UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

MASTER UNIVERSITARIO EN ANALISIS DE DATOS MASIVOS (BIG DATA)

TRABAJO FIN DE MÁSTER

Predicción de tendencias en series temporales financieras mediante algoritmos de machine learning y técnicas avanzadas de feature engineering

Maksym Sheptyuk Riabchynskiy

Dirigido por

Dr. Sergio Iglesias Pérez

CURSO 2024-2025

TÍTULO: Predicción de tendencias en series temporales financieras mediante algoritmos de machine learning y técnicas avanzadas de feature engineering

AUTOR: Maksym Sheptyuk Riabchynskiy

TITULACIÓN: MASTER UNIVERSITARIO EN ANALISIS DE DATOS MASIVOS (BIG DATA)

DIRECTOR/ES DEL PROYECTO: Dr. Sergio Iglesias Pérez

FECHA: septiembre de 2025

RESUMEN

Este trabajo aborda la predicción de tendencias en series temporales financieras mediante algoritmos de machine learning y técnicas avanzadas de feature engineering. El objetivo principal fue diseñar y validar un sistema automatizado capaz de generar señales de inversión a partir de un amplio conjunto de indicadores técnicos, optimizados mediante un proceso de selección de características, empleando algoritmos de evolutivos y modelos ML.

Se desarrolló un sistema de adquisición de datos integral que abarca desde la descarga y preparación de datos históricos hasta la construcción de un dataset enriquecido con más de 250 indicadores técnicos y sus variantes. Posteriormente, se aplicó un algoritmo genético bajo un enfoque selección utilizando modelo envolvente para identificar subconjuntos de variables con mayor capacidad predictiva, y se incorporó un ajuste sistemático de hiperparámetros que permitió optimizar rendimiento de los modelos ML. El núcleo principal del sistema fue XGBoost Regressor, validado a través de un extenso backtesting.

La simulación de una cartera diversificada de 25 activos entre 2020 y 2024 arrojó un ROI acumulado del 74 % y un CAGR del 11,68 %, con una tasa media de acierto del 61 % y caídas de capital acotadas incluso en entornos adversos como 2022. Además, se observó que la definición de la variable objetivo influye decisivamente en la rentabilidad, alcanzando configuraciones con CAGR próximos al 19 %.

Finalmente, la comparación con modelos alternativos como MLP y ARIMA, entrenados con series temporales de precios y volumen mostró que no existen métodos universalmente superiores: el rendimiento depende de diversos factores como: preprocesamiento y modelado de datos, arquitectura y configuración de modelo, cantidad y variedad de datos, configuración de hirparametros de modelos y mucho más. Esto refuerza la importancia de un enfoque flexible y adaptativo.

Palabras clave:

Machine learning, Predicción de tendencias financieras, Trading cuantitativo, Feature engineering, Algoritmos genéticos, Análisis predictivo.

ABSTRACT

This work addresses the prediction of financial time series trends using machine learning algorithms and advanced feature engineering techniques. The main goal was to design and validate an automated system capable of generating investment signals from a wide set of technical indicators, optimized through a feature selection process based on evolutionary algorithms.

An end-to-end data acquisition and processing system was developed, covering the download and preprocessing of historical data from multiple stocks and the construction of an enriched dataset with over 250 technical indicators and their variants. In the next stage, a genetic algorithm was applied under a wrapper approach to identify subsets of features with higher predictive power, and a systematic hyperparameter tuning process was incorporated to optimize the performance of the ML models. The core predictive model was the XGBoost Regressor, validated through extensive backtesting.

Backtesting on a diversified portfolio of 25 assets between 2020 and 2024 yielded an accumulated ROI of 74% and a compound annual growth rate (CAGR) of 11.68%, with an average accuracy of 61% and controlled drawdowns, even under adverse conditions such as 2022. Moreover, the definition of the target variable proved decisive for profitability, with some configurations reaching CAGR values close to 19%.

Finally, the comparison with alternative models such as MLP and ARIMA, trained with price and volume time series, showed that no method is universally superior; performance depends on factors such as data preprocessing, model architecture, dataset diversity, and hyperparameter settings. This highlights the importance of a flexible and adaptive approach.

Keywords:

Machine learning, Stock trend prediction, Quantitative trading, Feature engineering, Genetic algorithms, Predictive analytics

AGRADECIMIENTOS

Quiero expresar mi más sincero agradecimiento a mi tutor Dr. Sergio Iglesias Pérez, por su orientación y apoyo constante a lo largo de esta enriquecedora aventura académica que ha representado el desarrollo de este proyecto. Su guía ha sido fundamental para mantener el rumbo y alcanzar los objetivos planteados.

Extiendo también mi gratitud a todos los profesores del máster, cuyas clases me permitieron adquirir conocimientos imprescindibles y valiosos que han sido la base para poder avanzar con éxito en este trabajo.

Finalmente, quiero dedicar un reconocimiento especial a mi familia, por su apoyo incondicional y ánimo en cada etapa de este camino. Su confianza y respaldo emocional han sido el motor que me ha permitido perseverar y culminar este reto académico con ilusión y determinación.

“Los modelos son aproximaciones. Todos son erróneos, pero algunos son útiles.”

— George Edward Pelham Box (1919–2013)

George E. P. Box fue un estadístico británico que trabajó en control de calidad, análisis de series temporales, diseño de experimentos e inferencia bayesiana, considerado una de las mentes más brillantes de la estadística del siglo XX. Fuente: Wikipedia

TABLA RESUMEN

	DATOS
Nombre y apellidos:	Maksym Sheptyuk Riabchynskiy
Título del proyecto:	Predicción de tendencias en series temporales financieras mediante algoritmos de machine learning y técnicas avanzadas de feature engineering
Directores del proyecto:	Dr. Sergio Iglesias Pérez
El proyecto se ha realizado en colaboración de una empresa o a petición de una empresa:	NO
El proyecto ha implementado un producto: (esta entrada se puede marcar junto a la siguiente)	NO
El proyecto ha consistido en el desarrollo de una investigación o innovación: (esta entrada se puede marcar junto a la anterior)	SÍ
Objetivo general del proyecto:	Diseñar y validar un sistema predictivo capaz de anticipar tendencias en series temporales financieras mediante técnicas avanzadas de feature engineering y algoritmos de machine learning, evaluando su eficacia a través de simulaciones de backtesting sobre datos reales.

Índice

Índice de Figuras

Índice de Tablas

Capítulo 1. RESUMEN DE PROYECTO

1.1 Contexto

Los mercados financieros modernos operan bajo condiciones de alta incertidumbre, volatilidad y no linealidad, lo que dificulta considerablemente su análisis predictivo. Durante décadas, los analistas e inversores han recurrido a herramientas del análisis técnico como el RSI, MACD, medias móviles exponenciales, bandas de Bollinger y otros indicadores, para inferir comportamientos futuros de precios a partir de patrones históricos. No obstante, la eficacia de estas herramientas depende en gran medida del criterio subjetivo del analista, lo que introduce un alto grado de variabilidad en los resultados. Además, la presencia de sesgos cognitivos y emocionales en el proceso de toma de decisiones ha sido ampliamente documentada, tal como destaca Elder en su obra *Trading for a Living* [1].

A medida que los mercados se digitalizan y la disponibilidad de datos históricos aumenta, surge la necesidad de aplicar métodos automatizados que reduzcan la dependencia del juicio humano. El aprendizaje automático (*machine learning*) se presenta como una alternativa poderosa para extraer patrones ocultos en grandes volúmenes de datos y realizar predicciones más objetivas, replicables y ajustadas a contextos específicos. En esta línea, el presente proyecto se justifica en la creciente demanda de soluciones que integren análisis técnico con modelos de predicción automatizados, capaces de seleccionar y combinar indicadores de forma dinámica en función de su relevancia para distintos activos y ventanas temporales.

1.2 Planteamiento del problema

El problema central abordado en este trabajo radica en la falta de sistemas predictivos automatizados que integren indicadores técnicos con técnicas modernas de modelado basadas en datos. Muchos enfoques actuales emplean modelos como ARIMA, LSTM o redes neuronales profundas (DNN) aplicadas directamente sobre series de precios, ignorando la valiosa información contenida en los indicadores derivados. Al no realizar una selección óptima de variables, estos modelos pueden sufrir de sobreajuste, baja interpretabilidad o rendimiento inconsistente.

Así, la pregunta motriz que guía este proyecto es la siguiente: ¿puede un sistema híbrido, que automatice la selección de indicadores técnicos relevantes y los utilice como entrada en modelos de *machine learning*, mejorar la capacidad predictiva respecto a los métodos clásicos basados exclusivamente en precios? Resolver esta cuestión implica explorar no solo la precisión de las predicciones, sino también su estabilidad en distintos contextos de mercado, su aplicabilidad práctica y su capacidad para emitir señales claras de compra o venta en horizontes de 1 a 15 días.

1.3 Objetivos del proyecto

El objetivo principal de este trabajo es diseñar e implementar un sistema automatizado para la predicción de tendencias en mercados bursátiles, mediante el uso de algoritmos de aprendizaje automático combinados con técnicas avanzadas de ingeniería de características. El sistema busca identificar patrones y generar señales de compra y venta a corto plazo (horizonte de 1 a 15 días), utilizando tanto indicadores técnicos clásicos como variables derivadas, estadísticas de comportamiento y transformaciones no lineales de los datos.

Este proyecto propone una solución funcional que automatiza desde la adquisición y procesamiento de datos hasta la selección óptima de variables predictivas y el entrenamiento de modelos robustos. Su desarrollo permitirá evaluar la efectividad de un enfoque híbrido frente a métodos tradicionales basados únicamente en precios históricos, aportando mayor objetividad, repetibilidad y precisión a las decisiones de trading. Así, se pretende reducir el margen de error humano y mejorar la rentabilidad potencial de estrategias de inversión.

1.4 Resultados obtenidos

Se ha logrado construir un sistema fiable, reproducible y consistente, capaz de generar señales de inversión a partir de datos financieros reales. La simulación de la operativa sobre datos históricos (backtesting) demostró que el sistema alcanza beneficios sostenidos con riesgo controlado en el periodo 2020–2024 (véase 4.6 Resultados del proyecto)

Conforme a la línea base definida con horizonte de predicción de 5 días, los resultados globales de la cartera mostraron un ROI (Retorno de Inversión) acumulado del **+74 %**, con un CAGR (Tasa de Crecimiento Anual Compuesta) del **11,68 %**, una tasa media de aciertos del **61 %** y caídas de capital anuales acotadas (Tabla 16, Figura 49). Sin embargo, al modificar la construcción de la **variable objetivo** hacia horizontes más largos se obtuvieron rendimientos claramente superiores; por ejemplo, la configuración con horizonte de 8 días y ventana de 8 días el sistema propuesto alcanzó un CAGR cercano al **19 %**, evidenciando la influencia decisiva del horizonte en la rentabilidad del sistema.

En la comparativa final de resultados (Tabla 20) se validó la eficiencia del sistema propuesto basado en el modelado de indicadores y dataset, EDA, transformación automática, XGBoost Regressor, algoritmo genético y ajuste de hiperparámetros— dentro del flujo general de proyecto descrito en la Figura 7.

En conjunto, el sistema se mostró capaz de generar **beneficios consistentes y estables,** incluso en periodos de adversidad, ya que, en 2022, pese a la fuerte volatilidad de los mercados y las caídas de los principales índices bursátiles, la estrategia logró limitar las pérdidas y mantener la preservación del capital, sin necesidad de reentrenamientos frecuentes ni una supervisión intensiva.

1.5 Estructura de la memoria

La memoria se organiza en 8 capítulos.

- En Capítulo 1 se presenta un resumen general del proyecto.
- El Capítulo 2 expone los antecedentes y el planteamiento del problema.
- Capítulo 3 expone el detalle de los objetivos del trabajo.
- En Capítulo 4 se describe el desarrollo de la solución propuesta y los resultados.
- El Capítulo 5 presenta discusión sobre resultados obtenidos.
- El Capítulo 6 contiene las conclusiones.
- En 0 se proponen y se describen líneas futuras de trabajo.
- El Capítulo 8 detalla las referencias utilizadas.
- En Capítulo 9 se adjuntan los anexos.

Capítulo 2. ANTECEDENTES / ESTADO DEL ARTE

2.1 Estado del arte

En los últimos años, la aplicación de algoritmos de *machine learning* (ML) al análisis de mercados financieros ha crecido exponencialmente, motivada por la necesidad de gestionar la alta complejidad y no linealidad de las series temporales bursátiles. Una de las estrategias más recurrentes es el uso de indicadores técnicos como características para alimentar modelos supervisados, buscando identificar patrones predictivos en ventanas temporales de corto o medio plazo.

Estudios recientes han demostrado que la calidad de los indicadores técnicos y la selección apropiada de variables son factores clave para mejorar la precisión en la predicción de movimientos del mercado. *Ji et al.* [5] proponen un esquema adaptativo de selección de características que mejora sustancialmente el rendimiento del modelo Random Forest al transformar y seleccionar indicadores mediante técnicas de descomposición por wavelets.

Asimismo, *Zouaghia et al.* [9] desarrollaron un sistema híbrido que combina múltiples clasificadores (Random Forest, SVM, kNN, etc.) con reducción de dimensionalidad mediante PCA, demostrando que el ajuste de hiperparámetros y la selección de indicadores optimizados pueden aumentar la precisión predictiva.

El estudio de *Paramita* [6] compara tres métodos de selección de características: PCA, Information Gain y Recursive Feature Elimination (RFE), y concluye que RFE es el más eficaz para reducir la dimensionalidad sin pérdida de precisión.

Sagaceta-Mejía et al. [8] van más allá al aplicar una arquitectura de redes neuronales optimizada con un subconjunto reducido (5%) de indicadores seleccionados mediante análisis estadístico, mejorando la eficiencia computacional y la precisión predictiva en mercados emergentes.

Finalmente, *Moodi y Rafsanjani* [7] aplican métodos wrapper como Sequential Forward Selection y Sequential Backward Selection con múltiples estimadores de regresión para seleccionar indicadores óptimos a partir de más de 120 opciones.

En conjunto, la literatura muestra un creciente consenso sobre la necesidad de integrar métodos avanzados de selección de características junto con arquitecturas ML optimizadas para enfrentar la volatilidad y complejidad de los mercados bursátiles.

2.2 Contexto y justificación

El presente proyecto surge en el contexto de una creciente necesidad de automatización en el análisis de datos financieros, donde la volatilidad, la complejidad no lineal y la sensibilidad a eventos macroeconómicos dificultan la toma de decisiones basadas exclusivamente en criterios humanos o sistemas tradicionales de análisis técnico. El uso extensivo de indicadores técnicos en el análisis bursátil ha sido ampliamente adoptado, pero su interpretación manual sigue

expuesta a sesgos cognitivos, experiencia previa y componentes emocionales, lo que puede limitar la consistencia y efectividad de las decisiones de inversión.

Este proyecto propone una solución híbrida que combina modelos de aprendizaje automático con técnicas avanzadas de ingeniería de características para construir un sistema automatizado de predicción de tendencias bursátiles. Se enmarca en un enfoque aplicado y práctico, en el que se desarrolla un prototipo funcional que integra extracción de datos desde fuentes públicas, cálculo dinámico de indicadores técnicos, generación de señales binarias, y selección automática de características del dataset mediante algoritmos evolutivos.

El proyecto no se realiza en colaboración con una empresa específica, aunque su diseño y resultados están orientados a escenarios reales del sector financiero, siendo aplicables a plataformas de trading algorítmico o asistentes de inversión automatizados. En términos científicos, la investigación contribuye al campo del aprendizaje automático aplicado al análisis financiero, validando que la incorporación estructurada de características optimizadas mejora la capacidad predictiva frente a enfoques puramente basados en precios.

En este sentido, el proyecto aporta un marco reproducible para evaluar no solo modelos predictivos, sino también la relevancia de los indicadores seleccionados y su impacto en la rentabilidad simulada mediante *backtesting*. Se espera que los resultados puedan ser útiles para investigadores, desarrolladores de sistemas de trading y profesionales del análisis cuantitativo.

2.3 Planteamiento del problema

A partir del análisis del estado del arte, se identifica una brecha relevante entre los enfoques tradicionales de predicción bursátil y las necesidades prácticas de los operadores e inversores en mercados modernos. Aunque numerosos estudios han demostrado que el uso de indicadores técnicos como variables de entrada en modelos de aprendizaje automático mejora la precisión predictiva, existe una carencia de soluciones integrales que automaticen la selección de dichos indicadores en función del activo, contexto temporal y características de mercado.

Por otro lado, muchos modelos actuales se centran en el análisis directo de los precios sin aprovechar plenamente la información derivada de transformaciones técnicas o estadísticas. Esto puede conducir a modelos poco generalizables, sobre ajustados o con dificultades para emitir señales claras de trading.

La problemática se agrava al considerar que la mayoría de los enfoques no incluye un mecanismo sistemático para evaluar qué combinación de indicadores aporta valor predictivo real, ni para identificar los activos más predecibles según su comportamiento histórico o su estabilidad en distintos escenarios.

En consecuencia, este trabajo aborda la necesidad de diseñar un sistema que integre selección evolutiva de características, análisis de predictibilidad y entrenamiento de modelos robustos de predicción de tendencias. Este proyecto se plantea como una solución orientada al desarrollo práctico de estrategias de trading algorítmico basadas en datos históricos de mercados.

Capítulo 3. OBJETIVOS

3.1 Objetivos generales

El objetivo general de este proyecto es diseñar e implementar una aplicación integral de análisis y predicción bursátil, capaz de automatizar todo el flujo de trabajo necesario para la toma de decisiones operativas en mercados financieros.

El sistema desarrollado permitirá extraer datos históricos de precios desde fuentes públicas como Yahoo Finance, almacenarlos en fichero .csv y generar, de forma automática, conjuntos de datos enriquecidos mediante técnicas avanzadas de ingeniería. Estos conjuntos incluirán indicadores técnicos tradicionales (como RSI, MACD, OBV, Stochastic Oscillator), estadísticas temporales de corto plazo (como tendencia direccional, volatilidad sectorial, beta, retorno semanal o mensual, correlación con índices o materias primas como el oro), así como nuevas variables sintéticas generadas por expansión polinómica, codificación binaria de señales (compra/venta) y algoritmos de selección evolutiva.

