UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA BIOMÉDICA

TRABAJO FIN DE GRADO

EVALUACIÓN Y ESTUDIO DE LA INTERPRETABILIDAD DE REDES KOLMOGOROV-ARNOLD EN LA GENERACIÓN DE DATOS TABULARES SINTÉTICOS

PAULA RODRÍGUEZ SÁNCHEZ

2025

Grado en Ingeniería Biomédica

TRABAJO DE FIN DE GRADO

Título: Evaluación y estudio de la interpretabilidad de Redes Kolmogorov-Arnold en la generación de datos tabulares sintéticos.

Autor: Paula Rodríguez Sánchez

Tutor: Dr. Alberto Belmonte Hernández

Departamento: Señales, Sistemas y Radiocomunicaciones (SSR)

MIEMBROS DEL TRIBUNAL

Presidente: ----------

Vocal: ----------

Secretario: ----------

Suplente: ----------

Los miembros del tribunal arriba nombrados acuerdan otorgar la calificación de:

Fecha de lectura: ----------

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA BIOMÉDICA

TRABAJO FIN DE GRADO

EVALUACIÓN Y ESTUDIO DE LA INTERPRETABILIDAD DE REDES KOLMOGOROV-ARNOLD EN LA GENERACIÓN DE DATOS TABULARES SINTÉTICOS

PAULA RODRÍGUEZ SÁNCHEZ

2025

Resumen

Durante los últimos años, la generación de datos sintéticos se ha convertido en una herramienta esencial para facilitar el entrenamiento de modelos de aprendizaje automático en contextos donde los datos reales son escasos, sensibles o inaccesibles. Modelos como CTGAN y TVAE han sido ampliamente adoptados por su capacidad para capturar y replicar distribuciones complejas de datos tabulares, permitiendo preservar la privacidad y mitigar la escasez de datos reales [1].

Sin embargo, estas arquitecturas están basadas en redes neuronales profundas tradicionales, lo que limita su interpretabilidad y dificulta la validación transparente del proceso de generación.

Este proyecto propone como solución a este problema la incorporación de Redes Kolmogorov-Arnold (KANs) en modelos generativos de datos sintéticos tabulares. Estas redes, inspiradas en el teorema de representación de Kolmogorov-Arnold, reemplazan los pesos escalares tradicionales por funciones univariantes aprendibles (splines), situadas en los bordes entre nodos. Esta arquitectura nos permite descomponer funciones multivariantes complejas en representaciones más simples y comprensibles, facilitando de esta forma tanto la interpretación del modelo como la extracción simbólica de reglas matemáticas [2, 3, 4, 5].

El objetivo general del trabajo es implementar y evaluar una arquitectura generativa basada en KANs, que mantenga la precisión de los modelos generativos existentes como CTGAN o TVAE, pero que además aporte explicaciones interpretables sobre el proceso de generación de datos. Esto es especialmente relevante en sectores como el biomédico, donde la trazabilidad, la validez y la transparencia de los datos sintéticos son requisitos fundamentales para su adopción segura.

Se desarrollará una arquitectura generativa basada en KANs, entrenada y evaluada sobre conjuntos de datos tabulares y se analizará su rendimiento en términos de fidelidad estadística, diversidad, realismo y coherencia, así como su capacidad de explicación global y local del proceso generativo.

Palabras clave

Interpretabilidad, Datos sintéticos, Redes Kolmogorov-Arnold, Aprendizaje automático, Funciones spline, Simbolificación, Sparsity, Evaluación TSTR/TRTS, TVAE, CTGAN, Modelos generativos.

Summary

Over the past few years, synthetic data generation has emerged as a key technique to enable the training of machine learning models in contexts where real data is scarce, sensitive, or restricted due to privacy concerns. Generative models such as CTGAN and TVAE have gained widespread adoption for their ability to learn and replicate complex distributions in tabular datasets, thereby supporting both data augmentation and privacy preservation in critical domains [1].

Despite their effectiveness, these architectures rely on traditional deep neural networks, which inherently lack interpretability. This opacity hinders the transparent validation of the synthetic data generation process and limits the trust that domain experts—especially in regulated sectors—can place in the outcomes of these models.

To address this limitation, the present project proposes the integration of Kolmogorov–Arnold Networks (KANs) into the design of generative models for tabular data. KANs are a novel neural architecture inspired by the Kolmogorov–Arnold representation theorem. Instead of scalar weights, they employ learnable univariate functions—typically implemented as B-splines—on the edges between neurons. This design enables the decomposition of complex multivariate relationships into interpretable and mathematically tractable components. As a result, KAN-based models provide not only competitive generative performance but also allow for symbolic interpretation and rule extraction [2, 3, 4, 5].

The overarching goal of this project is to develop and evaluate a generative architecture based on KANs that matches or surpasses the performance of established models such as CTGAN and TVAE, while introducing interpretability as a first-class property of the system. This is particularly relevant in fields such as biomedicine, where traceability, transparency, and scientific validity are critical for the safe and ethical deployment of synthetic data.

The proposed KAN-based generative model will be trained and tested on real-world tabular datasets. Its evaluation will include metrics of statistical fidelity, diversity, realism, and structural coherence. Furthermore, the model's capacity for both global and local interpretability will be assessed through symbolic analysis and extraction of functional relationships governing the generation process.

Keywords

Interpretability, Synthetic data, Kolmogorov-Arnold Networks, Machine learning, Spline functions, Symbolification, Sparsity, TSTR/TRTS evaluation, TVAE, CTGAN, Generative models.

Índice

Listado de figuras

Listado de tablas

Glosario

IA - Inteligencia Artificial

KAN - Kolmogorov-Arnold Networks

TVAE - Tabular Variational Autoencoder

CTGAN - Conditional Tabular Generative Adversarial Network

DNN - Deep Neural Network

GDPR - General Data Protection Regulation

TSTR - Train on Synthetic, Test on Real

TRTS - Train on Real, Test on Synthetic

MLP - Multi-Layer Perceptron

VAE - Variational Autoencoder

GAN - Generative Adversarial Network

TabDDPM - Tabular Denoising Diffusion Probabilistic Model

KL - Kullback–Leibler

PDF - Probability Density Function

CPU - Central Processing Unit

GPU - Graphics Processing Unit

CUDA - Compute Unified Device Architecture

SDV - Synthetic Data Vault

SHAP - SHapley Additive exPlanations

LIME - Local Interpretable Model-agnostic Explanations

GLM - Generalized Linear Model

GAM - Generalized Additive Model

Introducción y objetivos

1.1 Introducción

La inteligencia artificial (IA) y, en particular, las redes neuronales profundas (DNNs), han evolucionado a un ritmo vertiginoso en los últimos años, integrándose en numerosos sectores y generando un impacto significativo en la sociedad. Tanto con el aumento de la capacidad de procesamiento como con la abundancia de datos etiquetados, estos sistemas han alcanzado y superado el rendimiento humano en tareas específicas de alta complejidad.

Sin embargo, el éxito de estos modelos ha venido acompañado de una creciente preocupación por su naturaleza de "caja negra", especialmente en lo que respecta a sus procesos de inferencia. La falta de transparencia en su lógica interna dificulta la comprensión tanto para los usuarios como para los expertos, lo que genera gran incertidumbre sobre su funcionamiento y confiabilidad [8]. En consecuencia, las preguntas fundamentales sobre estos sistemas no solo comienzan a recaer sobre el uso de los algoritmos, sino también sobre los principios que explican su funcionamiento.

Por esto, para garantizar la confiabilidad de un modelo, es indispensable contar con herramientas que permitan analizar y comprender el razonamiento detrás de sus decisiones. Una red neuronal explicable no solo busca realizar predicciones precisas, sino también proporcionar información clara sobre su funcionamiento, facilitando así su validación y adopción en entornos críticos.

Paralelamente, la generación de datos sintéticos mediante modelos generativos ha cobrado gran importancia, especialmente en sectores donde la privacidad y el acceso a datos reales son limitados, como la medicina, las finanzas y la ciberseguridad. Modelos como CTGAN [7] y TVAE [9] han demostrado ser soluciones efectivas para la generación de datos tabulares, permitiendo preservar la privacidad y mitigar la escasez de datos.

Pese a ello, estos modelos generativos también comparten el problema de la falta de interpretabilidad. Al igual que otras redes profundas, su funcionamiento interno es realmente difícil de analizar, lo que obstaculiza la validación de la calidad y fiabilidad de los datos que se generan [10]. Esta situación se vuelve especialmente crítica cuando los datos sintéticos se destinan al entrenamiento de sistemas encargados de tomar decisiones sensibles.

Figura 1.1: Esquema de modelo caja negra vs modelo explicable.

1.2 Importancia de la interpretabilidad en la IA

Uno de los principales desafíos actuales de la inteligencia artificial es su falta de transparencia, lo que genera desconfianza y plantea importantes problemas éticos. La dependencia de modelos cuyas decisiones no pueden ser comprendidas ni validadas resulta especialmente preocupante en contextos críticos, como la generación de datos sintéticos, donde la ausencia de interpretabilidad puede comprometer la calidad y utilidad de los datos generados [11].

Desde una perspectiva legal y ética, la interpretabilidad es clave para prevenir la discriminación algorítmica y garantizar la equidad. El "derecho a la explicación", recogido en el Artículo 22 del Reglamento General de Protección de Datos (GDPR) de la Unión Europea, establece que las personas no deben ser objeto de decisiones automatizadas con efectos legales o significativos, salvo bajo condiciones específicas [12]. Este marco normativo exige garantizar la intervención humana, el derecho a expresar el propio punto de vista y a impugnar decisiones automatizadas. Sin una comprensión clara del funcionamiento interno de los modelos, estos derechos no pueden garantizarse de forma efectiva [8, 12].

Más allá del marco regulador, la interpretabilidad es esencial para asegurar la responsabilidad y la seguridad en sectores de alto impacto como la salud, la robótica o la movilidad autónoma, donde errores derivados de correlaciones espurias pueden tener consecuencias graves [8]. En este sentido, regulaciones recientes como la Artificial Intelligence Act de la Unión Europea refuerzan la necesidad de desarrollar sistemas explicables y responsables [12].

A medida que la inteligencia artificial se integra en aspectos cada vez más relevantes de la sociedad, su interpretabilidad se convierte no solo en una exigencia técnica, sino también en una condición para su aceptación ética y social. En este contexto, presentamos el estudio de las Redes Kolmogorov–Arnold (KANs) [2, 3], una arquitectura neuronal reciente orientada a la explicabilidad. Este trabajo propone su aplicación a la generación de datos sintéticos tabulares, ofreciendo una alternativa más interpretable y flexible frente a los modelos generativos tradicionales [7].

1.3 Redes KAN para la generación de datos sintéticos

Los perceptrones multicapa (MLPs) han sido la base de los modelos de aprendizaje profundo debido a su capacidad para aproximar funciones no lineales, respaldada por el Teorema de Aproximación Universal [3]. Sin embargo, su uso presenta limitaciones en términos de interpretabilidad y eficiencia para capturar estructuras complejas.

La exploración de arquitecturas alternativas ha llevado al desarrollo de las Redes Kolmogorov-Arnold (KANs), que han surgido con fuerza como solución prometedora ante los desafíos de los MLPs. Inspiradas en el Teorema de Representación de Kolmogorov-Arnold, estas redes descomponen cualquier función continua multivariante en composiciones de funciones univariantes, proporcionando una representación más estructurada y comprensible [2, 3].

A diferencia de los MLP, que emplean funciones de activación fijas en los nodos, las KAN utilizan funciones de activación aprendibles ubicadas en los bordes. En esta arquitectura, cada peso es reemplazado por una función unidimensional parametrizada mediante un spline, lo que permite ajustar dinámicamente las transformaciones de entrada y mejorar la precisión de la representación final. Este diseño innovador nos ofrece múltiples ventajas: mayor capacidad para modelar estructuras matemáticas y relaciones complejas en los datos, eficiencia en el uso de parámetros y una interpretabilidad superior.

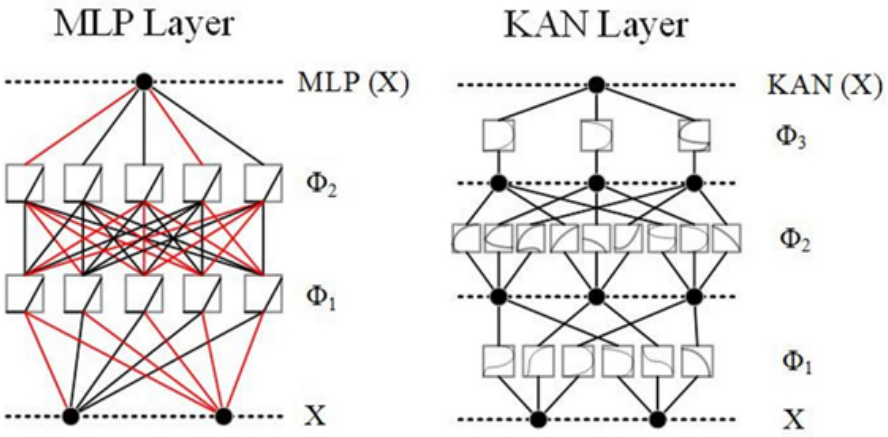


Figura 1.2: Esquema de arquitectura MLP vs arquitectura KAN.

Gracias a estas propiedades, consideramos las KANs como una alternativa muy interesante para la generación de datos sintéticos tabulares, no solo representando un avance en términos de precisión y eficiencia, sino también permitiendo un mayor control sobre la calidad de los datos generados, asegurando que sean representativos y útiles para aplicaciones críticas.

1.4 Objetivos del proyecto

El principal objetivo de este proyecto es la implementación de Redes Kolmogorov-Arnold (KANs) en la generación de datos sintéticos tabulares, con el fin de analizar y mejorar su interpretabilidad, proporcionando una comprensión matemática clara del funcionamiento interno del modelo y del proceso de toma de decisiones. Para ello se propone el esquema de trabajo presentado en la Figura 1.3. En primer lugar, se seleccionará un conjunto de datos relevante del ámbito sanitario, dado que este campo no solo presenta un elevado interés práctico, sino que también exige altos niveles de explicabilidad en los modelos utilizados.

A continuación, se desarrollarán y entrenarán arquitecturas generativas como TVAE [9] y CTGAN [7], así como sus variantes análogas implementadas mediante redes KAN (Kolmogorov–Arnold Networks). Sobre estas últimas se aplicarán distintas técnicas de reducción de complejidad, incluyendo la poda de nodos poco representativos y la aplicación de regresión

simbólica para aproximar los nodos mediante funciones matemáticas sencillas, entre otros métodos de simplificación.

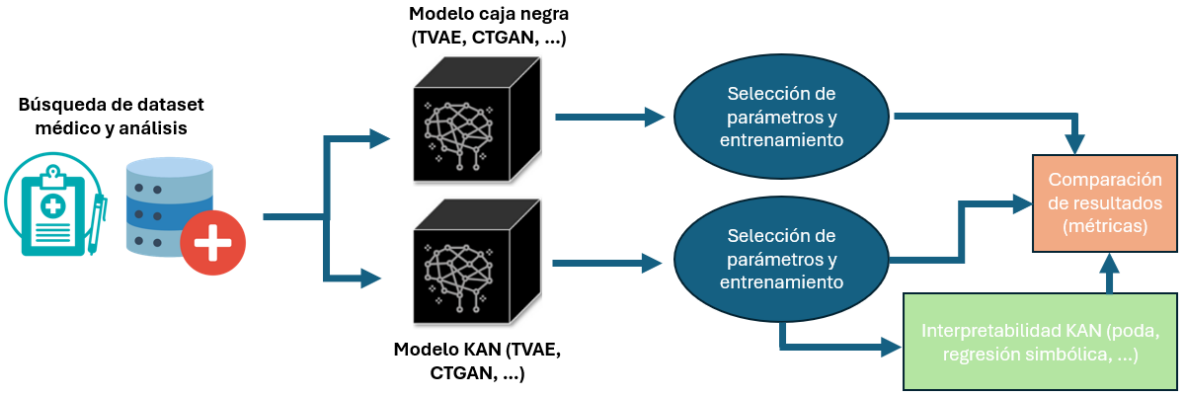


Figura 1.3: Esquema general del proyecto desarrollado.

Finalmente, se llevará a cabo una comparación exhaustiva de los modelos obtenidos empleando diversas métricas, tanto específicas del rendimiento generativo (como fidelidad y diversidad de los datos sintéticos) como del proceso de entrenamiento (pérdida, convergencia, tiempos de ejecución, etc.). El objetivo final es demostrar que las redes KAN permiten una mayor interpretabilidad de los modelos generativos, facilitando una comprensión más clara de las relaciones entre las variables de entrada y los mecanismos internos que conducen a la generación de salidas.

En particular, en el ámbito médico, donde la escasez de datos y la necesidad de preservar la privacidad son limitaciones críticas, este estudio busca mitigar estos desafíos, proporcionando una herramienta para la generación de conjuntos de datos de alta calidad que contribuyan a la investigación médica sin comprometer la privacidad de los pacientes. Además, el proyecto aborda preocupaciones éticas y sociales relacionadas con la transparencia, confianza y equidad en el uso de la inteligencia artificial, promoviendo datos sintéticos con distribuciones justas e inclusivas, y facilitando el cumplimiento de normativas regulatorias.

La memoria de este proyecto se estructura en varios capítulos, comenzando con el Capítulo 2, que recoge el estado del arte y presenta un análisis del contexto actual, así como una revisión de las herramientas disponibles en el ámbito de la interpretabilidad en inteligencia artificial. El Capítulo 3 detalla la metodología seguida durante el estudio, incluyendo el desarrollo del proceso, las pruebas ejecutadas y las estrategias utilizadas para alcanzar los objetivos planteados. El Capítulo 4 expone los resultados obtenidos tras la implementación y evaluación de las Redes Kolmogorov-Arnold, proporcionando un análisis de su desempeño en el contexto de la generación de datos sintéticos. Finalmente, en el Capítulo 5, se presentan las conclusiones extraídas a partir de los resultados, así como las posibles líneas de investigación futuras derivadas del estudio.

Estado del arte

2.1 Modelos interpretativos en Inteligencia Artificial:

La interpretabilidad en modelos de aprendizaje automático, aunque relativamente reciente en el campo de investigación, tiene raíces profundas en la estadística clásica. Sin embargo, con la aparición de modelos de mayor complejidad, se ha ido desplazando esta interpretabilidad a favor del rendimiento. Actualmente, estas técnicas de interpretabilidad se dividen en dos categorías principales: modelos intrínsecamente interpretables y métodos post-hoc [12]. Los primeros incluyen modelos cuya propia estructura facilita su comprensión directa. Entre ellos, destacan:

1. Regresión lineal y logística: La regresión lineal permite interpretar el efecto marginal de cada variable sobre la respuesta [13]. Mientras, la regresión logística transforma la salida lineal en probabilidades mediante una función logística, donde los coeficientes se interpretan en términos de log-odds o odds ratios [14].

2. Modelos lineales y aditivos generalizados (GLM y GAM): Estos modelos permiten salidas no gaussianas mediante funciones de enlace, lo que les permite ampliar su aplicabilidad sin perder interpretabilidad [15]. Por su parte, los modelos aditivos generalizados introducen funciones no paramétricas , permitiendo captar relaciones no lineales [16].

3. Árboles de decisión y RuleFit: Los árboles de decisión segmentan el espacio de entrada de forma jerárquica, permitiendo explicaciones locales y siguiendo el camino de una instancia desde la raíz hasta la hoja [17]. RuleFit combina reglas binarias derivadas de árboles con modelos lineales Lasso, capturando así interacciones interpretables

Por otro lado, los métodos post-hoc agnósticos se aplican tras el entrenamiento del modelo:

4. LIME y SHAP: LIME (Local Interpretable Model-agnostic Explanations) ajusta un modelo interpretable local alrededor de una instancia mediante perturbaciones y ponderaciones por proximidad [18, 19]. SHAP (SHapley Additive exPlanations), se basa en la teoría de juegos para atribuir de forma equitativa la contribución de cada característica a la predicción [20, 21].

5. Mecanismo de atención: Esta técnica permite visualizar cómo un modelo distribuye la importancia entre entradas. En modelos como el Transformer, este mecanismo de self-attention ofrece mapas de atención altamente interpretables [22].

Se observa que los modelos intrínsecamente interpretables sacrifican potencia predictiva por claridad, mientras que los métodos post-hoc permiten explicar modelos potentes a costa de fiabilidad de la propia explicación, punto entorno al cual gira el debate actual sobre la Inteligencia Artificial explicable, en especial con la reciente introducción de normativas éticas y legislación [23]. Como propuesta reciente, surge una prometedora y potente herramienta: las redes Kolmogorov–Arnold (KAN).

2.2 Redes Neuronales Kolmogorov Arnold

Las Redes Neuronales Kolmogorov–Arnold (KAN) son una de las propuestas más recientes y prometedoras en inteligencia artificial interpretable, al combinar eficiencia y transparencia. Se basan en un sólido marco teórico: el teorema de representación de Kolmogorov–Arnold. A continuación, se abordan sus principios matemáticos, la estructura interna de estas redes y su comparación con las MLP tradicionales, destacando su capacidad para generar interpretaciones claras en contextos críticos como el biomédico.

2.2.1 Fundamentos teóricos del teorema de Kolmogorov–Arnold

Las *Kolmogorov–Arnold Networks* (KANs) se fundamentan en el Teorema de Representación de Kolmogorov–Arnold, siendo este una formulación matemática profunda sobre la descomposición funcional de mapeos multivariables inicialmente propuesta por Andrey Kolmogorov en 1957 y posteriormente refinada por Vladimir Arnold en 1958 [1]. Gracias a este teorema se logró ofrecer respuesta a un caso particular del decimotercer problema de Hilbert, demostrándose que cualquier función continua de múltiples variables puede representarse mediante superposiciones finitas de funciones univariadas continuas y sumas aditivas [4].

En este teorema Kolmogorov demuestra que toda función continua $f : [0,1]^n \to \mathbb{R}$ puede expresarse como una superposición de funciones univariadas y la operación binaria de suma:

$$f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \varphi_{q,p}(x_p) \right), \tag{2.1}$$

donde las funciones $\varphi_{q,p} : [0,1] \to \mathbb{R}$ y $\Phi_q : \mathbb{R} \to \mathbb{R}$ son continuas. En esta formulación, la única operación multivariable que encontramos es la suma, pues todas las demás se reducen a combinaciones de funciones univariadas. A pesar del avance y tremenda simplificación para el aprendizaje automático que esto supone, muchas de estas funciones univariadas teóricas pueden ser no suaves o incluso fractales, lo cual puede generar dificultades a la hora de su aprendizaje práctico.Sin embargo, en el mundo real muchas de las funciones relevantes, especialmente en contextos científicos, físicos o biológicos, son suaves y cuentan con una estructura composicional escasa, lo cual nos facilita su representación práctica.

2.2.2 Comparación estructural con redes MLP

En la Figura 2.1, adaptada de [2], encontramos una comparación estructural entre las arquitecturas clásicas Multi-Layer Perceptron (MLP) y las redes Kolmogorov–Arnold (KAN), desde el punto de vista teórico, arquitectónico y funcional.

En una MLP las matrices W_i almacenan los pesos aprendibles, y las funciones σ son las activaciones no lineales fijas:

$$\text{MLP}(x) = (W_3 \circ \sigma \circ W_2 \circ \sigma \circ W_1)(x),$$

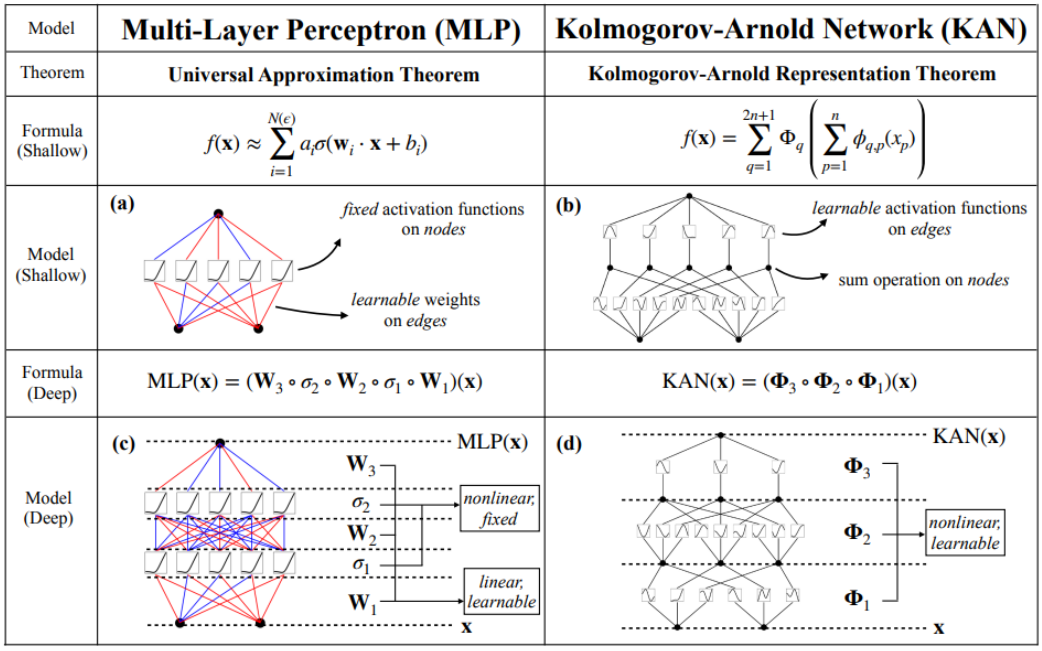


Figura 2.1: Comparación entre MLPs y KANs. Imagen adaptada de [2].

