamos de Random Forest conformados por cientos de Árboles de Decisión.

La falta de interpretabilidad de Random Forest no impide que sea uno de los modelos más efectivos y con mejor rendimiento en conjuntos de datos tabulares. Su robustez frente al sobreajuste y su habilidad para capturar relaciones no lineales complejas lo convierten en una de las opciones más usadas en tareas de clasificación y regresión.

2.3.2. Gradient Boosting

La familia de algoritmos de Gradient Boosting se basan, también, en el uso de múltiples Árboles de Decisión. No obstante, ahora el enfoque es diferente, pues estamos hablando de un modelo *ensemble* de tipo *boosting*. En este caso, los árboles se entrenan de manera secuencial, donde cada nuevo árbol intenta corregir los errores cometidos por los árboles anteriores. El proceso se realiza de forma iterativa, ponderando más los casos donde los árboles previos cometieron errores, lo que lleva a una optimización del modelo en función de las predicciones anteriores.

De nuevo estamos ante un modelo *ensemble* de Árboles de Decisión y, por tanto, su interpretabilidad es altamente complicada. No obstante, se han hecho múltiples esfuerzos por intentar interpretar estos modelos, con técnicas como SHAP

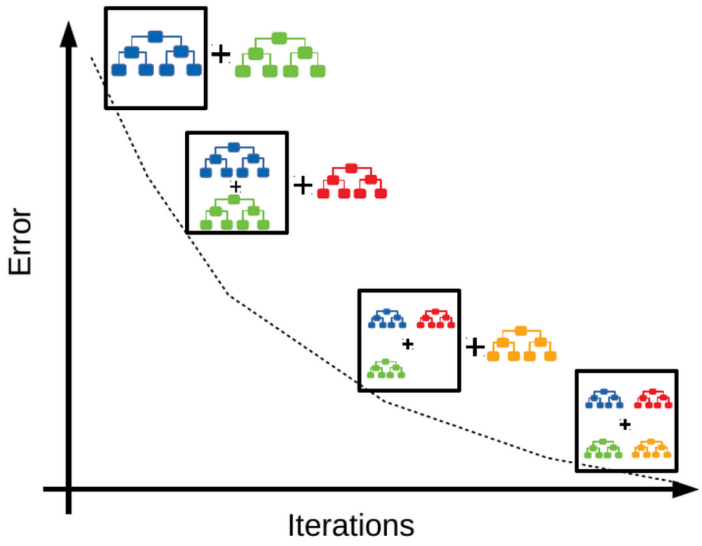

Figura 2.7: Correción de errores en Gradient Boosting. Imagen de [20]

(SHapley Additive exPlanations) o LIME (Local Interpretable Model-Agnostic Explanations), pero en la práctica enfrentan limitaciones y pueden ser sesgadas.

A partir de este enfoque general han surgido varias implementaciones específicas altamente optimizadas, entre las que destacan XGBoost, LightGBM y CatBoost.

XGBoost

XGBoost (*eXtreme Gradient Boosting*) es una de las implementaciones más conocidas y utilizadas de Gradient Boosting. Fue diseñado con el objetivo de mejorar la eficiencia y la precisión de los modelos tradicionales de boosting, y se caracteriza por incorporar una función de regularización explícita durante el entrenamiento, lo que ayuda a reducir el riesgo de sobreajuste. Además, XGBoost introduce técnicas avanzadas como el *pruning* (poda) de árboles y una gestión eficiente de valores nulos.

Uno de sus puntos fuertes es la escalabilidad: XGBoost puede manejar grandes volúmenes de datos gracias a su capacidad para paralelizar el entrenamiento de árboles y a su uso de técnicas de compresión de memoria. Sin embargo, su interpretabilidad sigue siendo limitada, al igual que en otros modelos de tipo ensemble.

LightGBM

LightGBM (*Light Gradient Boosting Machine*) es una implementación desarrollada por Microsoft que, al igual que XGBoost, busca mejorar el rendimiento y la eficiencia computacional del Gradient Boosting tradicional. Su principal aportación es el uso del algoritmo de *histogram-based learning*, que agrupa los valores continuos en bins discretos, reduciendo así el coste computacional sin perder demasiada precisión.

Otra característica diferencial de LightGBM es su estrategia de crecimiento de árboles por hojas (*leaf-wise*), como se observa en la figura 2.8, en lugar del crecimiento nivel por nivel que emplean otras variantes. Esto permite reducir aún más el error de entrenamiento, aunque puede llevar a un mayor riesgo de sobreajuste si no se controla adecuadamente.

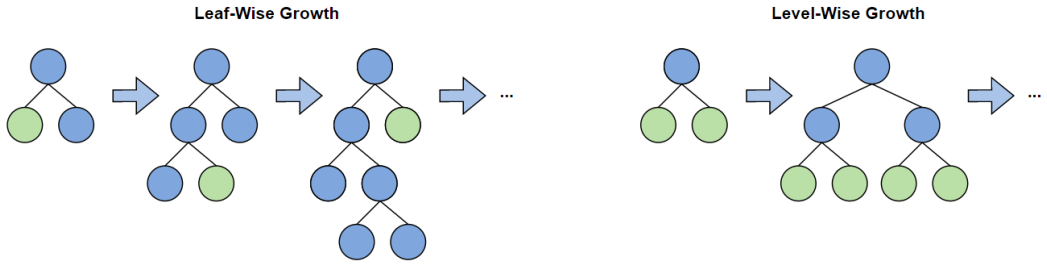

Figura 2.8: Visualización de *leaf-wise growth*, característica propia de LightGBM. Imagen de [21]

CatBoost

CatBoost, presentado hace apenas pocos años [22], introduce mejoras significativas en el tratamiento de variables categóricas, uno de los puntos débiles de muchas implementaciones de Gradient Boosting. En lugar de aplicar codificaciones previas como *one-hot* o *label encoding*, CatBoost maneja internamente estas variables mediante técnicas estadísticas que preservan la información del conjunto de entrenamiento sin introducir *leakage*.

Además, CatBoost emplea un esquema de entrenamiento ordenado que mitiga el *prediction shift*, un problema común en boosting secuencial. Esto lo convierte en un modelo robusto y fiable, incluso con configuraciones por defecto. Es una de las opciones más avanzadas y efectivas dentro de la familia de algoritmos de Gradient Boosting, y, en general, de problemas que involucran conjuntos de datos tabulares.

2.4. ¿Cómo se interpreta un modelo?

Antes de comenzar, debemos tener claro cómo se puede interpretar un modelo. InterpretML emplea dos métodos para este propósito:

- **Explicación Global**: Proporcionan información sobre cómo el modelo toma decisiones a nivel general, analizando su comportamiento en todo el conjunto de datos. Por ejemplo, dado un método GAM, que sigue la forma:

$$g(E[Y]) = \beta_0 + f_1(X_1) + f_2(X_2) + \cdots + f_n(X_n) \tag{2.14}$$

  La explicación global consistiría en visualizar las funciones f_i en el eje Y para un rango de valores concreto en el eje X.

Por ejemplo, supongamos que tenemos un modelo capaz de clasificar a las personas según sus ingresos en *bajos ingresos* (clase negativa) y *altos ingresos* (clase positiva), basándose en características como la edad. La explicación global implicaría observar cómo varía la contribución (o 'score') de las variables, como la edad, al cambiar de valor:

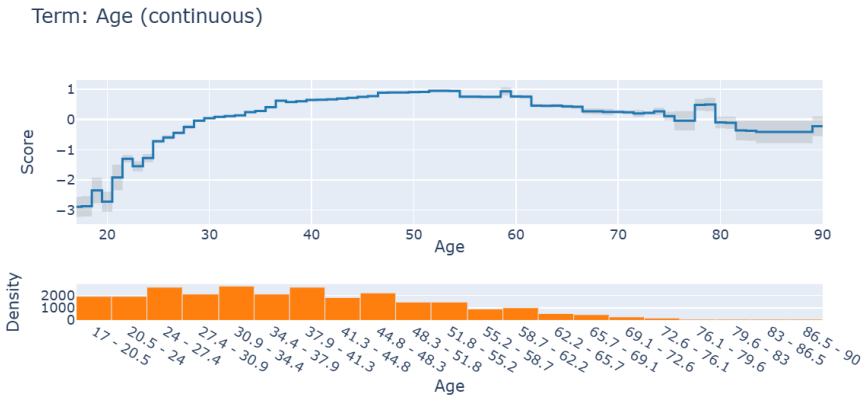

Figura 2.9: Evolución del 'score' de la variable Age. Imagen de [23]

Podemos observar varias tendencias:

- La contribución es muy baja para los más jóvenes, lo que sugiere que estos tienen menos probabilidad de pertenecer a la clase positiva del modelo.
- Desde los 20 hasta aproximadamente los 50 años, la puntuación aumenta gradualmente, indicando que la probabilidad de pertenecer a la clase positiva crece con la edad.
- Luego de los 50-55 años, la puntuación se estabiliza y comienza a decrecer ligeramente a partir de los 60-70 años. Esto sugiere que la relación entre edad e ingresos cambia en esa franja de edad.

- **Explicación Local**: Se centran en predicciones individuales, mostrando las razones específicas detrás de la decisión del modelo para un caso particular. Por tanto, a través de esta explicación podríamos observar qué influencia ha tenido cada variable en cada predicción.

  En este caso se puede observar cómo el estado conyugal (*MaritalStatus*) y el género (*Gender*) han sido factores clave para predecir la clase negativa, a pesar de que su educación (*Education*) tenía un impacto notable en la probabilidad de pertenecer a la clase positiva. Es interesante observar cómo en este caso la edad (*Age*), 30 años, no ha contribuido apenas a la predicción. Conectándolo con el anterior gráfico, podemos ver que cuando $Age = 30$ el 'score' es cercano a 0.