El sistema integrará herramientas de análisis exploratorio, escalado y corrección de sesgos en los datos (mediante transformaciones como Yeo-Johnson, LogScaler o RobustScaler. A continuación, se llevará a cabo un proceso de selección de características guiado por el algoritmo genético *genetic wrapper* (selección envolvente mediante algoritmo genético), utilizando modelos base como redes neuronales multicapa MLP, XGBoost, Ridge.

Finalmente, el sistema entrenará distintos modelos predictivos (XGBoost, MLPs) sobre horizontes de predicción de 1 a 15 días, optimizando sus hiperparámetros y evaluando su rendimiento mediante validación cruzada y simulaciones de backtesting. El objetivo final es construir un sistema que no solo prediga la dirección futura de activo financiero, sino que también emita señales de trading interpretables y cuantificables, capaces de mejorar la rentabilidad de las decisiones de inversión frente a métodos tradicionales.

3.2 Objetivos específicos

Con el fin de alcanzar el objetivo general del proyecto, se establecen los siguientes objetivos específicos:

- Implementar un sistema automatizado para la descarga y almacenamiento estructurado de datos financieros desde Yahoo Finance.
- Calcular indicadores técnicos y estadísticas relevantes que permitan enriquecer los datos históricos con información predictiva.
- Generar nuevas características mediante técnicas de transformación y codificación binaria.
- Aplicar procesos de normalización, corrección de sesgos y reducción de dimensionalidad para optimizar el espacio de variables.

- Utilizar algoritmos evolutivos tipo wrapper para seleccionar las características más relevantes.
- Entrenar y validar modelos de predicción (XGBoost, MLP, Ridge) sobre distintos horizontes temporales.
- Realizar un análisis de rendimiento mediante backtesting y comparar con metodologías clásicas como ARIMA o MLP modelados a partir de series de precios.

3.3 Beneficios del proyecto

El presente proyecto aporta beneficios tanto desde el punto de vista técnico como práctico. A nivel técnico, proporciona una metodología estructurada y reproducible para el desarrollo de sistemas predictivos en entornos financieros, integrando técnicas de ingeniería de características, selección de características utilizando algoritmo genético, y entrenamiento de modelos avanzados. Este enfoque permite evaluar la efectividad de distintos modelos y configuraciones en la predicción de tendencias bursátiles, lo que representa un valor añadido en el análisis cuantitativo aplicado.

Desde el punto de vista práctico, el sistema desarrollado facilita la toma de decisiones operativas a corto plazo, reduciendo la dependencia del juicio humano y aumentando la objetividad en la generación de señales de compra y venta. Además, al automatizar tareas complejas como la adquisición de datos, el cálculo de indicadores, la selección de variables y el entrenamiento de modelos, se agiliza el ciclo de análisis y se incrementa la eficiencia del proceso.

El proyecto también ofrece valor como herramienta educativa y de experimentación para investigadores, estudiantes o profesionales interesados en la aplicación de machine learning al análisis de mercados financieros. Finalmente, su diseño modular y escalable permite su futura ampliación o adaptación a otros contextos, activos o estrategias de inversión.

Capítulo 4. DESARROLLO DEL PROYECTO

4.1 Planificación del proyecto

Se adoptó una metodología **Waterfall (cascada) secuencial**, adecuada a un alcance cerrado y a un **horizonte de 3 meses**. El proyecto avanzó fase a fase, sin iteraciones, y solo se permitieron solapes puntuales entre tareas independientes para optimizar la carga semanal.

Roles (definición):

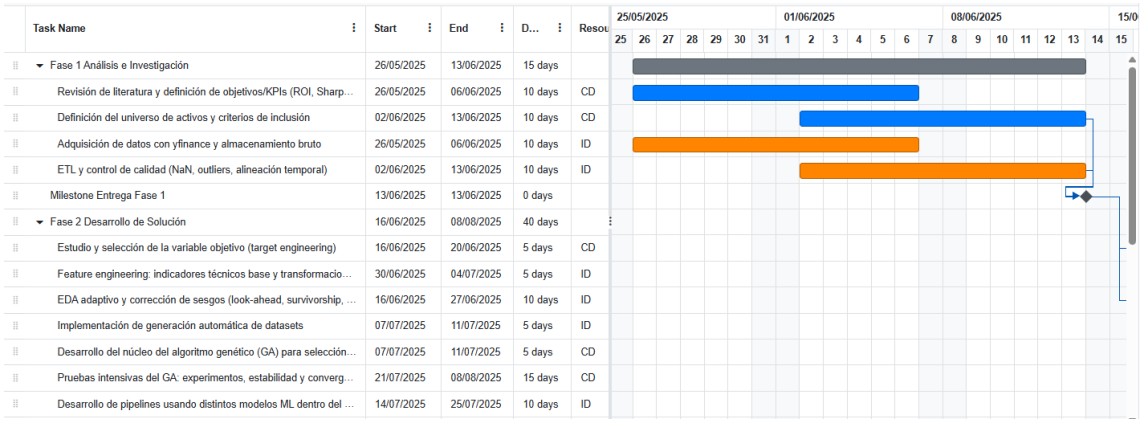
- **CD = Científico de Datos** (modelado, métricas, comparación de modelos).
- **ID = Ingeniero de Datos y, a la vez, Desarrollador ML Senior** (EDA de sesgos, ETL/pipelines, backtester, orquestación técnica).

Cada rol trabajó con **25 h/semana** (≈300 h por persona), sin exceder capacidad de presupuesto.

Estructura por fases y hitos (stage-gate):

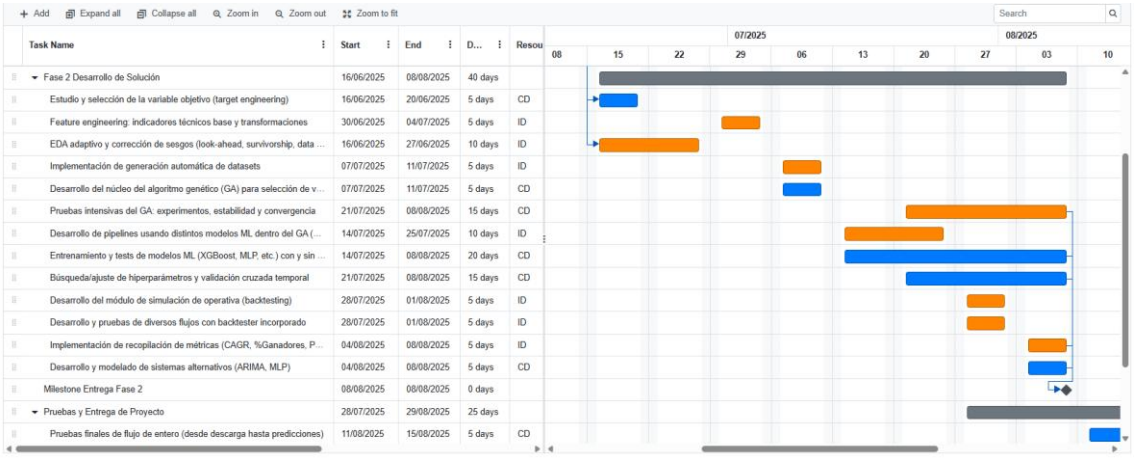
- **Fase 1** — Análisis e investigación (26/05 – 13/06)**:** revisión de literatura, delimitación del universo de activos**,** adquisición de datos**,** ETL y control de calidad.
- **Fase 2 —** Desarrollo y validación (16/06 – 08/08)**:** EDA adaptivo y corrección de sesgos, ingeniería de características y generación automática de datasets**,** núcleo del GA y sus pruebas**,** entrenamiento y validación de distintos modelos**,** desarrollo y pruebas de diseños alternativos (ARIMA, MLP sobre precios), backtesting y comparativas de rendimiento de modelos de cartera con los modelos tradicionales, comparativas de métricas, desarrollo de scripts de automatización GA, predicciones de modelos, etc.
- **Fase 3 —** Pruebas y cierre (11/08 – 29/08)**:** pruebas de flujo completo, refactorizaciones oportunas y pruebas finales, documentación, integración final de componentes y documentación.

La planificación fue **lineal por fases**, con entregables verificables en cada hito y **sin iteraciones**, combinando previsibilidad temporal con un uso eficiente de los roles **CD** e **ID (Desarrollador ML Senior)**. (véase Figura 1, Figura 2 y Figura 3)



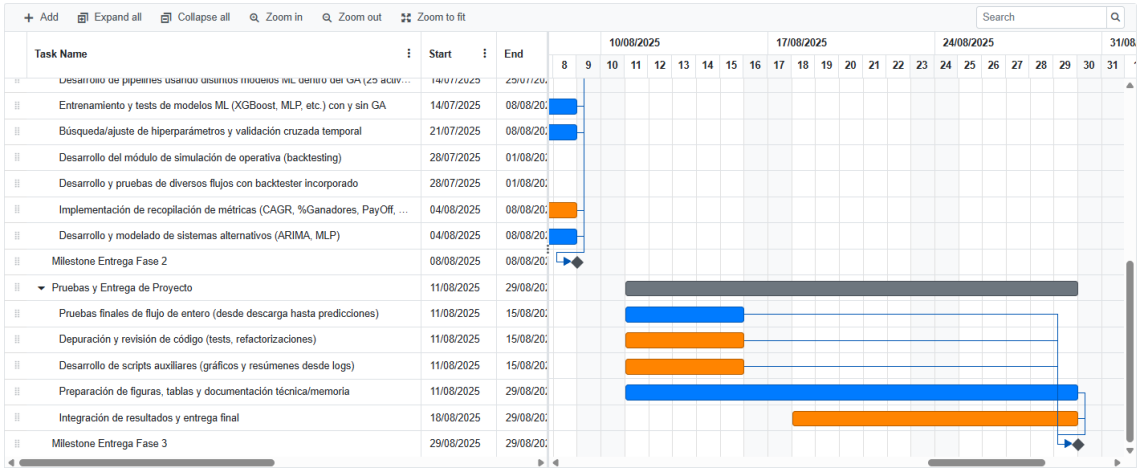
Task Name	Start	End	D...	Resou
Fase 1 Análisis e Investigación	26/05/2025	13/06/2025	15 days	
Revisión de literatura y definición de objetivos/KPIs (ROI, Sharp...	26/05/2025	06/06/2025	10 days	CD
Definición del universo de activos y criterios de inclusión	02/06/2025	13/06/2025	10 days	CD
Adquisición de datos con yfinance y almacenamiento bruto	26/05/2025	06/06/2025	10 days	ID
ETL y control de calidad (NaN, outliers, alineación temporal)	02/06/2025	13/06/2025	10 days	ID
Milestone Entrega Fase 1	13/06/2025	13/06/2025	0 days	
Fase 2 Desarrollo de Solución	16/06/2025	08/08/2025	40 days	
Estudio y selección de la variable objetivo (target engineering)	16/06/2025	20/06/2025	5 days	CD
Feature engineering: indicadores técnicos base y transformacio...	30/06/2025	04/07/2025	5 days	ID
EDA adaptivo y corrección de sesgos (look-ahead, survivorship, ...	16/06/2025	27/06/2025	10 days	ID
Implementación de generación automática de datasets	07/07/2025	11/07/2025	5 days	ID
Desarrollo del núcleo del algoritmo genético (GA) para selección...	07/07/2025	11/07/2025	5 days	CD
Pruebas intensivas del GA: experimentos, estabilidad y converg...	21/07/2025	08/08/2025	15 days	CD
Desarrollo de pipelines usando distintos modelos ML dentro del ...	14/07/2025	25/07/2025	10 days	ID

Figura 1 Planificación de proyecto, Fase 1 (Análisis e investigación)



Task Name	Start	End	D...	Resou
Fase 2 Desarrollo de Solución	16/06/2025	08/08/2025	40 days	
Estudio y selección de la variable objetivo (target engineering)	16/06/2025	20/06/2025	5 days	CD
Feature engineering: indicadores técnicos base y transformaciones	30/06/2025	04/07/2025	5 days	ID
EDA adaptivo y corrección de sesgos (look-ahead, survivorship, data ...	16/06/2025	27/06/2025	10 days	ID
Implementación de generación automática de datasets	07/07/2025	11/07/2025	5 days	ID
Desarrollo del núcleo del algoritmo genético (GA) para selección de v...	07/07/2025	11/07/2025	5 days	CD
Pruebas intensivas del GA: experimentos, estabilidad y convergencia	21/07/2025	08/08/2025	15 days	CD
Desarrollo de pipelines usando distintos modelos ML dentro del GA (...	14/07/2025	25/07/2025	10 days	ID
Entrenamiento y tests de modelos ML (XGBoost, MLP, etc.) con y sin ...	14/07/2025	08/08/2025	20 days	CD
Búsqueda/ajuste de hiperparámetros y validación cruzada temporal	21/07/2025	08/08/2025	15 days	CD
Desarrollo del módulo de simulación de operativa (backtesting)	28/07/2025	01/08/2025	5 days	ID
Desarrollo y pruebas de diversos flujos con backtester incorporado	28/07/2025	01/08/2025	5 days	ID
Implementación de recopilación de métricas (CAGR, %Ganadores, P...	04/08/2025	08/08/2025	5 days	ID
Desarrollo y modelado de sistemas alternativos (ARIMA, MLP)	04/08/2025	08/08/2025	5 days	CD
Milestone Entrega Fase 2	08/08/2025	08/08/2025	0 days	
Pruebas y Entrega de Proyecto	28/07/2025	29/08/2025	25 days	
Pruebas finales de flujo de entero (desde descarga hasta predicciones)	11/08/2025	15/08/2025	5 days	CD

Figura 2. Planificación de proyecto, Fase 2 (Desarrollo)



Task Name	Start	End
Entrenamiento y tests de modelos ML (XGBoost, MLP, etc.) con y sin GA	14/07/2025	08/08/20
Búsqueda/ajuste de hiperparámetros y validación cruzada temporal	21/07/2025	08/08/20
Desarrollo del módulo de simulación de operativa (backtesting)	28/07/2025	01/08/20
Desarrollo y pruebas de diversos flujos con backtester incorporado	28/07/2025	01/08/20
Implementación de recopilación de métricas (CAGR, %Ganadores, PayOff, ...	04/08/2025	08/08/20
Desarrollo y modelado de sistemas alternativos (ARIMA, MLP)	04/08/2025	08/08/20
Milestone Entrega Fase 2	08/08/2025	08/08/20
Pruebas y Entrega de Proyecto	11/08/2025	29/08/20
Pruebas finales de flujo de entero (desde descarga hasta predicciones)	11/08/2025	15/08/20
Depuración y revisión de código (tests, refactorizaciones)	11/08/2025	15/08/20
Desarrollo de scripts auxiliares (gráficos y resúmenes desde logs)	11/08/2025	15/08/20
Preparación de figuras, tablas y documentación técnica/memoria	11/08/2025	29/08/20
Integración de resultados y entrega final	18/08/2025	29/08/20
Milestone Entrega Fase 3	29/08/2025	29/08/20

Figura 3. Planificación de proyecto, Fase 3 (Pruebas y entrega de proyecto)

4.2 Descripción de la solución, metodologías y herramientas empleadas

4.2.1 Los mercados financieros: complejidad e incertidumbre

Los mercados financieros constituyen uno de los sistemas más complejos y dinámicos de la economía global. Cada día, millones de inversores y operadores participan en la compra y venta de activos, motivados por factores tan diversos como resultados empresariales, noticias económicas, decisiones políticas, expectativas de futuro e incluso emociones colectivas. Esta interacción constante genera una **volatilidad inherente**, que se traduce en movimientos impredecibles de precios a corto plazo y cambios abruptos en la tendencia de los activos [2].

Figura 4. Operadores en la Bolsa de Nueva York (NYSE). Fuente: Forbes

En palabras de Elder, *"la psicología de masas y las emociones colectivas juegan un papel crucial en la formación de tendencias y burbujas, dificultando aún más la anticipación de los movimientos del mercado"* [1].

Además, la información disponible para los agentes de mercado se actualiza de forma continua y en tiempo real, lo que provoca que las condiciones que afectan a los precios puedan variar en cuestión de minutos o segundos. Factores como eventos geopolíticos inesperados, anuncios de resultados o simplemente rumores pueden desencadenar reacciones en cadena difíciles de anticipar y modelizar.

Según la teoría de mercados eficientes propuesta por Fama, toda la información relevante está reflejada en los precios, lo que añade un grado más de incertidumbre y aleatoriedad al comportamiento bursátil [10]. Diversos estudios recientes han puesto de manifiesto la alta

complejidad y no linealidad de las series temporales bursátiles, lo que dificulta la extracción de patrones consistentes mediante técnicas tradicionales [5].

A pesar de este aparente caos, la búsqueda de patrones repetitivos y señales de comportamiento en los precios ha sido una constante a lo largo de la historia del análisis financiero. Sin embargo, la presencia de **incertidumbre estructural** y la influencia de factores externos, como destacan Peng et al. [2] y Ji et al. [5], junto con la complejidad de la **psicología de masas** analizada por Elder [1], hacen que la predicción precisa y sistemática de los movimientos de mercado sea un reto de gran envergadura. Además, la teoría de mercados eficientes propuesta por Fama [10] refuerza la idea de que estas dificultades afectan tanto a inversores individuales como a instituciones especializadas.

4.2.2 Indicadores técnicos y su uso tradicional

El análisis técnico es una de las herramientas más extendidas en el estudio de los mercados financieros. Su objetivo principal es identificar patrones, tendencias y puntos de giro mediante el análisis de datos históricos de precios y volúmenes. Para ello, se utilizan una variedad de **indicadores matemáticos** desarrollados a lo largo de décadas de práctica bursátil. (véase Figura 5)

Figura 5 Indicadores técnicos RSI y MACD aplicados a precios de Apple. Fuente: TradingView

Entre los indicadores más empleados destacan:

- **Índice de Fuerza Relativa (RSI, Relative Strength Index):** Oscilador que mide la velocidad y el cambio de los movimientos de precio, facilitando la identificación de zonas de sobrecompra y sobreventa. Se calcula según la siguiente fórmula:

$$\mathrm{RSI} = 100 - \frac{100}{1 + RS}$$

  Donde RS es la media de subidas dividida entre la media de bajadas en los últimos N periodos. Un valor de RSI superior a 70 suele indicar sobrecompra, mientras que valores por debajo de 30 apuntan a sobreventa.

