UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN


Biblioteca
UNIVERSITAS Miguel Hernández

# "IMPORTANCIA DEL PREPROCESAMIENTO DE DATOS EN PROBLEMAS DE PRONÓSTICO CLÍNICO"

TRABAJO FIN DE GRADO

Octubre - 2025

AUTOR: Carlos Peñalver Pérez

DIRECTOR: Antonio Peñalver Benavent

# AGRADECIMIENTOS

Quería dar gracias a:

Aquellos compañeros del Grado que han pasado a ser grandes amistades.

Aquella comida antes de comenzar la primera clase que inició los primeros lazos con compañeros de distintos cursos, los cuales han sido un apoyo durante estos años y seguirán siéndolo.

A cada uno de los profesores y profesoras que permiten aprender tanto más allá de la teoría dada en clase. En especial, mi tutor de TFG Álex que me inspiró a querer adentrarme en el mundo del Análisis de Datos.

Los amigos de toda la vida y familia, que han sido la base para llegar hasta aquí.

Biblioteca
UNIVERSITAS Miguel Hernández

# RESUMEN

Durante los últimos años, la aplicación de técnicas de aprendizaje automático en el ámbito sanitario ha permitido lograr sistemas de apoyo a la decisión clínica capaces de mejorar el diagnóstico y pronóstico de diversas enfermedades. Sin embargo, es importante ser consciente de la calidad de los datos, ya que esta influye directamente en el rendimiento de los modelos predictivos.

Esta investigación tiene como objetivo analizar la importancia de distintas técnicas de preprocesamiento de datos en problemas de pronóstico clínico, evaluando su impacto sobre el rendimiento de distintos algoritmos. Para ello, se han utilizado varios conjuntos de datos médicos públicos con sus respectivas técnicas de limpieza, codificación y normalización, para su posterior análisis.

Los resultados muestran que un preprocesamiento adecuado mejora la precisión y estabilidad de los modelos, especialmente en algoritmos sensibles a la escala y distribución de datos. El estudio evidencia que no solo incrementa se obtiene un incremento de exactitud, sino que también reduce el sesgo y variabilidad del pronóstico.

Este trabajo contribuye a resaltar la necesidad de una fase de preprocesamiento previa al desarrollo de sistemas de apoyo a la decisión en el ámbito clínico.

# ÍNDICE DE CONTENIDO

# ÍNDICE DE FIGURAS

**Capítulo 5: Conclusiones y trabajos futuros**

Biblioteca
UNIVERSITAS Miguel Hernández

# ÍNDICE DE TABLAS



Biblioteca
UNIVERSITAS Miguel Hernández

Biblioteca
UNIVERSITAS Miguel Hernández

# Capítulo 1: Introducción

## 1.1 - ENTORNO DE APLICACIÓN

En la actualidad, el sector sanitario está en constante evolución impulsado por la digitalización y el crecimiento descontrolado de los datos clínicos. La información de historiales médicos, el desarrollo de distintos dispositivos de monitorización y un mayor número de atención telemática han generado un gran volumen de información que, si es bien gestionada y se analiza correctamente, puede mejorar significativamente la toma de decisiones médicas. Sin embargo, la disponibilidad de estos datos no garantiza su utilidad, ya que se debe aplicar técnicas de preprocesamiento y análisis adecuadas para extraer el conocimiento relevante y reducir el impacto de aquellos datos incompletos, inconsistentes o ruidosos.

Conociendo este contexto, la ciencia de datos y el aprendizaje automático han cobrado una gran relevancia en la medicina, facilitando el desarrollo de modelos predictivos capaces de asistir en el diagnóstico, pronóstico y tratamiento de diversas enfermedades. Pero es importante tener en cuenta lo siguiente: la calidad de los datos utilizados en estos modelos es determinante en su rendimiento y precisión. Un preprocesamiento deficiente puede dar lugar a sesgos, errores en las predicciones y decisiones clínicas poco fiables. Es por ello que garantizar una correcta limpieza, normalización y selección de datos es un paso esencial para optimizar el desempeño de los algoritmos y obtener resultados útiles.

Nos encontramos ante múltiples desafíos con el uso de modelos de aprendizaje automático en el ámbito clínico, desde la variabilidad en la procedencia y formato de los datos hasta la necesidad de cumplir con regulaciones y ética. A pesar de estas dificultades, la implementación de diferentes técnicas avanzadas de preprocesamiento ha demostrado ser una estrategia clave para mejorar la precisión en los sistemas de apoyo a la decisión médica. Al aplicar estos preprocesamientos a la predicción de enfermedades crónicas y evaluar el riesgo de eventos adversos, pueden contribuir a una atención médica más eficiente y personalizada.

Este trabajo se enmarca en esta realidad, abordando la importancia del preprocesamiento de datos en problemas de pronóstico clínico. Tomando distintos conjuntos de datos médicos, se analizará el impacto que tiene el preprocesamiento en la precisión de modelos predictivos, proporcionando evidencia sobre la necesidad de una correcta manipulación de los datos antes de su uso con algoritmos de aprendizaje automático.

## 1.2 - JUSTIFICACIÓN DEL PROYECTO

El análisis de datos y sus distintas aplicaciones me fascinaron cuando descubrí lo que se podía hacer con ellos y su importancia para prácticamente cualquier sector. En especial, descubrir su aplicación tanto en competiciones como en el ámbito clínico fue determinante para decidir a qué me gustaría dedicarme.

Poniendo el foco en el campo de la salud, en el que pretendemos cuidar del ciudadano, trabajar con datos que no están correctamente preprocesados, puede derivar en que los distintos análisis que llevemos a cabo se alejen de la realidad y no puedan ser usados según su propósito. Mejorando la calidad de los datos antes de usarlos en modelos predictivos conseguimos tener unos resultados con mayor precisión, en algunos casos muy notable como veremos en esta investigación. Si conseguimos mejores métricas en el modelo, se traducirá en un diagnóstico más rápido, lo que significa ahorro en tiempo y presupuesto, factores que pueden llegar a salvar vidas.

Aunque términos como Inteligencia Artificial y Machine Learning son más comunes hoy en día, su aplicación en medicina no es nueva. Uno de los grandes problemas al que se enfrenta este sector al implementar estos métodos en la toma de decisiones es la falta de calidad de los datos y la necesidad de herramientas fiables.

Con esta investigación, se demostrará cómo un buen tratamiento de los datos antes de ser usados en un modelo eleva su precisión, aportando un gran valor a mi formación. Además, puede beneficiar a profesionales de la salud, investigadores y desarrolladores de sistemas de apoyo a la decisión, entre otros.

## 1.3 - OBJETIVOS

Una vez ya planteada la investigación de la importancia del preprocesamiento de datos en problemas de pronóstico clínico, a continuación, se exponen los objetivos que se pretenden alcanzar.

Objetivos principales:

- Analizar el impacto del preprocesamiento: demostrar cómo distintas técnicas de preprocesamiento influyen en el rendimiento de modelos predictivos.

- Evaluar modelos de Machine Learning: aplicar distintos algoritmos sobre datos clínicos con y sin preprocesamiento para comparar sus métricas.

- Visualizar resultados de forma clara: mostrar el impacto del preprocesamiento mediante gráficas, tablas o comparaciones fácilmente interpretables.

- Ayudar a la toma de decisiones clínicas: evidenciar que un buen tratamiento de datos contribuye a modelos más fiables para el apoyo a decisiones médicas.

Objetivos secundarios:

- Estudiar técnicas de preprocesamiento: investigar y aplicar métodos como limpieza de datos, normalización, codificación de variables, tratamiento de valores perdidos y balanceo de clases.

- Explorar distintos datasets clínicos: seleccionar, entender y preparar diferentes conjuntos de datos enfocados en problemas de salud como diabetes, ataques al corazón y tiroides.

- Comparar métricas de evaluación: utilizar métricas como accuracy, F1-score o AUC para valorar el rendimiento de los modelos.

Objetivos personales:

- Aplicar conocimientos adquiridos: integrar los conocimientos adquiridos durante el grado, especialmente Minería de Datos y desarrollo en Python.

- Profundizar en el análisis de datos clínicos: ganar experiencia en un campo con proyección como la salud digital.

- Mejorar habilidades prácticas: reforzar el uso de herramientas y librerías como pandas, scikit-learn, matplotlib, entre otras.

Objetivo transversal:

- Elaborar una guía práctica del preprocesamiento clínico: construir una referencia clara que muestre el valor del preprocesamiento en proyectos reales de análisis de datos en medicina, con aplicaciones potenciales en entornos reales.

## 1.4 - QUÉ NO SE PRETENDE

En este bloque se pretende acortar el alcance del trabajo y aclarar aquello que no forma parte de los objetivos, aunque esté relacionado con el tema.

- Comparar la eficiencia de los algoritmos: el objetivo no es determinar si hay un algoritmo mejor que otro, sino analizar cómo el preprocesamiento afecta a su rendimiento.

- Diseñar un sistema de apoyo a la decisión completo: aunque el trabajo se enmarca en ese contexto, no se desarrollará una herramienta final de soporte clínico.

- Profundizar en el tratamiento clínico de las enfermedades: los datos utilizados son anónimos y no se interpretarán desde un punto de vista médico o terapéutico.

# Capítulo 2: Antecedentes y estado de la cuestión

En el ámbito de la salud, cualquier ayuda es bienvenida, pero es fundamental garantizar que sea de calidad para que realmente cumpla su propósito. Desde que comenzamos a usar la tecnología en centros médicos, se ha almacenado una cantidad inmensa de datos que podemos emplear a nuestro favor. El problema surge cuando queremos utilizar estos datos y nos encontramos con que gran parte de esa información no es suficiente para cumplir su función.

Es en este punto donde nos damos cuenta de la importancia del preprocesamiento: no basta con haber almacenado tantos datos si luego no se pueden emplear de forma adecuada. Ante esta necesidad de tratar los datos, hoy en día existen muchas implementaciones en lenguajes de programación que permiten, por ejemplo, eliminar columnas vacías, normalizar los datos o realizar una selección de características.

Este planteamiento servirá como introducción para exponer el problema y las herramientas que se utilizarán en esta investigación para resolverlo, sentando así las bases para el desarrollo del proyecto.

## 2.1 - SITUACIÓN ACTUAL

Ya se ha mencionado la importancia de la calidad de los datos, pero todavía no se ha descrito cómo evaluar esta calidad. [2.1] Una propuesta es hacerlo en seis dimensiones: precisión, plenitud, consistencia, accesibilidad, alcance e inmediatez.

También es normal que surja la siguiente pregunta: ¿Qué causa una mala calidad de los datos? Se debe a una variedad de factores, entre ellos podemos encontrar: falta de estandarización, debido a la falta de formatos, definiciones y protocolos; falta de validación de datos, inevitable por la entrada manual de datos; integración de datos en varios sistemas, lo que puede generar errores, duplicaciones e inconsistencias.

Estos factores afectan directamente a la capacidad de construir modelos fiables, lo que refuerza la necesidad del preprocesamiento en el trabajo de los modelos predictivos.

## 2.2 - HERRAMIENTAS DISPONIBLES

A continuación, se mostrarán y describirán los distintos lenguajes de programación y librerías para llevar a cabo esta investigación. Es importante hacer una buena elección, ya que puede garantizar un mejor resultado en menor tiempo.

Un lenguaje de programación es un conjunto de reglas que permiten a los programadores dar instrucciones a un ordenador. Los lenguajes de programación [2.2] son la base de los programas y aplicaciones que usamos. Existen distintos lenguajes, cada uno con su propio vocabulario y sintaxis. Se diferencia en dos tipos de lenguajes de programación: bajo nivel y alto nivel. En los lenguajes de bajo nivel se incluyen lenguajes ensamblador y máquina, este primero con una lista de instrucciones básicas y difícil de leer, y el segundo que se entiende directamente por la CPU del ordenador. Los lenguajes de alto nivel están diseñados para ser fáciles de leer y entender. Un programador escribe el código en alto nivel para traducirlo a un formato que la máquina pueda ejecutar, compuesto por instrucciones en binario.

La programación es una tarea que puede llegar a requerir bastante tiempo y afectar a la eficiencia del desarrollo. Es en este punto donde las librerías de programación resultan muy útiles. Una librería de programación [2.3] es una colección de código desarrollado previamente que se puede utilizar para desarrollar software de manera más ágil. Estas son colecciones de código reutilizables para resolver problemas comunes. Con el uso de estas librerías, un desarrollador no tiene que construir un software desde cero. En esta investigación, las librerías serán una pieza clave para agilizar el desarrollo, facilitar el tratamiento de datos y simplificar la construcción de los modelos predictivos.

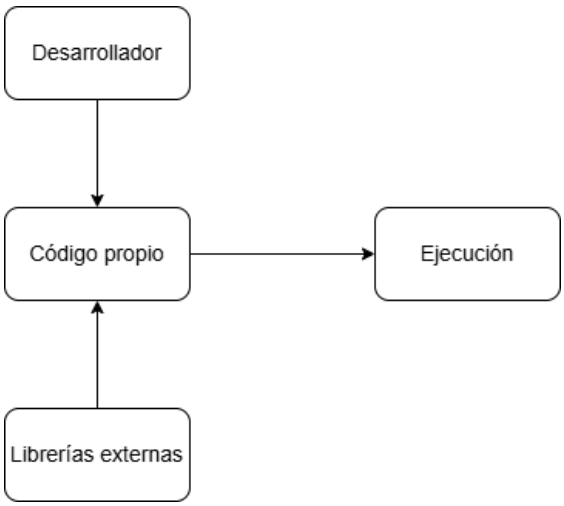


Figura 2.1: Integración de Librerías

### 2.2.1 - Lenguaje de programación: Python

Python es un lenguaje de programación de alto nivel, interpretado y multiparadigma, cuya primera versión fue desarrollada por Guido van Rossum en 1991. Cuenta con un diseño que prioriza la legibilidad del código, esto permite que los desarrolladores puedan escribir programas con mayor limpieza y comprensibilidad, reduciendo la probabilidad de errores y facilitando su mantenimiento [2.4]. Gracias a esta filosofía y a su simplicidad, Python ha sido ampliamente adoptado tanto en el ámbito académico como en el profesional.

En los últimos años, Python ha alcanzado una posición muy alta entre los lenguajes más utilizados del mundo. Como ejemplo, la encuesta para desarrolladores de Stack Overflow de 2023, en la ocupa el segundo lugar, solo por detrás de JavaScript [2.5]. Este crecimiento viene de la mano con su versatilidad, que lo hace útil para una amplia gama de aplicaciones, desde desarrollo web  hasta automatización de tareas, análisis de datos, inteligencia artificial, y ciencia de datos.

Una de las principales razones de su éxito en entornos de análisis es la disponibilidad de librerías, como Pandas para manipular datos estructurados, Numpy para cálculo numérico, Scikit-learn que incluye algoritmos de aprendizaje automático o Matplotlib para visualización. Con estas herramientas, se puede desarrollar un flujo completo de procesamiento, análisis y modelado de datos sin cambiar de lenguaje.

En el contexto de esta investigación, Python es un candidato a herramienta principal por su equilibrio entre facilidad de uso y potencia. Las tareas de preprocesamiento de datos clínicos pueden implementarse con muy pocas líneas de código gracias a las funcionalidades proporcionadas por sus librerías. Además, su buena integración con algoritmos de machine learning permite observar rápidamente cómo afectan estas transformaciones al rendimiento de los modelos.