La idea es aplicar estos dos métodos a cada uno de los modelos que queremos

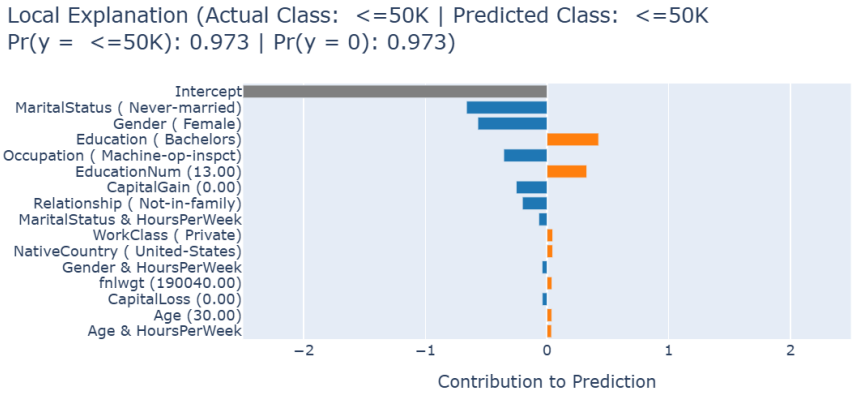

Figura 2.10: Contribución de cada variable sobre un cierto dato. Imagen de [23]

integrar, generando interpretaciones más completas y comparables, facilitando la comprensión del impacto de cada variable en diferentes enfoques de modelado y mejorando la transparencia en la toma de decisiones.

Capítulo 3

Desarrollo

El desarrollo del proyecto consistirá en tres fases:

1. **Integración de Modelos Interpretables**: Durante esta fase explicaremos los conceptos técnicos necesarios para entender cómo puede ser interpretado cada modelo, apoyándonos en el marco teórico previo. Esto nos permitirá integrarlos dentro de las herramientas proporcionadas por InterpretML, facilitando su uso en la evaluación y explicación de modelos de aprendizaje automático.
2. **Extensión de InterpretML**: Se ha desarrollado y publicado un paquete en PyPI que extiende las funcionalidades de InterpretML a modelos probabilísticos, como Naive Bayes o modelos generativos, permitiendo representar gráficamente los modelos de forma interpretable.
3. **Comparación de Modelos**: Finalmente, una vez implementemos los modelos mencionados, se realizará una comparación sistemática de los mismos en cuanto a su capacidad predictiva y su interpretabilidad.

3.1. Integración de Modelos

A continuación, se procede a la integración de los modelos previamente expuestos, detallando las técnicas empleadas para su interpretación y proporcionando comentarios adicionales.

3.1.1. Modelos Probabilísticos

3.1.1.1. Naive Bayes

El primero de los modelos integrados en el entorno de InterpretML ha sido Naive Bayes, utilizando la implementación disponible en la biblioteca *scikit-learn*. La asunción que tomaba Naive Bayes era que las características eran independientes dentro de cada clase, por lo que ya se puede intuir que la contribución de cada variable a la clasificación es aditivo. Esto facilita su interpretación, ya

que se pueden analizar los efectos individuales de cada característica de forma separada.

Recordemos que la fórmula de este modelo era la siguiente:

$$P(C \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid C)\ P(C)}{P(\mathbf{X})} = \frac{\prod_{i=1}^{n} P(X_i \mid C)\ P(C)}{P(\mathbf{X})} \tag{3.1}$$

En la propia fórmula se observa lo mencionado: La probabilidad de observar una característica concreta (X_i) dada la clase (C) tiene un efecto concreto y separado de las demás características. Pero, ¿cómo se puede cuantificar este efecto?

Para ello, nos apoyaremos en una técnica auxiliar. Consideremos una clasificación binaria con las clases **C** y **C'** y una función discriminante **d(X)** tal que:

$$f(\mathbf{X}) = \begin{cases} C & \text{si } d(\mathbf{X}) > 0 \\ \text{frontera de decisión} & \text{si } d(\mathbf{X}) = 0 \\ C' & \text{si } d(\mathbf{X}) < 0 \end{cases} \tag{3.2}$$

Nuestra función discriminante será la siguiente:

$$d(\mathbf{X}) = \ln\frac{P(C|\mathbf{X})}{P(C'|\mathbf{X})} = \ln\frac{P(C)}{P(C')} + \sum \ln\frac{P(X_i|C)}{P(X_i|C')} \tag{3.3}$$

Así, se puede observar fácilmente que la función (3.2) guarda una gran relación con (3.1), pues si $f(\mathbf{X}) = C => d(\mathbf{X}) > 0 => P(C|\mathbf{X}) > P(C'|\mathbf{X}) =>$ La probabilidad de que X pertenezca a la clase C es mayor a la probabilidad de que pertenezca a C'.

De esta manera, cuantificar el efecto de cada variable es trivial, pues basta con tomar:

- **Intercepto**: $\ln\frac{P(C)}{P(C')}$
- **Contribución individual de $\mathbf{X}_i$**: $\ln\frac{P(X_i|C)}{P(X_i|C')}$

Para la **explicación global** del modelo, sería suficiente con visualizar cada contribución individual en un conjunto de puntos distribuidos a lo largo del rango de la variable, lo que permitiría observar cómo varía el efecto de cada variable en función de su entrada, representando así la relación aprendida por el modelo como una función como la que podemos ver en la figura 2.9.

La **explicación local** de cada predicción realizada por Naive Bayes puede ser fácilmente interpretada con esta implementación dado que podemos separar los efectos de cada variable, ofreciendo una visualización como la que se observa en la figura 2.10. Las probabilidades $P(X_i|C)$ y $P(X_i|C')$, así como las probabilidades del intercepto, se pueden obtener directamente de los parámetros del modelo en *scikit learn*.

Por ejemplo, con el conjunto de datos IRIS binarizado, el resultado se puede observar en las figuras 3.1 y 3.2.

Figura 3.1: Contribuciones globales de cada variable de IRIS binarizado en un modelo Naive Bayes. De izquierda a derecha y de arriba a abajo: *Sepal Length*, *Sepal Width*, *Petal Length* y *Petal Width*. Cada gráfica se corresponde con la explicación global mencionada anteriormente, es decir, la fórmula de la contribución individual evaluada en el rango de valores de la variable. Por ejemplo, en el caso de *Sepal Width*, los valores bajos presentan una contribución negativa (menor que 0), lo que favorece la predicción de la clase 0. En cambio, los valores altos de *Sepal Width* muestran una contribución positiva, indicando una mayor probabilidad de que la observación pertenezca a la clase 1.

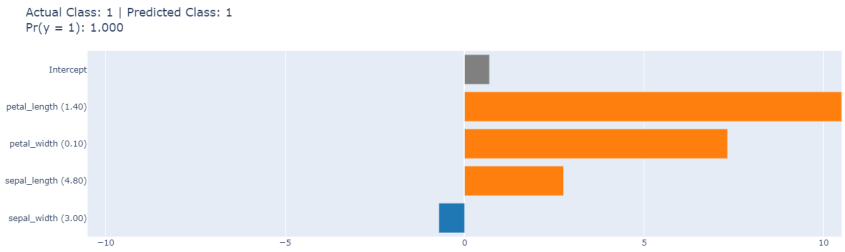

Figura 3.2: Explicación local de una instancia con los siguientes valores: Petal Length = 1.40, Petal Width = 0.10, Sepal Length = 4.80, Sepal Width = 3.80. Se observa como el modelo ha predecido la clase positiva, y además, qué variables han tenido más impacto. En este caso, Petal length y Petal Width han sido decisivas para tomar la decisión correcta. Por otra parte Sepal Width ha tenido una contribución negativa, indicativo de que podría ser un valor anómalo.

3.1.1.2. Tree Augmented Naive Bayes

Tras la ausencia de una implementación de TAN en la librería previamente utilizada (*scikit-learn*), se ha recurrido al uso del framework *pyAgrum*. No obstante, una vez adaptado Naive Bayes al entorno de InterpretML, la integración de TAN resulta muy similar, aunque introduce una diferencia clave. Como se mencionó en la sección anterior, TAN relaja la suposición de independencia condicional

presente en Naive Bayes, permitiendo que cada variable predictora tenga, como máximo, una dependencia adicional además de su relación con la variable objetivo.

Es decir, en este caso la contribución global de cada variable no se puede visualizar como una función donde en el eje X tenemos la variable y en el eje Y la contribución, pues cada variable puede verse afectada no solo por su relación directa con la variable objetivo, sino también por su dependencia con otra variable predictora. Para representar la contribución global de una variable en este contexto, es necesario utilizar un grid con todas las combinaciones posibles de la variable predictora y la otra variable de la que depende, generando un mapa de calor. Este enfoque nos permite observar cómo varía la contribución de una variable en función de sus valores y de los valores correspondientes de la variable con la que interactúa.

Por ejemplo, supongamos que queremos utilizar un Tree Augmented Naive Bayes para modelar las relaciones que existen en el conjunto de datos de Iris (adaptado para que solo existan dos clases) y poder hacer predicciones. En este caso, un posible árbol sería el siguiente:

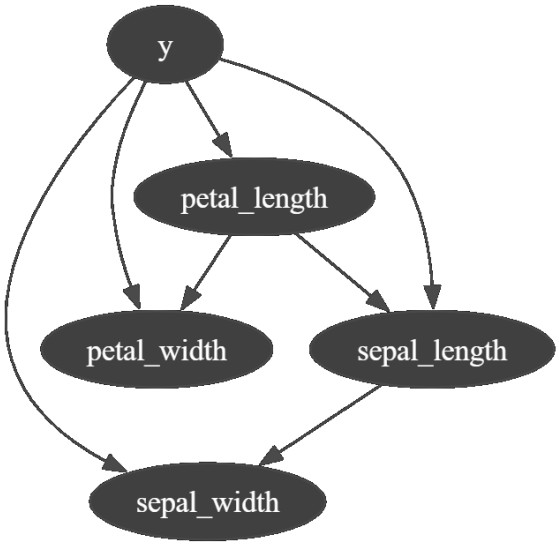

Figura 3.3: Ejemplo de árbol generado por TAN en el conjunto de datos de Iris

Donde observamos que *petal_length* es la única variable que solo depende de la variable objetivo (la especie de flor). Las demás, aparte de depender de la variable objetivo, tienen otro 'padre': *petal_width* y *sepal_length* dependen de *petal_length* y *sepal_width* de *sepal_length*.