- **MACD (Moving Average Convergence Divergence):** Herramienta que compara dos medias móviles exponenciales (de diferente periodo) para detectar cambios en la tendencia del precio. El MACD se define como:

$$\mathrm{MACD} = \mathrm{EMA}_{12}(\mathrm{Precio}) - \mathrm{EMA}_{26}(\mathrm{Precio})$$

  La señal se obtiene mediante la media móvil exponencial del MACD (habitualmente de 9 periodos). Los cruces del MACD con su señal se interpretan como señales de compra o venta.

- **Medias móviles exponenciales (EMA):** Promedios ponderados que otorgan más peso a los precios recientes. El cruce del precio con la EMA20, por ejemplo, es usado frecuentemente para detectar el inicio de tendencias alcistas (*bullish*) o bajistas (*bearish*).

A continuación, en la Tabla 1, se muestran algunos indicadores principales de los que se suelen utilizar en análisis técnico. Puede consultar Anexo 4 con la lista completa de indicadores y sus fórmulas utilizados en este proyecto.

Indicador	Fórmula breve	Propósito / Uso típico
RSI	$RSI = 100 - \frac{100}{1 + RS}$	Detecta sobrecompra o sobreventa
MACD	$MACD = EMA_{\text{fast}} - EMA_{\text{slow}}$ Señal = EMA(9) del MACD	Cambios de tendencia, señales de compra/venta
EMA	$EMA_t = \alpha \cdot P_t + (1 - \alpha) \cdot EMA_{t-1}$	Suaviza la tendencia, soportes/resistencias
SMA	SMA(n) = suma precios últimos n días / n	Tendencia general, referencia a largo plazo
Bandas de Bollinger	Banda sup. = SMA(n) + k·σ Banda inf. = SMA(n) – k·σ	Volatilidad, ruptura de rangos

Tabla 1. Indicadores técnicos de análisis de tendencias de marcados financieros

Estos indicadores permiten identificar oportunidades y riesgos en el mercado, facilitando la toma de decisiones basada en reglas objetivas. Sin embargo, presentan varias limitaciones:

- **Señales contradictorias:** Diferentes indicadores pueden ofrecer señales opuestas en el mismo activo y periodo.
- **Sensibilidad al régimen de mercado:** Su efectividad varía considerablemente según la tendencia, la volatilidad o el marco temporal.
- **Riesgo de sobreajuste:** El uso simultáneo de muchos indicadores puede llevar a conclusiones erróneas, generando patrones que sólo existen en los datos pasados

En la práctica, los analistas tienden a combinar múltiples indicadores con el objetivo de aumentar la precisión de sus estrategias. No obstante, la falta de criterios sistemáticos para seleccionar y combinar estos indicadores hace que la interpretación siga estando sujeta a la subjetividad y a posibles sesgos humanos [1].

4.2.3 Limitaciones del análisis tradicional y necesidad de un enfoque cuantitativo

Los indicadores técnicos son herramientas útiles en finanzas, sin embargo, presentan limitaciones: pueden dar **señales contradictorias**, ser sensibles al mercado y favorecer el sobreajuste, dificultando resultados robustos. Además, su selección manual introduce sesgos y no siempre aprovecha toda la información disponible.

Figura 6. Contradicción entre indicadores técnicos. Fuente: TradingView

Durante el periodo señalado en el recuadro rojo como se observa en la Figura 6, se aprecia una contradicción entre los indicadores técnicos analizados. Mientras uno de ellos (por ejemplo, MACD) señala una tendencia bajista o el inicio de una fase de debilidad en el precio, el otro (por ejemplo, Acumulación / Distribución o ADX) muestra señales opuestas, ya sea manteniendo una tendencia alcista o indicando fortaleza en el movimiento.

La presencia de señales contradictorias y la complejidad inherente de los mercados financieros evidencian las limitaciones de los métodos tradicionales basados únicamente en indicadores técnicos. Numerosos estudios recientes han demostrado que, para abordar con éxito estos retos, es necesario recurrir a enfoques capaces de capturar relaciones no lineales y patrones complejos, adaptándose además a la **naturaleza dinámica de los mercados** [2][5][13].

Por este motivo, en el presente proyecto se integran técnicas avanzadas como el aprendizaje automático, la selección automática de características, los métodos evolutivos y el análisis cuantitativo. Se realiza, asimismo, una evaluación comparativa de diversos algoritmos y una simulación mediante **backtesting**, permitiendo valorar el rendimiento real de las estrategias propuestas a partir de los modelos desarrollados.

Esta aproximación constituye la base metodológica sobre la que se articula el sistema descrito en los siguientes apartados.

4.2.4 Arquitectura y esquema general del flujo de trabajo

La arquitectura del sistema desarrollado que se muestra en la Figura 7, se basa en un pipeline modular y automatizado, diseñado para gestionar todo el proceso de predicción, desde la adquisición y preprocesamiento de los datos hasta la validación de los resultados obtenidos por los modelos predictivos. El proceso consta de siguientes fases:

- **REST Web API de Yahoo Finance**

  Utilización de la API pública de Yahoo Finance para obtener datos históricos de precios (OHLCV) de múltiples acciones y activos de referencia (AAPL, AMZN, S&P500, etc.).

- **Cliente Web**

  Librería Python yfinance que permite realizar la descarga de datos históricos diarios de diversos stocks financieros (mercados bursátiles) en formato de dataframe de pandas.

- **ETL: Valores Faltantes, Validación de Formatos.**

  Proceso de limpieza y estandarización de los datos, incluyendo: detección y gestión de valores NaN o datos faltantes (mediana, último valor conocido, etc.), validación de formatos y coherencia temporal y conversión de tipos.

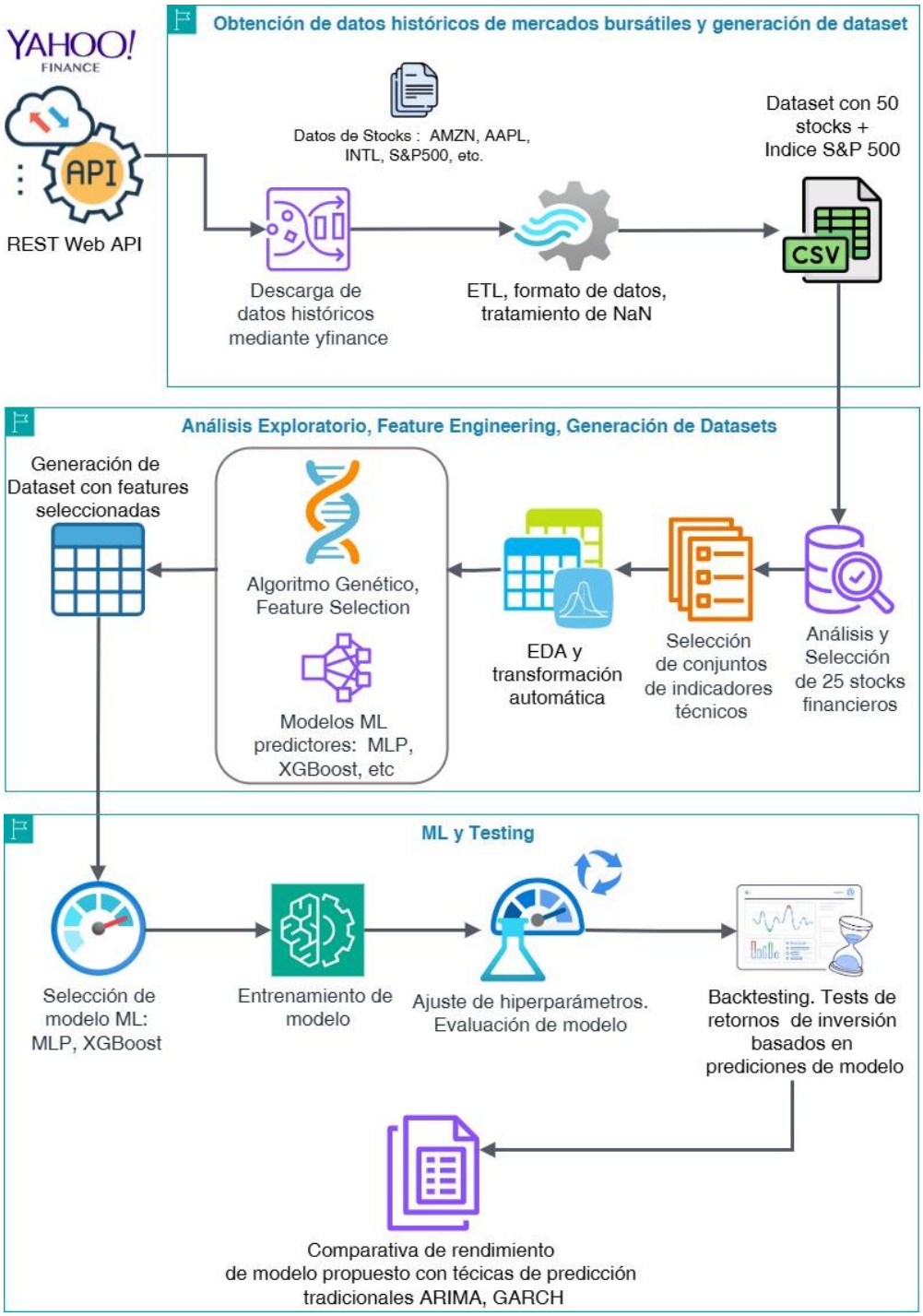


Figura 7. Arquitectura y esquema general del flujo de trabajo.

- **Dataset con 50 stocks + Indice S&P 500**

  Datos históricos de movimiento diario de precios de 50 stocks históricos seleccionados en este proyecto. Los datos se han guardado en formato de fichero csv

- **Análisis y Selección de Stocks Financieros**

  Selección de mercados y activos a partir de criterios cuantitativos (ratio Sharpe, volatilidad, beta, retornos anualizados, etc.) para formar el universo de estudio.

- **Selección de Conjuntos de Indicadores Técnicos**

  Generación de un pool inicial de indicadores técnicos (RSI, MACD, ADX, Bollinger Bands, medias móviles, etc.) para cada activo.

- **EDA y transformación automática**

  Análisis exploratorio y evaluación estadística de los indicadores técnicos seleccionados, identificando redundancias y asimetrías. Se aplica un pipeline automático que, en función de la distribución de cada variable (sesgo o skewness, curtosis, presencia de outliers), selecciona y ejecuta la transformación más adecuada—como logarítmica, Box-Cox, Yeo-Johnson o escalado robusto—para cada activo. Este proceso garantiza que los datos estén normalizados, comparables y preparados para el modelado posterior.

- **Algoritmo Genético - Selección de features**

  Se hará uso de algoritmo evolutivo (genético) para seleccionar automáticamente el subconjunto óptimo de indicadores (como features) que maximizan la capacidad predictiva de los modelos dónde se utilizan distintos modelos como:

  - **XGBoostRegressor** (Extreme Gradient Boosting)
  - **MLP** (Perceptrón Multicapa)

  con método wrapper en el que el individuo a evaluar es modelo con un determinado conjunto de features.

- **Generación de dataset con features seleccionadas**

  Creación de datasets finales, combinando activos y features seleccionadas, estructurados para la fase de modelado.

- **Selección de Modelos ML**

  Evaluación y selección de modelo ML según las métricas de rendimiento para usarlo en backtesting.

- **Entrenamiento de Modelo**

  Proceso de entrenamiento de modelo sobre el dataset con características seleccionadas por el algoritmo genético y obtención de métricas de su rendimiento.

- **Ajuste de hiperparámetros y evaluación de modelo**

  Medición de la capacidad predictiva sobre datos no vistos por el modelo (out of sample), utilizando métricas adecuadas (RMSE, MAE, R2).

- **Backtesting**

Simulación de operaciones reales aplicando las predicciones del modelo para evaluar el rendimiento de estrategias de inversión. Incluye cálculo de retornos, drawdowns, % de operaciones ganadores y acumulación de capital.

- **Comparativa de Rendimiento:**

  Análisis comparativo entre los modelos avanzados propuestos y técnicas tradicionales (ARIMA, GARCH), para justificar la superioridad del enfoque basado en ML.

4.2.5 Obtención de datos históricos de mercados bursátiles y generación de dataset

Para llevar a cabo esta tarea, se ha desarrollado una aplicación en Python que automatiza la descarga de datos históricos desde el servicio público Yahoo Finance para un universo de 50 activos representativos.

La aplicación recupera series temporales desde julio de 1999 hasta la fecha actual, incluyendo los campos esenciales de precios (apertura, cierre, máximos, mínimos) y volumen. Los datos se almacenan en un único fichero CSV (**AllStocksHistoricalData.csv**), estructurado y listo para su posterior tratamiento dentro del pipeline de ingeniería de características y modelado. El código fuente completo de esta aplicación puede consultarse en el Anexo 5.

En la Tabla 2. Muestra reducida de activos seleccionados para análisis y modelado ML se muestran los primeros 10 valores a modo de ejemplo. La tabla completa con todos los valores analizados se adjunta en el Anexo 1

Símbolo	Nombre de la empresa	Industria	Capitalización de mercado en billones USD
AAPL	Apple Inc.	Electrónica de consumo	3,15
ADI	Analog Devices, Inc.	Semiconductores	0,11
ADP	Automatic Data Processing, Inc.	Software - Aplicaciones	0,13
AMAT	Applied Materials, Inc.	Equipos y materiales semiconductores	0,14
AMD	Advanced Micro Devices, Inc.	Semiconductores	0,18
AMGN	Amgen Inc.	Fabricantes de medicamentos - General	0,15
AMZN	Amazon.com, Inc.	Venta minorista por Internet	2,22
AXP	American Express Company	Servicios de crédito	0,21
BKNG	Booking Holdings Inc.	Servicios de viajes	0,17
CAT	Caterpillar Inc.	Maquinaria agrícola y pesada	0,16

Tabla 2. Muestra reducida de activos seleccionados para análisis y modelado ML

Esa selección no es arbitraria, sino que responde a los siguientes criterios:

- **Diversificación sectorial:** Se han seleccionado empresas que representan diferentes sectores de la economía (tecnología, consumo, industria, salud, energía, etc.) para asegurar que los resultados del modelo no estén sesgados por un único sector o comportamiento de mercado.
- **Alta liquidez y relevancia:** Son empresas de gran capitalización y volumen de negociación, lo que garantiza la disponibilidad y calidad de los datos históricos. Además, estas empresas suelen ser referentes globales y están incluidas en los principales índices bursátiles (como S&P500 y Nasdaq).
- **Cobertura temporal amplia:** Al seleccionar compañías con amplio histórico disponible en Yahoo Finance, es posible cubrir periodos largos (en este caso, desde el año 2000 hasta la actualidad), permitiendo así evaluar el comportamiento de los modelos bajo distintas condiciones de mercado (tendencias alcistas, bajistas, crisis, recuperaciones, etc.).
- **Escalabilidad y robustez de experimentos:** Trabajar con 50 activos permite un equilibrio entre robustez estadística y viabilidad computacional, facilitando la comparación entre modelos y la validación cruzada sin incurrir en tiempos de cálculo excesivos.

4.2.6 Selección y análisis de activos financieros

La selección de activos financieros constituye un paso fundamental para garantizar la robustez y eficiencia en la aplicación de técnicas de predicción cuantitativa.
En este trabajo, el proceso se ha desarrollado siguiendo un enfoque objetivo y reproducible, basado en datos históricos diarios de más de 20 años para 50 activos bursátiles y el índice S&P 500. La metodología aplicada ha consistido en:

- **Cálculo sistemático de métricas clave** Rentabilidad, riesgo y liquidez para cada activo, empleando estadísticas anuales, mensuales y semanales (incluyendo ratios *Sharpe*, *Sortino*, Volatilidad, D*rawdown*, *beta* y volumen negociado). SSRN Working Paper (2015). *Common Metrics for Performance Evaluation* [15], [16].
- **Ranking multicriterio y filtrado** Los activos son seleccionados, priorizando aquellos con mejores ratios rentabilidad-riesgo y adecuada liquidez.
- **Validación visual y cuantitativa** de los activos seleccionados mediante gráficos comparativos y análisis de distribución de los indicadores.

Ratio Sharpe

El *Ratio Sharpe* cuantifica la rentabilidad adicional obtenida por cada unidad de riesgo asumida, ajustando por la volatilidad del activo o cartera. Es un estándar de referencia para comparar activos financieros (Sen, 2022) [15]. Su cálculo se basa en la diferencia entre la rentabilidad media y la tasa libre de riesgo, dividida por la volatilidad de los retornos:

$$\text{Sharpe} = \frac{\bar{r} - r_f}{\sigma}$$

Donde:

$\bar{r} =$ rentabilidad media del activo

$r_f =$ tasa libre de riesgo anual (en este estudio se ha utilizado un 2%)

σ = volatilidad de los retornos

Un **Ratio Sharpe** superior a **1** suele considerarse positivo en mercados desarrollados, mientras que valores próximos a 0 reflejan un exceso de rentabilidad insuficiente respecto al riesgo asumido. El desarrollo matemático completo y el ajuste de la volatilidad anualizada se recoge en los Anexo 2 y Anexo 3.

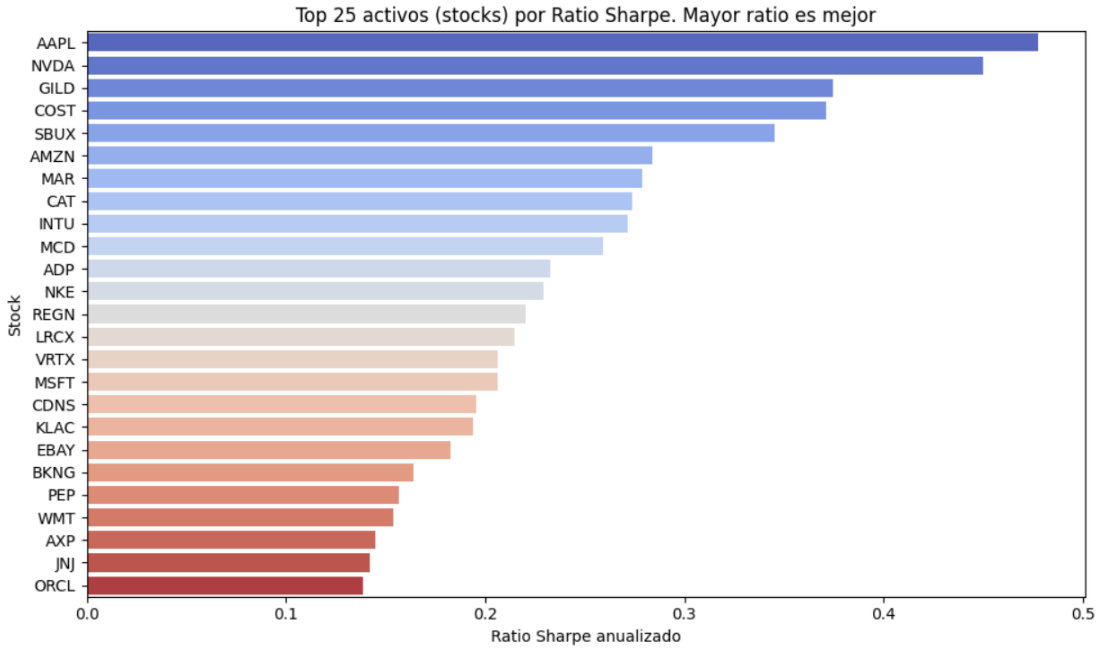


Figura 8. 25 mejores Ratios Sharpe de stocks financieros

En la Figura 8 se presentan los 25 activos con mejor Ratio Sharpe anualizado. Cabe destacar que los valores de Sharpe obtenidos en este análisis, todos inferiores a 1, son habituales cuando se consideran periodos históricos largos (en este caso, más de 20 años). En estos horizontes temporales, la acumulación de crisis y periodos de elevada volatilidad tiende a reducir el indicador, por lo que estos resultados deben interpretarse como normales y no necesariamente negativos.