En una red KAN profunda, cada matriz Φ_l contiene funciones $\varphi_{l,j,i}(x)$ que transforman localmente cada entrada con una spline ajustable:

$$\mathrm{KAN}(x) = (\Phi_3 \circ \Phi_2 \circ \Phi_1)(x),$$

2.2.3 Arquitectura redes neuronales KAN:

A diferencia de los MLPs, en los hemos visto que las funciones de activación son fijas y se aplican en los nodos, y donde las conexiones se definen mediante pesos lineales, las redes KANs van a transladar la no linealidad a los bordes mediante funciones univariadas aprendibles, las cuales van a acturae como pesos funcionales entre nodos.

Estas redes cuentan con capas completamente conectadas, al igual que en un MLP, pero en su caso cada capa se compone por una matriz de funciones univariadas, en lugar de una matriz de pesos lineales. Estas funciones se parametrizan como combinaciones de bases B-spline y se entrenan durante el proceso de optimización. De esta forma, cada conexión entre un nodo de entrada y uno de salida se modela mediante una función $\varphi(x)$ de la forma:

$$\varphi(x) = w_b\, b(x) + w_s \sum_i c_i B_i(x), \tag{2.2}$$

donde $b(x)$ es una función base fija , $B_i(x)$ representa las bases B-spline locales, c_i son los coeficientes aprendibles, y w_b, w_s son factores escalares que nos permiten controlar la contribución relativa de cada término.

Si nos centramos en el funcionamiento interno de una capa KAN, tenemos que dado un vector de entrada $\mathbf{x}_l \in \mathbb{R}^{n_l}$ en la capa l, la salida de la siguiente capa vendrá dada por:

$$x_{l+1,j} = \sum_{i=1}^{n_l} \varphi_{l,j,i}(x_{l,i}), \tag{2.3}$$

donde $\varphi_{l,j,i}$ denota la función univariada específica que conecta el nodo i de la capa l con el nodo j de la capa $l+1$. Esta operación puede expresarse en forma matricial utilizando una matriz de funciones Φ_l, donde cada entrada corresponde a una función $\varphi_{l,j,i}(\cdot)$ aplicada a la activación correspondiente. La salida final de la red se obtiene como la composición de estas capas funcionales:

$$\mathrm{KAN}(\mathbf{x}) = (\Phi_{L-1} \circ \cdots \circ \Phi_1 \circ \Phi_0)(\mathbf{x}), \tag{2.4}$$

siendo L el número total de capas. Esta construcción es completamente diferenciable, lo cual va a ser realmente útil al permitirnos entrenar KANs mediante retropropagación estándar. A su vez, gracias al uso de estas funciones univariadas explícitas, conseguimos la interpretación simbólica de sus componentes y la extracción de estructuras funcionales.

La Figura 2.2 resume gráficamente la implementación funcional del teorema de Kolmogorov–Arnold sobre estas redes, a la izquierda se representan las activaciones univariadas $\varphi_{l,j,i}$ aplicadas sobre cada borde del grafo, mientras que a la derecha nos encontramos con una de estas funciones de activación parametrizada como combinaciones de *B-splines* con distinta granularidad de malla, lo cual permitirá adaptar la precisión de la transformación local.

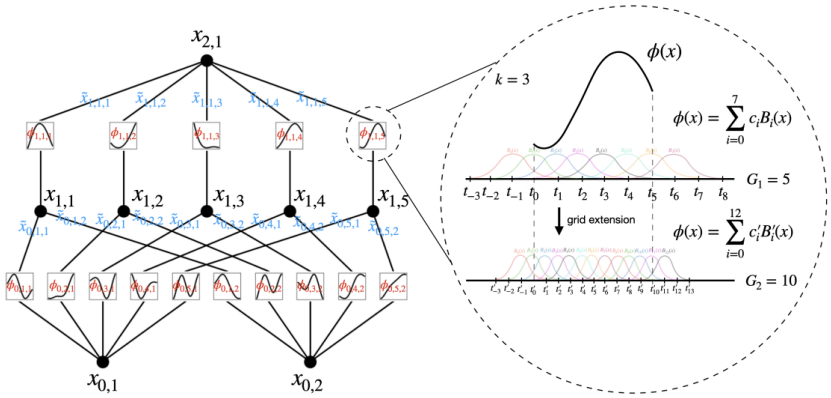


Figura 2.2: Esquema de una red KAN con activaciones definidas sobre las conexiones [2].

Formulación matricial y capas profundas

En la versión inicial de las redes KAN, el teorema de Kolmogorov-Arnold se traduce en una red de dos capas con estructura $[n, 2n+1, 1]$, donde tanto las funciones internas $\varphi_{q,p}$ como las externas Φ_q son interpretadas como capas funcionales [4].

Sin embargo, una ventaja clave reciente de las KANs es la posibilidad de generalizar esta arquitectura a redes más profundas a través del acople secuencial de estas capas, de forma similar a como se construyen actualmente las MLPs.

Si definimos una red KAN de profundidad L, esta va a estar compuesta por una secuencia de capas $\Phi^{(0)}, \ldots, \Phi^{(L-1)}$, donde cada capa va a ser una matriz de funciones univariadas entrenables:

$$\Phi^{(l)} = \left[\varphi_{j,i}^{(l)}\right]_{j=1,\ldots,n_{l+1}}^{i=1,\ldots,n_l}, \tag{2.5}$$

donde n_l y n_{l+1} indicarán el número de nodos en las capas l y $l+1$. Cada $\varphi_{j,i}^{(l)} : \mathbb{R} \to \mathbb{R}$ se modelará como una combinación de activaciones suaves y funciones base B-spline:

$$\varphi_{j,i}^{(l)}(x) = w_b \cdot \text{silu}(x) + w_s \cdot \sum_k c_{j,i,k}^{(l)} B_k(x), \tag{2.6}$$

donde $B_k(x)$ van a ser esas funciones B-spline localizadas, $c_{j,i,k}^{(l)}$ serán coeficientes aprendibles, y w_b, w_s pesos ajustables encargados de controlar la contribución de cada componente [5].

La activación del nodo j en la capa $l+1$ la vamos a obtener sumando las transformaciones univariadas de cada entrada:

$$x_j^{(l+1)} = \sum_{i=1}^{n_l} \varphi_{j,i}^{(l)}(x_i^{(l)}), \qquad j = 1, \ldots, n_{l+1}, \tag{2.7}$$

lo que puede expresarse de una forma más compacta como:

$$\mathbf{x}^{(l+1)} = \Phi^{(l)}(\mathbf{x}^{(l)}), \tag{2.8}$$

Esta matriz funcional $\Phi^{(l)}$ operará elemento a elemento sobre el vector de activaciones anterior.

Por lo tanto, la salida final de la red KAN completa se obtendría por composición funcional:

$$\text{KAN}(\mathbf{x}) = (\Phi^{(L-1)} \circ \cdots \circ \Phi^{(0)})(\mathbf{x}), \tag{2.9}$$

Todas las operaciones involucradas son diferenciables respecto a sus parámetros, lo cual es fundamental tanto para la retropropagación mediante descenso de gradiente, que requiere que todas las funciones usadas en la red cuenten con derivadas bien definidas, como para la actualización de parámetros internos de las funciones $\varphi_{j,i}^{(l)}$.

A modo de comparación, una MLP intercalaría matrices de pesos fijos W_l con funciones de activación uniformes σ:

$$\text{MLP}(x) = (W_{L-1} \circ \sigma \circ \cdots \circ W_0)(x),$$

mientras que una KAN integra la transformación y la activación en una única matriz funcional en cada capa:

$$\text{KAN}(x) = (\Phi_{L-1} \circ \cdots \circ \Phi_0)(x).$$

La formulación explícita de las matrices funcionales permite representar cada capa como:

$$\Phi^{(l)} = \begin{bmatrix} \varphi_{1,1}^{(l)} & \varphi_{1,2}^{(l)} & \cdots & \varphi_{1,n_l}^{(l)} \\ \varphi_{2,1}^{(l)} & \varphi_{2,2}^{(l)} & \cdots & \varphi_{2,n_l}^{(l)} \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_{n_{l+1},1}^{(l)} & \varphi_{n_{l+1},2}^{(l)} & \cdots & \varphi_{n_{l+1},n_l}^{(l)} \end{bmatrix}. \tag{2.10}$$

Finalmente, el caso clásico asociado al teorema de Kolmogorov–Arnold se recupera como una instancia particular con arquitectura $[n, 2n+1, 1]$:

$$f(x) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \varphi_{q,p}(x_p) \right), \tag{2.11}$$

donde las funciones $\varphi_{q,p}$ y Φ_q corresponden a las funciones internas y externas del teorema original [1].

A través de la estructura de estas redes vamos a conseguir representar funciones altamente complejas con un número de parámetros mucho menor, especialmente gracias al aprovechamiento simultáneo de grados de libertad estructurales (número de nodos por capa y tipos de conexión) e internos (forma específica de las funciones), lo que ha demostrado ventajas tanto en interpretabilidad como en precisión en tareas científicas de diversos estudios [1, 2, 5].

Modelado de funciones, complejidad paramétrica y regularización estructural

Esta granularidad que hemos mencionado se va a poder refinar progresivamente mediante un proceso conocido como *grid extension*, el cual consiste en ajustar una nueva spline fina sobre una spline previamente entrenada en una malla más gruesa. Suponiendo el caso en que queremos aproximar una función univariada $f(x)$ en un intervalo acotado $[a, b]$, una representación inicial con G_1 subintervalos generaría una base de funciones $B_i(x)$, mientras que una malla refinada con $G_2 > G_1$ produciría una nueva base $B'_j(x)$. Esta función refinada se ajustaría de forma que se minimizase la distancia entre ambas:

$$\{c'_j\} = \arg\min_{\{c'_j\}} \mathbb{E}_{x\sim p(x)} \left(\sum_{j=0}^{G_2+k-1} c'_j B'_j(x) - \sum_{i=0}^{G_1+k-1} c_i B_i(x) \right)^2, \tag{2.12}$$

Esto último puede resolverse mediante mínimos cuadrados, aplicandose de forma independiente a cada spline de la red. De esta forma logramos una mejora localizada y controlada de la precisión sin alterar la estructura funcional global de nuestra red[2].

Para calcular el número de parámetros total en una red KAN nos apoyaremos en tres parámetros clave ya mencionados: la profundidad L, el número de nodos por capa $\{n_l\}$, y la

granularidad K del grid spline. Si cada función univariada $\varphi_{j,i}^{(l)}$ se parametriza mediante K coeficientes spline más dos pesos residuales (w_b, w_s), entonces podemos llevar a cabo la estimación del número de parámetros de nuestra red KAN:

$$\text{Parámetros totales} \approx \sum_{l=0}^{L-1} n_l \cdot n_{l+1} \cdot (K+2).$$

Con esto podemos escalar el modelo con gran felxibilidad, pues con incrementos en K podemos aumentar la capacidad local sin alterar la topología y con modificaciones en la arquitectura $[n_0, \ldots, n_L]$ vamos a poder ajustar la complejidad global.

Otra de las propiedades fundamentales de las KANs es su compatibilidad con técnicas de regularización estructural, especialmente la inducida por sparsity. Gracias a su base de funciones univariadas parametrizadas se puede aplicar una regularización mediante norma L_1, que se va a encargar de forzar durante el entrenamiento a que muchas conexiones se anulen o se simplifiquen a través de penalizaciones sucesivas. Gracias a este paso se reduce la complejidad de nuestra red y se mejora su interpretabilidad, identificando más facilmente aquellas variables y funciones con mayor influencia sobre la predicción [2, 5].

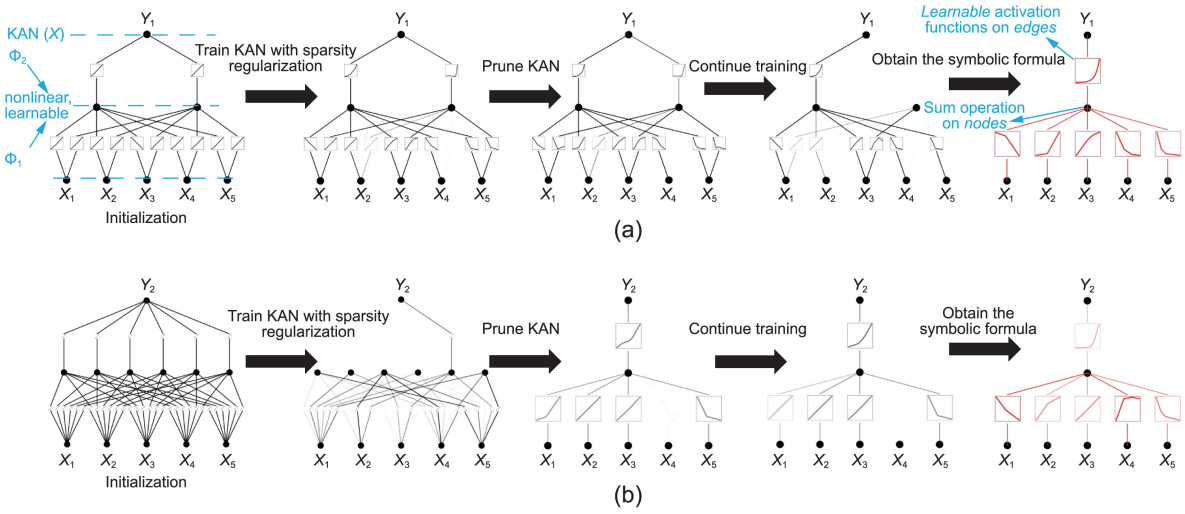


Figura 2.3: Esquema del proceso de entrenamiento de una red KAN con sparsity [2].

Desarrollo

3.1 Herramientas y datasets del proyecto

En esta sección se describe las herramientas principales y conjuntos de datos empleados para la realización del proyecto. En primer lugar, se destaca el uso de la librería Pykan, que implementa la base para el uso de las redes KAN mediante Python. La principal razón de presentar esta librería es que se ha modificado internamente en gran medida para poder obtener las funcionalidades requeridas para el proyecto, lo que ha supuesto la mayor contribución de este trabajo. En segundo lugar se detallará el conjunto de datos empleado para la realización de los experimentos propuestos que se detallarán más adelante.

3.1.1 Pykan

Para desarrollar este proyecto se ha recurrido al repositorio oficial de GitHub asociado al artículo *KAN: Kolmogorov–Arnold Networks* [2], disponible en `https://github.com/KindXiaoming/pykan`. Este repositorio ha permitido llevar a cabo una implementación completa y flexible de las redes KAN, permitiendo desplegar en la práctica las ideas teóricas previamente analizadas pues provee de la mayor parte de funcionalidades para hacer uso de este tipo de redes mediante la librería Pytorch para diferenciación automática. Durante el desarrollo se presentarán las modificaciones principales incorporadas para los propósitos y objetivos perseguidos en el trabajo. La herramienta y código se organiza en torno a tres componentes principales:

- `KANLayer`: Se implementa una capa básica de KAN, en la cual cada conexión entre nodos contiene una función univariada parametrizada mediante *B-splines*.
- `MultKAN`: Se presenta como una extensión modular que permite definir arquitecturas profundas (multicapa) de KAN, con formas arbitrarias como $[n, h_1, h_2, ..., 1]$. Esta capa va a gestionar la composición jerárquica de las distintas capas $\Phi^{(l)}$(2.3)–(2.4).
- `SymbolicKAN`: Esta capa permite la extracción simbólica de fórmulas mediante la implementación de técnicas como "sparsification", "pruning" y "symbolification". Con su integración se logrará fijar funciones de activación a expresiones de mayor simplicidad y ajustar los parámetros afines para aproximar transformaciones simbólicas explícitas. A su vez, el registro de los parámetros generados por este modulo permitirá visualizar las activaciones para facilitar el análisis interpretativo [2].

Para el análisis simbólico `Pykan` incluye una interfaz flexible, siendo sus funcionalidades más destacadas las siguientes (extraídas de la Tabla 6 del artículo original [2]):

Flujo de simplificación simbólica en Pykan

El proceso de simplificación de la red KAN y de su respectiva representación viene descrito en la siguiente figura 3.1, adaptada de [2]. Este flujo propuesto y simplificado para casos sencillos cuenta con los siguientes pasos:

Tabla 3.1: Funcionalidades disponibles en `pykan`.

Función	Descripción
`model.train(dataset)`	Entrena el modelo sobre un dataset dado.
`model.plot()`	Visualiza gráficamente la arquitectura y las funciones activas.
`model.prune()`	Elimina automáticamente nodos y conexiones irrelevantes.
`model.fix_symbolic(l,i,j,fun)`	Fija $\varphi_{l,i,j}$ a una función simbólica especificada `fun`.
`model.suggest_symbolic(l,i,j)`	Sugiere funciones simbólicas candidatas para $\varphi_{l,i,j}$.
`model.auto_symbolic()`	Aplica automáticamente la mejor sugerencia simbólica en toda la red.
`model.symbolic_formula()`	Devuelve la fórmula simbólica completa generada por la red.

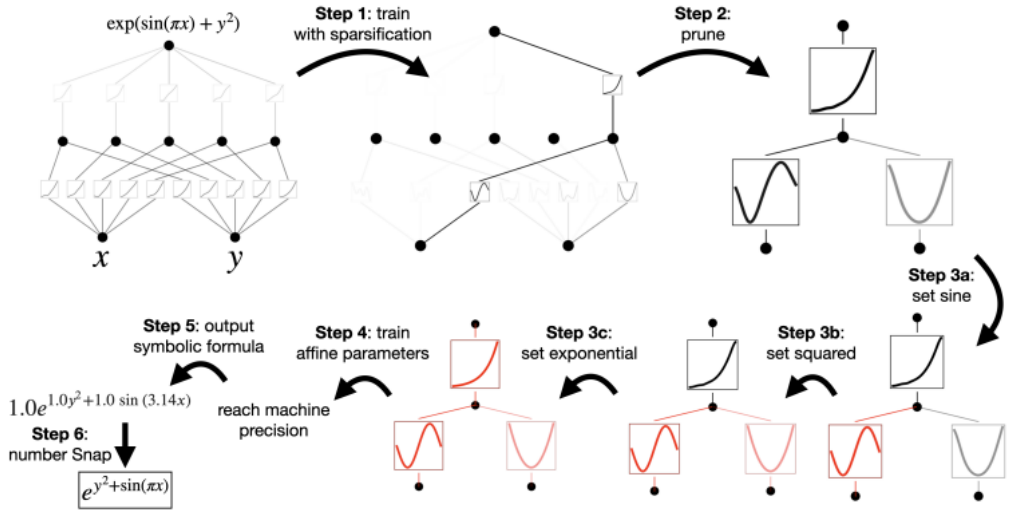


Figura 3.1: Proceso interactivo de simplificación y simbolificación en KANs [2].

1. **Entrenamiento con regularización por sparsity:** Entrenamos nuestra red KAN con penalizaciones L_1 y entropía sobre las funciones $\varphi_{j,i}$, forzando a que muchas de las activaciones pierdan relevancia.

2. **Poda automática (pruning):** Eliminamos los nodos que no tienen entradas ni salidas relevantes.

3. **Fijación simbólica (`fix_symbolic`):** Una vez identificadas las funciones significativas, se asignan formas simbólicas específicas a las activaciones, como $\sin(x)$, x^2 o $\exp(x)$, utilizando datos de entrada y salida para ajustar los parámetros afines.

4. **Ajuste fino (fine-tuning):** Se refinan los parámetros de escala, desplazamiento y salida para que nuestra red represente la transformación simbólica con máxima precisión.

5. **Extracción simbólica final:** A partir de la función `symbolic_formula()`, obtenemos una representación explícita de la función aprendida, expresada de forma algebraica.

3.1.2 Funcionalidades KAN y MultKAN de Pykan: Problemas y desafíos

La librería `Pykan` descrita en el apartado anterior es una gran fuente con la que comenzar a realizar desarrollos mediante el uso de redes KAN. Una de las funcionalidades más útiles es la clase `MultKAN` que extiende la `KAN` para manejar directamente funciones multivariadas, introduciendo una combinación de múltiples entradas en cada unidad de activación. Esto mejora la capacidad de representación del modelo, permitiendo capturar interacciones más complejas entre variables.

El análisis de las definiciones proporcionadas por la librería permite identificar varios aspectos clave que influyen de manera significativa en la posibilidad de utilizar directamente estas funcionalidades.

- **Definición de arquitectura `KAN`:** Por defecto, las capas `KAN` y `MultKAN` permiten definir únicamente arquitecturas secuenciales simples, basadas en una sucesión directa de capas. Esto limita considerablemente su integración en arquitecturas más complejas o no lineales, como las que requieren ramificaciones, múltiples entradas/salidas o estructuras condicionales.
- **Limitación del método `fit()`:** El entrenamiento de redes `MultKAN` se realiza exclusivamente mediante un método predefinido denominado `fit()`, que internamente invoca `forward()` y, si corresponde, `attribute()`. Este enfoque impide utilizar directamente las capacidades de `MultKAN` en arquitecturas definidas externamente (como las propuestas en el trabajo), donde el entrenamiento se controla desde otro bucle de optimización personalizado.
- **Dependencia crítica del método `forward()`:** Las funcionalidades internas de `MultKAN` dependen de que se haya ejecutado previamente el método `forward()`, ya que este almacena internamente activaciones y parámetros clave de cada capa. En ausencia de esta ejecución, no es posible acceder a herramientas derivadas como la poda (*pruning*), la extracción simbólica o la visualización de funciones (*plot*).
- **Requisito de activación explícita de `attribute()`:** Si se emplean métricas de regularización estructural como `edge_backward` o `node_backward`, la red necesita calcular las contribuciones de cada entrada a los nodos o salidas. Este cálculo se realiza mediante `attribute()`, el cual también requiere que las activaciones internas hayan sido correctamente registradas durante el paso `forward()`. De no cumplirse esta condición, la red no puede evaluar ni aplicar la regularización esperada.
- **Necesidad de modificación del código fuente para arquitecturas externas:** Para poder utilizar `MultKAN` dentro de arquitecturas generativas externas como las propuestas en el trabajo, ha sido necesario replicar manualmente el comportamiento del método `forward()` dentro del ciclo de entrenamiento externo. Esto incluye forzar el almacenamiento de activaciones y atributos intermedios, de manera que la red crea que ha sido entrenada mediante su propio método `fit()`, permitiendo así el uso posterior de sus herramientas específicas.

El uso de `MultKAN` en arquitecturas avanzadas requiere una modificación no trivial del flujo de entrenamiento, así como una gestión manual del almacenamiento de parámetros y activaciones. Esto contradice el principio de modularidad esperado en arquitecturas personalizadas y puede limitar la interoperabilidad con frameworks de mayor nivel. Este apartado tiene como objetivo poner de relieve la importancia del desarrollo y la programación integral del proyecto, ya que ha constituido la base fundamental para posibilitar el entrenamiento, la evaluación y la implementación de aplicaciones avanzadas basadas en redes KAN. Asimismo, se proporciona el código completo del proyecto con el fin de facilitar un análisis detallado de las implementaciones realizadas y de las implicaciones asociadas a cada una de las decisiones adoptadas durante el proceso de desarrollo.

`https://gitlab.com/GATV/tfg_paula`

3.1.3 Dataset: Heart Prediction Dataset (Quantum)

Para este proyecto se ha usado el dataset público `Heart Prediction Dataset (Quantum)`, disponible en Kaggle[1],compuesto por 500 registros con 7 variables y diseñado para el diagnóstico de enfermedades cardíacas. Las variables disponibles son las siguientes:

Tabla 3.2: Descripción de variables `Heart Prediction Dataset (Quantum)`.