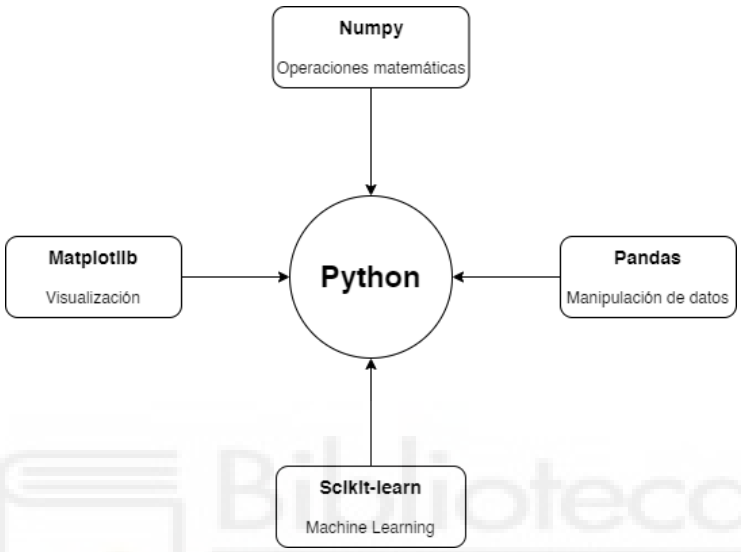


Figura 2.2: Ecosistema de Python en Ciencia de Datos

### 2.2.2 - Lenguaje de programación: R

R es un lenguaje de programación y entorno de software especializado en análisis estadístico, la visualización de datos y modelado matemático. Fue creado entre Robert Gentleman y Ross Ihaka en la Universidad de Auckland, en Nueva Zelanda , en 1993. Surgió como alternativa de código abierto al lenguaje S [2.6]. Desde entonces, R se ha consolidado como una herramienta fundamental en diferentes campos como la bioestadística, la epidemiología o la investigación médica, donde el tratamiento de datos es esencial.

Una de sus principales ventajas es su enfoque nativo hacia el análisis estadístico. Cuenta con un ecosistema de paquetes disponibles de CRAN, los usuarios pueden acceder a herramientas potentes y con una buena documentación. De entre estas herramientas se destaca Dplyr para la manipulación estructurada de datos, Ggplot para la visualización gráfica, Caret para la implementación de modelos de machine learning, o Tidyr para la preparación y limpieza de conjuntos de datos [2.7]. La existencia de estos paquetes, los cuales son prácticamente idénticos

a las librerías de Python, permiten una experiencia de análisis consistente sin importar el lenguaje elegido.

R también es reconocido por su capacidad de generar informes reproducibles gracias a R Markdown, una herramienta que permite integrar análisis, código y texto en el mismo documento [2.8]. Esta característica lo hace especialmente útil para contextos académicos y clínicos.

Para esta investigación, R es una alternativa sólida a Python. Su orientación estadística, amplio repertorio de paquetes especializados y su uso extendido en publicaciones científicas lo colocan como una opción a considerar.

### 2.2.3 - Librería: Pandas

Pandas es una librería de código abierto para Python. Fue diseñada por Wes McKinney en 2008 para la manipulación y análisis de datos estructurados. Desde entonces, es un pilar del ecosistema de ciencia de datos en Python [2.9].

Su principal fortaleza es el uso de dos estructuras de datos eficientes y fáciles de trabajar: las Series y DataFrames. Una Serie es un vector unidimensional, mientras que los DataFrames son tablas bidimensionales similares a las hojas de cálculo o bases de datos relacionales [2.10].

Dentro del contexto de esta investigación, Pandas desempeña un papel fundamental en las tareas de preprocesamiento de datos clínicos. Con sus funciones, podemos conseguir:

- Importar datos desde múltiples formatos.
- Detectar y tratar valores nulos o inconsistentes.
- Codificar variables categóricas.
- Normalizar columnas numéricas.
- Preparar el dataset final que será utilizado por los modelos Machine Learning.

Además, Pandas se integra sin problema con otras librerías del entorno Python, consiguiendo de esta manera un flujo de trabajo continuo, desde la carga de datos hasta la visualización de resultados.


Figura 2.3: Flujo de trabajo con pandas

### 2.2.4 - Librería: Scikit-learn

Scikit-learn es una de las librerías más populares de Python para la implementación de algoritmos de aprendizaje automático. Fue desarrollada en 2007 como parte de un proyecto de Google Summer of Code, por David Cournapeau. Desde ese entonces, fue mantenida por la comunidad de código abierto y respaldada por algunas instituciones como Inria y la fundación NumFOCUS [2.11].

En esta librería se puede encontrar una gran colección de herramientas para clasificación, regresión, clustering, reducción de dimensionalidad, validación cruzada y selección de características, todo a través de una API sencilla y coherente. Entre sus ventajas, encontramos la aplicación de algoritmos potentes en pocas líneas de código, ideal para los prototipos rápidos necesarios en esta investigación.

En el contexto de este trabajo, Scikit-learn se puede emplear para:

- Dividir los datos en conjunto de entrenamiento y prueba.
- Aplicar transformaciones como la estandarización o codificación.
- Entrenar y evaluar modelos de machine learning.
- Calcular métricas de rendimiento como precisión, F1-score o matriz de confusión.

Scikit-learn permite implementar todo el ciclo de análisis de forma estructurada y reproducible, gracias también a su integración con Pandas y Numpy.

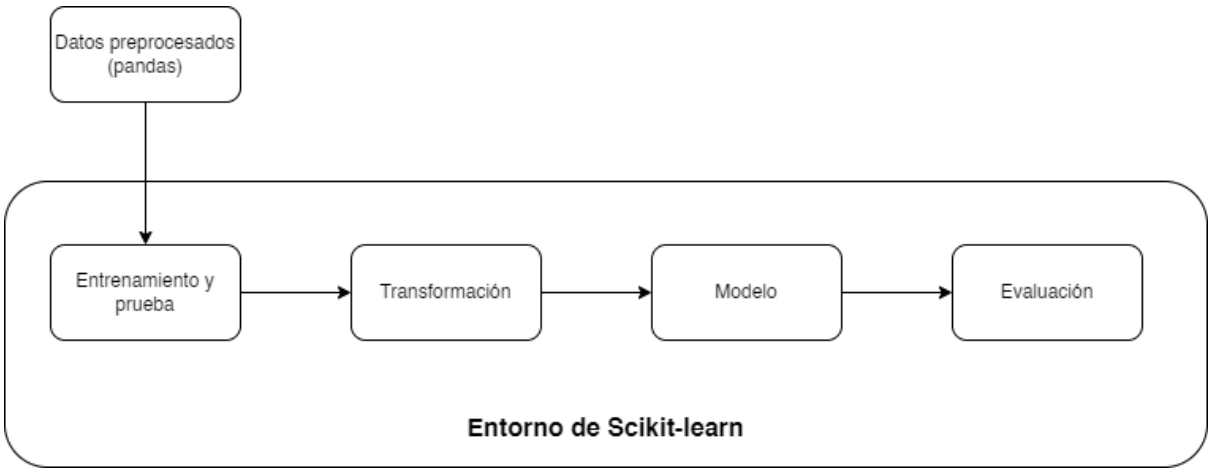


Figura 2.4: Flujo de trabajo con Scikit-learn

### 2.2.5 - Librería: Matplotlib

Matplotlib es una librería de visualización para Python que permite crear gráficos estáticos, interactivos y animados de alta calidad. Fue desarrollada en 2003 por John D. Hunter. Lo que pretendía su autor es ofrecer una alternativa a MATLAB para crear gráficos científicos en Python [2.12]. Desde su desarrollo, se fue convirtiendo en una de las herramientas más utilizadas en la exploración de datos, presentación de resultados y analizar comportamientos de modelos.

Matplotlib destaca por su flexibilidad, ya que permite generar desde diagramas de barras, histogramas o gráficos de dispersión, hasta tipos de visualizaciones más avanzadas como mapa de calor, gráficos de líneas múltiples o configuraciones personalizadas. Si se compara con otras librerías de mayor nivel como Seaborn o Plotly, su nivel de control sobre cada elemento gráfico lo hace una opción muy versátil [2.13].

En esta investigación, Matplotlib puede usarse para visualizar los efectos del preprocesamiento sobre los datos clínicos y representar resultados de modelos predictivos. También se debe conocer su integración con otras librerías, hasta el punto de crear gráficos directamente desde estructuras como DataFrames de Pandas.

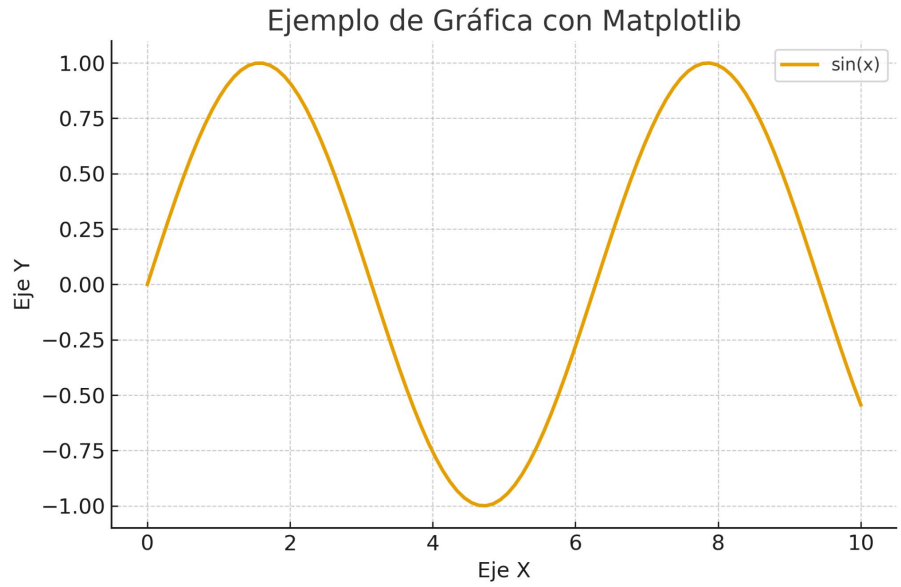


Figura 2.5: Ejemplo de Gráfica con Matplotlib

### 2.2.6 - Librería: Numpy

Numpy (Numerical Python) es una librería funcional para el cálculo funcional en Python, desarrollada por Travis Oliphant en 2005, como una evolución de Numeric y Numarray. Cuenta con estructuras de datos eficientes y funciones optimizadas, con las cuales logra procesar arreglos multidimensionales, operaciones matemáticas, álgebra lineal y transformaciones vectorizadas. Actualmente es adoptada como estándar para operaciones numéricas en ciencia de datos, ingeniería y computación científica [2.14].

La base de trabajo de esta librería es el objeto ndarray, que permite trabajar con colecciones de datos numéricos de forma mucho más rápida y eficiente que las listas nativas de Python. Con ayuda de este formato y operaciones vectorizadas, se elimina la necesidad de bucles explícitos, reduciendo el tiempo de ejecución y mejorando la legibilidad del código [2.15].

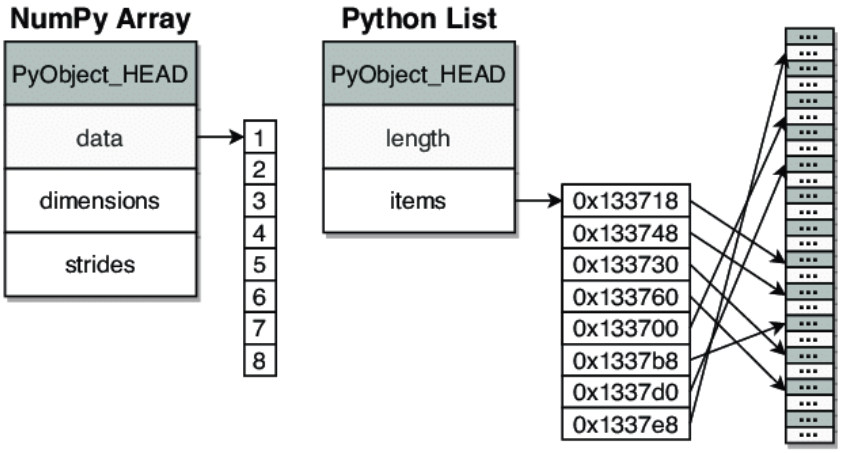


Figura 2.6: Comparación de ndarray y list en Pyhton

Aunque para esta investigación no se exprimirá todo el potencial de Numpy, esta librería actúa como infraestructura numérica de bajo nivel para otras como Pandas, Scikit-learn y Matplotlib y muchas otras. En el contexto de esta investigación, puede facilitar tareas como: cálculo de estadísticas, normalización y estandarización de columnas numéricas, creación de matrices u optimización de procesos repetitivos mediante operaciones vectorizadas.

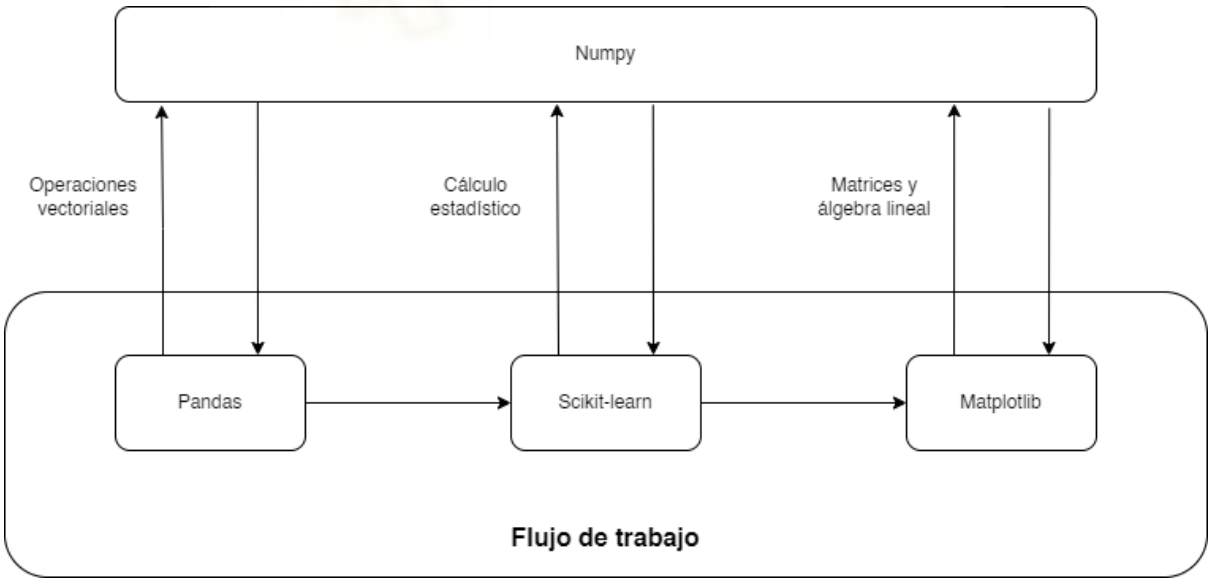


Figura 2.7: Rol de Numpy en el ecosistema

### 2.2.7 - Librería: XGBoost

XGBoost (Extreme Gradient Boosting) es una librería optimizada y con una eficiencia muy alta para la implementación del algoritmo gradient boosting basado en árboles de decisión. Fue desarrollado recientemente, en 2016 por Tianqi Chen, como parte de su trabajo de investigación. Desde ese momento, se ha convertido en una herramienta de referencia en competiciones de machine learning y proyectos aplicados, especialmente en análisis de datos estructurados [2.16] [2.17].

Sin profundizar demasiado en el algoritmo, XGBoost mejora el algoritmo de boosting evitando el sobreajuste, paralelización del entrenamiento, manejo eficiente de valores perdidos y soporte para grandes volúmenes de datos [2.17] [2.18]. Con estas características, consigue ser particularmente adecuado para aquellos entornos donde se requiere alto rendimiento predictivo.

En el contexto de esta investigación, XGBoost se postula como un modelo clave para analizar el impacto del preprocesamiento de datos clínicos. Además de sus altas capacidades, admite la incorporación de variables codificadas y escaladas, garantizando la integración con el resto de librerías.