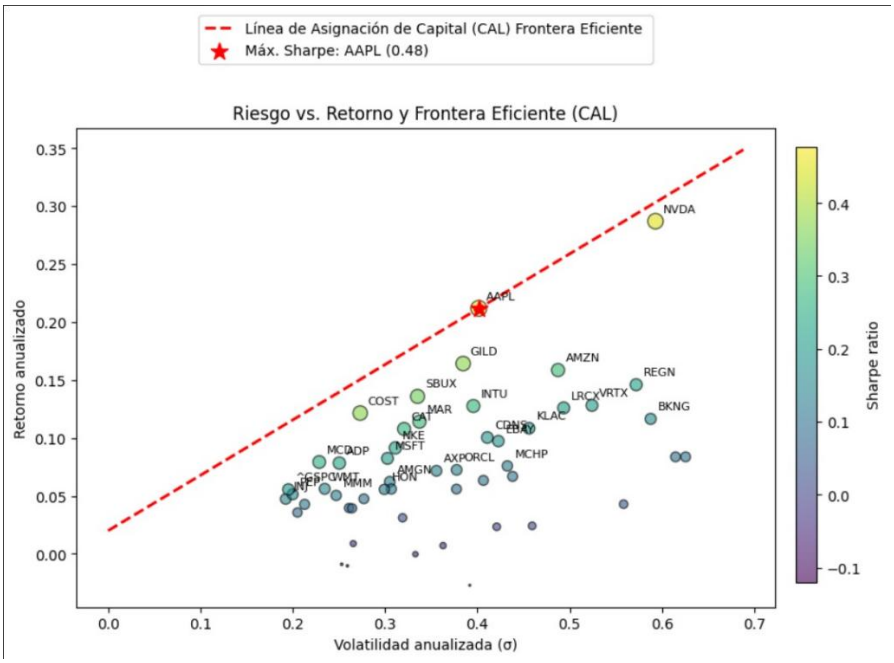


Figura 9. Retorno vs volatilidad de activos y Línea de Asignación de Capital (CAL)

La Figura 9 ilustra el compromiso entre riesgo (volatilidad anualizada) y retorno (rentabilidad anualizada) de los activos analizados. Cada burbuja representa un activo, cuyo tamaño y color son proporcionales a su Ratio Sharpe. La línea discontinua roja corresponde a la Línea de Asignación de Capital (CAL), cuya pendiente es el Sharpe máximo alcanzado (activo señalado con estrella roja). Los puntos más cercanos a esta línea ofrecen la mejor rentabilidad ajustada por riesgo

Volatilidad Anualizada

El SSRN Working Paper (2015) *Common Metrics for Performance Evaluation* [16] revisa las fórmulas de volatilidad diaria y anualizada, así como el cálculo de drawdown, destacando su importancia para la medición del riesgo real en series financieras. La volatilidad anualizada mide la **variabilidad de los retornos** de un activo a lo largo del tiempo, y es uno de los indicadores de riesgo más utilizados.

Suele preferirse una volatilidad baja, ya que implica un comportamiento más estable del activo. Sin embargo, una volatilidad excesivamente baja puede asociarse a menores oportunidades de rentabilidad.

$$\sigma_{\text{anual}} = \sigma_{\text{diaria}} \times \sqrt{252}$$

Donde:

$$\sigma_{\text{diaria}} = \sqrt{\frac{1}{N-1}\sum_{t=1}^{N}(r_t - \bar{r})^2}$$

σ_{diaria} es la desviación estándar de los retornos diarios.

En la presente selección, tras el filtro de Sharpe Ratio, se ha aplicado un criterio adicional basado en la volatilidad anualizada. Este enfoque permite priorizar activos que, además de mostrar una rentabilidad ajustada al riesgo favorable, presentan una mayor estabilidad a lo largo del tiempo

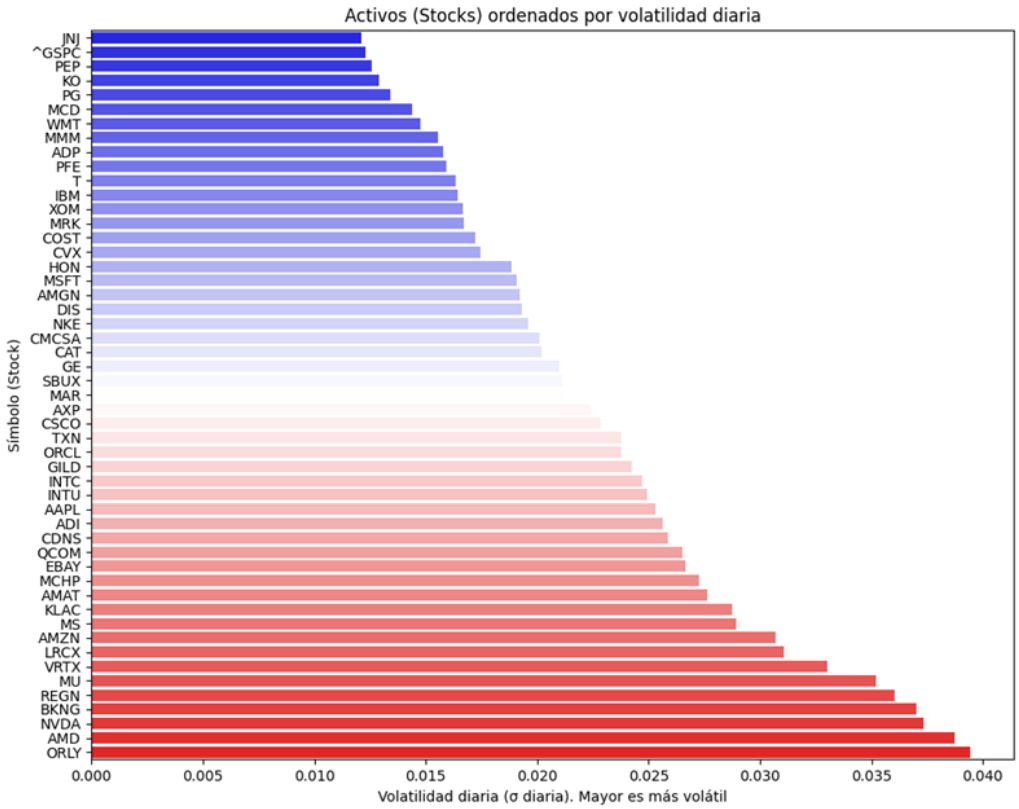


Figura 10. Volatilidad diaria de 50 activos seleccionados

Como se muestra en la Figura 10, se han ordenado los 50 activos según su volatilidad diaria. Este criterio permite identificar rápidamente los activos más estables frente a los más volátiles, facilitando un análisis comparativo del riesgo y la idoneidad para estrategias de inversión basadas en la estabilidad o la especulación.

Filtro de activos más resilientes

Para reforzar el proceso de selección, tras aplicar el filtro de Sharpe Ratio y volatilidad anualizada, incorporamos un paso adicional basado en la **resiliencia** de cada activo. Se calcula el **Drawdown** máximo de cada activo, definiendo como más resilientes aquellos con menor valor absoluto de drawdown (es decir, caídas históricas menos pronunciadas).

Al seleccionar los activos con los drawdowns más cercanos a cero, garantizamos la inclusión de títulos que no solo ofrecen buena rentabilidad ajustada al riesgo, sino también una mayor capacidad de recuperación frente a caídas extremas.

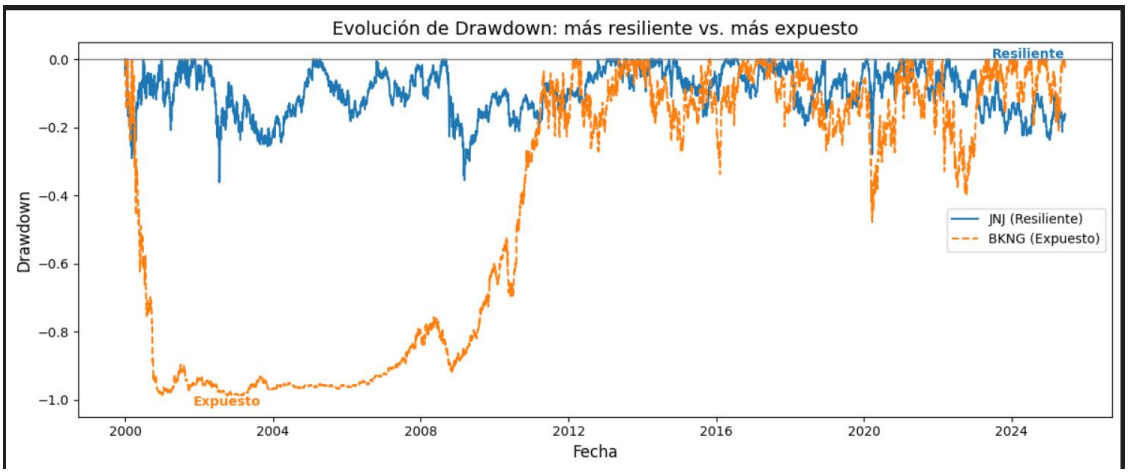


Figura 11. Evolución del Drawdown para JNJ (más resiliente) y BKNG (más expuesto)

En la Figura 11 se muestran los drawdowns de los dos extremos: JNJ, el más resiliente, cuya caída máxima no supera el –40 % y se recupera con rapidez; y BKNG, el más expuesto, que llega cerca del –100 % tras su pico y requiere varios años para volver a máximos, evidenciando su alta vulnerabilidad.

Beta

El coeficiente **Beta** mide la sensibilidad del retorno de un activo r_i frente a los movimientos del retorno del mercado de referencia r_m , normalmente representado por un índice amplio como el S&P 500. Se calcula como el cociente entre la covarianza de los retornos del activo y el mercado, y la varianza de los retornos del mercado:

$$\beta = \frac{\mathrm{Cov}(r_i, r_m)}{\mathrm{Var}(r_m)}$$

Donde: r_i es retorno del activo y r_m es el retorno del mercado

Un valor de Beta igual a 1 indica que el activo replica las variaciones del mercado. Valores superiores a 1 reflejan una **mayor volatilidad** respecto al mercado, mientras que valores inferiores a 1 indican un comportamiento más defensivo o menos expuesto a las oscilaciones globales. De este modo, Beta permite identificar **activos alineados** con la dinámica general del mercado, o bien aquellos más independientes. En este estudio, se ha priorizado la selección de activos con un Beta cercano a 1 para favorecer carteras diversificadas pero coherentes con el comportamiento del índice de referencia.

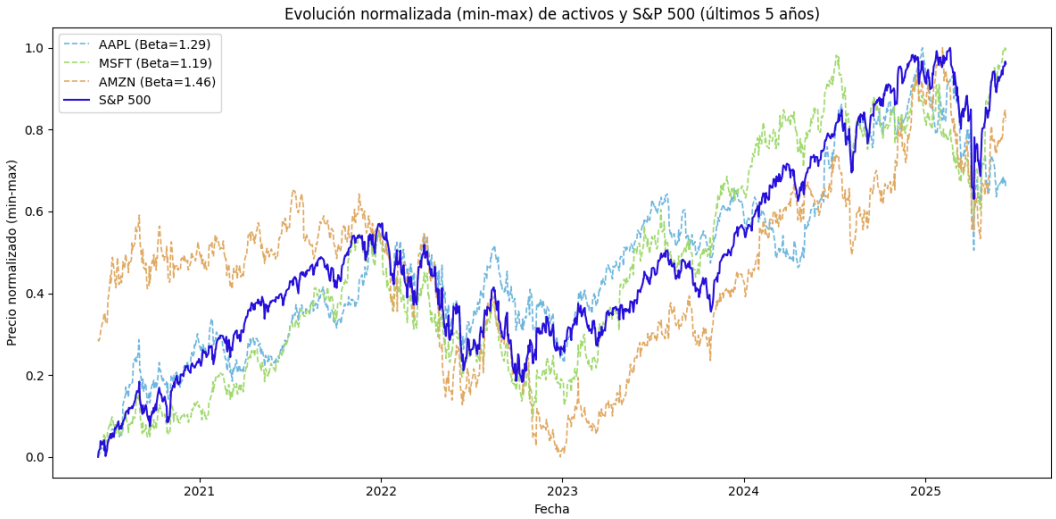


Figura 12. Evolución normalizada (min-max) de tres activos seleccionados y el S&P 500

La Figura 12 muestra la evolución comparativa de los precios normalizados (min-max) de tres activos representativos (AAPL, MSFT, AMZN) junto al índice S&P 500 en los últimos cinco años.

Selección de 25 mejores stocks (activos financieros) según múltiples criterios

En este trabajo, para seleccionar los 25 mejores activos financieros, se han combinado cinco métricas claves (véase Tabla 3) ampliamente utilizadas en la literatura:

Métrica	Descripción breve
Sharpe Ratio	Relaciona la rentabilidad excedente de un activo respecto a la tasa libre de riesgo con la volatilidad de sus retornos. Mide rentabilidad ajustada al riesgo.
β (Beta)	Mide la sensibilidad de un activo respecto a los movimientos del mercado (por ejemplo, S&P 500). Beta ≈ 1: comportamiento similar al índice; Beta > 1: más volátil.
MaxDrawdown	Mide la mayor caída máxima desde un punto alto a uno bajo durante un periodo. Indica el peor retroceso sufrido por el activo.
Sortino Ratio	Variante del Sharpe ratio que sólo penaliza la volatilidad negativa. Mide rentabilidad ajustada al riesgo de caídas.
Mean Daily Return %	Rentabilidad media periódica diaria expresada en porcentaje. Indica el rendimiento esperado del activo.

Tabla 3 Principales métricas para selección de activos financieros

Tal y como señala Sen (2022) en su estudio comparativo [15], la utilización conjunta de varias ratios de rentabilidad-riesgo y métricas de desempeño aporta una visión más robusta y equilibrada a la hora de evaluar activos financieros, especialmente en análisis de carteras diversificadas.

En un enfoque multicriterio se combina cada indicador normalizado en un único **índice compuesto** que permita ordenar los activos. Si denotamos por $m_{i,k}$ el valor de la métrica k $m_{i,k} \in [0,1]$ para el activo i y por w_k el peso asignado a dicha métrica, el **índice compuesto** S_i se define como:

$$S_i = \sum_{k=1}^{K} w_k \, m_{i,k} \quad \Big| \quad \sum_{k=1}^{K} w_k = 1$$

En nuestro caso se ha tomado cinco métricas:

$$k \in \{\text{Sharpe}, \beta, \text{MaxDrawdown}, \text{Sortino}, \text{MeanReturn \%}\}$$

y asignado pesos iguales $w_k = \frac{1}{5}$. Así, el índice compuesto queda:

$$S_i = \frac{1}{5}\left(m_{i,\text{Sharpe}} + m_{i,\beta} + m_{i,\text{MaxDrawdown}} + m_{i,\text{Sortino}} + m_{i,\text{MeanReturn \%}}\right)$$

Una vez calculado S_i para cada uno de los 50 activos, se seleccionan los **25 de mayor puntuación** para su inclusión en el análisis y la construcción de carteras.

En la Figura 13 se presenta un mapa de calor con las métricas clave para los 50 activos analizados, ordenados por el índice compuesto. La línea horizontal roja, señala el corte de selección: los 25 activos superiores (los primeros 25) han sido elegidos para el análisis posterior, al mostrar un mejor equilibrio multicriterio en rentabilidad, riesgo y resiliencia. (véase Tabla 4)

Número	Símbolo Stock (activo)	Nombre completo
1	NVDA	NVIDIA Corporation
2	AAPL	Apple Inc.
3	AMZN	Amazon.com Inc.
4	LRCX	Lam Research Corporation
5	SBUX	Starbucks Corporation
6	REGN	Regeneron Pharmaceuticals, Inc.
7	KLAC	KLA Corporation
8	BKNG	Booking Holdings Inc.
9	AMD	Advanced Micro Devices, Inc.
10	VRTX	Vertex Pharmaceuticals Incorporated
11	MAR	Marriott International, Inc.
12	CDNS	Cadence Design Systems, Inc.
13	CAT	Caterpillar Inc.
14	INTU	Intuit Inc.

15	GILD	Gilead Sciences, Inc.
16	MU	Micron Technology, Inc.
17	EBAY	eBay Inc.
18	AXP	American Express Company
19	AMAT	Applied Materials, Inc.
20	COST	Costco Wholesale Corporation
21	MSFT	Microsoft Corporation
22	ORCL	Oracle Corporation
23	ADI	Analog Devices, Inc.
24	MS	Morgan Stanley
25	NKE	NIKE, Inc.