Así, la contribución global de *petal_length* se puede visualizar exactamente de la misma manera que en Naive Bayes, mientras que para las demás deberíamos utilizar el método ya mencionado. Por ejemplo, para *sepal_width*:

Para entender cómo se han generado las puntuaciones de contribución, volvamos a la función discriminante que empleamos en Naive Bayes. En este caso,

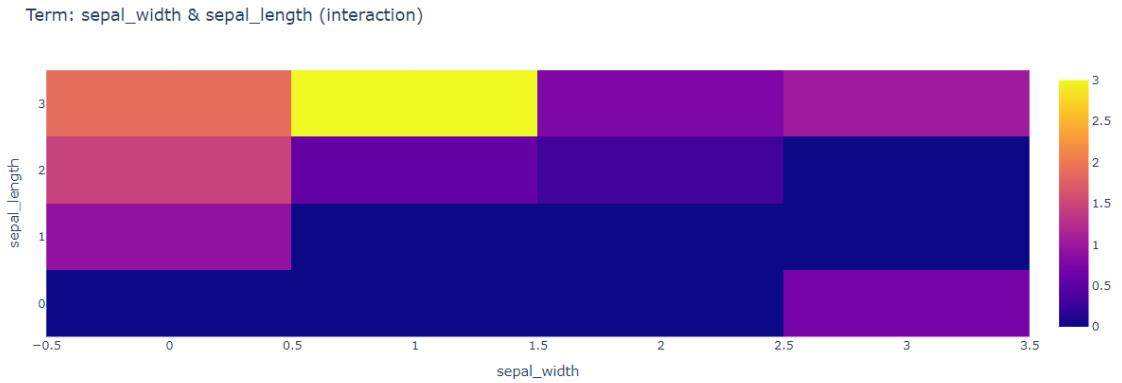

Figura 3.4: Contribución global de *Sepal Width* visualizada como un mapa de calor. En este caso, por ejemplo, valores altos de *Sepal Length* junto a valores bajos de *Sepal Width* proporcionan contribuciones muy positivas, favoreciendo la clase positiva. En cambio, valores bajos de *Sepal Length* y altos de *Sepal Width* favorecen la clase negativa. Estos valores se corresponden con la fórmula que se puede ver más abajo, **contribución de $\mathbf{X}_i$**

como decíamos, se admiten otras dependencias, pero la estrategia será la misma.

$$d(X) = \ln\frac{P(C|\mathbf{X})}{P(C'|\mathbf{X})} = \ln\frac{P(C)}{P(C')} + \sum_i \ln\frac{P(X_i|X_{\pi(i)}, C)}{P(X_i|X_{\pi(i)}, C')} \tag{3.4}$$

Donde $\pi(i)$ representa al padre de la variable i, si existe. Así:

- **Intercepto**: $\ln\frac{P(C)}{P(C')}$
- **Contribución de $\mathbf{X}_i$**: $\ln\frac{P(X_i|X_{\pi(i)}, C)}{P(X_i|X_{\pi(i)}, C')}$

De esta manera, la **explicación global** de cada variable X_i se visualizará de dos posibles maneras: o bien como una función que representa la relación entre la variable y su efecto en el caso de variables que no tengan dependencias más allá de la variable objetivo (como en el caso de Naive Bayes), o bien utilizando un grid con todas las combinaciones posibles de la variable predictora y la otra variable de la que depende, como ya se ha explicado.

Para la **explicación local** basta con acceder a la función/grid de la explicación global y seleccionar el efecto correspondiente al valor de la variable (o de las variables, en el caso de que dependa de otra). Estos valores se pueden obtener de la Distribución de Probabilidad Conjunta del modelo de pyAgrum.

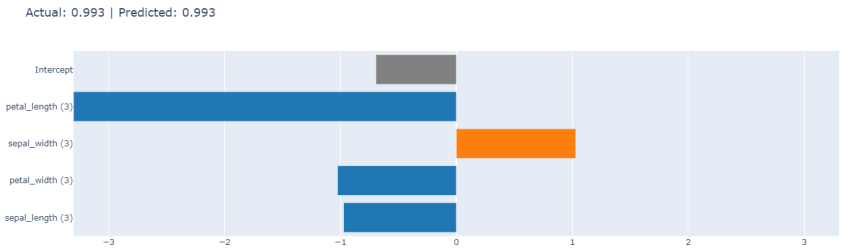

Figura 3.5: Explicación local para una instancia del conjunto de datos Iris binarizado y discretizado de un modelo TAN. En este caso, todas las variables exceptuando *Petal Length* han tenido una contribución individual negativa, favoreciendo, por tanto, a la predicción de la clase negativa para esa instancia.

3.1.2.1. Linear Discriminant Analysis (LDA)

En este caso sí que contamos con la implementación de LDA en *scikit-learn*, lo cual facilita su integración. Como comentábamos en la anterior sección, LDA genera una frontera de decisión lineal que separa ambas clases, representada por la siguiente fórmula (véase su desarrollo en 2.1.4):

$$2\left(\Sigma^{-1}(\mu_2-\mu_1)\right)^\top \mathbf{X} + (\mu_1-\mu_2)^\top\Sigma^{-1}(\mu_1-\mu_2) + 2\ln\left(\frac{\pi_2}{\pi_1}\right) = 0, \tag{3.5}$$

Que es la ecuación de una recta en la forma $\mathbf{a}^T\mathbf{X} + b = 0$, claramente lineal. De esta manera, podemos obtener los coeficientes (**a**) que acompañan a cada variable, haciendo que la interpretación del modelo sea directa. Así:

- **Intercepto**: $(\mu_1-\mu_2)^\top\Sigma^{-1}(\mu_1-\mu_2) + 2\ln\left(\frac{\pi_2}{\pi_1}\right)$
- **Contribución de $\mathbf{X}_i$**: $2\left(\Sigma^{-1}(\mu_2-\mu_1)\right)^\top X_i$

Así, la **explicación global** de cada variable se puede representar como la siguiente función: $f(X_i) = 2\left(\Sigma^{-1}(\mu_2-\mu_1)\right)^\top X_i$, que podemos observar que es una función lineal en X_i, ya que el término que acompaña a la X_i es constante. El valor de ese término (podemos denominarlos *coeficientes*) forma parte de los parámetros del modelo de *scikit-learn*.

Por otro lado, la **explicación local** se obtendría realizando la multiplicación del coeficiente por el valor de X_i, obteniendo así una manera de ver qué variables han tenido un mayor efecto en cada predicción.

3.1.2. Generalized Additive Models (GAM)

3.1.3.1. Neural Additive Models (NAM)

Los modelos NAM pertenecen al grupo de modelos aditivos, por lo que su interpretabilidad está asegurada. Como ya se comentó en la sección anterior, estos modelos combinan la transparencia de los modelos GAM con la flexibilidad de

Figura 3.6: Contribuciones globales de cada variable de IRIS binarizado en un modelo LDA. De izquierda a derecha y de arriba a abajo: *Sepal Length*, *Sepal Width*, *Petal Length* y *Petal Width*. Cada gráfica se corresponde con la explicación global mencionada anteriormente, es decir, la fórmula de la contribución individual evaluada en el rango de valores de la variable. Por ejemplo, en el caso de *Petal Width*, los valores bajos presentan una contribución cercana a 0, mientras que valores altos tienen una contribución muy negativa, favoreciendo a la clase 0.

las redes neuronales. La idea es que cada variable de entrada se modela independientemente con una red neuronal, aprendiendo una función no lineal específica para esta variable. La predicción final resultará de la suma de las salidas de estas redes. Esta estructura modular permite visualizar el efecto individual de cada variable sobre la predicción.

Para la integración de los modelos NAM en este proyecto, se analizaron distintas implementaciones públicas, todas ellas inspiradas en el trabajo original de Agarwal et al. (2021) [18], en el que se propone este nuevo modelo. Tras una revisión de las opciones existentes, se optó por utilizar la implementación disponible en el repositorio de GitHub: https://github.com/lemeln/nam. Este repositorio actúa como un wrapper diseñado con el propósito de integrar los NAM en el ecosistema de InterpretML. No obstante, es importante señalar que dicho repositorio no cuenta con mantenimiento, lo que implicó resolver manualmente algunas incompatibilidades y limitaciones para su correcto funcionamiento.

La propia implementación contenía un método para visualizar el efecto individual de cada variable, aunque se tuvieron que realizar diversas modificaciones para adaptarlo a la librería InterpretML. Además, cada red neuronal almacenaba también el bias, que finalmente era sumado para realizar la predicción final.

Así, la **explicación global** se puede obtener visualizando las funciones aprendidas por cada subred neuronal $f_i(X_i)$, como se observa en la figura 3.7, que representan el efecto no lineal de cada variable X_i sobre la predicción final. Estas funciones son directamente interpretables y permiten identificar tendencias, umbrales o saturaciones en el comportamiento del modelo respecto a cada variable.

Por su parte, la **explicación local** se obtiene evaluando $f_i(X_i)$ para un valor concreto de la variable en una observación determinada. De esta forma, es posible descomponer la predicción en las contribuciones individuales de cada variable, sumadas junto con el sesgo global del modelo, lo que facilita un análisis detallado de cada predicción.

Figura 3.7: Contribuciones globales de cada variable de IRIS binarizado en un modelo NAM. De izquierda a derecha y de arriba a abajo: *Sepal Length*, *Sepal Width*, *Petal Length* y *Petal Width*. Cada gráfica se corresponde con la explicación global, es decir, la fórmula de la contribución individual evaluada en el rango de valores de la variable. Las tendencias son muy similares a las que se vieron en las contribuciones globales de Naive Bayes. Las variables Sepal Length, Petal Length y Petal Width tienen una tendencia decreciente, por lo que a valores más altos, más se favorecerá la clase negativa. La tendencia es Sepal Width es la inversa.

