UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

MÁSTER UNIVERSITARIO EN ANÁLISIS DE DATOS MASIVOS

TRABAJO FIN DE MÁSTER

Predicción Inteligente del Comportamiento Mecánico de Vigas usando Machine Learning y Análisis por Elementos Finitos

Daniel López López

Dirigido por

Yudith Coromoto Cardinale

CURSO 2024 - 2025

TÍTULO: Predicción Inteligente del Comportamiento Mecánico de Vigas usando Machine Learning y Análisis por Elementos Finitos

AUTOR: Daniel López López

TITULACIÓN: MÁSTER UNIVERSITARIO EN ANÁLISIS DE DATOS MASIVOS

DIRECTOR DEL PROYECTO: Yudith Coromoto Cardinale

FECHA: Octubre de 2025

RESUMEN

Este trabajo presenta el desarrollo de modelos de aprendizaje automático para complementar análisis mediante elementos finitos (MEF). Se generan datos sintéticos a partir de modelos de vigas 1D en HyperMesh, obteniendo como resultados el desplazamiento máximo y la tensión máxima de Von Mises. Estos resultados se utilizan para entrenar modelos de aprendizaje automático capaces de realizar predicciones sin necesidad de recurrir al MEF.

Se entrenaron dos modelos para evaluar sus fortalezas y limitaciones: un Gradient Boosting y una red neuronal. El documento detalla todo el proceso, desde la generación de datos y el preprocesamiento hasta la optimización de hiperparámetros. Se destaca la importancia de la ingeniería de características (*feature engineering*) para lograr modelos con métricas de desempeño adecuadas. Los modelos desarrollados permiten predecir el comportamiento de vigas con distintas longitudes, materiales, tipos de sección y condiciones de contorno.

Finalmente, se implementó una interfaz gráfica simplificada, denominada Alabeam, que permite realizar predicciones sin necesidad de conocimientos avanzados en cálculo estructural o MEF. Esta herramienta demuestra el potencial del aprendizaje automático en fases de diseño, donde se requieren múltiples iteraciones de manera eficiente siendo capaz de ofrecer resultados en menos de un segundo.

ABSTRACT

This thesis presents the development of machine learning models to complement finite element analysis (FEA). Synthetic datasets were generated from 1D beam models in HyperMesh, obtaining maximum displacement and maximum Von Mises stress as outputs. These results were subsequently used to train machine learning models capable of making predictions without performing FEA.

Two models were trained to assess their respective strengths and limitations: a Gradient Boosting model and a neural network. The work details the entire workflow, from data generation and preprocessing to hyperparameter optimization, emphasizing the crucial role of feature engineering in achieving models with satisfactory performance metrics. The developed models enable the prediction of beam behavior for various lengths, materials, cross-sectional types, and boundary conditions.

Finally, a simplified graphical user interface, named Alabeam, was implemented, allowing users to make predictions without advanced knowledge of structural analysis or FEA. This tool demonstrates the potential of machine learning in design phases, where multiple iterations are required efficiently, providing results in less than a second.

Keywords: Finite Element Method (FEM), Structural Analysis, Machine Learning, Neural Networks, Gradient Boosting, Surrogate Modeling, PyNastran, OptiStruct, Streamlit, Structural Mechanics.

Índice general

Índice de Figuras

Índice de Tablas

Símbolos y abreviaturas

Símbolo	Nombre	Unidad
L	Longitud de la viga	mm
r_{ext}	Radio exterior	mm
r_{int}	Radio interior	mm
t	Espesor	mm
b	Base de la sección	mm
h	Altura de la sección	mm
A	Área de sección	mm^2
I	Momento de inercia	mm^4
J	Momento polar de inercia	mm^4
K	Factor de longitud efectiva	–
f_e	Matriz de cargas MEF	N
K_e	Matriz de rigidez MEF	N/mm , N/rad, N·mm/rad
d_e	Vector desplazamiento MEF	mm/rad
W	Módulo resistente	mm^3
E	Módulo de elástico	MPa (N/mm^2)
ρ	Densidad	kg/mm^3
ν	Coeficiente de Poisson	–
ρ	Densidad	kg/mm^3
F	Fuerza puntual	N
M	Momento puntual	N·mm
$N(x)$	Esfuerzo axial	N
w	Carga distribuida	N/mm
u, w	Desplazamiento	mm
$w(x)$	Desplazamiento a lo largo de x	mm
$u_{\text{máx}}$	Desplazamiento (*displacement*) máximo	mm
ε	Deformación	–
θ	Rotación	rad
σ	Tensión (*stress*)	MPa
MAE	Mean Absolute Error	mismas unidades del objetivo
MedAE	Median Absolute Error	mismas unidades del objetivo
RMSE	Root Mean Squared Error	mismas unidades del objetivo
MAPE	Mean Absolute Percentage Error	–
R^2	Coeficiente de determinación	–

Tabla 1. Glosario de símbolos y variables utilizadas en el proyecto.

Capítulo 1. INTRODUCCIÓN

En el ámbito de la ingeniería estructural, la predicción del comportamiento de componentes sometidos a cargas constituye una tarea esencial para garantizar la seguridad, la funcionalidad y la eficiencia de los diseños. Tradicionalmente, estos análisis se han abordado mediante métodos clásicos, como la teoría de vigas, o bien mediante simulaciones numéricas avanzadas, siendo el Método de los Elementos Finitos (MEF, en inglés *Finite Element Method*, FEM) la técnica más extendida en la práctica profesional. Si bien el MEF proporciona una elevada precisión y versatilidad, también implica un alto coste computacional y temporal, así como la necesidad de conocimientos especializados para cada nuevo caso de carga, geometría o material. Estas limitaciones se hacen especialmente evidentes en fases tempranas de diseño, donde resulta necesario evaluar múltiples configuraciones estructurales de forma ágil [1].

En los últimos años, el auge del aprendizaje automático (*machine learning*, ML) y el análisis de datos ha supuesto una transformación profunda en numerosas disciplinas de la ingeniería, gracias a su capacidad para identificar y modelar patrones complejos a partir de grandes volúmenes de datos experimentales o simulados [2]. A diferencia de los métodos tradicionales, el ML no requiere formular ni resolver de manera explícita las ecuaciones diferenciales que describen el comportamiento físico de un sistema, sino que construye modelos predictivos basados en la experiencia contenida en los datos. Esto permite obtener estimaciones rápidas y, en muchos casos, con una precisión comparable a la de métodos numéricos convencionales, pero con un coste computacional mucho menor.

La combinación de técnicas clásicas como el MEF con métodos basados en ML ha dado lugar a enfoques híbridos FEM–ML, que ya se han explorado en campos como la ingeniería civil y aeroespacial, mostrando un gran potencial para estimar el comportamiento mecánico de estructuras con gran eficiencia. Gracias a estas ventajas, el aprendizaje automático se plantea como un recurso complementario e incluso acelerador de los métodos clásicos de simulación. En el presente trabajo se explorará esta sinergia, analizando cómo el ML puede apoyar el análisis estructural y facilitar predicciones eficientes del comportamiento de componentes estructurales sometidos a distintas condiciones de carga.

Este proyecto propone el desarrollo de una herramienta que combine técnicas de aprendizaje automático con análisis estructural clásico, con el objetivo de predecir de forma automática y eficiente los principales resultados de interés en el estudio de vigas: la tensión máxima (Von Mises) y el desplazamiento máximo. Para ello, se utilizarán datos generados a partir de modelos MEF creados automáticamente en HyperMesh y resueltos mediante OptiStruct, abarcando una amplia variedad de combinaciones geométricas, materiales, condiciones de contorno y cargas.

Con el fin de controlar la complejidad del problema y facilitar el entrenamiento de los modelos, se ha optado por acotar el estudio a vigas bidimensionales modeladas como elementos tipo viga (1D), con distintas secciones, materiales y esquemas de carga. Este enfoque permite evaluar con precisión la aplicabilidad del aprendizaje automático en problemas estructurales, sin perder generalidad en cuanto a los principios fundamentales de la mecánica estructural.

Además del desarrollo del modelo predictivo, el proyecto incluye el diseño de una interfaz gráfica interactiva, que permita a cualquier usuario introducir las condiciones de diseño y obtener resultados de forma rápida e intuitiva. Como valor añadido, se incorporan funciones de recomendación inteligente orientadas a la optimización estructural: por ejemplo, sugerencias de materiales más ligeros o económicos que mantengan los requisitos de resistencia, o propuestas de modificaciones geométricas en la sección.

En conjunto, este trabajo tiene un doble objetivo: por un lado, demostrar la viabilidad y utilidad del aprendizaje automático como herramienta complementaria al análisis estructural clásico; y por otro, desarrollar un software práctico, denominado Alabeam, que actúe como asistente inteligente en el diseño y evaluación de vigas.

1.1 Planteamiento

El presente proyecto se fundamenta en la necesidad de agilizar el análisis estructural en fases de diseño, donde resulta imprescindible evaluar múltiples alternativas en plazos reducidos. Si bien el MEF es la referencia por su precisión y fiabilidad, su elevado coste computacional y la exigencia de conocimiento especializado limitan su aplicación en procesos iterativos y en exploraciones de diseño amplias.

En este marco, el aprendizaje automático ofrece una oportunidad para acelerar dichos análisis, al permitir la construcción de modelos predictivos capaces de estimar parámetros estructurales clave con rapidez una vez entrenados. En este proyecto se propone el desarrollo de un modelo de aprendizaje automático que, aprovechando la capacidad predictiva de estas técnicas, estime con precisión la tensión y el desplazamiento máximos de vigas a partir de sus características geométricas, materiales y de carga. Para ello, se empleará un conjunto de datos generado mediante simulaciones MEF automatizadas, garantizando así que el modelo aprenda sobre una base sólida y representativa.

Adicionalmente, se plantea la implementación de una interfaz gráfica orientada a facilitar la interacción con el sistema, de modo que los usuarios puedan introducir las condiciones de diseño de manera sencilla y obtener predicciones inmediatas sin necesidad de conocimientos avanzados en simulación. La herramienta resultante, denominada Alabeam, se concibe como un asistente inteligente destinado a apoyar el proceso de diseño estructural de vigas, proporcionando resultados rápidos, accesibles y eficientes.

1.2 Objetivos

El presente proyecto tiene como finalidad desarrollar una herramienta inteligente que combine técnicas de aprendizaje automático con análisis estructural clásico, para ofrecer predicciones fiables y eficientes del comportamiento de vigas bajo diversas condiciones de carga, geometría, material y apoyo.

1.2.1. Objetivo general

Diseñar e implementar una solución basada en aprendizaje automático capaz de predecir de forma precisa y eficiente las deformaciones y tensiones en vigas bidimensionales, utilizando datos generados mediante simulaciones por elementos finitos (MEF). Asimismo, integrar esta solución en una herramienta interactiva que proporcione recomendaciones estructurales orientadas a la optimización del diseño.

1.2.2. Objetivos específicos

- Automatizar la generación de modelos estructurales de vigas mediante scripts en TCL para HyperMesh, permitiendo crear configuraciones variadas en función de parámetros como longitud, tipo de sección, material, apoyos y esquema de cargas.
- Desarrollar un script en Python para generar combinaciones estructurales y cargas aleatorias, exportarlas a un archivo CSV y facilitar así generación de los modelos con TCL.
- Ejecutar los modelos MEF en OptiStruct de forma masiva (batch) para simular aproximadamente 4000 configuraciones distintas, obteniendo los resultados en archivos de salida estructurados (`.op2`).
- Utilizar la librería PyNastran para extraer de forma automatizada los resultados clave de las simulaciones: desplazamiento máximo y tensión máxima (Von Mises).
- Construir un conjunto de datos estructurado combinando parámetros de entrada y resultados MEF, apto para el entrenamiento de modelos de regresión supervisada.
- Entrenar y comparar distintos modelos de aprendizaje automático, como LightGBM (Gradient Boosting) y redes neuronales artificiales.
- Evaluar el rendimiento de los modelos mediante métricas como MAE, RMSE y R^2, y analizar su precisión en la estimación de variables estructurales frente a las soluciones MEF.
- Implementar una aplicación interactiva con Streamlit (Alabeam), que permita introducir condiciones de diseño y obtener predicciones de forma rápida e intuitiva, con posibilidad de comparar resultados entre modelos.
- Incorporar un sistema de recomendaciones inteligentes que sugiera materiales alternativos o modificaciones en la sección de la viga para mejorar el diseño desde un punto de vista estructural, económico o funcional.

1.3 Beneficios del proyecto

El desarrollo de este proyecto aporta una serie de beneficios significativos tanto desde el punto de vista técnico como práctico, con un especial énfasis en su aplicabilidad en el ámbito del diseño estructural asistido por inteligencia artificial. A continuación, se detallan las principales ventajas que ofrece la herramienta desarrollada:

- **Reducción del tiempo de análisis:** permite obtener estimaciones fiables del comportamiento estructural sin necesidad de realizar simulaciones por elementos finitos completas para cada caso, lo que acelera el proceso de diseño y validación.
- **Optimización estructural automatizada:** incorpora recomendaciones inteligentes que sugieren configuraciones alternativas (materiales, secciones, etc.) manteniendo la seguridad estructural, con el objetivo de reducir el peso o el coste del componente.
- **Apoyo a la toma de decisiones técnicas:** la herramienta no solo ofrece predicciones numéricas, sino que proporciona sugerencias basadas en criterios ingenieriles y de fabricación, facilitando decisiones más informadas durante las fases iniciales del diseño.
- **Aplicabilidad en entornos industriales:** Alabeam es especialmente útil en sectores como la ingeniería civil, la automoción, la aeronáutica o el diseño de estructuras metálicas, donde los análisis estructurales son frecuentes y el ahorro de tiempo resulta crítico.
- **Valor académico y formativo:** el proyecto puede servir como recurso didáctico para estudiantes y docentes, facilitando la comprensión del comportamiento estructural y del impacto de los parámetros de diseño, a la vez que introduce conceptos de inteligencia artificial aplicada a la ingeniería.
- **Escalabilidad y adaptabilidad:** gracias a su arquitectura modular y su interfaz accesible, la herramienta es fácilmente ampliable a otros tipos de elementos estructurales (pórticos, placas, etc.) o incluso a otros dominios como la optimización topológica o la fabricación aditiva.

Capítulo 2. MARCO TEÓRICO Y MARCO TECNOLÓGICO

El presente capítulo tiene como objetivo establecer los fundamentos teóricos que sustentan el desarrollo de este trabajo y las herramientas desarrolladas. Dado que el proyecto combina conceptos propios de la ingeniería estructural con técnicas de aprendizaje automático, el marco teórico se organiza en dos bloques principales: cálculo estructural y aprendizaje automático. Luego se describe el marco tecnológico que sustenta la solución propuesta en este trabajo.

2.1 Marco Teórico

En primer lugar, se abordan los principios de la teoría clásica de vigas, base del análisis estructural de componentes lineales sometidos a cargas. Se revisan los métodos tradicionales de resolución, las hipótesis fundamentales del modelo de Euler-Bernoulli y la transición hacia enfoques numéricos más generales, como el Método de los Elementos Finitos (MEF). Este método constituye la herramienta principal para el análisis computacional de estructuras, permitiendo evaluar el comportamiento de vigas con diferentes condiciones de contorno, tipos de carga y geometrías de sección.

En segundo lugar, se introduce el marco teórico del aprendizaje automático (*Machine Learning*, ML), con especial énfasis en los métodos supervisados aplicados a problemas de regresión. En este contexto, se abordan los principios fundamentales del entrenamiento, validación y evaluación de modelos predictivos, incluyendo tanto algoritmos de tipo ensamble como el HistGradientBoostingRegressor o redes neuronales. Estos enfoques permiten aprender las relaciones no lineales existentes entre las características geométricas, materiales y de carga de las vigas, y las respuestas estructurales correspondientes, como la tensión y el desplazamiento máximos. Con ello, se logra estimar dichas variables con alta precisión y en tiempos de cálculo significativamente menores que los requeridos por los métodos numéricos tradicionales basados en el MEF.

De este modo, en esta sección se establece la base teórica necesaria para comprender el desarrollo posterior del modelo propuesto y justificar las decisiones adoptadas en el trabajo.

2.1.1. Sistema de coordenadas y unidades

A lo largo del presente proyecto se adopta un sistema de referencia cartesiano tridimensional con el fin de describir de forma coherente la geometría, las cargas y las respuestas estructurales de las vigas analizadas. Las vigas se consideran contenidas en el plano XY, con el eje X coincidente con el eje longitudinal de la viga y el eje Y perpendicular a ella dentro del mismo plano. El eje Z se define como el eje que emerge del plano XY como se muestra en la Figura 2.1.

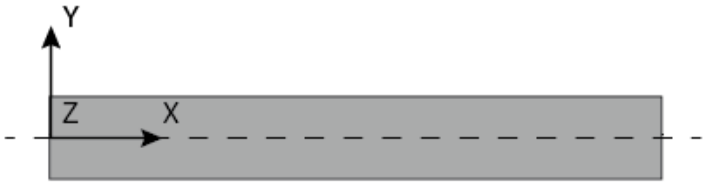


Figura 2.1. Representación del sistema de coordenadas

Las cargas aplicadas pueden actuar en la dirección X (cargas axiales) o en la dirección Y (cargas transversales), mientras que los momentos se aplican alrededor del eje Z, produciendo esfuerzos de flexión en el plano de la viga. Esta convención se mantiene de forma consistente tanto en la formulación teórica como en la generación de los modelos numéricos y en el tratamiento de datos para el aprendizaje automático. De este modo, todos los resultados de desplazamientos, tensiones y reacciones se expresan en el mismo sistema de coordenadas, garantizando la coherencia entre simulaciones FEM y modelos predictivos.

Sistema de unidades

En todo el trabajo se utiliza un sistema coherente basado en milímetros (mm), newtons (N) y megapascales (MPa). Esto implica, en particular, la equivalencia:

$$1\,\mathrm{MPa} = 1\,\mathrm{N/mm^2}.$$

Las magnitudes principales empleadas y sus unidades se muestran en la Tabla 2.1.

Tabla 2.1. Unidades adoptadas en el trabajo

Magnitud	Unidad
Longitud L, dimensiones de sección b, h, etc	mm
Área A	$\mathrm{mm^2}$
Momento de inercia I_z	$\mathrm{mm^4}$
Módulo resistente W_z	$\mathrm{mm^3}$
Desplazamientos u, v	mm
Módulo elástico E, tensiones σ	MPa ($\mathrm{N/mm^2}$)
Fuerzas nodales F_X, F_Y	N
Momentos M_Z	$\mathrm{N\,mm}$

2.1.2. Cálculo estructural

El análisis estructural constituye uno de los pilares fundamentales de la ingeniería, ya que permite determinar la respuesta de los elementos que componen una estructura frente a las cargas que actúan sobre ella. Entre estos elementos, las vigas ocupan un lugar destacado por su amplia utilización en la construcción, la ingeniería mecánica y la industria en general.

Una viga puede definirse como un elemento lineal sometido principalmente a esfuerzos de flexión y cortante, aunque también puede verse afectada por cargas axiales o momentos aplicados. En el contexto de este proyecto, se consideran vigas bidimensionales en el plano XY, capaces de soportar cargas puntuales en las direcciones X e Y, así como momentos aplicados alrededor del eje Z. Asimismo, se consideran dos configuraciones típicas de apoyo: vigas en voladizo y vigas biapoyadas. Las vigas en voladizo presentan un extremo empotrado y el otro libre, lo que genera máximos esfuerzos de flexión en la zona del empotramiento y desplazamiento en el extremo libre. Por su parte, las vigas biapoyadas se apoyan en ambos extremos, permitiendo el giro pero impidiendo el desplazamiento vertical, y representan una de las condiciones de contorno más comunes en el análisis estructural clásico.

Tradicionalmente, el cálculo de vigas se ha abordado mediante métodos analíticos basados en la teoría clásica de *Euler-Bernoulli* [3], que establece las relaciones entre cargas, esfuerzos internos, tensiones y desplazamientos bajo hipótesis simplificadoras. No obstante, este enfoque resulta limitado cuando se requiere analizar sistemas más complejos, con múltiples apoyos, cargas combinadas o secciones de geometría variable.

Para superar estas limitaciones, el *Método de los Elementos Finitos* (MEF) se presenta como una herramienta numérica de gran versatilidad, capaz de modelar el comportamiento estructural de vigas de forma generalizada. A través de la discretización del dominio estructural y la formulación matricial de las ecuaciones de equilibrio, el MEF permite obtener resultados precisos y fácilmente automatizables, facilitando el análisis de un elevado número de configuraciones con un mismo marco teórico. [1]

La Figura 2.2 ilustra el proceso de conversión de un modelo CAD tridimensional de un anclaje a su correspondiente malla de elementos finitos, paso fundamental para posibilitar su análisis estructural mediante el método de los elementos finitos (MEF).

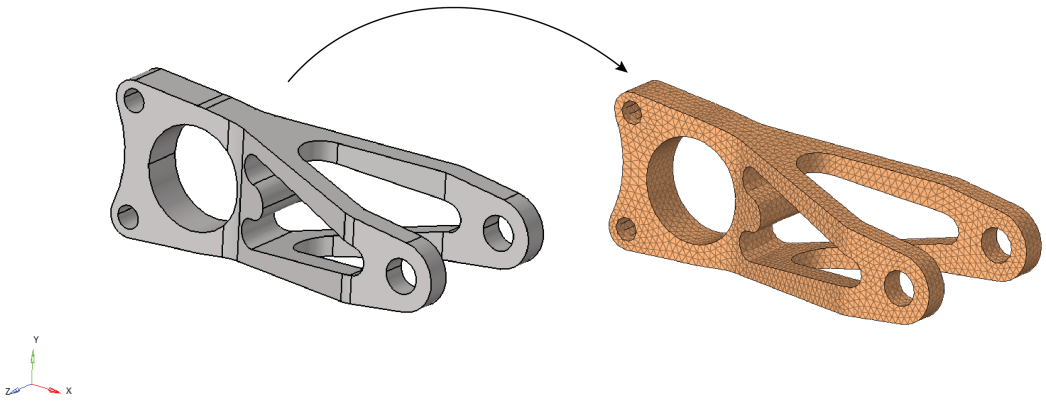


Figura 2.2. Geometría y mallado MEF de un anclaje

Finalmente, dado que en este proyecto se han empleado vigas con distintos tipos de sección transversal, también se revisan los conceptos fundamentales relacionados con las propiedades geométricas de la sección, como el área y el momento de inercia, parámetros que influyen directamente en la rigidez y el comportamiento estructural del elemento.

Introducción al cálculo de vigas

El análisis de vigas constituye uno de los problemas fundamentales en la ingeniería estructural. Una viga puede definirse como un elemento estructural esbelto cuya longitud es considerablemente mayor que sus dimensiones transversales, y que está sometido principalmente a cargas perpendiculares a su eje longitudinal. Estas cargas generan esfuerzos internos de cortante y flexión, que determinan las tensiones y deformaciones en el elemento.

El estudio clásico del comportamiento de vigas se basa en la teoría de *Euler–Bernoulli*, también conocida como teoría de vigas esbeltas. Esta formulación parte de una serie de hipótesis simplificadoras que permiten expresar de forma analítica la relación entre las cargas aplicadas y la respuesta estructural:

- El material es lineal, elástico e isótropo, y cumple la ley de Hooke.
- Las deformaciones son pequeñas, de modo que las ecuaciones de equilibrio pueden considerarse lineales.
- Las secciones planas antes de la deformación permanecen planas y perpendiculares al eje neutro después de deformarse.

Bajo estas hipótesis, el comportamiento de una viga en el plano XY puede describirse mediante la ecuación diferencial de la flexión (las expresiones que siguen se toman de [3, cap. 2–7]):

$$\frac{d^2}{dx^2}\left(EI\frac{d^2w(x)}{dx^2}\right) = q(x)$$

donde E es el módulo de elasticidad del material, I es el momento de inercia de la sección respecto al eje neutro, $w(x)$ es el desplazamiento vertical del eje de la viga y $q(x)$ representa la carga distribuida aplicada.

A partir de esta ecuación, pueden obtenerse las expresiones para los esfuerzos internos:

$$M(x) = -EI\frac{d^2w(x)}{dx^2}$$

$$V(x) = \frac{dM(x)}{dx}$$

donde $M(x)$ es el momento flector y $V(x)$ el esfuerzo cortante. Las tensiones normales en la fibra a una distancia y del eje neutro se calculan como:

$$\sigma(x,y) = \frac{M(x)\,y}{I}$$

En el caso de vigas con cargas axiales o momentos aplicados alrededor del eje Z, las ecuaciones de equilibrio deben ampliarse para incluir el efecto de las fuerzas longitudinales $N(x)$,

obtenidas a partir de la relación:

$$\frac{dN(x)}{dx} + p_x(x) = 0$$

donde $p_x(x)$ es la carga distribuida en dirección axial.

El cálculo analítico de vigas se realiza aplicando condiciones de contorno que dependen del tipo de apoyo (empotrado, articulado o libre), resolviendo las ecuaciones diferenciales de equilibrio y obteniendo los diagramas de cortante, momento y desplazamiento. En los casos más simples como vigas biapoyadas o en voladizo con cargas uniformes o puntuales, las soluciones pueden obtenerse de forma exacta mediante fórmulas conocidas. Sin embargo, para configuraciones más complejas, con múltiples cargas, secciones variables o apoyos intermedios, la resolución analítica resulta impracticable.

Por este motivo, el análisis estructural moderno recurre a métodos numéricos que permiten resolver el problema de forma generalizada. Entre ellos, el *Método de los Elementos Finitos* (MEF) se ha consolidado como la herramienta más versátil y precisa, ya que permite discretizar la viga en elementos y formular las ecuaciones de equilibrio en forma matricial, adaptándose fácilmente a cualquier tipo de geometría, condición de contorno o esquema de carga [1]

Limitaciones del cálculo manual

Aunque el cálculo analítico de vigas basado en la teoría de Euler–Bernoulli ofrece resultados exactos para un conjunto limitado de configuraciones, su aplicabilidad práctica se ve restringida por múltiples factores relacionados con la complejidad geométrica, las condiciones de contorno y la naturaleza de las cargas. La formulación diferencial que describe la flexión de una viga requiere integrar varias veces las ecuaciones de equilibrio y aplicar correctamente las condiciones de contorno para obtener el campo de desplazamientos. Este procedimiento, aunque viable en casos simples, se vuelve ineficiente o directamente inabordable en estructuras reales.

Entre las principales limitaciones del cálculo manual pueden destacarse las siguientes:

- **Geometrías complejas:** las expresiones analíticas sólo son válidas para vigas de sección constante. Cuando la sección varía a lo largo del eje, el momento de inercia $I(x)$ deja de ser constante, lo que impide obtener soluciones cerradas.
- **Condiciones de contorno múltiples o mixtas:** la presencia de apoyos intermedios, empotramientos parciales o combinaciones de restricciones dificulta enormemente la resolución manual de las ecuaciones diferenciales.
- **Cargas no uniformes o combinadas:** cuando la viga está sometida simultáneamente a cargas distribuidas, puntuales o momentos aplicados, la superposición de efectos requiere realizar integraciones sucesivas, lo que incrementa el riesgo de errores y la complejidad algebraica.

- **Materiales heterogéneos o anisótropos:** la variación espacial del módulo de elasticidad $E(x)$ introduce no linealidades que imposibilitan la resolución exacta mediante métodos tradicionales.
- **Análisis de sistemas estructurales:** el estudio de pórticos, entramados o estructuras continuas formadas por múltiples vigas conectadas requiere formular sistemas de ecuaciones simultáneas de gran tamaño, lo que excede la capacidad del cálculo manual.

Además de estas limitaciones prácticas, el cálculo manual carece de flexibilidad y automatización. Cada modificación en las cargas, geometría o apoyos obliga a repetir todo el proceso de resolución, lo que lo hace ineficiente en entornos de diseño iterativo o de optimización estructural. En consecuencia, resulta necesario recurrir a métodos numéricos que generalicen la formulación teórica y permitan resolver de manera sistemática cualquier configuración estructural.

Entre estos métodos, el MEF destaca por su capacidad para discretizar la estructura en elementos simples y resolver las ecuaciones de equilibrio de forma matricial. Este enfoque no solo amplía el rango de problemas tratables, sino que también facilita la automatización y la integración con procesos computacionales avanzados, como el aprendizaje automático, que se abordan en capítulos posteriores.

Fundamentos del Método de los Elementos Finitos aplicado a vigas 2D

El *Método de los Elementos Finitos* constituye una de las herramientas numéricas más empleadas en ingeniería para la resolución de problemas estructurales. Su principal ventaja radica en la capacidad de analizar estructuras con geometrías y condiciones de contorno arbitrarias mediante la discretización del dominio (Ω) continuo en un número finito de elementos más simples, interconectados en nodos como se muestra en la Figura 2.3.

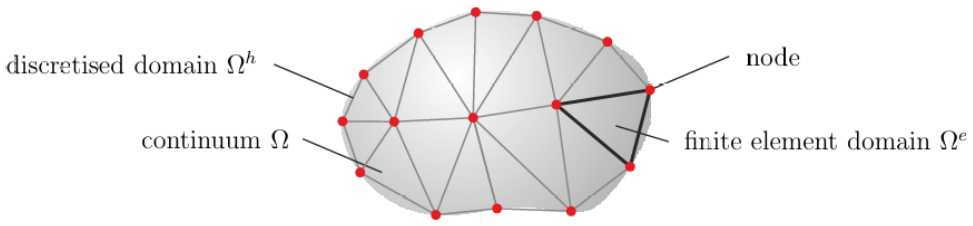


Figura 2.3. Aproximación lineal de la geometría

En el caso de estructuras tipo viga, resulta especialmente eficiente el uso de elementos unidimensionales (1D). Este tipo de elementos permiten representar el comportamiento axial, flexional y de cortante de una barra sin necesidad de modelar su geometría tridimensional completa. De este modo, se reduce drásticamente el número de grados de libertad y, por tanto, el coste computacional de la simulación, manteniendo una precisión adecuada para el análisis de estructuras esbeltas. Esta simplificación es especialmente ventajosa cuando se requiere generar un elevado número de modelos, como en el presente trabajo, orientado a la creación de una base de datos para el entrenamiento de modelos de aprendizaje automático.