Variable	Tipo	Descripción
`Age`	Numérica continua	Edad del paciente en años.
`Gender`	Categórica binaria	Género del paciente (0 = mujer, 1 = hombre).
`BloodPressure`	Numérica continua	Nivel de presión sanguínea registrado.
`Cholesterol`	Numérica continua	Nivel de colesterol del paciente.
`HeartRate`	Numérica continua	Frecuencia cardíaca en latidos por minuto.
`QuantumPatternFeature`	Numérica continua	Atributo derivado que captura patrones complejos no lineales.
`HeartDisease`	Binaria (objetivo)	0 = Sin enfermedad, 1 = Presencia de enfermedad cardíaca.

El campo adicional `QuantumPatternFeature` representa una proyección matemática de las variables de entrada que busca capturar relaciones intrínsecas complejas no triviales, como interacciones no lineales o estructuras de alta dimensión.

Análisis exploratorio de las variables aplicadas al proyecto:

En la Figura 3.2 (Izquierda) se observa la distribución de cada variable. Destacan `Age`, `Cholesterol` y `HeartRate`, que parecen relativamente uniformes dentro de ciertos rangos, sin grandes asimetrías. La Figura 3.2 (Derecha) refleja la matriz de dispersión entre pares de variables, que evidencia la ausencia de separaciones lineales claras entre las clases, justificando la necesidad de modelos capaces de capturar relaciones no lineales complejas.

Por otro lado, la matriz de correlación extraída 3.3 nos permite destectar una correlación negativa moderada entre `Cholesterol` y `HeartDisease` (-0.42), lo que nos permite deducir que niveles elevados de colesterol están inversamente relacionados con el diagnóstico positivo,

[1] `https://www.kaggle.com/datasets/shantanugarg274/heart-prediction-dataset-quantum`

un resultado contra intuitivo que podría estar sesgado por factores externos o características específicas del muestreo. También se observa una correlación positiva leve entre la edad y la presencia de enfermedad cardíaca (`Age` vs `HeartDisease`, 0.26), lo que sí concuerda con el conocimiento clínico previo.

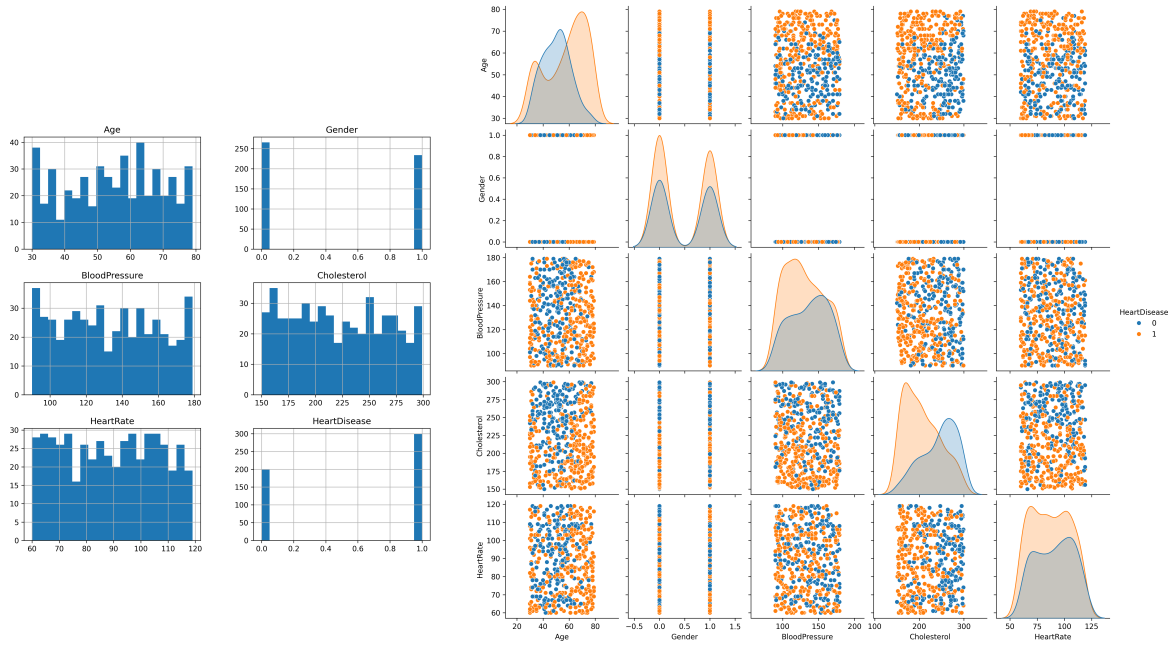


Figura 3.2: Distribución de frecuencias para las variables del dataset (Izquierda). Matriz de dispersión y densidades por tipo de diagnóstico (Derecha).

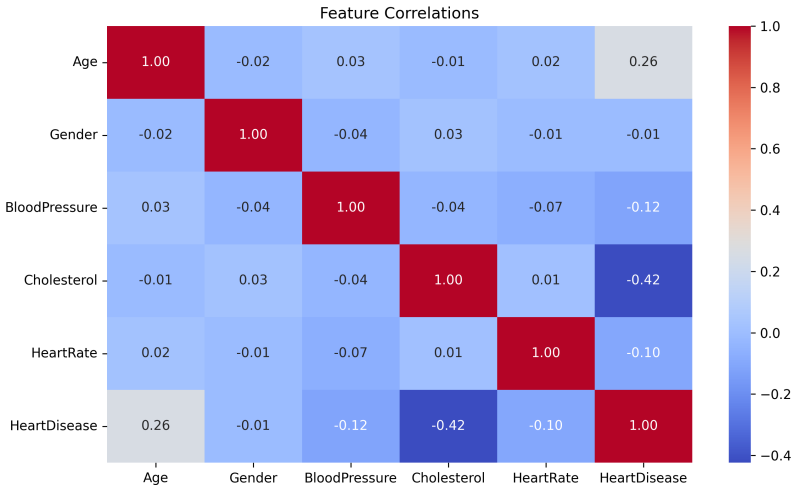


Figura 3.3: Mapa de correlación entre variables del dataset clínico original.

3.2 MLP frente a KAN en tarea de clasificación binaria

Diversos estudios han evidenciado que las redes KAN pueden superar a las MLP en tareas que involucran funciones no lineales, datos tabulares complejos, series temporales o sistemas dinámicos. Aunque, por norma general, presentan un mayor coste computacional, destacan por su capacidad de generalización, precisión simbólica y eficiencia en representación cuando se dispone de datos suficientes [2]. Para ilustrar esto se propone un ejemplo simplificado en una tarea de clasificación con el dataset seleccionado, comparando los resultados obtenidos a partir de una red neuronal MLP y una red KAN. Ambos modelos han sido entrenados y evaluados bajo las mismas condiciones para poder garantizar una comparación justa.

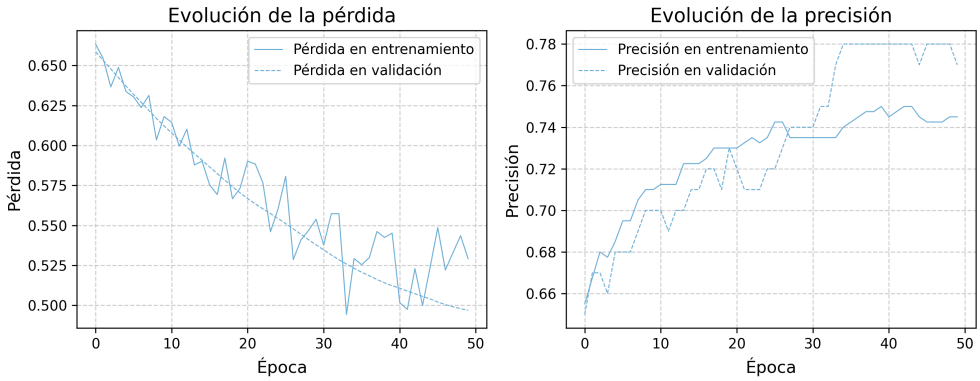


Figura 3.4: Evolución de la pérdida (loss) y precisión (accuracy) del modelo MLP.

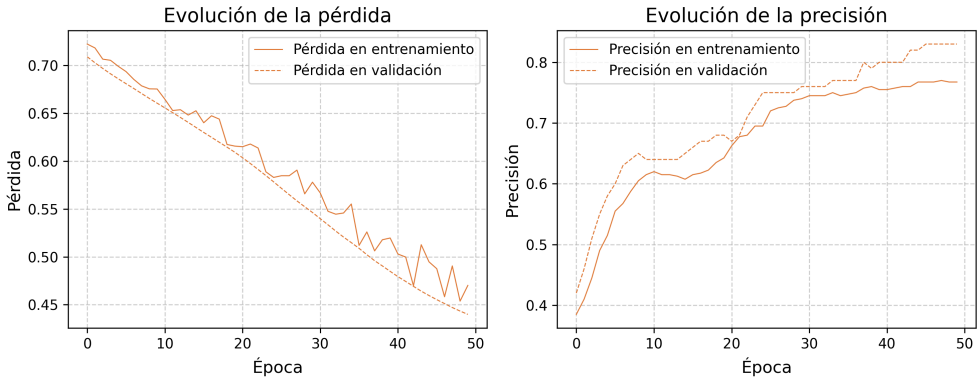


Figura 3.5: Evolución de la pérdida (loss) y precisión (accuracy) del modelo KAN.

La red MLP empleada está compuesta por una capa de entrada de tamaño 5, seguida de una capa oculta con 32 neuronas y activación ReLU, y una capa de salida con una única neurona y activación sigmoide. La arquitectura utilizada para la red KAN replica esta misma estructura funcional, sustituyendo las capas lineales por dos capas `KANLayer`: la primera transforma de 5 a 32 dimensiones y la segunda de 32 a 1. Por último, para generar a la salida un resultado interpretable como probabilidad binaria, se aplica igualmente una activación sigmoide. Ambos modelos han sido entrenados utilizando el optimizador Adam con una tasa de aprendizaje de

0,001, un número total de 50 épocas y un tamaño de batch de 128. Como función de pérdida se ha empleado la *binary cross-entropy* (BCE). Tras el entrenamiento se presentan las curvas de evolución de la pérdida (loss) y la precisión (accuracy) tanto en entrenamiento como en validación, así como diversas métricas de evaluación.

Métricas	**MLP**	**KAN**
Accuracy	0.7700	**0.8300**
Precision (global)	0.7761	**0.8209**
Recall (global)	0.8667	**0.9167**
F1 Score (global)	0.8189	**0.8661**
ROC AUC	0.8412	**0.8842**
PR AUC	0.8913	**0.9218**
Matthews CorrCoef	0.5123	**0.6425**
Precision (Macro avg)	0.7668	**0.8347**
Recall (Macro avg)	0.7458	**0.8083**
F1-score (Macro avg)	0.7519	**0.8166**
Precision (Weighted avg)	0.7687	**0.8319**
Recall (Weighted avg)	0.7700	**0.8300**
F1-score (Weighted avg)	0.7653	**0.8265**

Tabla 3.3: Métricas de evaluación entre MLP y KAN en problema de clasificación binaria.

Ambos modelos muestran una disminución sostenida de la pérdida y una mejora en precisión (Figuras 3.4 y 3.5). KAN alcanza un rendimiento superior, con precisión de validación por encima del 82 %, frente al 78 % de MLP. Las métricas macro representan el promedio simple entre clases, mientras que las ponderadas ajustan el peso por frecuencia de clase (Tabla 3.3). En ambos casos, KAN presenta mejores resultados, indicando mayor robustez y equilibrio entre clases.

Los resultados analizados en su conjunto determinan que KAN supera en esta tarea al modelo MLP de forma consistente en todas las métricas evaluadas, pudiendo confirmar empíricamente en este caso la superioridad estructural y funcional de estas redes frente a arquitecturas clásicas como el MLP. Con esto se demuestra que para una tarea de clasificación el uso de redes KAN puede superar a los MLP a lo que se añade la mejora en la interpretabilidad interna de la red. El reto de este trabajo es comprobar que sucede en casos de problemas generativos por lo que se verá a continuación la aplicación de estas redes a estos problemas y un análisis exhaustivo de la interpretabilidad.

3.3 Aplicación de redes KAN a la generación de datos tabulares

Una vez demostrada la eficacia de las redes KAN en tareas relativamente menos complejas, se va a analizar su potencial en un contexto más exigente: la generación de datos sintéticos tabulares. Esta tarea consiste en aproximar la distribución conjunta de un conjunto de datos reales, con el objetivo de generar muestras nuevas que conserven tanto las propiedades estadísticas originales como su utilidad en tareas analíticas posteriores. En este trabajo se van a emplear los modelos *Tabular Variational Autoencoder* (TVAE) y *Conditional Tabular Generative Adversarial Network* (CTGAN), especialmente adaptados al formato tabular.

3.3.1 Tabular Variational Autoencoder (TVAE)

Se ha tomado como punto de partida la implementación del repositorio oficial de SDV [7], a la cual se le han realizado algunas modificaciones que se detallan a continuación. En la arquitectura original de TVAE (Figura 3.6), el codificador toma como entrada los datos transformados y los procesa mediante una red neuronal *feedforward* definida por una secuencia de capas `Linear` y `ReLU`. A partir de la salida final se generan dos vectores: la media $\boldsymbol{\mu}$ y el logaritmo de la varianza $\log \boldsymbol{\sigma}^2$, que definen una distribución normal multivariante sobre el espacio latente. La muestra latente $\mathbf{z}$ se obtiene mediante reparametrización:

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \quad \text{con } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \boldsymbol{\sigma} = \exp\left(0{,}5 \cdot \log \boldsymbol{\sigma}^2\right) \tag{3.1}$$

El decodificador recibe la muestra latente $\mathbf{z}$ y la transforma de vuelta al espacio original de los datos, utilizando una red *feedforward* inversa, que proyecta desde el espacio latente hacia el espacio de entrada original, con una estructura simétrica al codificador. La última capa es lineal y produce una reconstrucción del vector original. Se aprende también un parámetro $\boldsymbol{\sigma}_{\text{recon}}$ por dimensión, que representa la desviación estándar asociada a cada variable continua reconstruida.

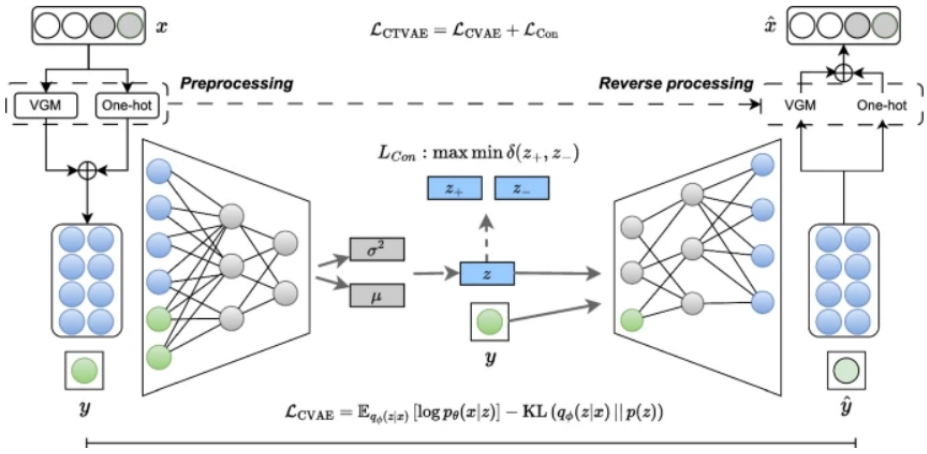


Figura 3.6: Arquitectura general del modelo Tabular VAE [6].

Los datos de entrada al codificador son transformados mediante el módulo `DataTransformer` de TVAE, que ha sido modificado con el objetivo de mejorar la trazabilidad entre las variables transformadas y los atributos originales. Originalmente, este módulo aplicaba un `ClusterBasedNormalizer` basado en modelos Gaussian Mixture, donde cada variable continua se representaba mediante dos partes: un valor normalizado y una codificación `softmax` de componentes latentes. Esto complicaba la interpretación directa de las variables, aumentaba la dimensión de entrada y rompía la correspondencia uno a uno con los atributos originales.

Para simplificar esta transformación y facilitar su interpretación, se ha implementado una versión propia del `DataTransformer`. En ella, las variables continuas se transforman usando un `SimpleNormalizer`, que aplica una estandarización clásica (media cero y desviación típica

uno), seguida de una activación tipo `tanh`, de forma que cada variable continua se representa con una única dimensión. Por otro lado, las variables categóricas se codifican con *one-hot encoding* y activación `softmax`, manteniendo una representación clara y directa. De esta forma cada dimensión transformada puede relacionarse de forma única con su atributo original (Tabla 3.4), lo que va a ser realmente útil para el análisis del modelo.

Tabla 3.4: Correspondencia entre variables transformadas y atributos originales

Variable transformada	Atributo original
x_1	Age
x_2	Gender_0 (Masculino)
x_3	Gender_1 (Femenino)
x_4	BloodPressure
x_5	Cholesterol
x_6	HeartRate
x_7	QuantumPatternFeature
x_8	HeartDisease_0 (sin enfermedad cardíaca)
x_9	HeartDisease_1 (con enfermedad cardíaca)

La función de pérdida aplicada va a combinar la reconstrucción (error cuadrático para variables continuas y entropía cruzada para discretas) y la divergencia de Kullback-Leibler, que va a actuar como regularizador sobre el espacio latente, minimizando la distancia entre la distribución posterior aproximada generada por el encoder, caracterizada por los vectores de media $\boldsymbol{\mu}$ y desviación estándar $\boldsymbol{\sigma}$, y la distribución normal estándar $\mathcal{N}(\mathbf{0}, \mathbf{I})$:

$$\mathcal{D}_{\mathrm{KL}} = -\frac{1}{2} \sum_{i=1}^{d} \left(1 + \log \sigma_i^2 - \mu_i^2 - \sigma_i^2\right) \tag{3.2}$$

donde d es la dimensión del espacio latente. Por otro lado, la pérdida de reconstrucción $\mathcal{L}_{\mathrm{recon}}$ se obtiene con la suma de dos términos, para las variables continuas x_i se aplica una pérdida cuadrática ponderada por una varianza aprendida σ_i^2 junto con un término de regularización logarítmico, mientras que para las variables discretas x_j, se utiliza la entropía cruzada entre las verdaderas etiquetas y las probabilidades predichas. Formalmente, se expresa como:

$$\mathcal{L}_{\mathrm{recon}} = \sum_{i \in \mathcal{C}} \left(\frac{(x_i - \tanh(\hat{x}_i))^2}{2\sigma_i^2} + \log \sigma_i \right) + \sum_{j \in \mathcal{D}} \mathrm{CE}(x_j, \hat{x}_j) \tag{3.3}$$

donde $\mathcal{C}$ es el conjunto de variables continuas, $\mathcal{D}$ el conjunto de variables discretas, y CE la función de entropía cruzada. La pérdida total del modelo se expresa entonces como:

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{recon}} \cdot \texttt{factor} + \mathcal{D}_{\mathrm{KL}} \tag{3.4}$$

donde `factor` va a ser el hiperparámetro que ajusta el peso relativo de la pérdida de reconstrucción frente a la divergencia KL.

El modelo se entrena durante un número fijo de épocas, con optimizador Adam y regularización L2 y ese se almacena con el objetivo de poder ser utilizado para la generación de nuevos datos sintéticos a través del muestreo de vectores latentes $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ que se introducen en el decodificador para generar nuevas muestras sintéticas. A las salidas continuas se les aplica nuevamente la función `tanh`, con el fin de mantenerlas dentro del rango deseado. Finalmente, con la transformación inversa del `DataTransformer`, se devuelven los datos a su dominio original, respetando tanto las codificaciones categóricas como las escalas originales.

3.3.2 Implementación y entrenamiento TVAE MultKAN

Para esta implementación se reemplazan las capas de neuronas totalmente conectadas originales presentes tanto en el codificador como en el decodificador por una red `MultKAN` en cada caso, con el objetivo de poder modelar no linealidades complejas con una estructura más interpretable y regularizable, manteniendo el esquema del autoencoder variacional.

En la Figura 3.7 se presenta la arquitectura seleccionada para la red. Cada capa de la red `MultKAN` se configura a partir de los valores del array `compress_dims`, que definen el número de nodos por capa oculta, y `embedding_dim`, que determina el tamaño del espacio latente. Por otro lado, el valor de `data_dim` define el tamaño de los datos de entrada y salida del modelo, que coincide con el de los datos originales, generando una topología perfectamente simétrica centrada en el espacio latente.

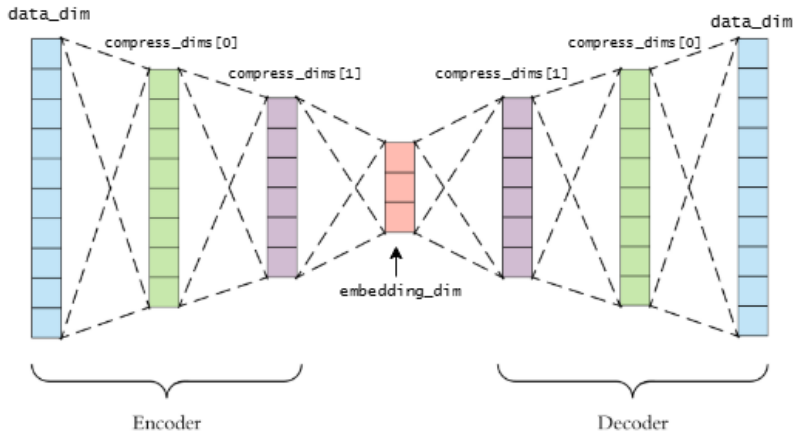


Figura 3.7: Arquitectura red neuronal TVAE seleccionada para el proyecto.

La integración de la red `MultKAN` en el modelo generativo TVAE ha requerido una modificación significativa del flujo estándar de entrenamiento, debido a que este modelo aplica una lógica personalizada que difiere del entrenamiento nativo de `MultKAN`. En particular, el modelo TVAE calcula explícitamente variables intermedias fundamentales como la media $\boldsymbol{\mu}$, la varianza logarítmica $\log \boldsymbol{\sigma}^2$ y la muestra latente $\mathbf{z}$, lo cual impide el uso directo del método `fit()` de `MultKAN`, diseñado para arquitecturas más rígidas y secuenciales.

Para superar esta limitación, se ha replicado de forma manual y controlada la lógica de propagación hacia adelante (forward pass) de `MultKAN`, ejecutando cada capa de forma explícita dentro del ciclo de entrenamiento del TVAE. Esta estrategia ha permitido capturar y almacenar todas las activaciones internas necesarias durante el entrenamiento, sin necesidad de modificar la arquitectura general del modelo generativo. Gracias a este enfoque, ha sido posible compatibilizar la capacidad expresiva y explicativa de las redes KAN con la flexibilidad estructural y funcional requerida por TVAE.

Una vez obtenidas las activaciones y parámetros intermedios durante la propagación hacia adelante, estos se han asignado manualmente como atributos internos del modelo `MultKAN`, simulando el comportamiento habitual de su método interno. Esta intervención ha resultado imprescindible para habilitar el uso de funcionalidades avanzadas que dependen de dichos parámetros, como la regularización estructural, la poda de conexiones o la extracción simbólica de funciones.

Gracias a esta integración, se han incorporado técnicas de regularización basadas en sparsity durante el entrenamiento. Estas incluyen penalización L1 para eliminar conexiones innecesarias, regularización por entropía para fomentar activaciones selectivas, y restricciones sobre los coeficientes simbólicos para inducir representaciones más simples y generalizables. Todas estas penalizaciones se activan de forma progresiva en las últimas fases del entrenamiento, guiadas por métricas que cuantifican la influencia de cada conexión en la salida del modelo.

El modelo final se guarda de forma modular, incluyendo tanto los parámetros aprendidos como su configuración estructural y datos auxiliares. Esta organización asegura la trazabilidad del experimento y facilita su reutilización, permitiendo aplicar técnicas posteriores como la poda o la regresión simbólica gracias a la disponibilidad de toda la información interna del entrenamiento.

3.3.3 Conditional Tabular Generative Adversarial Networks (CTGAN)

En el modelo CTGAN, el generador recibe como entrada un vector aleatorio $z \sim \mathcal{N}(0, I)$ concatenado con un vector de condición que especifica las categorías que se quieren sintetizar con mayor probabilidad. Esta condición va a permitir al generador centrar la síntesis de muestras en clases concretas, realmente útil en casos de desequilibrios en el conjunto de datos [7]. Este vector combinado se procesa a través de una serie de capas residuales, que aplican transformaciones no lineales sobre la entrada. Finalmente, se aplica una capa lineal que transforma la representación intermedia en el vector de salida, cuyo tamaño coincide con la dimensión de los datos transformados.

Por otro lado, el discriminador está diseñado para recibir conjuntos de instancias agrupadas en bloques de tamaño indicado por el parámetro `pac`. En otras palabras, se generan "pac" batches con el generador, se le añaden las variables condicionantes a los batches y esta es la entrada al discriminador. Con esta estrategia, el modelo va a poder evaluar el realismo de los datos a nivel de grupos, pudiendo analizar patrones colectivos.