3.2. Extensión de InterpretML

3.2.1. Objetivos y Motivación

Con el objetivo de ampliar las capacidades explicativas de la librería InterpretML, se ha desarrollado una extensión centrada en la integración de modelos probabilísticos y otros. Mientras que InterpretML proporciona soporte para modelos aditivos, árboles de decisión o lineales, esta ampliación permite incorporar modelos que manejan incertidumbre de forma explícita, lo que resulta especialmente útil en contextos donde no solo interesa la predicción puntual, sino también la confianza asociada a dicha predicción.

La motivación principal ha sido facilitar la interpretación de modelos probabilísticos desde una perspectiva coherente con el enfoque de InterpretML, reutilizando su infraestructura de explicación global y local. De este modo, se muestra un análisis más profundo del comportamiento de los modelos, permitiendo entender cómo se distribuyen las probabilidades entre clases, cómo influye cada variable en dichas distribuciones, y cómo varía la incertidumbre a nivel de observación.

3.2.2. Descripción y Publicación del Paquete

La extensión incorpora los modelos mencionados en la sección anterior, incluyendo tanto algoritmos clásicos como propuestas más recientes basadas en redes neuronales. Para cada uno de ellos se ha diseñado una clase específica que implementa las funciones *explain_global* y *explain_local*, asegurando su compatibilidad con las visualizaciones ya existentes en InterpretML.

- **explain_local**: Genera las explicaciones locales a nivel de instancia. Calcula, para cada ejemplo de entrada, la contribución individual de cada característica a la predicción del modelo. Por ejemplo, en Naive Bayes, se basa en el cociente logarítmico de las probabilidades condicionales por clase y añade un término de intercepto correspondiente al logaritmo de las probabilidades a priori.
- **explain_global**: Proporciona las explicaciones globales sobre el comportamiento del modelo. Analiza cómo varía la importancia de cada característica en función de su valor, generando el gráfico que representan la relación entre cada variable y la predicción, además de superponer la densidad de datos para facilitar su interpretación.

A continuación se presenta un resumen del código más relevante de cada función (para el caso de Naive Bayes), destacando las transformaciones clave aplicadas para obtener las explicaciones:

```
def explain_local(self, X, y=None, name=None):
    # ...
    def conditional_probabilities(model, X):
        class_0_cp = ...
        class_1_cp = ...
        return class_0_cp, class_1_cp

    class_0_prior = model.class_prior_[0]
    class_1_prior = model.class_prior_[1]
    intercept = np.log(class_0_prior / class_1_prior)
    for i, instance in enumerate(X):
        c0_cp, c1_cp = conditional_probabilities(model, instance)
        scores = np.log(c1_cp / c0_cp)
        data_dict["scores"] = scores
        data_dict["extra"] = {"scores": [intercept], "names": ["Intercept"]}
        # ...
```

Listing 3.1: Fragmentos clave de `explain_local`. Por una parte se calculan las probabilidades condicionales, que, como vimos, son la clave para calcular las contribuciones. Por otra parte se calcula el intercepto a partir de las probabilidades a priori.

```
def explain_global(self, name=None):
    def get_ratio(model, value, index):
        log_ratio = ...
        return log_ratio

    for index, _ in enumerate(self.feature_names_in_):
        grid_points = np.linspace(...)
        model_graph = [get_ratio(model, val, index) for val in grid_points]
        data_dict = {
            "names": grid_points,
            "scores": model_graph,
            "density": {
                "scores": self.bin_counts_[index],
                "names": self.bin_edges_[index],
            }
        }
```

Listing 3.2: Fragmentos clave de `explain_global`. En resumen, se utiliza la fórmula para obtener la contribución individual en un punto concreto (*get_ratio*), y se evalúa en el rango completo de la variable, obteniendo la contribución global.

Estas dos funciones, `explain_local` y `explain_global`, han sido las más relevantes en el proceso de creación de las nuevas clases, ya que constituyen el núcleo de la lógica interpretativa de cada modelo. En ellas ha sido necesario profundizar en el funcionamiento interno de cada algoritmo, identificar cómo calcula sus predicciones, con qué parámetros opera y de qué forma se puede traducir esa lógica en términos explicativos comprensibles. No obstante, para que estas clases fueran plenamente compatibles con la infraestructura de InterpretML, se ha llevado a cabo un estudio detallado de la arquitectura de la librería: explorando su estructura interna, jerarquías de clases, métodos auxiliares o formatos de salida esperados. Solo con este conocimiento ha sido posible diseñar una integración coherente, que mantenga la arquitectura original y extensible de la plataforma.

El paquete ha sido publicado en PyPI, como se puede observar en la figura 3.8, bajo el nombre interpret-extension, y está concebido como una herramienta abierta y extensible, pensada para fomentar el uso de modelos interpretables más allá de los que soporta InterpretML. El código fuente de la extensión se encuentra en el siguiente repositorio de Github: https://github.com/bmihaljevic/tfg-interpretml

3.3. Comparativa de Modelos

Una vez introducidos los modelos interpretables con los que vamos a trabajar, además de otros modelos caja negra, el objetivo será comparar el rendimiento de los mismos en diferentes conjuntos de datos. Para ello se ha estudiado el uso de varios benchmarks, de manera que los resultados se puedan reproducir y sean comparados de manera justa.

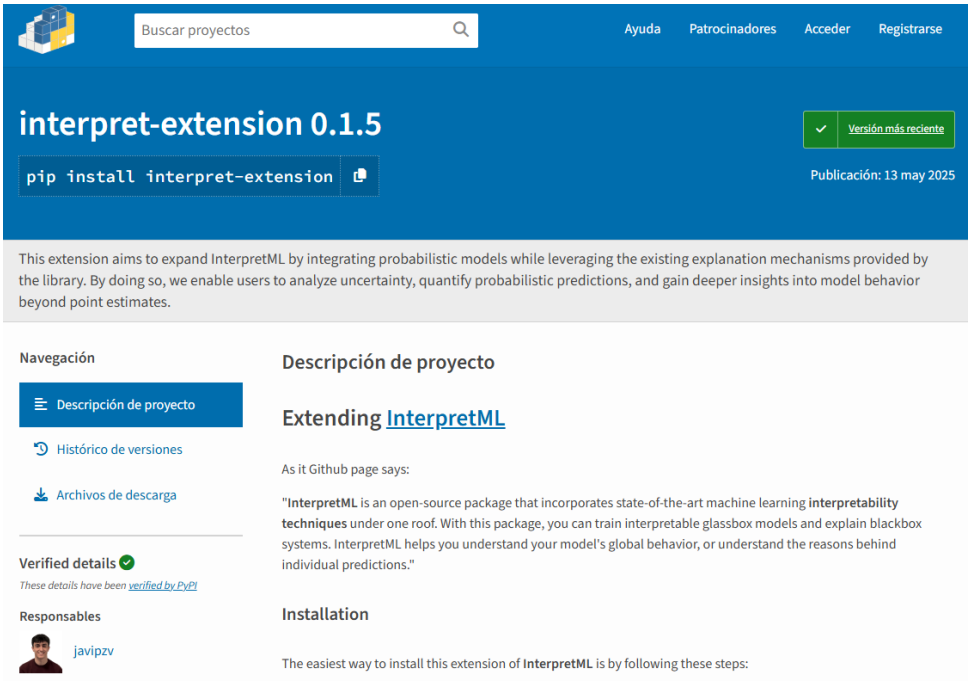

Figura 3.8: Paquete publicado en PyPI, https://pypi.org/project/interpret-extension

La elección ha sido el benchmark **TabZilla**, que, introducido en 2023 [10], permite evaluar modelos sobre más de 140 conjuntos de datos tabulares de características variadas, lo que facilita realizar una comparativa robusta y representativa de los modelos en diversos escenarios.

3.3.1. Benchmarking con TabZilla

Para llevar a cabo la comparativa de modelos se ha empleado el benchmark TabZilla. TabZilla es un framework que permite comparar un amplio número de algoritmos de aprendizaje automático sobre una extensa y diversa colección de conjuntos de datos tabulares. Además de posibilitar la comparación directa entre algoritmos, TabZilla también proporciona herramientas para analizar las propiedades de los conjuntos de datos (a través de metafeatures) y estudiar cómo dichas propiedades influyen en el rendimiento de los diferentes modelos.

Figura 3.9: TabZilla, benchmark utilizado para el experimento.

La implementación de TabZilla está disponible públicamente en el repositorio https://github.com/naszilla/tabzilla. El framework se estructura en torno a dos funcionalidades principales. Por un lado, permite la **ejecución de experimentos**, entendidos como la evaluación de un algoritmo sobre un conjunto de da-

tos específico, donde se realizan múltiples muestreos de hiperparámetros y se evalúan los resultados sobre distintas particiones de los datos. Por otro lado, proporciona herramientas para la **extracción de metafeatures**, descriptores numéricos que caracterizan cada conjunto de datos.

TabZilla trabaja principalmente con conjuntos de datos tabulares provenientes de OpenML, y facilita la incorporación de nuevos datasets o algoritmos mediante interfaces compatibles con scikit-learn. La integración de nuestros algoritmos en TabZilla ha sido sencilla, por lo que ha sido posible incluirlos en el proceso de evaluación y compararlos en igualdad de condiciones con otros métodos ya implementados en el benchmark.

3.3.2. Datasets elegidos

Los conjuntos de datos que proporciona TabZilla son de OpenML, una plataforma abierta diseñada para compartir conjuntos de datos, resultados experimentales u otras tareas de machine learning. Facilita el acceso a numerosos datasets etiquetados y de diferentes tipos, organizados para su uso en investigación. Dependiendo del objetivo de predicción, los conjuntos de datos disponibles son de diversos tipos. Nosotros escogeremos varios de los etiquetados como 'binary', pues nuestro objetivo es comparar el rendimiento de los modelos en tareas de clasificación binaria, que es el enfoque principal de este trabajo.