La Figura 2.4 muestra un modelo mallado mediante elementos unidimensionales (1D), los cuales conforman el modelo MEF global. Cada elemento posee su propio sistema de referencia local, donde se definen sus propiedades y ecuaciones de comportamiento. De este modo, el modelo de elementos finitos se construye a partir de la unión de múltiples elementos individuales, que al ensamblarse forman la estructura global.

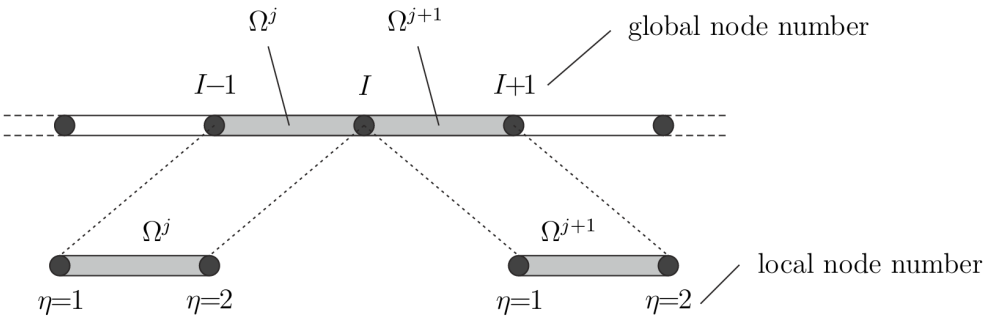


Figura 2.4. Representación de la discretización de un elemento 1D

En el caso de una viga plana contenida en el sistema XY, cada elemento de viga se define por dos nodos situados en sus extremos, con tres grados de libertad por nodo: desplazamiento axial u en la dirección X, desplazamiento transversal v en la dirección Y y rotación θ_z alrededor del eje Z. El vector de desplazamientos nodales del elemento puede expresarse como [4]:

$$\mathbf{d_e} = \begin{bmatrix} u_1 \\ v_1 \\ \theta_1 \\ u_2 \\ v_2 \\ \theta_2 \end{bmatrix}$$

La relación entre las fuerzas nodales y los desplazamientos se establece mediante la matriz de rigidez del elemento:

$$\mathbf{f_e} = \mathbf{K_e}\,\mathbf{d_e}$$

Para un elemento de viga-columna 2D (que considera esfuerzos axiales y de flexión en el plano), la matriz de rigidez local en coordenadas del elemento se expresa como:

$$\mathbf{K_e} = \begin{bmatrix} \frac{AE}{L} & 0 & 0 & -\frac{AE}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{2EI}{L} \\ -\frac{AE}{L} & 0 & 0 & \frac{AE}{L} & 0 & 0 \\ 0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{2EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix}$$

En la matriz de la página anterior, A es el área de la sección transversal, E el módulo de elasticidad, I el momento de inercia y L la longitud del elemento.

Cuando la viga no está alineada con el eje X, se requiere transformar las magnitudes del sistema local al sistema global de coordenadas. Para ello se utiliza una matriz de transformación $\mathbf{T}$, de modo que la rigidez en coordenadas globales se obtiene como:

$$\mathbf{K_e^{(g)}} = \mathbf{T}^T \mathbf{K_e} \mathbf{T}$$

Tras el cálculo de las matrices de rigidez de todos los elementos, se procede al ensamblaje en una matriz global $\mathbf{K}$, que relaciona el vector global de desplazamientos $\mathbf{d}$ con el de fuerzas nodales externas $\mathbf{F}$:

$$\mathbf{K}\,\mathbf{d} = \mathbf{F}$$

El sistema se resuelve imponiendo las condiciones de contorno adecuadas por ejemplo, desplazamientos nulos en apoyos o empotramientos y aplicando las cargas externas en los nodos correspondientes. La resolución del sistema proporciona los desplazamientos nodales, a partir de los cuales pueden obtenerse los esfuerzos internos, tensiones y deformaciones de cada elemento.

El uso de elementos 1D de viga constituye, por tanto, una solución equilibrada entre precisión y eficiencia. Este enfoque permite representar de forma fiable el comportamiento estructural de vigas rectas bajo cargas axiales, transversales y momentos flectores, al tiempo que posibilita la generación automática de miles de modelos con distintos materiales, secciones y condiciones de contorno, lo cual resulta esencial para la aplicación de técnicas de aprendizaje automático basadas en grandes conjuntos de datos.

Tipos de secciones transversales y propiedades geométricas

El comportamiento estructural de una viga depende de manera directa tanto de las propiedades geométricas de su sección transversal como de las características mecánicas del material empleado. Estos parámetros determinan la rigidez del elemento frente a cargas axiales, cortantes y de flexión, y condicionan la magnitud de las tensiones y desplazamientos que se generan bajo una determinada carga.

Influencia de la geometría de la sección La geometría de la sección transversal se caracteriza mediante propiedades como el área A y el momento de inercia I, los cuales intervienen de forma explícita en la formulación de las ecuaciones del comportamiento estructural. Mientras que el área define la capacidad portante frente a esfuerzos axiales, el momento de inercia determina la resistencia del elemento frente a la flexión. Un valor mayor de I implica una mayor rigidez a flexión y, por tanto, menores desplazamientos y tensiones en servicio.

En este proyecto se han considerado varios tipos de secciones transversales representativas de distintas configuraciones estructurales:

- **Sección rectangular:** de dimensiones base b y altura h, con un momento de inercia dado por $I = \frac{b\,h^3}{12}$. Es una de las secciones más comunes y presenta un comportamiento simétrico frente a la flexión en el plano principal.
- **Sección rectangular hueca:** definida por dimensiones exteriores b_e, h_e y dimensiones interiores b_i, h_i. Su momento de inercia se obtiene como la diferencia entre las inercias de los rectángulos exterior e interior:

  $$I = \frac{b_e\,h_e^3 - b_i\,h_i^3}{12}$$

  Esta configuración optimiza la relación rigidez-peso, reduciendo masa sin pérdida significativa de capacidad resistente.
- **Sección circular maciza:** de radio r, cuyo momento de inercia respecto al eje neutro es $I = \frac{\pi\,r^4}{4}$. Se emplea en elementos donde la carga puede actuar en distintas direcciones y se requiere un comportamiento isotrópico.
- **Sección circular hueca:** de radios exterior r_e e interior r_i, con momento de inercia

  $$I = \frac{\pi}{4}(r_e^4 - r_i^4)$$

  Este tipo de sección presenta una elevada eficiencia estructural, combinando buena rigidez a flexión y torsión con un peso reducido.
- **Sección en I:** formada por un alma y dos alas, concentra la mayor parte del material lejos del eje neutro, maximizando el momento de inercia con una cantidad mínima de material. Por ello, ofrece una rigidez a flexión muy elevada con un peso relativamente bajo, siendo la más utilizada en estructuras metálicas.

En general, secciones con un momento de inercia mayor reducen los desplazamientos verticales y las tensiones máximas bajo una misma carga, mientras que las secciones huecas o aligeradas permiten mantener una buena rigidez con menor masa, lo que resulta ventajoso en aplicaciones donde el peso es un factor crítico.

Influencia del material El material constituye otro de los factores determinantes en el comportamiento de una viga. Sus propiedades mecánicas principalmente el módulo de elasticidad E y el límite elástico σ_y influyen directamente en la rigidez y la capacidad resistente del elemento. El módulo E interviene en la relación constitutiva entre tensión y deformación ($\sigma = E\varepsilon$), de modo que materiales con un valor de E mayor presentan menores deformaciones bajo una misma carga aplicada [5].

En el presente trabajo se han considerado cuatro materiales metálicos representativos: acero al carbono, acero inoxidable, aluminio y titanio. Los dos primeros presentan una elevada rigidez y resistencia, lo que los hace idóneos para estructuras sometidas a grandes esfuerzos, mientras que el aluminio y el titanio destacan por su menor densidad y buena relación resistencia-peso, siendo habituales en aplicaciones donde el peso es un factor crítico. De esta forma, la selección del material implica un compromiso entre rigidez, peso y capacidad de deformación, criterios que condicionan tanto el comportamiento estructural como la respuesta obtenida por los modelos de predicción desarrollados.

Tensión equivalente de Von Mises y criterio de fluencia En análisis estructural de materiales dúctiles se emplea de forma predominante el criterio de energía de distorsión, cuyo indicador escalar es la tensión equivalente de Von Mises. Este escalar sintetiza un estado triaxial de tensiones en una única magnitud comparables con el límite elástico del material, de modo que la fluencia se produce cuando

$$\sigma_{\mathrm{vm}} \;\geq\; \sigma_y.$$

En términos de las tensiones principales $\sigma_1, \sigma_2, \sigma_3$, la definición clásica es [3]:

$$\sigma_{\mathrm{vm}} \;=\; \sqrt{\tfrac{1}{2}\Big[(\sigma_1-\sigma_2)^2+(\sigma_2-\sigma_3)^2+(\sigma_3-\sigma_1)^2\Big]} \;=\; \sqrt{3\,J_2},$$

donde J_2 es el segundo invariante del desviador de tensiones. En estado de tensión plana $(\sigma_3 = 0)$, habitual en láminas o cuando las tensiones fuera del plano son despreciables, se obtiene

$$\sigma_{\mathrm{vm}} \;=\; \sqrt{\sigma_x^2+\sigma_y^2-\sigma_x\sigma_y+3\,\tau_{xy}^2}.$$

En vigas sometidas principalmente a flexión y cortante, la contribución dominante a σ_{vm} proviene de la tensión normal de flexión y, en menor medida, de la cortadura. Los solvers de elementos finitos para elementos 1D (como CBEAM) calculan la envolvente de σ_{vm} en fibras de la sección, lo que permite identificar directamente la zona crítica del elemento.

Razones para utilizar σ_{vm} como objetivo de entrenamiento:

- Es invariante frente a rotaciones de ejes y agrega de forma coherente estados multiaxiales en un único escalar interpretable frente a σ_y.
- Presenta una superficie de fluencia suave y diferenciable, más apropiada para ajuste numérico y para modelos de aprendizaje que la alternativa de Tresca, que es más conservadora pero no suave.
- Correlaciona bien con el inicio de la plasticidad en materiales metálicos dúctiles isotrópicos, que son los materiales considerados en este trabajo.
- Está disponible de forma directa en la salida del solver, reduciendo ambigüedades de posproceso y facilitando la trazabilidad entre simulación y modelo.

Limitaciones y alcance. El uso de σ_{vm} es adecuado para materiales dúctiles en régimen elástico-lineal hasta las proximidades de σ_y. No es un criterio de rotura frágil, para materiales cuasi-frágiles o fenómenos dominados por tracción pura podrían ser preferibles otros indicadores. En este proyecto, centrado en aceros, aluminio y titanio en el marco elástico, σ_{vm} resulta un objetivo robusto y físicamente significativo para el entrenamiento y evaluación de los modelos predictivos, ya que todos los materiales seleccionados son metales dúctiles.

2.1.3. Aprendizaje automático

En los últimos años, el aprendizaje automático se ha consolidado como una herramienta de gran utilidad en el ámbito de la ingeniería, gracias a su capacidad para identificar patrones complejos y realizar predicciones precisas a partir de grandes volúmenes de datos. A diferencia de los métodos analíticos o numéricos tradicionales, que requieren la formulación explícita de las ecuaciones que rigen el comportamiento físico, los modelos de ML aprenden directamente las relaciones entre las variables de entrada y salida a partir de los datos disponibles.

En el contexto del análisis estructural, esta aproximación resulta especialmente valiosa, ya que permite aproximar el comportamiento de elementos como las vigas sin necesidad de realizar una simulación completa mediante el Método de los Elementos Finitos para cada caso. Una vez entrenado, el modelo es capaz de predecir en cuestión de milisegundos variables estructurales de interés como la tensión máxima y el desplazamiento máximo a partir de las características geométricas, de material y de carga del elemento.

El proceso general del aprendizaje automático puede dividirse en varias etapas: recopilación y preparación de los datos, selección de las variables más representativas (*feature engineering*), elección del modelo, entrenamiento mediante un conjunto de datos de entrenamiento y posterior validación del rendimiento con un conjunto de prueba independiente. La calidad de las predicciones depende tanto de la representatividad de los datos de entrada como de la capacidad del modelo para generalizar patrones no vistos durante el entrenamiento.

En este proyecto se han empleado dos tipos de modelos supervisados de regresión con el objetivo de comparar su precisión y comportamiento: un HistGradientBoostingRegressor (HGBR) y una red neuronal artificial (RNA). El primero pertenece a la familia de los modelos basados en árboles de decisión y se caracteriza por su eficiencia y robustez frente a datos tabulares con posibles interacciones no lineales. La red neuronal, por su parte, constituye un enfoque más flexible capaz de aproximar funciones de alta complejidad, aunque requiere un mayor número de parámetros y un proceso de entrenamiento más cuidadoso para evitar el sobreajuste.

El análisis comparativo entre ambos enfoques permite evaluar las ventajas e inconvenientes de cada técnica en la predicción de resultados estructurales, y valorar su idoneidad para la integración en herramientas de diseño predictivo y optimización estructural.

Fundamentos del aprendizaje automático y la regresión supervisada

En regresión supervisada se busca un modelo f_θ que aproxime la relación $y \approx f_\theta(\mathbf{x})$ entre un conjunto de características $\mathbf{x} \in \mathbb{R}^p$ (propiedades geométricas, de material, condiciones de contorno y cargas) y una respuesta estructural continua y (p. ej., tensión o desplazamiento máximos). El entrenamiento consiste en minimizar el riesgo empírico

$$\hat{\mathcal{R}}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}\big(y_i, f_\theta(\mathbf{x}_i)\big),$$

con pérdidas típicas MSE/MAE y regularización para favorecer la generalización [6]. La validación (hold-out, CV k-fold o CV con grupos por geometría/sección) estima el rendimiento fuera de la muestra [7] y guía la selección de hiperparámetros, evitando fuga de información cuando existen instancias muy correlacionadas.

En problemas estructurales, el diseño de rasgos (*feature engineering*) es clave: incorporar escalas físico-dimensionadas (p. ej., términos del tipo L^3/EI o L^4/EI) y razones adimensionales (esbeltez, h/b, r_e/t, etc.) mejora la estabilidad numérica y reduce la varianza del estimador al alinear el modelo con la teoría de vigas. Esta idea es compatible con modelos de muy distinta naturaleza:

- **Modelos basados en árboles** (p. ej., HistGradientBoostingRegressor): capturan interacciones y no requieren estandarización de entradas; la regularización efectiva surge de la profundidad, el tamaño mínimo de hoja, el *learning rate* y el *subsampling*.
- **Redes neuronales**: requieren escalado/normalización de entradas; su mayor capacidad para relaciones no lineales se controla con *early stopping*, L2 y/o *dropout* [8].

El rendimiento debe reportarse con métricas absolutas (RMSE, MAE) y relativas (MAPE cuando procede), y, en este contexto, con *errores normalizados por escalas físicas* (p. ej., desplazamiento normalizado por L^3/EI o L^4/EI según el esquema de carga/contorno), lo que facilita interpretar la precisión en términos de la teoría de vigas.

Rasgos físico-informados y escalas

Con el objetivo de guiar el aprendizaje mediante conocimiento físico, se han definido características que reflejan directamente los mecanismos de rigidez y la influencia de las condiciones de contorno y de las cargas. A nivel conceptual:

Propiedades de sección y material Se emplean área A, momento de inercia a flexión I_z, módulo resistente $W_z = I_z/(h/2)$ y una aproximación de la inercia torsional J (tipo Saint-Venant para secciones delgadas [3]) como indicadores de rigidez. El módulo elástico E se incorpora a través de EI, magnitud central en la teoría de vigas, junto con densidad o límites resistentes cuando procede. Derivados útiles:

$$EI, \quad \frac{I_z}{A^2}, \quad \frac{J}{A}, \quad r_z = \sqrt{\frac{I_z}{A}}, \quad \frac{L}{r_z}.$$

Estas magnitudes capturan rigideces efectivas y esbeltez, que correlacionan con desplazamientos y tensiones.

Condiciones de contorno: longitud efectiva Las constantes de la solución de la viga dependen del tipo de apoyo/empotramiento. Para incorporarlo de forma compacta se utiliza un *factor de longitud efectiva* K (función del par de apoyos) y sus potencias/interacciones:

$$L_{\text{eff}} = K\,L, \qquad \frac{L_{\text{eff}}^3}{EI},\ \frac{L_{\text{eff}}^4}{EI},\ \frac{EI}{L_{\text{eff}}},\ \frac{EI}{L_{\text{eff}}^2}.$$

La Tabla 2.2 muestra los valores típicos de K (referencia [3]) para determinar la longitud efectiva en una viga.

Tabla 2.2. Factores de longitud efectiva K

Configuración estructural	**K**
Voladizo: empotrado–libre	2.0
Voladizo invertido: libre–empotado	2.0
Biapoyada: articulado–articulado	1.0
Empotrada en ambos extremos: empotrado–empotado	0.5
Empotrado–articulado (o articulado–empotrado)	0.7

Así se aproximan, de manera unificada, las distintas constantes de proporcionalidad de los desplazamientos en función del contorno, mejorando la transferibilidad del modelo entre configuraciones.

Cargas agregadas y escalas de respuesta Se agregan componentes de carga por dirección (p. ej., F_Y, F_X) y momentos aplicados en Z, junto con posiciones adimensionales x/L cuando existen. Se definen escalas físicas que aproximan el orden de magnitud de la respuesta:

$$\text{escala de desplazamiento:}\quad \delta_{\text{scale}} \sim \frac{F_Y L^3}{EI},\quad \delta_{\text{scale}}^{(\text{eff})} \sim \frac{F_Y L_{\text{eff}}^3}{EI},\quad \frac{F_Y}{L}\ (\text{intensidad}),$$

$$\text{escala de tensión:}\quad \sigma_{\text{scale}} \sim \frac{M_{\text{ref}}}{W_z},$$

donde M_{ref} es una relación del momento máximo (p. ej., Pab/L para cargas puntuales en viga simplemente apoyada, extendido de forma aditiva para varias acciones). Estas escalas, y sus interacciones, informan al modelo sobre cómo varían $w_{\max}$ y $\sigma_{\max}$ con L, EI, la distribución de cargas y el contorno.

Interacciones y transformaciones Se incluyen interacciones físicamente motivadas (p. ej., $\sigma_{\text{scale}} \cdot L_{\text{eff}}$, $L_{\text{eff}} \cdot L^3/EI$) y transformaciones logarítmicas suaves ($\log(1+x)$) para estabilizar rangos amplios y aproximar relaciones potenciales. En árboles de gradiente estas expansiones reducen la profundidad necesaria; en redes neuronales facilitan la optimización y la calibración de la salida.

Implicaciones para HGBR y RNA Los modelos tipo HGBR aprovechan de forma nativa particiones por región de características sin requerir estandarización; las escalas físico-informadas mejoran su sesgo inductivo y la extrapolación local. En la RNA, además del escalado de entradas, el uso de L_{eff}, EI y razones adimensionales reduce el *condicionamiento* del problema y acelera la convergencia. En ambos casos, la validación por grupos (p. ej., por tipo de sección o familia geométrica) evita sobreestimar el rendimiento cuando hay instancias muy similares.

En conjunto, este diseño de características integra de forma explícita la teoría de vigas (rigidez por flexión EI, efecto del contorno vía K, escalas L^3/EI y L^4/EI y métricas de sección W_z, r_z) dentro del proceso de aprendizaje, lo que permite a los modelos predecir con mayor fidelidad las variables objetivo (tensión y desplazamiento máximos) y facilita comparar, en condiciones equitativas, el desempeño de HistGradientBoostingRegressor y de la red neuronal.

Modelo basado en árboles: HistGradientBoostingRegressor

[9], [10]

El HistGradientBoostingRegressor (HGBR) es un método de *boosting* de gradiente para regresión que utiliza árboles de decisión como aprendices débiles y una discretización en histogramas para acelerar el entrenamiento. Su idea central es construir un modelo aditivo

$$F_M(\mathbf{x}) = \sum_{m=1}^{M} \nu\, h_m(\mathbf{x}),$$

donde h_m es un árbol de decisión ajustado sobre los residuos (o, más rigurosamente, el *negativo del gradiente* de la pérdida) calculados respecto al modelo acumulado F_{m-1}, y $\nu \in (0, 1]$ es el *learning rate* (*shrinkage*). En cada iteración,

$$g_i^{(m)} = -\left.\frac{\partial \mathcal{L}\big(y_i, F(\mathbf{x}_i)\big)}{\partial F}\right|_{F=F_{m-1}},$$

y el árbol h_m se ajusta para aproximar $g^{(m)}$. El proceso reduce iterativamente el error de entrenamiento manteniendo un buen control de la varianza gracias al *shrinkage*, la profundidad limitada de los árboles y restricciones de hoja.

Discretización por histogramas Antes de buscar cortes, HGBR binariza cada característica en un número fijo de bloques (*bins*). Esto: reduce el coste computacional de evaluar candidatos de partición, aporta regularización adicional al suavizar ruido, hace que el método sea robusto a escalado y valores extremos moderados.

Pérdida, regularización y robustez La pérdida típica es MSE, aunque en presencia de atípicos y heteroscedasticidad puede emplearse *Huber* o *Quantile* (para intervalos de predicción). La regularización se controla mediante:

- Learning rate ν (p. ej., 0.02–0.1): valores menores requieren más árboles pero suelen generalizar mejor.
- Estructura del árbol: `max_leaf_nodes` (p. ej., 16–64) o `max_depth` (3–8).
- Tamaño de hoja: `min_samples_leaf` (p. ej., 20–200) limita el sobreajuste local.
- L2 sobre los valores de hoja (`l2_regularization`) y submuestreo estocástico de filas/columnas cuando esté disponible.
- Bins: `max_bins` (p. ej., 64–255) controla la granularidad de cortes.

Con *early stopping* (fracción de validación 10–20 %), el número de iteraciones M se determina automáticamente (típicamente unas centenas a mil+).

Preprocesado y rasgos Los árboles no requieren estandarización. No obstante, la inclusión de rasgos *físico-informados* (p. ej., EI, L_{eff}, L_{eff}^3/EI, W_z, r_z, esbeltez L/r_z) mejora el sesgo inductivo: el modelo parte de relaciones cercanas a la teoría de vigas y necesita menos profundidad para captarlas. Las variables categóricas (tipo de apoyo, tipo de sección) deben tratarse como categóricas (one-hot o manejo nativo si la implementación lo permite) para evitar una falsa ordenación.

Diagnóstico e interpretabilidad

- Importancia de características (*gain*/permutación) para identificar los predictores dominantes (suele destacar EI, L_{eff}, escalas $\sim L^3/EI$ y W_z en tensiones).
- Dependencia parcial (PDP) y *ICE* para estudiar cómo varían $w_{\text{máx}}$ y $\sigma_{\text{máx}}$ con L, EI, F_Y o M_Z, manteniendo el resto fijo.
- Análisis de errores por familia (sección, apoyo, rango de cargas) para detectar sesgos sistemáticos.

Ventajas y limitaciones Ventajas: entrenamiento rápido, robustez a valores atípicos (*outliers*) leves (con Huber), poco preprocesado, buena interpretación local y fuerte rendimiento en datos tabulares. Limitaciones: extrapola peor fuera del dominio de entrenamiento (típico en árboles) y puede infraestimar tendencias suaves si falta profundidad o bins.

Modelo de red neuronal para regresión

Las redes neuronales artificiales (RNA) aproximan funciones no lineales mediante la composición de capas afines y activaciones. Para un problema de regresión con dos salidas (tensión máxima y desplazamiento máximo), resulta natural una arquitectura *multitarea* con un *tronco* común y, opcionalmente, *cabezas* específicas:

$$\hat{\mathbf{y}} = \left(\hat{y}^{(\sigma)}, \hat{y}^{(\delta)}\right) = f_\theta(\mathbf{x}),$$

donde f_θ es una MLP (perceptrón multicapa o en inglés *multilayer perceptron*) con L capas ocultas.

Arquitectura y activaciones

- Capas ocultas: 2–4 capas densas con 64–256 neuronas suelen equilibrar capacidad y generalización en datos tabulares.
- Activación: ReLU o GELU por su estabilidad y capacidad para modelar no linealidades.
- Normalización: *BatchNorm* o *LayerNorm* (opcional) para estabilizar el entrenamiento.
- Multicabeza: una cabeza para tensión máxima ($\hat{\sigma}_{\text{máx}}$) y otra para desplazamiento máximo ($\hat{\delta}_{\text{máx}}$) permite pérdidas y escalas diferentes por tarea.

Pérdida, escalado de objetivos y optimización Las RNA son sensibles a escalas. Es recomendable:

- Estandarizar entradas continuas (media cero, varianza unitaria) y codificar categorías.
- Normalizar objetivos por una escala física (p. ej., $\tilde{\delta} = \delta/(F_Y L_{\text{eff}}^3/EI)$, $\tilde{\sigma} = \sigma/(M_{\text{ref}}/W_z)$). El modelo aprende $\tilde{y}$ y, al inferir, se *desnormaliza*. Esto reduce la heteroscedasticidad y acelera la convergencia.
- Pérdida: MSE o Huber por cabeza; suma ponderada si las tareas tienen magnitudes distintas. Las ponderaciones pueden fijarse para igualar las varianzas de cada objetivo.

Regularización y control del sobreajuste

- Early stopping con paciencia 20–50 épocas sobre una validación separada (o CV).
- Weight decay (L2 10^{-5}–10^{-3}) y dropout ligero (0.05–0.2) en capas intermedias.
- Data splitting con grupos (por geometría/tipo de sección/apoyos) para evitar fuga de información.

Incertidumbre y calibración Además del valor puntual, pueden estimarse intervalos:

- MC Dropout: mantener activo el *dropout* en inferencia y muestrear múltiples pases.
- Pérdida cuantílica: dos salidas por objetivo (percentiles p5/p95) para intervalos directos.

Diagnóstico e interpretabilidad

- Curvas de aprendizaje (entrenamiento vs. validación) para vigilar sobreajuste/infraparametrización.
- Sensibilidades (gradientes, *saliency*) y PDP sobre entradas clave (L_{eff}, EI, F_Y, M_Z), interpretando tendencias físicas (p. ej., $w_{\text{máx}} \uparrow$ con L y $\downarrow$ con EI).

Ventajas y limitaciones Presenta una gran capacidad para relaciones altamente no lineales y para compartir información en multitarea. Limitaciones: requieren más cuidado en preprocesado/escalado, son menos interpretables de base y pueden sobreajustar si no se regularizan y validan correctamente.

Comparación entre HGBR y RNA

	HGBR	**Red neuronal**
Preprocesado	Mínimo, robusto a escalas	Requiere estandarización y normalización
Capacidad no lineal	Media	Muy alta (aproxima funciones suaves/compuestas)
Regularización	ν, hojas, profundidad, L2, bins, early stopping.	Early stopping, weight decay, dropout, arquitectura
Extrapolación	Limitada fuera del dominio visto	También limitada; mejora con rasgos físico-informados
Interpretabilidad	Buena (importancias, PDP/ICE)	Media (curvas de sensibilidad)
Robustez a ruido	Alta (con Huber y hojas mínimas)	Media, sensible a escalas y atípicos si no se cuida
Entrenamiento	Rápido.	Más costoso, requiere ajuste fino

En este proyecto, ambos modelos se entrenan con la misma partición/validación (preferiblemente cruzada por grupos) y se evalúan con métricas absolutas (RMSE, MAE) y relativas (MAPE), además de errores normalizados por escalas físicas ($\sim L_{\text{eff}}^3/EI$ para desplazamientos; M/W_z para tensiones). La comparación conjunta permite valorar precisión y robustez.

2.2 Marco Tecnológico

Para llevar a cabo este proyecto se ha empleado un conjunto de herramientas tecnológicas que cubren todas las etapas del flujo de proyecto, desde la generación y resolución de modelos estructurales hasta el entrenamiento de los modelos de aprendizaje automático y el desarrollo de la aplicación final. La correcta integración de estas herramientas ha permitido automatizar el proceso de simulación, procesar grandes volúmenes de resultados y construir una interfaz interactiva que facilita el uso de los modelos desarrollados.

- **Altair HyperMesh:** software de preprocesado FEM utilizado para modelar automáticamente vigas con diferentes combinaciones de geometría, material, condiciones de contorno y cargas. La automatización de la generación de modelos se ha llevado a cabo mediante el uso de scripts en lenguaje TCL, permitiendo crear de forma eficiente miles de configuraciones estructurales distintas. HyperMesh también incluye Hyperview, módulo de posprocesado de la suite Altair HyperWorks, diseñado para visualizar, analizar e interpretar los resultados obtenidos de simulaciones por elementos finitos.
- **Altair OptiStruct:** *solver* de elementos finitos encargado de resolver los modelos generados en HyperMesh. OptiStruct proporciona resultados precisos de desplazamientos, tensiones y reacciones nodales, los cuales constituyen la base del conjunto de datos empleado para el entrenamiento de los modelos de aprendizaje automático. En el Anexo C se describe en detalle el funcionamiento de OptiStruct y su integración con HyperMesh, explicando cómo ambas herramientas interactúan en el proceso de modelado y análisis estructural.
- **PyNastran:** librería de Python utilizada para la lectura y procesamiento automatizado de archivos de resultados FEM en formato `.op2`. Gracias a esta herramienta se han extraído de forma automática los valores de interés como desplazamientos, tensiones y reacciones garantizando la trazabilidad y consistencia de los datos obtenidos.
- **Python:** lenguaje principal del proyecto. Se ha empleado tanto para la generación de combinaciones estructurales y el preprocesamiento de los datos como para la implementación de los modelos predictivos y el desarrollo de la interfaz gráfica. Su ecosistema de librerías científicas ha permitido cubrir todo el flujo de trabajo de manera integrada.
- **Librerías de aprendizaje automático:** se han utilizado bibliotecas especializadas como scikit-learn, LightGBM, TensorFlow y Keras para la implementación, entrenamiento y validación de los modelos de regresión. Estas herramientas proporcionan algoritmos optimizados y facilitan la experimentación con diferentes arquitecturas y estrategias de entrenamiento.
- **Streamlit:** framework de desarrollo web utilizado para construir la aplicación Alabeam, una herramienta interactiva que permite al usuario obtener predicciones estructurales a partir de los parámetros geométricos, materiales y de carga definidos. Streamlit ofrece una integración directa con Python y una interfaz sencilla para la visualización de resultados en tiempo real.