### 2.2.8 - Librería: Imbalanced-learn

Imbalanced-learn (o Imblearn) es una librería que complementa a Scikit-learn diseñada para tratar conjuntos de datos con clases desbalanceadas [2.19]. Este es un problema común en muchos contextos clínicos, donde las clases positivas (existencia de enfermedad) pueden representar un porcentaje relativamente bajo. Cuando en un modelo predictivo se encuentra esta desproporción, se obtienen valores de precisión global altos pero un bajo rendimiento en la clase minoritaria, algo inaceptable en aplicaciones médicas.

Esta librería proporciona un conjunto de técnicas, como Oversampling y Undersampling, estrategias combinadas y herramientas para el preprocesamiento en presencia de desequilibrios. Con estas técnicas se permite modificar el conjunto de entrenamiento de manera que el modelo reciba una representación más equilibrada de las clases, mejorando su capacidad para detectar correctamente los casos minoritarios [2.20].

En este trabajo, con la implementación de Imblearn se puede conseguir un balanceo de clases gracias a sus técnicas, fundamental para datasets clínicos. La librería se integra fácilmente con Scikit-learn, lo que permite incorporar el balanceo como parte del flujo de entrenamiento sin romper la estructura.

### 2.2.9 - Técnicas de preprocesamiento

Las técnicas de preprocesamiento empleadas, incluyendo imputación de valores, codificación de variables categóricas, normalización y balanceo de clases, serán descritas en detalle en los capítulos correspondientes a la metodología y los experimentos. De este modo, se presentarán siempre en el contexto real en el que fueron aplicadas, facilitando así su comprensión y justificación.

## 2.3 - VALORACIÓN

Una vez analizadas las herramientas y alternativas disponibles para el desarrollo de la investigación, es necesario hacer una valoración que justifique las elecciones tomadas. El objetivo principal es asegurar que el entorno seleccionado sea el más adecuado para abordar con rigor el análisis del preprocesamiento de datos en contextos clínicos.

En primer lugar, se ha optado por el uso de Python como lenguaje de programación para el proyecto. Esta decisión se debe tanto a su popularidad dentro del ámbito clínico como a la amplia comunidad, así como su fácil integración con otros entornos de desarrollo. Con esta elección, se dispone de una gran cantidad de librerías especializadas en tratamiento y análisis de datos, permitiendo así cubrir todo el flujo del trabajo: carga de datos, limpieza y evaluación del modelo predictivo. Además, la curva de aprendizaje y sus recursos disponibles han facilitado su elección.

Por otro lado, se ha considerado la posibilidad de utilizar el lenguaje R, ya que es reconocido por su robustez en análisis estadísticos y su amplio uso en estudios médicos. Sin embargo, en el contexto de esta investigación, en la que se centra en la manipulación de datos clínicos y comparación del impacto en el preprocesamiento, Python presenta una mayor flexibilidad. Además, cuenta con mejor integración con librerías de Machine Learning como Scikit-learn o XGBoost.

En cuanto a las librerías específicas, la combinación de Pandas, Numpy y Scikit-learn, permite implementar un flujo de trabajo completo, estructurado y reproducible, lo cual es fundamental para este proyecto de carácter experimental con numerosas ejecuciones. Estas herramientas permiten trabajar con datos clínicos de forma eficiente, aplicar técnicas de preprocesamiento con pocos comandos, y conectar directamente con algoritmos de machine learning. Por su parte, matplotlib facilita la visualización de resultados, imprescindible para interpretar el efecto del preprocesamiento sobre los modelos. Además, el uso de librerías especializadas como XGBoost y Imblearn para tratar problemas de desbalance de clases añade profundidad y control a los experimentos realizados.

Biblioteca
UNIVERSITAS Miguel Hernández

# Capítulo 3: Hipótesis de trabajo

La finalidad de este capítulo es formular de forma clara y estructurada la hipótesis que guiará el resto del proyecto. Para ello, se identificarán las variables que intervienen en el estudio, se explicará de forma general el diseño del experimento y se presentarán los criterios para evaluar.

Con la formulación de esta hipótesis no sólo se pretende proporcionar un marco de referencia para el análisis posterior, sino que también permite establecer un hilo entre los objetivos planteados y la metodología experimental para dar sentido al siguiente capítulo.

De la misma forma que cualquier investigación, aquí se parte de una hipótesis que pretende ser contrastada mediante un proceso metodológico. Para este trabajo, centrado en el ámbito del análisis de datos clínicos, se busca evaluar el impacto que tienen las distintas técnicas de preprocesamiento sobre el rendimiento de los modelos predictivos. Tal y como se ha expuesto en los capítulos anteriores, existen múltiples herramientas y técnicas disponibles para mejorar la calidad de nuestros datos, pero de todas maneras, aún es necesario comprobar de manera empírica hasta qué punto estas transformaciones están influyendo en la eficacia de los algoritmos de aprendizaje automático.

## 3.1 - FORMULACIÓN DE LA HIPÓTESIS

Llegado a este punto, es el momento de presentar la hipótesis central de la investigación. Este es el momento clave para su presentación, una vez definida la orientación del trabajo y seleccionadas las herramientas que se emplearán para su desarrollo.

Se plantea que: “El preprocesamiento de los datos clínicos mejora de forma significativa el rendimiento de los modelos predictivos en términos de precisión, F1-score y otras métricas relevantes”

Durante la realización del experimento, se trabajará con distintas variables, las cuales podemos clasificar según su dependencia. Diferenciamos entre variables independientes y dependientes, siendo la primera la aplicación (o no) del preprocesamiento y la segunda el rendimiento del

modelo. En la siguiente tabla están recogidas las distintas variables a tener en cuenta durante la investigación:

| Tipo de variable | Nombre | Descripción |
|---|---|---|
| Independiente | Nivel de preprocesamiento | Presencia o ausencia de técnicas de preprocesamiento en los datos. |
| Dependiente | Accuracy | Proporción de predicciones correctas. |
| Dependiente | F1-score | Equilibrio entre precisión y recall. |
| Dependiente | Recall (sensibilidad) | Capacidad de detectar casos positivos correctamente. |

Tabla 3.1: Variables de la investigación

Tiene sentido pensar que unos datos incompletos o desbalanceados dificultan su interpretación, ya sea la humana o la digital para los modelos machine learning. Es por este motivo, que el preprocesamiento afecta directamente al rendimiento de los modelos, como ya hemos visto anteriormente, existen distintos problemas comunes en datos clínicos, como campos faltantes, ruido o desbalance. Además, tanto a la salud de los pacientes como su confianza en estas técnicas aumentará con modelos con mayor precisión para detectar distintas enfermedades.

Finalmente, se deben determinar los distintos criterios de validación para determinar si la hipótesis se cumple o no. A continuación, se recoge la propuesta de criterios que guiarán la evaluación:

| Criterio observado | Resultado esperado | Interpretación en caso contrario |
|---|---|---|

| | | |
|---|---|---|
| Mejora en métricas clave al aplicar el preprocesamiento | Aumento significativo respecto a la versión sin procesar | Hipótesis débil o no validada si la mejora es marginal o inexistente |
| Consistencia en varios datasets | Comportamiento repetido en distintos conjuntos de datos clínicos | Podría deberse al azar, sesgo del dataset o algoritmo implementado |
| Mejora notable en modelos más sensibles al preprocesamiento | Mejora visible especialmente en modelos que se benefician de estas técnicas | Si la mejora es nula incluso en estos modelos, podría indicar irrelevancia del preprocesamiento |
| Aumento en recall o sensibilidad en clases minoritarias (tras balanceo) | Incremento relevante tras aplicar técnicas como SMOTE | Si la clase minoritaria sigue siendo ignorada, el balanceo no está siendo efectivo |

Tabla 3.2: Criterios de validación

## 3.2 - FUNDAMENTACIÓN DE LA HIPÓTESIS

Este apartado tiene como finalidad justificar de forma teórica la hipótesis planteada. Ya se ha mencionado en distintas ocasiones los problemas que existen en los datos clínicos, pero ahora se debe explicar por qué el preprocesamiento puede ser una solución efectiva.

Antes de continuar, recordamos los problemas más recurrentes:

| **Problema** | **Descripción** |
|---|---|
| Valores faltantes | Información incompleta debido a errores de captura, fallos en registros o ausencias. |
| Desbalanceo de clases | Una clase está representada con mucha menor frecuencia. |
| Ruido o errores de entrada | Datos mal introducidos, inconsistentes o con |

| | |
|---|---|
| | medidas poco fiables. |
| Variables categóricas mal codificadas | Presencia de datos no numéricos sin estandarizar, lo que dificulta el aprendizaje automático. |

Tabla 3.3: Problemas en datos clínicos

Tenemos distintas técnicas que pueden influir de forma positiva en el rendimiento de modelos machine learning, estas son:

- Imputación de valores faltantes: deducir los valores faltantes de la parte conocida de los datos. [3.1]

- Normalización: garantiza la uniformidad de las magnitudes numéricas de las características. [3.2]

- Codificación de variables categóricas: transformar variables con atributos discretos a formatos numéricos para los algoritmos. [3.3]

- Balanceo de clases: equilibrar el número de muestras para cada clase. [3.4]

Durante esta investigación, se estarán usando datos clínicos de fuentes abiertas, de carácter público, pero representan perfectamente los escenarios reales de pacientes con diagnósticos médicos. El objetivo con este tipo de datasets es reflejar los desafíos que aparecen en entornos clínicos reales: registros incompletos, errores en la introducción manual de información, variables con distribuciones muy desbalanceadas, entre otros.

En el contexto del trabajo, la calidad de los datos es un factor decisivo para obtener unos buenos resultados. Si se entrena un modelo predictivo con datos sin tratar, se está corriendo el riesgo de que el modelo aprenda patrones erróneos, irrelevantes o sesgados. Para poner este problema sobre una situación real, un modelo puede "aprender" a predecir simplemente que nadie está enfermo, con una precisión aparentemente alta, pero sin ninguna utilidad clínica real. Este

problema es muy común en datasets con clases desbalanceadas, algo sucede con frecuencia en el ámbito médico.

También debemos conocer los algoritmos que se utilizarán en esta investigación, los cuales son especialmente sensibles al formato y calidad de datos. Tenemos unos tipos de algoritmos que su debilidad es una mala normalización, mientras que otros se verán perjudicados por los valores nulos. Por tanto, el preprocesamiento no es solo recomendable, sino prácticamente necesario para obtener resultados fiables con estos algoritmos.

Teniendo en cuenta esta necesidad, la diferencia entre un modelo con y sin preprocesamiento no es únicamente deseable desde el punto de vista técnico, sino también relevante desde la práctica: con una mejora en recall o F1-score se puede lograr una mayor capacidad para detectar enfermedades, reducir falsos negativos y, en definitiva, tomar decisiones más acertadas.

## 3.3 - LIMITACIONES DEL ENFOQUE

Antes de saltar al siguiente capítulo y ver la metodología y sus resultados, deben localizar las limitaciones que se encontrarán en esta investigación. Es importante detectar estos aspectos, debido a la influencia que pueden tener en los resultados finales, además de reducir el alcance del estudio. El conocimiento de estos es un aspecto positivo para poder mejorar en estudios futuros. De entre las limitaciones que se han podido encontrar, se quiere destacar tres ellas, las que se pueden considerar que podrían lograr un mayor impacto. Antes de su presentación, es necesario aclarar que no tienen porqué aparecer en este trabajo, especialmente, aquellas relacionadas con los datos de terceros.

En primer lugar, están las limitaciones relacionadas con los propios datos que se emplearán para entrenar los modelos. Se trabajará con datasets públicos, lo que coloca sobre la mesa la posibilidad de una cantidad de registros pobre. Por cuestión de seguridad, los datos suelen estar anonimizados, algo que podría hacer imposible ciertas interpretaciones clínicas. Lo que se considera más importante, la calidad original de los datos, debido al desconocimiento del tratamiento previo. Existe la posibilidad de que el dataset ya haya pasado por algún tipo de depuración, o que algunos problemas no se puedan corregir, como errores de introducción manual.

Por otro lado, se cuentan con limitaciones del diseño experimental, ya que el número de algoritmos empleados ha sido reducido, para adecuarme a los plazos y extensión del TFG. No se ha aplicado validación cruzada completa, lo que puede suponer que los resultados pueden estar condicionados por la división concreta del conjunto de datos. Tampoco se ha aplicado un análisis estadístico formal para validar la significancia de las diferencias observadas entre las versiones con y sin preprocesamiento. Esta es una línea de mejora importante para trabajos posteriores.

Por último, el experimento se ha llevado a cabo en un tiempo restringido, lo que limita la profundidad o número de pruebas posibles.

Estas limitaciones expuestas no invalidan los resultados obtenidos, pero deben tenerse en cuenta al interpretarlos o extenderlos a otros contextos.

Biblioteca
UNIVERSITAS Miguel Hernández

# Capítulo 4: Metodología y resultados

## 4.1 - DESCRIPCIÓN DE LOS DATASETS

La búsqueda de datasets válidos puede llegar puede llegar a ser un desafío para alguien que no está familiarizado con el análisis de datos, pero existen sitios web en los que buscar fácilmente los datos. Para los datasets necesarios se ha recurrido a Kaggle [4.1] un repositorio inmenso de modelos publicados por la comunidad, datos y código. En esta comunidad podemos encontrar estudiantes, desarrolladores e investigadores, donde además de aprender, se celebran competiciones de Machine Learning. Para esta investigación, se seleccionarán varios datasets de Kaggle relacionados con la salud que se ajusten a las necesidades desarrolladas en capítulos anteriores.

### 4.1.1 - DATASET: DIABETES

El primer dataset seleccionado es Diabetes Dataset [4.2], creado por el Instituto Nacional de Diabetes y Enfermedades Digestivas y Renales. El objetivo de este conjunto de datos es predecir si un paciente tiene diabetes con distintas mediciones. Cuenta con 768 instancias para el trabajo.

En sus variables encontramos: 7 números enteros, 2 números reales. Podemos ver sus distintas variables en la siguiente tabla:

| Nombre de la variable | Descripción | Tipo |
|---|---|---|
| Pregnancies | Número de embarazos | Numérica entera |
| Glucose | Concentración de glucosa | Numérica entera |
| BloodPressure | Presión arterial diastólica | Numérica entera |

| | | |
|---|---|---|
| SkinThickness | Espesor del pliegue cutáneo del tríceps (mm) | Numérica entera |
| Insulin | Insulina sérica de 2 horas | Numérica entera |
| BMI | Índice de masa corporal | Numérica continua |
| DiabetesPedigreeFunction | Función que asigna la probabilidad de padecer diabetes a partir de la historia familiar | Numérica continua |
| Age | Edad del paciente en años | Numérica entera |
| Outcome | Clase, resultado positivo o negativo | Numérica entera |

Tabla 4.1: Variables Diabetes

### 4.1.2 - DATASET: HEALTHCARE DATASET STROKE DATA

Este es el segundo dataset con el que se trabajará en la investigación, sobre ataques al corazón [4.3], con un total de 5110 instancias. A diferencia del anterior, en este caso a penas se puede encontrar información sobre estos datos, simplemente el usuario que lo publicó. La elección de este dataset se debe a su variedad de tipos en las variables que lo componen.

En sus variables encontramos: 1 ID, 5 cadenas de texto, 3 número entero, 2 números decimales y 1 binaria. Podemos saber más de ellas en la siguiente tabla.

| **Nombre de la variable** | **Descripción** | **Tipo** |
|---|---|---|
| ID | Número identificativo. | ID |
| Gender | Género del paciente. | Categórica nominal |
| Age | Edad del paciente. | Numérica continua |

| | | |
|---|---|---|
| Hypertension | Indica si el paciente tiene hipertensión (0 = No, 1 = Sí). | Categórica binaria |
| Heart_disease | Indica si el paciente tiene enfermedades cardiacas previas (0 = No, 1 = Sí). | Categórica binaria |
| Ever_married | Indica si el paciente ha estado alguna vez casado. | Categórica binaria |
| Work_type | Tipo de ocupación del paciente. | Categórica nominal |
| Residence_type | Tipo de residencia del paciente (urbana o rural). | Categórica nominal |
| Avg_glusose | Nivel promedio de glucosa en sangre. | Numérica continua |
| BMI | Índice de masa corporal del paciente. | Numérica continua |
| Smoking_status | Estado del hábito tabáquico. | Categórica nominal |
| Stroke | Variable objetivo: indica si el paciente ha sufrido un accidente cerebrovascular (0 = No, 1 = Sí). | Categórica binaria |

Tabla 4.2: Variables de Healthcare dataset stroke data

Podemos observar que este dataset es notablemente más entendible, con unas variables más simples que se podrían obtener con una primera consulta al médico, lo que podría acelerar la detección del problema con un modelo fiable.