Tabla 4. Los stocks de 25 activos financieros seleccionados

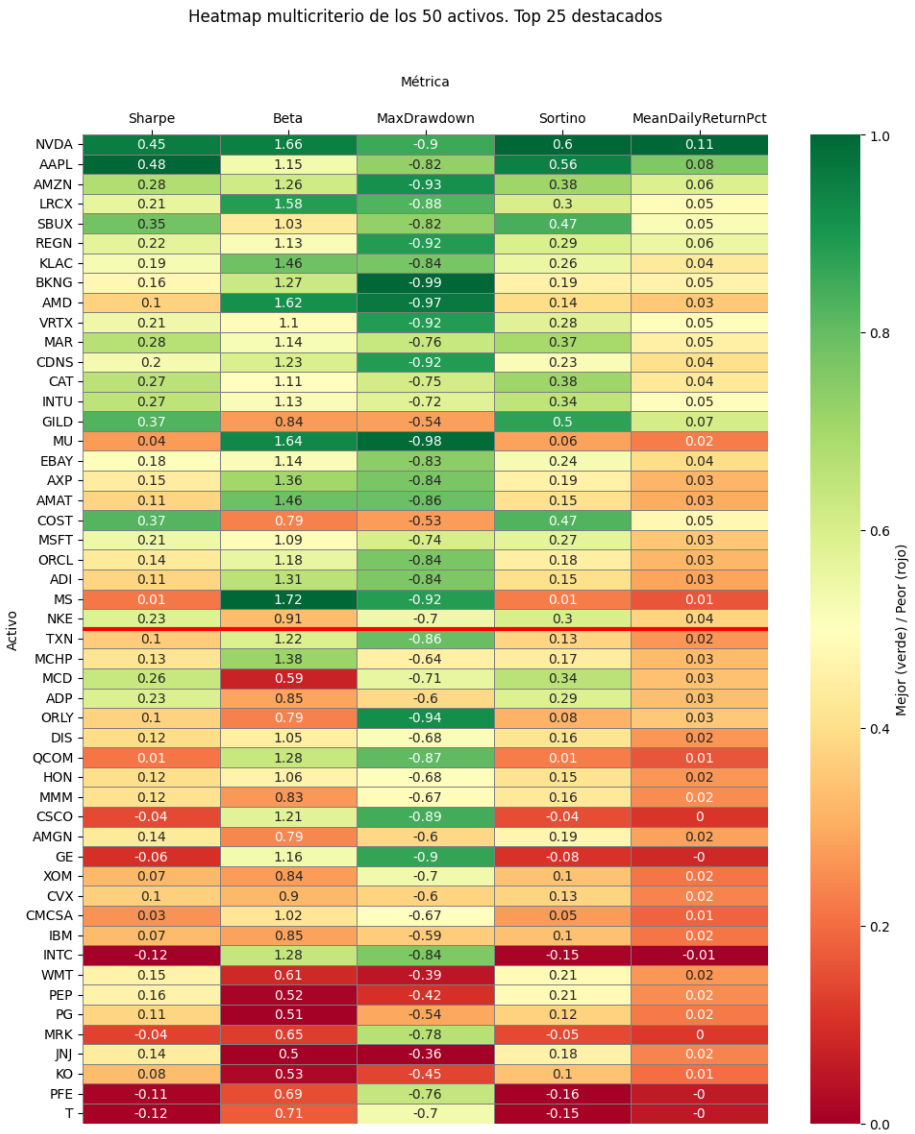


Figura 13. Mapa de calor multicriterio para los 50 activos y los 25 mejores

4.2.7 Selección de juego completo indicadores técnicos

Tal y como señala Mostafavi y Hooman (2025) [17], la selección de indicadores técnicos es un aspecto clave para el éxito en la predicción de precios bursátiles mediante modelos cuantitativos y de aprendizaje automático. En su estudio, emplean un enfoque sistemático para identificar los indicadores más relevantes entre diferentes familias (tendencia, momento, volatilidad, volumen), utilizando técnicas de selección y reducción de dimensionalidad como PCA y análisis de importancia en modelos Random Forest, XGBoost, SVR y LSTM.

En este trabajo, se ha optado por una estrategia similar, priorizando un subconjunto de indicadores ampliamente reconocidos tanto en la literatura académica como en la práctica profesional [17], [8], [9], y seleccionando variantes para capturar diferentes ventanas temporales y estilos de mercado.

- **Indicadores de tendencia** (como medias móviles, ADX, Parabolic SAR) capturan la dirección dominante del mercado, permitiendo identificar fases alcistas, bajistas o laterales.
- **Indicadores de momento** (ej. RSI, ROC, CCI, Williams %R) reflejan la velocidad y persistencia de los movimientos de precios, ayudando a anticipar aceleraciones y posibles agotamientos de tendencia.
- **Osciladores** (como Stochastic Oscillator, MACD) son especialmente útiles en rangos laterales y detectan condiciones de sobrecompra/sobreventa y puntos de reversión potencial.
- **Indicadores de volumen** (OBV, volumen medio) aportan información sobre la fuerza subyacente detrás de los movimientos de precio, distinguiendo entre movimientos respaldados por participación institucional o por menor volumen.
- **Indicadores de volatilidad** (como ATR, Ulcer Index) permiten ajustar estrategias al riesgo y adaptarse a entornos cambiantes, captando la amplitud y frecuencia de las fluctuaciones.
- Finalmente, la inclusión de **indicadores de convergencia/divergencia** (como el propio MACD o el RSI con sus señales de divergencia) facilita la detección de cambios sutiles en el equilibrio entre compradores y vendedores.

En el Anexo 4 se describen todos los grupos de indicadores técnicos de mercado con sus fórmulas que se han utilizado en este proyecto. Adicionalmente se han incluido señales binarias derivadas de los principales indicadores técnicos. (véase Tabla 5)

Columna binaria	Regla	Descripción
BINARY_RSI_14_OVERBOUGHT	1 si RSI_14 > 70, 0 en otro caso	Señal de sobrecompra según RSI.
BINARY_RSI_14_OVERSOLD	1 si RSI_14 < 30, 0 en otro caso	Señal de sobreventa según RSI.

BINARY_EMA_10_UPTREND	1 si Close > EMA_10 y Close_(t-1) ≤ EMA_10_(t-1), 0 en otro caso	Cruce alcista sobre EMA 10.
BINARY_MACD_12_26_9_UP	1 si MACD_12_26_9 > 0, 0 en otro caso	MACD positivo (tendencia alcista).

Tabla 5. Señales binarias calculadas a partir de valores de indicadores clave

4.2.8 Variables objetivo (Target)

La predicción de tendencias en series temporales financieras exige definir una variable objetivo robusta, capaz de captar tanto la dirección como la intensidad de la tendencia local. Una aproximación ampliamente reconocida en la literatura es el uso de la pendiente de una regresión lineal ajustada sobre una ventana móvil de precios, técnica conocida como **slope-detection labeling** (Chang et al., 2021 [18]).

En este contexto, la pendiente de la regresión lineal no se interpreta como una etiqueta de clase, sino como una magnitud continua que cuantifica la **dirección y la fuerza** del movimiento esperado dentro de un horizonte temporal futuro. Para cada instante temporal, la pendiente (*slope*) de la regresión lineal se calcula sobre una ventana móvil (por ejemplo, de 15 días, con 10 días de pasado y 5 de futuro respecto al punto de interés). Este proceso se representa en la Figura 14 , que muestra el cálculo de la pendiente sobre los precios de cierre de AAPL, junto al horizonte de predicción asociado.

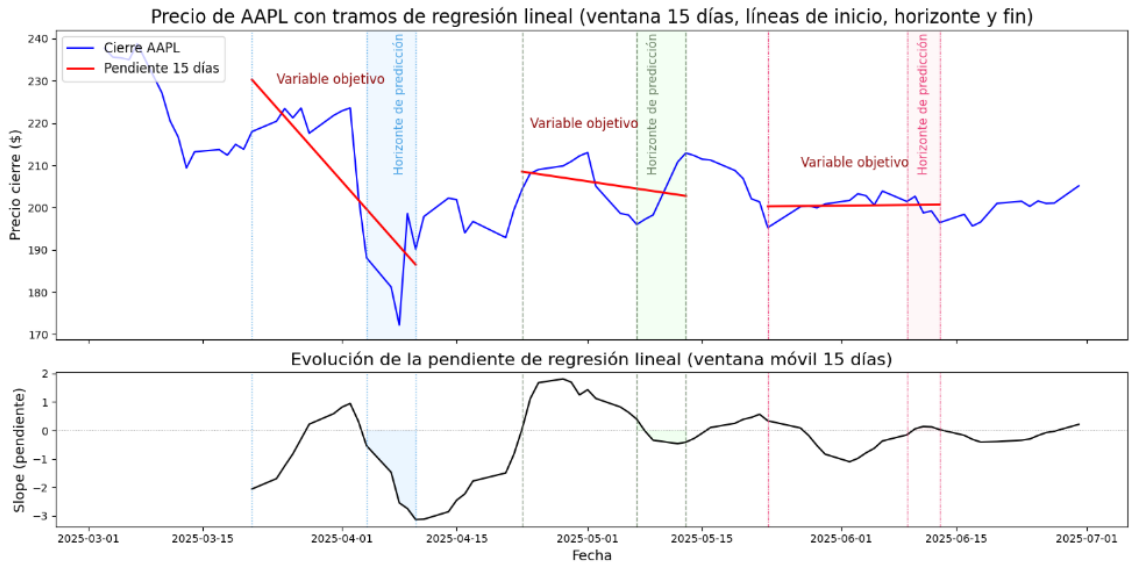


Figura 14. Pendiente de regresión lineal y su evolución sobre precios de cierre de AAPL

No obstante, el valor de la pendiente de regresión lineal depende de la magnitud del precio, lo que dificulta la comparación entre activos. Para superar esta limitación, se propone la utilización del **ángulo universal de tendencia**, que transforma la pendiente normalizada en un ángulo adimensional:

$$\theta = \arctan\left(\frac{\text{slope}}{\text{precio actual} \cdot r}\right)$$

donde:

slope es la pendiente de la regresión lineal dentro de ventana por ejemplo 15 días

precio actual es el precio del activo en el instante central,

r es un cambio relativo de referencia (por ejemplo, r =0.01, es decir, 1 % diario).

El ángulo resultante se expresa en grados, se recorta al intervalo $[-75°, +75°]$ y se normaliza linealmente al rango [0, 1]:

$$\text{Ángulo normalizado} = \frac{\theta + 75}{150}$$

De modo que:

$$-75° \to 0.0\,, \quad 0° \to 0.5\,, \quad +75° \to 1.0$$

De esta manera, la variable objetivo resultante permite comparar tendencias y su intensidad de forma universal entre diferentes activos y periodos temporales.

Para incrementar la robustez y analizar la sensibilidad del enfoque, se han generado variables objetivo utilizando diferentes tamaños de ventana y horizontes de predicción. Concretamente, se consideran ventanas de **10, 15 y 20 días**, y horizontes futuros de **5, 10 y 15** días, lo que permite evaluar la estabilidad del método propuesto bajo distintos escenarios temporales

4.2.9 Generación del dataset completo

El dataset final generado mediante el pipeline automático en Python (véase Anexo 6) que constituye la base sobre la que se apoya todo el trabajo experimental de este proyecto. El dataset resultante contiene actualmente **95.525** filas y **307** columnas, tanto las cotizaciones diarias como sus indicadores técnicos derivados, abarcando **25** activos financieros del mercado estadounidense desde **marzo de 2010** hasta **junio de 2025** siguiendo un orden cronológico dentro de cada símbolo bursátil.

Cada columna que refleja el valor de un indicador técnico y emplea la convención **INDICADOR_PARAMETROS**, facilitando la identificación inequívoca del tipo de indicador y sus configuraciones. Así, columnas como: EMA_10, RSI_14, MACD_12_26_9, ADX_14, CCI_20, BB_UPPER_20_2 u OBV_EMA_10 reflejan la aplicación sistemática de indicadores de tendencia, momento, volatilidad, volumen y osciladores, cada uno calculado sobre múltiples ventanas y parámetros. Esta estandarización no solo aporta claridad y trazabilidad, sino que también facilita el análisis comparativo de la importancia de cada señal.

El dataset incluye **275** variables de entrada (tras excluir las columnas de fecha, símbolo, precios y variables objetivo), calculadas automáticamente sobre precios de cierre, apertura, máximos, mínimos y volumen, e incorpora además **20** variables objetivo (targets) bajo la nomenclatura

TARGET_TREND_ANG_{window}_{horizon}. Estas variables objetivo representan ángulos de tendencia futura normalizados en el rango [0, 1], calculados mediante regresión lineal sobre ventanas de 10 y 15 días y diferentes horizontes (de **1** a **10** y de **1** a **15** días, respectivamente), permitiendo modelar y anticipar la inclinación y persistencia de tendencias a distintos plazos.

La generación y enriquecimiento del dataset se realiza a través de un pipeline robusto y reproducible que emplea librerías Python como:

- **numpy** para cálculos numéricos vectorizados.
- **pandas** para manejo de dataframes.
- La librería open source **stock-indicators** (https://pypi.org/project/stock-indicators/) para la obtención fiable y reproducible de indicadores técnicos sobre los precios y el volumen.

Todo el proceso está gestionado y orquestado mediante funciones propias implementadas en la clase **Feature_Generator.py** y el script maestro **Build_Dataset_All_Features.py**, que automatizan la generación de cientos de variables técnicas, estadísticas y binarias sobre todo el histórico para cada símbolo, realizando el procesamiento en paralelo para maximizar la eficiencia (véase Anexo 6).

Este enfoque garantiza la eliminación de valores nulos, la correcta alineación temporal entre variables y targets, y la ausencia total de data leakage. El resultado es un archivo enriquecido, exhaustivo y perfectamente preparado para su uso en tareas avanzadas de machine learning, experimentos de selección de variables, validación cruzada temporal y análisis de robustez en el contexto financiero.

La siguiente Figura 15 representa una muestra reducida (4 indicadores y una variable objetivo) de dataset completo con una variable objetivo TARGET_TREND_15_5

	Fecha	Symbol	Open	Close	High	Low	Volume	EMA_10	RSI_14	CCI_14	ROC_10	TARGET_TREND_ANG_15_5
0	2010-03-31	AAPL	7.07746	7.06273	7.11112	7.04651	430659600	6.90875	72.13914	142.04556	4.85455	0.69704
1	2010-04-01	AAPL	7.13516	7.09189	7.17484	6.99511	603145200	6.94205	73.04038	119.75222	5.03905	0.69855
2	2010-04-05	AAPL	7.06213	7.16762	7.16822	7.05582	684507600	6.98306	75.27710	117.81813	7.30709	0.68454
3	2010-04-06	AAPL	7.15891	7.19918	7.22022	7.12284	447017200	7.02235	76.16461	118.75470	6.58062	0.67162
4	2010-04-07	AAPL	7.19948	7.23104	7.27071	7.17273	628502000	7.06030	77.05989	116.40754	5.36006	0.67423
5	2010-04-08	AAPL	7.22623	7.21150	7.25929	7.15410	572989200	7.08779	75.19442	96.19802	4.61259	0.68391
6	2010-04-09	AAPL	7.25598	7.26680	7.26981	7.22683	334182800	7.12034	76.89885	104.59031	6.67990	0.66846
7	2010-04-12	AAPL	7.27913	7.28183	7.30527	7.26740	333026400	7.14970	77.35429	112.30588	4.93291	0.65859
8	2010-04-13	AAPL	7.26890	7.28603	7.29716	7.24636	306210800	7.17449	77.48786	96.83839	4.32027	0.63821
9	2010-04-14	AAPL	7.37169	7.38401	7.38762	7.33533	404076400	7.21258	80.39314	130.04272	4.17210	0.67206

Figura 15. Muestra reducida del Dataset_All_Features.csv

4.2.10 Análisis exploratorio de dataset y la transformación

Antes de entrenar los modelos predictivos, se realizó un análisis de correlación entre los principales indicadores técnicos seleccionados, representativos de distintas familias (tendencia, momento, osciladores, volatilidad y volumen). Como muestra la Figura 16, la mayoría de los

indicadores presentan **baja correlación entre sí** (valores próximos a cero), lo que sugiere que cada uno aporta información relativamente independiente al conjunto de variables de entrada.

No obstante, se observan algunas correlaciones moderadas o altas entre indicadores de la misma naturaleza, como es habitual (por ejemplo, entre EMA_10 y EMA_14, o entre distintos osciladores), lo que pone de manifiesto la importancia de seleccionar correctamente los parámetros para evitar redundancia y sobreajuste en el modelo.

Este análisis exploratorio refuerza la validez de la selección de features y respalda la diversidad informacional del dataset para la fase de modelado.

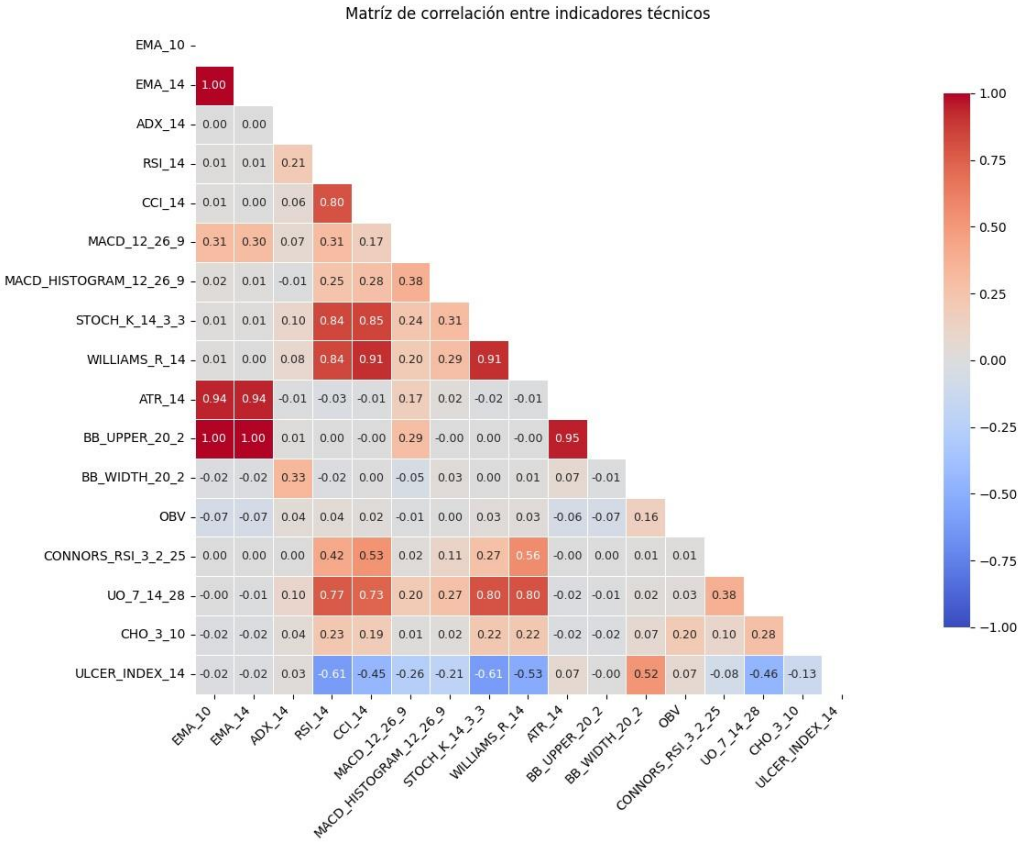


Figura 16. Correlación entre features de indicadores técnicos

El análisis de los histogramas de los principales indicadores técnicos y la variable objetivo, realizado sobre el activo AAPL (véase Figura 17), revela la presencia de asimetrías y distribuciones no normales en muchas de las variables. Si bien algunos indicadores, como el RSI o la variable objetivo, muestran distribuciones relativamente simétricas, otros como, por ejemplo, ATR, OBV, BB_WIDTH_20_2 o CHO_3_10 — presentan sesgos marcados, colas largas o acumulaciones de valores poco favorables para los algoritmos de aprendizaje automático.