Capítulo 3. ESTADO DEL ARTE

El uso de técnicas de aprendizaje automático en ingeniería estructural ha crecido de forma notable en la última década. La motivación principal es doble. Por un lado, los métodos de elementos finitos ofrecen gran precisión pero presentan costes computacionales elevados cuando se exploran espacios de diseño amplios y múltiples combinaciones de materiales, secciones y condiciones de contorno. Por otro lado, los modelos de aprendizaje automático pueden aproximar respuestas estructurales de interés con tiempos de cálculo muy reducidos, lo que habilita predimensionados interactivos, análisis masivos y aplicaciones de asistencia al diseño. La literatura reciente recoge tanto estudios de revisión como demostraciones aplicadas sobre vigas y marcos, así como avances en modelos informados por la física. A continuación se sintetizan las líneas más relevantes para este trabajo.

3.1 Modelos *surrogate* para acelerar o sustituir el FEM

Los modelos *surrogate* se entrenan con datos obtenidos de simulaciones de alta fidelidad para predecir con rapidez magnitudes estructurales. Diversas revisiones y contribuciones muestran que esta estrategia permite mantener errores controlados mientras reduce el tiempo de cómputo de forma sustancial. En el ámbito estructural, se emplean métodos de reducción de orden y aproximaciones de respuesta, así como modelos puramente basados en datos que actúan como sustitutos del FEM en tareas de evaluación y toma de decisiones en tiempo casi real [11]-[13].

3.2 Aprendizaje profundo para campos de tensiones y desplazamientos

Cuando el objetivo es predecir campos completos de respuesta, las redes neuronales profundas han mostrado gran capacidad para aproximar distribuciones de tensiones o campos de desplazamiento a partir de información geométrica y de carga. Resultados representativos demuestran errores bajos frente al FEM y un potencial para integrarse en cadenas de cálculo aceleradas, especialmente en tareas de evaluación rápida y diseño asistido [11], [14].

3.3 Ensembles de árboles en problemas tabulares estructurales

En problemas con entradas tabulares que recogen propiedades de sección, material y cargas, los ensambles de árboles de gradiente, y en particular la variante de histograma, ofrecen precisión, robustez y un preprocesado moderado. Además cuentan con soporte nativo para valores ausentes, parada temprana y restricciones de monotonía, lo que facilita flujos de trabajo reproducibles y eficientes en ingeniería [15], [16]. Esta familia es un buen contrapunto a las redes neuronales cuando la estructura del dato es tabular y el diseño de características está informado por la física.

3.4 Redes informadas por la física para vigas

Las redes neuronales informadas por la física incorporan ecuaciones de equilibrio y relaciones constitutivas en la función de pérdida. En vigas de *Euler–Bernoulli* se han publicado marcos que resuelven problemas directos e inversos y que mejoran la eficiencia de datos, con aplicaciones a respuestas dinámicas bajo cargas móviles o a escenarios con medidas escasas. Estos trabajos señalan una vía de integración entre conocimiento físico y aprendizaje automático para problemas de viga y pórtico [11], [17].

3.5 Aplicaciones específicas en vigas como *surrogate* del FEM

Existen contribuciones que sustituyen explícitamente el FEM de vigas por modelos de aprendizaje para predecir respuestas máximas y transitorias. En particular, se han presentado comparativas entre árboles potenciados y redes neuronales, validando los resultados frente a soluciones FEM. Estas evidencias son directamente relevantes para este TFM, centrado en vigas en el plano XY con cargas en X y Y y momentos en Z, diferentes secciones y varios materiales [18].

3.6 Síntesis y contribución del trabajo

Las dos corrientes más útiles para el problema abordado son las siguientes. Primero, los ensambles basados en gradiente con histograma resultan muy eficaces en datos tabulares con rasgos físico informados. Segundo, las redes neuronales ofrecen mayor flexibilidad para capturar no linealidades complejas y constituyen la base de enfoques informados por la física. La contribución de este trabajo consiste en construir un conjunto de datos FEM específico de vigas 2D, entrenar y comparar un HistGradientBoostingRegressor y una red neuronal sobre el mismo espacio de entrada, y entregar una aplicación interactiva que permita elegir el modelo en función del compromiso entre rapidez, precisión y robustez [11], [15].

3.7 Modelos industriales: Altair PhysicsAI como *surrogate* CAE de propósito general

Dentro de las soluciones comerciales destaca Altair PhysicsAI, integrado en la plataforma HyperWorks, cuyo objetivo es entrenar modelos predictivos a partir de datos de simulación y ensayo para realizar predicciones de física en nuevos diseños con gran rapidez. La propuesta se alinea con la tendencia de sustituir o acelerar solvers tradicionales mediante modelos de aprendizaje automático entrenados con datos CAE históricos y reutilizables entre proyectos y geometrías similares [19], [20].

PhysicsAI se presenta como una tecnología de *geometric deep learning* que aprende la relación entre forma y desempeño para cualquier física. Según la documentación corporativa, una vez entrenados, los modelos pueden ofrecer predicciones hasta mil veces más rápidas que un análisis FEM convencional, lo que habilita estudios de alternativas y validaciones preliminares en tiempos muy reducidos [21], [22].

La herramienta está expuesta en el entorno de HyperMesh y permite crear conjuntos de datos, entrenar y validar modelos, y generar predicciones directamente sobre modelos CAD o mallas importadas, sin necesidad de parametrizar variables de diseño y admitiendo remallados [23]. La Figura 3.1 muestra la estructura de trabajo de PhysicsAI.

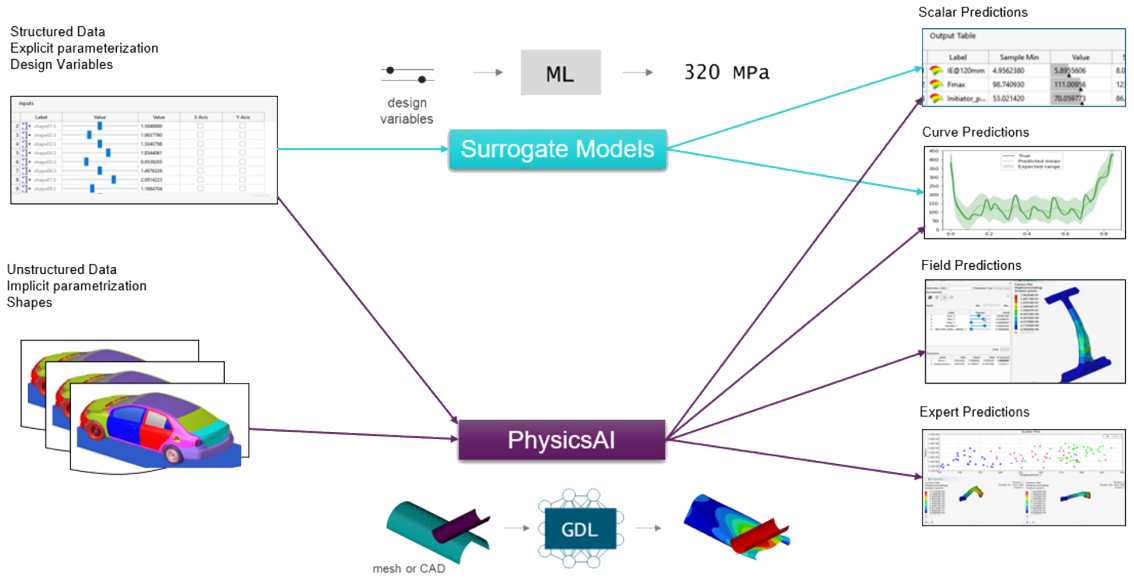


Figura 3.1. Flujo de trabajo de PhysicsAI

En las versiones recientes se han incorporado arquitecturas específicas para el entrenamiento, entre ellas el *Graph Context Neural Simulator* y el *Transformer Neural Simulator*. La introducción de estas variantes busca suavizar contornos de predicción, reducir la sensibilidad al tamaño de malla y mejorar los tiempos en GPU, lo que refuerza su aplicabilidad en escenarios con mallas heterogéneas y requisitos de respuesta interactiva [24]. Diversas guías y casos de uso muestran su despliegue como *solver* de IA dentro de flujos de trabajo de HyperWorks con el lema de un único modelo y un único solver, y con integración en procesos de optimización de diseño acelerados [20], [25], [26].

Relación con el presente trabajo El enfoque de Altair PhysicsAI refuerza la tendencia industrial hacia modelos predictivos que actúan como sustitutos acelerados del FEM. En este trabajo se adopta la misma filosofía al entrenar y comparar un HistGradientBoostingRegressor y una red neuronal sobre un conjunto de datos FEM de vigas 2D, con el objetivo de obtener predicciones rápidas de desplazamiento y tensión máximos. Aunque la solución propuesta es académica y centrada en un dominio específico, se alinea conceptualmente con la estrategia de *surrogate modelling* presente en herramientas industriales como PhysicsAI, y pone de manifiesto las ventajas prácticas de integrar modelos de ML en flujos de diseño estructural.

3.8 Protocolos de validación y comparabilidad

La literatura reciente insiste en la necesidad de protocolos de validación que eviten estimaciones optimistas del rendimiento cuando los datos presentan correlaciones estructurales. En problemas de vigas es común que múltiples instancias compartan geometría o tipo de apoyo. Para garantizar comparabilidad se recomienda: validación cruzada con agrupación por familia geométrica o por sección, separación explícita entre evaluación dentro del dominio y fuera del dominio, por ejemplo dejando fuera un tipo de sección completo para medir extrapolación, uso combinado de métricas absolutas y relativas, y normalización de errores por escalas físicas coherentes con la teoría de vigas. Estas pautas permiten interpretar las diferencias entre modelos y trasladar resultados a escenarios de diseño reales.

3.9 Generalización y extrapolación en modelos supervisados

Los modelos basados en árboles y las redes neuronales muestran comportamientos diferentes cuando se evalúan fuera del rango visto en entrenamiento. Los primeros tienden a ser conservadores en regiones no muestreadas, mientras que las redes neuronales pueden extrapolar suavemente si el diseño de características está bien alineado con la física del problema. En ambos casos la incorporación de variables físico informadas, así como restricciones simples como monotonías esperadas, mejora la estabilidad de la extrapolación y atenúa errores sistemáticos en bordes del dominio.

3.10 Generación de datos y cobertura del espacio de diseño

Como los datos se obtienen por simulación, es importante cubrir bien el espacio de diseño para que el modelo aprenda casos variados y no solo unos pocos. Para ello se combinan tres ideas sencillas:

- Repartir las muestras por tipos de sección y por rangos de carga,
- Usar un muestreo que llene uniformemente las combinaciones continuas (p. ej., *latin hypercube*)
- Añadir más casos justo donde el error del modelo es mayor (*active learning*).

Además, se documentan claramente los límites de validez del conjunto de datos y las zonas con pocas muestras, para evitar extrapolaciones no deseadas y poder interpretar los resultados con cautela.

3.11 Conclusión del Estado del Arte

En conclusión, cuando se dispone de datos de simulación, los modelos supervisados son una vía eficaz para acelerar el análisis estructural con pérdidas mínimas de precisión. La combinación de rasgos informados por la física con métodos robustos en datos tabulares y con redes neuronales permite obtener predicciones fiables a muy bajo coste computacional.

La literatura coincide en cuatro buenas prácticas: validar con agrupación para evitar fugas de información, distinguir con claridad los casos fuera del dominio de entrenamiento, cuantificar la incertidumbre de las predicciones y asegurar una cobertura adecuada del espacio de diseño. Con estas ideas como guía, el siguiente capítulo presenta la metodología de este trabajo: generación del conjunto de datos mediante FEM, diseño de características, y entrenamiento y validación comparativa de un HistGradientBoostingRegressor y de una red neuronal, junto con los criterios de evaluación usados para valorar su desempeño en condiciones de uso realistas.

Capítulo 4. METODOLOGÍA

La metodología seguida en este trabajo se estructura en una secuencia de etapas que abarcan desde la definición paramétrica de vigas y la generación automática de modelos hasta el entrenamiento y la validación de modelos de aprendizaje automático, concluyendo con su integración en una aplicación interactiva. El flujo completo comprende:

1. Diseño paramétrico de la viga y muestreo de combinaciones.
2. Generación automática de modelos en HyperMesh.
3. Resolución FEM con OptiStruct.
4. Extracción de respuestas desde ficheros `.op2` mediante `PyNastran`.
5. Construcción y depuración del conjunto de datos.
6. Preprocesado y diseño de características.
7. Entrenamiento y validación comparativa de un HistGradientBoostingRegressor y de una red neuronal.
8. Selección de modelos y despliegue en la aplicación Alabeam.

4.1 Generación masiva de modelos estructurales

Dado que no se disponía de datos experimentales o históricos reales, esta fase tuvo como objetivo la generación de datos sintéticos que representaran el comportamiento estructural de vigas bajo distintos escenarios de carga y geometría. Para ello se modelaron de forma teórica miles de vigas mediante simulaciones por elementos finitos, garantizando que los resultados (tensiones y desplazamientos) fueran físicamente consistentes y realistas, incluyendo tanto configuraciones con niveles de esfuerzo admisibles como otras próximas al fallo. Este enfoque permitió disponer de un conjunto de datos suficientemente amplio, diverso y fiable para el entrenamiento de los modelos de aprendizaje automático.

4.1.1. Definición del espacio de diseño

Se definió un espacio de diseño que combina la geometría de la sección, el material, la longitud, las condiciones de contorno y el esquema de cargas. Los parámetros se muestrearon de forma aleatoria controlada mediante un script en Python, generando un archivo `.csv` maestro con miles de combinaciones. La estrategia de muestreo se orientó a cubrir rangos realistas y equilibrar la frecuencia de los distintos tipos de sección y apoyo.

4.1.2. Construcción automática en HyperMesh

La creación de los modelos FEM se automatizó en HyperMesh mediante un script en TCL que lee el `.csv` maestro y construye para cada fila una viga 1D en el plano XY XY. A cada modelo se le asignan las propiedades de material y sección, las condiciones de contorno y las cargas correspondientes. Finalmente, cada caso se exporta en formato `.fem` listo para su resolución con Optistruct.

4.1.3. Ejecución FEM con OptiStruct

Los ficheros `.fem` se resolvieron con OptiStruct para obtener las respuestas estructurales. La salida binaria `.op2` incluye los desplazamientos, tensiones y reacciones nodales. El proceso se ejecutó por lotes con control de errores, permitiendo detectar y reintentar automáticamente los casos fallidos.

4.2 Extracción de resultados y construcción del dataset

A continuación se detalla el proceso de extracción de resultados para posteriormente generar el conjunto de datos para el entrenamiento.

4.2.1. Lectura de resultados con PyNastran

La lectura y el procesado de los `.op2` se realizó con PyNastran. Para cada modelo se extrajeron el desplazamiento máximo y la tensión equivalente de Von Mises máxima, junto con identificadores que permiten trazar cada muestra hasta su definición paramétrica original.

4.2.2. Ensamblado del conjunto de datos

Se construyó un dataset tabular que integra, para cada modelo, las entradas de diseño y las salidas objetivo. Se llevaron a cabo verificaciones de consistencia, eliminación de duplicados y controles de rango.

4.3 Preprocesado y diseño de características

Una vez obtenido el conjunto de datos, se realizan las siguientes tareas de preprocesado.

4.3.1. Limpieza y transformaciones iniciales

Antes del entrenamiento se normalizaron identificadores, se codificaron categorías y se homogenizaron unidades. Se verificó la ausencia de valores ausentes en las variables esenciales y se aplicaron conversiones simples cuando fue necesario.

4.3.2. Características físico informadas

Se incorporaron al dataset magnitudes derivadas que capturan aspectos clave del comportamiento de vigas. Entre ellas se incluyen propiedades de sección y rigidez, métricas de esbeltez, longitudes efectivas asociadas a las condiciones de contorno y escalas físicas de respuesta para desplazamiento y tensión. Estas características ayudan a estabilizar la relación entrada salida y a mejorar la capacidad predictiva de los modelos.

4.4 Entrenamiento de modelos

Finalizado el preprocesamiento del conjunto de datos se procede al entrenamiento de los modelos como se detalla a continuación.

4.4.1. Modelos considerados

Se entrenaron y compararon dos enfoques supervisados: un HistGradientBoostingRegressor y una red neuronal densa de regresión. En este capítulo se presentan únicamente el papel de cada modelo dentro del flujo y el protocolo de validación. Las configuraciones concretas y sus justificaciones se detallan en el Capítulo 5.

4.4.2. Esquema de validación y particiones

Con el objetivo de obtener una estimación robusta del rendimiento se empleó validación cruzada con agrupación por familia geométrica y tipo de apoyo. Este esquema evita fuga de información cuando existen muestras muy similares y permite evaluar el comportamiento del modelo frente a configuraciones no vistas durante el entrenamiento. En paralelo se reservó un conjunto de prueba estratificado para la evaluación final.

4.4.3. Métricas de evaluación

Se utilizaron métricas absolutas como MAE y RMSE, el coeficiente de determinación R^2 y errores relativos normalizados por escalas físicas de referencia. Este conjunto de métricas facilita interpretar la precisión en términos de la teoría de vigas y compara de forma homogénea ambos modelos.

4.5 Iteración y mejora del proceso

Crear un modelo de aprendizaje automático implica un proceso repetitivo de ajustes y mejoras, tanto en los parámetros de entrenamiento como en el tratamiento de los datos, para lograr un mejor desempeño. A continuación se explica el proceso que se ha seguido el trabajo.

4.5.1. Ciclo de refinamiento

El desarrollo siguió un ciclo iterativo. En una primera etapa se entrenaron modelos con un preprocesado mínimo y el rendimiento fue insuficiente. A partir de ese diagnóstico se enriquecieron las características físico informadas y se ajustaron los parámetros de entrenamiento. Tras varias iteraciones se alcanzó un equilibrio estable entre precisión y complejidad, momento en el que se fijaron los modelos definitivos.

4.5.2. Selección final

La selección se basó en el desempeño medio y en la estabilidad entre pliegues de validación. También se consideraron la rapidez de inferencia y la interpretabilidad práctica en el contexto de la aplicación final. El resultado es un par de modelos listos para uso: uno basado en árboles y otro neuronal, ambos con rendimiento contrastado y con configuraciones reproducibles.

4.6 Integración en la aplicación Alabeam

Una vez validados los modelos se integraron en una aplicación interactiva desarrollada en Streamlit. La interfaz permite definir rápidamente las propiedades geométricas, materiales, apoyos y cargas, y devuelve la predicción de desplazamiento máximo y tensión de Von Mises máxima. El usuario puede seleccionar qué modelo emplear en cada consulta, de modo que se cubren necesidades de exploración rápida y análisis más detallado.

4.7 Reproducibilidad y control de calidad

Cada muestra del dataset mantiene un vínculo directo con su definición paramétrica, el fichero `.fem` correspondiente y el resultado `.op2` del que se extrajo la información. Este encadenamiento permite auditar cualquier predicción y rehacer el flujo completo si es necesario.

Se fijaron semillas aleatorias en el muestreo y en el entrenamiento. Se registraron versiones de librerías y de scripts, y se documentó el entorno de ejecución. Este control facilita la comparación entre iteraciones y la replicación de resultados.

Antes del entrenamiento se aplicaron comprobaciones automáticas de rangos y coherencia dimensional. Durante la validación se monitorizaron curvas de aprendizaje y mapas de errores por familia de secciones y apoyos. Tras el despliegue se realizaron pruebas funcionales en la interfaz para verificar la consistencia de las predicciones en distintos escenarios de entrada.

4.8 Resumen de la metodología

La metodología propuesta automatiza la generación de miles de modelos de vigas, obtiene respuestas FEM de referencia y construye un conjunto de datos coherente para el entrenamiento de dos enfoques complementarios. La inclusión de características físico informadas y un protocolo de validación con agrupación permitió mejorar de forma significativa el rendimiento frente a los primeros intentos con preprocesado mínimo. Finalmente se integraron ambos modelos en una herramienta interactiva que habilita predicciones ágiles y replicables en el ámbito del análisis estructural.

La Figura 4.1 sintetiza el flujo de trabajo establecido por la metodología comentada anteriormente.

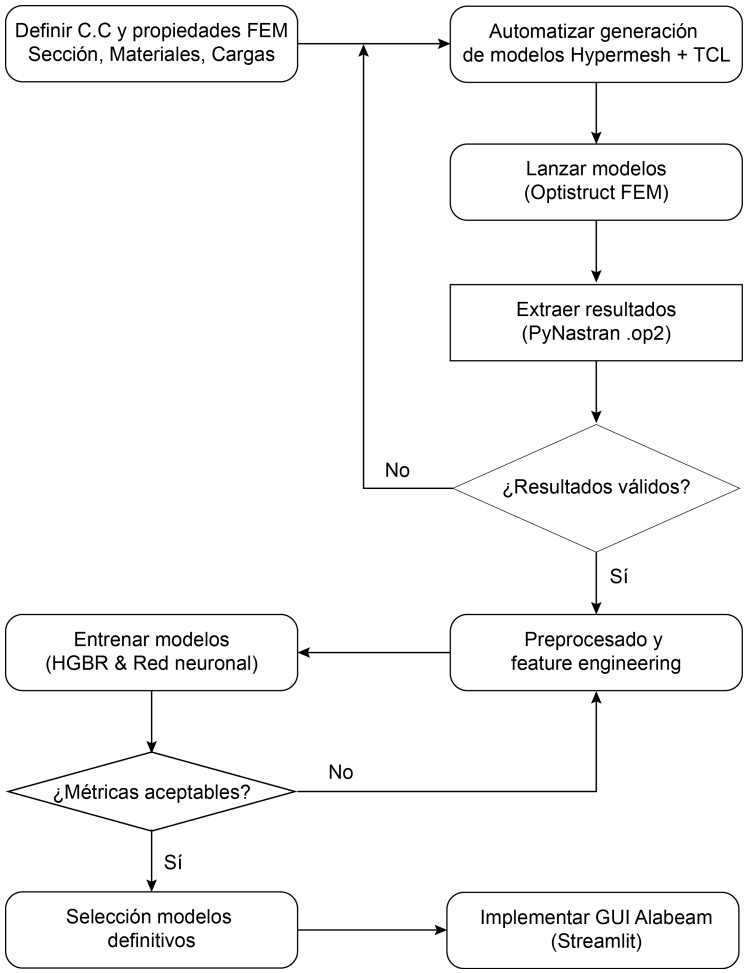


Figura 4.1. Metodología del proyecto

4.9 Recursos y entorno computacional

Se empleó un ordenador portátil con Intel Core i9, 32 GB de RAM y SSD de 1 TB. El flujo de trabajo se apoyó en Altair HyperMesh (licencia de estudiante), Altair OptiStruct, y un entorno Python con librerías de código abierto (scikit-learn, TensorFlow, PyNastran, Streamlit).

El cómputo total aproximado fue de 24 h, distribuidas en: generación de modelos FEM, resolución en OptiStruct, postprocesado con PyNastran y entrenamiento/validación de los modelos.

4.10 Costes estimados

El coste y presupuesto estimado del proyecto se detalla en el Apéndice A.

Capítulo 5. DESARROLLO DEL PROYECTO

En este capítulo se detalla todo el proceso que se ha llevado a cabo para el entrenamiento de los modelos de aprendizaje automático y el posterior desarrollo de la herramienta Alabeam. Todos los análisis descritos en este capítulo se ejecutaron con el entorno indicado en la Sección 5.2 y se mantuvo el sistema de referencia de la Sección 2.1.1. El código desarrollado para la generación de modelos y el entrenamiento de los modelos predictivos comentado en esta sección se encuentra disponible en un repositorio público en *Github*, ver referencia [27].

5.1 Arquitectura general del sistema

El flujo de trabajo seguido en este proyecto se ha diseñado de forma modular, de manera que cada componente tecnológico de la Sección 2.2 cumple una función específica dentro del proceso completo, garantizando la trazabilidad y la automatización del conjunto. La Figura 5.1 ilustra de forma esquemática la estructura general del sistema y la interacción entre sus principales bloques.

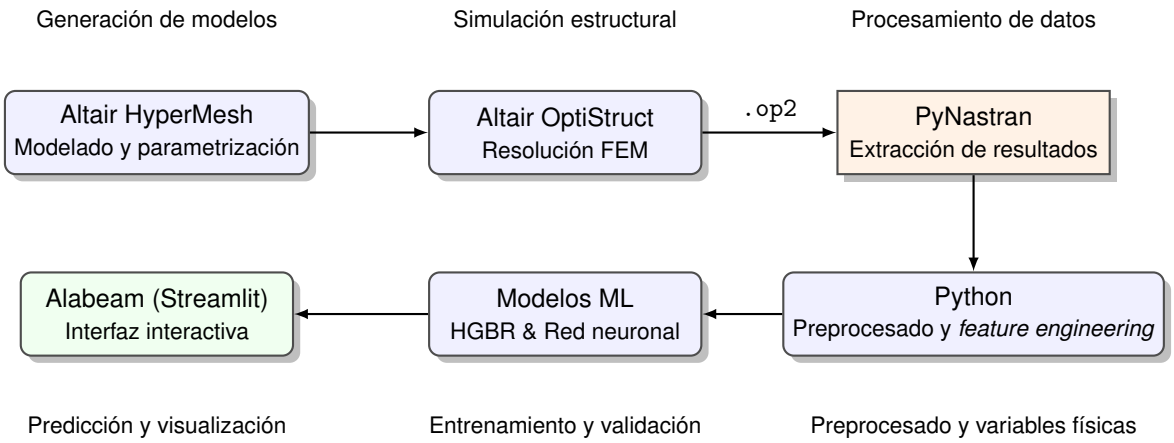


Figura 5.1. Arquitectura general del sistema desarrollado para la aplicación Alabeam

En primer lugar, la generación de los modelos estructurales se realiza en Altair HyperMesh, donde mediante *scripts* en lenguaje TCL se automatiza la creación de vigas con diferentes combinaciones de geometría, material, apoyos y cargas. Estos modelos se exportan posteriormente al *solver* Altair OptiStruct, que se encarga de resolverlos utilizando el Método de los Elementos Finitos (MEF).

Los resultados obtenidos principalmente desplazamientos, tensiones y reacciones nodales se almacenan en archivos binarios (`.op2`), los cuales son procesados automáticamente mediante la librería PyNastran. Este paso permite extraer las magnitudes relevantes de cada simulación y convertirlas en un formato estructurado adecuado para su análisis posterior en Python.

Una vez recopilados los resultados, se lleva a cabo el preprocesamiento y la generación de nuevas variables derivadas mediante scripts en Python.

En esta etapa se calculan propiedades geométricas (área, momentos de inercia, módulo resistente, etc.), parámetros de rigidez (EI) y factores adimensionales asociados a la esbeltez o la longitud efectiva. Este proceso constituye la base del *feature engineering*, que permite optimizar la capacidad predictiva de los modelos de aprendizaje automático.

Posteriormente, los datos preprocesados se utilizan para el entrenamiento y validación de los modelos desarrollados en scikit-learn, TensorFlow y Keras, concretamente un *HistGradientBoostingRegressor* y una red neuronal. Los modelos entrenados se integran finalmente en la aplicación Streamlit denominada Alabeam, que ofrece una interfaz interactiva en la que el usuario puede introducir las características de una viga y obtener instantáneamente las predicciones de tensión y desplazamiento máximos.

En resumen, la arquitectura propuesta conecta de forma secuencial las etapas de modelado, resolución, extracción de datos, aprendizaje y visualización, permitiendo un flujo de trabajo completamente automatizado y reproducible. La metodología de implementación y los detalles técnicos seguirá el flujo explicado en la Sección 4.

5.2 Entorno y versionado

A efectos de reproducibilidad, la Tabla 5.1 recoge las versiones del software y librerías empleadas.

Tabla 5.1. Versiones de software y librerías empleadas

Software/herramienta	Versión y entorno
Altair HyperMesh	2025.1 (Windows 11)
Altair OptiStruct	2025.1
PyNastran	1.4.1 (Python 3.12.3, WSL Ubuntu 22.04)
scikit-learn	1.7.1
TensorFlow/Keras	2.20
Streamlit	1.49.1
Compilador LaTeX	TeX Live 2025 (pdfLaTeX)

5.3 Alcance y tipo de análisis

El trabajo se centra en vigas 2D en el plano XY. Esta elección responde a tres motivos principales:

- Permite acotar el problema a un dominio físico bien conocido y con formulación sólida.
- Facilita generar un volumen grande de simulaciones FEM con trazabilidad y control de parámetros.
- Ofrece un banco de pruebas idóneo para demostrar que modelos de aprendizaje automático pueden aproximar con buena precisión las respuestas de un *solver* de elementos finitos.

Las vigas se emplean de forma generalizada en numerosos sectores industriales y constituyen elementos fundamentales en prácticamente cualquier tipo de construcción, permitiendo representar y analizar una amplia variedad de escenarios estructurales. La Figura 5.2 muestra una estructura reticular formada por vigas.

Figura 5.2. Ejemplo de vigas en construcción [28]

Las vigas permiten aislar de forma clara la flexión en el plano y el esfuerzo axial, generar un gran número de simulaciones MEF y, sobre todo, disponer de magnitudes físico informadas (como EI, W_z, L_{ef} o escalas del tipo L_{ef}^3/EI) que se incorporan de manera directa al *feature engineering* (ver Sección 2.1.2).

En este marco, se adopta un sistema de coordenadas en el que el eje X coincide con el eje longitudinal de la viga, el eje Y es transversal dentro del plano y el eje Z emerge del plano. Las cargas consideradas son fuerzas puntuales en X e Y y momentos alrededor de Z (ver Sección 2.1.1); las condiciones de contorno incluyen configuraciones típicas como biapoyada, empotrada, empotrada–articulada y voladizo, modelando la longitud efectiva mediante un factor K acorde a cada caso. Se supone un comportamiento lineal elástico e isótropo, en régimen estático y de pequeñas deformaciones, y se discretiza con elementos 1D de viga con grados de libertad (u, v, θ_z) por nodo, lo que permite mallas muy eficientes manteniendo la fidelidad en las respuestas de interés.

El propósito es demostrar que modelos de aprendizaje automático pueden aproximar con buena precisión salidas de referencia obtenidas con un *solver* de elementos finitos en un dominio físico acotado y bien interpretado. Concretamente, a partir de combinaciones de entradas geométricas, de material, de contorno y de carga, se buscan predicciones del desplazamiento máximo $w_{\text{máx}}$ en la dirección Y y de la tensión de Von Mises máxima $\sigma_{\text{máx}}$.