Como se ha mencionado previamente, TabZilla proporciona métodos para estudiar las características de los propios datasets. Esto es relevante, pues una vez obtengamos los resultados, podremos verificar si existe alguna *metafeature* (por ejemplo, la curtosis) que a menudo beneficie a los modelos interpretables. Por ejemplo, Grinsztajn et al. [9] observaron que los GBDTs, como XGBoost y CatBoost, suelen desempeñarse mejor en datasets grandes y con características irregulares, mientras que las redes neuronales destacan en conjuntos más pequeños o con relaciones más suaves y aditivas.

Es evidente que los modelos interpretables se benefician especialmente de conjuntos de datos donde la relación entre las variables predictoras y la variable objetivo sea suave y aditiva. Por ejemplo, Levy et al. [24] observaron que, en experimentos simulados donde el modelo verdadero era puramente aditivo, los modelos aditivos obtuvieron una mayor precisión. No obstante, a medida que aumentaba la fuerza de las interacciones, los métodos ensemble (como Random Forest) los superaban en rendimiento.

A pesar de estas limitaciones, parece existir cierta evidencia de que muchas relaciones en datos reales pueden aproximarse mediante funciones aditivas. En una comparativa con diez conjuntos de datos reales [25] (salud, finanzas, etc.), modelos basados en GAM obtuvieron AUCs casi idénticas a las de los mejores modelos de conjunto, con una diferencia media inferior al 0.2 %, lo que sugiere que en muchos problemas reales la suposición de aditividad es razonable.

Sin embargo, esta característica no siempre es evidente de antemano, por lo que nuestro enfoque se centra en explorar otros tipos de metafeatures. Este es un tema relevante que, hasta donde sabemos, no ha sido investigado en profundidad,

por lo que podría dar lugar a nuevas perspectivas.

Algunos autores han mencionado ciertas fortalezas y fragilidades de los GAMs y modelos aditivos. Por ejemplo, Zhang et al. [26] observaron que Naive Bayes es altamente robusto al ruido en los datos y en conjuntos de datos con un ratio señal/ruido muy bajo (es decir, cuando la señal es débil o está oculta entre mucho ruido), este modelo funcionaba mejor que otros más complejos. Por otro lado, Chang et al. [25] mostraron que ciertos artefactos presentes en los datos, como anomalías locales o umbrales muy marcados tienden a pasar desapercibidos para los GAMs basados en splines debido a su suavidad. En cambio, los GAMs basados en árboles (como EBM) sí logran capturar este tipo de patrones abruptos. En la práctica, esto implica que conjuntos de datos con discontinuidades bruscas (una metafeature similar a tener variables muy sesgadas o binarizadas) pueden perjudicar el rendimiento de los GAMs suaves.

Para la elección de los conjuntos de datos de nuestro experimento, hemos decidido basarnos en el artículo que presenta el benchmark TabZilla [10], seleccionando varios de los datasets que ellos han considerado difíciles para la gran mayoría de modelos. Concretamente, la definición que utilizan es la siguiente:

*"**Hard for most algorithms**. This criterion is designed to include datasets on which most algorithms were not able to reach top performance. In particular, a dataset satisfies the criterion if the fourth-best log loss out of 19 algorithms is at least 7% worse than the top log loss. In other words, this criterion will include datasets on which one, two, or three algorithms were able to stand out in terms of performance. [...]"*

De entre los datasets que se consideran dentro de este criterio, hemos seleccionado 15. En la tabla X se especifican cuáles son, además de algunas metafeatures obtenidas gracias a TabZilla.

Dataset	Inst	Feat	Skewness	Kurtosis
ada_agnostic	4562	48	NaN	NaN
colic	368	26	0.285	-1.173
credit-approval	690	15	-0.529	-0.979
credit-g	1000	20	1.368	1.624
electricity	45312	8	-0.404	-0.579
elevators	16599	18	-0.428	-0.773
heart-h	294	13	0.126	-0.931
jasmine	2984	144	-0.245	-1.322
kc1	2109	21	NaN	NaN
nomao	34465	118	0.381	-0.477
phoneme	5404	5	0.059	-0.803
profb	672	9	NaN	NaN
qsar-biodeg	1055	41	0.148	-1.175
socmob	1156	5	-0.361	-1.147
SpeedDating	8378	120	NaN	NaN

Cuadro 3.1: Datasets utilizados en el experimento

Podemos observar que los 15 datasets varían significativamente en número de instancias y número de características. Esto permitirá evaluar el rendimiento de

los modelos en escenarios con distintas complejidades y tamaños, desde datasets pequeños y medianos hasta otros con decenas de miles de instancias y un amplio rango de atributos. Además, se incluyen estadísticas descriptivas básicas (skewness y kurtosis).

3.3.3. Metodología de Evaluación

Para garantizar una evaluación robusta y justa de los modelos de aprendizaje automático, se ha seguido la metodología utilizada por el framework *TabZilla*. Cada uno de los 15 conjunto de datos ha sido dividido en 10 particiones (folds) para aplicar *cross-validation*, permitiendo así una estimación fiable del rendimiento de los modelos independientemente de su partición.

Además, con el fin de estudiar la variabilidad en el rendimiento de los modelos frente a diferentes configuraciones, cada algoritmo se ha ejecutado **5 veces por conjunto de datos** utilizando combinaciones aleatorias de hiperparámetros. A esto se suma una ejecución adicional con optimización de hiperparámetros, llevada a cabo mediante la biblioteca *Optuna*, con el objetivo de explorar automáticamente el espacio de búsqueda y encontrar configuraciones óptimas.

Durante cada experimento se registraron métricas de rendimiento tales como *accuracy* o *Log Loss*, entre otras, así como el tiempo de entrenamiento. Estas variables serán utilizadas posteriormente para realizar comparaciones cuantitativas entre los modelos evaluados.

3.3.4. Resultados

A continuación se presentan los resultados obtenidos en la comparativa de modelos. Se han evaluado diferentes modelos (explicables y no explicables) sobre los 15 datasets mencionados anteriormente y en las condiciones que se han explicado en la sección previa.

Comparativa de Rendimiento

En la tabla 3.2 podemos observar los resultados que se han obtenido por cada modelo. Las columnas relativas al rendimiento se han obtenido haciendo una media de las 5+1 ejecuciones de cada modelo y dataset. Se ha decidido utilizar el *Accuracy* y *Log Loss*, dos métricas fundamentales a la hora de evaluar la eficacia de un modelo de clasificación. Mientras que la *Accuracy* indica el porcentaje de aciertos del modelo sobre los datos de test, el *Log Loss* proporciona una medida más sensible que penaliza más las predicciones incorrectas y con poca confianza.

En la tabla podemos ver como **CatBoost** se posiciona como el modelo más destacado, obteniendo tanto la mejor precisión (0.856) como el menor log loss (0.330), y ocupando el primer lugar en ambos rankings, por lo que no solo es un modelo preciso, sino también confiable a la hora de estimar probabilidades.

Le sigue muy de cerca **XGBoost**, con una precisión de 0.854 y un log loss de 0.340, ocupando la segunda y tercera posición respectivamente. Ambos mode-

Modelo	Test Accuracy	Test Log Loss	Accuracy Rank	Log Loss Rank
CatBoost	0.856	0.330	**1**	**1**
EBM	0.849	0.339	3	2
GaussianNB	0.751	2.152	10	10
LDA	0.816	0.415	6	5
LightGBM	0.833	0.642	4	9
LogisticReg	0.799	0.445	7	6
NAM	0.793	0.460	9	8
RandomForest	0.827	0.403	5	4
TAN	0.797	0.441	8	6
XGBoost	0.854	0.340	2	3

Cuadro 3.2: Comparación de modelos con métricas de precisión y log loss

los comparten una característica esencial: son métodos de boosting basados en árboles de decisión, una técnica que ha demostrado en numerosas ocasiones su capacidad para adaptarse a distintos tipos de datos y relaciones no lineales. De hecho, estos modelos se podrían considerar el estado del arte en conjuntos de datos tabulares. En esta misma línea se encuentra **LightGBM**, que, si bien en este estudio no alcanza los mismos niveles de rendimiento, sigue mostrando un comportamiento competitivo en varios conjuntos de datos concretos.

El tercer puesto global lo ocupa **EBM** (Explainable Boosting Machine), que consigue alcanzar un gran rendimiento competitivo (0.849 de accuracy, 0.339 de log loss) a pesar de tener un enfoque más centrado en la interpretabilidad. A pesar de que no se suele considerar entre los modelos más nombrados en competiciones de machine learning o en contextos donde prima únicamente el rendimiento, EBM demuestra que es posible alcanzar resultados muy cercanos a los de modelos punteros como CatBoost o XGBoost sin renunciar a la transparencia en la toma de decisiones. De hecho, ha conseguido ser el mejor modelo hasta en 4 datasets.

En el otro extremo, modelos clásicos como Gaussian Naive Bayes o Logistic Regression presentan un rendimiento sensiblemente inferior. GaussianNB, en particular, obtiene la peor puntuación tanto en precisión (0.751) como en log loss (2.152), quedando relegado al último puesto en ambos rankings. Esto era algo esperable, dado que este tipo de modelo realiza suposiciones muy estrictas sobre la distribución de los datos, lo cual rara vez se cumple en la práctica.

Por otra parte, TAN, LDA, NAM y Random Forest muestran un rendimiento intermedio. Aunque no alcanzan las cifras de los modelos de boosting, siguen siendo opciones bastante válidas en contextos donde se priorizan otros factores como la interpretabilidad (en el caso de TAN, LDA y NAM) o la robustez frente al overfitting (como ocurre con Random Forest).

Cuando se observa en profundidad el desglose por dataset 3.3, se pone de manifiesto la consistencia de los modelos de boosting: tanto XGBoost, LightGBM como CatBoost aparecen con frecuencia entre los tres mejores en términos de precisión. En concreto, CatBoost y XGBoost son los modelos más precisos en 4 de los 15 conjuntos de datos respectivamente, seguido por XGBoost (3 veces).