### 4.1.3 - DATASET: THYROID DISEASE DATA SET

En último lugar, un dataset que trata sobre distintas enfermedades relacionadas con la tiroides[4.4], el cual cuenta con 3772 entradas. El conjunto proviene del Instituto Garvan de Investigación Médica. También se puede descargar desde UCI, de la misma forma que el primer dataset que se ha presentado.

El dataset sigue la misma línea que los anteriores, recogiendo datos sobre los pacientes tanto básicos cómo avanzados. Estas son las variables que tiene:

| Nombre de la variable | Descripción | Tipo |
|---|---|---|
| Age | Edad del paciente. | Numérica continua |
| Sex | Sexo del paciente (M = masculino, F = femenino). | Categórica nominal |
| On_thyroxine | Si el paciente está actualmente tomando tiroxina. | Categórica binaria |
| Query_on_thyroxine | Si el paciente ha sido consultado respecto al uso de tiroxina. | Categórica binaria |
| On_antithyroid_medication | Si el paciente toma medicación antitiroidea. | Categórica binaria |
| Sick | Si el paciente tiene otra enfermedad (no tiroidea). | Categórica binaria |
| Pregnant | Si la paciente está embarazada. | Categórica binaria |
| Thyroid_surgery | Si ha tenido cirugía tiroidea. | Categórica binaria |
| I131_treatment | Si ha recibido tratamiento | Categórica binaria |

| | | |
|---|---|---|
| | con I131. | |
| Query_hypothyroid | Si se sospecha de hipotiroidismo. | Categórica binaria |
| Query_hyperthyroid | Si se sospecha de hipertiroidismo. | Categórica binaria |
| Lithium | Si está tomando litio. | Categórica binaria |
| Goitre | Presencia de bocio. | Categórica binaria |
| Tumor | Presencia de tumor. | Categórica binaria |
| Hypopituitary | Presencia de hipopituitarismo. | Categórica binaria |
| Psych | Presencia de trastorno psiquiátrico. | Categórica binaria |
| TSH | Nivel de TSH (hormona estimulante de tiroides). | Numérica continua |
| T3 | Nivel de T3 (triyodotironina). | Numérica continua |
| TT4 | Nivel total de T4 (tiroxina). | Numérica continua |
| T4U | Índice de captación de T4 (T4 uptake). | Numérica continua |
| FTI | Índice de tiroides libre (Free Thyroxine Index). | Numérica continua |
| TBG | Nivel de globulina fijadora de tiroxina (TBG). | Numérica continua |
| Referral_source | Fuente de derivación (ej. médico, otro hospital...). | Categórica nominal |

| | | |
|---|---|---|
| Class | Clase objetivo: diagnóstico | Categórica nominal |

Tabla 4.3: Variables de Thyroid Disease Data Set

Se puede observar como hay una gran variedad de tipos y un gran número de variables, con esto se tendrán más opciones a la hora preprocesar.

## 4.2 - PREPROCESAMIENTO APLICADO

Finalmente, en este punto se explicarán las distintas técnicas de preprocesamiento que se emplearán a los distintos datasets ya descritos. En primer lugar, se explicarán las técnicas que aparecen en todos los conjuntos seleccionados, para luego, dar paso a aquellas específicas de cada uno. Siempre que la técnica lo permita, se mostrará un ejemplo visual del cambio que sucede en el dataset.

### 4.2.1 - PREPROCESAMIENTO APLICADO: TÉCNICAS TRANSVERSALES COMUNES

De las técnicas empleadas en la investigación, se cuenta con cuatro que están presentes en los distintos casos, estas son:

- Balanceo de clases con SMOTE.
- Reducción de dimensionalidad con PCA.
- Selección de características con SelectKBest.
- Normalización de datos.

A continuación, una explicación de su uso, con los ejemplos adecuados para ayudar en su comprensión.

Cuando se quiere entrenar un modelo con la variable objetivo sin balancear se pueden encontrar algunos problemas, debido al patrón de datos de la clase dominante que supera a los de la clase

con menos frecuencia. Generalmente, este problema es común en conjuntos de datos clínicos, donde la clase con la frecuencia más baja es aquella que se intenta predecir. Un modelo construido con datos desequilibrados puede presentar una precisión muy alta, pero no predecir correctamente la clase que se espera, la de menor frecuencia. Esto puede generar una impresión de que el modelo funciona correctamente cuando en realidad no es así.

Para solucionar estos problemas de desequilibrio, se puede recurrir a dos soluciones que consisten en equilibrar los datos de la variable objetivo: undersampling y oversampling.

Undersampling es una técnica que consiste en mantener los datos de la clase de menor frecuencia y reducir la cantidad de los de la clase de mayor frecuencia, balanceando así la variable objetivo. La ventaja de usar undersampling es la reducción de almacenamiento de datos y tiempo de ejecución de código, debido a que la cantidad de datos será menor. Su contraparte es que al reducir la cantidad de datos, se traduce en una menor cantidad de información al modelo, por este motivo se descarta esta técnica de balanceo.

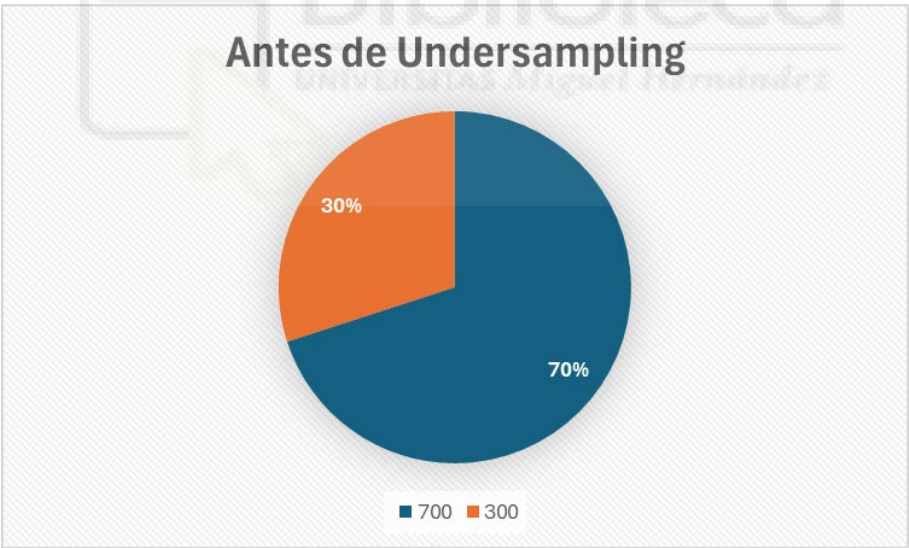


Figura 4.1: Variable objetivo antes de Undersampling

Como podemos observar en la imagen anterior, tenemos un gráfico con una relación en la variable objetivo desequilibrada, siendo la clase de menor frecuencia la indicada en color naranja, siendo ésta la que queremos predecir en este ejemplo. A continuación, los resultados de aplicar Undersampling.

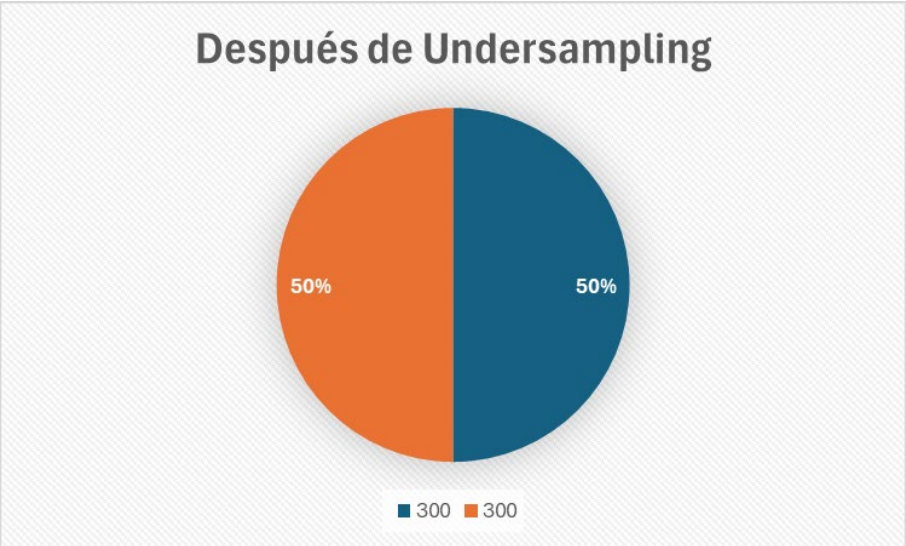


Figura 4.2: Variable objetivo después de Undersampling

Se observa cómo se consigue ese equilibrio entre clases a costa de eliminar 400 entradas que son azules.

Continuando ahora con Oversampling, que es la técnica inversa que su hermana, aumenta el número de registros de la clase con menor frecuencia. Para lograr esto, podemos duplicar aleatoriamente los registros de la clase con menos frecuencia, lo que puede afectar al modelo ya que habrá mucha información idéntica. Una ventaja de esta técnica es evitar que se pierda información de la clase con mayor frecuencia, pero el almacenamiento y procesamiento aumenta considerablemente.

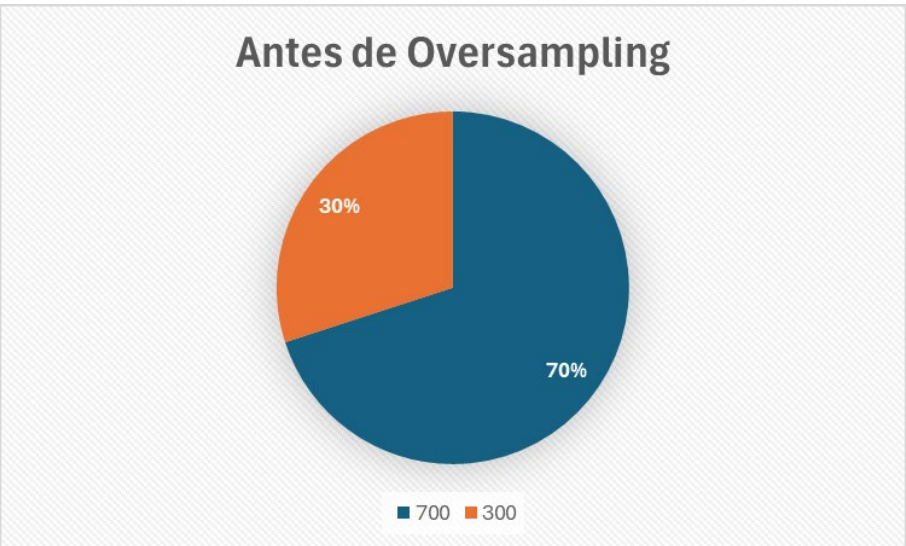

Figura 4.3: Variable objetivo antes de Oversampling

Podemos observar el mismo caso que el anterior, queremos predecir naranja, pero tenemos un desequilibrio notable hacia azul.

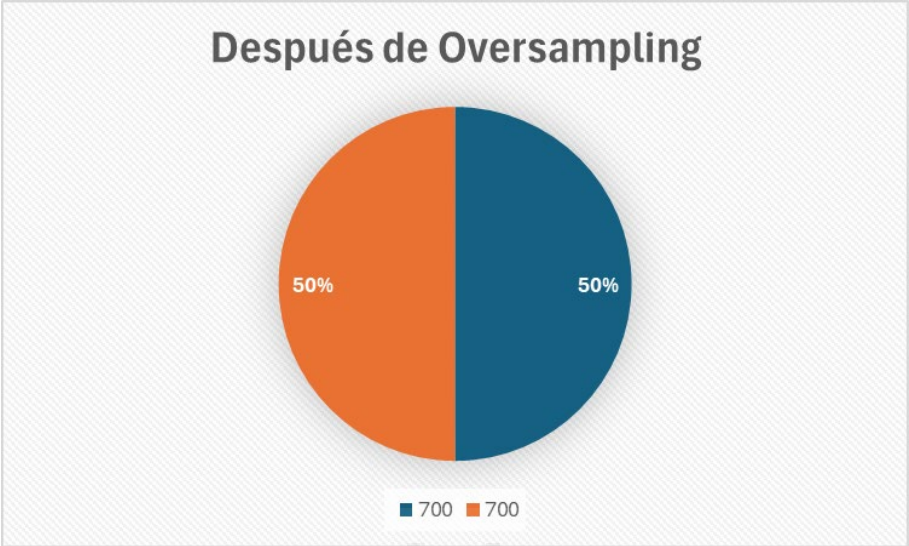


Figura 4.4: Variable objetivo después de Oversampling

Como se puede observar, se han equilibrado ambas clases, pero esta vez aumentando el número de datos totales. Ahora llega la pregunta: ¿Por qué usar Oversampling si realmente los datos son idénticos? Para evitar la repetición de estos datos, podemos usar la técnica SMOTE. [4.5] SMOTE (Synthetic Minority Over-sampling Technique) se presenta como una propuesta de Oversampling en la que la clase minoritaria aumenta sus registros creando ejemplos "sintéticos", en lugar de duplicar los existentes. Inspirado en técnicas de aumento de datos usadas en reconocimiento de caracteres, este enfoque actúa en el espacio de características, no en el espacio de datos. Para cada muestra minoritaria, se generan nuevos puntos a lo largo de los segmentos que la unen con sus k vecinos más cercanos, eligiendo aleatoriamente cuántos usar. La muestra sintética se obtiene interpolando entre una instancia y su vecino con un valor aleatorio entre 0 y 1. Así, se expande la región de decisión de la clase minoritaria, mejorando la generalización del modelo. [4.6]

La normalización es una forma específica de escalado de atributos que transforma su gama a una estándar. Cualquier técnica como esta, será necesaria cuando el conjunto de datos tiene

rangos variables, englobando diversas técnicas adaptadas a diferentes distribuciones de datos y requisitos del modelo.

Los datos normalizados aumentan el rendimiento y precisión de los modelos, especialmente a los algoritmos basados en métricas de distancia, como K-Nearest Neighbors. Se fomenta la estabilidad en el proceso de optimización, con una convergencia más rápida durante los entrenamientos basados en gradiente. Por otro lado, los datos tratados con esta técnica son más fáciles de interpretar y comprender. [4.7]

A continuación, un ejemplo de normalización de datos:

| **Datos sin normalizar** | | | **Datos normalizados** | | |
|---|---|---|---|---|---|
| **Paciente** | **Edad** | **Glucosa** | **Paciente** | **Edad** | **Glucosa** |
| A | 20 | 85 | A | -1,22 | -1,18 |
| B | 50 | 140 | B | 0 | -0,03 |
| C | 80 | 200 | C | 1,22 | 1,21 |

Tabla 4.5: Ejemplo de normalización de datos

Se puede observar en esta tabla el cambio de valor en datos tras el proceso de normalización. A simple vista, es algo difícil de comprender sólo con una tabla de valores. Si se crea una gráfica para cada versión de los datos, se apreciará mejor la curva de normalización.