El objetivo no es sustituir el análisis MEF general, sino evidenciar su aceleración cuando el problema está bien parametrizado y el modelo se entrena dentro de un espacio de diseño adecuadamente muestreado. Quedan fuera de alcance fenómenos como la dinámica transitoria, la no linealidad geométrica o material, el pandeo no lineal global, el contacto y los modelos 3D de sólido, que se consideran líneas futuras razonables una vez consolidada la metodología en vigas 2D.

5.4 Generación de modelos de vigas

El conjunto de casos MEF se genera restringiendo el problema al plano XY y limitando las condiciones de contorno a los extremos de la viga. Esta decisión reduce de forma drástica el número de combinaciones necesarias manteniendo la riqueza física suficiente para entrenar y validar modelos predictivos. Todas las longitudes, cargas y propiedades se expresan siguiendo un sistema de unidades coherente definido en la Sección 2.1.1.

Dominio geométrico La longitud libre L de la viga se define en el intervalo $[150,\ 5000]$ mm, albergando así los casos más comunes de longitud de vigas. Se consideran cinco familias de secciones transversales: circular maciza, circular hueca (tubo), rectangular maciza, rectangular hueca y perfil en I. Cada sección se define por sus dimensiones mínimas necesarias (radio/diámetros en circulares; ancho, canto y espesores en rectangulares e I), garantizando geometrías físicamente realizables (espesores positivos y relaciones internas $<$ externas).

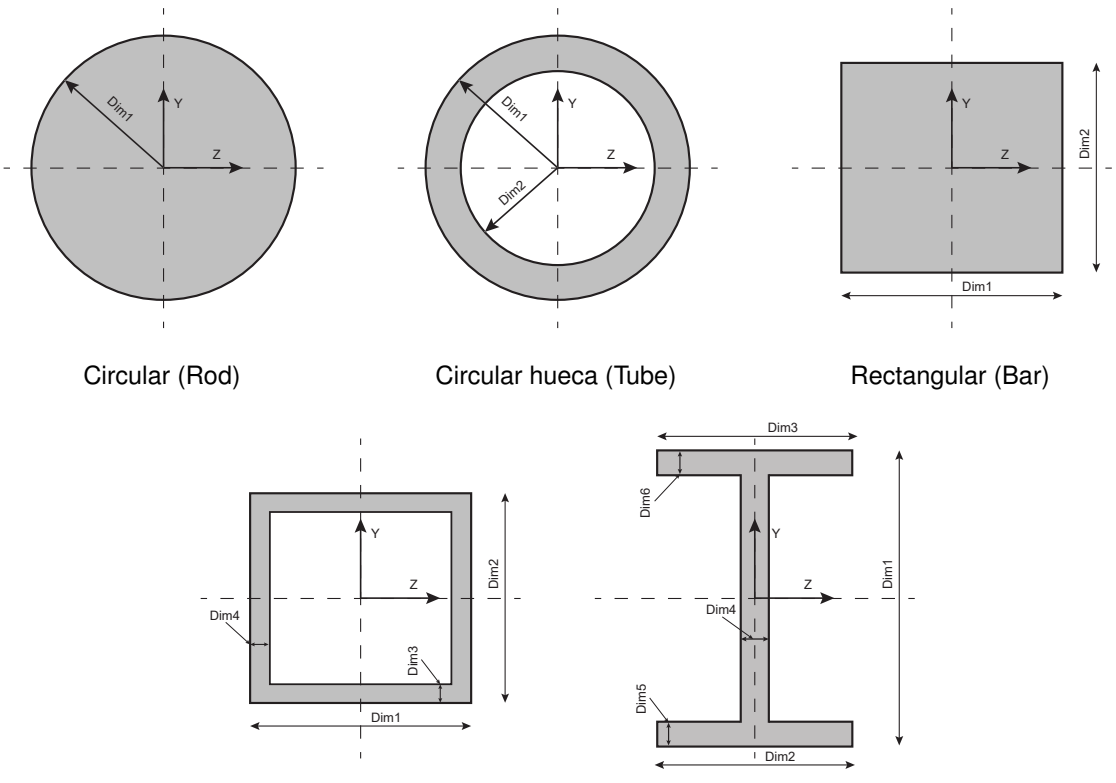


Figura 5.3. Familias de secciones transversales consideradas

En la Fig. 5.3 se ilustran las cinco familias de secciones utilizadas junto con sus dimensiones geométricas básicas. Para mantener la coherencia con el entorno de modelado y los *scripts* de generación, dichas dimensiones se han nombrado siguiendo la convención de HyperMesh como Dim1–Dim6. La Tabla 5.2 recoge la correspondencia entre cada Dim y la magnitud geométrica específica en cada tipo de sección, de modo que la misma notación se utiliza de forma consistente en la generación de modelos, el preprocesado y el postprocesado.

Tabla 5.2. Equivalencias Dim1–Dim6 (formato HyperMesh).

Sección	D1	D2	D3	D4	D5	D6
Circular	Radio	—	—	—	—	—
Circular hueca	Radio ext.	Radio int.	—	—	—	—
Rectangular	Ancho	Alto	—	—	—	—
Rectangular hueca	Ancho	Alto	Esp. vertical	Esp. horizontal	—	—
I	Alto	Ancho ala inf.	Ancho ala sup.	Esp. alma	Esp. ala inf.	Esp. ala sup.

Materiales Se emplean materiales metálicos habituales en estructuras: acero al carbono, acero inoxidable, aluminio y titanio, modelados como lineales elásticos e isótropos. La Tabla 5.3 resume las propiedades utilizadas en el muestreo (módulo de elasticidad E, coeficiente de Poisson ν, densidad ρ en kg mm^{-3} y límite elástico σ_y en MPa).

Tabla 5.3. Propiedades mecánicas de materiales empleados

Material	E [MPa]	ν [-]	ρ [kg/mm^3]	σ_y [MPa]
Acero	210 000	0.30	7.8×10^{-6}	370
Acero inoxidable	210 000	0.30	7.9×10^{-6}	170
Aluminio	70 000	0.32	2.7×10^{-6}	270
Titanio	120 000	0.32	4.5×10^{-6}	830

Condiciones de contorno Se consideraron dos tipos de restricción: empotramiento, que bloquea todos los grados de libertad del nodo (u, v, θ_z), y apoyo simple (articulación), que impide el desplazamiento en el punto de apoyo pero permite el giro. Para simplificar y evitar mecanismos, las restricciones sólo se aplican en los extremos de la viga y se descartan combinaciones del tipo "simple–libre". Con ello, las configuraciones válidas son: *voladizo* (empotramiento–libre), *biapoyada* (simple–simple) y *mixta* (empotramiento–simple o simple–empotramiento), tratadas como equivalentes a biapoyada a efectos de longitud efectiva. Estas configuraciones se utilizan también para fijar el factor K de longitud efectiva cuando procede (véase Tabla 2.2).

Cargas y casos de carga Cada modelo admite hasta tres acciones puntuales en total, combinando: fuerzas en X (axiales: tracción/compresión), fuerzas en Y (transversales: flexión en el plano) y momentos alrededor de Z (pares flectores en el plano). Las fuerzas puntuales y momentos se aplican en posiciones internas $x \in (0, L)$. Se adopta un convenio de signos coherente con el sistema de ejes descrito en la Sección 2.1.1.

Para garantizar casos físicamente coherentes y útiles para el entrenamiento, la generación aplica controles en tres niveles.

- Validez geométrica: se imponen espesores positivos en secciones huecas (con $D_i < D_o$), altura de alma positiva en perfiles en I $(h_w > 0)$ y no inversión ala/alma; además, se filtran combinaciones con momentos de inercia o módulos resistentes no válidos $(I \leq 0,\ S \leq 0)$ y se acotan esbelteces extremas.
- Condiciones de contorno: sólo se permiten restricciones en los extremos y se descartan configuraciones con mecanismos (p. ej., simple–libre), asegurando que no queden grados de libertad sin restringir.
- Síntesis de cargas: para cada viga se genera una carga dominante que sitúa la tensión máxima objetivo en el entorno del límite elástico, muestreando k con sesgo hacia el rango subcrítico $[0.8, 1.0]$ y una fracción en $[1.0, 1.2]$, de modo que $\sigma_{\text{target}} = k\,\sigma_y$ y $M_{\text{target}} = \sigma_{\text{target}}\,S$. Con esto se logra asegurar que hay casos con tensiones cercanas al límite elástico e incluso algunos que lo superan.

A partir de M_{target}, se calcula una fuerza puntual o un momento equivalente (según el tipo de apoyo y la posición), y se evalúa una estimación conservadora de la flecha δ. La magnitud de la carga se ajusta iterativamente hasta cumplir una ventana de servicio $\delta \in [0.01\,L,\ 0.10\,L]$ (reducciones $\times 0.85$ o incrementos $\times 1.15$ con intentos acotados), si no es factible, el caso se descarta. Para evitar casos demasiado "limpios" o triviales, se añaden hasta dos cargas secundarias de pequeña entidad (2–12 % de la dominante) con posiciones internas muestreadas en $[0.1\,L,\ 0.9\,L]$ y ligeras perturbaciones en cuartos/tercios, limitando el total a tres cargas. Se evita aplicar cargas a menos de 10 mm de los extremos. Los modelos no convergentes o con respuestas anómalas se marcan para reintento y, en su caso, se descartan antes del preprocesado. La trazabilidad se garantiza mediante nombres de caso y un CSV maestro que codifica entradas, restricciones y cargas generadas.

5.4.1. Modelado en HyperMesh/OptiStruct

Las vigas se han modelado en HyperMesh empleando elementos 1D CBEAM con propiedad PBEAML, que permite definir la geometría de la sección (circular, tubo, rectangular, cajón o perfil en I) de forma paramétrica y coherente con la convención Dim1–Dim6 descrita previamente. Cada CBEAM dispone de seis GDL clásicos por nodo $(u_x, u_y, u_z, \theta_x, \theta_y, \theta_z)$; en el planteamiento 2D el eje del elemento coincide con X y la flexión se produce en el plano XY (momento alrededor de Z). La orientación local del elemento se fija mediante el vector de offset de dirección (tercer y cuarto campo del CBEAM), asegurando que el eje fuerte quede alineado con Y.

Las condiciones de contorno se imponen con tarjetas SPC (*Single Point Constraint*). El apoyo simple se representa restringiendo las tres traslaciones (123) y dejando libres las rotaciones, mientras que el empotramiento fija los seis GDL (123456). Las cargas se aplican con FORCE (componentes en X o Y) y MOMENT (alrededor de Z), ubicadas en nodos interiores o extremos según el caso.

En *case control* se activan los resultados necesarios en formato OP2: DISPLACEMENT, SPCFORCE y STRESS, que posteriormente se extraen con PyNastran para obtener el desplazamiento máximo ($w_{\text{máx}}$) y la tensión máxima ($\sigma_{\text{VM,máx}}$).

El OP2 es un fichero binario de resultados (formato Nastran/OptiStruct) que almacena, por cada caso de carga (*subcase*) y paso de carga/tiempo, tablas de magnitudes postproceso en registros no formateados (Fortran). Es compacto y rápido de leer, pero no es legible a simple vista: requiere un lector específico (p. ej., PyNastran). Entre las tablas habituales están los desplazamientos nodales (OUG*), tensiones/esfuerzos en elementos (OES*), fuerzas de elementos (OEF*) y fuerzas de reacción (OQG*/SPCFORCE). Cada registro referencia IDs de nodos/elementos y puede venir en sistemas de coordenadas globales o locales, por lo que es importante mantener la consistencia de ejes. En el *case control* se solicitaron DISPLACEMENT, STRESS y SPCFORCE en OP2; posteriormente, PyNastran extrae de las tablas correspondientes $w_{\text{máx}}$ y $\sigma_{\text{VM,máx}}$, junto con verificaciones de integridad (subcases, unidades y mapeo de IDs).

La malla de elementos se genera dividiendo la longitud L en elementos uniformes (típicamente del orden de decenas de milímetros) para capturar con estabilidad la respuesta de flexión sin coste excesivo. Las propiedades de sección se asignan con PBEAML seleccionando el tipo según la denominación en HyperMesh (ROD, TUBE, BAR, BOX, I) y los parámetros geométricos correspondientes, el material se define con MAT1 (módulo E, ν y densidad), siguiendo el sistema de unidades $\{\text{mm}, \text{N}, \text{MPa}\}$.

Ejemplo práctico A modo de ejemplo, se toma una viga tubular de $L = 1000$ mm en titanio con apoyo simple en ambos extremos mallada con 100 elementos CBEAM uniformes entre los nodos extremos (representada en la Figura 5.4). La sección se definió con PBEAML, TUBE indicando diámetro exterior e interior; las restricciones se aplicaron con SPC tipo 123 en los extremos (permitiendo giro), y se prescribió una acción mediante MOMENT alrededor de Z en un nodo interior. En *case control* se activaron DISPLACEMENT(OUTPUT2), SPCFORCE(OUTPUT2) y STRESS(OP2,ALL) para volcar desplazamientos y tensiones al fichero binario .op2. Este patrón es representativo del resto de casos generados de forma automática.

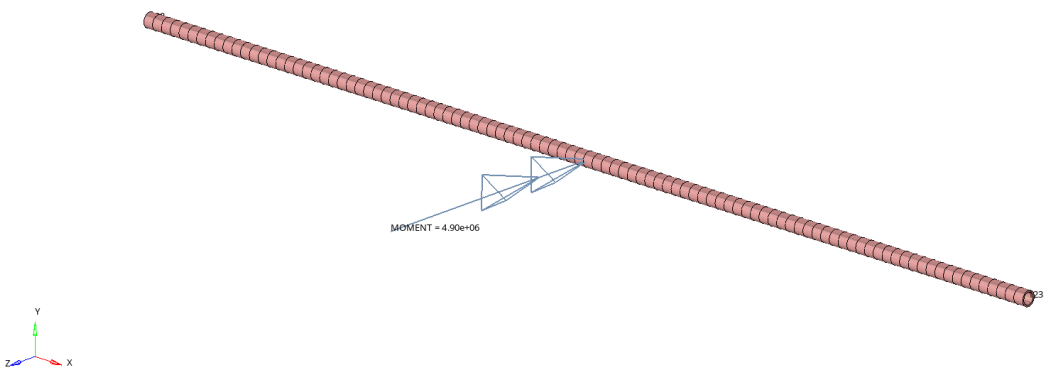


Figura 5.4. Viga de titanio modelada en HyperMesh

5.4.2. Generación automática del fichero CSV de casos

La automatización comienza con un script en Python que construye un fichero CSV maestro con todos los casos a simular. Este fichero será leído por el script TCL de HyperMesh para generar los modelos .MEF de manera desatendida. El CSV define, por fila, la longitud de la viga, el tipo de sección y sus dimensiones según la convención Dim1–Dim6, el material con sus propiedades básicas, la configuración de apoyos en los extremos y hasta tres acciones puntuales o pares en posiciones internas. Se añade un esquema de nombres de caso que permite trazar cada modelo desde la generación hasta el postproceso, por ejemplo, *Beam_TUBE_4900mm_simple_simple_Titanium* (sección tubular, $L = 4900$ mm, biapoyada y material titanio).

Como se mencionó previamente, para cada combinación válida, el script sintetiza un caso de carga dominante con el objetivo de situar la tensión máxima cerca del límite elástico sin exceder ventanas de servicio en flecha. Para ello se selecciona un factor k sesgado hacia el intervalo subcrítico 0.8–1.0 y con una fracción en 1.0–1.2, se fija un objetivo de tensión $\sigma_{\text{target}} = k\,\sigma_y$ y se convierte en un objetivo de momento $M_{\text{target}} = \sigma_{\text{target}}\,S$ mediante el módulo resistente S de la sección. En función de la configuración de apoyos y de la posición interna elegida, M_{target} se traduce a una fuerza o a un momento aplicado. Con una estimación conservadora de flecha, el script ajusta iterativamente la magnitud para que el desplazamiento máximo quede dentro de un intervalo relativo respecto a la longitud, típicamente entre $0.01\,L$ y $0.10\,L$. Si no es posible cumplir simultáneamente los criterios de tensión y flecha, el caso se descarta.

El procedimiento registra en fromato CSV (*Comma Separated Values*), además del nombre del modelo y los parámetros geométricos y de material, el tipo y dirección de cada carga, su magnitud en N o N·mm y su posición en mm. El ejemplo mostrado en la Tabla 5.4 ilustra una fila de salida con una viga circular maciza de 150 mm en acero, en voladizo, con un momento alrededor de Z aplicado en una posición interna:

Tabla 5.4. Ejemplo de una fila del CSV mostrado en formato vertical para facilitar su lectura

Campo	Valor
model_name	Beam_ROD_150mm_clamped_None_Steel
L [mm]	150
section_type	ROD
dim1, dim2, dim3	40.89, 0, 0
dim4, dim5, dim6	0, 0, 0
material	Steel
E [MPa]	210000
nu [-]	0.3
density [N/mm^3]	7.8e-06
yield_strength [MPa]	370
support_L, support_R	clamped, None
num_cargas	1
c1_type	moment
c1_dir	Z
c1_mag [N·mm]	-5429301.16
c1_pos [mm]	75.0
c2_*	—
c3_*	—

El código completo del generador de CSV se incluye en los apéndices y se referencia en el texto principal para facilitar la reproducción. Se recomienda fijar una semilla aleatoria cuando se requiera repetibilidad exacta del conjunto de casos.

5.4.3. Automatización en HyperMesh con TCL

El proceso de creación masiva de modelos se ejecuta con un script TCL que lee el CSV maestro y construye, para cada fila, un modelo .hm y su correspondiente .MEF listo para OptiStruct. El script trabaja en modo desatendido: borra el modelo activo, crea materiales, propiedades y secciones, mallado 1D, condiciones de contorno, cargas, caso de carga y opciones de salida, y finalmente exporta el input deck del *solver* Optistruct.

El flujo interno es el siguiente. Primero abre el CSV en UTF-8 y salta el encabezado. De cada línea extrae nombre del modelo, longitud, tipo de sección y dimensiones, material y propiedades básicas, configuración de apoyos y hasta tres acciones. Con esos datos crea el material MAT1 y una propiedad PBEAML asociada. La sección se define con una entidad *beamsection* y se parametriza según el tipo: ROD usa una dimensión, TUBE dos, BAR dos, BOX cuatro, e I hasta seis. Para TUBE, BOX e I se incluyen validaciones básicas de geometría para evitar combinaciones no físicas.

La geometría se genera como una línea entre el origen y la longitud L. El mallado lineal se hace con un tamaño L/100, lo que produce del orden de un centenar de elementos por viga y asegura una resolución suficiente en flexión sin coste excesivo.

La orientación local del elemento se fija con un vector de referencia para mantener el eje fuerte en el plano XY. En perfiles I se usa una orientación alternativa para alinear alma y alas de forma coherente.

Las condiciones de contorno se agrupan en un *load collector* SPC. Para un apoyo simple se restringen las tres traslaciones en el nodo extremo y se dejan libres las rotaciones, para un empotramiento se fijan todos los grados de libertad. La identificación de los nodos extremos se realiza mediante cajas de selección alrededor de x = 0 y x = L con una tolerancia delta proporcional a L. Las cargas externas se incorporan desde el CSV recorriendo hasta tres posibles registros. Cada una define tipo, dirección, magnitud y posición a lo largo de la viga. El script localiza el nodo más cercano a la posición indicada y aplica, en función del tipo, una fuerza en X o Y o un par alrededor de Z.

A continuación se crea un *loadstep* estático con los *load collectors* de SPC y cargas activos. En *case control* se habilitan desplazamientos, fuerzas de restricción y tensiones para salida binaria OP2, que más adelante se leerá con PyNastran. El modelo se guarda como .hm y se exporta a .fem usando la plantilla de OptiStruct configurada en la ruta *optistruct template*. Si la plantilla no existe, se informa por consola.

El script usa codificación UTF-8 para leer el CSV y separa por comas, si los nombres de modelo o materiales contienen comas, conviene entrecomillar esos campos en el CSV. La tolerancia delta para localizar nodos puede ajustarse en función de la densidad de malla. El tamaño de elemento L/100 es un compromiso entre coste y precisión, para cargas muy localizadas se puede reducir. La gestión de combinaciones de apoyos excluye mecanismos y normaliza la etiqueta *None* en el extremo libre. Las magnitudes de las cargas se asumen en N para fuerzas y N·mm para momentos, coherentes con el sistema de unidades (Sección 2.1.1.

El *script* completo, con comentarios y rutas parametrizables de entrada y salida, se incluye en el apéndice y puede referenciarse como Apéndice B.1. Para reproducibilidad, se recomienda fijar una convención estable de nombres de modelo y conservar junto a cada .MEF su .hm y la fila original del CSV.

5.4.4. Extracción de resultados OP2 con PyNastran y construcción del dataset

Tras la ejecución de todos los modelos en el *solver* OptiStruct, los resultados se almacenan en ficheros OP2 binarios. Para consolidar la información en un único *dataset* tabular se emplea un script en Python basado en PyNastran que recorre la carpeta de resultados, lee cada OP2 y extrae, por modelo, el desplazamiento nodal máximo y una medida de tensión máxima representativa (Von Mises). Finalmente, estos valores se unen al CSV de entrada que contenía las características geométricas, de material, apoyos y cargas, de forma que se obtiene un único fichero con entradas y objetivos de aprendizaje.

PyNastran permite acceder a tablas de desplazamientos nodales (familia OUG), tensiones en elementos de barra y viga (familia OES) y fuerzas en restricciones (SPCFORCE). En cada OP2 se carga el primer subcase estático y se calcula el desplazamiento máximo como la norma euclídea de los vectores de desplazamiento nodal.

Para la tensión se emplea la salida de tensiones de elementos CBEAM que proporciona tensiones principales por fibra. Se toma como *proxy* de tensión máxima la mayor entre las dos tensiones principales en cada elemento y fibra y luego el máximo global del modelo. Esto es consistente con la identificación de la zona más crítica bajo flexión y cubre tanto tracción como compresión, ya que se usa el valor absoluto.

Consistencia de unidades y ejes. Los desplazamientos salen en milímetros y las tensiones en MPa, coherentes con el sistema $\{\text{mm}, \text{N}, \text{MPa}\}$ definido en el proyecto. Las tablas pueden venir en ejes globales o locales según configuración del caso de carga. Con los elementos 1D CBEAM y el *case control* empleado, las magnitudes recuperadas se interpretan en el sistema global, por lo que no se requieren transformaciones adicionales.

Combinado con el CSV de entrada. Para construir el *dataset* final se crea una tabla intermedia con las columnas `model_name`, `max_displacement` (desplazamiento máximo) y `max_stress` (tensión máxima) extraídas de cada OP2 y se realiza una unión interna con el CSV original de generación por la clave `model_name`. El resultado contiene, por fila, todas las características de entrada más los objetivos para aprendizaje supervisado.

El *script* contempla que alguna tabla no esté presente y devuelve nulos en ese caso. Si un OP2 no puede leerse o el modelo no converge, se omite su fila. Es recomendable registrar el número de OP2 procesados, los que aportan desplazamientos, los que aportan tensiones y los no procesados, para asegurar trazabilidad. En total se generaron 4075 modelos y fueron descartados 17, quedando finalmente un *dataset* con 4058 modelos para entrenar.

Las columnas añadidas al *dataset* se muestran en la Tabla 5.5, incluyendo las variables objetivo de desplazamiento y tensión como columnas nuevas, además de las ya presentes en el CSV de entrada:

Tabla 5.5. Salidas extraídas por PyNastran e incorporadas al dataset final.

Columna	Descripción
`max_displacement`	Desplazamiento nodal máximo del modelo en mm
`max_stress`	Tensión máxima tomada como envolvente de principales en elementos CBEAM (Von Mises), en MPa

El código completo de extracción y combinación con Pandas se incluye en el apéndice y puede referenciarse como Apéndice B.1.

5.5 Preprocesado del conjunto de datos

El conjunto de datos en bruto reúne, para cada viga, su geometría, material, configuración de apoyos, hasta tres cargas puntuales o momentos y los resultados MEF relevantes. El propósito del preprocesado es transformar esa información en un conjunto de características numéricas estables, con sentido físico y adecuadas para el aprendizaje supervisado, de modo que los modelos reduzcan varianza y mejoren su capacidad de generalización.

5.5.1. Carga y utilidades numéricas

Se lee el CSV original y se define una división segura `safe_div`(a, b, ε) con un épsilon pequeño para evitar divisiones por cero o por valores muy próximos a cero. Esta precaución es necesaria cuando se normaliza por áreas pequeñas, módulos resistentes o rigideces.

5.5.2. Propiedades de sección por tipología

Para cada tipo de sección se calculan de forma cerrada el área A, el momento de inercia respecto al eje de flexión I_z, el módulo resistente $W_z = I_z/c$ y una aproximación de la inercia torsional J. Se contemplan las cinco familias de secciones empleadas en el trabajo: rectangular maciza, rectangular hueca, circular maciza, tubular circular y perfil en I. En el perfil en I se compone I_z con el teorema de ejes paralelos sumando alma y alas. En las secciones huecas se verifica que los espesores efectivos sean positivos.

5.5.3. Rigidez y esbeltez

Se generan magnitudes clásicas de vigas que guían el aprendizaje: la rigidez a flexión $EI = E\,I_z$, el radio de giro $r_z = \sqrt{I_z/A}$ y razones como L/r_z o L^3/EI. Estas variables aparecen en fórmulas de flecha y tensiones y aportan una base física a las relaciones que el modelo debe capturar.

5.5.4. Longitud efectiva y factor K

Se calcula la longitud efectiva $L_{\text{eff}} = K\,L$, donde K depende de los apoyos. Se utilizan los valores habituales: $K = 2{,}0$ para voladizo, $K = 1{,}0$ para biapoyada, $K = 0{,}5$ para doble empotramiento y $K = 0{,}7$ para empotrada–articulada. Además se añaden potencias e interacciones de L_{eff} que son especialmente informativas para el desplazamiento.

5.5.5. Agregación de cargas y magnitudes equivalentes

Las cargas aplicadas en las vigas se condensan en agregados con significado físico:

- Resultantes $\mathrm{FX}_{\text{total}}$ y $\mathrm{FY}_{\text{total}}$.
- Momento acumulado respecto al extremo izquierdo $\mathrm{MZ}_{\text{total,left}}$.
- Un sustituto del momento flector máximo $M_{\text{max,sss}}$ basado en expresiones de viga simplemente apoyada. Para una fuerza puntual P aplicada a distancia a, se aproxima $M_{\text{máx}} \approx P\,a(L-a)/L$. Los momentos aplicados contribuyen con su magnitud.

Se normalizan además las posiciones de aplicación de las cargas como fracción de la longitud, x/L, para disminuir sensibilidad a la escala absoluta.

5.5.6. Escalas físicas para tensiones y flechas

Se construyen dos escalas guía que anclan el orden de magnitud:

$$\sigma_{\text{scale}} = \frac{M_{\text{max,sss}}}{W_z}, \qquad \delta_{\text{scale}} = \frac{\text{FY}_{\text{total}}\, L^3}{EI}.$$

Para reflejar el efecto de los apoyos se incluyen versiones con longitud efectiva, por ejemplo $\delta_{\text{scale,eff}} = \text{FY}_{\text{total}}\, L_{\text{eff}}^3/EI$, así como intensidades de carga $\text{FY}_{\text{total}}/L$ y $\text{MZ}_{\text{total,left}}/L$.

5.5.7. Interacciones y transformaciones estabilizadoras

Se añaden interacciones entre escalas relevantes y combinaciones polinómicas centradas en el desplazamiento, como L_{eff}^3/EI, L_{eff}^4/EI y productos con L_{eff}. Para mitigar heterogeneidad y atípicos moderados se aplican transformaciones logarítmicas estables $\log(1+x)$ sobre magnitudes positivas o en valor absoluto, por ejemplo $\log EI$, $\log L_{\text{eff}}$, $\log \delta_{\text{scale}}$ y $\log \sigma_{\text{scale}}$.

5.5.8. Control de calidad y salida

Durante el cálculo se incluyen comprobaciones geométricas y numéricas: se asegura $D_{\text{i}} < D_{\text{o}}$ en tubos, se aplican épsilon en denominadores y se recortan combinaciones incoherentes. El resultado es un CSV con un subconjunto ordenado de columnas que se usa directamente en el entrenamiento.

5.5.9. Impacto del preprocesado en las métricas

Los análisis iniciales con variables crudas arrojaron errores elevados por mezcla de escalas, ausencia de rasgos físico informados y sensibilidad a posiciones absolutas. Tras introducir EI, L_{eff}, razones como L_{eff}^3/EI y escalas σ_{scale} y δ_{scale}, el aprendizaje se alinea con la teoría de vigas. En la práctica, los modelos ajustan correcciones alrededor de relaciones ya aproximadas y se observa una reducción del error y una mejora de la generalización frente a combinaciones no vistas exactamente en el entrenamiento.

5.6 Entrenamiento con HistGradientBoostingRegressor

Este apartado describe el procedimiento seguido para entrenar modelos de regresión basados en HistGradientBoostingRegressor (HGBR) para dos objetivos independientes: desplazamiento máximo y tensión de Von Mises máxima. El flujo implementa carga de datos, selección de rasgos numéricos, particionado entrenamiento prueba, ajuste del modelo, evaluación con métricas estándar y persistencia del modelo entrenado para su uso posterior en la aplicación.

El uso de HGBR es adecuado en datos tabulares con relaciones no lineales moderadas e interacciones entre variables. La discretización en histogramas acelera el ajuste y aporta regularización adicional, lo que ayuda a estabilizar el aprendizaje con rasgos físico informados. En este proyecto se observó una mejora sustancial de las métricas frente a modelos entrenados sobre variables crudas, en línea con lo esperado cuando se incorporan escalas y razones derivadas de la teoría de vigas.

5.6.1. Datos de entrada y selección de rasgos

Se parte del CSV preprocesado y se filtran las columnas candidatas definidas en la configuración del proyecto, restringiéndolas a variables numéricas disponibles. Las matrices de entrada X se completan con ceros en valores ausentes, y se crean dos vectores objetivo y: uno para `max_displacement` (desplazamiento máximo) y otro para `max_stress` (tensión máxima). El conjunto de rasgos finales coincide con la lista de características físico informadas resultante del preprocesado. Las variables objetivo se mantuvieron sin transformar con el fin de evitar posibles problemas en la fase de predicción y facilitar la interpretación de los resultados.