Dataset	Inst	Feat	Best	2nd Best	3rd Best
ada_agnostic	4562	48	XGBoost (0.858)	CatBoost (0.857)	EBM (0.855)
colic	368	26	XGBoost (0.875)	CatBoost (0.867)	RandomForest (0.858)
credit-approval	690	15	EBM (0.880)	CatBoost (0.877)	RandomForest (0.871)
credit-g	1000	20	XGBoost (0.769)	EBM (0.760)	CatBoost (0.746)
electricity	45312	8	XGBoost (0.944)	LightGBM (0.914)	CatBoost (0.905)
elevators	16599	18	EBM (0.905)	LogisticReg (0.899)	CatBoost (0.898)
heart-h	294	13	CatBoost (0.837)	EBM (0.830)	RandomForest (0.827)
jasmine	2984	144	CatBoost (0.817)	XGBoost (0.815)	LightGBM (0.812)
kc1	2109	21	CatBoost (0.859)	LightGBM (0.859)	EBM (0.859)
nomao	34465	118	LightGBM (0.974)	XGBoost (0.973)	CatBoost (0.972)
phoneme	5404	5	LightGBM (0.910)	CatBoost (0.907)	XGBoost (0.905)
profb	672	9	CatBoost (0.751)	LightGBM (0.713)	EBM (0.686)
qsar-biodeg	1055	41	EBM (0.882)	LightGBM (0.872)	CatBoost (0.869)
socmob	1156	5	EBM (0.952)	XGBoost (0.948)	RandomForest (0.944)
SpeedDating	8378	120	LightGBM (0.871)	LightGBM (0.871)	EBM (0.870)

Cuadro 3.3: Mejores modelos para cada dataset junto a su accuracy

Esto sugiere que los modelos de boosting son capaces de generalizar bien en una gran variedad de dominios, independientemente del número de instancias o características del dataset.

Otro dato muy relevante es que EBM se posiciona como el mejor modelo en hasta 4 datasets, al igual que CatBoost y XGBoost: *elevators*, *heart-h*, *qsar-biodeg* o *socmob*. Esto puede sugerir que, además de ser explicable, EBM puede ser una alternativa eficaz, a pesar de no ser uno de los primeros modelos que se nos vienen a la cabeza cuando pensamos en predicción de datos tabulares. Como han demostrado diversos estudios, este modelo desafía la norma tradicional que asociaba a los modelos interpretables con una menor precisión, rompiendo así el supuesto trade-off entre exactitud e interpretabilidad.

Curiosamente, además de lo que se podía esperar, ha habido un par de resultados sorprendentes. Por ejemplo, en el conjunto de datos *elevators*, el modelo de Regresión Logística ha logrado ser el segundo con mejor rendimiento, alcanzando una precisión del 89.9 %, solo por detrás de EBM y superando a los modelos de boosting, lo que puede indicar que este dataset, lo que puede indicar que este dataset presenta una estructura lineal relativamente clara o que las características relevantes están bien separadas linealmente. Por otro lado, sorprende el bajo rendimiento de Random Forest, que tan solo ha podido alcanzar el tercer puesto en rendimiento en 4 datasets, sin llegar a ninguna posición más alta. Esto podría deberse a que, en ciertos casos, su capacidad para manejar interacciones complejas no ha sido suficiente para superar a modelos más especializados como EBM o los métodos de boosting.

Comparativa de Tiempos

La Tabla 3.4 muestra una comparativa de los tiempos medios de entrenamiento y prueba de cada modelo, junto con sus respectivos rankings. En términos generales, se puede observar una clara distinción entre modelos simples e inter-

Modelo	Train (s)	Test (s)	Rank Train	Rank Test
CatBoost	29.598	0.011	8	5
EBM	36.846	0.003	10	1
GaussianNaiveBayesModel	0.074	0.004	1	2
LDA	0.406	0.004	2	2
LightGBM	2.271	0.046	5	9
LogisticReg	1.506	0.004	4	2
NAM	30.562	0.084	9	10
RandomForest	0.436	0.016	3	6
TAN	6.61	0.017	6	7
XGBoost	12.072	0.034	7	8

Cuadro 3.4: Tiempos de entrenamiento y prueba, junto con rankings por modelo

pretables frente a modelos complejos y de tipo ensemble.

Modelos como **GaussianNB**, **LDA** y **LogisticReg** destacan por su rapidez, con tiempos de entrenamiento inferiores al segundo y tiempos de inferencia casi instantáneos. Esto es algo evidente; son los modelos más 'simples', que no requieren de iteraciones como los modelos de boosting y se entrenan de una pasada.

Por otro lado, los modelos más complejos presentan diferencias notables. **EBM** (Explainable Boosting Machine), a pesar de ser un modelo interpretable, requiere el mayor tiempo de entrenamiento de todos los evaluados (36.8s), aunque compensa con un tiempo de inferencia extremadamente bajo. Este comportamiento se puede atribuitr a su proceso de entrenamiento basado en boosting por componentes aditivos, lo que implica múltiples etapas de ajuste fino. De forma similar, **NAM**, también interpretable, se sitúa como el segundo más lento en entrenamiento (30.6s) y, a diferencia de EBM, presenta además un tiempo de predicción elevado.

En cuanto a los modelos no interpretables, destacan **CatBoost** y **XGBoost** por sus tiempos relativamente altos de entrenamiento (29.6s y 12.1s respectivamente), mientras que **RandomForest** sorprende con un tiempo de entrenamiento muy bajo (0.43s).

Tradeoff: Precisión - Explicabilidad

En la Figura 3.10 se muestra una representación gráfica de los rankings de precisión frente a los rankings de tiempo de entrenamiento. Cuanto más a la izquierda y abajo se sitúe un modelo, mejor es su combinación de rendimiento y eficiencia temporal.

Lo primero que se destaca en el gráfico es el excelente posicionamiento de **CatBoost** y **XGBoost**, modelos de boosting no interpretables que han logrado una alta precisión (ranks 1 y 2 respectivamente), con tiempos de entrenamiento relativamente altos. En contraste, **EBM** se posiciona como uno de los modelos más precisos (rank 3 en accuracy), pero con el peor tiempo de entrenamiento (rank 10), lo que muestra el coste computacional de su interpretabilidad.

El modelo **GaussianNB**, aunque es extremadamente eficiente en entrenamiento

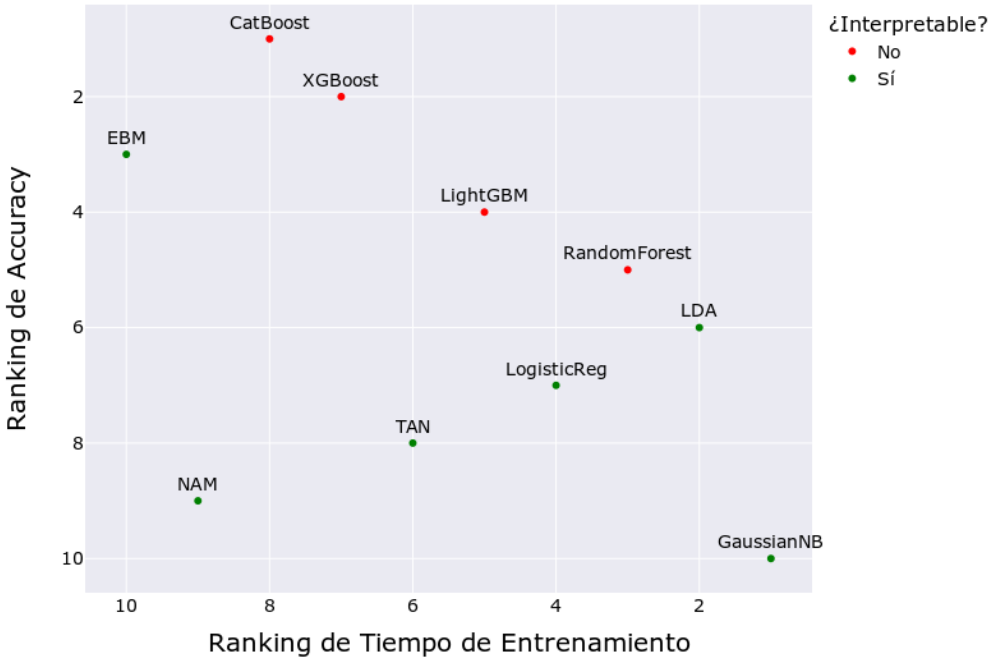

Figura 3.10: Gráfico comparativo de los ránkings Precisión vs Tiempo de Entrenamiento

(rank 1), presenta un rendimiento pobre en precisión (rank 10). Algo similar sucede con **NAM**, que muestra un trade-off desfavorable, con bajo rendimiento (rank 9) y alto tiempo de entrenamiento (rank 9). **LDA** y **LogisticReg** logran un equilibrio razonable entre eficiencia y rendimiento, lo que los hace adecuados para escenarios donde se prioriza la interpretabilidad y simplicidad.

Desde el punto de vista visual, se observa una interesante diferencia entre los modelos interpretables (marcados en verde). Por un lado, modelos como **GaussianNB**, **LDA** o **LinearModelInterpret** muestran tiempos de entrenamiento muy bajos, pero también una precisión más limitada. Por otro lado, modelos como **EBM** y **NAM** o **TAN** son más complejos e interpretables, pero a costa de un tiempo de entrenamiento más alto, especialmente los dos primeros. Esto puede indicar que, en algunos casos, la interpretabilidad no es gratuita: modelos que buscan un equilibrio entre explicabilidad y rendimiento, parecen pagar un precio en términos de coste computacional.

Glassbox vs Blackbox

A pesar de ser menos reconocidos o citados como modelos punteros en tareas con datos tabulares, los algoritmos de tipo *glassbox* han demostrado en este estudio que pueden ofrecer un rendimiento competitivo frente a soluciones más complejas.

De todos ellos, **EBM** ha sido el más destacado. No solo se ha mantenido consistentemente entre los tres mejores modelos en varios datasets, sino que también ha logrado ser el más preciso en 4 de los 15 conjuntos evaluados. Lo más relevante es que logra estos resultados manteniendo una estructura completamente interpretable, lo que rompe con el clásico trade-off entre interpretabilidad y rendimiento. Si bien su coste computacional es más elevado, EBM demuestra que es posible alcanzar resultados propios del estado del arte sin renunciar a la transparencia del modelo.