Para abordar este problema y mejorar la calidad del dataset, se implementa una clase Python que analiza el sesgo (skewness) y la curtosis de cada variable, aplicando de manera automática la transformación más adecuada en cada caso (log-transform, Box-Cox, Yeo-Johnson, etc.). Estas

técnicas, ampliamente recomendadas en la literatura, permiten estabilizar la varianza y limitar la influencia de valores anómalos.

En definitiva, esta estrategia de preprocesamiento garantiza que la normalización y estandarización de los datos refleje fielmente el comportamiento de cada acción (stock), permitiendo construir modelos más fiables y evitando sesgos introducidos por la agregación de activos con distribuciones muy diferentes.

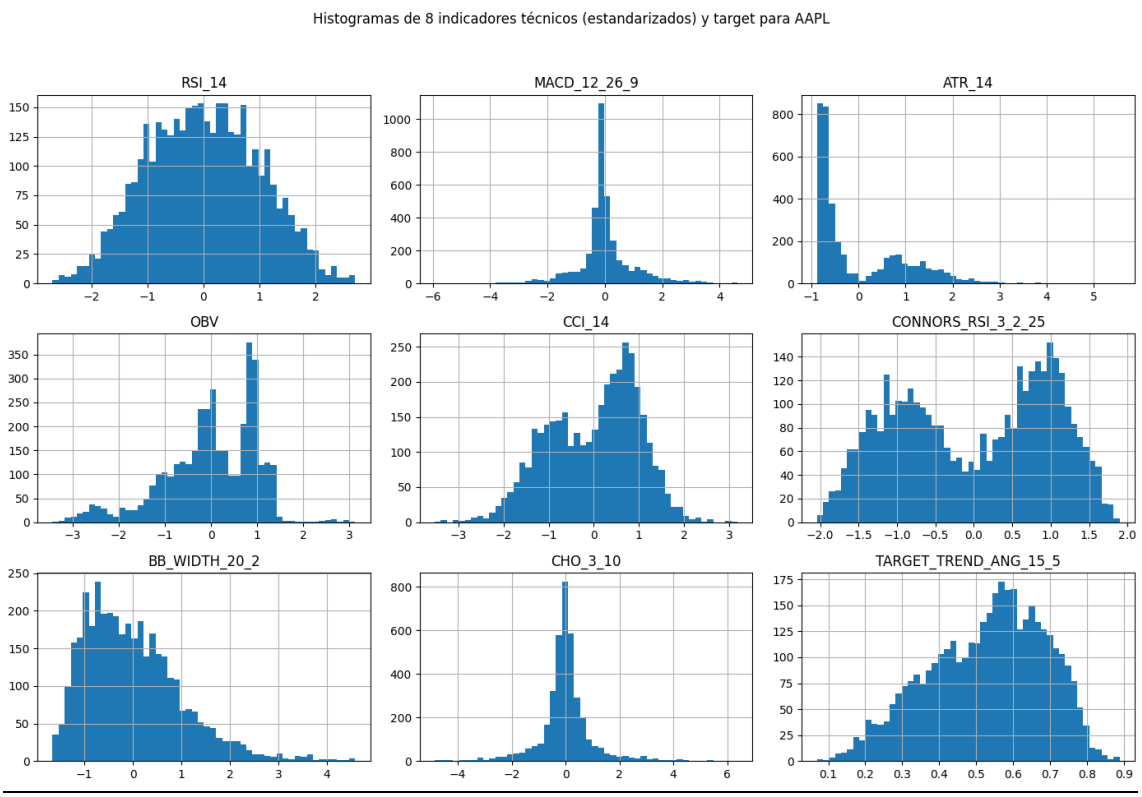


Figura 17. Histogramas de 8 indicadores técnicos estandarizados sin transformar

El preprocesamiento de variables numéricas en nuestro dataset se basa en una normalización adaptativa, cuyo objetivo es corregir asimetrías y escalas problemáticas antes del modelado.

Para ello, **cada variable** es analizada individualmente y, en función de su distribución estadística, se le aplica una transformación matemática adecuada. Todo el proceso se realiza *stock por stock* para evitar mezclar distribuciones de activos diferentes.

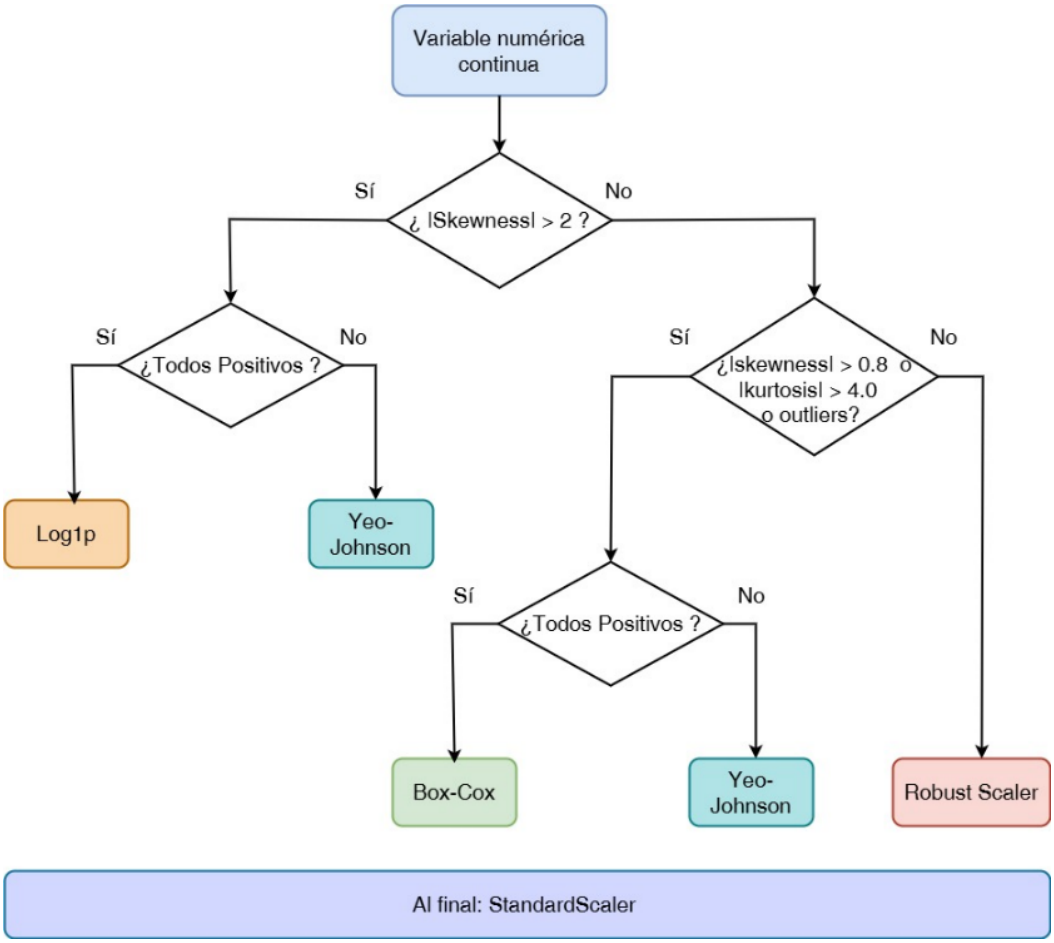


Figura 18 Flujo de transformación automática de features de indicadores

La lógica de selección automática de la transformación a aplicar sobre cada variable numérica se muestra en la Figura 18. En función de la asimetría, curtosis y presencia de outliers, el pipeline selecciona entre transformaciones logarítmica, Box-Cox, Yeo-Johnson o escalado robusto.

a. **Control del sesgo extremo** (**|skewness| > 2**):

   Si la variable presenta una asimetría muy acusada, el tratamiento varía en función del dominio de los valores:

   - **Todos los valores positivos (series > 0)**: se aplica transformación logarítmica (*log1p*) para estabilizar la varianza y reducir la influencia de los extremos.
   - **Existen valores negativos**: se utiliza *Yeo-Johnson*, ya que permite transformar variables con valores negativos o cero.

b. **Sesgos moderados, colas largas o outliers** (**|skewness| > 0.8** o **|kurtosis| > 4.0 o *outliers*):**

Cuando la variable muestra sesgo moderado, apuntamiento elevado o presencia de outliers (valores fuera de Q1 - 3 * IQR o Q3 + 3 * IQR), el pipeline selecciona una transformación de potencia:

- **Todos los valores positivos** (series > 0): se aplica *Box-Cox*, adecuada para aproximar la normalidad en distribuciones positivas.
- **Existen valores negativos**: se aplica *Yeo-Johnson* por su flexibilidad ante valores negativos.

c. **Variables "saludables"** (**|skewness| ≤ 0.8**, **|kurtosis| ≤ 4.0** y **sin outliers**):

Si la variable es aproximadamente simétrica, sin colas largas y sin valores atípicos destacables, se emplea *RobustScaler*, que escala y centra la variable limitando el efecto de posibles valores extremos residuales.

d. **Estandarización final (todas las variables)**:

Tras las transformaciones anteriores, todas las variables pasan por un *StandardScaler* (media cero, varianza uno), asegurando la homogeneidad de escalas para algoritmos sensibles a la magnitud de los datos.

e. **Procesamiento stock por stock (por símbolo)**:

Todo este procedimiento se ejecuta de forma independiente para cada activo financiero, evitando que la heterogeneidad entre stocks distorsione los parámetros estadísticos y asegurando que cada serie reciba el tratamiento óptimo según sus propias características.

El Anexo 7 contiene fórmulas para cada tipo de transformación y el Anexo 8 contiene los detalles de su implementación en Python.

Como se observa en la Figura 19, la aplicación del pipeline de normalización adaptativa ha permitido mejorar significativamente la simetría y distribución de muchos indicadores técnicos, especialmente en aquellos que presentaban fuertes sesgos o colas largas, como **ATR_14** y **BB_WIDTH_20_2**. Tras la transformación, la mayoría de las variables muestran distribuciones mucho más próximas a la normalidad.

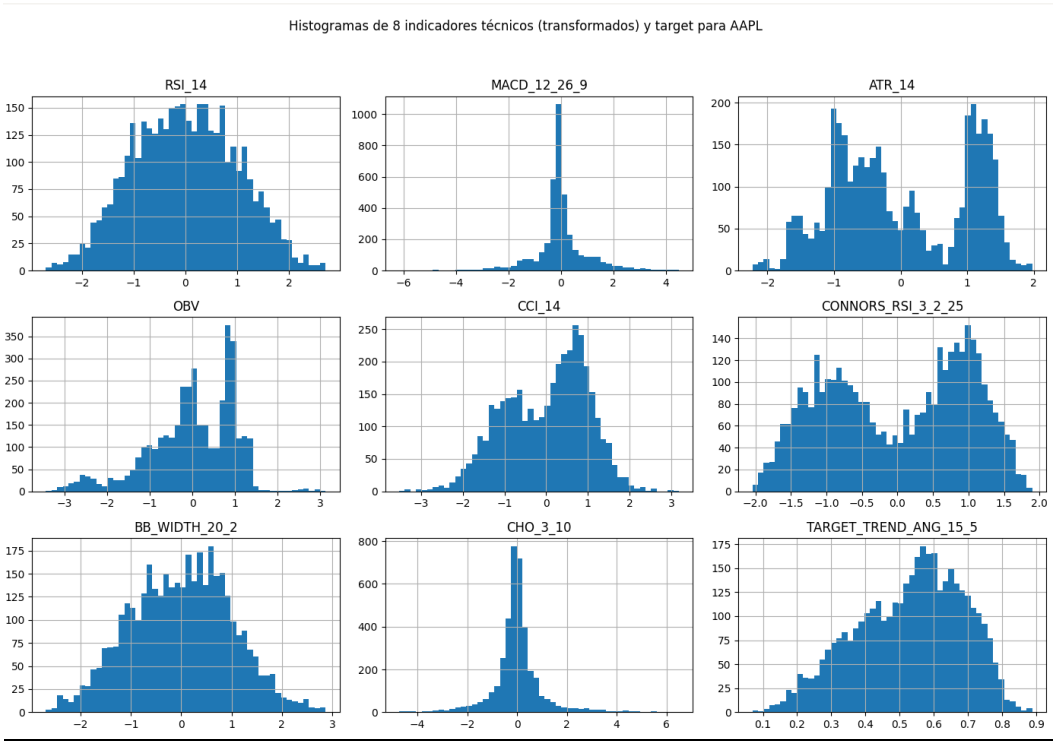


Figura 19. Histogramas de indicadores técnicos transformados y normalizados

4.2.11 Arboles de decisión con Gradient Boosting (XGBoost)

XGBoost (Extreme Gradient Boosting) [19], [20] es un algoritmo de aprendizaje supervisado basado en la técnica de Gradient Boosting, ampliamente utilizado por su eficiencia, precisión y capacidad para manejar grandes volúmenes de datos heterogéneos. La idea fundamental detrás de XGBoost es construir un modelo predictivo robusto mediante el ensamblado secuencial de múltiples árboles de decisión débiles, corrigiendo en cada iteración los errores cometidos por los modelos anteriores a través del gradiente de la función de pérdida.

A nivel formal (véase Figura 20) la predicción de XGBoost para una muestra x_i se expresa como la suma de las predicciones de K árboles individuales:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}$$

donde:

$\hat{y}_i$ es la predicción para la muestra i

K es el número total de árboles

$f_k(\boldsymbol{x_i})$ es la función de predicción de cada árbol,

$f_k \in \mathcal{F}$ es el espacio de funciones posibles (conjunto de árboles de decisión)

La función objetivo que XGBoost minimiza combina la función de pérdida $l(y_i, \hat{y}_i)$ y un término de regularización $\Omega(f_k)$ que penaliza la complejidad del modelo:

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

donde la regularización típica de cada árbol está definida como:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda |w|^2$$

donde:

T es el número de hojas del árbol

w los valores asignados a cada hoja

γ, λ son hiperparámetros de penalización.

En el caso de regresión, la función de pérdida habitual es el error cuadrático medio (MSE):

$$l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$$

Esta formulación permite a XGBoost construir modelos robustos y evitar el sobreajuste, manteniendo un equilibrio entre precisión y complejidad del modelo.

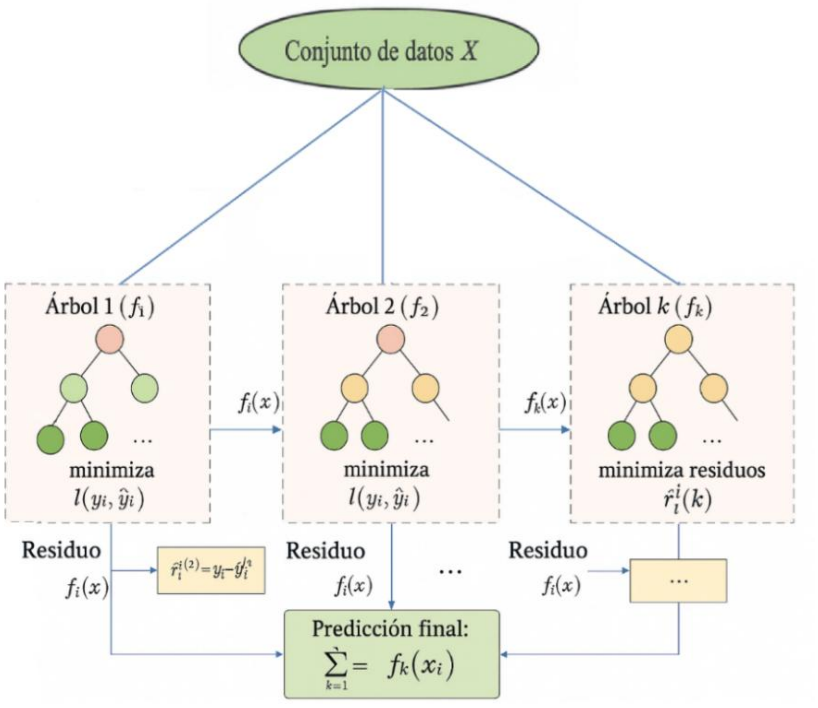


Figura 20. Esquema del funcionamiento de XGBoost Regressor.

4.2.12 Entrenamiento de modelo XGBoost Regressor y predicción de tendencia de precios

Para la predicción de la tendencia de precios con distintos horizontes temporales se emplea el modelo XGBoost Regressor (véase 4.2.11). El entrenamiento se realiza utilizando nuestro dataset normalizado, que incluye todos los indicadores técnicos y variables binarias generadas para los 25 activos seleccionados incluyendo el activo objetivo. El proceso se estructura de la siguiente manera:

Preparación y ordenación del dataset

Se utilizan los datos de los 25 activos seleccionados, desde 2010 hasta la actualidad, asegurando que cada muestra contenga tanto los indicadores como las variables binarias relevantes. El dataset completo se ordena previamente **por símbolo (stock) y fecha**. Esta ordenación estricta garantiza que, en cada fase de entrenamiento y validación, los datos correspondientes a cada activo siguen una secuencia temporal y se evita cualquier posible *data leakage* entre periodos y activos.

Selección del target

Se define la variable objetivo (target) a predecir, correspondiente a la tendencia de precios en el horizonte temporal y longitud de ventana seleccionados.