5.6.2. Particionado y estabilidad numérica

Cada objetivo se entrena de forma independiente aplicando una partición aleatoria entrenamiento prueba con proporción fijada en la configuración (30 %) y semilla reproducible (42).

5.6.3. Ajuste del modelo y predicción

Para cada objetivo se instancia un HGBR con los hiperparámetros definidos en la configuración del proyecto y se ajusta con los datos de entrenamiento. A continuación se predicen las respuestas sobre el subconjunto de prueba.

5.6.4. Métricas de evaluación

Se evalúa el rendimiento en el conjunto de prueba con las métricas habituales: coeficiente de determinación R^2, error absoluto medio (MAE) y error absoluto mediano (MedAE). Adicionalmente se reportan los tamaños de entrenamiento y prueba para contextualizar los resultados.

5.6.5. Importancia de rasgos y diagnóstico

Para facilitar la interpretación se calculan dos medidas de importancia:

- Importancia por ganancia media a partir de un modelo auxiliar Random Forest entrenado sobre el mismo conjunto de entrenamiento.
- Importancia por permutación, calculada sobre el conjunto de prueba para el HGBR y resumida en las diez características más influyentes.

Estas medidas permiten verificar que el modelo se apoya en variables con sentido físico como EI, L_{eff}, escalas de flecha del tipo L_{eff}^3/EI y módulos resistentes W_z, entre otras.

5.6.6. Persistencia y trazabilidad

Cada modelo entrenado se guarda en local junto con las características usadas. Esto garantiza que, en inferencia, la aplicación cargue exactamente la misma configuración de rasgos que se empleó en entrenamiento, evitando desviaciones entre fases.

5.6.7. Resultados en el conjunto de prueba

Los resultados del modelo final se evaluaron utilizando el conjunto de prueba, obtenido al dividir el conjunto total de datos en un 70 % para entrenamiento y un 30 % para validación del rendimiento del modelo. La Tabla 5.6 muestra las métricas obtenidas.

Tabla 5.6. Rendimiento del HGBR en el conjunto de prueba. El desplazamiento está en mm y la tensión en MPa.

target	R2	MAE	MedAE	n_train	n_test
max_displacement	0.843	11.6189	2.67829	2839	1218
max_stress	0.960	36.7916	14.1722	2839	1218

Los resultados muestran un ajuste muy alto para la tensión máxima y un rendimiento sólido para el desplazamiento. Es habitual que el desplazamiento resulte más difícil de predecir porque depende con mayor sensibilidad de la posición relativa de las cargas y de la longitud efectiva, mientras que la tensión se ancla bien a escalas basadas en el módulo resistente y en sustitutos del momento máximo.

5.6.8. Importancia de características

A continuación se resumen las diez características más influyentes según dos enfoques complementarios: un *Random Forest* auxiliar entrenado sobre el mismo conjunto de entrenamiento (importancia por ganancia, Tabla 5.7 y 5.9), y la importancia por permutación calculada sobre el conjunto de prueba para el propio HGBR, ver Tabla 5.8 y 5.10. Se observan patrones coherentes con la teoría de vigas: para el desplazamiento destacan combinaciones que incluyen L_{eff} y escalas de tipo L^3/EI o L^4/EI, mientras que para la tensión domina la escala σ inducida por el momento sobre el módulo resistente.

Tabla 5.7. Top-10 importancia de características con Random Forest auxiliar para max_displacement

feature	importance (RF)
sigma_scale_L_effective_squared	0.296539
sigma_delta_effective_interaction	0.110485
L_effective_squared_L_over_rz	0.069631
L_effective_4_over_EI	0.062302
normalized_load	0.034780
moment_over_stiffness	0.031084
delta_scale_effective	0.030350
log_delta_scale	0.027441
Iz_over_A2	0.021235
J_over_A	0.018755

Tabla 5.8. Top-10 importancia por permutación para max_displacement

feature	importance (permutation)
sigma_scale_L_effective_squared	0.490651
sigma_delta_effective_interaction	0.061058
L_effective_squared_L_over_rz	0.004851
L_effective_4_over_EI	0.032919
normalized_load	0.000000
moment_over_stiffness	0.052159
delta_scale_effective	0.019375
log_delta_scale	0.017604
Iz_over_A2	0.083135
J_over_A	0.007130

Tabla 5.9. Top-10 importancia de características con Random Forest auxiliar para max_stress

feature	importance (RF)
log_sigma_scale	0.816521
Iz_over_A2	0.042763
sigma_delta_effective_interaction	0.035271
sigma_scale	0.011597
sigma_delta_interaction	0.006522
load_over_stiffness	0.005669
K_factor_squared	0.005457
K_factor_cubed	0.005354
K_factor	0.005180
load_intensity	0.004783

Tabla 5.10. Top-10 importancia por permutación para max_stress

feature	importance (permutation)
log_sigma_scale	1.713227
Iz_over_A2	0.068850
sigma_delta_effective_interaction	0.107981
sigma_scale	0.003374
sigma_delta_interaction	0.001788
load_over_stiffness	0.003760
K_factor_squared	0.042617
K_factor_cubed	0.000000
K_factor	0.000000
load_intensity	0.001911

En conjunto, la importancia aprendida por el modelo coincide con las escalas físicas que gobiernan el problema. En tensión, la dominancia de log_sigma_scale y de razones geométricas normalizadas como Iz_over_A2 es consistente con que σ se ancla en el momento y el módulo resistente. En desplazamiento, la relevancia de combinaciones con L_{eff} y con δ escala la respuesta con L^3/EI y explica la sensibilidad a las condiciones de apoyo y a la posición de cargas. Esta alineación mejora la interpretación de los resultados y favorece la generalización.

5.6.9. Conclusión HGBR

El modelo puede considerarse finalizado, ya que en la última iteración de entrenamiento se obtuvieron métricas satisfactorias para ambas variables objetivo, como se muestra en la Tabla 5.6.

Para el modelo de desplazamientos máximos se alcanzó un coeficiente de determinación $R^2 = 0.843$ y un error medio absoluto de 11.62 mm. Además, el error absoluto mediano fue de 2.678 mm, lo que indica que el 50 % de las predicciones presentan un error inferior a dicho valor. Considerando estos resultados, el modelo de desplazamientos puede calificarse como aceptable, ya que los errores obtenidos son bajos en relación con el objetivo de estimar el desplazamiento máximo en vigas.

En cuanto a las tensiones máximas, se obtuvo un coeficiente de determinación $R^2 = 0.960$ y un error medio absoluto de 36.79 MPa. El error absoluto mediano fue de 14.17 MPa, lo que implica que la mitad de las predicciones presentan un error inferior a este valor. Estos resultados son muy satisfactorios, especialmente si se compara con el límite elástico de los materiales empleados como el aluminio, con una resistencia de 270 MPa.

Por último, los modelos entrenados se guardaron como artefactos independientes en formato `joblib`, uno por cada variable objetivo: `model_max_displacement_HGB.joblib` y `model_max_stress_HGB.joblib`. Esta separación permite seleccionarlos desde la interfaz Alabeam según la predicción requerida, reducir el tiempo y la memoria de carga al utilizar solo el predictor necesario, y facilitar su mantenimiento al poder actualizar cada archivo de forma aislada sin afectar al resto del sistema.

5.7 Entrenamiento de la red neuronal

La red neuronal se entrena a partir del conjunto de datos ya preprocesado (Sección 5.5), con dos objetivos independientes: `max_displacement` (desplazamiento máximo) y `max_stress` (tensión máxima). El propio *script* construye la matriz de características excluyendo identificadores y campos categóricos crudos (`model_name`, `section_type`, `material`, `support_L`, `support_R`), convierte todo a `float32` y sustituye valores no válidos (NaN, $\pm\infty$) por 0.0, lo que permite entrenar con datos reales donde algunas magnitudes pueden no aplicar en ciertos casos. Esta preparación se implementa en la función `safe_prepare_features` del fichero de entrenamiento de la red neuronal.

Para evaluar con rigor, el conjunto de datos se divide de forma reproducible en tres particiones: prueba (30 %), validación (20 % de la parte restante) y entrenamiento. Se fija una semilla global (42) para NumPy y TensorFlow, de modo que los resultados sean repetibles (`DEFAULT_RANDOM_STATE=42`). Antes de entrenar se estandarizan las características con `StandardScaler` con el que se obtienen los mejores resultados (también se probaron alternativas como `MinMax`, `Robust` o `PowerTransformer`), guardando posteriormente el escalador y la lista de columnas utilizadas para asegurar su uso idéntico en despliegue.

El *script* ofrece tres arquitecturas densas en Keras. Las variantes `simple` y `medium` usan bloques densos con activación ReLU, normalización por lotes y abandono, además de regularización L2 en los pesos.

En concreto, la opción `simple` emplea capas de 64 y 32 neuronas con abandonos de 0.30 y 0.20, mientras que `medium` amplía la capacidad con 128, 64 y 32 neuronas y abandonos de 0.35 y 0.25. La arquitectura `ultimate` sigue un diseño en paralelo con tres caminos: uno profundo (64–32–16), otro ancho (128–32) y un bloque residual ligero (64–64 con salto), que se concatenan y pasan por capas finales de 32 y 16 neuronas antes de la salida escalar. Este esquema busca capturar relaciones de distinta complejidad sin disparar el número de parámetros y mostró buen compromiso entre sesgo y varianza en los ensayos.

El plan de optimización utiliza el optimizador Adam y una función de pérdida Huber, adecuada cuando puede haber valores atípicos moderados. Para estabilizar y acelerar la convergencia se combinan varias rutinas: parada temprana con restauración de los mejores pesos, reducción automática de la tasa de aprendizaje al estancarse la validación y un programador triangular cíclico opcional que explora pequeñas variaciones periódicas en la tasa de aprendizaje. Los hiperparámetros por defecto son 250 épocas como máximo, tamaño de lote 32 y una paciencia de 30 épocas.

El entrenamiento se realiza por objetivo: para cada una de las variables `max_displacement` y `max_stress` se ajusta un modelo de salida única, se evalúa sobre el conjunto de prueba y se registran métricas estándar de regresión (R^2, MAE, MedAE, RMSE y MAPE). Tras el ajuste, el *script* guarda el mejor modelo en formato `.keras` junto con el escalador y las columnas de entrada en un `.joblib`, y genera además un resumen `JSON` con rutas y métricas para trazabilidad. Esta lógica se encapsula en las funciones `train_one_target` y `run_training`, e incluye una interfaz de línea de comandos que permite elegir objetivo, arquitectura, tipo de escalado y tamaños de partición.

Con este flujo, la red neuronal queda entrenada de forma reproducible, con preprocesado coherente y artefactos de despliegue versionados, de modo que posteriormente puede cargarse el modelo y aplicar exactamente el mismo escalado sobre las mismas características al integrarlo en la aplicación.

5.7.1. Resultados del modelo neuronal

La red neuronal se entrenó utilizando la tres arquitecturas mencionadas anteriormente (*simple, medium, ultimate*) y se evaluaron de forma independiente para cada una de las dos variables objetivo.

En la Tabla 5.11 se recogen los resultados obtenidos para los tres modelos entrenados de la red neuronal (`Simple`, `Medium` y `Ultimate`) en la predicción tanto del desplazamiento máximo como de la tensión máxima.

Para el caso del desplazamiento máximo, el modelo `Ultimate` ofrece el mejor rendimiento, alcanzando un coeficiente de determinación $R^2 = 0.8686$, un error medio absoluto (MAE) de 10.20 mm y un error cuadrático medio (RMSE) de 26.49 mm. Por tanto, se selecciona este modelo como el definitivo para la predicción de desplazamientos en vigas, dado que presenta una mayor capacidad de generalización y un error significativamente menor respecto a las otras arquitecturas.

En el caso de la tensión máxima, el modelo `Simple` obtiene el mejor equilibrio entre precisión y estabilidad, con un $R^2 = 0.9290$, un MAE de 54.41 MPa y un RMSE de 107.30 MPa. Aunque el modelo `Ultimate` presenta un MAE ligeramente inferior, su peor rendimiento en R^2 y RMSE indica una mayor dispersión en los errores, lo que resulta menos adecuado en este contexto, donde es prioritario minimizar los errores elevados cercanos al límite elástico del material. Por ello, se adopta el modelo `Simple` como el más apropiado para la predicción de tensiones.

Tabla 5.11. Resultados de los modelos neuronales para el desplazamiento y la tensión máximos

Variable objetivo	Modelo	R^2	MAE	RMSE
Desplazamiento máx. [mm]	Simple	0.7902	13.05	33.47
	Medium	0.8360	11.03	29.59
	Ultimate	**0.8686**	**10.20**	**26.49**
Tensión máx. [MPa]	**Simple**	**0.9290**	54.41	**107.30**
	Medium	0.9264	55.00	109.25
	Ultimate	0.8982	**50.98**	128.42

En la Figura 5.5 se presentan las curvas de entrenamiento y validación del error (*Loss*) y del error medio absoluto (MAE) correspondientes a las tres arquitecturas de red neuronal desarrolladas para la predicción del desplazamiento máximo: `Simple`, `Medium` y `Ultimate`.

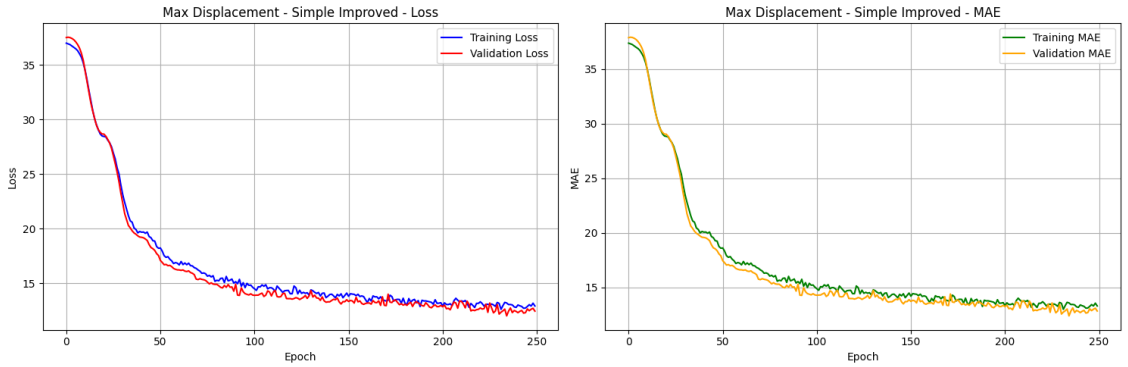
En todos los casos, las curvas muestran una disminución pronunciada del error durante las primeras épocas, seguida de una fase de estabilización en la que la pérdida converge hacia un valor mínimo. Este comportamiento es indicativo de un entrenamiento adecuado, sin síntomas de divergencia ni sobreajuste severo.

En la arquitectura `Simple`, las curvas de entrenamiento y validación evolucionan de forma muy similar, manteniendo una diferencia reducida entre ambas a lo largo de las épocas. Esto sugiere una buena capacidad de generalización, aunque el valor final del error es más elevado en comparación con las arquitecturas más complejas, lo que limita su precisión.

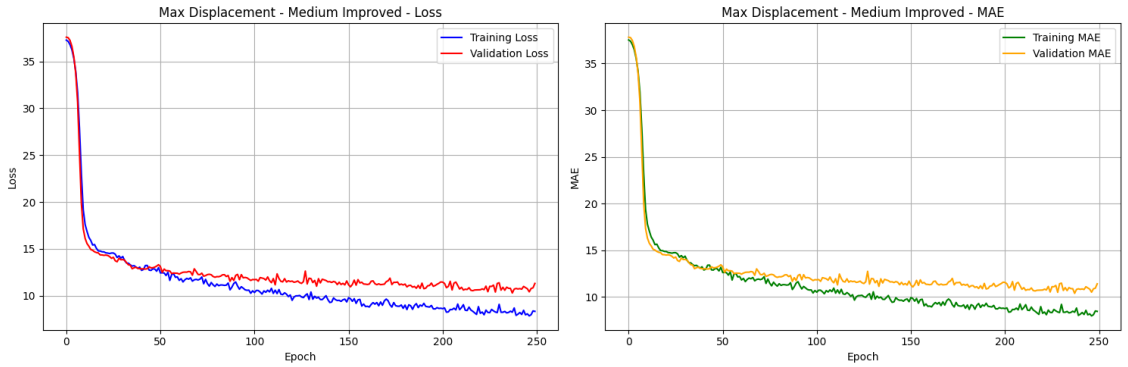
El modelo `Medium` presenta una convergencia más lenta, pero alcanza valores de pérdida y MAE inferiores a los del modelo `Simple`. La diferencia entre las curvas de entrenamiento y validación aumenta ligeramente, lo que indica un ajuste más fino de los parámetros y una mayor capacidad de representación, aunque con un ligero riesgo de sobreajuste en las últimas épocas.

Por último, la arquitectura `Ultimate` muestra el mejor rendimiento global. Se observa una reducción notable del error de entrenamiento hasta valores muy bajos y una validación que se mantiene estable sin incrementos significativos, lo que refleja un equilibrio adecuado entre ajuste y generalización. No obstante, la separación entre las curvas de entrenamiento y validación es algo mayor que en los modelos previos, lo que puede asociarse a una mayor complejidad de la red y, por tanto, a una sensibilidad superior frente al ruido de los datos.

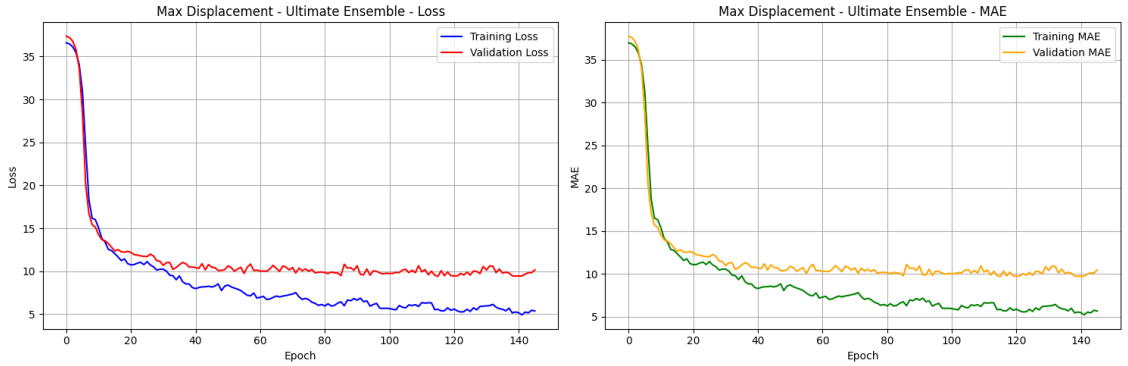
En conjunto, las gráficas confirman que el modelo `Ultimate` es el que alcanza la mejor precisión en la predicción del desplazamiento máximo, con un proceso de aprendizaje eficiente y una validación estable, justificando su selección como arquitectura final.



(a) Simple



(b) Medium



(c) Ultimate

Figura 5.5. Evolución de la pérdida (*Loss*) y del MAE durante el entrenamiento de las tres arquitecturas de red neuronal para la predicción del desplazamiento máximo

En la Figura 5.6 se muestran las curvas de entrenamiento y validación del error (*Loss*) y del error medio absoluto (MAE) correspondientes a las tres arquitecturas de red neuronal empleadas para la predicción de la tensión máxima: `Simple`, `Medium` y `Ultimate`.

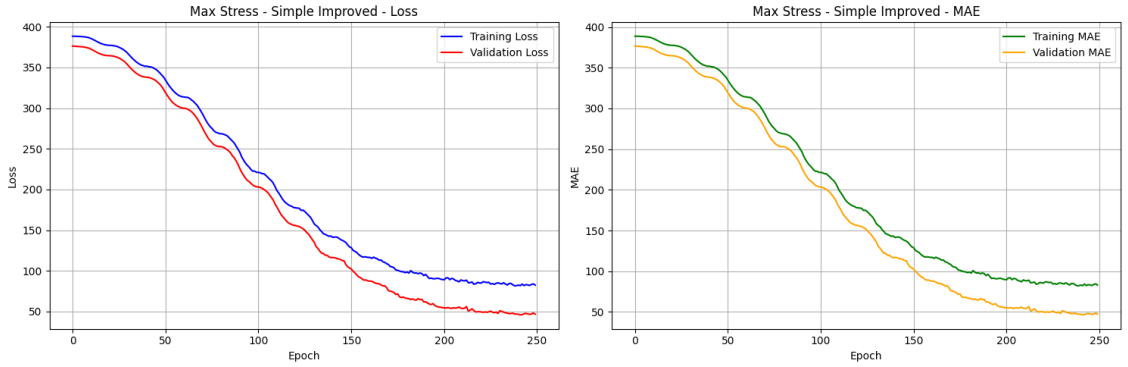
En general, las tres arquitecturas presentan un comportamiento estable durante el entrenamiento, con una reducción rápida del error en las primeras épocas y una tendencia a la convergencia conforme avanza el proceso de aprendizaje. Esto indica que el modelo ha logrado ajustarse correctamente a los datos sin presentar problemas significativos de divergencia.

En el modelo `Simple`, tanto la pérdida de entrenamiento como la de validación disminuyen de forma progresiva y casi paralela, con una separación pequeña entre ambas curvas. Este comportamiento evidencia una buena capacidad de generalización y ausencia de sobreajuste, aunque el ritmo de aprendizaje es más lento y los valores finales de error son superiores a los de las arquitecturas más complejas.

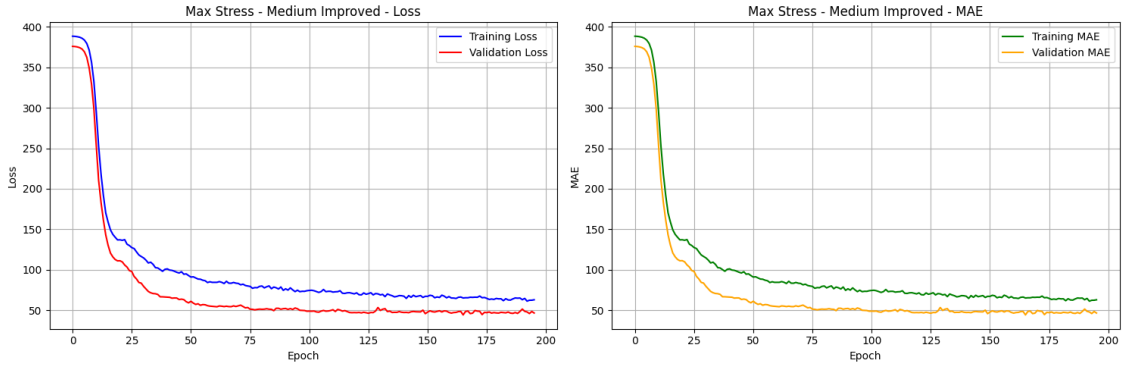
La arquitectura `Medium` muestra una caída más rápida de la pérdida durante las primeras épocas y alcanza valores de error menores que el modelo `Simple`. La ligera separación entre las curvas de entrenamiento y validación sugiere que la red ha captado mejor las relaciones no lineales de los datos, aunque empieza a reflejar una tendencia leve al sobreajuste hacia el final del entrenamiento.

Por último, el modelo `Ultimate` alcanza la convergencia más estable y uniforme, con curvas de entrenamiento y validación prácticamente paralelas y con una diferencia muy reducida. Este resultado confirma que la arquitectura es capaz de generalizar correctamente y mantener un error controlado, aunque la pérdida de validación se estabiliza ligeramente por encima de la de entrenamiento, lo cual es un comportamiento normal en redes bien ajustadas.

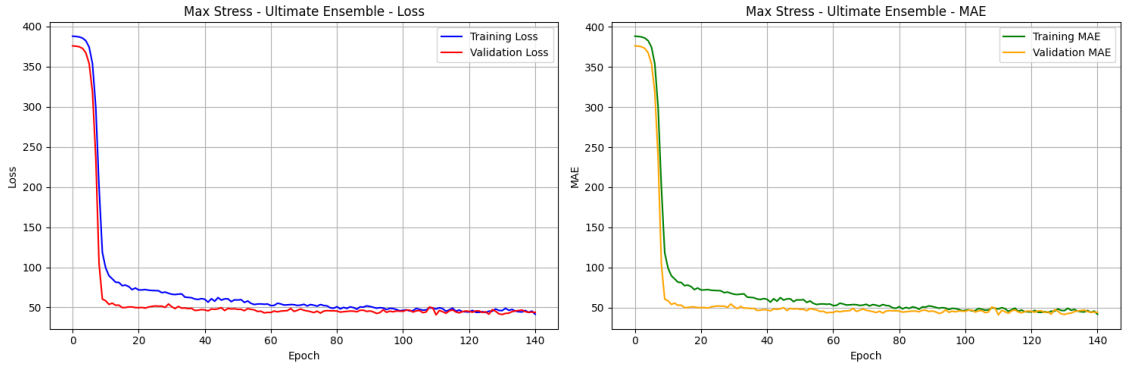
En conjunto, las gráficas de las curvas de pérdida confirman que todas las arquitecturas se entrenaron de forma adecuada, siendo la red `Ultimate` la que presenta un aprendizaje más eficiente y una validación más consistente. No obstante, el modelo `Simple` se considera más robusto frente a posibles errores extremos, por lo que se selecciona finalmente para la predicción de tensiones máximas.



(a) Arquitectura `Simple`



(b) Arquitectura `Medium`

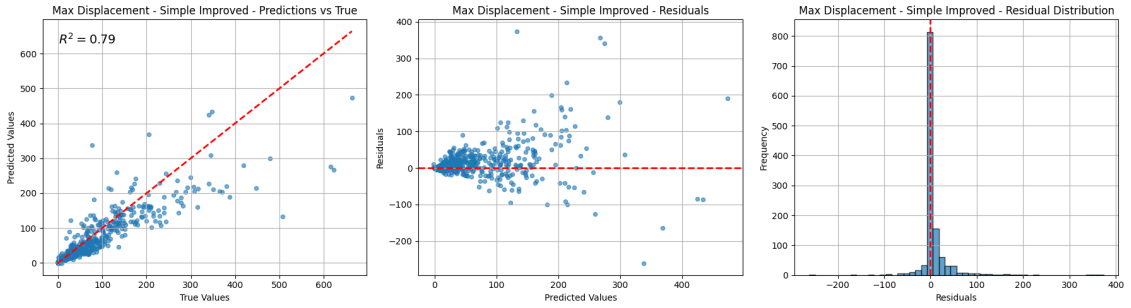


(c) Arquitectura `Ultimate`

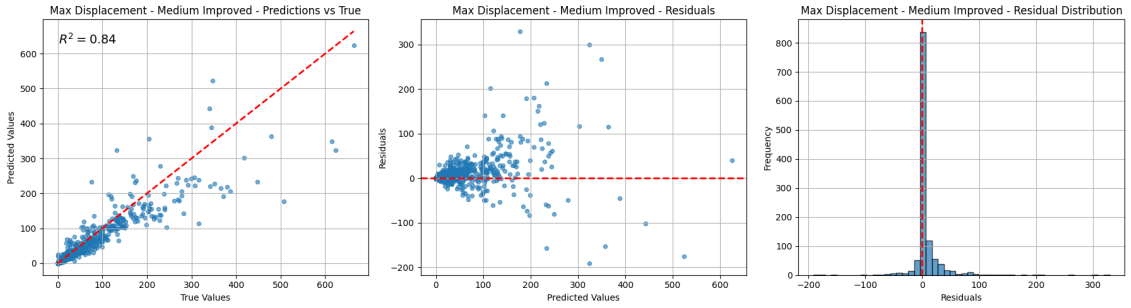
Figura 5.6. Evolución de la pérdida (*Loss*) y del MAE durante el entrenamiento de las tres arquitecturas de red neuronal para la predicción de la tensión máxima

En la Figura 5.7 se presentan los resultados de las predicciones para las tres arquitecturas de red neuronal desarrolladas (`Simple`, `Medium` y `Ultimate`) en la estimación del desplazamiento máximo. Cada fila muestra, de izquierda a derecha, la comparación entre valores reales y predichos, el diagrama de residuos y la distribución de los errores.

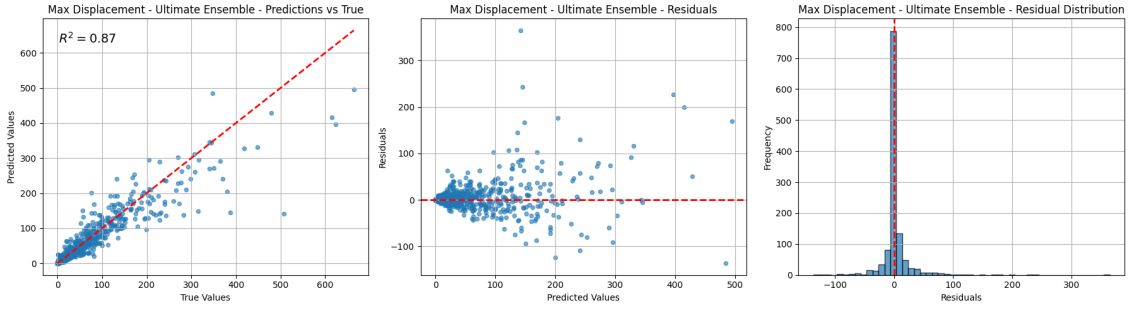
En los gráficos de dispersión (predicciones frente a valores reales) se observa que la nube de puntos sigue de manera general la diagonal ideal (línea discontinua roja), lo que indica una buena correlación entre los valores predichos y los reales.



(a) Arquitectura `Simple`



(b) Arquitectura `Medium`



(c) Arquitectura `Ultimate`

Figura 5.7. Comparación valores reales/predichos, distribución y análisis de residuos para las tres arquitecturas de red neuronal en la predicción del desplazamiento máximo

Este comportamiento mejora progresivamente desde la arquitectura `Simple` hasta la `Ultimate`, siendo esta última la que alcanza un coeficiente de determinación más alto ($R^2 = 0.87$) y una alineación más estrecha con la línea de referencia, lo que evidencia una mayor precisión del modelo.

Los diagramas de residuos muestran que, en los tres casos, los errores se distribuyen de forma aproximadamente simétrica alrededor de cero, sin patrones sistemáticos evidentes. Sin embargo, se aprecia una ligera dispersión creciente de los residuos a medida que aumentan los valores predichos, especialmente en los modelos `Simple` y `Medium`, lo que indica que las predicciones tienden a ser menos precisas para desplazamientos más grandes. En el modelo `Ultimate`, la dispersión es más contenida y la concentración de los residuos en torno a cero es mayor, lo que confirma una mejor capacidad de generalización.