NAM, por su parte, representa una propuesta moderna que combina las ventajas de las redes neuronales con una estructura interpretativa. Aunque su rendimiento general ha sido bastante peor (puesto 8 en accuracy), ha mantenido cierta regularidad en varios datasets.

LDA y **Gaussian Naive Bayes** son modelos clásicos y extremadamente rápidos tanto en entrenamiento como en predicción. Aunque en términos globales su rendimiento ha sido inferior al de los modelos más sofisticados, LDA ha obtenido una precisión media (0.816) a modelos mucho más complejos como RandomForest (0.827) o LightGBM (0.833). En esta misma línea, también es destacable el comportamiento de **TAN**, un modelo que ha combinado buenos tiempos con un rendimiento aceptable en múltiples datasets.

Como era de esperar, los modelos blackbox han dado un resultado general mejor, pues son considerados el estado del arte en conjuntos de datos tabulares. No obstante, se demuestra que los modelos glassbox siguen siendo opciones altamente válidas, no solo por su velocidad y transparencia, sino también por su potencial competitivo en situaciones concretas.

3.3.5. Discusión

Los resultados obtenidos permiten sacar varias conclusiones, en especial en relación al rendimiento de modelos explicables frente a los modelos considerados “caja negra". La diversidad de datasets seleccionados (con distintas características, tamaños y complejidades) ha permitido evaluar el comportamiento de cada modelo en diversos escenarios, lo que proporciona robustez a las conclusiones.

En cuanto a rendimiento, los modelos de boosting (CatBoost, XGBoost, LightGBM) han demostrado un liderazgo claro. CatBoost, en particular, ha sido el más preciso y fiable en la estimación de probabilidades. Sin embargo, lo más destacable ha sido el rendimiento de EBM (Explainable Boosting Machine), que ha conseguido superar a estos modelos en varios datasets. Este resultado es especialmente significativo ya que EBM mantiene una estructura completamente interpretable, desafiando así la percepción tradicional de que la precisión y la interpretabilidad están reñidas.

Desde el punto de vista de la eficiencia computacional, los modelos complejos tienen poco que hacer con modelos clásicos como GaussianNB, LDA y Logistic Regression, que son los más competitivos en términos de tiempo de entrenamiento. No obstante, su simplicidad también limita su rendimiento predictivo. Un caso interesante fue el de Logistic Regression, que logró un rendimiento sor-

prendentemente alto en el dataset *elevators*, lo que sugiere que, en contextos con relaciones lineales bien definidas, siguen siendo opciones válidas.

La comparativa de tiempos evidenció que la interpretabilidad puede conllevar un coste computacional elevado. EBM, pese a su excelente rendimiento, fue el modelo más lento en entrenamiento. NAM, también interpretable, mostró un balance menos favorable al combinar bajo rendimiento con alta demanda computacional, lo que puede plantear dudas sobre su viabilidad.

El análisis del trade-off entre precisión y explicabilidad visualizado en la Figura de rankings muestra dos grandes grupos: modelos altamente precisos pero costosos computacionalmente (CatBoost, EBM, XGBoost) y modelos muy eficientes pero menos precisos (GaussianNB, LDA). Esta división sugiere que la elección del modelo debe estar guiada por las prioridades del contexto: si se valora la transparencia y rapidez, los modelos glassbox simples son útiles; si prima el rendimiento absoluto, los modelos ensemble siguen dominando.

Finalmente, se ha evidenciado que los modelos explicables tienen una oportunidad real en aplicaciones prácticas, especialmente EBM, que logra equilibrar transparencia y precisión. Si bien los blackbox continúan liderando en rendimiento puro, el hecho de que un modelo explicable haya sido el mejor en varios datasets podría avalar su uso en escenarios sensibles como medicina, finanzas o justicia, donde la interpretabilidad es un requisito.

En conclusión, este trabajo no solo confirma el dominio de los modelos de boosting en tareas de clasificación binaria, sino que también abre la puerta a la adopción de soluciones explicables sin que ello implique sacrificar significativamente la precisión.

Capítulo 4

Análisis de impacto

Este trabajo de fin de grado, *Integración y evaluación de modelos interpretables utilizando la librería interpretML*, ha tenido un impacto significativo a nivel personal. Además, a lo largo del desarrollo del proyecto se han tomado decisiones que persiguen tanto la innovación técnica como una aportación al ecosistema de inteligencia artificial, promoviendo la transparencia, la explicabilidad y la confianza en los modelos de machine learning.

4.1. Impacto personal

El desarrollo de este proyecto ha supuesto una importante oportunidad tanto formativa como de investigación. Me ha permitido crecer y aprender en múltiples aspectos, desde la integración y extensión de librerías de código abierto, como es el caso de InterpretML, hasta la publicación de un paquete propio en PyPI. Gracias a ello, he comprendido de primera mano cómo se puede contribuir activamente a herramientas utilizadas por la comunidad científica y técnica. Además, la investigación y lectura de diversos artículos relacionados con modelos interpretables me han ayudado a estructurar mejor el trabajo, organizar las ideas, y sobre todo a desarrollar una mirada crítica sobre los resultados obtenidos.

Por otra parte, este proyecto me ha abierto las puertas a la Inteligencia Artificial Explicable (XAI), un campo de la IA con un gran potencial en el presente y el futuro. La posibilidad de entender y explicar cómo un modelo llega a una determinada decisión no solo es útil desde el punto de vista técnico, sino que es clave para garantizar transparencia en entornos sensibles, como la salud o la economía.

4.2. Impacto social

En relación a la última idea, también se puede afirmar que este trabajo puede tener impacto a nivel social. Este proyecto contribuye a la democratización de la

inteligencia artificial, fomentando la comprensión y supervisión de los modelos por parte de usuarios no expertos, como un médico o un economista.

Así, se promueve una IA más accesible, transparente y justa, lo que se traduce en una mayor confianza de la sociedad en las decisiones automatizadas, algo en lo que hoy en día todavía todavía hay mucho escepticismo. El hecho de que las predicciones puedan explicarse de forma clara mejora la aceptación y percepción pública de la IA.

4.3. Impacto empresarial y económico

Los resultados de este trabajo tienen potencial de aplicación en entornos empresariales, especialmente en sectores que necesiten tomar decisiones automatizadas bajo criterios de transparencia y trazabilidad, como la sanidad, las finanzas, los seguros o los recursos humanos. En estos contextos, los modelos interpretables son una herramienta clave para facilitar auditorías, justificar decisiones ante organismos reguladores y generar confianza en los usuarios y clientes finales.

La integración de clasificadores explicables en la librería InterpretML, junto con el desarrollo de una extensión publicada en PyPI, es una solución práctica que puede ser fácilmente adoptada por equipos de ciencia de datos. Esta herramienta contribuye a reducir el tiempo de desarrollo, simplificando el proceso de validación y mejora la comprensión del comportamiento del modelo sin necesidad de recurrir a técnicas menos fiables.

Desde el punto de vista económico, el uso de modelos explicables puede traducirse en importantes beneficios: permite evitar errores costosos derivados de decisiones mal interpretadas, reducir riesgos legales asociados a modelos black-box, y optimizar los recursos humanos y computacionales durante el ciclo de vida de los modelos.

4.4. ODS de la Agenda 2030

El proyecto guarda relación con varios Objetivos de Desarrollo Sostenible (ODS):

- **ODS 9: Industria, innovación e infraestructura**, al promover herramientas de IA accesibles y abiertas que fomentan la innovación responsable.
- **ODS 16: Paz, justicia e instituciones sólidas**, al facilitar la construcción de sistemas algorítmicos más transparentes, auditables y confiables, especialmente en sectores regulados.

Capítulo 5

Conclusiones y trabajo futuro

Durante el desarrollo de este Trabajo Fin de Grado se ha llevado a cabo un análisis exhaustivo sobre modelos interpretables aplicados al aprendizaje automático en datos tabulares, centrando los esfuerzos en la integración de dichos modelos en el framework abierto InterpretML. Esta integración no solo ha permitido ampliar las capacidades actuales de la librería, sino que ha facilitado una evaluación sistemática del rendimiento de modelos explicables frente a alternativas más complejas y no interpretables (blackbox).

Uno de los principales logros del proyecto ha sido la adaptación y extensión de *InterpretML* para dar soporte a una amplia gama de modelos, incluyendo clasificadores probabilísticos (Naive Bayes, TAN, LDA) o modelos aditivos (NAM). El desarrollo del paquete *interpret-extension* y su publicación en PyPI permite estandarizar y distribuir las herramientas desarrolladas, haciendo posible su reutilización en futuros proyectos y por otros investigadores o profesionales interesados en interpretabilidad.

Asimismo, se ha llevado a cabo una comparativa extensa de estos modelos sobre el benchmark *TabZilla*, utilizando una colección diversa de datasets de clasificación binaria. Los resultados obtenidos muestran que los modelos glassbox, especialmente EBM, son capaces de obtener rendimientos muy competitivos frente a modelos como CatBoost o XGBoost, alcanzando incluso la mejor precisión en varios conjuntos. Este hallazgo refuerza la idea de que, en contextos donde la interpretabilidad es un requisito crítico, no es necesario renunciar al rendimiento predictivo.

A nivel personal, este proyecto ha supuesto una etapa importante en mi formación. Me ha permitido consolidar conocimientos teóricos y aplicados sobre modelos probabilísticos, aprendizaje aditivo, benchmarking, además de adquirir experiencia en herramientas prácticas como PyTorch, scikit-learn y Optuna. Por otro lado, ha fomentado habilidades menos técnicas pero también relevantes, como la publicación y documentación de paquetes, la gestión de dependencias, y la comunicación de resultados de forma clara y estructurada.