Train-Test Split cronológico

Se realiza una división temporal estricta del dataset, respetando el orden cronológico. Los datos de entrenamiento corresponden a los periodos anteriores al año objetivo de validación; los datos de test pertenecen exclusivamente al año a predecir. Esto garantiza que no existe data leakage y simula un entorno real de predicción.

Entrenamiento del modelo

Para cada fold, se entrena un modelo XGBoost Regressor sobre el conjunto de entrenamiento correspondiente y se evalúa sobre el conjunto de test (año objetivo).

Validación cruzada cronológica

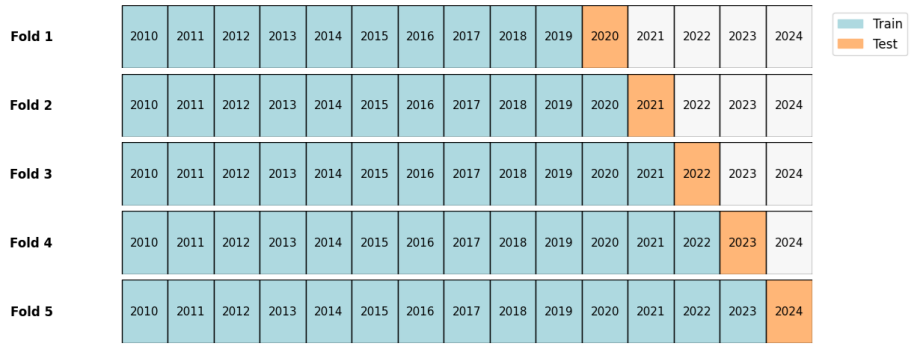


Figura 21. Configuración de validación cruzada cronológica con 5 folds

Como se muestra en la Figura 21 se implementa una validación cruzada basada en folds anuales. Cada fold utiliza como test un año concreto entre 2020 y 2024, empleando para el entrenamiento únicamente los años previos. De esta manera, se obtienen **5 folds** independientes y temporalmente coherentes.

Evaluación y métricas

Para cada fold, se calculan las principales métricas de regresión (MSE, RMSE, MAE y R2)

Finalmente, se reporta el promedio y la desviación estándar de cada métrica sobre de los 5 folds.

Resumen de métricas por fold (año test) para stock NVDA (Nvidia Corporation):

Sobre la variable objetivo como ángulo universal de tendencia (véase 4.2.8) con ventana de regresión **15 días** y el horizonte de predicción **5 días** se han obtenido siguientes métricas

Año test	Nº Filas Train	Nº Filas Test	MSE	RMSE	MAE	R^2
2020	78.087	253	0,0149	0,1221	0,0955	0,6110
2021	80.333	252	0,0127	0,1128	0,0915	0,7323
2022	82.602	253	0,0268	0,1637	0,1332	0,6513
2023	84.927	253	0,0262	0,1618	0,1279	0,3646
2024	86.550	252	0,0253	0,1592	0,1295	0,5284

Tabla 6. Métricas de rendimiento por fold, XGBoost Regressor, NVDA.

Los resultados obtenidos muestran valores de **R^2** superiores al **0,5** en todos los años de test y alcanzan valores por encima de **0,7** en algunos periodos, lo que indica una capacidad predictiva significativa del modelo para anticipar la tendencia de precios de NVDA en ventanas temporales futuras. Además, los valores de RMSE y MAE son bajos en relación con la escala normalizada de la variable objetivo, lo que refleja un buen ajuste en términos absolutos y relativos.

La predicción de tendencias en mercados financieros sigue siendo un problema de alta complejidad debido a la naturaleza dinámica y ruidosa de los datos. Por tanto, valores de R^2 superiores a 0,3 podrían considerarse positivos y realistas en este contexto, reflejando la capacidad del modelo para anticipar cambios de tendencia más allá de lo que permitiría un modelo aleatorio o ingenuo (véase Tabla 7).

Métrica	Media	Mediana	STD	Mínimo	Máximo
MSE	0,0212	0,0253	0,0068	0,0127	0,0268
RMSE	0,1439	0,1592	0,0244	0,1128	0,1637
MAE	0,1155	0,1279	0,0202	0,0915	0,1332
R^2	0,5775	0,611	0,1399	0,3646	0,7323

Tabla 7. Estadísticas de métricas de XGBoost Regressor para stock NVDA

En la Figura 22 se muestra la comparación entre la evolución de la variable objetivo: ángulo normalizado de tendencia de precios sobre activo NVDA con horizonte de predicción **5 días** siendo la **ventana** de **15 días** y su predicción generada por el modelo XGBRegressor entrenado sobre el dataset completo con **275** características para el activo **NVDA** durante el periodo de test para años 2022 y 2023.

Se observa una alta concordancia entre ambas curvas, lo que evidencia la capacidad del modelo para capturar la dinámica temporal de la serie. Las diferencias puntuales entre las curvas se producen, principalmente, en episodios de alta volatilidad o en transiciones rápidas, circunstancias en las que cualquier modelo predictivo tiende a presentar mayores desafíos.

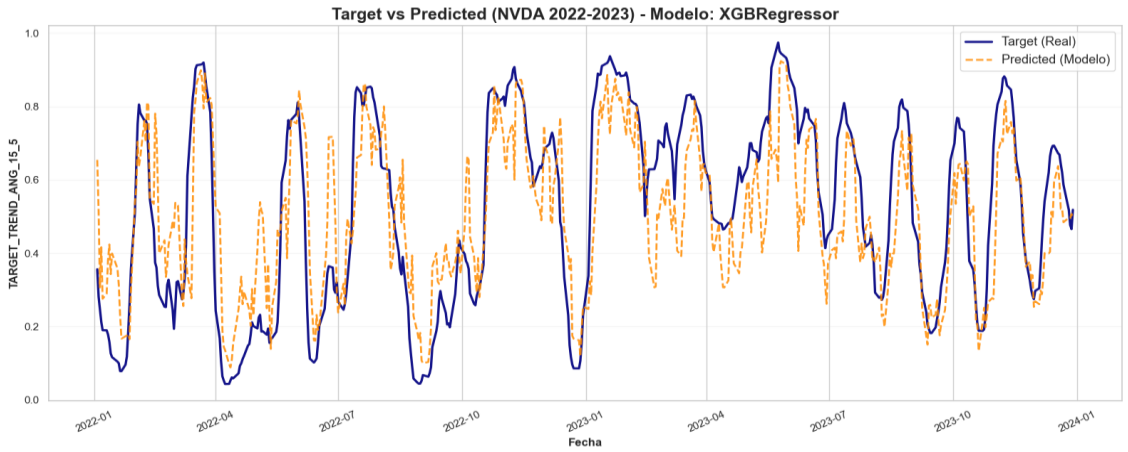


Figura 22. Comparativa de variable objetivo y su predicción, XGBRegressor para NVDA

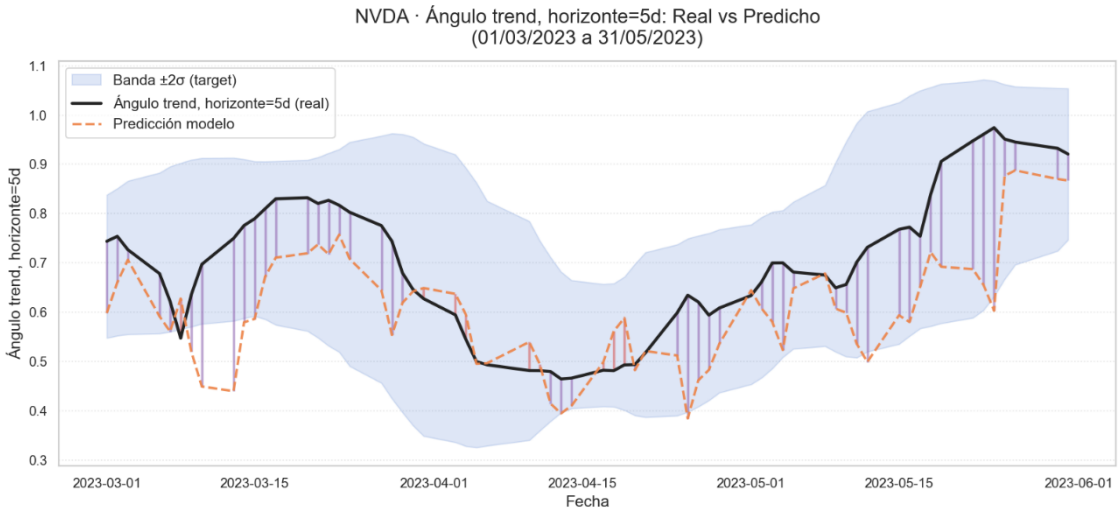


Figura 23. Ángulo de tendencia real y predicho para NVDA

Como se observa en la Figura 23 las predicciones del modelo siguen relativamente de cerca la evolución real del ángulo de tendencia de precios manteniéndose la mayor parte del tiempo dentro de las bandas de ±2σ. Dónde σ es la desviación estándar calculada sobre precios con una

ventana móvil de 20 días. Esto indica que el modelo no solo replica la tendencia general, sino que además es capaz de inferir reglas y patrones subyacentes en la serie, limitando los errores extremos a situaciones poco frecuentes.

La Figura 24 muestra la comparación entre la evolución real de los precios de cierre de NVIDIA y la predicción del modelo XGBoost Regressor sobre el ángulo de tendencia para el horizonte de 5 días.

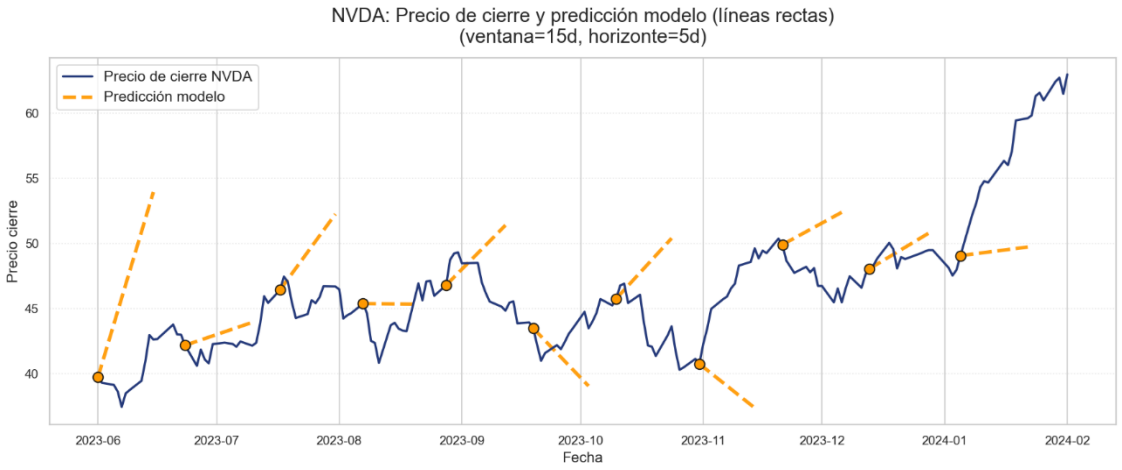


Figura 24. Predicción de tendencia de precios de NVDA (XGBoost Regressor)

Al igual que en el caso de NVIDIA (véase Tabla 6), se ha realizado de forma equivalente para el resto de activos de la cartera. En todos los casos, se ha aplicado el mismo procedimiento de validación cruzada y evaluación del error. (véase Figura 25)

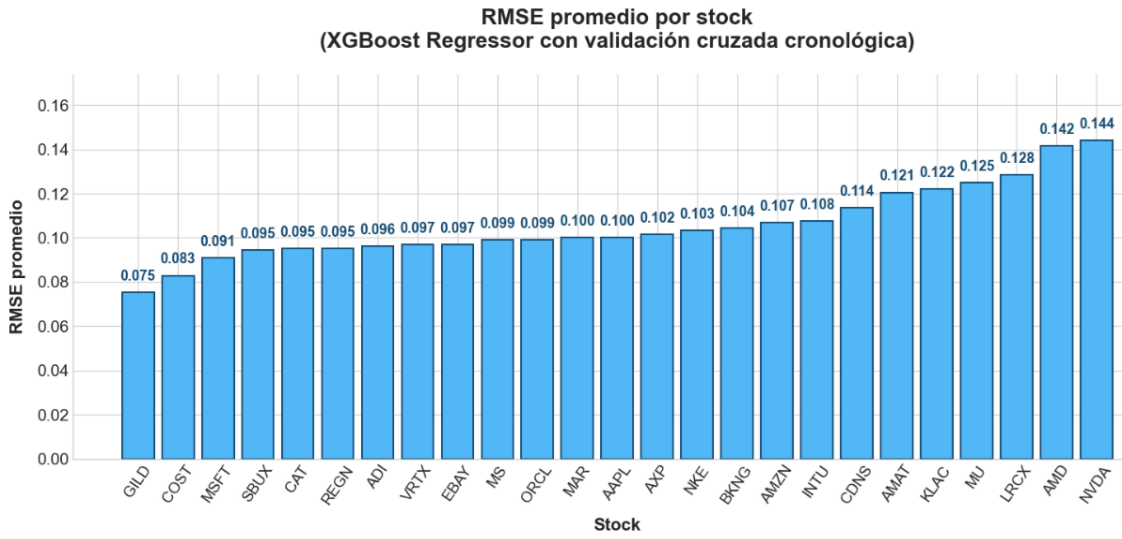


Figura 25. RMSE promedio de XGBoost Regressor para cada stock

El RMSE obtenido en la predicción de la tendencia (variable objetivo normalizada entre 0 y 1) puede transformarse directamente a grados utilizando la siguiente relación:

$$\text{Error en grados} = \text{RMSE} \times 150$$

Dónde: 150 es el rango de +- 75°

$$Stock\ GILD, el\ \text{error típico} \approx 0{,}075 \times 150 = 11{,}25°$$

$$Stock\ NVDA, el\ \text{error típico} \approx 0{,}144 \times 150 = 21{,}6°$$

El RMSE promedio por activo financiero, expresado en la escala original del ángulo de tendencia, oscila entre $11°$ y $22°$. Esto significa que el modelo puede equivocarse en la predicción de la pendiente en ese rango, siendo más preciso en acciones con tendencias más claras y menor volatilidad.

Estos resultados evidencian que el modelo XGBoost Regressor puede captar la dirección y fuerza de la tendencia con una precisión suficiente para aplicaciones de análisis cuantitativo y toma de decisiones en trading.

En la Tabla 8 se muestran los valores medios de las principales métricas de evaluación (RMSE, MSE, MAE y R^2) obtenidas mediante validación cruzada temporal para cada uno de los activos seleccionados (véase 4.2.6). En la Tabla 9 se muestran las estadísticas descriptivas de estas métricas. Se observa que el modelo XGBoost Regressor presenta mejor capacidad predictiva (menor RMSE medio) en valores como **GILD (0,075)** y **COST (0,083)**, mientras que otros como **NVDA (0,144)** y **AMD (0,142)** resultan más difíciles de predecir, posiblemente debido a su mayor volatilidad. El coeficiente de determinación R^2 medio oscila entre **0,51** y **0,68**, lo que indica un ajuste moderado a bueno del modelo según la acción considerada.

Activo Financiero	RMSE	MSE	MAE	R^2
GILD	0,0753	0,0058	0,0577	0,6355
COST	0,083	0,0071	0,0668	0,5095
MSFT	0,091	0,0086	0,0725	0,5839
SBUX	0,0945	0,0091	0,0722	0,575
CAT	0,0952	0,0092	0,0754	0,6815
REGN	0,0953	0,0092	0,0738	0,6435
ADI	0,0961	0,0093	0,0758	0,5763
VRTX	0,0968	0,0097	0,0731	0,5675
EBAY	0,0971	0,0097	0,0769	0,6647
MS	0,0989	0,0099	0,0775	0,6555
ORCL	0,0992	0,01	0,075	0,6205
MAR	0,1002	0,0106	0,0777	0,6408
AAPL	0,1003	0,0104	0,0784	0,5872
AXP	0,1017	0,0106	0,0786	0,5754
NKE	0,1032	0,0109	0,0791	0,6392
BKNG	0,1045	0,0112	0,0807	0,6527

AMZN	0,1068	0,0121	0,0848	0,6038
INTU	0,1076	0,012	0,086	0,5834
CDNS	0,1137	0,0133	0,089	0,5726
AMAT	0,1205	0,0147	0,0939	0,637
KLAC	0,122	0,015	0,0949	0,608
MU	0,1249	0,0158	0,097	0,63
LRCX	0,1284	0,0167	0,1008	0,6105
AMD	0,1415	0,0202	0,11	0,6236
NVDA	0,1439	0,0212	0,1155	0,5775

Tabla 8. Métricas promediadas de XGBoost Regressor para 25 activos.

Métrica	Media	STD	Mínimo	Máximo
RMSE	0.1057	0.0166	0.0753 (GILD)	0.1439 (NVDA)
MSE	0.0117	0.0037	0.0058 (GILD)	0.0212 (NVDA)
MAE	0.0825	0.0133	0.0577 (GILD)	0.1155 (NVDA)
R^2	0.6102	0.0392	0.5095 (COST)	0.6815 (CAT)

Tabla 9. Estadísticas de métricas de modelo XGBoost Regressor sobre 25 activos.

4.2.13 Selección de tamaño de ventana y horizonte de predicción óptimos

Se realizaron dos experimentos de rendimiento de modelo empleando ventanas de 10 y 15 días para analizar el efecto del horizonte de predicción. Se utilizaron datos de 2015 a junio de 2025 desde nuestro dataset completo con todas las características normalizadas (véase 4.2.9), seleccionando tres acciones de distintos sectores (MSFT, AMZN, CAT), y aplicando una partición cronológica (80% entrenamiento, 20% validación).