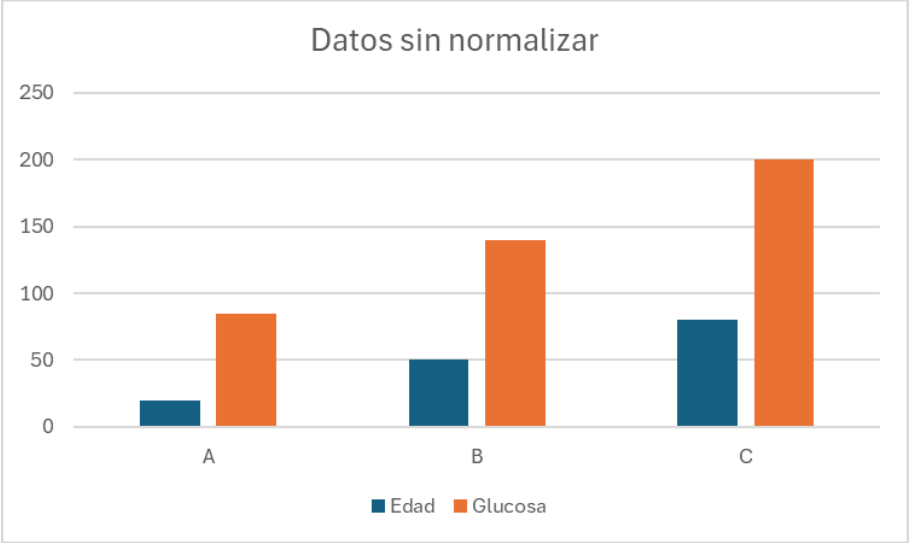


Figura 4.6: Datos sin normalizar

En esta primera gráfica se observa los datos previos a la normalización, se puede ver como aumenta de forma progresiva. Esto no tiene por qué ser así en todos los datasets, es un ejemplo.

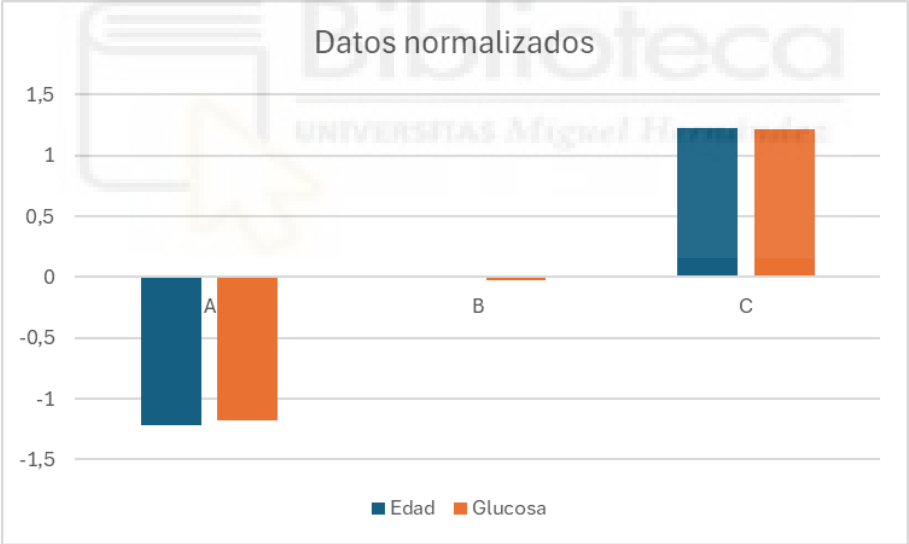


Figura 4.7: Datos normalizados

Tras el proceso de normalización y colocar los datos en una gráfica, se puede observar esa tendencia de distribución normal que se busca, teniendo en el centro de los datos el valor cercano a cero. Para este ejemplo solo se cuenta con tres entradas, pero al aumentar el número se puede observar con mayor claridad la curva.

La reducción de dimensionalidad es un problema común en el análisis de datos. A menudo, se trabaja con datos que tienen un número demasiado elevado de variables, las cuales pueden dificultar la detección de patrones y relaciones entre ellas. Aplicando esta técnica, se logra reducir el número de características de nuestros datos, manteniendo la información esencial, consiguiendo así una simplificación de datos y mejorar la eficiencia computacional de los modelos.

Una técnica popular de reducción de dimensionalidad en el Análisis de Componentes principales (PCA), que es la que se emplea en esta investigación. Esta técnica estadística reduce la dimensionalidad de los datos manteniendo la mayor cantidad de información posible. Explicándolo de una forma más coloquial, PCA transforma un conjunto de variables originales en un conjunto más pequeño de variables, que se conocen como Componentes Principales.

Para mostrar de forma breve su explicación matemática, se puede decir que su objetivo es encontrar la proyección de máxima varianza de los datos. Para lograrlo, encuentra los vectores propios de la matriz de covarianza. Estos vectores propios son direcciones en el espacio de las variables que tienen la propiedad de que al proyectar los datos en estas direcciones, se maximiza la varianza de los datos. Una vez encontrados los vectores propios, se utilizan para construir las componentes principales. La primera componente principal equivale a la dirección en el espacio de las variables que tiene mayor varianza de los datos, repitiendo hasta que han construido tantas componentes principales como variables originales. Los Componentes Principales con los valores propios más altos explican la mayor parte de la varianza de los datos y se consideran más importantes. [4.8]

Esta técnica resulta algo más complicada de comprender inclusive con un ejemplo, de todas maneras, a continuación, se muestran dos gráficos en los que se muestra el comportamiento de PCA.

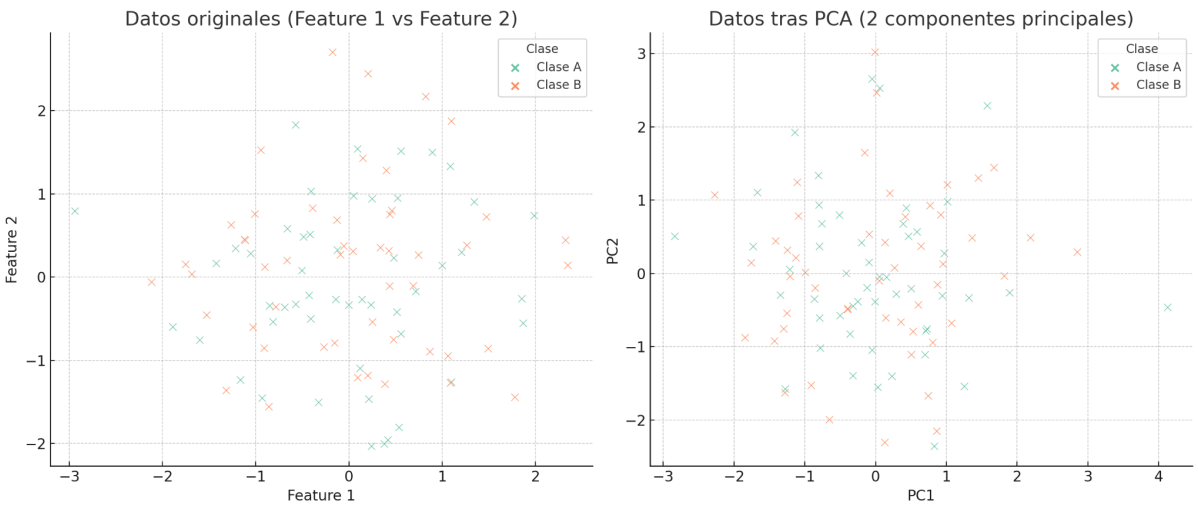


Figura 4.8: Comparativa del uso de PCA

A la izquierda se pueden visualizar dos atributos seleccionados arbitrariamente, es decir, existen más atributos pero solo podemos representar dos, pues son las dimensiones con las que se cuenta. Se observa que las clases no se separan bien y la distribución no parece significativa.

Tras aplicar PCA y observar el resultado, se puede apreciar que se ha proyectado los datos en un nuevo sistema de coordenadas que maximiza la varianza y mejora la separación entre clases.

Con la anterior figura se expresa de forma clara por qué se aplica PCA: extraer la información relevante dispersa en muchas dimensiones y transformarla en una representación más eficiente para el análisis y los modelos.

La selección de características es una técnica fundamental en machine learning que permite identificar y conservar las variables más relevantes de un conjunto de datos. Con esta reducción, no solo se consigue mejorar el rendimiento del modelo, sino que disminuye el riesgo de overfitting.

En este experimento, se ha utilizado una técnica basada en la información mutua entre cada variable independiente y la variable objetivo. Esta métrica evalúa la dependencia entre dos variables, lo que permite identificar cuáles aportan mayor información útil para la predicción.

Para aplicar esta estrategia en este trabajo, se ha utilizado el método SelectKBest, un técnica ampliamente utilizada por su simplicidad y eficiencia. Pertenece a los métodos “filter-based”, los cuales seleccionan las características en función de criterios estadísticos. SelectKBest asigna una puntuación a cada característica basándose en medidas estadísticas. Posteriormente, selecciona K variables con mayor puntuación, reduciendo de esta manera el conjunto de características a un subconjunto más manejable y relevante para los algoritmos. [4.9]

Para esta técnica, se puede observar en la siguiente imagen la selección de SelectKBest para un dataset que no se emplea en este proyecto. Se ve reflejado en el gráfico la puntuación otorgada por la técnica a cada característica.

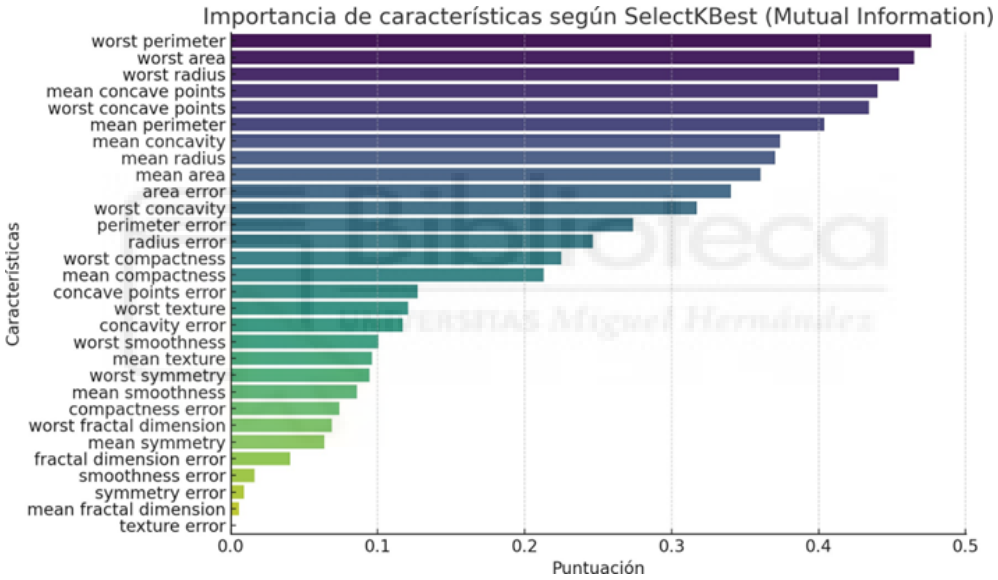


Figura 4.9: BMI antes de la imputación

## 4.2.2 - PREPROCESAMIENTO APLICADO: DIABETES

Este primer conjunto de datos es ampliamente conocido por su uso en problemas de clasificación binaria dentro del ámbito clínico. En este dataset se tiene como objetivo diagnosticar diabetes. Aunque el dataset está relativamente limpio y bien estructurado, presenta ciertos aspectos que justifican la aplicación de técnicas de preprocesamiento, para facilitar la interpretación y mejorar el rendimiento de los modelos.

Para evitar valores atípicos o registros erróneos, como pueden ser ceros en variables fisiológicamente imposibles, que afectarán al entrenamiento de los modelos, se sustituyen dichos valores por la mediana de cada uno. Con esta decisión se logra reducir un impacto negativo en la predicción y mejora la representatividad estadística de los datos sin alterar la tendencia. Más adelante se podrá ver un ejemplo visual de esta técnica ante un problema similar.

Por otro lado, tras la normalización, se ha aplicado una transformación de potencia, conocida como PowerTransformer. Su objetivo es reducir el sesgo estadístico presente en las variables. Este método ajusta las distribuciones a una forma más simétrica y normal, consiguiendo una mejor estabilidad numérica y el rendimiento de los modelos lineales.

### 4.2.3 - PREPROCESAMIENTO APLICADO: STROKE

El siguiente dataset también es conocido en el mundo del análisis de datos, está diseñado con la finalidad de predecir la probabilidad de que un paciente sufra un accidente cerebrovascular. Se trata de una clasificación binaria, en la que la variable objetivo indica si el paciente ha sufrido, con un 1, o no ha sufrido, con un 0, este problema. Por otro lado, este conjunto presenta una mayor diversidad de variables, por lo que se puede esperar que el preprocesamiento juegue un papel más relevante.

De la misma forma que en el apartado anterior, existen columnas que no necesitamos y lo único que aportarán será disminuir la precisión de nuestro modelo. Habiendo desarrollado anteriormente este proceso, sólo es necesario saber que la columna “id” será eliminada.

El dataset incluye algunos registros en su columna “BMI” (índice de masa corporal) que están vacíos. Esta variable es relevante en un análisis clínico, ya que puede estar asociada a enfermedades cardiovasculares y la presencia de valores nulos puede afectar negativamente a los modelos predictivos, debido a que son muchos los algoritmos que no pueden procesar entradas incompletas. Esta información nos lleva a tener que hacer un tratamiento adecuado de estos valores.

Para resolver este problema se ha optado por la imputación mediante la mediana, utilizando SimpleImputer de scikit-learn. Este método tiene un funcionamiento simple: reemplazar los valores faltantes por la mediana de los valores disponibles. El motivo de usar la mediana frente a la media es porque esta es más robusta frente a los valores atípicos. El índice de masa corporal puede presentar cierta dispersión o casos extremos, por este motivo se considera que mantiene una representación más fiel.

Es posible ver este tratamiento de datos comparando con gráficas de frecuencia. A continuación se muestra el cambio producido para este proyecto. Se debe observar el aumento del valor 28 tras la imputación, lo que significa que ese es el valor de la mediana. Se puede comprobar que la distribución sigue siendo similar a la anterior, con la ventaja de que ahora tenemos los valores nulos con información.