5.1. Líneas de Trabajo Futuro

Este proyecto abre varias líneas de investigación y desarrollo que podrían tratarse en otros trabajos:

- **Ampliación continuada de interpretML**: Aunque se han integrado modelos relevantes, quedan otros enfoques interpretables por incorporar. Explorar cómo adaptar las visualizaciones de interpretML a estos modelos sería un paso natural.
- **Aplicación a contextos reales sensibles**: Dado que la motivación de este trabajo nace de aplicaciones sensibles (sanidad, justicia o finanzas, por ejemplo), una línea interesante sería validar estas herramientas en colaboración con instituciones reales, donde la interpretabilidad no sea solo deseable, sino un requisito.
- **Exploración de nuevos criterios de interpretabilidad**: En este trabajo se ha asumido un enfoque centrado en la descomposición aditiva y las visualizaciones individuales. Sería interesante incorporar métricas cuantitativas de interpretabilidad (por ejemplo, sparsity, simulatability) y analizar su correlación con la comprensión humana.

En definitiva, este trabajo ha permitido demostrar que los modelos interpretables no solo son viables en términos de precisión, sino que, con el diseño y herramientas adecuadas, pueden integrarse eficazmente en pipelines de machine learning actuales, manteniendo la transparencia de las decisiones algorítmicas.

5.2. Cumplimiento de Objetivos

Hagamos un análisis sobre los objetivos planteados inicialmente:

- **O1: Integración de clasificadores bayesianos en interpretML**: Se integraron los modelos Naive Bayes y TAN utilizando scikit-learn y pyAgrum, adaptados a la estructura explicativa de interpretML.
- **O2: Adaptación de las funcionalidades explicativas de interpretML a los nuevos modelos**: Se implementaron explicaciones globales y locales para los nuevos modelos, incluyendo visualizaciones personalizadas (como mapas de calor en TAN), y adaptando las de otros modelos como NAM.
- **O3: Aplicación de los modelos en conjuntos de datos reales**: Los modelos fueron evaluados en 15 datasets de clasificación binaria con diversas características mediante el benchmark TabZilla. En este caso fueron reales; para un análisis más extenso, se podrían utilizar datasets simulados.
- **O4: Integración de modelos aditivos con redes neuronales en interpretML:** Se integraron los Neural Additive Models (NAM), resolviendo incompatibilidades y adaptando visualizaciones para su uso con InterpretML.
- **O5: Comparativa de modelos en precisión e interpretabilidad**: Se realizó una evaluación sistemática comparando modelos interpretables y caja

negra usando métricas como accuracy y log loss.

Por tanto, se puede concluir que todos los objetivos planteados al inicio del proyecto han sido cumplidos, contribuyendo tanto a la extensión de la librería interpretML como al análisis del potencial de los modelos interpretables en contextos reales.

Bibliografía

[1] E. H. Shortliffe, R. Davis, S. G. Axline, B. G. Buchanan, C. Green y S. N. Cohen, «Computer-based consultations in clinical therapeutics: Explanation and rule acquisition capabilities of the MYCIN system», 1975, https://www.sciencedirect.com/science/article/pii/0010480975900099.

[2] J. Benitez, J. Castro e I. Requena, «Are artificial neural networks black boxes?», 1997, https://ieeexplore.ieee.org/abstract/document/623216.

[3] R. Andrews, J. Diederich y A. B. Tickle, «Survey and critique of techniques for extracting rules from trained artificial neural networks», 1995, https://www.sciencedirect.com/science/article/pii/0950705196819204.

[4] T. Hastie y R. Tibshirani, «Generalized Additive Models», 1986, https://www.jstor.org/stable/2245459.

[5] S. Lundberg y S.-I. Lee, «A Unified Approach to Interpreting Model Predictions», 2017, https://arxiv.org/abs/1705.07874.

[6] A. B. Arrieta, N. Díaz-Rodríguez, J. D. Ser et al., «Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI», 2019, https://arxiv.org/abs/1910.10045.

[7] A. Adadi y M. Berrada, «Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)», 2018, https://ieeexplore.ieee.org/abstract/document/8466590.

[8] H. Nori, S. Jenkins, P. Koch y R. Caruana, «InterpretML: A Unified Framework for Machine Learning Interpretability», 2019, https://arxiv.org/abs/1909.09223.

[9] L. Grinsztajn, E. Oyallon y G. Varoquaux, «Why do tree-based models still outperform deep learning on tabular data?», 2022, https://arxiv.org/abs/2207.08815.

[10] D. McElfresh, S. Khandagale, J. Valverde et al., «When Do Neural Nets Outperform Boosted Trees on Tabular Data?», 2024, https://arxiv.org/abs/2305.02997.

[11] S. Kruschel, N. Hambauer, S. Weinzierl, S. Zilker, M. Kraus y P. Zschech, «Challenging the Performance-Interpretability Trade-Off: An Evaluation of Interpretable Machine Learning Models», 2025, https://doi.org/10.1007/s12599-024-00922-2.

[12] L. Wang, Y. Liu, M. Mammadov, M. Sun y S. Qi, «Discriminative Structure Learning of Bayesian Network Classifiers from Training Dataset and Testing Instance», 2019, https://www.researchgate.net/publication/333061798_Discriminative_Structure_Learning_of_Bayesian_Network_Classifiers_from_Training_Dataset_and_Testing_Instance.

[13] W. Nuttall, E. Patelli, E. Smith, D. Webbe-Wood y S. Middleburgh, «The Way Ahead», en *Perspectives on Engineering Uncertainty: Civil Nuclear Energy Safety and Efficiency*. 2025. dirección: https://doi.org/10.1007/978-3-031-83254-3_6.

[14] *Machine Learning Explained*, https://ml-explained.com/blog/linear-discriminant-analysis-explained, 2021.

[15] B. Ghojogh y M. Crowley, «Linear and Quadratic Discriminant Analysis: Tutorial», 2019, https://arxiv.org/abs/1906.02590.

[16] E. Guldogan, F. Yagin, A. Pinar et al., «A proposed tree-based explainable artificial intelligence approach for the prediction of angina pectoris», 2023, https://doi.org/10.1038/s41598-023-49673-2.

[17] Y. Lou, R. Caruana, J. Gehrke y G. Hooker, «Accurate intelligible models with pairwise interactions», 2013, https://doi.org/10.1145/2487575.2487579.

[18] R. Agarwal, L. Melnick, N. Frosst et al., «Neural Additive Models: Interpretable Machine Learning with Neural Nets», 2021, https://arxiv.org/abs/2004.13912.

[19] S. Talebi, *10 Decision Trees are Better Than 1*, https://towardsdatascience.com/10-decision-trees-are-better-than-1-719406680564/, 2023.

[20] *A Quick Guide to Boosting Algorithms in Machine Learning*, https://brainalyst.in/boosting-algorithms-in-machine-learning/.

[21] K. J. Wong, *CatBoost vs. LightGBM vs. XGBoost*, https://towardsdatascience.com/catboost-vs-lightgbm-vs-xgboost-c80f40662924/, 2022.

[22] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush y A. Gulin, «CatBoost: unbiased boosting with categorical features», 2019, https://arxiv.org/abs/1706.09516.

[23] InterpretML, *Explainable Boosting Machine*, https://interpret.ml/docs/ebm.html.

[24] J. J. Levy y A. J. O'Malley, «Don't dismiss logistic regression: the case for sensible extraction of interactions in the era of machine learning», 2020, https://doi.org/10.1186/s12874-020-01046-3.

[25] C.-H. Chang, S. Tan, B. Lengerich, A. Goldenberg y R. Caruana, «How Interpretable and Trustworthy are GAMs?», 2021, https://arxiv.org/abs/2006.06466.

[26] Q. L. Yiyan Zhang e Y. Xin, «Research on eight machine learning algorithms applicability on different characteristics data sets in medical classification tasks», 2024, https://www.frontiersin.org/journals/computational-neuroscience/articles/10.3389/fncom.2024.1345575/full.

JAVIER PEREZ VARGAS

MemoriaTFG___JavierPerezVargas.pdf

Turnitin Memoria Final

TFG ETSIINF (Moodle PP)

Universidad Politecnica de Madrid

Detalles del documento

Identificador de la entrega

trn:oid:::1:3266877303

Fecha de entrega

2 jun 2025, 3:55 p.m. GMT+2

Fecha de descarga

2 jun 2025, 4:06 p.m. GMT+2

Nombre de archivo

24065_JAVIER_PEREZ_VARGAS_MemoriaTFG___JavierPerezVargas_83714_509680747.pdf

Tamaño de archivo

1.8 MB

54 Páginas

14.947 Palabras

81.021 Caracteres

3% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text

Exclusions

- 1 Excluded Source

Top Sources

0% Internet sources
0% Publications
3% Submitted works (Student Papers)

Top Sources

0% Internet sources
0% Publications
3% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1 Student papers
Universidad Politécnica de Madrid 1%

2 Student papers
The University of Manchester <1%

3 Student papers
Universidad Pública de Navarra <1%

4 Student papers
Universidad de Santiago de Chile <1%

5 Student papers
TU Delft <1%

6 Student papers
Eastern Institute of Technology <1%

7 Student papers
unbosque <1%

8 Student papers
Universidad de Salamanca <1%

9 Student papers
University of Florida <1%

10 Student papers
City University <1%

11 Student papers
Corporación Universitaria Minuto de Dios, UNIMINUTO <1%

12 Student papers

Universidad Autónoma de Nuevo León <1%

13 Student papers

Unviersidad de Granada <1%

14 Student papers

Imperial College of Science, Technology and Medicine <1%

15 Student papers

King's College <1%

16 Student papers

Universidad Internacional de la Rioja <1%

17 Student papers

Universidad Carlos III de Madrid - EUR <1%

18 Student papers

University of Wales Swansea <1%

19 Student papers

ITESM: Instituto Tecnologico y de Estudios Superiores de Monterrey <1%

20 Student papers

Johns Hopkins Unversity <1%

21 Student papers

RMIT University <1%

22 Student papers

Universidad TecMilenio <1%

23 Student papers

Universidade de Sao Paulo <1%

24 Student papers

University of New South Wales <1%

25 Student papers

Tilburg University <1%

26 Student papers

Universidad Carlos III de Madrid <1%

27 Student papers

University of Birmingham <1%

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Mon Jun 02 18:15:03 CEST 2025
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)