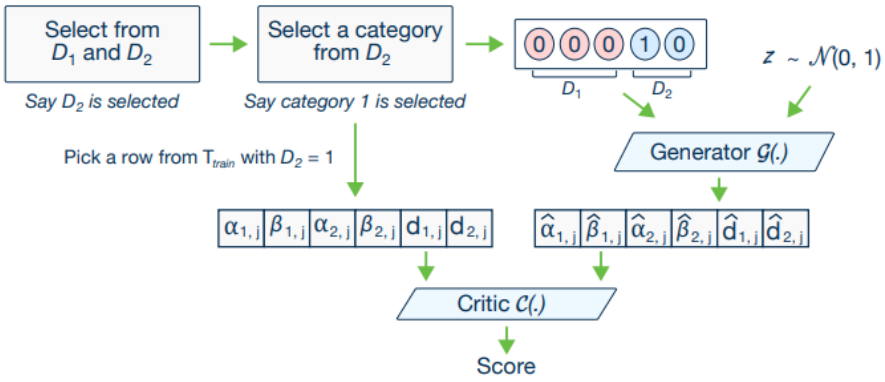


Figura 3.8: Arquitectura del modelo CTGAN dónde el generador construye filas sintéticas condicionadas a columnas discretas y el generador acumula varios batches para realizar la predicción [7].

La Figura 3.8 ilustra el proceso de entrenamiento condicional utilizado en CTGAN. En primer lugar, se selecciona aleatoriamente una columna categórica D_k y una de sus categorías posibles, utilizando un muestreo proporcional a la log-frecuencia para garantizar una exploración equilibrada de todas las clases. A partir de esa condición, se elige una fila real del conjunto de entrenamiento que contenga dicha categoría. Las variables continuas asociadas a esa fila se transforman mediante una normalización basada en la función *tanh*, que acota sus valores en el rango para estabilizar el entrenamiento. Las variables categóricas restantes se codifican mediante vectores one-hot y se reconstruyen utilizando una activación softmax, que permite modelar de forma probabilística la pertenencia a cada categoría. El generador toma como entrada un vector de ruido $z \sim \mathcal{N}(0, 1)$ junto con la codificación condicional de la categoría seleccionada, y produce una muestra sintética $\hat{x}$ en el mismo formato estructurado. Esta muestra, junto con los datos reales, se introduce en el discriminador para calcular la puntuación adversarial y optimizar el modelo [9].

La función de pérdida total aplicada en CTGAN se basa en el marco teórico de Wasserstein GAN con penalización de gradiente (WGAN-GP), cuyo objetivo es minimizar la distancia de Wasserstein entre la distribución real y la generada [10]. Esta formulación mejora la estabilidad del entrenamiento y mitiga problemas comunes en GANs clásicos como el colapso del generador. La función de pérdida del generador se define como:

$$\mathcal{L}_G = -\mathbb{E}_{z \sim p(z)} \left[D(G(z)) \right] \tag{3.5}$$

donde $G(z)$ representa la muestra sintética generada a partir del vector latente $z \sim \mathcal{N}(0, 1)$, y $D(\cdot)$ es la salida del discriminador (también denominado *critic* en el contexto de WGAN). Esta pérdida impulsa al generador a producir muestras a las que el discriminador asigne puntuaciones elevadas, acercándolas a la distribución real.

Por su parte, la función de pérdida del discriminador incorpora la penalización de gradiente propuesta en WGAN-GP para garantizar que el critic sea una función 1-Lipschitz. La pérdida se expresa como:

$$\mathcal{L}_D = \mathbb{E}_{\hat{x}\sim\mathbb{P}_g}\left[D(\hat{x})\right] - \mathbb{E}_{x\sim\mathbb{P}_r}\left[D(x)\right] + \lambda \cdot \mathbb{E}_{\tilde{x}\sim\mathbb{P}_{\tilde{x}}}\left[\left(\|\nabla_{\tilde{x}} D(\tilde{x})\|_2 - 1\right)^2\right] \tag{3.6}$$

donde: $\mathbb{P}_r$ es la distribución de los datos reales; $\mathbb{P}_g$ es la distribución de los datos generados; $\mathbb{P}_{\tilde{x}}$ representa una distribución interpolada entre muestras reales y generadas; λ es el coeficiente de penalización, típicamente fijado a 10.

Esta penalización adicional fuerza el gradiente del discriminador a mantenerse cercano a la unidad en los puntos interpolados, lo cual es necesario para preservar la propiedad de Lipschitz y garantizar una estimación válida de la distancia de Wasserstein. Así, el discriminador no solo aprende a distinguir entre ejemplos reales y sintéticos, sino que también proporciona una señal de entrenamiento más estable y significativa para el generador.

Ambas funciones de pérdida se aplican de manera condicionada, seleccionando aleatoriamente una columna categórica y una de sus categorías mediante muestreo proporcional a la log-frecuencia. Esta estrategia permite al modelo explorar de forma equilibrada todas las clases posibles, mejorando su capacidad de generación en dominios tabulares con distribución desigual entre categorías.

3.3.4 Implementación y entrenamiento CTGAN KAN

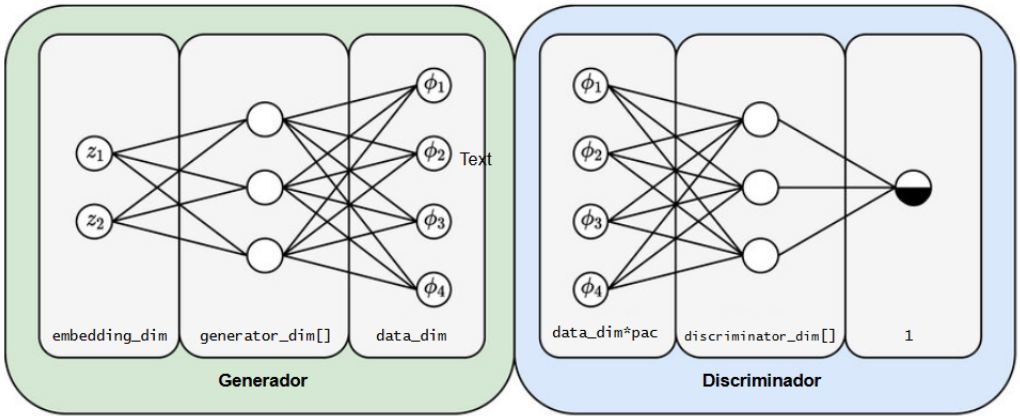


Figura 3.9: Arquitectura red neuronal CTGAN seleccionada para el proyecto.

La arquitectura seleccionada para esta red neuronal se basa en la sustitución de las capas densas tradicionales del generador y del discriminador por bloques construidos con la red `MultKAN` (Figura 3.9). Estas capas se configuran mediante los parámetros `generator_dim` y `discriminator_dim`, que determinan el número de nodos en cada capa oculta. Adicionalmente, el parámetro `embedding_dim` establece la dimensión del vector latente que sirve como entrada al generador, mientras que `data_dim` define el tamaño final de la muestra generada. En el caso del discriminador, esta dimensión se amplía mediante un factor de agrupación (`pac`), para permitir un tratamiento conjunto de varias muestras simultáneas.

Durante el entrenamiento, el modelo realiza un seguimiento del valor de las funciones de pérdida correspondientes al generador y al discriminador en cada época, permitiendo así evaluar su estabilidad y evolución. Una vez finalizado el proceso de entrenamiento, el generador puede utilizarse para producir nuevas instancias sintéticas, que pueden generarse de manera completamente aleatoria o bien condicionadas a determinadas categorías, permitiendo controlar la distribución de las muestras obtenidas. Las muestras generadas se transforman de nuevo al espacio original de los datos mediante la inversión del proceso aplicado por el módulo de preprocesamiento, asegurando la coherencia estructural con respecto al conjunto de datos real.

El uso de `MultKAN` en el modelo CTGAN presenta una serie de desafíos técnicos análogos a los vistos en la integración previa en TVAE. En particular, la red `MultKAN` no registra de forma automática las activaciones intermedias si no se utiliza su flujo interno de entrenamiento, lo cual imposibilita el uso de herramientas avanzadas como la regularización estructural o la extracción simbólica, salvo que se intervenga directamente en el ciclo de entrenamiento.

Para resolver esta limitación, se ha desarrollado una función auxiliar que emula la propagación hacia adelante de `MultKAN`, permitiendo almacenar explícitamente las salidas intermedias de cada capa, incluyendo preactivaciones, salidas interpoladas y escalas de activación, así como estadísticas relevantes de nodos y conexiones. Esta intervención manual posibilita la incorporación posterior de penalizaciones por sparsity en fases avanzadas del entrenamiento, mediante una estrategia progresiva equivalente a la empleada en el caso de TVAE.

Durante el desarrollo se identificaron, además, desafíos específicos relacionados con la gestión de la entrada condicional al generador. Dado que la concatenación del vector latente con variables categóricas genera vectores de tamaño variable, fue necesario incorporar una comprobación dinámica que ajusta automáticamente la anchura de entrada de la red `MultKAN`, generando arquitecturas adaptadas al tamaño real del vector condicional en cada caso.

Finalmente, al igual que en el caso de TVAE, el modelo se almacena de forma modular para garantizar su trazabilidad y reutilización. Junto con el archivo de parámetros entrenados, se guarda un archivo de configuración estructural y un conjunto de datos que incluye las activaciones intermedias necesarias para la aplicación posterior de técnicas avanzadas como la poda o la regresión simbólica.

3.4 Consideraciones para sparsity, poda y regresión simbólica

Una característica destacada de las redes KAN es su elevada interpretabilidad, además de su capacidad para generar datos sintéticos de alta calidad. En este proyecto se han desarrollado y adaptado herramientas específicas para analizar en profundidad su funcionamiento interno, lo que permite una comprensión más detallada de su comportamiento. Durante las fases finales del entrenamiento, se introduce de forma automática una etapa de regularización (esparsificación) que penaliza progresivamente aquellas conexiones menos relevantes, favoreciendo así representaciones más compactas y eficientes. Esta penalización se basa en métricas internas

que reflejan la contribución de cada componente al resultado final del modelo, y se incorpora directamente en la función de pérdida, facilitando una optimización que prioriza tanto el rendimiento como la simplicidad estructural.

$$\begin{aligned}
\mathcal{R}_{\text{enc}} &= \lambda_{\text{l1}} \cdot \|\boldsymbol{w}_{\text{enc}}\|_1 + \lambda_{\text{entropy}} \cdot H(\boldsymbol{a}_{\text{enc}}) + \lambda_{\text{coef}} \cdot \|\boldsymbol{c}_{\text{enc}}\|_2^2 + \lambda_{\text{coefdiff}} \cdot \|\nabla \boldsymbol{c}_{\text{enc}}\|_2^2 \\
\mathcal{R}_{\text{dec}} &= \lambda_{\text{l1}} \cdot \|\boldsymbol{w}_{\text{dec}}\|_1 + \lambda_{\text{entropy}} \cdot H(\boldsymbol{a}_{\text{dec}}) + \lambda_{\text{coef}} \cdot \|\boldsymbol{c}_{\text{dec}}\|_2^2 + \lambda_{\text{coefdiff}} \cdot \|\nabla \boldsymbol{c}_{\text{dec}}\|_2^2 \\
\mathcal{L}_{\text{total}} &= \mathcal{L}_{\text{principal}} + \alpha \cdot (\mathcal{R}_{\text{enc}} + \mathcal{R}_{\text{dec}})
\end{aligned} \tag{3.7}$$

En las ecuaciones anteriores, $\boldsymbol{w}$ representa los pesos de las conexiones, $\boldsymbol{a}$ las activaciones de las unidades, y $\boldsymbol{c}$ los coeficientes simbólicos aprendidos por el modelo. La función $H(\cdot)$ denota la entropía de activación, promoviendo una utilización más selectiva de las unidades, mientras que $\nabla \boldsymbol{c}$ representa las diferencias entre coeficientes adyacentes, incentivando una representación más suave y coherente.

La selección de los hiperparámetros viene dada por la búsqueda del equilibrio entre reducción estructural, especialización funcional y simplificación simbólica. Primero, se aplica una penalización L1 con un valor de $\lambda_{\text{l1}} = 1{,}0$, que ayuda a eliminar conexiones innecesarias haciendo que muchos pesos tomen valor cero. Este valor se ha seleccionado por su efectividad para inducir una estructura más escasa sin afectar demasiado al rendimiento del modelo. En segundo lugar, se aplica una penalización sobre la entropía de activación, con $\lambda_{\text{entropy}} = 2{,}0$, que tiene como objetivo que sólo un pequeño subconjunto de las unidades se active en cada paso.

Para facilitar la interpretación simbólica del modelo, se añaden además dos penalizaciones más pequeñas. La primera, con $\lambda_{\text{coef}} = 0{,}001$, limita el tamaño de los coeficientes simbólicos, favoreciendo la aparición de fórmulas más simples. La segunda, con $\lambda_{\text{coefdiff}} = 0{,}001$, penaliza las diferencias entre coeficientes similares, ayudando a mantener una representación más suave y coherente entre distintas partes de la red.

Todas estas penalizaciones se combinan en un único término que se añade a la función de pérdida principal mediante un factor de escala $\alpha = 0{,}001$. Este término se activa progresivamente durante las últimas épocas del entrenamiento, permitiendo que el modelo aprenda primero a representar bien los datos y después mejore su estructura interna, volviéndose más simple, eficiente y fácil de interpretar desde el punto de vista simbólico.

A continuación, se aplica una poda estructural que elimina nodos y conexiones de baja relevancia, reduciendo la complejidad de la red sin comprometer su rendimiento. El modelo entrenado puede ser visualizado a nivel de arquitectura y funcionamiento interno, lo que permite explorar la organización de capas, nodos y enlaces, así como cuantificar su importancia relativa. Finalmente, se realiza la extracción simbólica de fórmulas matemáticas que describen el comportamiento de la red, facilitando el análisis del impacto de cada variable y la detección de patrones representativos. Para todo esto se han creado módulos específicos de código para poder emplear estas funciones proporcionadas por `pykan` y extenderlas al uso en las redes planteadas en el proyecto.

Resultados

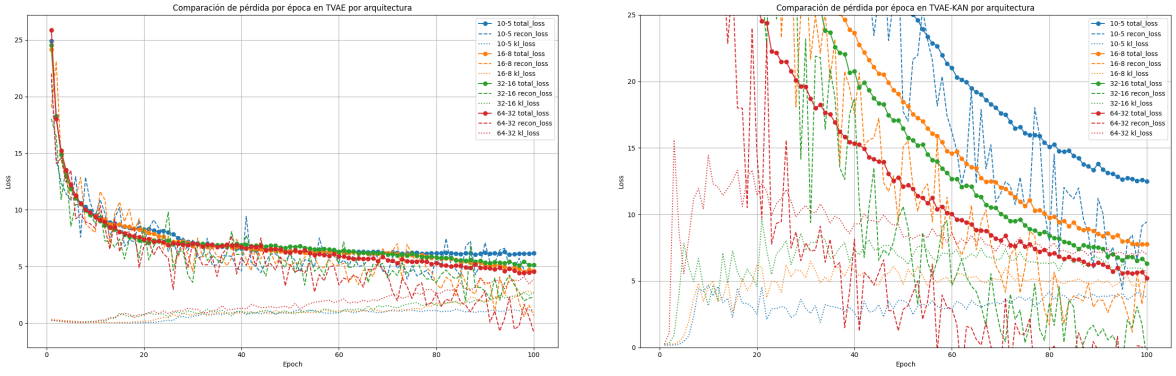
4.1 Resultados implementación de redes Kolmogorov Arnold en modelo generativo TVAE

En esta sección se presentará el análisis de los resultados obtenidos tanto con el modelo original generativo TVAE como con su modificación para la implementación de redes KAN, tanto a nivel de utilidad predictiva como calidad estadística. Se estudiarán también los resultados interpretativos obtenidos a partir de las funcionalidades derivadas de la implantación de la red MultKAN.

4.1.1 Resultados de entrenamiento y evaluación TVAE y TVAE KAN

Se realizan pruebas con cuatro arquitecturas distintas del modelo, con diferente número de neuronas en sus capas ocultas y distinto tamaño del espacio latente. Estas arquitecturas se definen mediante parejas de valores: [10, 5], [16, 8], [32, 16] y [64, 32]. Para el codificador, el primer número indica la cantidad de neuronas en la primera capa oculta, mientras que el segundo determina tanto el número de neuronas de la segunda capa oculta como el tamaño del vector de muestras latente. En el decodificador, la estructura se invierte: el segundo valor define la primera capa oculta y el primero la segunda. El tamaño del vector de entrada y de salida está determinado por la transformación de los datos, y se mantiene constante en 9 para las cuatro arquitecturas evaluadas.

Se entrenan las cuatro configuraciones y se obtienen los resultados representados en las siguientes Figuras, que reflejan la evolución de la pérdida de entrenamiento por época para las dos variantes, TVAE original y TVAE con redes KAN implementadas. Para cada arquitectura se muestran los diferentes componentes de la función de loss: `recon_loss` como la pérdida de reconstrucción, `kl_loss` como la divergencia KL y `total_loss` como la suma total de ambas.



(a) TVAE **(b)** KAN

Figura 4.1: Pérdidas modelos TVAE (izquierda) y TVAE-KAN en mismo rango (derecha).

Estos modelos han sido entrenados durante 100 épocas, con un tamaño del batch de 32 muestras. Se ha aplicado el optimizador Adam con un *learning rate* de 1×10^{-5} y un factor de pérdida de 2. La red `MultKAN` se ha inicializado con una base *spline* de orden 3 y 8 intervalos por dimensión (`grid=8`, `k=3`). Las funciones base se han fijado como `silu` (`base_fun=''silu''`) y se ha definido `mult_arity=2`, con el objetivo de permitir interacciones no lineales mediante nodos multiplicativos. Se introdujo ruido inicial en las *splines* con `noise_scale=0.3` y se fijó el entrenamiento en GPU (`device=''cuda''`), con una semilla fija para reproducibilidad (`seed=1`).

Se observa que TVAE-KAN parte de valores iniciales realmente altos (**400**), frente a valores más bajos de TVAE (**25**). A pesar de esta diferencia inicial, inducida por la complejidad adicional introducida por las redes KAN, su arquitectura muestra una convergencia rápida y eficaz. En la Figura 4.1, se observa como las curvas de la pérdida total reflejan valores medios entre 5.5 y 7.0 para KAN, frente a valores entre 4.4 y 5.3 para TVAE, mientras que la divergencia KL en KAN permanece elevada (7), mientras que en TVAE se mantiene mucho más baja (2.5–4.0).

Cabe destacar de estos resultados que un modelo puede contar con valores altos de KL, pero generar datos más diversos, fieles o útiles, ya que no se está sobreajustando a los patrones superficiales del entrenamiento. KAN, con técnicas como la esparsificación regulada, no busca replicar cada valor exacto, sino que se centra en buscar patrones estructurales y relaciones funcionales, evitando el sobreajuste y generando datos sintéticos que retienen mejor las correlaciones y dependencias entre variables, aprendiendo un espacio latente más regular.

Por lo tanto, a pesar de estas aparentes desventajas numéricas en las métricas internas, a continuación se demuestra que los datos generados por KAN obtienen mejores resultados en tareas externas, lo cual refleja que, aunque las gráficas de entrenamiento muestren un ajuste menos preciso, KAN es capaz de captar estructuras más útiles y generalizables.

4.1.2 Generación de datos con TVAE y análisis estadístico

Una vez completado el entrenamiento de los modelos, se genera un conjunto de datos sintéticos con el mismo tamaño que el conjunto de datos real. Se evalúa a continuación la precisión con la que estos datos generados reproducen las características del grupo original.

En primer lugar, se comparan los errores absolutos medios (MAE) por variable generada. En la Figura 4.21 se observa que el modelo TVAE-KAN mejora de forma consistente el MAE en variables clave como QuantumPatternFeature, Cholesterol, HeartRate y la variable objetivo HeartDisease, especialmente en arquitecturas más profundas como [64, 32] y [32, 16]. Por ejemplo, el MAE medio en HeartDisease se reduce de 0.1630 a 0.1300, mientras que en QuantumPatternFeature baja de 0.2037 a 0.1613. Por otro lado, el rendimiento en la variable Gender se mantiene similar entre ambos modelos.

Se aplica ahora el *test t de Student* para comparar las medias de ambas distribuciones. Se obtienen los valores p (*p-values*): cuanto más bajo sea este valor, mayor es la evidencia de que

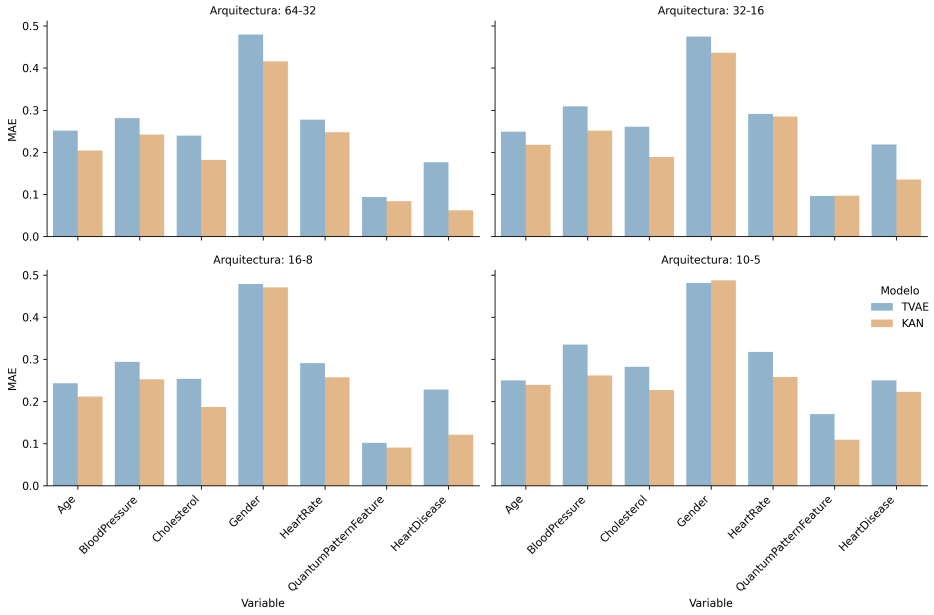


Figura 4.2: Comparación del MAE por variable para TVAE y TVAE-KAN.

las medias son distintas. Por el contrario, un valor p alto indica que no se puede rechazar la hipótesis nula, es decir, que no hay pruebas suficientes para afirmar que las medias difieren de forma significativa entre los datos reales y los generados. En el contexto de generación sintética, valores p altos son deseables, ya que sugieren que el modelo ha logrado replicar correctamente la media de las distribuciones originales.

Tabla 4.1: Comparación pvalues de t-Test por arquitectura y variable para TVAE y KAN

Arquitectura	**Age**		**BloodPressure**		**HeartRate**		**QuantumPatternFeature**	
	TVAE	KAN	TVAE	KAN	TVAE	KAN	TVAE	KAN
64-32	0.1425	0.1252	0.0000	0.3826	0.4284	0.0444	0.4191	0.1629
32-16	0.0026	0.7329	0.0000	0.4934	0.5770	0.1969	0.1957	0.1509
16-8	0.7973	0.0000	0.0000	0.0002	0.4076	0.0771	0.8004	0.5370
10-5	0.1438	0.0000	0.0000	0.0000	0.0018	0.3403	0.2558	0.1244

La Tabla 4.1 muestra que KAN presenta una mayor capacidad para preservar la media real de las variables clínicas, al mostrar un mayor número de valores p superiores a 0.05. Este comportamiento se vuelve más evidente en arquitecturas medias y grandes como [32, 16] y [64, 32].

Por ejemplo, para la variable BloodPressure, KAN alcanza valores p de 0.3826 [64, 32] y 0.4934 [32, 16], frente a los valores nulos de TVAE en ambos casos. Cabe destacar que la obtención de valores p nulos (0.0000) no significa que la media sea cero, sino que la diferencia observada es demasiado grande como para atribuirse al azar, es decir, que el modelo no ha conseguido

replicar correctamente la media de esa variable, con una reconstrucción deficiente donde no se ha logrado capturar el centro de la distribución real. En el caso de HeartRate, KAN mantiene valores p aceptables como 0.0444 [64, 32] y 0.1969 [32, 16], mientras que TVAE solo obtiene un valor p superior a 0.05 en la arquitectura [32, 16]. Para Age, KAN logra un p-value notable de 0.7329 con arquitectura [32, 16], frente al 0.0026 de TVAE en esa misma configuración.