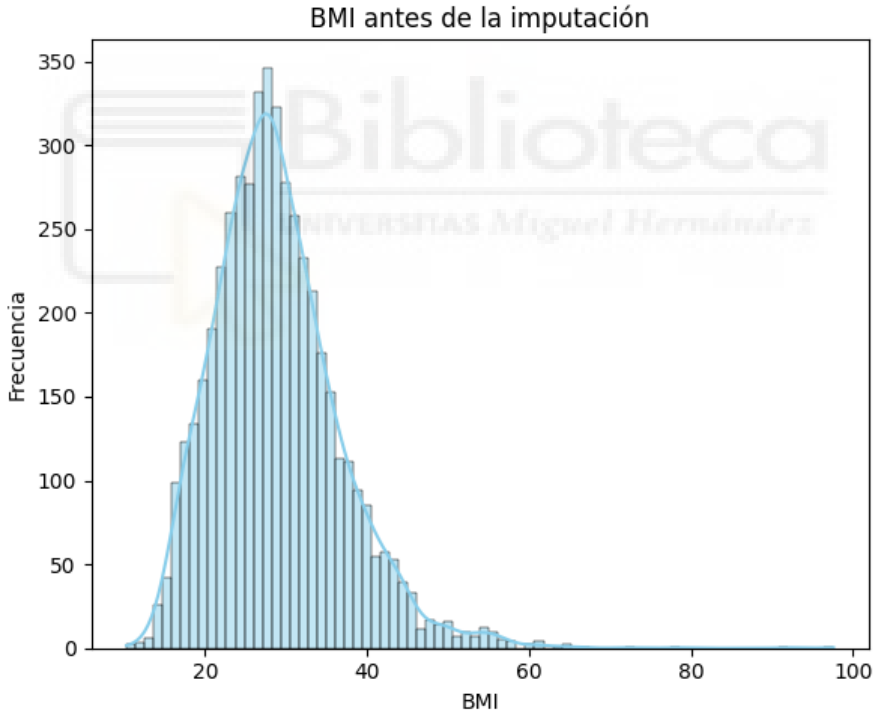


Figura 4.10: BMI antes de la imputación

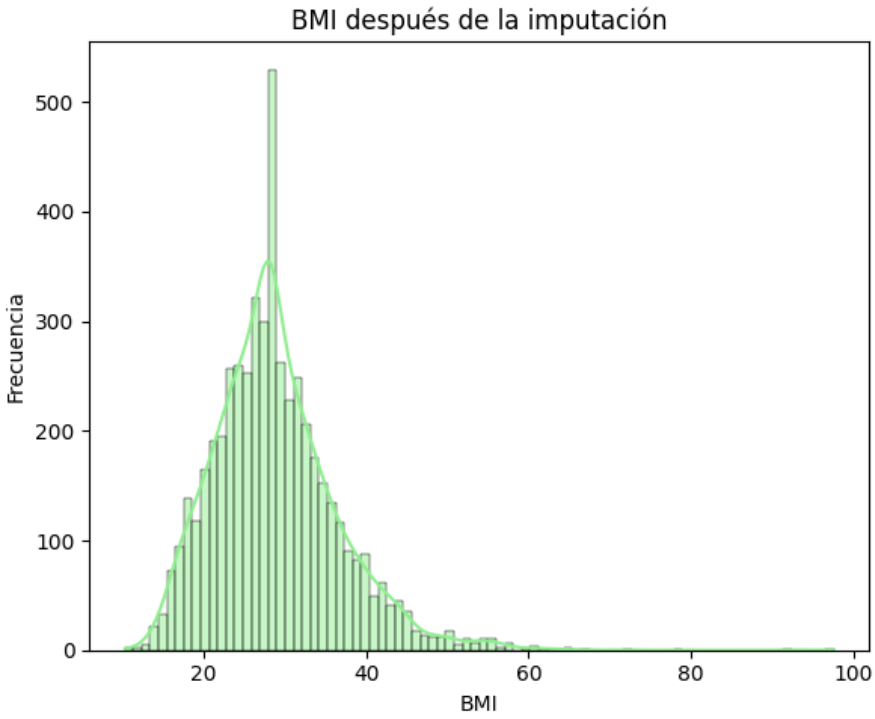


Figura 4.11: BMI después de la imputación

Por último, otro paso en el preprocesamiento es la codificación de variables categóricas. En concreto, las variables “gender”, “ever_married”, “work_type”, “Residence_type”, “smoking_status” contiene valores de tipo texto, lo que resulta incompatible con la mayoría algoritmos de aprendizaje automático.

Para solucionar esto, se puede usar la técnica One-Hot Encoding. Esta técnica consiste en transformar cada valor categórico en una nueva columna binaria (con 0 o 1), indicando la presencia o ausencia de esa categoría en cada fila. Es importante indicar en estas líneas de código la cláusula “drop_first=True”, para evitar la colinealidad y no perder información.

A continuación, una tabla que muestra el resultado de una transformación, para mayor comprensión de la modificación que se está llevando a cabo. Se puede observar como no se genera la columna gender_Female, ya que si no es Male ni Other, entonces será Female por descarte.

| ID | gender_Male | gender_Other |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 0 | 0 |
| 3 | 0 | 1 |

Tabla 4.12: Ejemplo de One-Hot Encoding con “drop_first=True”

### 4.2.4 - PREPROCESAMIENTO APLICADO: THYROID DISEASE

El conjunto de datos sobre esta enfermedad tiene como finalidad ayudar en la detección de distintas disfunciones en la glándula tiroides, como el hipotiroidismo o el hipertiroidismo. Su diferencia principal con el resto de datasets tratados es que este presenta una mayor complejidad por número de variables sumado a la diversidad de formatos y tipos de datos. Sus distintos tipos de variables y valores desconocidos o faltantes hacen que este dataset sea un caso muy bueno para aplicar una gran variedad de técnicas de procesamiento de datos.

Para comenzar, se aplican algunas técnicas que ya se han mencionado antes, como puede ser la imputación de valores faltantes, con el cambio de que esta vez se está reemplazando “?” por valores NaN, algo necesario para realizar la imputación correctamente. También se aplica la conversión de la variable objetivo a valores numéricos y codificamos las variables categóricas.

Por otro lado, de forma más específica de este dataset se aplica la eliminación de columnas completamente vacías. Este conjunto presentaba columnas que no tenían ningún valor, por eso motivo se considera mejor eliminarla completamente a añadir los datos como anteriormente.

En última instancia, se ha optado por separar características y variable objetivo. Antes de su desarrollo, es necesario saber que un conjunto de datos se suele organizar en una tabla que:

- Cada fila es una instancia.
- Cada columna es una variable.
- Una de esas columnas es la que queremos predecir.

Y para entrenar un modelo de aprendizaje automático es necesario:

- Un conjunto de características (X): variables independientes.
- Una variable objetivo (Y); lo que se pretende predecir.

Conociendo esta información, ya se puede explicar con mayor claridad el objetivo de esta técnica: guardar la columna de la variable objetivo en la variable "Y" y eliminarla de "X". De esta forma se evita que el modelo pueda hacer trampa y aprender directamente la respuesta, obteniendo de esta forma resultados engañosos y artificialmente altos que no correspondan con el rendimiento real del modelo.

## 4.3 - MODELOS ENTRENADOS

Tras realizar distintos métodos de preprocesamiento, es el momento de entrenar los distintos modelos de aprendizaje automático que han sido seleccionados en esta investigación. Se han escogido algoritmos de distinta naturaleza con el objetivo de analizar la diferencia de impacto en el trabajo del preprocesamiento.

En este apartado se presentan los modelos empleados, importante antes de evaluar las métricas obtenidas.

### 4.3.1 - SUPPORT VECTOR MACHINE (SVM)

Este primer algoritmo es proveniente de Sklearn, útil para problemas tanto de clasificación, como regresión. Su objetivo es encontrar un hiperplano de separación entre las instancias de dos clases. El hiperplano no es ni más ni menos que un producto escalar de un vector de variables. [4.10]

La diferencia que tiene SVM respecto a otros métodos de separación lineal, es que entre todos los posibles hiperplanos que se encargan de dividir las instancias en dos partes, se escoge aquel que ofrece un margen máximo. El margen que se comenta, es calculado como la máxima distancia entre las instancias frontera, tal como se muestra en la siguiente figura:

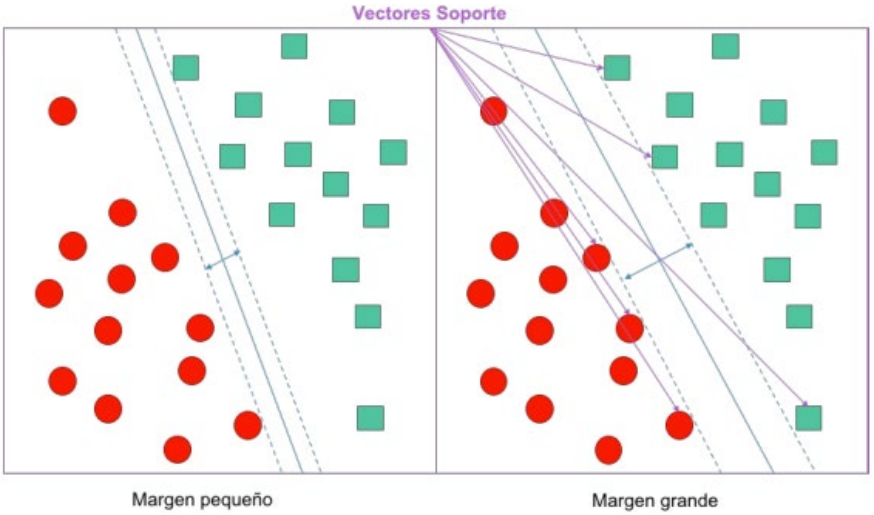


Figura 4.13: Margen de SVM, sacado de [4.10]

El nombre de esta técnica es determinado precisamente por las instancias en las que se "apoya" la frontera de decisión. Se trata de los puntos más cercanos al hiperplano, e influyen directamente en la orientación para alcanzar su margen máximo.

Tras esta introducción al algoritmo puede surgir la duda de cómo se obtiene esta orientación óptima del hiperplano. En el caso de SVM el parámetro más importante es el coste, denominado como C. Un valor bajo de coste acepta cometer un cierto número de errores de clasificación, bajando ligeramente la calidad de predicción, pero con una mejor generalización. Por otro lado, un valor alto permitirá un mejor ajuste del modelo sobre los datos, pero con mayor riesgo de sobreaprendizaje.

### 4.3.2 - K-NEAREST NEIGHBORS

El algoritmo KNN es un clasificador de aprendizaje supervisado no paramétrico, el método que emplea es la proximidad para realizar clasificaciones o predicciones sobre la agrupación de un punto de datos individual. Si bien se puede usar para problemas de regresión o clasificación, generalmente se usa como un algoritmo de clasificación, con la premisa de encontrar puntos similares cercanos.


Figura 4.14: Ejemplo de trabajo de KNN, sacado de [4.11]

El valor k en este modelo define cuántos vecinos se verificarán para determinar la clasificación del punto dado. Por ejemplo, si k=1, la instancia se asigna a la misma clase del vecino más cercano. La elección del valor k dependerá principalmente de los datos de entrada, ya que con datos con mayor ruido funcionarán mejor con valores más altos. El valor de k debe ser impar para evitar empates en la elección, como se puede ver en la imagen anterior. [4.11]

### 4.3.3 - REGRESIÓN LOGÍSTICA

La regresión logística es una técnica de análisis de clasificación lineal utilizada cuando se pretende conocer la relación entre una variable dependiente y una o más variables explicativas independientes, ya sean cualitativas o cuantitativas. Su objetivo es predecir la probabilidad de que ocurra un evento que caracteriza la variable dependiente (como puede ser un caso positivo en este estudio), conociendo los valores de las variables independientes. [4.12]

Esta regresión se modela de la misma forma que la regresión lineal, pero con la función sigmoidal, lo que dejaría el modelo con una gráfica similar a la de la imagen:

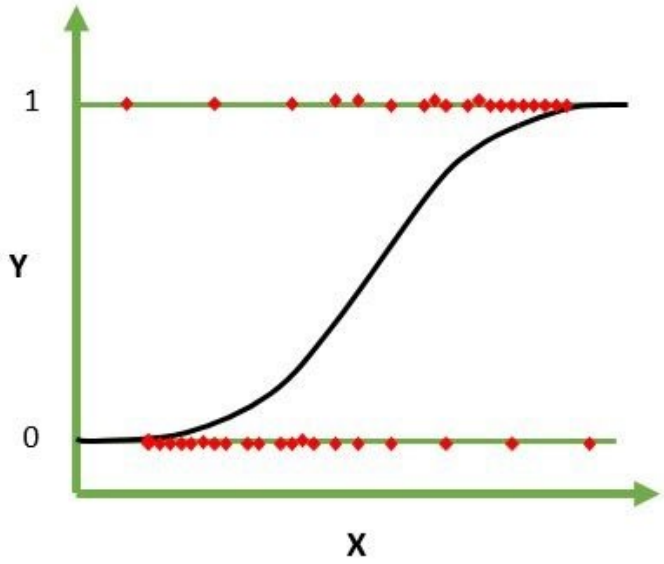


Figura 4.15: Función Sigmoidal de Regresión Logística, sacado de [4.13]

Aplicando esta función, la función objetivo nos dará valores entre 0 y 1 indicando si el valor pertenece a la clase 0 o 1, interpretando el resultado como negativo o positivo en el campo de esta investigación. [4.13]

### 4.3.4 - GRADIENT BOOSTING CON XGBOOST

Entre los modelos utilizados se incluye un modelo basado en Gradient Boosting, una técnica basada en combinar árboles de decisión de forma secuencial para reducir el error secuencial. En este proyecto se utilizó la implementación optimizada XGBoost, conocida por su mayor eficiencia.

XGBoost parte de una predicción inicial, con la que calcula los “residuos”. Una vez esta primera iteración, se crea un árbol de decisión con una puntuación similar a estos árboles residuales. Se realizan cálculos de la similitud de los cálculos y de la ganancia de similitud de la división posterior. Con estos datos se puede determinar una entidad y un umbral para un nodo. La salida del árbol se convierte de nuevo en residual para construir otro árbol, hasta que los residuales dejan de producirse o se termina el número de iteraciones especificado.

De una forma más coloquial: cada árbol subsiguiente aprende a partir de los árboles anteriores y no tiene asignado el mismo peso.

A la hora de predecir con este modelo, la salida de cada árbol multiplicado por una tasa de aprendizaje se suma a la predicción inicial para llegar a un valor final o una clasificación. [4.14]

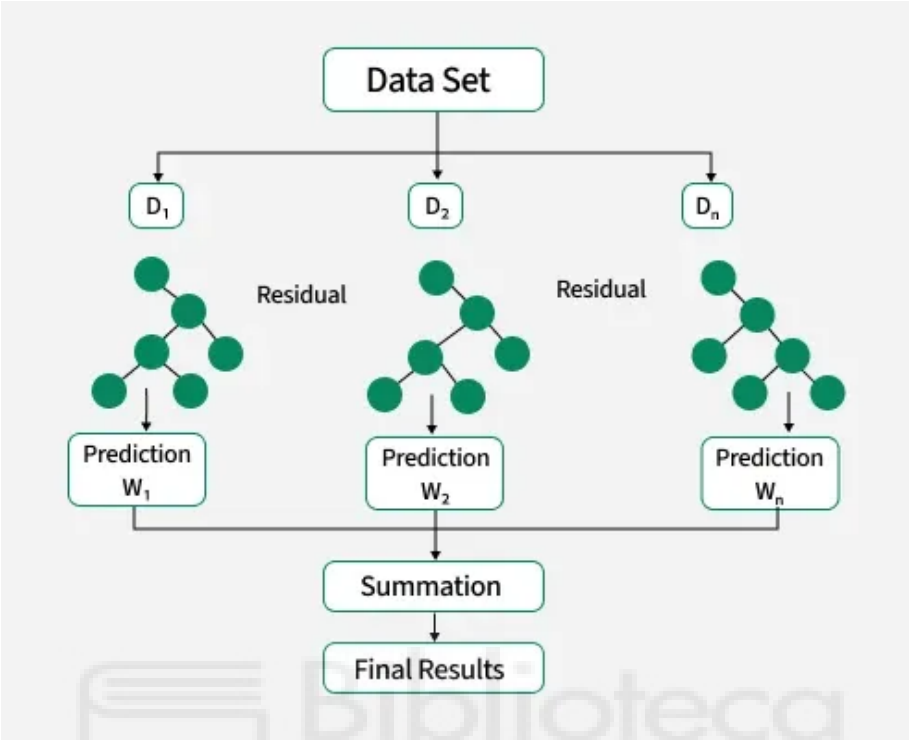


Figura 4.16: Esquema de XGBoost, sacado de [4.15]

### 4.3.5 - RANDOM FOREST

Random forest es otro algoritmo basado en árboles de decisión, esta vez bajo la técnica bagging y aleatoriedad de características para crear un bosque de árboles de decisión que no están correlacionados. Mientras que los árboles de decisión consideran todas las posibles divisiones de características, los bosques aleatorios sólo seleccionan un subconjunto de esas características.

Este tipo de algoritmo cuenta con tres hiperparámetros principales, los cuales deben de configurarse antes del entrenamiento: tamaño del nodo, cantidad de árboles y cantidad de características muestreadas.