Por último, las distribuciones de residuos presentan una forma aproximadamente normal, centrada en cero, lo que sugiere que los errores no muestran sesgos significativos. La arquitectura `Ultimate` destaca por tener una distribución más estrecha, reflejando menor variabilidad y mayor consistencia en las predicciones.

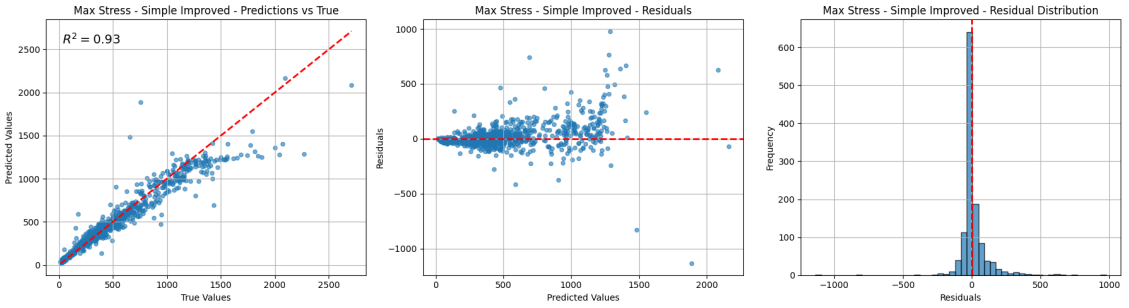
En conjunto, el análisis de las gráficas confirma que el modelo `Ultimate` es el más preciso y equilibrado para la predicción del desplazamiento máximo, al combinar una alta correlación entre valores reales y predichos con una distribución de errores estable y centrada.

En la Figura 5.8 se presentan los resultados obtenidos en la predicción de la tensión máxima para las tres arquitecturas de red neuronal: `Simple`, `Medium` y `Ultimate`. Cada una de ellas incluye, de izquierda a derecha, el gráfico de correlación entre valores reales y predichos, el diagrama de residuos y la distribución de los errores.

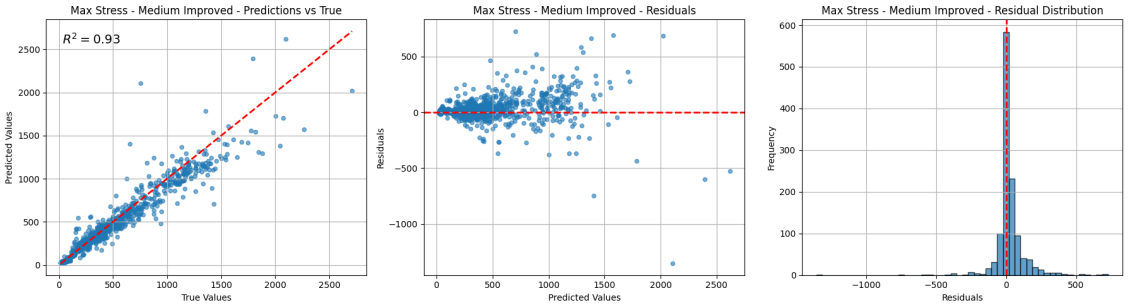
En los diagramas de dispersión se aprecia una fuerte correlación entre los valores reales y los predichos, con los puntos distribuidos mayoritariamente a lo largo de la diagonal de referencia (línea discontinua roja). El modelo `Simple` alcanza un coeficiente de determinación $R^2 = 0.93$, lo que evidencia un excelente ajuste entre las predicciones y los valores reales. Las arquitecturas `Medium` y `Ultimate` presentan también un buen desempeño ($R^2 = 0.93$ y $R^2 = 0.90$, respectivamente), aunque con una ligera mayor dispersión en los valores más altos de tensión.

El análisis de los residuos muestra que, en los tres modelos, los errores se distribuyen de forma aproximadamente simétrica en torno a cero, sin tendencias sistemáticas claras, lo que indica que las predicciones no presentan sesgos evidentes. En el caso de la arquitectura `Simple`, los residuos son más compactos y se concentran alrededor del eje horizontal, reflejando una buena generalización y menor variabilidad. En cambio, en el modelo `Ultimate`, los residuos muestran una dispersión ligeramente mayor, especialmente para tensiones elevadas, lo que sugiere una mayor sensibilidad a los valores extremos del conjunto de datos.

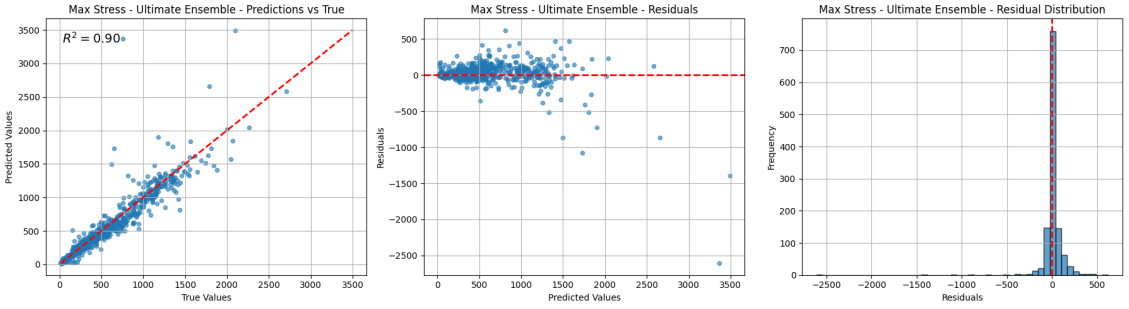
Las distribuciones de los residuos presentan una forma casi normal, centrada en cero, con colas reducidas y una frecuencia muy elevada en torno al valor medio. De nuevo, la arquitectura `Simple` destaca por una distribución más estrecha y simétrica, mientras que los modelos `Medium` y `Ultimate` presentan colas algo más extendidas, consecuencia de una mayor complejidad en la red.



(a) Arquitectura `Simple`



(b) Arquitectura `Medium`



(c) Arquitectura `Ultimate`

Figura 5.8. Comparación valores reales/predichos, distribución y análisis de residuos para las tres arquitecturas de red neuronal en la predicción de la tensión máxima

En conjunto, el análisis de estas gráficas confirma la alta capacidad predictiva del modelo `Simple`, que combina una excelente correlación entre valores reales y estimados con una distribución de errores bien equilibrada y sin sesgos. Por estos motivos, este modelo se considera el más adecuado para la predicción de la tensión máxima en vigas.

Finalmente los modelos seleccionados son la arquitectura *ultimate* para los desplazamientos y la arquitectura *simple* para la tensión máxima. Para su uso posterior, cada predictor se guarda como artefactos independientes: el modelo en formato Keras y el escalador con las columnas de entrada. En particular, para desplazamiento máximo se genera `models/model_max_displacement_ultimate.keras` y `models/scaler_max_displacement_ultimate.joblib`; para tensión máxima, `models/model_max_stress_simple.keras` y `models/scaler_max_stress_simple.joblib`. Esta separación permite cargar solo el predictor necesario en la aplicación, garantiza que el preprocesado reproducible (escalado y orden de variables) sea idéntico en inferencia y facilita la actualización de cada objetivo de forma aislada.

5.8 Predicción empleando los modelos entrenados

5.8.1. Predicción con el modelo HGBR

Para realizar predicciones sobre nuevos casos se emplea el *script* `HGBR_predict.py`. El programa carga los modelos entrenados en formato `joblib`, prepara las entradas de acuerdo con el conjunto de características usado en el entrenamiento y, si existen valores reales en el CSV, calcula métricas y genera gráficos de diagnóstico. El flujo interno es:

- `load_model`: carga un paquete `joblib` con las claves `"hgb"` (modelo), `"features"` (lista de variables) y `"eps"` (parámetro almacenado para referencia).
- `preprocess_input`: toma el CSV, selecciona solo columnas numéricas y las cruza con `ML_CANDIDATE_FEATURES` del módulo `config`. El resultado es la matriz `X` con el orden de variables correcto.
- `predict_and_plot`: aplica `hgb.predict(X)` y, si hay verdad terreno, calcula R^2 y `MAE`; además guarda tres figuras: correlación real–predicho, histograma de residuales y residuales frente a predicción.

La ejecución por línea de comandos es:

```
python HGBR_predict.py <input_csv> \
<model_max_displacement_HGB.joblib> \
<model_max_stress_HGB.joblib> \
<output_dir>
```

donde `<input_csv>` es un fichero con las mismas columnas de entrada utilizadas en el entrenamiento (no es necesario escalar ni normalizar), y `<output_dir>` es el directorio donde se deja el informe gráfico y las predicciones.

El *script* espera, si están disponibles, las columnas objetivo `max_displacement` y `max_stress` (definidas en `config.TARGET_COLUMNS`).

La Tabla 5.12 muestra las entradas necesarias para hacer predicciones a partir de un conjunto de datos ya procesado utilizando el modelo HGBR entrnado. En la Tabla 5.13 se muestran las salidas esperadas una vez terminadas las predicciones.

Tabla 5.12. Entradas requeridas por el script de predicción con HGBR

Entrada	Descripción
`<input_csv>`	Archivo `CSV` con las características numéricas de entrada. Si contiene las columnas `max_displacement` y `max_stress`, el script calcula y reporta las métricas de evaluación correspondientes.
`model_max_displacement_HGB.joblib`	Modelo `joblib` entrenado para el objetivo de desplazamiento máximo.
`model_max_stress_HGB.joblib`	Modelo `joblib` entrenado para el objetivo de tensión máxima.
`<output_dir>`	Directorio de salida donde se guardan los gráficos generados y el archivo `CSV` con las predicciones.

Tabla 5.13. Salidas generadas por el script `HGBR_predict.py`

Fichero generado	Contenido
`correlacion_max_displacement.png`	Gráfico de dispersión entre valores reales y predichos para el desplazamiento máximo, con línea de identidad y anotación del R^2 y `MAE`.
`residuales_hist_max_displacement.png`	Histograma de los residuales obtenidos en la predicción del desplazamiento máximo.
`residuales_vs_pred_max_displacement.png`	Gráfico de residuales frente a valores predichos, utilizado para diagnosticar posibles sesgos del modelo.
`correlacion_max_stress.png`	Gráfico análogo al anterior, correspondiente al objetivo de tensión máxima.
`residuales_hist_max_stress.png`	Histograma de residuales obtenidos en la predicción de la tensión máxima.
`residuales_vs_pred_max_stress.png`	Gráfico de residuales frente a valores predichos para la tensión máxima.
`predicciones.csv`	Archivo `CSV` de salida que incluye las predicciones generadas en nuevas columnas: `max_displacement_pred` y `max_stress_pred`.

Notas finales: el modelo HGBR no requiere estandarización, el script filtra de forma segura las columnas no numéricas y respeta el orden de `features` almacenado en los paquetes `joblib`. Si el conjunto de columnas del CSV no coincide con el esperado, el cruce con `ML_CANDIDATE_FEATURES` evita errores de forma silenciosa, aunque conviene revisar el registro de métricas y los gráficos para detectar posibles pérdidas de información en la entrada.

5.8.2. Predicción con red neuronal

Este script (`NN_predict.py`) realiza predicción con los modelos entrenados de Keras. Carga el modelo `.keras` y el escalador guardado en `.joblib`, prepara las entradas desde un CSV y genera las predicciones para el objetivo indicado. Si el CSV trae la columna de verdad terreno, calcula R^2 y MAE y guarda figuras de evaluación.

Flujo de trabajo

1. Carga del archivo `.joblib` que contiene el escalador y la lista de columnas usadas en entrenamiento (`feature_cols`).
2. Lectura del CSV de entrada y construcción de la matriz `X` exactamente con `feature_cols`. Las columnas ausentes se crean con valor `0.0` y se fuerza el tipo numérico seguro (`float32`).
3. Transformación de `X` con el escalador y carga del modelo `.keras`.
4. Predicción del objetivo en todo el conjunto.
5. Si existe la columna objetivo, cálculo de R^2 y MAE y generación de tres figuras: correlación real vs. predicho, histograma de residuales y residuales frente a predicción.
6. Opcionalmente, exportación de un CSV con las predicciones añadiendo una nueva columna `predicted_<target>`.

Parámetros principales

- `--data`: ruta al CSV con las variables de entrada.
- `--model`: ruta al modelo `.keras`.
- `--scaler`: ruta al `.joblib` exportado durante el entrenamiento; incluye `scaler` y `feature_cols`.
- `--target`: nombre del objetivo a predecir (`max_stress` o `max_displacement`).
- `--ycol` (opcional): nombre alternativo de la columna objetivo si no coincide con `--target`.
- `--outdir`: carpeta donde se guardan las figuras y el informe de métricas.
- `--save_csv` (opcional): ruta para escribir un CSV con las predicciones añadidas.

Ejemplos de uso

```
--- Tensión máxima ---
python NN_predict.py \
--data data/test.csv \
--model models/model_max_stress_ultimate.keras \
--scaler models/scaler_max_stress_ultimate.joblib \
--target max_stress \
--outdir eval_stress \
--save_csv predicciones_stress.csv
```

```
--- Desplazamiento máximo ---
python NN_predict.py \
--data data/test.csv \
--model models/model_max_displacement_ultimate.keras \
--scaler models/scaler_max_displacement_ultimate.joblib \
--target max_displacement \
--outdir eval_disp
```

Salida Si el CSV incluye la variable objetivo, el script imprime R^2 y MAE en consola y guarda en `--outdir` las figuras `correlacion_<target>.png`, `residuales_hist_<target>.png` y `residuales_vs_pred_<target>.png`. Con `--save_csv` se genera un CSV con la columna `predicted_<target>`.

Nota práctica Si aparecen avisos de columnas faltantes significa que el CSV no contiene todas las características de entrenamiento. Para explotar bien el modelo conviene reconstruir dichas columnas con el mismo preprocesado que se aplicó antes de entrenar.

5.9 Conclusiones del desarrollo de modelos

Se ha implementado un flujo completo para la generación y explotación de datos sintéticos orientado a la predicción del comportamiento estructural de vigas. La cadena metodológica ha comprendido: (i) modelado paramétrico y mallado en HyperMesh, (ii) resolución numérica con Optistruct, (iii) extracción automática de resultados mediante PyNastran para conformar el *dataset* con variables de entrada (longitud, tipo de sección, material, cargas y condiciones de contorno) y salidas de interés (desplazamiento y tensión máximos), y (iv) un preprocesado extensivo con *feature engineering* basado en expresiones de cálculo de vigas, con el fin de incorporar información física relevante y estabilizar el aprendizaje.

Con un total de 4058 casos y una partición 70/30 (2 839 para entrenamiento y 1 218 para prueba), el modelo HGBR ha mostrado un rendimiento sólido en ambos objetivos: para *desplazamiento máximo* se alcanzó $R^2 = 0.843$, MAE $= 11.62$ mm y MedAE $= 2.68$ mm; para *tensión máxima*, $R^2 = 0.960$, MAE $= 36.79$ MPa y MedAE $= 14.17$ MPa. Estos resultados confirman que un modelo de *boosting* con histogramas, correctamente regularizado e informado por variables ingenieriles, es capaz de capturar con eficacia las no linealidades del problema.

Con objeto de comparar enfoques, se entrenaron distintas arquitecturas de red neuronal. La mejor configuración para *desplazamiento* fue la arquitectura `ultimate`, con $R^2 = 0.8686$, MAE $= 10.20$ mm, MedAE $= 1.91$ mm y RMSE $= 26.49$ mm, superando al HGBR en todas las métricas principales y mostrando, por tanto, una mayor capacidad de generalización para este objetivo. En *tensión*, la arquitectura `simple` ofreció el mejor compromiso para el uso previsto del modelo, con $R^2 = 0.9290$, MAE $= 54.41$ MPa, MedAE $= 25.96$ MPa y RMSE $= 107.30$ MPa. Aunque algunas arquitecturas profundas reducen ligeramente el MAE, su peor R^2 y/o RMSE indican mayor variabilidad y errores extremos más probables, algo indeseable cerca del límite elástico.

En consecuencia, y de acuerdo con los análisis de aprendizaje, dispersión y residuos:

- Para desplazamiento máximo, se selecciona la red neuronal `ultimate`, por su mejor R^2, menor MAE y MedAE, y distribución de errores más concentrada.
- Para tensión máxima, se adopta el modelo neuronal `simple` (frente a alternativas más complejas), por su equilibrio entre precisión global y control de errores grandes (RMSE competitivo), criterio más seguro en un contexto con restricciones de resistencia.

Durante la validación se ha observado que el MAPE puede ser engañoso en desplazamientos debido a la presencia de valores reales cercanos a cero, lo que magnifica el porcentaje de error, por ello, el análisis se ha apoyado preferentemente en R^2, MAE, MedAE y RMSE. Asimismo, aunque los datos sintéticos permiten explorar un dominio amplio de diseño con bajo coste, la capacidad de *extrapolación* fuera del espacio de parámetros muestreado debe considerarse limitada.

En suma, el pipeline propuesto integra simulación numérica, extracción sistemática de resultados y aprendizaje supervisado con *feature engineering* físico, alcanzando modelos precisos y estables para ambos objetivos. Esta base es adecuada para su integración en la aplicación Alabeam y para su extensión futura a: (i) ampliación y estratificación del *dataset* con nuevos materiales y esquemas de carga, (ii) calibración con datos experimentales o de campo, (iii) técnicas de estimación de incertidumbre (*ensembles*, *MC dropout*) y umbrales de confianza operativos, y (iv) optimización *multiobjetivo* (peso–rigidez–tensión) asistida por los modelos seleccionados.

5.10 Implementación de la interfaz Alabeam

La interfaz Alabeam se ha desarrollado con Streamlit para ofrecer una herramienta de predicción accesible a usuarios sin experiencia en cálculo. La aplicación encapsula el preprocesado y los modelos entrenados, permitiendo seleccionar la geometría y las condiciones de contorno de una viga 2D y obtener al instante la estimación del desplazamiento máximo y de la tensión equivalente de Von Mises máxima.

La Figura 5.9 muestra el aspecto general de la interfaz de Alabeam.

Figura 5.9. Interfaz gráfica de la app Alabeam en Streamlit

5.10.1. Estructura del proyecto

El proyecto se encuentra en el repositorio referenciado en [29]. El proyecto de Alabeam sigue la siguiente estructura explicada en el Apéndice B.2

El archivo `Alabeam.py` contiene la interfaz y el enrutado de acciones. El módulo `preprocessing.py` replica exactamente el pipeline de creación de rasgos utilizado en el entrenamiento, lo que garantiza que las columnas, los escalados y las transformaciones son consistentes. En `models` se alojan los estimadores basados en árboles (`.joblib`). En `models_neural` se guardan los modelos Keras (`.keras`), los escaladores (`.joblib`) y el listado canónico de columnas de entrada para la red (`feature_columns_neuralnet.csv`).

5.10.2. Flujo de uso

1. El usuario define la longitud de la viga, el tipo de sección, sus dimensiones nominales, el material, la configuración de apoyos y hasta tres acciones concentradas o momentos. La nomenclatura de dimensiones sigue la empleada en HyperMesh y la tabla de equivalencias presentada en el desarrollo. En la Figura 5.10 se muestra una viga de ejemplo de 1500 mm de longitud, sección tipo rectangular hueca y material aluminio.

Figura 5.10. Ejemplo de selección de una viga en Alabeam

2. La aplicación valida unidades y rango. Se trabaja en mm, N y MPa. Se comprueban espesores positivos (Figura 5.11), esbeltez razonable y ausencia de mecanismos en los extremos (Figura 5.12).

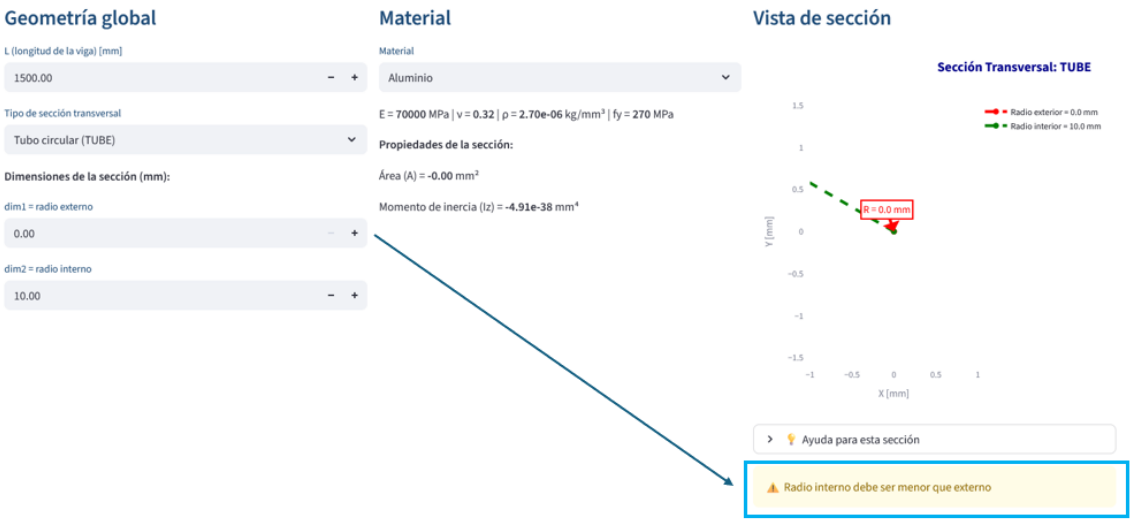


Figura 5.11. Ejemplo de advertencia por radio externo incoherente

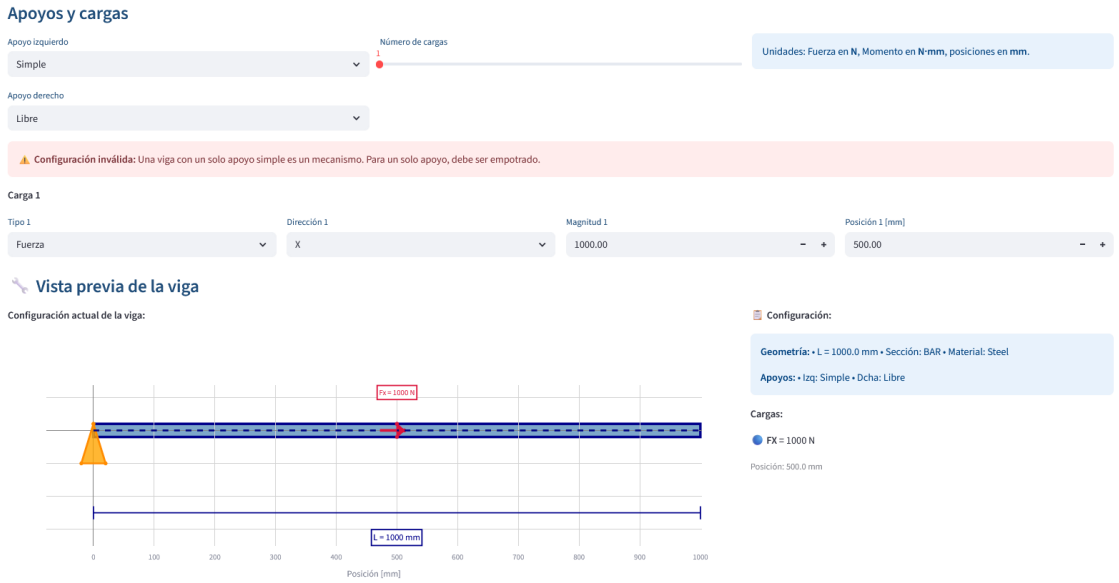


Figura 5.12. Ejemplo de advertencia por mecanismo

3. Como se muestra en la Figura 5.13 en la zona inferior se dispone del área de selección de cargas, donde se permite seleccionar hasta tres. También se muestra un esquema para previsualizar el modelo de la viga según los datos introducidos.

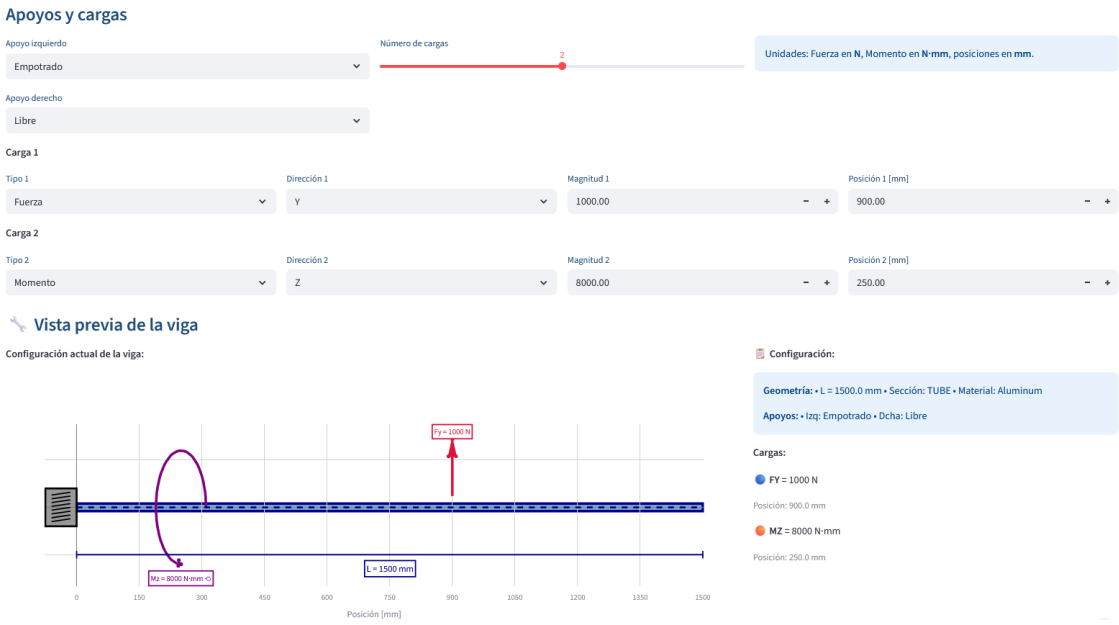


Figura 5.13. Selección de cargas y visualización

4. Con los datos validados, `preprocessing.py` genera las variables físico-informadas utilizadas en el entrenamiento: magnitudes como EI, W_z, r_z, escalas L^3/EI, L^4/EI, razones adimensionales y agregados de carga. Para la red neuronal, se reordena el vector de entrada con `feature_columns_neuralnet.csv` y se aplica el escalado guardado.

5. El usuario puede elegir el modelo de predicción: HGBR o red neuronal (Figura 5.14). Alabeam carga en memoria los modelos la primera vez y los reutiliza en sesiones posteriores. Además, se implementa el uso de si se desea un factor de seguridad para obtener predicciones más conservativas.

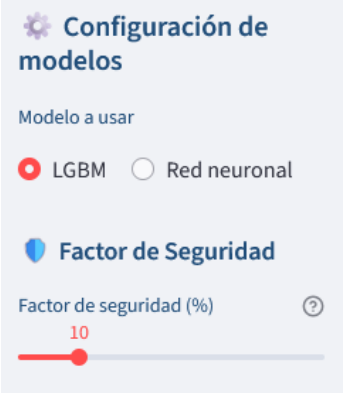


Figura 5.14. Selección de modelos y factor de seguridad

6. Se ejecuta la predicción y se presentan los resultados principales: desplazamiento máximo [mm] y tensión de Von Mises máxima [MPa]. De forma auxiliar se muestran recomendaciones según los resultados obtenidos. La Figura 5.15 muestra unos resultados de ejemplo con recomendaciones.

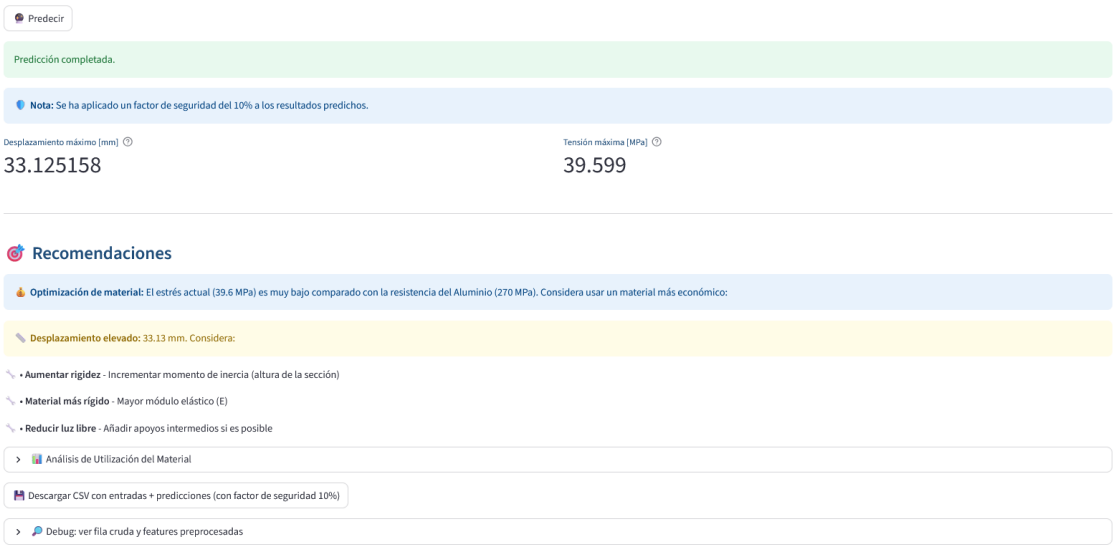


Figura 5.15. Resultado de predicciones en Alabeam

Se muestra de forma gráfica el porcentaje de uso del material para considerar el cambio por otro material. En el ejemplo de la Figura 5.16 las tensión máximas están muy por debajo del límite elástico por lo que se podría considerar otro material más económico.

Figura 5.16. Porcentaje de tensión respecto al límite elástico del material

7. Opcionalmente se puede descargar un resumen en CSV con entradas, variables y salidas para trazabilidad.

5.10.3. Motores de predicción

HGBR. Se emplean los ficheros `model_max_displacement_logHGB.joblib` y `model_max_stress_logHGB.joblib`. No requieren estandarización de entradas, pero sí el mismo conjunto de rasgos que en entrenamiento. El sufijo `logHGB` indica que se entrenó en un espacio de rasgos y objetivos coherente con las escalas físicas, tal y como se documenta en el capítulo de entrenamiento.