Se aplica ahora el test de Mann-Whitney U, que va a permitir comparar los datos evaluando si existen diferencias significativas entre reales y sintéticos a nivel de distribución, extendiendo el análisis anterior sobre las medias.

Tabla 4.2: Comparación de pvalues de Mann Whitney U-Tests por arquitectura y variable para TVAE y KAN

Arquitectura	**Age**		**BloodPressure**		**HeartRate**		**QuantumPatternFeature**	
	TVAE	KAN	TVAE	KAN	TVAE	KAN	TVAE	KAN
64-32	0.1216	0.0795	0.0000	0.6031	0.3812	0.0960	0.3212	0.1109
32-16	0.0018	0.9575	0.0000	0.4253	0.6054	0.2798	0.2136	0.1668
16-8	0.7004	0.0000	0.0000	0.0160	0.3983	0.0970	0.7433	0.5196
10-5	0.0918	0.0000	0.0000	0.0000	0.0011	0.6496	0.2662	0.0846

Se obtienen valores, sobre todo en arquitecturas como [32, 16] y [64, 32], consistentemente mayores para KAN, superando el umbral de p>0.05 en la mayoría de las variables evaluadas, especialmente en variables BloodPressure y HeartRate, donde TVAE falla sistemáticamente. Este fallo vendría explicado por la reconstrucción basada en MSE aplicada por este modelo, que tiende a suavizar las distribuciones y a reducir la variabilidad, pudiendo derivar en que la distribución de las muestras sintéticas no respete las irregularidades del histograma original. KAN, en cambio, es capaz de capturar mejor esta estructura con sus oscilaciones locales, obteniendo así esos valores p mayores. Este modelo destaca especialmente en arquitecturas más pequeñas en variables como HeartRate o QuantumPatternFeature, fallando en Age para configuraciones mínimas. Esto último vendría explicado por su tamaño, donde al ser una red tan pequeña sus funciones spline no cuentan con la suficiente resolución como para aproximar correctamente la densidad real de la variable edad.

Se continúa con la prueba Chi-cuadrado sobre variables categóricas, que evalúa la frecuencia de aparición de las categorías Gender y HeartDisease.

Tabla 4.3: Comparación de Chi-cuadrado por arquitectura y variable categórica para TVAE y KAN

Arquitectura	**HeartDisease**		**Gender**	
	TVAE	KAN	TVAE	KAN
64-32	0.1994	0.5824	0.2066	0.8452
32-16	0.9467	1.0000	0.7526	0.7220
16-8	0.5668	1.0000	0.1277	0.9433
10-5	0.2932	0.3473	0.4433	0.6127

Destaca la arquitectura [16, 8], donde KAN alcanza un ajuste perfecto para HeartDisease y realmente alto para Gender, lo cual contrasta con valores notablemente más bajos en TVAE. En configuraciones como [64, 32] y [32, 16], tanto TVAE como KAN tienden a mantener buenos resultados, destacando los de KAN al ser ligeramente superiores. En arquitecturas más pequeñas, [10, 5], se observa un rendimiento general menor, especialmente por parte de TVAE.

Por último, se analizan las distancias coseno entre las distribuciones, con el objetivo de comparar si presentan una forma parecida, aunque sus valores no sean exactamente iguales. Se tratan las distribuciones como vectores y se mide el ángulo entre ellos: cuanto menor sea el resultado obtenido, menor será la distancia y mayor el parecido.

Tabla 4.4: Comparación de distancia coseno por arquitectura y variable para TVAE y KAN

Arquitectura	**Age**		**BloodPressure**		**HeartRate**		**QuantumPatternFeature**	
	TVAE	KAN	TVAE	KAN	TVAE	KAN	TVAE	KAN
64-32	0.2874	0.2006	0.3441	0.2212	0.3228	0.2426	0.1770	0.1116
32-16	0.2507	0.2207	0.3074	0.2212	0.3493	0.2066	0.1490	0.1058
16-8	0.2668	0.1965	0.3964	0.2239	0.3033	0.1895	0.1824	0.1066
10-5	0.2468	0.1882	0.4471	0.1645	0.3234	0.1559	0.1748	0.1074

KAN vuelve a superar sistemáticamente a TVAE en todas las configuraciones y arquitecturas propuestas, con distancias menores en todas las variables. Los valores reflejados en la Tabla 4.4 muestran una mayor similitud direccional entre los vectores de la distribución real y sintética: para la variable BloodPressure, KAN reduce consistentemente la distancia coseno respecto a TVAE en todas las arquitecturas, pasando de 0.3441 a 0.2212 en la arquitectura [64, 32] y de 0.4471 a 0.1645 en la más reducida [10, 5]. Del mismo modo, en HeartRate, KAN alcanza una distancia de 0.1559 en [10, 5], frente a 0.3234 de TVAE. Estas diferencias reflejan una mejor preservación de la estructura geométrica de los datos reales. Cabe destacar que las variables categóricas aparentan menor sensibilidad ante la disminución de tamaño de las arquitecturas, manteniéndose igualmente las mejoras en resultados de KAN.

4.1.3 Evaluación TSTR y TRTS modelo TVAE

Para evaluar la utilidad de los datos sintéticos generados por cada modelo, se aplican dos protocolos: TSTR (entrenar con sintéticos, probar con reales) y TRTS (entrenar con reales, probar con sintéticos). Esto permite comprobar si los datos sintéticos conservan la estructura predictiva de los reales. La evaluación se realiza con tres modelos de clasificación complementarios: Random Forest (RF) y Gradient Boosting (GB), cuentan con alta capacidad para capturar estructuras y relaciones no lineales, lo que permite evaluar si los datos sintéticos conservan patrones complejos similares a los del conjunto real; Regresión Logística, que actúa como un referente base al ser un modelo lineal, permitiendo analizar si los datos generados mantienen las relaciones simples y si las variables mas relevantes siguen siendo informativas. Así, se obtiene una visión robusta y equilibrada de la calidad de los datos generados.

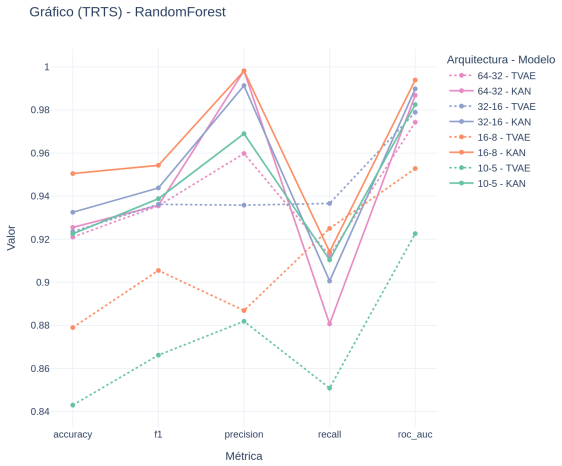


Figura 4.3: TRTS.

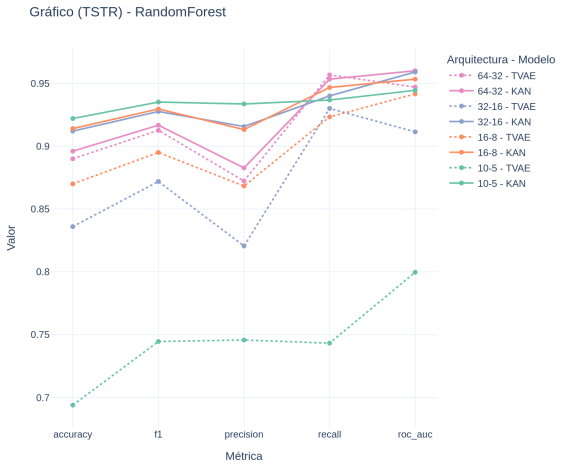


Figura 4.4: TSTR.

Figura 4.5: Evaluación TSTR y TRTS por arquitectura y modelo con Random Forest.

Se observa que el clasificador Random Forest muestra una mejora constante al evaluar los datos que ha generado el modelo TVAE con MultKAN, tanto en las métricas TSTR como en las TRTS (Figura 4.5). Buen rendimiento en TSTR indica que los datos sintéticos van a permitir entrenar modelos generalizables sobre datos reales, mientras que los buenos resultados en TRTS nos reflejan alta capacidad del modelo para preservar los patrones reales útiles. Destacan las arquitecturas más compactas, tanto la [10,5] como la [16,8], donde TVAE original obtiene F1 scores de 0,7835 y 0,7446 en TSTR, frente a 0,9386 y 0,9351 con KAN. En TRTS, la mejora también es clara, pues KAN alcanza un AUC de 0,9932 frente a 0,9292 de TVAE.

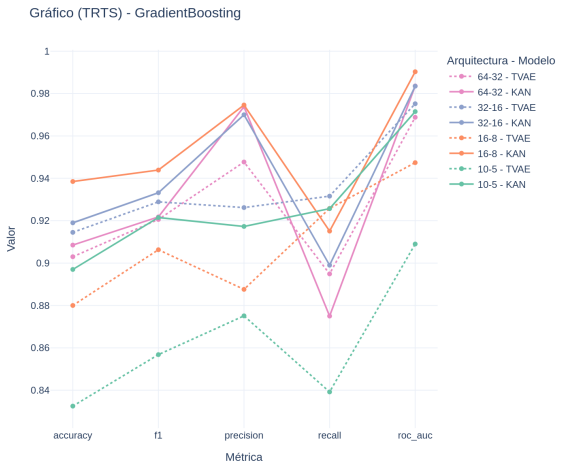


Figura 4.6: TRTS.

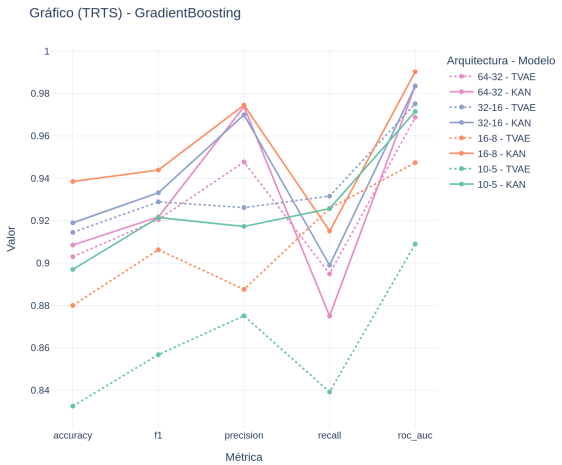


Figura 4.7: TSTR.

Figura 4.8: Evaluación TSTR y TRTS por arquitectura y modelo con Gradient Boosting.

Gradient Boosting muestra conclusiones similares, de nuevo especialmente con las arquitecturas de menor tamaño. Destaca en la Figura 4.8, la arquitectura [16,8], que muestra una mejora notable en TSTR, pasando de una F1 de 0,7341 con el TVAE original a 0,9272 con KAN, y un salto en ROC AUC de 0,7831 a 0,9507.

Figura 4.9: TRTS.

Figura 4.10: TSTR.

Figura 4.11: Evaluación TSTR y TRTS por arquitectura y modelo con Linear Regression.

Por último, se extraen las métricas para el modelo de Regresión Logística. En la Figura 5.6 se observa que el rendimiento en TRTS mejora en la arquitectura [12,6], donde el recall pasa de 0,8568 (TVAE) a 0,9862 (KAN), y el F1 de 0,8736 a 0,9464. Destaca también el deterioro en rendimiento de TVAE en su configuración más reducida, con [10, 5], en comparación con KAN que obtiene valores como 0,9862 en recall.

4.1.4 Análisis interpretativo modelo TVAE KAN

Para obtener las visualizaciones detalladas del grafo funcional interno de la red, se restauran todas las activaciones internas necesarias y se calculan las métricas de relevancia, como la conectividad entre nodos y subnodos. Se aplica este proceso sobre las configuraciones [16, 8] y [10,5]. De las Figuras 4.12, 4.13, 4.14, 4.15 se extrae un comportamiento asimétrico del modelo generativo, donde el codificador tiende a presentar más densidad y estructuración de conexiones en las capas iniciales, mientras que el decodificador las presenta en las capas profundas. Esto concuerda con la funcionalidad de estos sistemas, donde el codificador comprime la información recibida y el decodificador la reconstruye para recuperar los detalles de los datos originales, necesitando mayor capacidad representacional para su correcta expansión. Se observan los efectos de la esparsificación del entrenamiento, con estructuras depuradas y conexiones consideradas como útiles y esenciales por el modelo. En el modelo [16,8] encontramos mayor densidad de conexiones en las capas intermedias, mientras que el modelo [10,5] presenta una topología más delgada, reflejando mayor enfoque en relaciones esenciales.

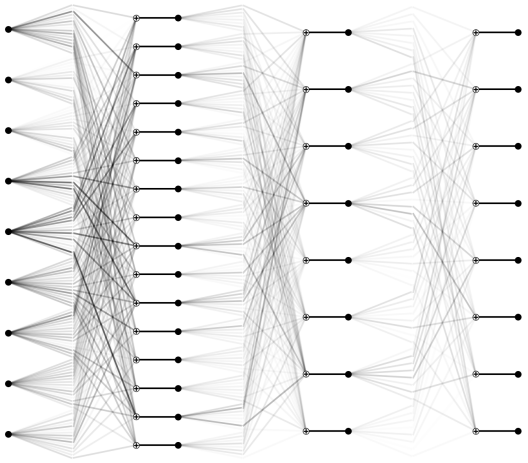


Figura 4.12: Codificador [16, 8].

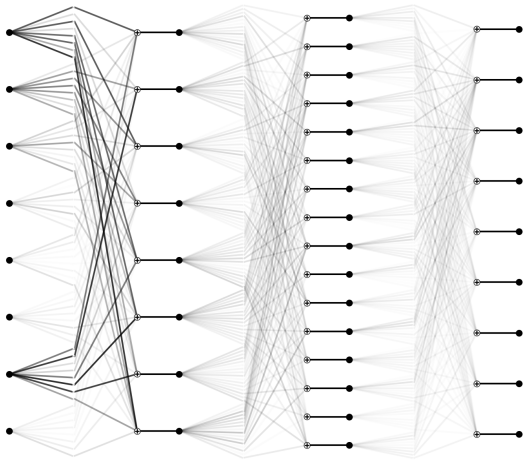


Figura 4.13: Decodificador [16, 8].

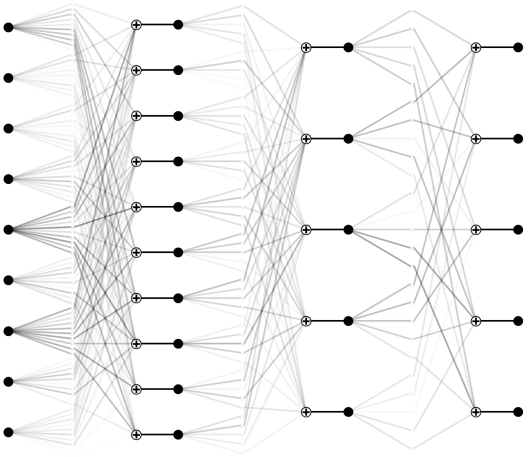


Figura 4.14: Codificador [10,5].

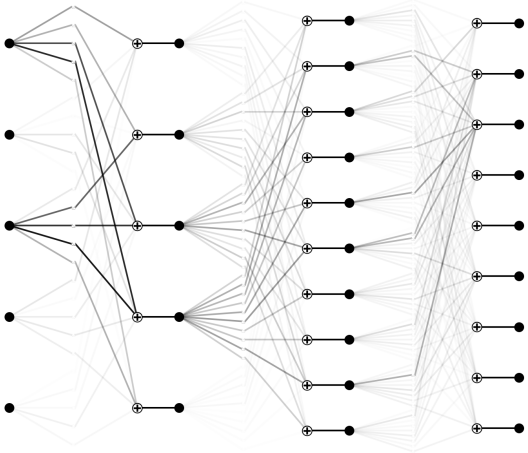


Figura 4.15: Decodificador [10,5].

A continuación, se aplica el proceso de poda sobre distintas arquitecturas. Para este procedimiento, la relevancia de cada nodo es evaluada respecto a dos métricas concretas: su actividad de entrada y su actividad de salida. Ambas se miden como el máximo valor de norma L1 de las funciones de activación conectadas al nodo. Si ambos valores están por debajo de un umbral mínimo, el nodo es considerado como no relevante y se elimina.

Se ejecuta esta poda sobre la arquitectura del decodificador con capas [64, 32], definiendo tres umbrales distintos: 0.04, 0.05 y 0.06. Los resultados (4.5) se van a ver reflejados sobre la segunda capa oculta de la arquitectura: con 0.04 se conservan 38 nodos, con 0.05 se reducen a 28 y con 0.06 únicamente se mantienen 11 nodos activos.

Tabla 4.5: Resumen de las dimensiones del *decoder* antes y después de la poda para distintos umbrales (arquitectura [64, 32]).

Umbral	Antes de la poda	Después de la poda
0,04	[32, 32, 64, 9]	[32, 32, 38, 9]
0,05	[32, 32, 64, 9]	[32, 32, 28, 9]
0,06	[32, 32, 64, 9]	[32, 32, 11, 9]

Para la siguiente prueba se aplica de nuevo el proceso de poda sobre el decodificador con arquitectura [32, 16] y con umbrales 0.06, 0.08 y 0.1, obteniendo los siguientes resultados (4.14): con 0.06 se conservan 23 nodos, con 0.05 se reducen a 16 y con 0.06 únicamente se mantienen 7 nodos activos.

Tabla 4.6: Resumen de las dimensiones del *decoder* antes y después de la poda para distintos umbrales (arquitectura [32, 16]).

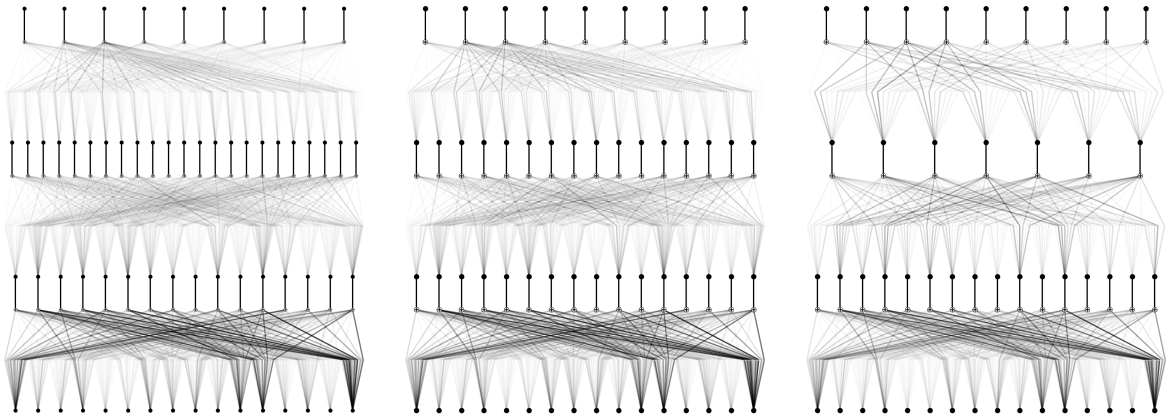
Umbral	Antes de la poda	Después de la poda
0,06	[16, 16, 32, 9]	[16, 16, 23, 9]
0,08	[16, 16, 32, 9]	[16, 16, 19, 9]
0,10	[16, 16, 32, 9]	[16, 16, 7, 9]

Se observa que la poda actúa de forma más agresiva en la tercera capa oculta tanto del codificador como del decodificador, lo cual puede deberse a la combinación de varios factores. En primer lugar, al estar cerca de la capa latente o de salida, esta capa suele concentrar nodos que ya no aportan información nueva, sino que simplemente transmiten combinaciones redundantes de activaciones anteriores. Además, las activaciones en capas profundas tienden a ser de menor magnitud, lo que incrementa la probabilidad de que queden por debajo del umbral de poda definido.

Se presentan las visualizaciones de la red de los modelos podados que reflejan cómo se eliminan gradualmente los nodos menos relevantes, conservando únicamente aquellas regiones relevantes y rutas funcionales esenciales. Se identifican las zonas con mayor densidad conectiva y los nodos con mayor contribución estructural restantes en la poda de mayor agresividad, con umbral 0.1 (Figura 4.16).

Por último se aplica la extracción de fórmulas simbólicas ya que a través de esta operación, se convierte el comportamiento de cada función de activación en una expresión simbólica analítica. Para ello, se analizan las activaciones aprendidas por la red de forma que, si una activación cuenta con una forma reconocible, se puede forzar su conversión simbólica al asignarse una función base f a dicha activación $\varphi_{i,j}$ de la capa l concreta. Después, para que la salida de la red tenga una forma similar a la de esta función base, se ajustan cuatro parámetros a, b, c y d, que permiten dicha aproximación con una expresión del tipo:

$$y \approx c \cdot f(ax + b) + d$$



Umbral: 0.06 Umbral: 0.08 Umbral: 0.1

Figura 4.16: Comparación visual del decodificador [32,16] tras poda con diferentes umbrales.

usando valores tomados de forma experimental. Esta aproximación se consigue mediante una búsqueda en una malla de valores posibles y una interpolación lineal para afinar el resultado, obteniéndose así de forma secuencial las fórmulas compactas e interpretables para cada componente de la red. Con este proceso se logra finalmente trazar la contribución funcional de cada variable de entrada sobre los vectores latentes y sobre los datos reconstruidos.

Se extraen entonces las fórmulas para el codificador y decodificador de los modelos TVAE-KAN con estructura [10,5] y [16,8]. A partir de estas fórmulas se realiza un estudio simbólico centrado en tres niveles: el espacio latente $\mathbf{z}$, la reconstrucción $\hat{\mathbf{x}}$ y la composición completa $\hat{\mathbf{x}} = f_{\text{dec}}(f_{\text{enc}}(\mathbf{x}))$. Los archivos con esta información se pueden obtener desde el siguiente enlace:

`https://drive.google.com/drive/folders/1pxovjkl74QyhZ2vq416ncrbHXn3XPP7T`

Para estudiar la influencia local de cada variable, se realiza un análisis de sensibilidad mediante derivadas parciales. Esto nos va a permitir cuantificar la influencia de cada variable de entrada sobre el espacio latente, así como el efecto de cada componente latente sobre la reconstrucción final (en este estudio, $\mathbf{x} = \mathbf{0}$):

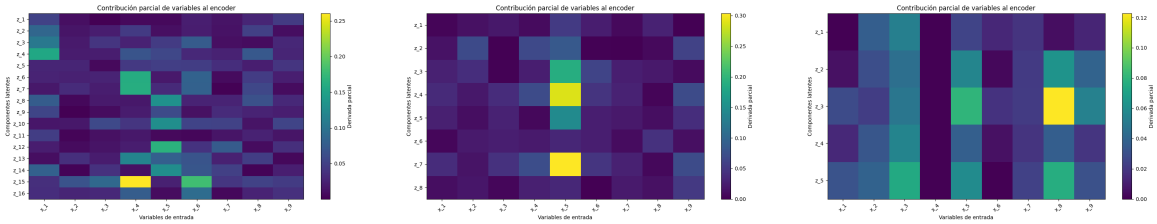
$$A_{ji} = \left| \frac{\partial f_{\text{enc}}^{(j)}}{\partial x_i} \right|_{\mathbf{x}=0}, \quad B_{kj} = \left| \frac{\partial g_{\text{dec}}^{(k)}}{\partial z_j} \right|_{\mathbf{z}=0} \tag{4.1}$$

En la matriz $A \in \mathbb{R}^{d \times p}$, cada entrada A_{ji} representa el valor absoluto de la derivada parcial de la j-ésima componente del espacio latente respecto a la variable de entrada x_i, lo cual refleja cómo una pequeña variación en x_i afecta localmente a $z_j = f_{\text{enc}}^{(j)}(x)$. De forma análoga, en la matriz $B \in \mathbb{R}^{p \times d}$ cada entrada B_{kj} mide el valor absoluto de la derivada parcial de la variable reconstruida $\hat{x}_k = g_{\text{dec}}^{(k)}(z)$ respecto a la componente latente z_j, indicando la sensibilidad de la reconstrucción a variaciones en este espacio.

En la Figura 4.17 se muestran los mapas de sensibilidad local para las arquitecturas $[32, 16]$, $[16, 8]$ y $[10, 5]$, respectivamente. En todos los casos, se observa un patrón claro: los codificadores tienden a distribuir la información entre un conjunto reducido de variables de entrada x_i, generando componentes latentes z_j con dependencias localizadas. Por ejemplo, en la arquitectura $[32, 16]$, componentes como z_5, z_9 o z_{13} dependen principalmente de variables como x_3 (género), x_4 (presión arterial) o x_6 (frecuencia cardíaca). Este comportamiento se mantiene en arquitecturas más compactas como $[16, 8]$, donde variables como x_1 (edad) y x_7 (QuantumPatternFeature) influyen claramente en ciertos z_j.