El algoritmo de bosque aleatorio se compone de una colección de árboles de decisión y cada árbol en el conjunto está compuesto por una muestra de datos extraída de un conjunto de entrenamiento con reemplazo, llamado bootstrapping. De esa muestra de entrenamiento, un

tercio de ella se reserva como datos de prueba, lo que se conoce como la muestra fuera de bolsa, a la que se vuelve más adelante. Luego, se inyecta otra instancia de aleatoriedad mediante el embolsado de características, lo que agrega más diversidad al conjunto de datos y reduce la correlación entre los árboles de decisión. Dependiendo del tipo de problema, la determinación de la predicción variará. Para una tarea de regresión, se promedian los árboles de decisión individuales, y para una tarea de clasificación, el voto mayoritario, es decir, la variable categórica más frecuente, arrojará la clase prevista. Finalmente, la muestra fuera de la bolsa se utiliza para la validación cruzada, finalizando esa predicción. [4.16]

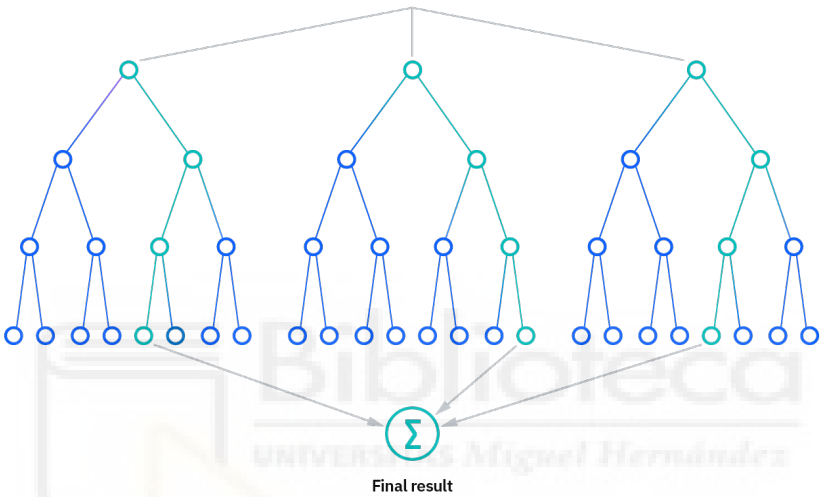


Figura 4.17: Representación de Random Forest, sacado de [4.16]

### 4.3.6 - NAÏVE BAYES

Naïve Bayes es un algoritmo de clasificación que trata de predecir la categoría de un dato mediante la probabilidad, suponiendo que todas las características son independientes entre sí.

Este algoritmo tiene unas suposiciones fundamentales que se deben cumplir con el preprocesamiento: [4.17]

- Independencia de características.
- Las características continuas se distribuyen normalmente.
- Las características discretas tienen distribuciones multinomiales.
- Las características son igualmente importantes.
- Sin datos faltantes.

A continuación, se presenta una simulación del trabajo del algoritmo:

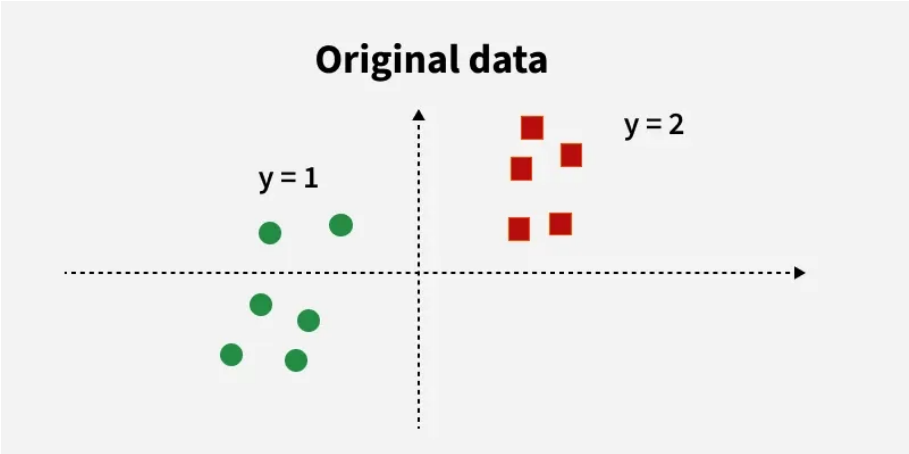


Figura 4.18: Naïve Bayes: Datos originales, sacado de [4.17]

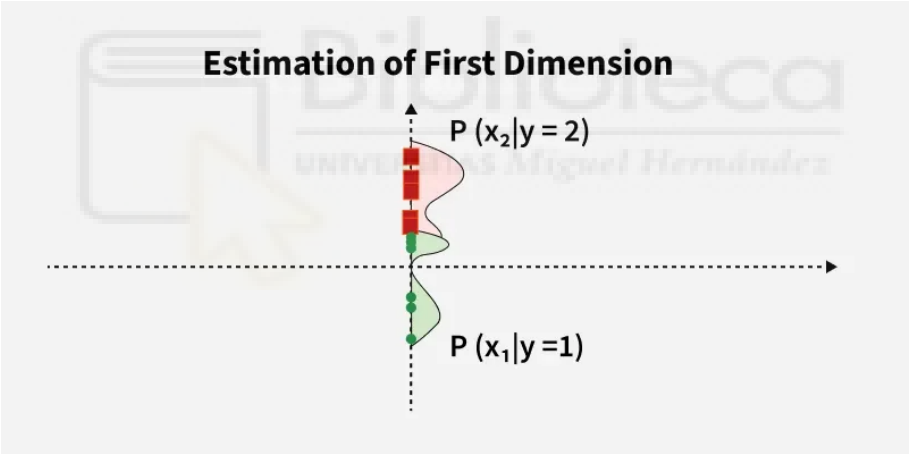


Figura 4.19: Naïve Bayes: Estimación de la primera dimensión, sacado de [4.17]

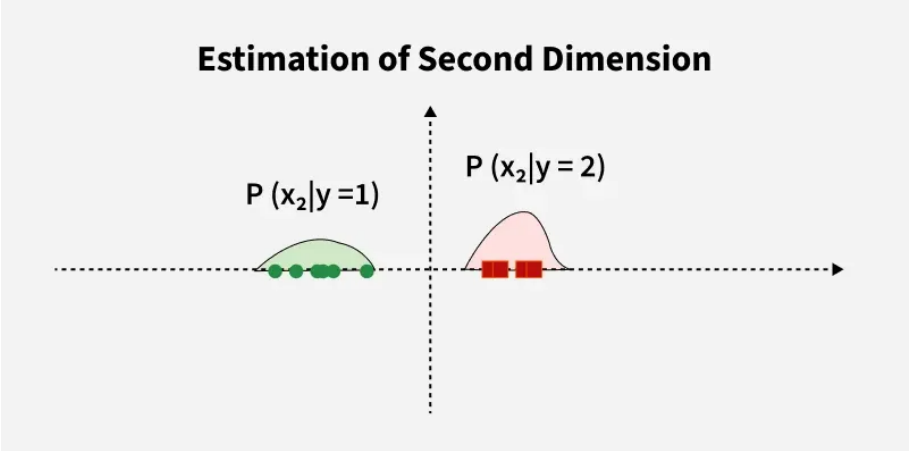


Figura 4.20: Naïve Bayes: Estimación de la segunda dimensión, sacado de [4.17]

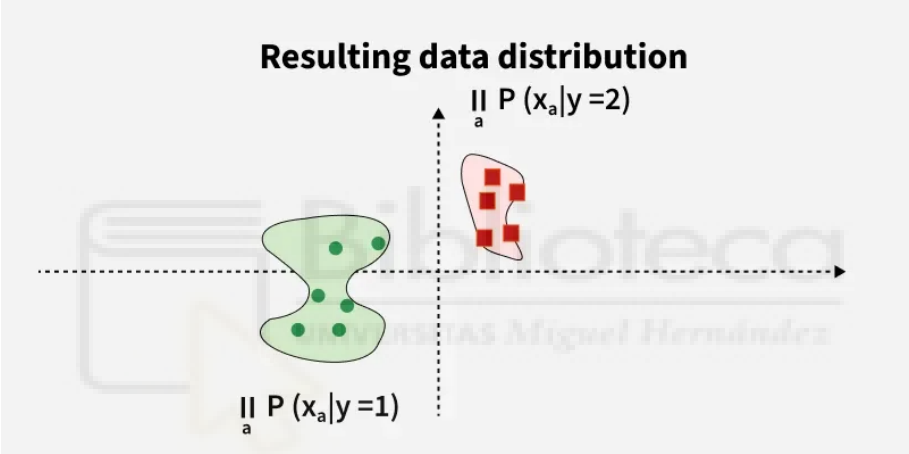


Figura 4.21: Naïve Bayes: Resultado de la distribución, sacado de [4.17]

## 4.4 - MÉTRICAS OBTENIDAS

En este apartado se presentan los resultados que se han obtenido al evaluar las métricas obtenidas antes y después de aplicar las técnicas de preprocesamiento descritas anteriormente. El objetivo aquí es cuantificar el impacto de este proceso en la capacidad predictiva de los modelos y analizar cómo ha variado su rendimiento, teniendo en cuenta factores externos.

Antes de profundizar en los resultados, es necesario conocer las métricas que nos van a permitir evaluar la calidad de las predicciones. Gracias a Sklearn podemos mostrar en pantalla una gran variedad de métricas, como pueden ser:

- Precision: relevancia de las predicciones positivas.
- Recall: proporción de verdaderos positivos detectados.
- F1-Score: media armónica de precisión y sensibilidad.
- Accuracy: proporción de aciertos.

Es muy importante a la hora de trabajar con estas métricas saber cuáles son las que tienen más relevancia, por ello, para este estudio que tiene un ámbito clínico, se tomarán las métricas "Precision", para comprobar la precisión del modelo y "Recall", para observar la tasa de detección de positivos.

### 4.4.1 - MÉTRICAS OBTENIDAS: DIABETES

En primer lugar, el algoritmo de Regresión Lineal ha sido seleccionado para trabajar con este dataset. Tras la primera ejecución, se cuenta con un modelo con una precisión del 65% y un 41% de detección de positivos. El primer resultado obtenido es algo pobre, sobre todo pensando que estamos ante un pronóstico clínico.

Aplicando el preprocesamiento ya descrito con anterioridad, se consigue una mejora notable: la precisión del modelo sube levemente hasta un 71%, y lo más importante, la detección de positivos aumenta con creces hasta un 67%. Esta es una mejora importante, ya que aumentamos la cantidad de pacientes positivos que se pueden diagnosticar.

El gran aumento en las métricas, especialmente en la detección de positivos, se debe al uso de SMOTE y su trabajo en crear nuevos ejemplos sintéticos en clase minoritaria, los positivos. Este dataset contaba un desbalance en sus clases, lo que provocaba un gran sesgo hacia los pacientes sanos.

Por otro lado, el dataset se ha empleado para entrenar un modelo con XGBoost. Siguiendo el procedimiento como en el algoritmo anterior, con la primera ejecución se obtiene una precisión en el modelo del 68%, y una tasa de detección de positivos de 49%. De la misma forma que antes, los resultados son pobres para el contexto de trabajo.

Tras aplicar el preprocesamiento se consigue un aumento de las métricas muy notable: una leve mejora en la precisión del modelo con un 75% y una detección de positivos disparada al 78%.

Con este algoritmo se pueden alcanzar mejores métricas, de partida sin preprocesar y otras con una mejoría sorprendente. De la misma forma que se ha desarrollado antes, el uso de este dataset mejora bastante al aplicar SMOTE, debido a la gran diferencia en el número de pacientes sanos frente a los enfermos.

Finalmente, una tabla a modo de resumen de ambos modelos con y sin preprocesamiento:

| Modelo | Precisión del modelo | Detección de positivos |
|---|---|---|
| Regresión Lineal Sin PP | 65% | 41% |
| Regresión Lineal con PP | 71% | 67% |
| XGBoost sin PP | 68% | 49% |
| XGBoost con PP | 75% | 78% |

Tabla 4.22: Resultados en Diabetes

### 4.4.2 - MÉTRICAS OBTENIDAS: STROKE

El siguiente bloque de modelos han sido entrenados con el dataset Stroke, con los algoritmos Support Vector Machine (SVM) y k-Nearest Neighbors (KNN). En este punto, se han escogido datasets con peculiaridades que se observarán en los resultados, a diferencia del apartado anterior que se ha obtenido una mejora sencilla y clara.

En primera instancia, el algoritmo SVM obtiene unas métricas positivas de partida, teniendo una precisión del 77% en el modelo y 83% en la detección de positivos. Ya se puede observar un porcentaje alto en ambas métricas sin preprocesar si comparamos con el apartado anterior. Esto se puede deber a un mejor dataset de partida y/o una mejor elección del algoritmo para los datos empleados.

De todas maneras, no quita que sea posible mejorar el modelo con preprocesamiento. Al aplicar las distintas técnicas se consigue una mejoría: la precisión del modelo aumenta al 87% y la detección de positivos al 88%. No sucede una mejora tan drástica como anteriormente porque ya se parte de unas métricas aceptables, pero aún así, se puede observar como la depuración del dataset con la eliminación del ruido, permite al algoritmo encontrar un margen óptimo más fácilmente.

A continuación, un algoritmo que sorprende con sus resultados: KNN. Ejecutando en un primer lugar sin preprocesamiento se recogen unas métricas muy altas, obteniendo una precisión en el modelo del 90% y una detección de positivos de 97%.

Cuando se aplica el preprocesamiento se encuentra con algo inesperado, si bien el valor de la precisión del modelo aumenta levemente hasta 91%, con la detección de positivos se observa una bajada hasta 95%. Parece ser que este caso está contradiciendo la hipótesis inicial, pero será en el siguiente apartado en el que se desarrolle este caso particular.

| Modelo | Precisión del modelo | Detección de positivos |
|---|---|---|
| SVM Sin PP | 77% | 83% |
| SVM con PP | 87% | 88% |
| KNN sin PP | 90% | 97% |
| KNN con PP | 91% | 95% |

Tabla 4.23: Resultados en Stroke

### 4.4.3 - MÉTRICAS OBTENIDAS: TIROIDES

Como último dataset escogido para este proyecto está Tiroides, con el cual se han entrenado modelos usando los algoritmos Random Forest y Naïve Bayes. De la misma forma que antes, con la intención de traer variedad de algoritmos (además de datasets) y ver el impacto que tiene el preprocesamiento en ellos, se han seleccionado estos por tener distintas peculiaridades.

Random Forest es un algoritmo que obtiene unas métricas inusuales, pero con una conclusión positiva que se desarrollará posteriormente. Se obtienen unas métricas iniciales de 84% en la precisión del modelo y 99% en la detección de positivos. Observando ese 99% en la detección de positivos se puede pensar que no es necesario realizar un preprocesamiento, pero no se debe olvidar que contemplamos dos métricas.

Tras aplicar el preprocesamiento se puede notar un equilibrio entre ambas métricas, consiguiendo un 95% en ambas. Más adelante se desarrollará la positividad de este balance, ya que a pesar de reducir esta cantidad de detección de positivos, es necesario saber por qué se estaban obteniendo esos resultados.

En último lugar, un modelo entrenado con Naïve Bayes, probablemente el más difícil de tratar por su naturaleza. Se parte de unas métricas bajas, con 41% y 34% en precisión del modelo y detección de positivos respectivamente.

En la segunda ejecución con el respectivo preprocesamiento se observa una mejora, aumentando a 72% en la precisión del modelo y 50% en la detección de positivos. Si bien el porcentaje de mejora es positivo, es todavía bajo para el contexto en el que se sitúa la investigación.

| **Modelo** | **Precisión del modelo** | **Detección de positivos** |
|---|---|---|
| Random Forest Sin PP | 84% | 99% |

| | | |
|---|---|---|
| Random Forest con PP | 95% | 95% |
| Naïve Bayes sin PP | 41% | 34% |
| Naïve Bayes con PP | 72% | 50% |

Tabla 4.24: Resultados en Tiroides

Algo notable en este punto, es la gran diferencia que puede haber usando el mismo dataset en distintos modelos, especialmente en este último. Ya salta a la vista que el preprocesamiento no es ningún tipo de "procedimiento infalible" para mejorar modelos. En el siguiente apartado se desarrollará el motivo de las distintas métricas obtenidas, además de razonar el funcionamiento de los algoritmos ya descritos en apartados anteriores.