Red neuronal. Se usan `model_max_displacement_neuralnet.keras` y `model_max_stress_neuralnet.keras` junto con los escaladores `scaler_displacement_neuralnet.joblib` y `scaler_stress_neuralnet.joblib`. Antes de inferir, Alabeam aplica el transformador correspondiente y garantiza el orden de las columnas incluyéndolo en el *scaler*.

5.10.4. Entradas disponibles

- Longitud entre 150 y 5000 mm.
- Secciones: circular maciza, circular hueca, rectangular maciza, rectangular hueca y perfil en I. Las dimensiones se introducen con la misma convención de HyperMesh.
- Materiales: aceros al carbono e inoxidable, aluminio y titanio. Sus propiedades por defecto son las usadas en el dataset, editables por el usuario si lo desea.
- Apoyos: empotrado–libre, simple–simple y mixtos empotrado–simple o simple–empotrado.
- Acciones: hasta tres cargas puntuales en X o Y y momentos alrededor de Z, con posición a lo largo de la luz.

Factor de seguridad El usuario puede introducir un factor de seguridad $FS \geq 1$ para mayorar de forma conservadora las salidas. Las magnitudes mostradas se actualizan como

$$\sigma^{\star}_{\mathrm{vm}} = FS\,\sigma_{\mathrm{vm}}, \qquad w^{\star}_{\mathrm{máx}} = FS\,w_{\mathrm{máx}},$$

y la comparación con el material se realiza con el valor mayorado. El indicador de utilización se define como

$$\text{utilización} = \frac{\sigma^{\star}_{\mathrm{vm}}}{\sigma_y},$$

donde σ_y es el límite elástico del material seleccionado.

Sistema de recomendaciones Además de los valores numéricos, Alabeam genera recomendaciones automáticas que ayudan a interpretar la predicción y a orientar decisiones de rediseño. El motor aplica reglas simples basadas en dos señales: la utilización por tensión y el desplazamiento máximo.

Reglas basadas en tensión

Sea $r = \sigma^{\star}_{\mathrm{vm}}/\sigma_y$.

- Si $r < 0.3$ (baja utilización), se sugiere optimización por material. Por ejemplo, si se usa titanio se propone acero o aluminio; si se usa acero, se propone aluminio. El objetivo es reducir coste o masa cuando existe mucho margen.
- Si $0.9 < r < 1.0$ (zona próxima al límite), se emite un aviso e indica opciones para incrementar resistencia del material o aumentar área de sección.
- Si $r \geq 1.0$ (supera el límite elástico), se marca como condición crítica y se recomiendan materiales de mayor resistencia o un incremento de sección.
- Si $r > 0.5$ y no se cumple ninguno de los casos anteriores, se informa de una utilización moderada con margen razonable.

Reglas basadas en desplazamiento

Con el desplazamiento mayorado $w^{\star}_{\mathrm{máx}}$:

- Si $w^{\star}_{\mathrm{máx}} > 10\,\mathrm{mm}$, se recomienda aumentar rigidez elevando el momento de inercia de la sección, emplear un material con mayor módulo elástico E o reducir la luz mediante apoyos intermedios.
- Si $w^{\star}_{\mathrm{máx}} < 1\,\mathrm{mm}$, se informa de rigidez adecuada.

Reglas combinadas y de eficiencia

- Si $r > 0.9$ y $w^{\star}_{\mathrm{máx}} > 5\,\mathrm{mm}$, se emite una recomendación de revisión completa del diseño, dado que fallan simultáneamente rigidez y resistencia.
- Si $r < 0.2$ y $w^{\star}_{\mathrm{máx}} < 2\,\mathrm{mm}$, se identifica oportunidad clara de optimización, ya sea reduciendo dimensiones o empleando materiales más económicos.

5.10.5. Ejecución y despliegue

El archivo `requirements.txt` fija versiones de `scikit-learn`, `tensorflow`, `joblib`, `pandas`, `numpy` y `streamlit`. La aplicación se ejecuta con:

```
pip install -r requirements.txt
streamlit run Alabeam.py
```

En ejecución local, Alabeam guarda en caché los modelos y escaladores para minimizar el tiempo de respuesta. El log de predicción registra fecha, versión del modelo cargado y métricas de referencia almacenadas en `models/metrics` y en `models_neural/*performance.csv`.

Con el fin de facilitar la interacción con los modelos desarrollados y ofrecer una herramienta accesible desde cualquier dispositivo, la aplicación *Alabeam* se ha desplegado en la plataforma Streamlit Cloud. Este entorno permite ejecutar aplicaciones de *machine learning* con interfaz gráfica de forma sencilla y gratuita, integrando tanto la lógica de predicción como la visualización de resultados.

El despliegue se ha realizado utilizando la capa gratuita del servicio, lo que permite acceder a la aplicación directamente a través de la dirección web:

`https://Alabeam.streamlit.app`

Esta versión alojada en la nube ejecuta la interfaz desarrollada en *Streamlit* y los modelos entrenados de desplazamiento y tensión máxima, proporcionando una experiencia de usuario intuitiva y sin necesidad de instalación local. No obstante, la principal limitación del plan gratuito es que la instancia permanece activa únicamente durante aproximadamente 12 horas de inactividad. Transcurrido este tiempo, el servicio entra en modo *sleep* (reposo), por lo que al intentar acceder nuevamente es necesario pulsar el botón *“Wake up”* o *“Despertar”*. En ese momento, el entorno tarda unos segundos en reactivarse antes de que la aplicación vuelva a estar completamente operativa.

A pesar de esta limitación, el despliegue en *Streamlit Cloud* constituye una solución eficaz para la difusión y validación funcional del proyecto, al permitir el acceso remoto a *Alabeam* sin requerir configuraciones locales adicionales.

Capítulo 6. RESULTADOS Y DISCUSIÓN

6.1 Validación externa con 50 casos independientes

Para verificar la capacidad de generalización del modelo fuera de los datos usados en entrenamiento y prueba, se construyó un conjunto adicional de 50 vigas no vistas. Con este lote se evaluó el HistGradientBoostingRegressor y la red neuronal entrenados y se representó la correlación entre valores reales y predichos. La línea discontinua indica la recta identidad $y = x$.

Resultados con HGBR En la predicción de tensión de Von Mises máxima se obtuvo $R^2 \approx 0{,}978$ y MAE $\approx 21{,}38$ MPa (Figura 6.2), con puntos muy próximos a la diagonal y residuales acotados, aunque con ligera heterocedasticidad a tensiones altas (Figura 6.3). Para el desplazamiento máximo el ajuste fue más modesto ($R^2 \approx 0{,}872$, MAE $\approx 8{,}43$ mm) (Figura 6.1): los puntos de mayor desplazamiento muestran mayor dispersión y una tendencia a subestimar los valores más grandes, coherente con un sesgo hacia regímenes más rígidos del espacio de diseño. Esto también se ve reflejado en la gráfica de residuos de la Figura 6.4.

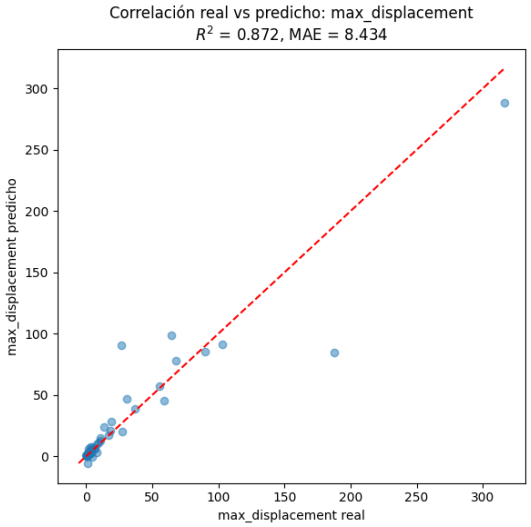


Figura 6.1. HGBR - Desplazamiento máximo.

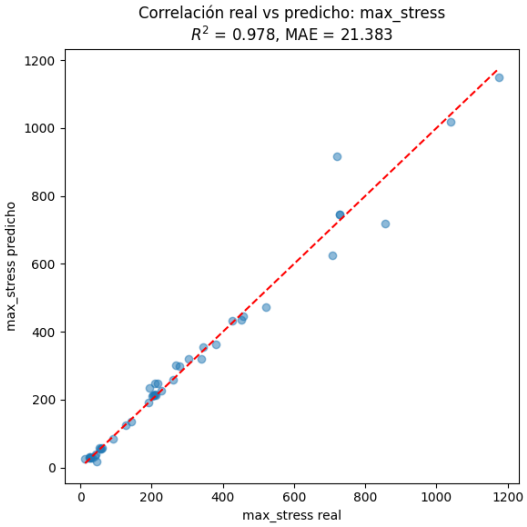


Figura 6.2. HGBR - Tensión máxima.

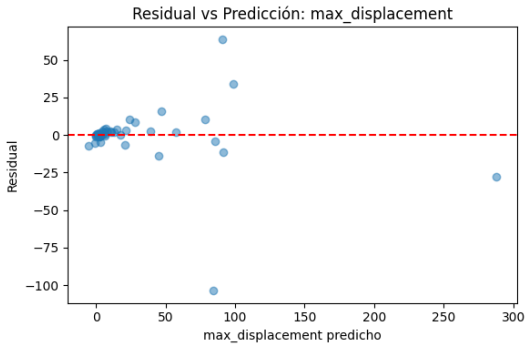


Figura 6.3. HGBR - Residuos desplazamiento máximo

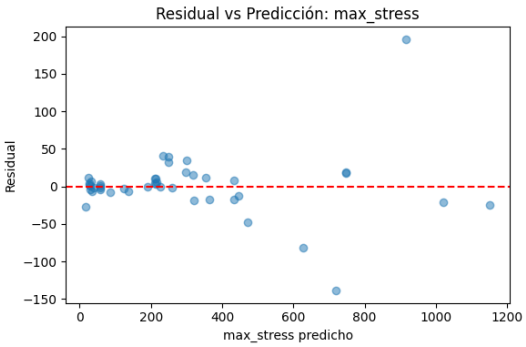


Figura 6.4. HGBR - Residuos tensión máxima

Resultados con la red neuronal En el mismo conjunto independiente, la red neuronal alcanzó $R^2 \approx 0{,}988$ y MAE $\approx 21{,}89$ MPa para tensión máxima (Figura 6.6), con dispersión similar a la de HGBR en el rango alto (Figura 6.7). En desplazamiento máximo el salto fue notable: $R^2 \approx 0{,}980$ y MAE $\approx 4{,}09$ mm (Figura 6.5), con una nube muy ceñida a la diagonal y residuales estrechos salvo algunos casos aislados de gran flecha (Figura 6.8).

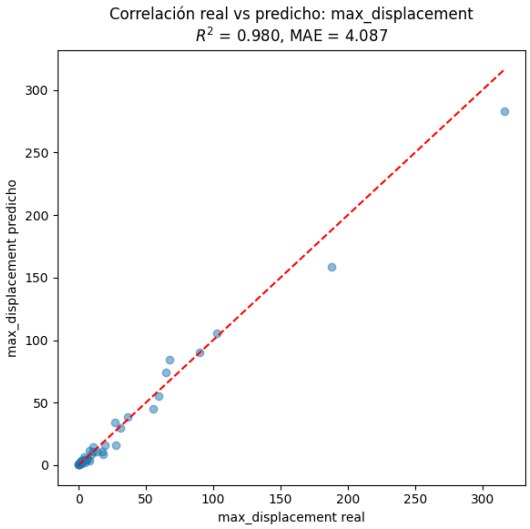


Figura 6.5. NN - Desplazamiento máximo

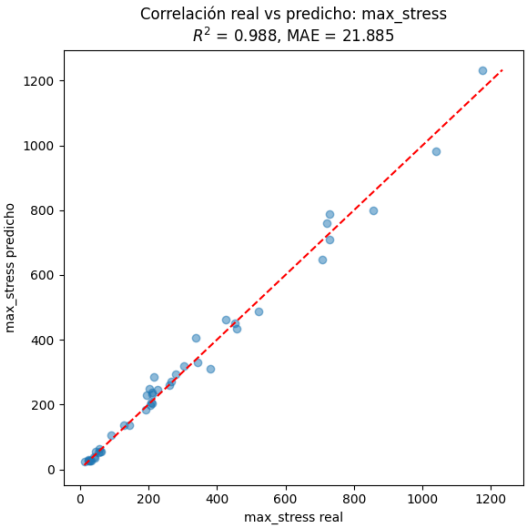


Figura 6.6. NN - Tensión máxima

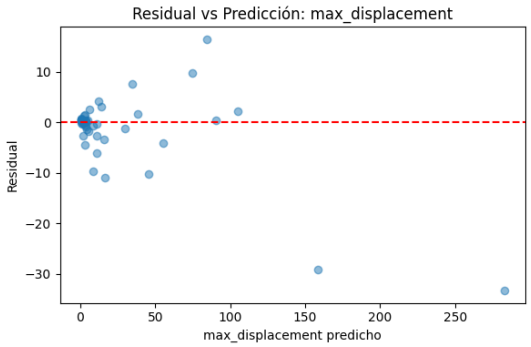


Figura 6.7. NN - Residuos desplazamiento máximo

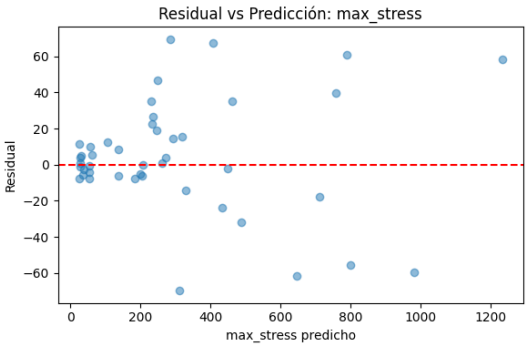


Figura 6.8. NN - Residuos tensión máxima

La Tabla 6.1 muestra un resumen de los resultados obtenidos para cada variable objetivo en los dos modelos.

Tabla 6.1. Métricas sobre el conjunto externo de 50 vigas no vistas

Modelo	Objetivo	R^2	MAE
HGBR	max_stress	0,978	21,38 MPa
HGBR	max_displacement	0,872	8,43 mm
Red neuronal	max_stress	0,988	21,89 MPa
Red neuronal	max_displacement	0,980	4,09 mm

Los diagramas de correlación confirman que ambos modelos capturan bien las tendencias globales. En tensión, los dos presentan un ajuste muy similar; la diferencia de MAE es marginal y dentro del ruido esperado por la propia discretización del problema y la variabilidad geométrica. En desplazamiento, la red neuronal mantiene la pendiente y la alineación con $y = x$ a lo largo de todo el rango, mientras que el HGBR muestra mayor dispersión en el extremo de grandes flechas. Los diagramas de residuales refuerzan esta conclusión: para HGBR aparecen colas más anchas en los casos con desplazamientos altos; en la red neuronal los residuales se distribuyen de forma más estrecha y aproximadamente centrada, con heterocedasticidad moderada en tensiones elevadas.

Estos resultados son coherentes con las métricas obtenidas durante el entrenamiento: el HGBR ya mostraba un rendimiento muy fuerte en la predicción de tensiones y un desempeño más limitado en desplazamientos cuando no se controlaba cuidadosamente la escala física de entrada. Tras el preprocesado y la incorporación de rasgos físico informados, la red neuronal aprovechó mejor la información de escala y linealidad local, especialmente para la flecha, donde el comportamiento suave y continuo favorece a arquitecturas densas con activaciones no lineales.

Para predicción de tensión máxima, HGBR y red neuronal ofrecen precisión equivalente en términos prácticos sobre datos no vistos. Para desplazamiento máximo, la red neuronal es claramente superior en este experimento independiente, con casi la mitad del MAE y un R^2 sensiblemente mayor. Con base en ello, una estrategia razonable es:

1. Emplear la red neuronal como modelo de referencia para desplazamientos;
2. Mantener el HGBR o la propia red neuronal para tensiones, indistintamente, dado su rendimiento similar;
3. Considerar un enfoque combinado o ensemblado si se desea robustez extra frente a casos límite, y complementar con estimación de incertidumbre (p. ej., pérdidas cuantílicas en HGB o *dropout* en inferencia para la red) cuando se trabaje cerca de los bordes del espacio de diseño.

En todos los casos, la calidad de la predicción seguirá estando condicionada por la cobertura del dataset en las regiones de mayor flexibilidad y por la correcta aplicación del preprocesado físico informado que alinea las entradas con las escalas L, EI y su interacción con las cargas.

En conjunto, la validación externa respalda la solidez de los modelos: las métricas son coherentes con las obtenidas en el conjunto de prueba y el patrón de puntos se alinea con la recta identidad en ambos objetivos. Con base en estos resultados, se consideran los modelos suficientemente estables para su integración en la interfaz Alabeam y su uso en escenarios de predicción rápida. Como trabajos futuros, puede reforzarse la cobertura en regiones de cargas elevadas y longitudes efectivas extremas, e incorporar diagnósticos adicionales de residuos estratificados por sección y tipo de apoyo.

6.2 Comparación de resultados: caso de estudio real

En este apartado se presenta un análisis comparativo entre las predicciones obtenidas con la herramienta Alabeam y el análisis numérico realizado con HyperMesh/Optistruct para una viga de validación adicional, generada desde cero en HyperMesh. El objetivo es evaluar la precisión de los modelos de aprendizaje frente a una simulación MEF detallada, y discutir aspectos prácticos como tiempo de ejecución, requisitos de licencia y la curva de aprendizaje necesaria para cada enfoque.

6.2.1. Descripción del caso de estudio

La viga analizada se muestra en la Figura 6.9 y tiene las siguientes características:

- **Longitud:** $L = 1775$ mm.
- **Material:** aluminio (propiedades empleadas en HyperMesh/OptiStruct según la configuración del modelo).
- **Condiciones de contorno:** empotrada en el extremo izquierdo y libre en el extremo derecho (cantiléver).
- **Sección:** perfil tipo *I* — dimensiones mostradas en la Figura 6.10 (captura adjunta).
- **Carga:** carga aplicada en eje Y (hacia abajo) situada a 1579.75 mm desde el extremo empotrado.

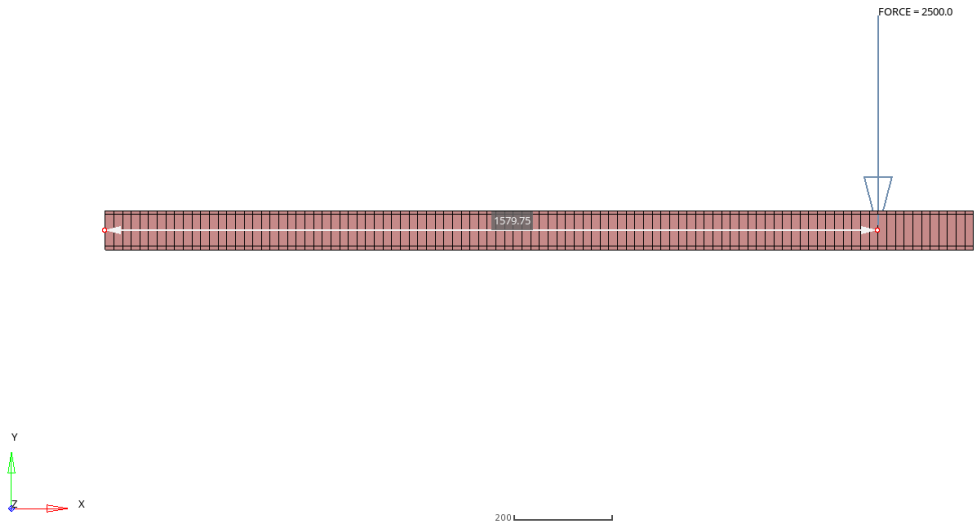


Figura 6.9. Modelo de la viga utilizada

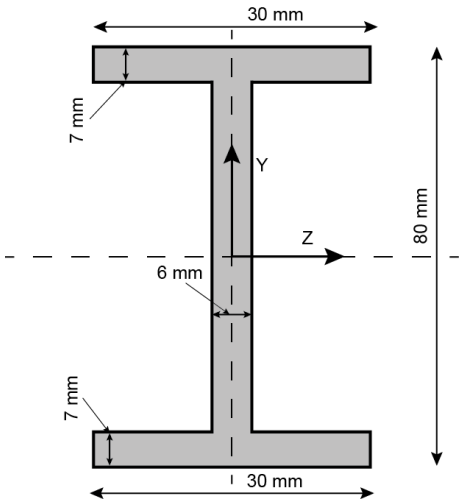


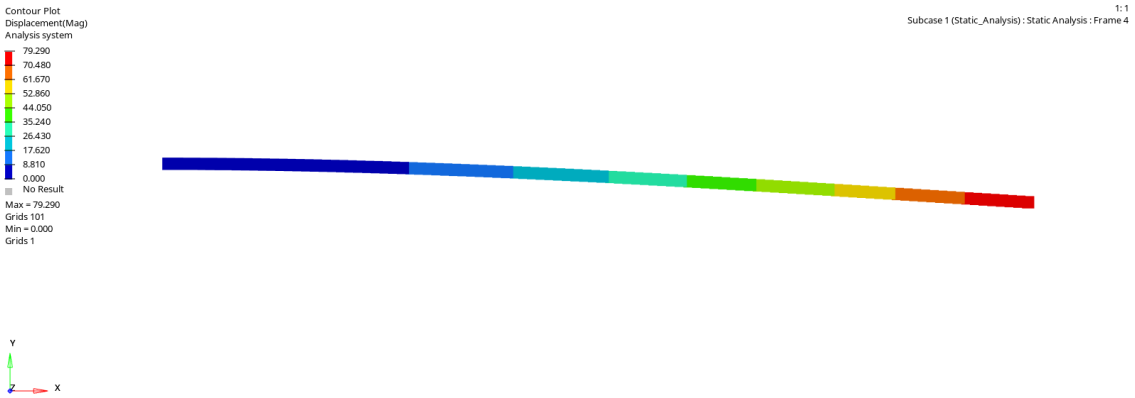
Figura 6.10. Dimensiones de la sección I empleada en el modelo

6.2.2. Resultados de la simulación FE (HyperMesh / OptiStruct)

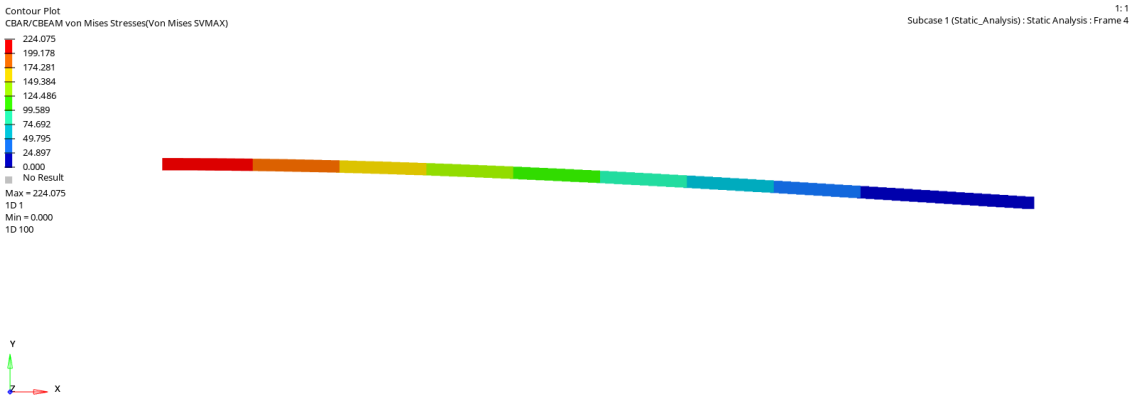
La solución obtenida mediante OptiStruct para la configuración descrita arroja los siguientes valores de referencia:

- **Desplazamiento máximo:** $u_{\text{máx}}^{\text{FE}} = 79.29$ mm.
- **Tensión máxima (Von Mises):** $\sigma_{\text{máx}}^{\text{FE}} = 224.075$ MPa.

En la Figura 6.11 se presentan los resultados obtenidos en HyperView, donde la imagen (a) corresponde al campo de desplazamientos y la imagen (b) muestra la distribución de tensiones de Von Mises.



(a) Desplazamientos



(b) Tensiones von Mises

Figura 6.11. Resultados FE: desplazamientos y tensiones para la viga de 1775 mm.

6.3 Resultados de Alabeam

A continuación se presentan los valores predichos por la aplicación Alabeam, basada en el modelo de red neuronal, para la misma viga analizada mediante el modelo de elementos finitos en HyperMesh/OptiStruct:

- **Desplazamiento máximo (Alabeam):** $u_{\text{máx}}^{\text{AB}} = 82.57$ mm
- **Tensión máxima (AlaBeam, Von Mises):** $\sigma_{\text{máx}}^{\text{AB}} = 248.32$ MPa

6.3.1. Métricas de comparación

Para cuantificar la diferencia entre la simulación FE (HyperMesh/OptiStruct) y la predicción de Alabeam se emplean las siguientes métricas:

$$\text{Error absoluto (EA)} = |x^{\text{FE}} - x^{\text{AB}}|$$

$$\text{Error relativo (ER)} = \frac{|x^{\text{FE}} - x^{\text{AB}}|}{|x^{\text{FE}}|} \times 100\,\%$$

donde x es la magnitud de interés ($u_{\text{máx}}$ o $\sigma_{\text{máx}}$).

En la Tabla 6.2 se muestran los valores de ambos análisis con las muestras de el error obtenido entre la simulación hecha por MEF y con Alabeam.

Tabla 6.2. Comparación entre los resultados del análisis FE (HyperMesh/OptiStruct) y la predicción de Alabeam (modelo de red neuronal) para la viga de validación.

Magnitud	FE (HyperMesh)	Alabeam	Error absoluto	Error relativo
Desplazamiento máximo [mm]	79.29	82.57	3.28	4.13 %
Tensión máxima (Von Mises) [MPa]	224.08	248.32	24.25	10.82 %

La Figura 6.12 presenta un gráfico de barras que permite comparar de forma visual las diferencias entre ambos análisis.

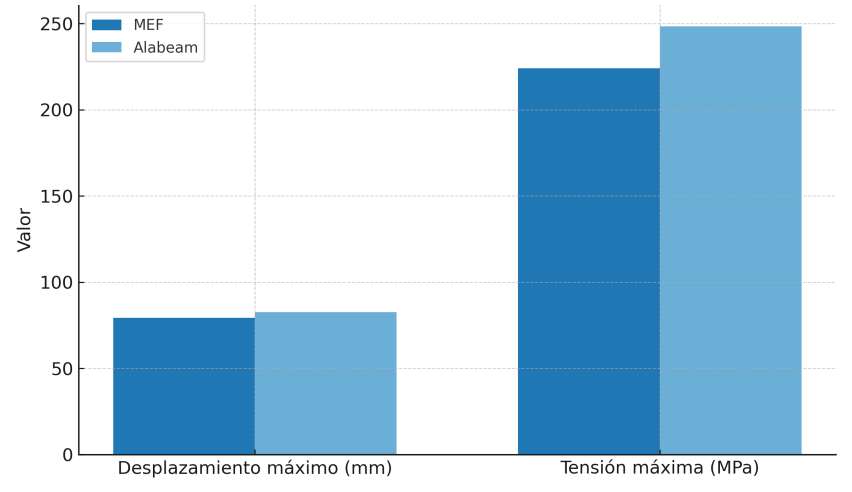


Figura 6.12. Comparación entre MEF (HyperMesh) y Alabeam

6.3.2. Discusión

Los resultados muestran una excelente correlación entre la predicción de Alabeam y el análisis realizado mediante el modelo de elementos finitos. El error relativo en desplazamiento máximo es del 4.13 %, mientras que para la tensión máxima alcanza un 10.82 %. Estos valores son plenamente aceptables considerando que el modelo de red neuronal ha sido entrenado con datos sintéticos generados bajo una gran variedad de geometrías, materiales y condiciones de contorno.

Además, Alabeam ofrece la ventaja de obtener resultados en menos de un segundo, sin necesidad de disponer de licencias de software comercial ni conocimientos avanzados de modelado FE, lo que supone una alternativa rápida y eficiente para análisis preliminares o tareas de diseño conceptual.

6.3.3. Comparativa práctica: tiempo, licencia y curva de aprendizaje

Tiempo

- **HyperMesh + OptiStruct:** preparación del modelo (creación de la sección I, mallado, definición de propiedades, materiales, condiciones de contorno y carga, y ejecución) suele requerir desde varios minutos hasta horas según la complejidad y la pericia del usuario. La ejecución del *solver* en un modelo de viga 1D es rápida (segundos), pero la preparación y postprocesado consumen la mayor parte del tiempo.
- **AlaBeam:** la predicción es prácticamente instantánea (orden de milisegundos a segundos) una vez introducidos los parámetros en la GUI y aplicado el preprocesado automático.

Licencia

- **HyperMesh/OptiStruct:** software comercial con coste de licencia, típicamente usado por empresas de CAE. Implica inversión económica y, en muchos casos, acceso institucional.
- **AlaBeam:** herramienta propia basada en modelos entrenados; su despliegue (por ejemplo en Streamlit Cloud) puede ser gratuito o con coste reducido. Sin embargo, la validez de las predicciones depende del dataset y no sustituye la necesidad del software CAE para análisis de alta fidelidad.

Conocimientos requeridos

- **HyperMesh/OptiStruct:** requiere conocimientos en MEF, preparación de mallas, elección de elementos y criterios de convergencia; además del manejo del software (HyperMesh/HyperView) y de la interpretación correcta de resultados (tensiones, factores de seguridad, etc.).
- **AlaBeam:** orientado a usuarios con conocimientos de ingeniería básica que deseen estimaciones rápidas. Sin embargo, interpretar resultados críticos (p. ej. tensiones cercanas al límite elástico) aún requiere juicio técnico.

6.3.4. Conclusión del caso de estudio

El caso de validación realizado demuestra la capacidad de la herramienta Alabeam para reproducir con elevada precisión los resultados de un análisis estructural mediante elementos finitos. A pesar de tratarse de un modelo simplificado basado en datos sintéticos, las discrepancias respecto al resultado de referencia obtenido con HyperMesh/OptiStruct se mantienen por debajo del 11 % en ambos parámetros analizados, lo que evidencia la robustez del enfoque propuesto.

Además, la predicción mediante Alabeam se obtiene de forma instantánea y sin necesidad de disponer de licencias comerciales ni de conocimientos avanzados en modelado FEM.