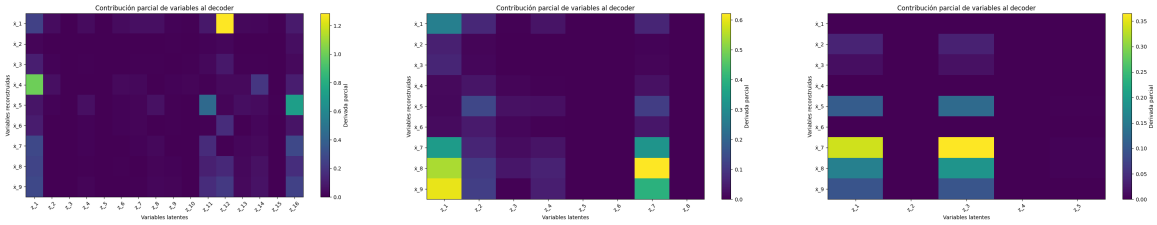
En los decodificadores, el modelo sigue una estrategia aún más direccionada: las variables reconstruidas $\hat{x}_k$ tienden a depender fuertemente de uno o dos componentes latentes específicos. Esto es especialmente visible en $[32, 16]$, donde $\hat{x}_3$ y $\hat{x}_6$ se reconstruyen principalmente a partir de z_2 y z_{13}, y se replica en $[10, 5]$, donde se conserva esta organización a pesar de la reducción de dimensiones. Este patrón recurrente sugiere que la arquitectura TVAE con KAN tiende a organizar el espacio latente de forma eficiente, distribuyendo la información de forma interpretativa durante la codificación y concentrándola de forma selectiva en la reconstrucción.

Encoder: sensibilidad local por arquitectura



(a) Arquitectura [32, 16] **(b)** Arquitectura [16, 8] **(c)** Arquitectura [10, 5]

Decoder: sensibilidad local por arquitectura



(d) Arquitectura [32, 16] **(e)** Arquitectura [16, 8] **(f)** Arquitectura [10, 5]

Figura 4.17: Sensibilidad local de las conexiones para distintas arquitecturas, separadas por encoder (arriba) y decoder (abajo).

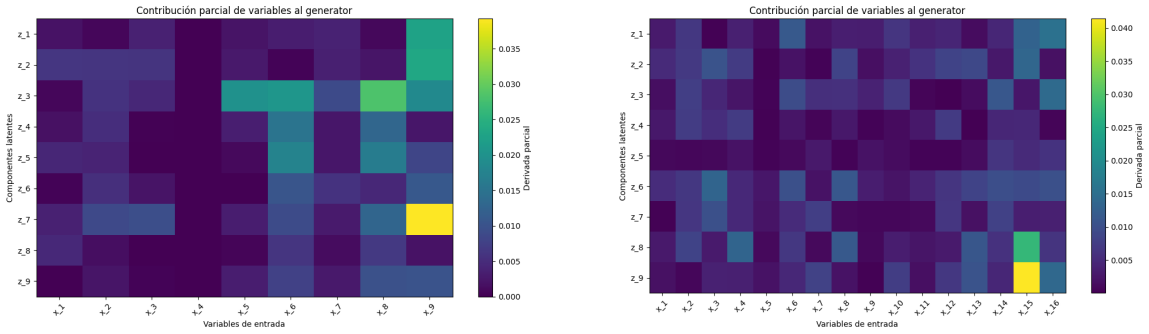
Como parte final del análisis, se realiza la composición de ambas funciones para estudiar directamente la relación entre las variables de entrada x_i y las salidas reconstruidas $\hat{x}_k$:

$$\hat{\mathbf{x}} = f_{\text{dec}}(f_{\text{enc}}(\mathbf{x})) = \hat{\mathbf{x}}(\mathbf{x})$$

Aplicando la regla de la cadena , se define una matriz de contribución total $C \in \mathbb{R}^{p \times p}$. Esta matriz mide la sensibilidad de cada salida reconstruida respecto a cada entrada original:

$$C_{ki} = \sum_{j=1}^{d} \left| \frac{\partial g_{\text{dec}}^{(k)}}{\partial z_j} \right| \cdot \left| \frac{\partial f_{\text{enc}}^{(j)}}{\partial x_i} \right| = (B \cdot A)_{ki}$$

Aquí, las matrices A y B recogen respectivamente las derivadas absolutas del codificador y del decodificador. Al multiplicarlas, su producto permite interpretar cómo se propaga la influencia de cada variable de entrada a través del modelo completo generando distintos mapas de calor para cada arquitectura:



(a) Generador (10,5)

(b) Generador (24,12)

Figura 4.18: Contribuciones comparadas del generador para dos arquitecturas: (10,5) y (24,12).

La Figura 4.18 muestra los mapas de calor resultantes para las arquitecturas $[10, 5]$, $[16, 8]$ y $[32, 16]$. En todos los casos se mantiene una estructura de contribuciones localizadas, donde cada $\hat{x}_k$ tiende a depender principalmente de un subconjunto reducido de variables de entrada x_i. En particular, en la arquitectura $[32, 16]$ se observa una fuerte influencia de x_5 (colesterol) y x_6 (frecuencia cardíaca) sobre variables como $\hat{x}_1$ (edad) y $\hat{x}_5$ (colesterol), reflejando una coherencia con los análisis de sensibilidad previos. En las arquitecturas más compactas, como $[16, 8]$ y $[10, 5]$, esta relación se conserva, aunque las contribuciones relevantes están más concentradas y se reduce el número de pares $x_i \to \hat{x}_k$ con influencia significativa, lo que sugiere que el modelo mantiene la estructura funcional esencial incluso con menos capacidad.

4.1.5 Tiempo medio para la generación de muestras con TVAE:

En la tabla 4.7 se muestran los tiempos medios de entrenamiento del modelo TVAE y su variante TVAE-MultKAN, diferenciando por arquitectura y dispositivo (CPU o GPU). El uso de GPU mediante CUDA ofrece ventajas notables respecto a CPU, con reducciones de tiempo consistentes en las configuraciones originales. Se observa que las versiones con MultKAN implementado introducen un sobrecoste computacional considerable, atribuible tanto al cálculo adicional de activaciones como a la aplicación de sparsificación estructurada y a la gestión explícita de parámetros en formato interpretable.

Tabla 4.7: Tiempos medios de procesamiento por arquitectura y tipo de red para TVAE.

Arquitectura	Modelo	CPU (s)	CUDA (s)
(10, 5)	TVAE classic	0.0362	0.0349
(10, 5)	TVAE_MultKAN	0.2842	0.2772
(16, 8)	TVAE classic	0.0324	0.0356
(16, 8)	TVAE_MultKAN	0.2479	0.2622
(32, 16)	TVAE classic	0.0295	0.0349
(32, 16)	TVAE_MultKAN	0.3389	0.2530
(64, 32)	TVAE classic	0.0331	0.0363
(64, 32)	TVAE_MultKAN	0.3543	0.2559

En paralelo a este estudio y una vez extraídas las fórmulas simbólicas, se ha medido también el tiempo medio de ejecución para la generación de muestras sintéticas a partir de estas representaciones. Los resultados de la tabla 4.8 reflejan que la generación de muestras a partir de fórmulas simbólicas es extremadamente eficiente, con valores de varios órdenes de magnitud inferiores a los tiempos requeridos por los modelos entrenados y evaluados anteriormente. Esto destaca gracias a su aplicación directa en la creación de estructuras de cálculo simples y eficientes (electrónica o procesadores de cálculo).

Tabla 4.8: Tiempos medios de procesamiento con fórmula simbólica para TVAE.

Modelo	Arquitectura	Tiempo medio (s)
TVAE	(16-8)	0.000021
TVAE	(10-5)	0.000002

4.1.6 Análisis de la simetría codificador-decodificador:

Se realiza ahora un análisis sobre la simetría del modelo, con el objetivo de evaluar si las transformaciones simbólicas aproximadas extraídas del codificador y del decodificador permiten una adecuada reconstrucción de una muestra tras pasar por el espacio latente.

Al analizar estos resultados se tiene en cuenta que, en modelos generativos como TVAE, el objetivo principal no es reconstruir cada muestra individual con precisión, sino aprender una representación latente que permita generar datos con una distribución global similar a la real. De esta forma, el modelo puede llegar a sacrificar la exactitud puntual en la reconstrucción para mantener una estructura latente coherente: TVAE no está optimizado exclusivamente para la reconstrucción, su función de pérdida incluye el término de regularización KL, que penaliza desviaciones respecto a una distribución normal en el espacio latente. Esto obliga al modelo a generalizar y evita que memorice los datos de entrada, lo que puede llegar a derivar en errores de reconstrucción más altos que en un autoencoder clásico, especialmente en variables categóricas o discretas.

Además, al analizar esta reconstrucción utilizando fórmulas simbólicas extraídas del modelo, se debe tener en cuenta que dichas expresiones son aproximaciones de las transformaciones

internas. Al simplificar el comportamiento de la red, estas fórmulas pueden también aumentar el error en la reconstrucción de muestras individuales.

Tabla 4.9: Errores medios de reconstrucción para fórmulas simbólicas.

Arquitectura	Error absoluto medio	Error relativo medio
(10-5)	6.4363	1.0329
(16-8)	7.3375	1.2061

Los resultados obtenidos para las arquitecturas [10,5] y [16,8], reflejados en la Tabla 4.9, muestran la media de los errores absolutos y relativos sobre 100 evaluaciones, con errores absolutos medios de 6.43 y 7.34, respectivamente, y errores relativos medios superiores a la unidad. Estos resultados, si bien no son óptimos, se interpretan como una evaluación adicional que se centra en la interpretabilidad del modelo. Para este análisis no se emplea directamente la red neuronal entrenada, sino una versión simbólica del sistema codificador-decodificador, por lo que los errores observados reflejan los límites actuales de la aproximación simbólica, y no del rendimiento original del modelo neuronal.

Estos resultados obtenidos no invalidan la calidad del generador, pero evidencian el reto inherente a representar de forma interpretable procesos complejos: mientras que la red original prioriza eficiencia y rendimiento, su versión simbólica proporciona una visión explicativa del funcionamiento interno del modelo.

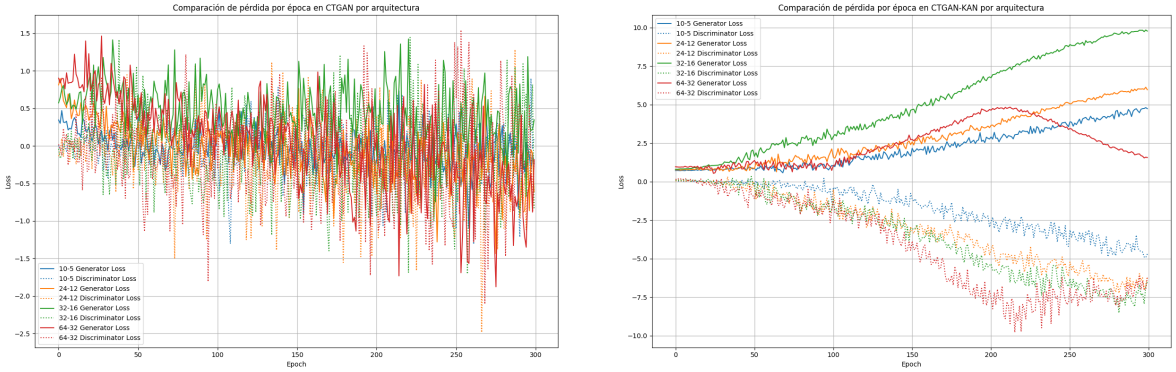
4.2 Resultados implementación de redes Kolmogorov Arnold en modelo generativo CTGAN

En esta sección se analizan y presentarán todos los resultados obtenidos tras implementar la arquitectura MultKAN en el modelo generativo CTGAN. Se presentarán los resultados de entrenamiento, evaluación y explicabilidad.

4.2.1 Resultados de entrenamiento y evaluación CTGAN y CTGAN KAN

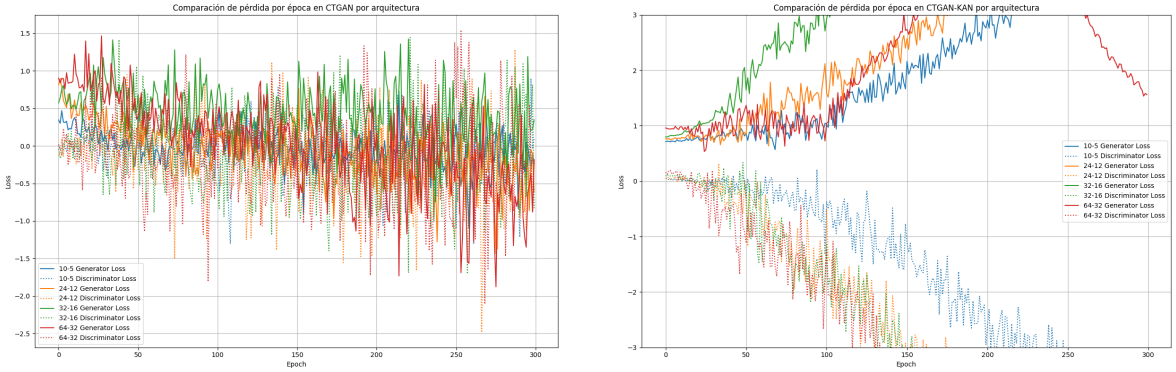
Para este modelo se han definido y evaluado también cuatro arquitecturas distintas, variando de nuevo la profundidad de las redes empleadas en el generador y discriminador: [10, 5], [24, 12], [32, 16] y [64, 32]. Cada una de estas parejas de valores define simétricamente la estructura interna de ambos componentes, estableciendo el número de nodos en sus capas ocultas. En el generador, el primer valor del par representa el tamaño de la primera capa oculta, mientras que el segundo valor determina la segunda capa. Por su parte, el discriminador toma como entrada un conjunto de muestras, que pueden ser tanto reales como generadas, cuya dimensión es del tamaño del vector de datos transformado multiplicada por el parámetro `pac`. A partir de dicha entrada, el discriminador aplica las capas especificadas con el par de valores, terminando con una salida escalar que representa la probabilidad de que las muestras sean reales.

Se entrenan las distintas configuraciones y se obtienen los resultados representados en las siguientes Figuras, que reflejan la evolución de la pérdida de entrenamiento por época para las dos variantes, CTGAN original y CTGAN con redes KAN implementadas. Para cada arquitectura se indican dos parámetros almacenados: `Generator Loss` como la pérdida del generador y `Discriminator Loss` como la del discriminador.



(a) CTGAN **(b)** KAN

Figura 4.19: Pérdidas modelos CTGAN (izquierda) y CTGAN-KAN (derecha).



(a) CTGAN **(b)** KAN

Figura 4.20: Pérdidas modelos CTGAN (izquierda) y CTGAN-KAN en mismo rango (derecha).

En la Figura 4.19a se muestra cómo evolucionan las funciones de pérdida del modelo CTGAN a lo largo del entrenamiento durante 300 épocas. La pérdida del generador $\mathcal{L}_G$ se mantiene en valores positivos y con mucha variación, lo que refleja que el entrenamiento no llega a ser del todo estable. Por su parte, la pérdida del discriminador $\mathcal{L}_D$ también presenta oscilaciones, pero disminuye de forma más consistente, llegando a valores negativos por debajo de $-1{,}0$. Esto sugiere que el discriminador está aprendiendo a diferenciar correctamente entre datos reales y sintéticos, manteniéndose más fuerte que el generador durante el proceso.

En la Figura 4.19b se representa la evolución de CTGAN-KAN. Durante este entrenamiento se observa una disminución sostenida de $\mathcal{L}_D$, que alcanza valores alrededor de $-8{,}3$, reflejando un progreso constante en su capacidad para separar ambas distribuciones. En el caso del generador se observa un incremento en $\mathcal{L}_G$, que evidencia que esta estructura aún no logra engañar al discriminador, pero también que está recibiendo gradientes útiles para mejorar. El análisis conjunto de las pérdidas muestra una dinámica donde el generador va ajustando progresivamente sus parámetros ante un discriminador cada vez más potente, que va mejorando su habilidad de detección sin saturación ni colapso. Este efecto se ve reflejado especialmente en la arquitectura más grande [64, 32], donde el generador comienza a mejorar logrando reducir su pérdida incluso frente a un discriminador realmente eficaz.

Finalmente, para un análisis más justo, en la Figura 4.20 se muestran ambas variantes utilizando el mismo rango vertical en el eje de pérdidas. Esta representación permite observar que el modelo con redes KAN, aunque presenta mayores valores absolutos de pérdida, sigue un patrón más estable y progresivo. Esto no debe interpretarse como un peor rendimiento de CTGAN-KAN, sino como una consecuencia de su arquitectura más expresiva y regularizada. Estas redes imponen tanto restricciones estructurales como esparsificación de forma que, aunque llegan a dificultar inicialmente la optimización del generador, acaban derivando en representaciones más generalizables y simbólicamente interpretables. Por tanto, como se ha mencionado, un valor más alto de $\mathcal{L}_G$ no implica menor calidad en los datos generados, sino un proceso de ajuste más exigente y gradual, que puede resultar en una mejor fidelidad estadística y utilidad práctica, como se corrobora en análisis posteriores mediante métricas TSTR, TRTS y estadísticas.

4.2.2 Generación de datos con CTGAN y análisis estadístico

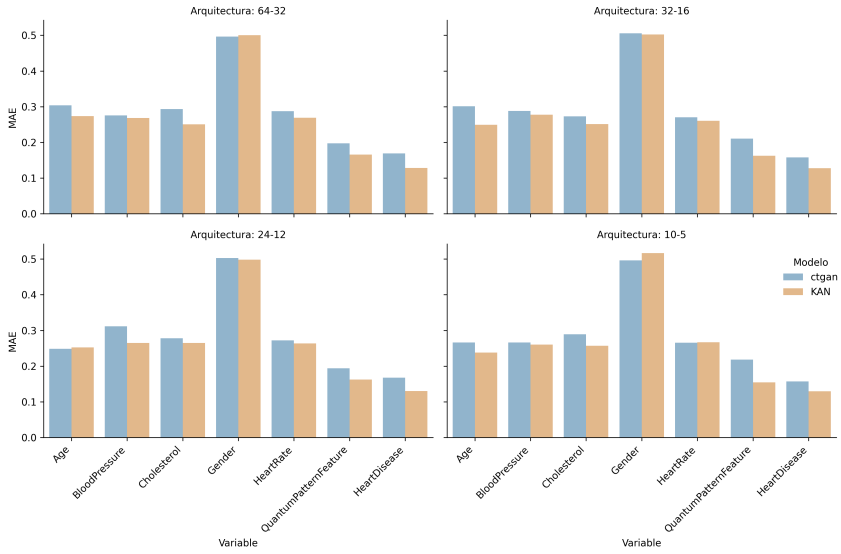


Figura 4.21: Comparación del MAE por variable para CTGAN y CTGAN-KAN.

Se realiza el análisis estadístico sobre los conjuntos de datos sintéticos generados por cada modelo y arquitectura de forma análoga a cómo se ha realizado sobre TVAE.

Siguiendo el mismo orden, se analizan en primer lugar los errores absolutos medios (MAE) por variable generada. En la Figura 4.21 se observa que KAN mejora la fidelidad sintética en variables como QuantumPatternFeature y BloodPressure, especialmente en arquitecturas profundas [64, 32] o [32, 16], mientras que el rendimiento es similar en variables como Gender.

Siguiendo con el estudio, se aplica el *test t de Student* para comparar las medias de ambas distribuciones. Los resultados presentados en la Tabla 4.10 muestran que el modelo KAN obtiene sistemáticamente valores p más altos que CTGAN en variables relevantes como BloodPressure, HeartRate y QuantumPatternFeature. Por ejemplo, en la arquitectura $[64, 32]$, KAN alcanza un valor p de 0.2750 en HeartRate, frente a un valor nulo de CTGAN, y 0.9325 en QuantumPatternFeature, frente también a un valor nulo en el modelo clásico.

Tabla 4.10: Comparación p-values de t-Test: CTGAN y KAN

Arquitectura	**Age**		**BloodPressure**		**HeartRate**		**QuantumPatternFeature**	
	CTGAN	KAN	CTGAN	KAN	CTGAN	KAN	CTGAN	KAN
64-32	0.7797	0.1765	0.2398	0.0119	0.0000	0.2750	0.0000	0.9325
32-16	0.0000	0.1765	0.0000	0.0119	0.0000	0.2750	0.0000	0.9325
24-12	0.0288	0.0171	0.0000	0.0555	0.2556	0.9950	0.0081	0.5723
10-5	0.0000	0.3745	0.0000	0.0455	0.0000	0.2727	0.0332	0.0024

Se aplica el test U de Mann–Whitney, obteniendo resultados que refuerzan las conclusiones extraídas anteriormente, KAN genera distribuciones más similares a los datos reales que CTGAN, alcanzando en general valores p mayores. Por ejemplo, en la variable HeartRate, KAN alcanza valores p de 0.2970 ($[64, 32]$ y $[32, 16]$), frente a valores nulos en CTGAN. En configuraciones más compactas como $[10, 5]$, KAN mantiene esta ventaja en todas las variables, como se observa en BloodPressure (0.0455 frente a 0.0000) y HeartRate (0.4270 frente a 0.0000).

Tabla 4.11: Comparación de p-values de Mann–Whitney U-Tests: CTGAN y KAN

Arquitectura	**Age**		**BloodPressure**		**HeartRate**		**QuantumPatternFeature**	
	CTGAN	KAN	CTGAN	KAN	CTGAN	KAN	CTGAN	KAN
64-32	0.6799	0.2159	0.3047	0.0147	0.0000	0.2970	0.0000	0.5228
32-16	0.0000	0.2159	0.0000	0.0147	0.0000	0.2970	0.0000	0.5228
24-12	0.0443	0.0202	0.0000	0.0563	0.2781	0.8861	0.0024	0.9985
10-5	0.0000	0.4304	0.0000	0.0455	0.0000	0.4270	0.0569	0.0011

Se calcula también la métrica de Chi-cuadrado para evaluar la similitud entre las distribuciones de las variables categóricas. Los resultados se muestran en la Tabla 4.12, donde se observa que el modelo CTGAN-KAN presenta una mayor estabilidad en sus resultados, especialmente en la variable Gender, donde alcanza valores elevados en todas las configuraciones, destacando 0.9596 en la arquitectura $[10, 5]$ y 0.8336 en $[24, 12]$. Por el contrario, CTGAN muestra mayor variabilidad, con valores más extremos en ambas direcciones, como 0.3713 en $[64, 32]$ y 1.0000 en $[10, 5]$, lo que sugiere un ajuste menos consistente.

En la variable HeartDisease, KAN mantiene una estabilidad en torno a 0.43 y 0.56 en todas las arquitecturas, mientras que CTGAN fluctúa más ampliamente, desde 0.4037 hasta 1.0000. Este comportamiento indica que, en general, KAN ofrece una mayor robustez en la preservación de variables categóricas, especialmente en arquitecturas más reducidas.

Tabla 4.12: Comparación de Chi-cuadrado: CTGAN y KAN

Arquitectura	**HeartDisease**		**Gender**	
	CTGAN	KAN	CTGAN	KAN
64-32	0.6771	0.4315	0.3713	0.5434
32-16	1.0000	0.4315	0.6499	0.5434
24-12	0.5346	0.5632	0.7934	0.8336
10-5	0.4037	0.4315	1.0000	0.9596

Por último, se obtienen las distancias coseno entre variables. En los resultados se observa superioridad de CTGAN-KAN. Destaca la variable QuantumPatternFeature, donde KAN presenta valores alrededor de 0.11, en comparación con cifras notablemente más altas para CTGAN. A partir de esto extraemos, de nuevo, que CTGAN-KAN logra una mejor reconstrucción del patrón latente de las variables en el espacio de representación aún incluso en las configuraciones más reducidas.

Tabla 4.13: Comparación de distancia coseno: CTGAN y KAN

Arquitectura	**Age**		**BloodPressure**		**HeartRate**		**QuantumPatternFeature**	
	CTGAN	KAN	CTGAN	KAN	CTGAN	KAN	CTGAN	KAN
64-32	0.2498	0.2270	0.3043	0.2084	0.3403	0.2144	0.2255	0.1132
32-16	0.3845	0.2270	0.2426	0.2084	0.3336	0.2144	0.1924	0.1132
24-12	0.2847	0.2321	0.4904	0.2115	0.2971	0.2180	0.2031	0.1133
10-5	0.2239	0.2234	0.3447	0.2094	0.3478	0.2255	0.1598	0.1104

4.2.3 Evaluación TSTR y TRTS modelo CTGAN

Se aplica sobre el modelo CTGAN original y CTGAN modificado con KAN la evaluación con TSTR y TRTS para analizar la utilidad predictiva. Se evalúa con los mismos tres modelos anteriormente mencionados: RF, GB y LR.