## 4.5 - ANÁLISIS Y COMPARACIÓN DE RESULTADOS

Una vez presentados los resultados, en este apartado van a ser analizados e interpretados tras aplicar los modelos de aprendizaje automático a los diferentes conjuntos de datos. El objetivo en este punto es identificar los patrones de comportamiento entre algoritmos y comprobar hasta qué punto el preprocesamiento de los datos ha contribuido a mejorar el rendimiento predictivo, validando así la hipótesis planteada al inicio.

En el conjunto de datos de Diabetes, tanto la Regresión logística como el modelo XGBoost muestran una mejora significativa tras aplicar técnicas de preprocesamiento. En el caso de la Regresión Logística, hay un aumento de ambas métricas del 65% al 71% y del 41% al 67%. De forma similar, en el modelo XGBoost se experimenta un incremento del 68% al 75% y del 49% al 78%. ¿Cómo se puede justificar esta mejora?

Si se observa el dataset, es posible ver a simple vista el sesgo que hay en las clases, algo común en los datos en el ámbito clínico. La mejora viene dada principalmente por la normalización de variables y balanceo de clases con SMOTE, lo que reduce este sesgo mencionado hacia la clase

mayoritaria, permitiendo al modelo identificar un mayor número de pacientes con diagnóstico positivo.

Se debe destacar que, aunque XGBoost parte de una precisión superior, la Regresión Logística logra una mejora relativa mejor, lo que sirve para evidenciar que este algoritmo tiene una mayor sensibilidad a los datos de entrada y el tratamiento dado. Esto refuerza la idea de que los modelos lineales, al depender directamente de la escala y distribución de las variables, se benefician en mayor medida de las transformaciones aplicadas durante el preprocesamiento.

Avanzando ahora al dataset Stroke, se encuentran resultados diferenciados según el tipo de algoritmo. Por un lado, el modelo SVM muestra una mejora notable tras aplicar preprocesamiento, pasando del 77% al 87% en precisión y del 83% al 88% en detección de positivos. Conociendo la naturaleza del algoritmo ya descrita antes, se puede atribuir el incremento a la reducción del ruido y a la normalización de las variables, facilitando al algoritmo la búsqueda del margen óptimo de separación entre clases. Como es un algoritmo sensible a la escala de atributos, la normalización permite que todas las variables contribuyan de forma equilibrada al cálculo.

Por otro lado, el modelo KNN apenas sufre variaciones en sus métricas tras el preprocesamiento, manteniendo valores muy elevados en ambas métricas: precisión del 90% al 91% y la detección de positivos del 97% al 95%. Este comportamiento es esperado si se conoce la naturaleza del algoritmo, basado en distancias locales, debido a que las transformaciones de los datos no alteran significativamente las relaciones entre instancias. De hecho, en modelos de este tipo, una normalización excesiva o rebalanceo artificial de las clases puede suavizar los contrastes locales que el algoritmo necesita para clasificar.

Este comportamiento refuerza la idea de que la efectividad del preprocesamiento depende no solo del conjunto de datos, sino también del paradigma de aprendizaje empleado.

En último lugar, el conjunto de datos de Tiroides. Los resultados muestran también unos comportamientos diferenciados entre los modelos de distinta naturaleza. El modelo Random Forest presenta una mejora significativa tras el preprocesamiento, incrementando su precisión del 84% al 95%, aunque se observa una ligera reducción en la detección de positivos (de 99%

a 95%). Este cambio debe interpretarse como positivo, ya que se trata de un reequilibrio entre sensibilidad y precisión, donde el modelo pasa a hacer predicciones más consistentes y menos propensas a falsos positivos. La función del preprocesamiento, al reducir ruido y homogeneizar variables, permite a Random Forest generar árboles más estables y con divisiones más representativas de los patrones reales de datos.

En el caso del modelo Naïve Bayes, también se obtiene una mejora evidente, aunque partiendo de valores iniciales más bajos. La precisión aumenta del 41% al 72%, mientras que la detección de positivos pasa del 34% al 50%. Este incremento se debe principalmente a que el preprocesamiento mejora la distribución de las variables y reduce la presencia de valores atípicos, lo que molesta a la hipótesis del modelo. Sin embargo, el rendimiento de este modelo sigue siendo limitado debido a su propia formulación: asume la independencia entre características y una distribución estadística específica, condiciones que rara vez se van a cumplir en datos clínicos reales.

Biblioteca
UNIVERSITAS Miguel Hernández

# Capítulo 5: Conclusiones y trabajos futuros

Después de haber finalizado la investigación, es el momento de recoger las conclusiones derivadas a raíz de los objetivos propuestos y expansiones que se pueden llevar a cabo de cara al futuro, ya sea de otro ámbito o concretar aún más el caso.

## 5.1 - VALIDACIÓN DE LA HIPÓTESIS

En conjunto, los resultados obtenidos en los distintos datasets confirman que el preprocesamiento de los datos constituye un paso esencial en la construcción de modelos de pronóstico clínico, ya que mejora de forma notable la capacidad predictiva y reducir la cantidad de falsos negativos. Si se traslada a la práctica, se traduce en una detección más temprana y fiable de pacientes en riesgo, consiguiendo un aumento de la eficacia de las herramientas de apoyo al diagnóstico y reforzando la fiabilidad de decisiones clínicas basadas en datos.

Los experimentos descritos evidencian que el impacto del preprocesamiento no es uniforme, sino que depende del tipo de algoritmo y de la naturaleza de los datos empleados. Los modelos lineales como puede ser Regresión Logística, mostraron una mejora relativa mayor al aplicar normalización y balanceo de clases, ya que estos son bastante sensibles a la escala y distribución de las variables. De una manera similar, los modelos basados en fronteras como SVM, se beneficiaron de la homogeneización de los datos, al facilitar la búsqueda de márgenes óptimos de separación de clases.

Por el contrario, otro tipo de algoritmos que se fundamentan en regiones locales, como KNN, presentaron una mejora menos significativa, lo que demuestra que las transformaciones globales no siempre llegan a aportar ventajas cuando la decisión del modelo depende de cercanías. En el caso de modelos de ensamblado, como pueden ser Random Forest o XGBoost, se observó una mayor estabilidad, pero también una mejora adicional notable cuando las distintas técnicas redujeron el ruido y permitió un aprendizaje más equilibrado.

Finalmente, el modelo que menos beneficio obtiene en este ámbito, Naïve Bayes, de naturaleza probabilística. Si bien sí obtuvo una mejora, el rendimiento siguió condicionado por las suposiciones matemáticas del algoritmo, como la independencia entre variables y la forma de las distribuciones.

En conclusión, estos resultados permiten afirmar que el preprocesamiento no se debe de pensar como un conjunto de pasos universales, sino como un proceso adaptativo y dependiente del modelo. Este debe diseñarse en función del tipo de algoritmo y las características del dataset. Esta conclusión consolida la hipótesis central del estudio: el tratamiento previo de datos es determinante para mejorar el rendimiento predictivo, pero su efecto varía en función de la naturaleza del modelo y del problema clínico abordado.

## 5.2 - CONCLUSIONES

El presente trabajo revela las conclusiones obtenidas a partir del trabajo de investigación centrado en verificar si el preprocesamiento tiene un impacto positivo en los modelos predictivos clínicos.

En primer lugar, se afirma que se han podido lograr los objetivos principales propuestos para el proyecto. Esto implica en la demostración de la mejoría de los modelos tras el preprocesamiento. Para lograr esto, se ha necesitado de la evaluación de los modelos Machine Learning con y sin preprocesamiento, visualizando sus métricas de forma clara y evidenciando este aumento de precisión a un modelo más fiable.

Por otro lado, también se han abordado los objetivos secundarios, los cuáles nunca han carecido de importancia. El estudio de distintas técnicas de preprocesamiento ha sido fundamental para lograr los objetivos principales, ya que, como hemos podido comprobar, estas técnicas no deben usarse como “técnicas universales”. Ligado a este estudio, también ha sido necesario conocer los distintos datasets que se han usado para entrenar modelos, los cuales han sido tres los seleccionados finalmente, pero han sido varios más los que no han pasado el proceso de selección, ya sea por su naturaleza o información pobre. Conocer las distintas métricas que se pueden obtener de los modelos entrenados también ha sido una labor de mucha importancia,

debido a que de esta forma se puede hacer más clara la visualización de los resultados, mostrando lo que se considera más importante en el sector clínico.

Antes de comenzar la investigación, se ha valorado el uso de las tecnologías para llevarlo a cabo, siendo Python la elección. Este lenguaje ha sido escogido sobre R por considerarse más liviano a la hora de programar, porque la realidad es que la investigación se podría llevar a cabo en ambos. El uso de librerías es crucial cuando hablamos del sector de análisis de datos, y es posible encontrar librerías similares en ambos lenguajes, pero una vez más, Python presenta una curva de aprendizaje más llevadera.

| **Objetivos** | **Logrado** |
|---|---|
| Analizar el impacto del preprocesamiento | ✓ |
| Evaluar modelos de Machine Learning | ✓ |
| Visualizar resultados de forma clara | ✓ |
| Ayudar a la toma de decisiones clínicas | ✓ |
| Estudiar técnicas de preprocesamiento | ✓ |
| Explorar distintos datasets clínicos | ✓ |
| Comparar métricas de evaluación | ✓ |
| Aplicar conocimientos adquiridos | ✓ |
| Profundizar en el análisis de datos clínicos | ✓ |
| Mejorar habilidades prácticas | ✓ |
| Elaborar una guía práctica del preprocesamiento clínico | ✓ |

Tabla 5.1: Resumen de los objetivos logrados

## 5.2 - POSIBLES DESARROLLOS FUTUROS

Una vez mostrados los resultados obtenidos, este trabajo abre la puerta a múltiples desarrollos futuros. Una de las posibles ampliaciones puede ser la inclusión de nuevos algoritmos de aprendizaje automático profundo, como pueden ser las redes neuronales, ya que pueden aprovechar representaciones más complejas de los datos clínicos.

También, siempre me ha resultado interesante la idea de automatizar el proceso de preprocesamiento, mediante técnicas como AutoML o pipelines inteligentes que seleccionen dinámicamente las transformaciones más adecuadas según las características del dataset.

Desde el punto de vista aplicado, una evolución natural de este estudio consistiría en integrar el sistema dentro de un entorno clínico real, evaluando su desempeño en la práctica mediante datos hospitalarios o registros electrónicos de salud.

Este trabajo constituye una base sólida sobre la que construir futuras líneas de investigación orientadas a la optimización y automatización del preprocesamiento de datos clínicos, con el objetivo final de mejorar la precisión, eficiencia y aplicabilidad de los sistemas de apoyo a la decisión médica.

Biblioteca
UNIVERSITAS Miguel Hernández

Biblioteca
UNIVERSITAS Miguel Hernández

# Bibliografía


[2.1] Gestión de la calidad de los datos en el sector sanitario: cinco prácticas recomendadas.
Mariam Anwar
Enlace
8 de abril de 2025

[2.2] Lenguajes de programación: tipos y características
Enlace
8 de abril de 2025

[2.3] ¿Qué son las librerías en programación y para qué sirven?
Enlace
8 de abril de 2025

[2.4] Python Software Foundation. “The Python Tutorial”
Enlace
19 de mayo de 2025

[2.5] Stack Overflow Developer Survey 2023
Enlace
19 de mayo de 2025

[2.6] The R Project for Statistical Computing – *About R*
Enlace
19 de mayo de 2025

[2.7] CRAN – *Packages documentation*
Enlace
19 de mayo de 2025


Biblioteca UNIVERSITAS Miguel Hernández

[2.8] Wickham, H., & Grolemund, G. (2016). *R for Data Science*
Hadley Wickham y Garrett Grolemund
19 de mayo de 2025

[2.9] pandas Documentation
Enlace
19 de mayo de 2025

[2.10] ¿Qué es un DataFrame?
Enlace
19 de mayo de 2025

[2.11] scikit-learn documentation
Enlace
20 de mayo de 2025

[2.12] matplotlib Documentation
Enlace
20 de mayo de 2025

[2.13] Análisis en profundidad: Plotly vs Matplotlib en Python
Matt Popovic
Enlace
20 de mayo de 2025

[2.14] NumPy Documentation
Enlace
20 de mayo de 2025

[2.15] Numpy - ndarray
Enlace
20 de mayo de 2025

Biblioteca
UNIVERSITAS Miguel Hernández

[2.16] XGBoost Documentation
Enlace
20 de mayo de 2025

[2.17] *XGBoost: A scalable tree boosting system*
Chen, T. y Guestrin, C
20 de mayo de 2025

[2.18] A Gentle Introduction to XGBoost for Applied Machine Learning
Jason Brownlee
Enlace
20 de mayo de 2025

[2.19] imbalanced-learn documentation
Enlace
20 de mayo de 2025

[2.20] *SMOTE: Synthetic Minority Over-sampling Technique*.
Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall y W. Philip Kegelmeyer
20 de mayo de 2025

[3.1] Imputación de valores faltantes
Enlace
9 de junio de 2025

[3.2] ¿Qué es la normalización en el aprendizaje automático?
Sejal Jaiswal
Enlace
9 de junio de 2025

[3.3] Codificación de variables categóricas con Python y R: Técnicas y conceptos clave.
Enlace
9 de junio de 2025

[3.4] Estudio del desbalance de clases en bases de datos de microarrays de expresión genética mediante técnicas de Deep Learning
H. Cruz-Reyes, A. Reyes-Nava, E. Rendón-Lara, R. Alejo
Enlace
9 de junio de 2025

[4.1] Kaggle
Enlace
11 de junio de 2025

[4.2] Diabetes Dataset
Enlace
11 de junio de 2025

[4.3] healthcare-dataset-stroke-data
Enlace
11 de junio de 2025

[4.4] Thyroid Disease Data Set
Enlace
11 de junio de 2025

[4.5] Como lidiar con el desbalanceo de datos
João Vitor de Miranda
Enlace
12 de junio de 2025

[4.6] SMOTE: Synthetic Minority Over-sampling Technique
Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, W. Philip Kegelmeyer
Enlace
12 de junio de 2025

Biblioteca
UNIVERSITAS Miguel Hernández

[4.7] ¿Qué es la normalización en el aprendizaje automático?
Sejal Jaiswal
Enlace
12 de junio de 2025

[4.8] Simplifica tus datos con PCA en Python: Reducción de Dimensionalidad usando Numpy
Enlace
12 de junio de 2025

[4.9] Optimizing Performance: SelectKBest for Efficient Feature Selection in Machine Learning
Enlace
12 de junio de 2025

[4.10] Módulo 5.3 Métodos avanzados en clasificación.
Prof. Alberto Fernández Hilario
Enlace
6 de octubre de 2025

[4.11] ¿Qué es el algoritmo KNN?
Enlace
7 de octubre de 2025

[4.12] Regresión logística múltiple
Ochoa Sangrador C, Molina Arias M, Ortega Páez E
Enlace
7 de octubre de 2025

[4.13] ¿En qué consiste la regresión logística? ¿Qué es la regularización?
Luis Torres
Enlace
7 de octubre de 2025

Biblioteca
UNIVERSITAS Miguel Hernández

[4.14] Cómo funciona el algoritmo XGBoost

Enlace

7 de octubre de 2025

[4.15] Implementation of XGBoost (eXtreme Gradient Boosting)t

Enlace

7 de octubre de 2025

[4.16] ¿Qué es el bosque aleatorio?

Enlace

7 de octubre de 2025

Biblioteca
UNIVERSITAS Miguel Hernández