Por contra partida, un análisis con HyperMesh requiere una configuración detallada del modelo, tiempos de ejecución mucho mayores y un coste computacional y económico significativamente superior.

En conclusión, este caso de estudio ilustra el potencial de las técnicas de aprendizaje automático para agilizar el análisis estructural, abriendo la puerta a su aplicación en fases tempranas de diseño, optimización o evaluación de configuraciones paramétricas sin depender de un software de simulación convencional.

Capítulo 7. CONCLUSIONES

7.1 Conclusiones del trabajo

El objetivo principal de este trabajo fue demostrar la viabilidad de emplear técnicas de *Machine Learning* para aproximar resultados de análisis estructural obtenidos por el Método de los Elementos Finitos. Para ello se diseñó y validó un flujo completo que integra generación de datos por simulación (HyperMesh + OptiStruct), extracción automática de resultados mediante PyNastran, un preprocesado con *feature engineering* físico-informado y el entrenamiento de modelos supervisados (HGBR y redes neuronales).

Los principales resultados y conclusiones son los siguientes:

- **Generación y datos.** Se generó un conjunto sintético de 4058 modelos mediante simulaciones MEF, empleando una partición de datos $70\,\%/30\,\%$ (2 839 casos para entrenamiento y 1 218 para prueba). El pipeline automático garantiza trazabilidad y reproducibilidad entre la definición geométrica (HyperMesh), la ejecución (OptiStruct) y la extracción de magnitudes de interés (desplazamiento y tensión máximos).
- **Preprocesado.** El *feature engineering* aplicado (ecuaciones de vigas, variables geométricas y codificaciones categóricas) mejora la representatividad física de las entradas y facilita el aprendizaje, reduciendo la necesidad de arquitectura excesivamente compleja.
- **Rendimiento de HGBR.** El modelo *Histogram Gradient Boosting Regressor* mostró un comportamiento robusto y competitivo:
  - *Desplazamiento máximo:* $R^2 = 0.843$, $\mathrm{MAE} = 11.6189$ mm, $\mathrm{MedAE} = 2.6783$ mm.
  - *Tensión máxima:* $R^2 = 0.960$, $\mathrm{MAE} = 36.7916$ MPa, $\mathrm{MedAE} = 14.1722$ MPa.

  Estos valores muestran que, con un conjunto de features bien diseñado, los métodos de boosting son capaces de capturar las no linealidades relevantes con alta estabilidad.
- **Rendimiento de las redes neuronales.** Se entrenaron varias arquitecturas y se compararon con HGBR:
  - Arquitectura `ultimate` para *desplazamiento*: $R^2 = 0.8686$, $\mathrm{MAE} = 10.2020$ mm, $\mathrm{MedAE} = 1.9112$ mm, $\mathrm{RMSE} = 26.4861$ mm, $\mathrm{MAPE} = 113.82\,\%$ (n_test $= 1218$).
  - Arquitectura `simple` para *tensión*: $R^2 = 0.92897$, $\mathrm{MAE} = 54.4107$ MPa, $\mathrm{MedAE} = 25.9554$ MPa, $\mathrm{RMSE} = 107.2986$ MPa, $\mathrm{MAPE} = 26.21\,\%$ (n_test $= 1218$).

  En resumen: la red neuronal `ultimate` supera a HGBR en la predicción de desplazamientos (mejores R^2, MAE y MedAE), mientras que para tensiones la arquitectura más simple ofreció el mejor compromiso entre ajuste y control de errores extremos (RMSE competitivo).

- **Selección final de modelos.** Atendiendo tanto a métricas como al análisis de dispersión y residuos, se adoptaron las siguientes decisiones:
  - *Desplazamiento máximo:* modelo seleccionado — red neuronal `ultimate`.
  - *Tensión máxima:* modelo seleccionado — red neuronal `simple` (por su mejor control de errores extremos en comparación con arquitecturas más profundas).
- **Despliegue y usabilidad.** La aplicación Alabeam fue implementada en Streamlit y desplegada en la capa gratuita de Streamlit Cloud (URL pública). Esto facilita la validación y difusión, aunque el plan gratuito impone limitaciones operativas (instancias que entran en reposo tras inactividad).

Limitaciones principales

- Los datos son sintéticos: aunque permiten explorar un dominio amplio, la validez fuera del espacio muestreado (extrapolación) es limitada hasta contar con validación experimental.
- Criterios como el MAPE pueden resultar engañosos cuando existen valores reales cercanos a cero; por ello se ha priorizado R^2, MAE, MedAE y RMSE en la evaluación.
- El estudio se ha acotado a vigas 2D/simple, por lo que la aplicabilidad a placas, sólidos o ensamblajes requiere trabajo adicional.
- No se han incluido (en este trabajo) estimaciones sistemáticas de incertidumbre en las predicciones (intervalos de confianza), imprescindible para usos industriales críticos.

Aportaciones y valor añadido

- Demostración práctica y reproducible de un flujo completo: desde la generación automática de modelos MEF hasta el despliegue de una aplicación interactiva con modelos predictivos.
- Comparativa cuantitativa entre un método de boosting (HGBR) y redes neuronales, con justificación de selección por objetivo.
- Implementación de un preprocesado con base física que mejora la interpretabilidad y la estabilidad del aprendizaje.
- Código y modelos empaquetados para facilitar su reutilización y ampliación.

En conclusión, este trabajo demuestra que el *Machine Learning* puede ser una herramienta eficaz y eficiente para acelerar el análisis estructural basado en MEF, ofreciendo predicciones inmediatas con una precisión adecuada para muchas tareas de diseño exploratorio y cribado. Las limitaciones señaladas delimitan un camino claro de mejora que permitirá, con la incorporación de validación experimental y mayor complejidad en los datos, trasladar la metodología a escenarios industriales reales.

7.2 Conclusiones personales

En este trabajo he logrado demostrar la viabilidad de integrar simulación numérica y aprendizaje automático para la predicción rápida del comportamiento estructural de vigas. El proceso me ha permitido familiarizarme con la generación automatizada de modelos MEF (HyperMesh / OptiStruct), la extracción de resultados con PyNastran y el entrenamiento de modelos predictivos (HGBR y redes neuronales). Personalmente, el mayor aprendizaje ha sido comprender la importancia del *feature engineering* físico-informado para obtener modelos robustos y generalizables. Este proyecto me ha proporcionado competencias prácticas y metodológicas que pienso aplicar en futuros proyectos.

Capítulo 8. FUTURAS LÍNEAS DE TRABAJO

El presente trabajo ha demostrado la viabilidad de aplicar técnicas de *Machine Learning* al análisis estructural, utilizando datos generados mediante simulaciones por el Método de los Elementos Finitos (MEF). A pesar de que el estudio se ha centrado en un caso muy concreto el comportamiento de vigas 2D bajo distintas condiciones geométricas, materiales y de contorno, los resultados obtenidos abren múltiples vías de desarrollo y mejora, tanto a nivel metodológico como de aplicación práctica.

8.1 Ampliación del dominio estructural

Una primera línea de trabajo consiste en extender el enfoque actual hacia otros tipos de elementos estructurales más complejos:

- Vigas tridimensionales y pórticos: incorporar grados de libertad en las tres direcciones espaciales, así como los efectos de flexión, torsión y cortante combinados.
- Placas y láminas: emplear elementos de tipo `CQUAD4` o `CTRIA3` para modelar estructuras bidimensionales y estudiar tensiones en membrana y flexión.
- Sólidos 3D: extender el entrenamiento a modelos volumétricos, permitiendo predecir campos de tensiones o desplazamientos locales en piezas reales.

8.2 Incremento de la complejidad del modelo de datos

El dataset actual se ha generado mediante modelos sintéticos con hipótesis lineales y condiciones de contorno simples. Futuros desarrollos podrían considerar:

- Materiales no lineales (plásticos, compuestos o viscoelásticos).
- Cargas dinámicas o térmicas, explorando el comportamiento temporal de la estructura.
- Interacciones entre elementos, permitiendo representar estructuras completas y ensamblajes.

8.3 Mejoras en el modelado y entrenamiento

En el ámbito del aprendizaje automático, existen múltiples vías de mejora:

- Optimización de hiperparámetros mediante *grid search* o algoritmos bayesianos.
- Modelos híbridos Físico–Informados (*Physics-Informed Neural Networks, PINNs*) que integren directamente las ecuaciones del MEF en la función de pérdida.
- Redes neuronales convolucionales o gráficas (GNNs), capaces de aprender directamente de la malla de elementos finitos sin necesidad de un preprocesado manual.
- Aumento del dataset mediante técnicas de generación automática y validación cruzada más exhaustiva.

8.4 Despliegue y usabilidad de la herramienta

La aplicación Alabeam, desarrollada en *Streamlit*, representa un primer paso hacia la integración de modelos predictivos en herramientas accesibles para ingenieros. En el futuro podrían implementarse:

- Un entorno web más robusto, desplegado en servidor propio o en contenedor Docker, que evite las limitaciones de la versión gratuita.
- Un módulo de exportación automática a software CAD/CAE.
- Una interfaz más avanzada, que permita personalizar geometrías, condiciones de contorno y materiales, además de mostrar resultados visuales del campo de desplazamientos o tensiones.

8.5 Aplicación industrial y validación experimental

Finalmente, sería deseable realizar una validación experimental o comparativa frente a resultados reales o modelos de alta fidelidad, evaluando la precisión de los modelos de aprendizaje automático en contextos industriales. Esta fase permitiría medir el impacto real de la metodología propuesta y sentar las bases para su incorporación en el flujo de trabajo de diseño estructural asistido por inteligencia artificial.

Conclusión

En conjunto, las líneas expuestas apuntan a la consolidación de un paradigma emergente: la fusión entre el análisis numérico tradicional y las técnicas de aprendizaje automático. La capacidad de los modelos de *Machine Learning* para aproximar comportamientos estructurales complejos de manera instantánea constituye una herramienta de enorme potencial para acelerar procesos de simulación, optimización y diseño en ingeniería estructural.

Bibliografía

[1] O. C. Zienkiewicz, R. L. Taylor y J. Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals*, 6.ª ed. Butterworth-Heinemann, 2005, ISBN: 9780750663205.

[2] M. A. Bessa, R. Bostanabad, Z. Liu et al., «A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality,» *Computer Methods in Applied Mechanics and Engineering*, vol. 320, págs. 633-667, 2017, ISSN: 0045-7825. DOI: 10.1016/j.cma.2017.03.037. dirección: https://www.sciencedirect.com/science/article/pii/S0045782516314803.

[3] S. P. Timoshenko y J. N. Goodier, *Theory of Elasticity*, inglés, 3.ª ed. New York: McGraw-Hill, 1970.

[4] K.-J. Bathe, *Finite Element Procedures*, inglés. Upper Saddle River, NJ: Prentice Hall, 1996.

[5] J. M. Gere y B. J. Goodno, *Mechanics of Materials*, inglés, 8.ª ed. Stamford, CT: Cengage Learning, 2012.

[6] T. Hastie, R. Tibshirani y J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, inglés, 2.ª ed. New York: Springer, 2009.

[7] S. Arlot y A. Celisse, «A survey of cross-validation procedures for model selection,» inglés, *Statistics Surveys*, vol. 4, págs. 40-79, 2010. DOI: 10.1214/09-SS054.

[8] I. Goodfellow, Y. Bengio y A. Courville, *Deep Learning*, inglés. Cambridge, MA: MIT Press, 2016. dirección: https://www.deeplearningbook.org/.

[9] G. Ke, Q. Meng, T. Finley et al., «LightGBM: A Highly Efficient Gradient Boosting Decision Tree,» inglés, en *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, págs. 3146-3154.

[10] scikit-learn developers. «Histogram-based Gradient Boosting (User Guide).» Implementation details, early stopping, categorical handling. (2025), dirección: https://scikit-learn.org/stable/modules/ensemble.html#histogram-based-gradient-boosting (visitado 06-10-2025).

[11] B. Etim, A. Al-Ghosoun, J. Renno, M. Seaid y M. S. Mohamed, «Machine Learning-Based Modeling for Structural Engineering: A Comprehensive Survey and Applications Overview,» *Buildings*, vol. 14, n.º 11, pág. 3515, 2024. DOI: 10.3390/buildings14113515. dirección: https://www.mdpi.com/2075-5309/14/11/3515.

[12] L. Mainini y K. Willcox, «A surrogate modeling approach to support real-time structural assessment and decision-making,» en *AIAA SciTech*, AIAA, 2015. dirección: https://kiwi.oden.utexas.edu/papers/Dynamic-data-driven-surrogate-Mainini-Willcox.pdf.

[13] J. Kudela et al., «Recent advances and applications of surrogate models for finite element-based computations,» *Soft Computing*, vol. 27, págs. 7337-7361, 2022. DOI: 10.1007/s00500-022-07362-8. dirección: https://doi.org/10.1007/s00500-022-07362-8.

[14] L. Liang, M. Liu, C. Martin y W. Sun, «A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis,» *Journal of The Royal Society Interface*, vol. 15, n.º 138, pág. 20 170 844, 2018. DOI: 10.1098/rsif.2017.0844. dirección: https://pmc.ncbi.nlm.nih.gov/articles/PMC5805990/.

[15] scikit-learn developers. «Features in Histogram Gradient Boosting Trees.» Accessed: 2025-10-05. (2025), dirección: https://scikit-learn.org/stable/auto_examples/ensemble/plot_hgbt_regression.html.

[16] scikit-learn developers. «Time-related feature engineering.» Accessed: 2025-10-05. (2025), dirección: https://scikit-learn.org/stable/auto_examples/applications/plot_cyclical_feature_engineering.html.

[17] B. Zhen, C. Xu y L. Ouyang, «Physics-Informed Neural Networks-Based Wide-Range Parameter Displacement Inference for Euler–Bernoulli Beams on Foundations Under a Moving Load Using Sparse Local Measurements,» *Applied Sciences*, vol. 15, n.º 11, pág. 6213, 2025. DOI: 10.3390/app15116213. dirección: https://www.mdpi.com/2076-3417/15/11/6213.

[18] P. Vurtur Badarinath, M. Chierichetti y F. Davoudi Kakhki, «A Machine Learning Approach as a Surrogate for a Finite Element Analysis: Status of Research and Application to One Dimensional Systems,» *Sensors*, vol. 21, n.º 5, pág. 1654, 2021. DOI: 10.3390/s21051654. dirección: https://www.mdpi.com/1424-8220/21/5/1654.

[19] Altair Engineering Inc. «Altair® PhysicsAI™ Geometric Deep Learning.» (2023), dirección: https://altair.com/physicsai (visitado 05-10-2025).

[20] Altair Engineering Inc. «Altair Releases Altair® HyperWorks® 2025.» (2025), dirección: https://altair.com/newsroom/news-releases/altair-releases-altair-hyperworks-2025 (visitado 05-10-2025).

[21] Altair Engineering Inc. «AI-Powered Engineering.» (2025), dirección: https://altair.com/ai-powered-engineering (visitado 05-10-2025).

[22] Altair Engineering Inc. «Altair HyperWorks 2025.1 Overview.» (2025), dirección: https://altair.com/hyperworks-2025 (visitado 05-10-2025).

[23] Altair Engineering Inc. «PhysicsAI — HyperWorks 2025.1 Help.» (2025), dirección: https://help.altair.com/hwdesktop/hwx/topics/reference/extensions/physicsai_r.htm (visitado 05-10-2025).

[24] Altair Engineering Inc. «Altair PhysicsAI 2025 Release Notes.» (2025), dirección: https://help.altair.com/hwcfdsolvers/altair_help/topics/release_notes/rn_2025_physicsai_r.htm (visitado 05-10-2025).

[25] T. Insight. «Altair Physics AI — AI in Simulation.» (2025), dirección: https://www.trueinsight.io/physics-ai (visitado 05-10-2025).

[26] T. Insight. «Step-by-Step Walkthrough: Using Altair PhysicsAI.» (2024), dirección: https://www.trueinsight.io/blog/physicsai-walkthrough (visitado 05-10-2025).

[27] D. López. «Código MEF/ML (release v1.0).» (2025), dirección: https://github.com/LPZdani/AlaBeam-ficheros-generacion-FEM-y-ML.git (visitado 11-10-2025).

[28] PebSteel. «What is a foundation beam steel structure? Principles of steel beam foundation layout.» (2024), dirección: https://pebsteel.com/en/what-is-a-foundation-beam-steel-structure-principles-of-steel-beam-foundation-layout/ (visitado 08-08-2024).

[29] D. López. «Alabeam (release v1.0).» (2025), dirección: https://github.com/LPZdani/AlaBeam.git (visitado 11-10-2025).

Apéndice A. Presupuesto y costes estimados

Criterios de estimación. El precio del equipo se ha estimado a partir del coste de mercado de un portátil con especificaciones equivalentes (Intel Core i9, 32 GB de RAM y SSD de 1 TB), tomando como referencia una configuración media de gama alta y redondeando a una cifra conservadora de 2 200 €. Las licencias de Altair HyperMesh/OptiStruct se han considerado a coste cero por tratarse de licencias académicas de estudiante. Las horas de ingeniería (120 h) se han calculado por descomposición de tareas:

- Definición del espacio de diseño y scripting para generación automática de modelos.
- Preparación y lanzamiento de simulaciones en OptiStruct.
- Postprocesado con PyNastran y depuración de datos.
- Entrenamiento, validación iterativa y análisis de errores.
- Integración y pruebas de la interfaz AlaBeam.

La imputación económica de las horas de ingeniería se han valorado con una tarifa orientativa de 35 €/h. Se considera una incertidumbre razonable ($\pm 15 - 20\,\%$) asociada a variaciones de mercado. Se contempla la presentación del trabajo en un congreso académico, incluyendo inscripción, viaje, alojamiento y dietas. El importe mostrado (875 €) es orientativo y representa un evento de ámbito europeo. Esta partida queda sujeta a disponibilidad de financiación institucional y puede reducirse en caso de colaboración de algún departamento.

La siguiente tabla resume los costes estimados del desarrollo del proyecto.

Tabla A.1. Presupuesto y costes estimados. La asistencia a congreso se contempla como partida opcional

Concepto	**Detalle**	**Coste**
Portátil (equipo de cómputo)	Intel Core i9, 32 GB RAM, SSD 1 TB	2.200 €
Licencia HyperMesh (estudiante)	Uso académico	0 €
Python + librerías	sk-learn, TF, PyNastran, Streamlit	0 €
Horas de cálculo	∼24 h (local)	*n/a*
Horas de ingeniería	Desarrollo y validación (120 h × 35 €/h)	4.200 €
Asistencia a congreso (opcional)	Inscripción, viaje, alojamiento y dietas	875 €
Total		**7.275 €**

Notas.

- Si el equipo ya estaba disponible, el coste puede imputarse como amortización proporcional a la duración del TFM, en cuyo caso el importe directo sería menor.
- No se imputan costes energéticos al tratarse de ejecución local en entorno académico.

Apéndice B. Repositorio y código fuente

Este apéndice recopila la organización del repositorio y las rutas de los códigos (scripts) principales para las herramientas diseñadas para el desarrollo de los modelos y la interfaz gráfica Alabeam.

B.1 Herramientas para generar los modelos

Este repositorio recoge todos los *scripts* (herramientas) que se han creado para realizar el entrenamiento de los modelos, desde la generación del dataset hasta la predicción de los mismos. El repositorio se puede encontrar en la referencia [27]

```
1.Generador_Modelos/
        generador_HM.tcl
        generador_casos.py
2.Preprocessing/
        Extraccion_OP2.py
        preprocessing.py
3.Train/
        Dataset_V01_total.csv
        HGBR_training.py
        NN_training.py
4.Predict/
        HGBR_predict.py
        NN_predict.py
config.py
requirements.txt
```

Notas de uso

- Sistema de unidades mm, N y MPa en todos los scripts.
- Los scripts que generan datos fijan una semilla aleatoria fijaga en *config.py*

Referencia rápida a scripts

En la Tabla B.1 se muestran los ficheros del repositorio y la descripción de cada uno de ellos.

Tabla B.1. Descripción del código del repositorio

ruta	propósito
`1.Generador_Modelos/generador_casos.py`	genera el CSV maestro de combinaciones
`1.Generador_Modelos/generador_HM.tcl`	crea modelos .hm y exporta .fem a partir del CSV
`2.Preprocessing/Extraccion_OP2.py`	lee OP2 y agrega desplazamiento y tensión máximos
`2.Preprocessing/preprocessing.py`	cálculo de rasgos físico-informados y limpieza
`3.Train/HGBR_training.py`	entrenamiento y validación del modelo HGBR
`3.Train/NN_training.py`	entrenamiento y validación de la red neuronal
`4.Predict/HGBR_predict.py`	predicción con el modelo HGBR final
`4.Predict/NN_predict.py`	predicción con la red neuronal final
`config.py`	constantes compartidas y utilidades

B.2 Repositorio de la aplicación Alabeam

A continuación se documenta la estructura y contenidos del repositorio de la aplicación AlaBeam, la cual integra la interfaz gráfica desarrollada en *Streamlit* con los modelos de predicción (HGBR y red neuronal) y las utilidades de preprocesado. Este repositorio se puede encontrar en la referencia [29]

Estructura del repositorio

```
Alabeam/
        Alabeam_logo.png
        README.md
        alabeam.py       % App principal (Streamlit)
        config.py        % Configuración general
        models/
                model_max_displacement_HGB.joblib
                model_max_stress_HGB.joblib
        models_neural/
                feature_columns_neuralnet.csv
                model_max_displacement_neuralnet.keras
                model_max_stress_neuralnet.keras
                scaler_displacement_neuralnet.joblib
                scaler_stress_neuralnet.joblib
        preprocessing.py  % Preprocesado
        requirements.txt
```

Notas generales de uso

- Todos los scripts y modelos trabajan con el sistema de unidades mm, N y MPa.
- La semilla aleatoria para generación reproducible de datos está definida en `config.py`.
- Los modelos binarios se encuentran en `models/` (HGBR) y `models_neural/` (red neuronal y escaladores).
- Para evitar problemas de compatibilidad, se recomienda crear un entorno virtual (conda o venv) e instalar las dependencias con `pip install -r requirements.txt`.

Instrucciones de ejecución rápida

1. Crear un entorno e instalar dependencias.
2. Ejecutar la aplicación localmente:

   ```
   streamlit run alabeam.py
   ```

3. Acceder en el navegador a `http://localhost:8501` (o a la URL de Streamlit Cloud si está desplegado).

Referencia rápida a scripts y ficheros

Tabla B.2. Índice rápido de ficheros principales del repositorio Alabeam

Ruta / fichero	**Propósito / descripción**
`alabeam.py`	Script principal de la interfaz gráfica (Streamlit). Gestiona la carga de modelos, el preprocesado de entradas y la visualización de resultados.
`config.py`	Archivo de configuración con constantes globales (unidades, semilla, rutas a modelos, nombres de columnas, parámetros de normalización).
`preprocessing.py`	Implementa las transformaciones de *feature engineering* usadas tanto en entrenamiento como en predicción (escalares, variables derivadas de la geometría de vigas, encoding de secciones/materiales).
`models/model_max_displacement_HGB.joblib`	Modelo HGBR entrenado para `max_displacement`.
`models/model_max_stress_HGB.joblib`	Modelo HGBR entrenado para `max_stress`.
`models_neural/feature_columns_neuralnet.csv`	Lista de columnas/orden que requiere la red neuronal para las entradas.
`models_neural/model_max_displacement_neuralnet.keras`	Red neuronal entrenada para desplazamiento máximo (modelo Keras).
`models_neural/model_max_stress_neuralnet.keras`	Red neuronal entrenada para tensión máxima (modelo Keras).
`models_neural/scaler_*.joblib`	Objetos `scaler` usados para normalizar/denormalizar entradas y salidas en la predicción.
`requirements.txt`	Dependencias del proyecto (Streamlit, scikit-learn, joblib, tensorflow/keras, pandas, numpy, etc.).

Consideraciones de reproducibilidad y despliegue

- Mantener sincronizados `feature_columns_neuralnet.csv` y el orden de columnas esperado por los modelos. Cualquier cambio en el preprocesado debe versionarse conjuntamente con los modelos.
- Para despliegue público se ha usado Streamlit Cloud (plan gratuito). En este entorno la aplicación puede entrar en reposo tras periodos de inactividad; consulte la sección de despliegue del TFM para más detalles.
- Incluir en el repositorio un ejemplo mínimo de entrada (CSV de ejemplo) y tests unitarios básicos para las transformaciones críticas.

Apéndice C. Estructura de un fichero .fem (Optistruct & HyperMesh)

Este apéndice describe la estructura y las principales cartas (*bulk data entries*) de un modelo de viga resuelto con OptiStruct a partir de un fichero `.fem` exportado desde HyperMesh. Se usa como referencia el caso `Beam_TUBE_1000mm_simple_clamped_Titanium.fem`. La información mostrada en este apéndice ha sido extraída de la documentación oficial de Altair (ver referencia [22]).

Estructura general del .fem

Un fichero `.fem` de OptiStruct sigue la organización clásica tipo NASTRAN:

- Case Control: define el tipo de análisis, casos de carga (*subcase*) y salidas.
- Bulk Data: contiene la malla (nodos, elementos), propiedades, materiales, cargas y restricciones.

La información de los modelos se define mediante tarjetas (*cards*) preestablecidas por *OptiStruct*, las cuales permiten describir una amplia variedad de configuraciones y tipos de análisis.

En el modelo analizado, el *Case Control* es:

```
SUBCASE        1
LABEL Static_Analysis
ANALYSIS STATICS
SPC  =       1
LOAD =       2
DISPLACEMENT(OUTPUT2,) = ALL
SPCFORCE(,OUTPUT2,,,) = ALL
STRESS(OP2,ALL) = ALL
```

Esto define un análisis estático (SUBCASE 1) con:

- Conjunto de restricciones `SPC=1`.
- Conjunto de cargas `LOAD=2`.
- Solicitud de salidas: desplazamientos, fuerzas de SPC y tensiones a ficheros binarios OP2/OUTPUT2.

Malla: nodos (`GRID`) y elementos (`CBEAM`)

La malla es unidimensional a lo largo del eje X, con 101 nodos (0 mm a 1000 mm) que discretizan la viga en 100 elementos:

```
GRID           1        0.0     0.0     0.0
...
GRID         101     1000.0     0.0     0.0
```

Los elementos empleados son CBEAM (elementos viga con sección y propiedad asociadas). Cada tarjeta `CBEAM` referencia una propiedad (PID), dos nodos extremos y un vector de orientación local (v) que fija el eje local de la sección:

```
CBEAM          1       1       1       20.0     0.0     1.0
...
CBEAM        100       1     100     1010.0     0.0     1.0
```

Notas: (1) `PID=1` vincula cada elemento a la propiedad de viga (sección y material). (2) El vector de orientación no necesita ser unitario; sólo define la dirección del eje local (p. ej., hacia Z) para rotar la sección.

Propiedad de viga: `PBEAML` (sección tubular)

La propiedad de los elementos `CBEAM` se define con `PBEAML` usando una sección *TUBE* (tubo circular), indicando geometría:

```
PBEAML         1       1            TUBE                              +
+       73.56   60.27   0.0
```

donde `PID=1` y `MID=1` (material asociado). Para `TUBE`, los parámetros corresponden a radio exterior y radio interior (en mm), por lo que aquí se modela un tubo con $D_{\text{ext}} \approx 73.56$ mm y $D_{\text{int}} \approx 60.27$ mm.

Material: `MAT1` (isótropo lineal)

El material isótropo se define con `MAT1` (módulo elástico E, coeficiente de Poisson ν, densidad ρ, etc.):

```
MAT1           1  112000.0        0.32    4.5-6
```

En el caso de esta viga se emplea titanio cuyas propiedades son: $E \approx 112$ GPa, $\nu = 0.32$, $\rho \approx 4.5 \times 10^{-6}$ kg/mm^3 (densidad ~ 4500 kg/m^3). *Nota de unidades:* el modelo emplea millímetros (mm) y Newton (N), por lo que E se expresa en MPa.

Condiciones de contorno: `SPC` (simple y empotrado)

Las restricciones `SPC` fijan grados de libertad (DOF) en nodos específicos. En el modelo:

```
SPC            1       1     123      0.0
SPC            1     101  123456      0.0
```

- En el nodo 1 se fijan traslaciones $(1, 2, 3)$ y se dejan libres las rotaciones $(4, 5, 6)$ representando un apoyo simple.
- En el nodo 101 se fijan todos los DOF $(1–6)$ reflejando un empotramiento.

Cargas: `LOAD/MOMENT`

El caso de carga referencia `LOAD=2`, que agrupa las tarjetas de carga del *bulk data*. En este ejemplo se aplica un momento nodal:

```
MOMENT          2       51        0  1.0    0.0    0.0    -3.537+7
```

Este momento actúa en el nodo 51 (aproximadamente en el centro de la viga), con magnitud y dirección dadas por los campos de la tarjeta (momento concentrado en un sistema cartesiano global, `CID=0`). Esta carga produce un estado de flexión representativo para la validación.

Resumen del modelo empleado

- **Longitud:** 1000 mm (101 nodos, 100 elementos `CBEAM`).
- **Sección:** tubular (`PBEAML/TUBE`) con $r_{\text{ext}} = 73.56$ mm y $r_{\text{int}} = 60.27$ mm.
- **Material:** `MAT1` (Titanio) con $E = 112$ GPa, $\nu = 0.32$, $\rho = 4.5 \times 10^{-6}$ kg/mm^3.
- **Apoyos:** nodo 1 $\rightarrow$ apoyo simple (123); nodo 101 $\rightarrow$ empotramiento (123456).
- **Carga:** `MOMENT` en nodo 51, agrupado en `LOAD=2`.
- **Análisis:** estático lineal (`ANALYSIS STATICS`), con salidas de desplazamientos y tensiones.

Buenas prácticas al generar .fem desde HyperMesh

- Verificar unidades coherentes (mm–N–MPa) para `GRID`, `PBEAML`, `MAT1` y cargas.
- Asegurar una orientación de viga (`CBEAM` vector) estable: evita alinearla con el eje de la viga para no degenerar el sistema local.
- Comprobar que PID y MID están correctamente enlazados (`CBEAM` $\rightarrow$ `PBEAML` $\rightarrow$ `MAT1`).
- Documentar casos de carga y SPC con identificadores únicos, especialmente cuando se generen lotes sintéticos.
- Solicitar salidas (`DISPLACEMENT`, `STRESS`, etc.) acordes con el postproceso (por ejemplo, extracción con PyNastran).