Los resultados obtenidos con el modelo Random Forest muestran que, de nuevo, KAN mejora en rendimiento para tareas TSTR, en concreto en arquitecturas como [24,12] y [10,5], donde se obtiene mayor accuracy, F1 y ROC AUC. En [24,12], KAN alcanza un F1 de 0.7330 frente a 0.6656 de CTGAN, presentando también un aumento notable en ROC AUC, lo que indica una mayor utilidad de los datos sintéticos generados. En TRTS, las diferencias son menos marcadas y no siempre favorables a KAN, a pesar de que en términos de ROC AUC sí mantiene un mayor equilibrio en algunas arquitecturas.

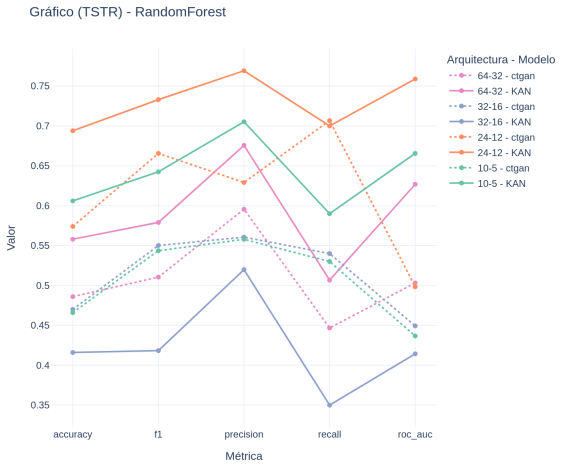


Figura 4.22: TSTR.

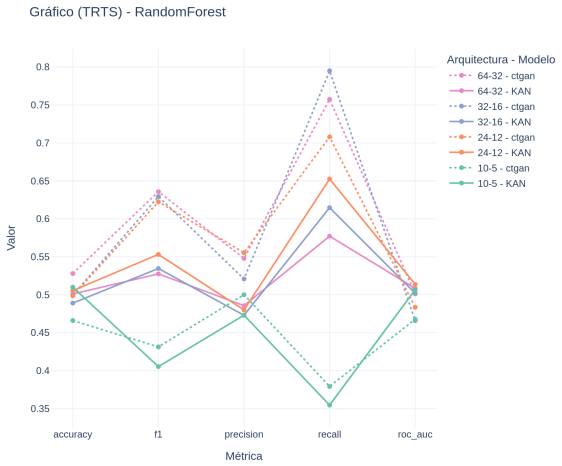


Figura 4.23: TRTS.

Figura 4.24: Evaluación TSTR y TRTS por arquitectura y modelo con Random Forest.

En GB se observan resultados donde de nuevo CTGAN-KAN denota superioridad, destacando también en arquitecturas compactas como [24,12] y [10,5], con mejoras destacadas en accuracy (0.704), F1 (0.7329) y ROC AUC (0.7570). En TRTS, las diferencias son más sutiles, pero KAN mantiene valores más equilibrados en recall y ROC AUC.

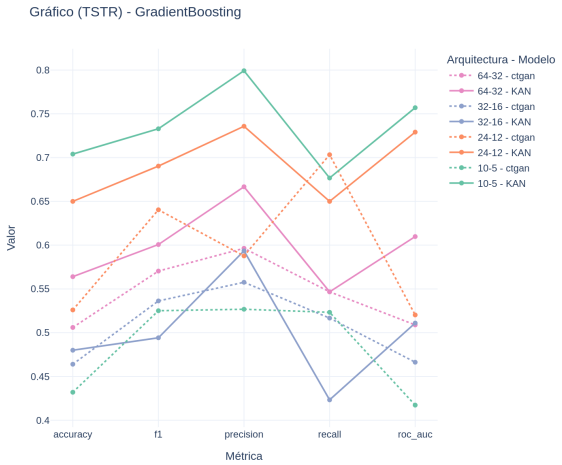


Figura 4.25: TSTR.

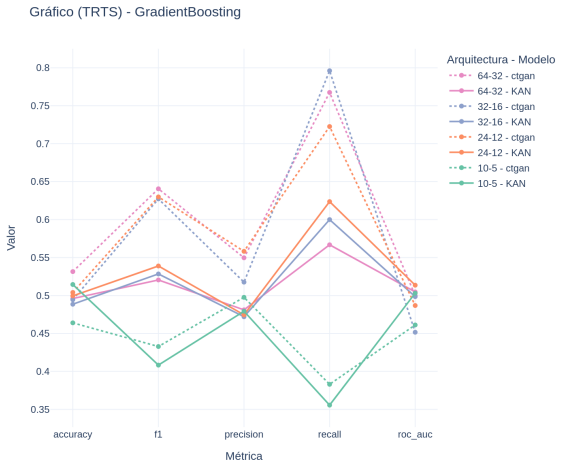


Figura 4.26: TRTS.

Figura 4.27: Evaluación TSTR y TRTS por arquitectura y modelo con Gradient Boosting

Por último, para RL, CTGAN-KAN destaca en TSTR en arquitecturas más profundas como 64-32, con mejoras claras en accuracy (0.628 vs 0.458), precision (0.8476 vs 0.5455) y ROC AUC (0.7606 vs 0.3782).

Figura 4.28: TSTR.

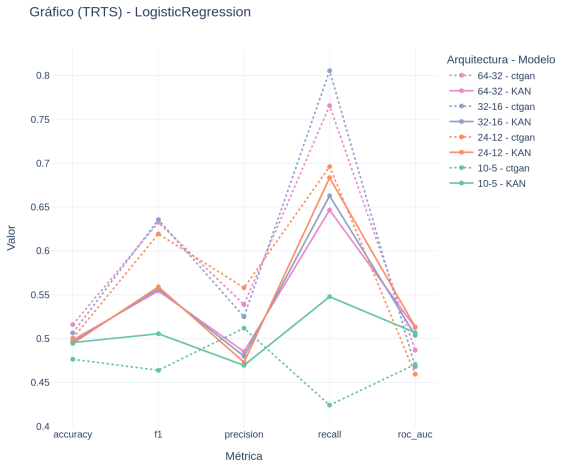


Figura 4.29: TRTS.

Figura 4.30: Evaluación TSTR y TRTS por arquitectura y modelo con Logistic Regression.

4.2.4 Análisis interpretativo modelo CTGAN KAN

Siguiendo la misma lógica aplicada durante el análisis interpretativo de TVAE-KAN, se estudian ahora los resultados obtenidos al aplicar el mismo enfoque sobre el modelo CTGAN-KAN.

Extraemos la visualización de los generadores del modelo para las arquitecturas [24, 12] y [10, 5]. La arquitectura [24,12] presenta una estructura densa y con alta conectividad interna (Figura 4.31), mientras que la arquitectura [10,5] (Figura 4.32) refleja una representación más legible, lo que favorece el análisis interpretativo al tener un menor número de nodos y conexiones.

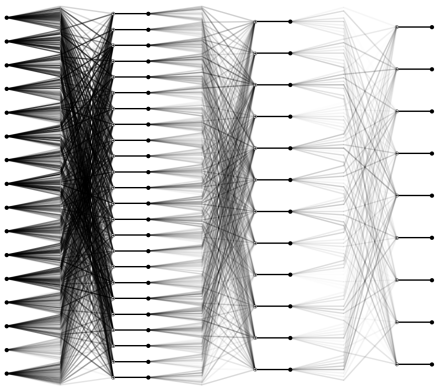


Figura 4.31: Representación arquitectura generator [24, 12] .

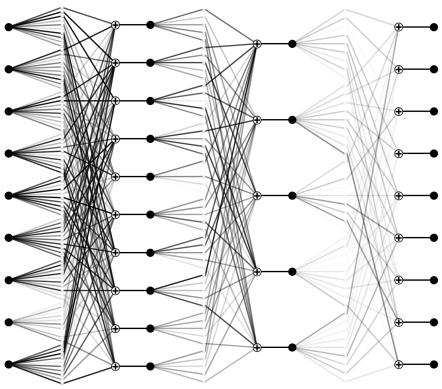


Figura 4.32: Representación arquitectura generator [10, 5].

Ahora, sobre los modelos entrenados, aplicamos la poda estructural. Se analiza la estructura [24,12] como ejemplo, aplicando un umbral de 0.20 sobre el generador y de 0.08 sobre el discriminador. Se observa tras la ejecución que en el generador, a pesar de que la agresividad de la poda es mayor, se han eliminado unicamente dos nodos de la segunda capa oculta, que pasa de tener doce a diez nodos. En el caso del discriminador, con un umbral de poda menor, se eliminan nodos tanto de la primera como de la segunda capa oculta, pasando de [24, 12] a [15, 6].

Tabla 4.14: Resumen de las dimensiones del *generador* y *discriminador* antes y después de la poda con umbrales: 0,20 (generador) y 0,08 (discriminador). Arquitectura [24, 12].

Umbral	Red	Antes de la poda	Después de la poda
0,20	Generador	[16, 24, 12, 9]	[16, 24, 10, 9]
0,08	Discriminador	[370, 24, 12, 9]	[370, 15, 6, 1]

En la Figura 4.33 se representa la estructura interna del generador tras la poda, donde se observa la reducción mencionada de conexiones en la tercera capa. En este estudio no se representan visualmente las estructuras internas del bloque discriminador debido a las elevadas dimensiones que presenta su arquitectura, consecuencia directa del mecanismo de agrupación en la entrada definido por el parámetro `pac`.

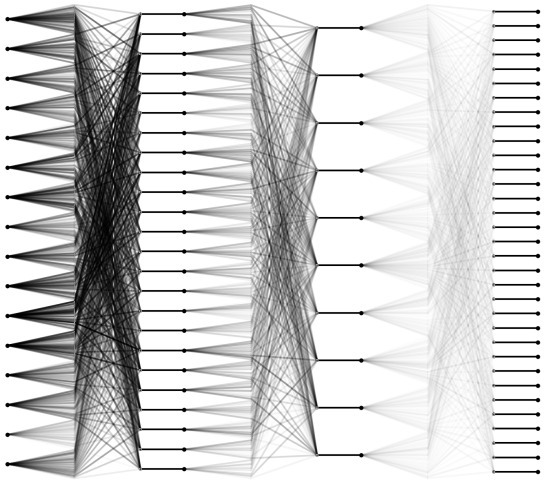


Figura 4.33: Visualización del generador [24,12] tras la poda con umbrales 0.20.

Por último, se realiza la extracción de fórmulas simbólicas tanto para el generador como para el discriminador de las arquitecturas [24,12] y [10,5], replicando el procedimiento previamente ejecutado sobre las redes del codificador y decodificador de TVAE. De estas fórmulas se extrae igualmente la matriz de sensibilidad mediante derivadas parciales absolutas evaluadas en un punto de referencia $\mathbf{x} = \mathbf{0}$.

Las Figuras 4.34 y 4.35 muestran cómo varía la sensibilidad de los componentes latentes del generador en relación a las variables de entrada. En el caso de la arquitectura [10, 5] 4.34, se detectan contribuciones más concentradas, donde algunos componentes latentes z_3, z_5 y z_7 muestran una dependencia clara de las variables x_5 (colesterol total), x_6 (frecuencia cardíaca) y x_9 (presencia de enfermedad cardíaca). A partir de este patrón se deduce una estructura más simple y dirigida, en la que cada componente z_j se ha especializado en codificar información relevante de una o pocas variables clínicas concretas. Por ejemplo, z_7 presenta una alta sensibilidad respecto a x_9, lo que indica que la presencia de enfermedad cardíaca tiene un papel determinante en la construcción de este componente latente. Se observa también una fuerte contribución de x_5 sobre z_3, lo cual refleja la importancia del colesterol como variable discriminativa.

En cambio, para la arquitectura [24, 12] (Figura 4.35), se observa una distribución más repartida de las contribuciones, donde varios componentes latentes muestran sensibilidad a múltiples variables de entrada. Por un lado, esto ofrece una representación interna con mayor capacidad de codificación, pero por otro, acaba derivando en una menor interpretabilidad directa. Aun así, se detecta una contribución destacada de x_{15}, que corresponde a x_6 (frecuencia cardíaca), al componente z_8, lo que refuerza la importancia de esta variable fisiológica en la estructura latente del generador, incluso en modelos más complejos.

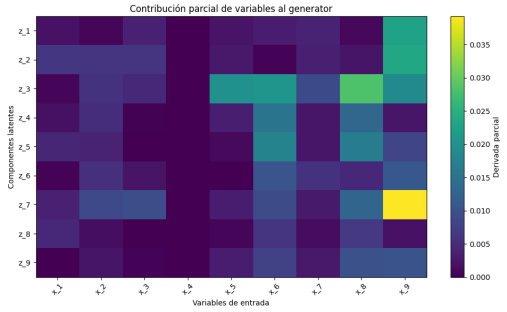


Figura 4.34: Contribuciones arquitectura generador (10,5).

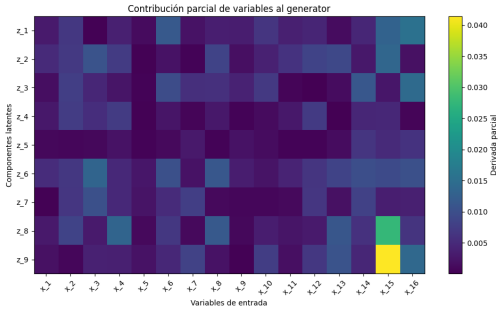


Figura 4.35: Contribuciones arquitectura generador (24,12).

En ambos casos, el mapeo clínico de las variables representadas permite interpretar estos patrones. Se conoce que la variable x_5 (colesterol total) es un factor de riesgo clave en enfermedad coronaria, que x_6 (frecuencia cardíaca) es un indicador de disfunción autonómica o estrés cardiovascular y que x_9 es la variable objetivo binaria que determina la presencia de enfermedad cardíaca.

4.2.5 Tiempo medio para la generación de muestras con CTGAN:

La tabla 4.15 presenta los tiempos medios de procesamiento para el modelo CTGAN (generador) y su variante CTGAN-MultKAN (generador). Al igual que en TVAE, el uso de CUDA ofrece mejoras en tiempo de ejecución para las versiones clásicas.

Tabla 4.15: Tiempos medios de procesamiento por arquitectura y tipo de red para CTGAN.

Arquitectura	Modelo	CPU (s)	CUDA (s)
(10, 5)	CTGAN classic	0.1347	0.0931
(10, 5)	CTGAN_MultKAN	0.1673	0.3051
(24, 12)	CTGAN classic	0.0540	0.1137
(24, 12)	CTGAN_MultKAN	0.1711	0.2834
(32, 16)	CTGAN classic	0.1397	0.1025
(32, 16)	CTGAN_MultKAN	0.2156	0.2895
(64, 32)	CTGAN classic	0.1457	0.0950
(64, 32)	CTGAN_MultKAN	0.2567	0.2587

En cuanto a la inferencia simbólica, la tabla 4.16 muestra los tiempos medios obtenidos tras 1000 generaciones de muestras mediante las ecuaciones simbólicas extraídas del generador de CTGAN. A pesar del mayor coste de entrenamiento, la generación simbólica continúa siendo extremadamente rápida lo cual sigue siendo el gran punto a destacar pues este tipo de arquitecturas también podrían implementarse de manera eficiente en dispositivos electrónicos o de cómputo.

Tabla 4.16: Tiempos medios de procesamiento con fórmula simbólica para CTGAN.

Modelo	Arquitectura	Tiempo medio (s)
CTGAN	(24-12)	0.000519
CTGAN	(10-5)	0.000055

En conjunto, los resultados obtenidos muestran que, a pesar de que la arquitectura con MultKAN introduce un mayor coste computacional, sobre todo al usar GPU, dicho incremento es coherente con la complejidad añadida por estas redes. A pesar de este aumento en los tiempos de generación, estos modelos han demostrado ofrecer beneficios: mayor interpretabilidad estructural, posibilidad de esparsificación y extracción simbólica de las funciones generadoras. Gracias a esto se ve reforzada su aplicabilidad en contextos donde la trazabilidad del modelo cuenta con gran relevancia.

Por otro lado, la inferencia basada en fórmulas simbólicas extraídas del generador destaca con valores extremadamente bajos: los tiempos de generación son del orden de microsegundos, superando ampliamente la velocidad de cualquier red neuronal activa. Esto sugiere que, una vez entrenado el modelo y extraída su representación simbólica, es posible desplegar soluciones de generación de datos sintéticos prácticamente instantáneas. Sin embargo, ha de tenerse en cuenta la importancia de encontrar un equilibrio adecuado entre la velocidad de cómputo alcanzada mediante fórmulas y estructuras simplificadas, y la precisión o capacidad generativa que estas representan. Se debe buscar el punto en el que la red cuente con una profundidad suficiente como para que sus representaciones simbólicas ajusten correctamente las distribuciones objetivo y produzcan resultados fiables, pero sin alcanzar una complejidad estructural que impida la extracción y comprensión de dichas fórmulas, haciéndolas excesivamente complicadas.

Conclusiones

Recientemente, la importancia y necesidad de trabajar con modelos de aprendizaje de máquina interpretables ha cobrado un papel central, especialmente en campos críticos como el biomédico, donde la comprensión de sus resultados se considera tan relevante como su precisión. A lo largo de este proyecto se ha abordado la implementación de las Redes Kolmogorov-Arnold en modelos generativos, tanto en TVAE como en CTGAN, con el objetivo de aportar una nueva capa de transparencia [1].

Uno de los puntos clave derivados de esta integración ha sido el compromiso entre la capacidad de codificación e interpretabilidad simbólica en función del tamaño de la arquitectura KAN aplicada. A través de las distintas pruebas ejecutadas, se han obtenido resultados que evidencian cómo una red KAN de menor tamaño (con menos funciones base y menor profundidad) es capaz de alcanzar rendimientos comparables a los de las redes originales de mayor tamaño, lo que concuerda con otros estudios previos comparativos [5]. Se ha observado también que arquitecturas más profundas mejoran la capacidad del modelo, aunque también generan fórmulas simbólicas más complejas y menos interpretables, resaltando la importancia de ajustar la arquitectura para equilibrar interpretabilidad y fidelidad en los datos generados.

Por otro lado, una de las aportaciones más relevantes obtenidas gracias a la implementación de estas redes ha sido la capacidad de rastrear la contribución de cada componente latente a las variables reconstruidas. Con su estructura simbólica, las KAN nos han permitido obtener las fórmulas necesarias para poder observar cómo determinadas variables clínicas, como colesterol o frecuencia cardíaca, influyen directamente en la codificación y generación de otras. Este nivel de descomposición funcional de la red va a ser realmente útil en entornos clínicos reales, donde la implantación de normativas éticas estrictas, como GDPR, exigen trazabilidad y justificación de los procesos automatizados [23], [10].

En el contexto específico del conjunto de datos relacionado con la enfermedad cardíaca estudiado, se han obtenido resultados altamente ilustrativos. La visualización de los mapas de sensibilidad y de contribución total permite una interpretación clínica directa de los resultados del modelo, donde se ha observado que variables clave como el colesterol, la frecuencia cardíaca o la edad se identifican como factores determinantes del riesgo, lo cual concuerda con el conocimiento médico establecido sobre esta patología. Esta trazabilidad ha supuesto una herramienta fundamental para una síntesis de datos ética y transparente, que ha permitido auditar y explicar las bases de las predicciones generadas. De esta forma, el proceso de generación de datos sintéticos produce registros clínicamente verosímiles y de utilidad práctica, asegurando que reflejen patrones reales de la patología cardíaca y puedan emplearse con confianza en investigaciones o aplicaciones médicas futuras, respaldando el uso de estas redes como una alternativa viable con garantías de trazabilidad, explicabilidad y utilidad real.

Bibliografía

[1] T. Ji, Y. Hou, and D. Zhang, "A comprehensive survey on kolmogorov arnold networks (kan)," 2025.

[2] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark, "Kan: Kolmogorov-arnold networks," 2025.

[3] S. SS, K. AR, G. R, and A. KP, "Chebyshev polynomial-based kolmogorov-arnold networks: An efficient architecture for nonlinear function approximation," 2024.

[4] J. Schmidt-Hieber, "The kolmogorov-arnold representation theorem revisited," 2021.

[5] Z. Liu, P. Ma, Y. Wang, W. Matusik, and M. Tegmark, "Kan 2.0: Kolmogorov-arnold networks meet science," 2024.

[6] A. X. Wang, M. Le, H.-T. Duong, B. Nguyen, and B. Nguyen, "Ctvae: Contrastive tabular variational autoencoder for imbalance data," *Knowledge and Information Systems*, vol. 67, pp. 5335–5354, 03 2025.

[7] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan," 2019.

[8] Z. C. Lipton, "The mythos of model interpretability," 2017.

[9] Z. Zhao, A. Kunar, H. V. der Scheer, R. Birke, and L. Y. Chen, "Ctab-gan: friedman," 2021.

[10] A. A. Barr, R. Rozman, and E. Guo, "Generative adversarial networks vs large language models: a comparative study on synthetic tabular data generation," 2025.

[11] Y. Zhang, P. Tino, A. Leonardis, and K. Tang, "A survey on neural network interpretability," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, p. 726–742, Oct. 2021.

[12] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," 2017.

[13] A. K. Kuchibhotla, L. D. Brown, A. Buja, and J. Cai, "All of linear regression," 2019.

[14] M. K. Chung, "Introduction to logistic regression," 2020.

[15] Y. Qiu, J. Liu, and D. Wang, "Truthful generalized linear models," 2022.

[16] A. Solonen and S. Staboulis, "On bayesian generalized additive models," 2023.

[17] Y. Izza, A. Ignatiev, and J. Marques-Silva, "On explaining decision trees," 2020.

[18] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," 2016.

[19] Z. Zhou, G. Hooker, and F. Wang, "S-lime: Stabilized-lime for model explanation," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, (New York, NY, USA), p. 2429–2438, Association for Computing Machinery, 2021.

[20] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," 2017.

[21] A. M. Salih, Z. Raisi-Estabragh, I. B. Galazzo, P. Radeva, S. E. Petersen, K. Lekadir, and G. Menegaz, "A perspective on explainable artificial intelligence methods: Shap and lime," *Advanced Intelligent Systems*, vol. 7, June 2024.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[23] R. Binns, "Fairness in machine learning: Lessons from political philosophy," 2021.

[24] L. Ziyin, M. Chen, and M. Arjovsky, "Kolmogorov–arnold networks," *arXiv preprint arXiv:2402.15244*, 2024.

[25] A. Goncalves, P. Ray, B. Soper, L. Stevens, L. Coyle, and A. P. Sales, "Generation and evaluation of synthetic patient data," *BMC Medical Research Methodology*, vol. 20, no. 1, pp. 1–38, 2020.

[26] "Proposal for a regulation of the european parliament and of the council laying down harmonised rules on artificial intelligence (artificial intelligence act)," 2021.

Anexos

A Aspectos éticos, económicos, sociales y ambientales

A.1 Introducción

En la actualidad, la generación de datos sintéticos ha surgido como una herramienta potente fundamental para la investigación, especialmente en dominios como el clínico. Este Trabajo Fin de Grado se centra en la generación de datos sintéticos tabulares con la integración de Redes Kolmogorov–Arnold (KAN) en modelos generativos tradicionales como Tabular VAE (TVAE) y CTGAN. El objetivo principal del estudio radica en aprovechar las ventajas que ofrecen las KAN en términos de eficiencia e interpretabilidad, construyendo modelos generativos más explicables, auditables y confiables [24].

Dado el contexto de creciente adopción de técnicas de inteligencia artificial en aplicaciones médicas, es necesario considerar para este proyecto las posibles repercusiones éticas, económicas, sociales y medioambientales, de acuerdo con los principios de sostenibilidad e impacto responsable.

En esta sección se va a examinar la relevancia del trabajo en cada una de estas áreas: primero se identifican y describen los principales impactos detectados, a continuación se profundiza en la importancia de la explicabilidad de los modelos en entornos médicos y, finalmente, se presentan unas conclusiones generales sobre el valor añadido y la sostenibilidad global de la propuesta.

A.2 Descripción de impactos relevantes relacionados con el proyecto

Aspectos éticos

La aplicación de datos sintéticos en salud permite proteger la privacidad de los pacientes al evitar el uso directo de información personal. Se entrenan modelos con datos estadísticamente similares a los reales, pero artificiales, realizando análisis sin comprometer la confidencialidad [25]. En este proyecto, el uso de redes KAN aporta un beneficio adicional: al ser modelos

interpretables, es posible entender cómo se generan los datos sintéticos y detectar posibles decisiones injustas o erróneas. Esto refuerza la transparencia y se alinea con principios éticos como el "derecho a la explicación" en decisiones automatizadas, especialmente importantes en entornos clínicos [23].

Aspectos económicos

La generación de datos sintéticos también tiene ventajas económicas, pues permite disponer de grandes volúmenes de datos sin necesidad de recolectar información real, lo cual reduce costes y acelera el desarrollo de estos modelos. En el ámbito bioméico esto se vuelve de gran utilidad, pues la obtención de datos es un proceso tanto caro como complicado. Además, las redes KAN, al ser más compactas y eficientes, requieren menos recursos computacionales, lo que reduce el tiempo de entrenamiento y el consumo energético. También facilita la validación de los modelos, ya que su estructura interpretable hace más sencillo su análisis y mantenimiento, alcanzando con esto el desarrollo de soluciones más rápidas, accesibles y sostenibles desde el punto de vista económico.

Aspectos sociales

Desde una perspectiva social, los datos sintéticos pueden ayudar a reducir desigualdades en el acceso a la inteligencia artificial. Instituciones con menos recursos pueden generar datos de calidad sin depender de grandes bases de datos, democratizando así la investigación. Por otro lado, si estos datos sintéticos se diseñan correctamente, pueden mejorar la equidad de los modelos, ayudando a que funcionen mejor para grupos minoritarios [23]. A su vez, la explicabilidad obtenida a partir de las redes KAN implementadas en este estudio favorece la confianza por parte de profesionales y pacientes en el uso de IA en el campo médico al ofrecer una mayor comprensión sobre cómo se toman las decisiones.

Aspectos medioambientales

Uno de los retos actuales de la inteligencia artificial es su impacto ambiental, debido al alto consumo energético del entrenamiento de modelos complejos. Este proyecto aborda este problema utilizando redes KAN, que son más ligeras y eficientes que otras arquitecturas como las redes neuronales profundas tradicionales, como se ha visto en los análisis desplegados durante el estudio. Gracias a su diseño, estas redes requieren menos cómputo, reduciendo tanto su consumo eléctrico como sus emisiones asociadas, promoviendo así un avance hacia una inteligencia artificial con menor huella ecológica, sin sacrificar ni precisión ni utilidad. Esto resulta crucial para alinear el desarrollo tecnológico con los Objetivos de Desarrollo Sostenible (ODS), en particular con el objetivo 12 sobre producción y consumo responsables.

A.3 Análisis detallado de alguno de los principales impactos

En el contexto de este proyecto, donde se trabaja con datos médicos sintéticos y modelos generativos explicables, se puede considerar que, entre los impactos analizados, el ético es el más relevante. En relación con el problema de la privacidad, los datos clínicos se encuentran estrictamente protegidos por distintas leyes, como el Reglamento General de Protección de Datos (RGPD) en la Unión Europea [23], lo cual restringe notablemente el acceso a los mismos. La investigación requiere acceso a esta información, por lo que los datos sintéticos emergen como una potente solución al generar registros parecidos a los reales sin contener información de personas reales, reduciendo el riesgo de filtraciones [25]. El uso de redes KAN y su diseño específico contribuye a esto mencionado, buscando generalizar patrones sin memorizar ejemplos concretos [24].

Destaca también la transparencia como otro de los aspectos esenciales a analizar. En comparación con otros modelos cuya compresnión es altamente compleja, las KAN permiten visualizar y entender la relación funcional aprendida entre variables. Esta interpretabilidad no solo aumenta la confianza de los profesionales y pacientes de la salud en las herramientas de IA, sino que también facilita el cumplimiento de regulaciones emergentes que exigen explicaciones sobre estas decisiones automáticas. La Unión Europea, con su propuesta de Ley de IA (AI Act) [26], catalogará los sistemas de generación de datos para salud como potencialmente de alto riesgo, requiriendo medidas de transparencia, documentación y supervisión humana. El enfoque de este Trabajo de Fin de Grado se alinea tanto con estas exigencias como con las presentes en otras normativas vigentes, como el RGDP, pudiendo explicar cómo se generan los datos sintéticos.

En conjunto, el proyecto promueve una inteligencia artificial que protege la privacidad y es transparente por diseño, aspectos realmente importantes en aplicaciones médicas.

A.4 Conclusiones

Este análisis de los impactos ético, económico, social y medioambiental demuestra que este proyecto ha integrado criterios de sostenibilidad y responsabilidad de forma completa, generando datos sintéticos de forma responsable y combinando innovación técnica con principios éticos y sostenibles. Este enfoque ha permitido establecer a lo largo del estudio las bases necesarias para avanzar hacia una inteligencia artificial más justa, explicable y respetuosa con el entorno.

B Presupuesto económico

COSTE DE MANO DE OBRA (coste directo)	**Horas**	**Precio/hora**	**Total**
	600	15 €	**9.000 €**

COSTE DE RECURSOS MATERIALES (coste directo)	**Precio de compra**	**Uso en meses**	**Amortización (en años)**	**Total**
Ordenador personal (Software incluido)	1.500,00 €	6	5	**150,00 €**
Impresora láser	500,00 €	6	5	**50,00 €**
Otro equipamiento	–	–	–	–

COSTE TOTAL DE RECURSOS MATERIALES	**200,00 €**

GASTOS GENERALES (costes indirectos)	15 %	sobre CD	**1.380,00 €**
BENEFICIO INDUSTRIAL	6 %	sobre CD+CI	**634,80 €**

MATERIAL FUNGIBLE	
Impresión	**100,00 €**
Encuadernación	**300,00 €**

SUBTOTAL PRESUPUESTO		**11.614,80 €**
IVA APLICABLE	21 %	**2.439,11 €**
TOTAL PRESUPUESTO		**14.053,91 €**

C Tablas y gráficas:

C.1 Anexo:

C.2 Gráficas y tablas TSTR-TRTS modelo TVAE:

Tabla 5.1: Evaluación TSTR y TRTS por arquitectura y modelo con Random Forest (TVAE vs KAN)

Arquitectura	Evaluación	Modelo	Accuracy	F1	Precision	Recall	ROC AUC
64-32	TSTR	TVAE	0.890	0.9126	0.8723	0.9567	0.9470
64-32	TSTR	KAN	0.896	0.9167	0.8827	0.9533	0.9600
64-32	TRTS	TVAE	0.921	0.9355	0.9598	0.9124	0.9743
64-32	TRTS	KAN	0.9255	0.9357	0.9982	0.8807	0.9868
32-16	TSTR	TVAE	0.876	0.9022	0.8563	0.9533	0.9490
32-16	TSTR	KAN	0.912	0.9276	0.9156	0.9400	0.9590
32-16	TRTS	TVAE	0.9055	0.9154	0.9525	0.8810	0.9719
32-16	TRTS	KAN	0.9325	0.9438	0.9913	0.9006	0.9898
16-8	TSTR	TVAE	0.870	0.8950	0.8683	0.9233	0.9416
16-8	TSTR	KAN	0.914	0.9296	0.9132	0.9467	0.9533
16-8	TRTS	TVAE	0.879	0.9055	0.8869	0.9250	0.9528
16-8	TRTS	KAN	0.9505	0.9543	0.9981	0.9142	0.9939
10-5	TSTR	TVAE	0.694	0.7446	0.7458	0.7433	0.7997
10-5	TSTR	KAN	0.922	0.9351	0.9336	0.9367	0.9444
10-5	TRTS	TVAE	0.843	0.8662	0.8819	0.8509	0.9226
10-5	TRTS	KAN	0.9225	0.9388	0.9690	0.9104	0.9825

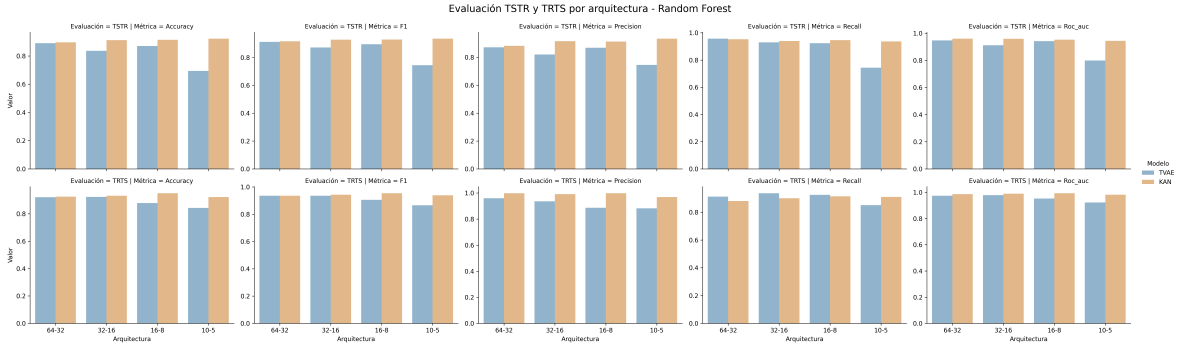


Figura 5.1: Evaluación TSTR y TRTS por arquitectura y métrica con Random Forest (TVAE vs KAN).

Tabla 5.2: Evaluación TSTR y TRTS por arquitectura y modelo con Gradient Boosting (TVAE vs KAN).

Arquitectura	Evaluación	Modelo	Accuracy	F1	Precision	Recall	ROC AUC
64-32	TSTR	TVAE	0.890	0.9123	0.8746	0.9533	0.9560
64-32	TSTR	KAN	0.908	0.9258	0.8969	0.9567	0.9584
64-32	TRTS	TVAE	0.903	0.9206	0.9477	0.8949	0.9688
64-32	TRTS	KAN	0.9085	0.9218	0.9738	0.8750	0.9836
32-16	TSTR	TVAE	0.850	0.8804	0.8440	0.9200	0.9166
32-16	TSTR	KAN	0.900	0.9183	0.9006	0.9367	0.9592
32-16	TRTS	TVAE	0.9145	0.9289	0.9262	0.9316	0.9752
32-16	TRTS	KAN	0.9190	0.9332	0.9700	0.8990	0.9835
16-8	TSTR	TVAE	0.858	0.8875	0.8459	0.9333	0.9301
16-8	TSTR	KAN	0.910	0.9266	0.9073	0.9467	0.9605
16-8	TRTS	TVAE	0.880	0.9063	0.8876	0.9258	0.9474
16-8	TRTS	KAN	0.9385	0.9439	0.9746	0.9151	0.9903
10-5	TSTR	TVAE	0.674	0.7341	0.7188	0.7500	0.7831
10-5	TSTR	KAN	0.912	0.9272	0.9211	0.9333	0.9507
10-5	TRTS	TVAE	0.8325	0.8568	0.8751	0.8392	0.9090
10-5	TRTS	KAN	0.8970	0.9215	0.9173	0.9257	0.9715

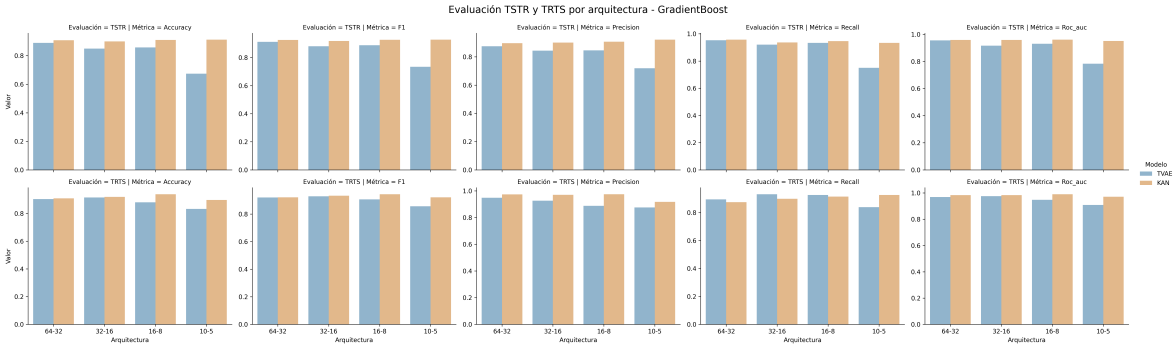


Figura 5.2: Evaluación TSTR y TRTS por arquitectura y métrica con Gradient Boosting (TVAE vs KAN).

Tabla 5.3: Evaluación TSTR y TRTS por arquitectura y modelo con Logistic Regression (TVAE vs KAN).

Arquitectura	Evaluación	Modelo	Accuracy	F1	Precision	Recall	ROC AUC
64-32	TSTR	TVAE	0.830	0.8653	0.8248	0.9100	0.9077
64-32	TSTR	KAN	0.900	0.9180	0.9032	0.9333	0.9748
64-32	TRTS	TVAE	0.9375	0.9490	0.9724	0.9268	0.9852
64-32	TRTS	KAN	0.9110	0.9296	0.9060	0.9545	0.9814
32-16	TSTR	TVAE	0.816	0.8521	0.8230	0.8833	0.8997
32-16	TSTR	KAN	0.852	0.8763	0.8792	0.8733	0.9338
32-16	TRTS	TVAE	0.945	0.9536	0.9642	0.9433	0.9901
32-16	TRTS	KAN	0.904	0.9266	0.8925	0.9634	0.9760
16-8	TSTR	TVAE	0.836	0.8723	0.8187	0.9333	0.9292
16-8	TSTR	KAN	0.842	0.8645	0.8905	0.8400	0.9157
16-8	TRTS	TVAE	0.901	0.9213	0.9184	0.9242	0.9719
16-8	TRTS	KAN	0.9185	0.9312	0.8910	0.9752	0.9901
10-5	TSTR	TVAE	0.726	0.7658	0.7860	0.7467	0.7826
10-5	TSTR	KAN	0.776	0.8205	0.7901	0.8533	0.8729
10-5	TRTS	TVAE	0.852	0.8736	0.8911	0.8568	0.9461
10-5	TRTS	KAN	0.927	0.9464	0.9096	0.9862	0.9804

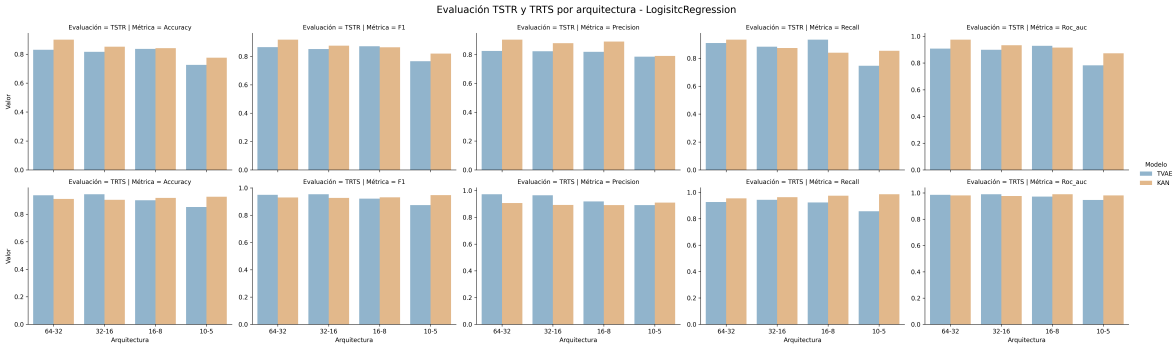


Figura 5.3: Evaluación TSTR y TRTS por arquitectura y métrica con Logistic Regression (TVAE vs KAN).

C.3 Gráficas TSTR-TRTS modelo CTGAN:

Tabla 5.4: Evaluación TSTR y TRTS por arquitectura y modelo con Random Forest (CTGAN vs KAN)

Arquitectura	Evaluación	Modelo	Accuracy	F1	Precision	Recall	ROC AUC
64-32	TSTR	CTGAN	0.486	0.5105	0.5956	0.4467	0.5032
64-32	TSTR	KAN	0.558	0.5790	0.6756	0.5067	0.6269
64-32	TRTS	CTGAN	0.528	0.6358	0.5479	0.7574	0.5012
64-32	TRTS	KAN	0.501	0.5275	0.4856	0.5772	0.5084
32-16	TSTR	CTGAN	0.470	0.5501	0.5606	0.5400	0.4493
32-16	TSTR	KAN	0.416	0.4183	0.5198	0.3500	0.4143
32-16	TRTS	CTGAN	0.500	0.6294	0.5208	0.7951	0.4659
32-16	TRTS	KAN	0.489	0.5346	0.4730	0.6147	0.5019
24-12	TSTR	CTGAN	0.574	0.6656	0.6291	0.7067	0.4984
24-12	TSTR	KAN	0.694	0.7330	0.7692	0.7000	0.7589
24-12	TRTS	CTGAN	0.4985	0.6225	0.5554	0.7080	0.4834
24-12	TRTS	KAN	0.5055	0.5531	0.4800	0.6525	0.5137
10-5	TSTR	CTGAN	0.466	0.5436	0.5579	0.5300	0.4367
10-5	TSTR	KAN	0.606	0.6425	0.7052	0.5900	0.6656
10-5	TRTS	CTGAN	0.466	0.4313	0.5000	0.3792	0.4677
10-5	TRTS	KAN	0.510	0.4053	0.4731	0.3546	0.5067

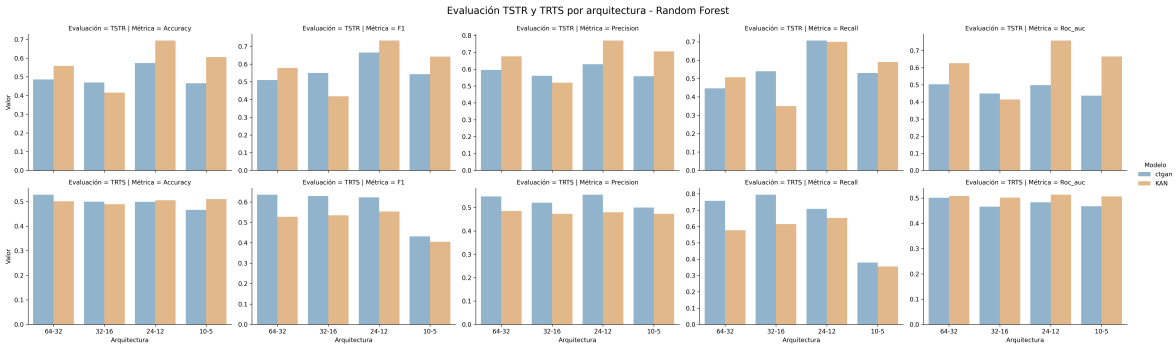


Figura 5.4: Evaluación TSTR y TRTS por arquitectura y métrica con Random Forest (CTGAN vs KAN).

Tabla 5.5: Evaluación TSTR y TRTS por arquitectura y modelo con Gradient Boosting (CTGAN vs KAN)

Arquitectura	Evaluación	Modelo	Accuracy	F1	Precision	Recall	ROC AUC
64-32	TSTR	CTGAN	0.506	0.5704	0.5964	0.5467	0.5089
64-32	TSTR	KAN	0.564	0.6007	0.6667	0.5467	0.6098
64-32	TRTS	CTGAN	0.5315	0.6406	0.5497	0.7675	0.4999
64-32	TRTS	KAN	0.496	0.5205	0.4811	0.5668	0.5047
32-16	TSTR	CTGAN	0.464	0.5363	0.5576	0.5167	0.4663
32-16	TSTR	KAN	0.480	0.4942	0.5935	0.4233	0.5110
32-16	TRTS	CTGAN	0.4945	0.6273	0.5176	0.7961	0.4516
32-16	TRTS	KAN	0.4885	0.5284	0.4720	0.6000	0.4985
24-12	TSTR	CTGAN	0.526	0.6404	0.5877	0.7033	0.5203
24-12	TSTR	KAN	0.650	0.6903	0.7358	0.6500	0.7291
24-12	TRTS	CTGAN	0.5040	0.6299	0.5582	0.7226	0.4868
24-12	TRTS	KAN	0.4995	0.5389	0.4745	0.6237	0.5136
10-5	TSTR	CTGAN	0.432	0.5251	0.5268	0.5233	0.4173
10-5	TSTR	KAN	0.704	0.7329	0.7992	0.6767	0.7570
10-5	TRTS	CTGAN	0.4640	0.4328	0.4976	0.3830	0.4612
10-5	TRTS	KAN	0.5145	0.4083	0.4793	0.3556	0.5031

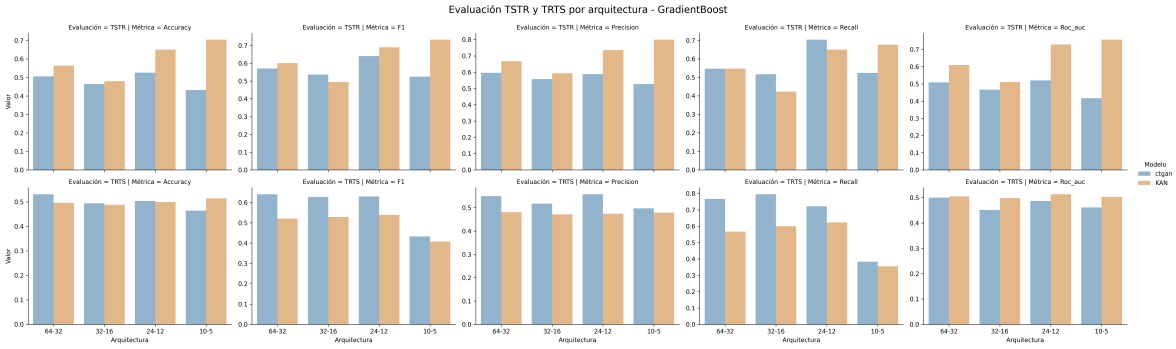


Figura 5.5: Evaluación TSTR y TRTS por arquitectura y métrica con Gradient Boosting (CTGAN vs KAN).

Tabla 5.6: Evaluación TSTR y TRTS por arquitectura y modelo con Logistic Regression (CTGAN vs KAN)

Arquitectura	Evaluación	Modelo	Accuracy	F1	Precision	Recall	ROC AUC
64-32	TSTR	CTGAN	0.458	0.5622	0.5455	0.5800	0.3782
64-32	TSTR	KAN	0.628	0.5991	0.8476	0.4633	0.7606
64-32	TRTS	CTGAN	0.516	0.6325	0.5388	0.7656	0.4868
64-32	TRTS	KAN	0.4985	0.5544	0.4852	0.6466	0.5126
32-16	TSTR	CTGAN	0.482	0.5350	0.5798	0.4967	0.4617
32-16	TSTR	KAN	0.544	0.4795	0.7609	0.3500	0.6359
32-16	TRTS	CTGAN	0.5065	0.6357	0.5250	0.8054	0.4680
32-16	TRTS	KAN	0.4960	0.5567	0.4799	0.6628	0.5038
24-12	TSTR	CTGAN	0.602	0.7069	0.6332	0.8000	0.5570
24-12	TSTR	KAN	0.580	0.5047	0.8629	0.3567	0.7179
24-12	TRTS	CTGAN	0.5005	0.6194	0.5580	0.6961	0.4595
24-12	TRTS	KAN	0.4945	0.5591	0.4731	0.6834	0.5135
10-5	TSTR	CTGAN	0.446	0.5345	0.5390	0.5300	0.3950
10-5	TSTR	KAN	0.542	0.4318	0.8447	0.2900	0.6811
10-5	TRTS	CTGAN	0.4765	0.4639	0.5119	0.4242	0.4707
10-5	TRTS	KAN	0.4955	0.5056	0.4695	0.5478	0.5068

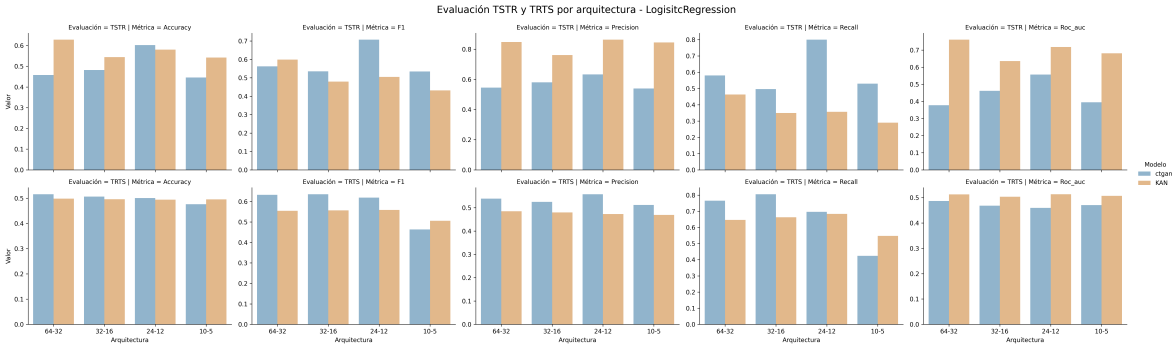


Figura 5.6: Evaluación TSTR y TRTS por arquitectura y métrica con Logistic Regression (CTGAN vs KAN).