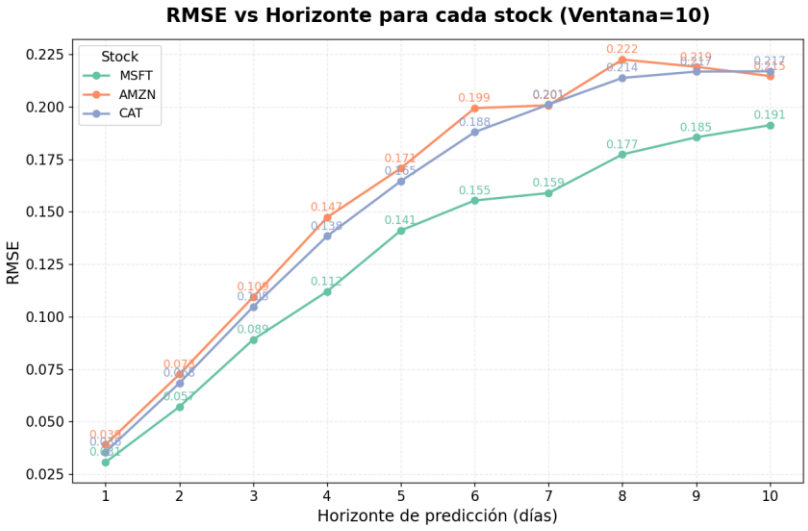


Figura 26. RMSE de XGBoost Regressor con ventana 10 días para diferentes horizontes

En ambos casos, los resultados reflejados en la Figura 26 y Figura 27 muestran que el error de predicción (RMSE) aumenta progresivamente a medida que se amplía el horizonte. Este comportamiento, coherente en los dos experimentos, confirma que anticipar la tendencia bursátil resulta más complejo cuanto más lejos se quiere predecir, debido a la mayor influencia de la volatilidad y el componente aleatorio del mercado.

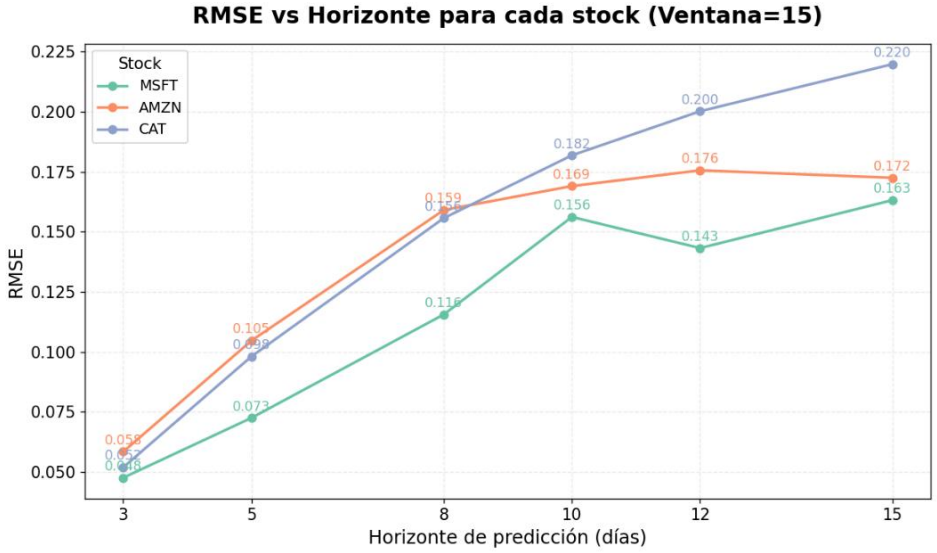


Figura 27. RMSE de XGBoost Regressor con ventana 15 días para diferentes horizontes

Aunque pueden existir pequeñas diferencias entre activos o ventanas, el patrón global es claro en ambos gráficos: a mayor horizonte, mayor error. Estos hallazgos subrayan la importancia de seleccionar adecuadamente el horizonte y el tamaño de ventana, buscando el equilibrio entre suavizar el ruido y mantener la capacidad predictiva.

Al analizar distintas ventanas y horizontes de predicción se determina que cada opción tiene sus ventajas e inconvenientes:

- Horizontes cortos (3 días) reducen la volatilidad y mejoran la precisión, pero limitan el potencial de beneficio, ya que los precios se mueven menos en poco tiempo.
- Horizontes largos (15 días) ofrecen mayores oportunidades si la tendencia es clara, pero también mayor incertidumbre y error.

Por ello, la combinación seleccionada es de una ventana de **15 días** y el horizonte de predicción de **5 días** que busca equilibrar precisión y potencial de beneficio.

4.2.14 Algoritmo genético en selección de características

Aunque los modelos como Random Forest o XGBoost calculan internamente la importancia relativa de cada característica, dichas métricas suelen evaluar mayormente contribuciones individuales y lineales.

En cambio, tal como señalan Zhila Yaseen Taha, Abdulhady Abas Abdullah y Tarik A. Rashid (2021) [21], los algoritmos genéticos permiten identificar relaciones complejas y no lineales entre variables, evitando óptimos locales y mejorando significativamente la precisión en escenarios de alta dimensionalidad.

Se parte de un conjunto inicial compuesto por **275** características técnicas (features) derivadas de indicadores financieros, con el objetivo principal de reducirlo a un subconjunto de entre **20 y 50** variables óptimas para cada activo, mediante la aplicación de un algoritmo genético (GA). Esta reducción es esencial para mejorar la generalización del modelo, minimizar el ruido, evitar el sobreajuste y mitigar la maldición de la dimensionalidad, optimizando al mismo tiempo la eficiencia computacional.

Este enfoque se considera ventajoso frente a la importancia de características tradicional y demuestra un mayor rendimiento en conjuntos de datos complejos. Además, estudios como el de Kocyigit et al. (2024) [22] refuerzan esta perspectiva

Algoritmo Genético (GA) propuesto para la selección de características

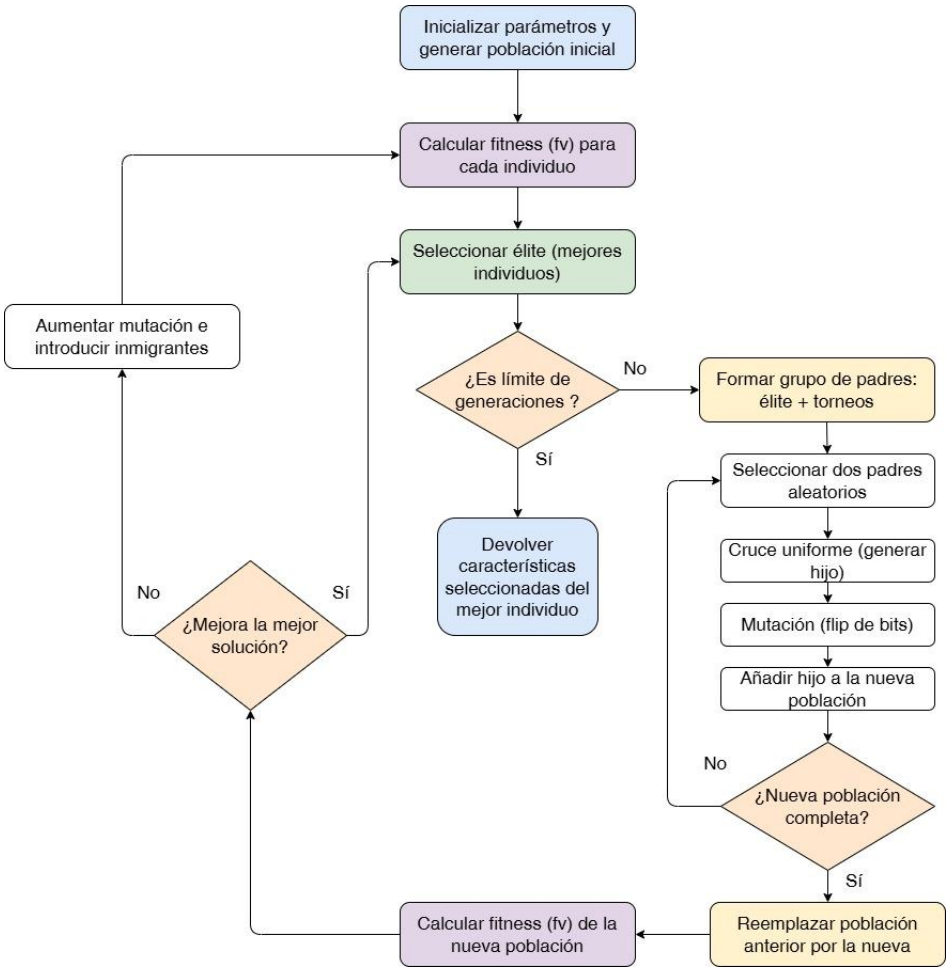


Figura 28. Algoritmo Genético para selección de características

a. Inicialización de la población

Se generan aleatoriamente 25 cromosomas, cada uno con una selección entre un mínimo y máximo (15-50) de características de dataset. Todas selecciones tienen mismo tamaño de cromosomas, pero distintos genes.

b. Evaluación de individuos

Para cada cromosoma (conjunto de features seleccionadas), el GA entrena el modelo de regresión especificado (por ejemplo, XGBRegressor, ElasticNet, Ridge, etc.) utilizando únicamente esas variables.

La evaluación de cada individuo se realiza sobre el conjunto de datos de validación proporcionado al algoritmo (por ejemplo, un bloque temporal fuera de entrenamiento), empleando la métrica definida (como RMSE o R^2, según configuración). El fitness final de cada individuo corresponde al valor obtenido en este test fuera de muestra, pero la partición temporal y la métrica de evaluación dependen de los parámetros de entrada del propio GA.

c. Selección

Los 10 cromosomas con la mejor métrica (élite) pasan automáticamente a la siguiente generación. El resto de la población se completa seleccionando por torneos aleatorios de tres individuos.

d. Cruce (crossover)

Se generan nuevos cromosomas mediante cruce uniforme entre pares de padres, mezclando indicadores según una máscara binaria generada aleatoriamente y respetando los límites de features activas. (véase Figura 29)

e. Mutación

Con una probabilidad de 30-50%, se modifica aleatoriamente la selección de algún indicador en los hijos, manteniendo la diversidad.

f. Reemplazo de población

La población anterior se reemplaza totalmente por la nueva generación (élite + hijos), repitiendo el proceso durante 10 - 20 generaciones.

g. Diversidad e inmigración

Si durante varias generaciones no hay mejora de métrica seleccionada (RMSE, R2, etc.), se incrementa la tasa de mutación o se introducen nuevos cromosomas aleatorios para evitar estancamiento.

h. Resultado final

Tras N generaciones (según parámetro de número máximo de generaciones), se reportan los indicadores técnicos seleccionados por el cromosoma con mejor métrica.

El algoritmo es parametrizado por lo que permite definir cada aspecto como: tamaño de población, número de generaciones, cantidad mínima y máxima de características (features) a seleccionar, probabilidad de mutación, tamaños de torneo, número de individuos de élite, métrica objetivo como: RMSE, MSE, MAE, R2.

Codificación de genes como features

Cada cromosoma se representa como un vector cuyos genes codifican directamente la selección de variables del modelo (columnas con valores de indicadores técnicos). Cada gen contiene el nombre y uno o varios parámetros de un indicador técnico diferente (véase 4.2.9), permitiendo que cada individuo represente un subconjunto único de variables.

Durante la operación de cruce (crossover), una máscara binaria determina, para cada posición, si el hijo hereda la feature del Padre 1 o del Padre 2, garantizando la diversidad y recombinación eficiente de las combinaciones de indicadores. La siguiente Figura 29 muestra el mecanismo de cruce uniforme aplicado sobre cromosomas codificados con nombres de features reales:

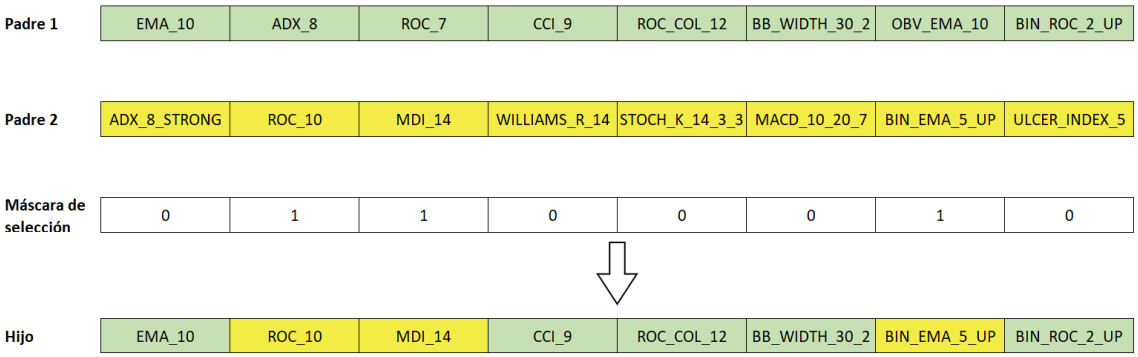


Figura 29 Cruce de cromosomas con una máscara binaria

En este proyecto el algoritmo genético ha sido implementado en Python para ofrecer una mayor flexibilidad y permitir futuras personalizaciones. El código puede consultarse en el Anexo 9.

4.2.15 Comprobación de eficiencia de algoritmo genético de selección de features

Para evaluar la eficacia del algoritmo genético (véase 4.2.14) en la selección de variables, se ha realizado un experimento sobre un conjunto de **7 activos** bursátiles de distintos sectores, comparando su rendimiento frente a la selección aleatoria utilizando modelo **XGBoost Regressor**. El tamaño de conjunto se ha limitado a **15 features**.

Para ambas estrategias (algoritmo genético y selección aleatoria), se ha empleado un esquema de validación cruzada **cronológica** de tipo K-Fold por años (véase Figura 30).

Figura 30. Validación cruzada (K-Fold) cronológica utilizada en test de eficiencia de GA

En este enfoque, el conjunto de datos de cada activo financiero se divide en **3 folds** anuales consecutivos. En cada fold, el modelo y la selección de variables se entrenan con todos los datos desde **2010** hasta el inicio de la ventana de validación interna (**Test GA**), que se utiliza solo para ajustar el algoritmo genético, sin emplearse para calcular métricas finales.

Finalmente, la evaluación definitiva del modelo se realiza sobre el año objetivo de cada fold (**Test Out Of Sample**). En Figura 31 y Figura 32 se muestran los gráficos comparativos de métricas RMSE y R2 obtenidas tanto entrenando el modelo XGBoost Regressor utilizando el conjunto de características aleatorias contra las seleccionadas por GA.

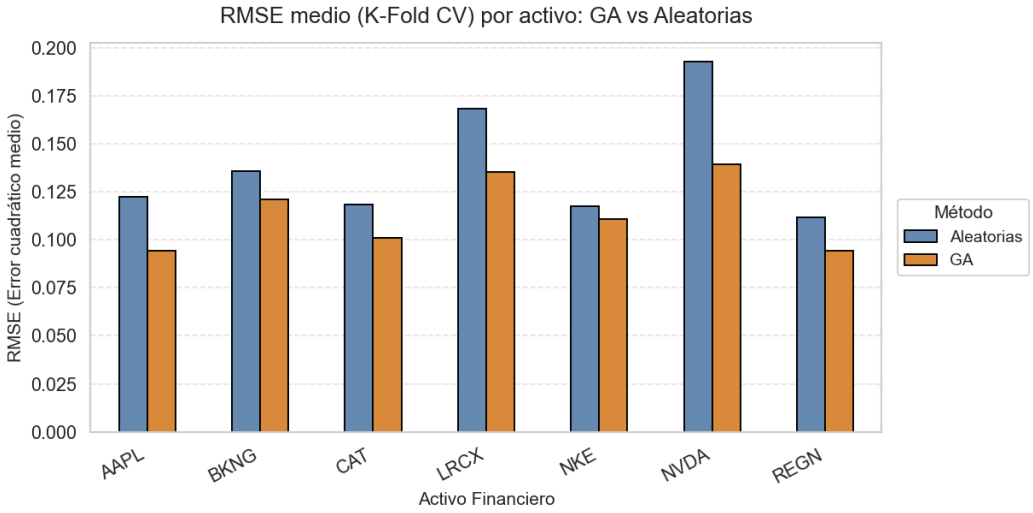


Figura 31. Métricas de RMSE XGBoost Regressor (aleatorias vs GA)

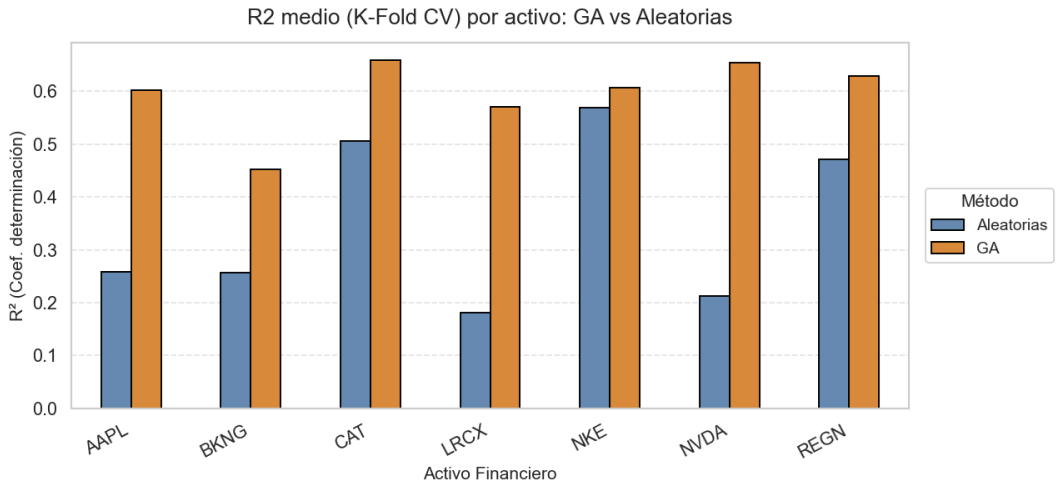


Figura 32. Métricas de R2 XGBoost Regressor (aleatorias vs GA)

Los experimentos de esta comparativa se han realizado con la siguiente configuración de GA:

Parámetro	Valor	Descripción breve
fitness_model	'XGBRegressor'	Modelo base para evaluar cada subconjunto de features.
fitness_metric	'rmse'	Métrica objetivo que se busca optimizar (Root Mean Squared Error en este caso).
n_pop	25	Número de individuos en cada generación.
n_gen	20	Número total de generaciones que se ejecutan.
elite	10	Número de mejores individuos que pasan automáticamente a la siguiente generación (elitismo).
mut_prob	0.5	Probabilidad de mutar cada gen (feature) de un individuo.
random_state	42	Semilla de aleatoriedad para obtener resultados reproducibles.
max_active	15	Número máximo de features permitidas activas en cada individuo.
min_active	15	Número mínimo de features activas en cada individuo (fijado a 15 para este experimento).
tournament_size	3	Número de individuos que compiten en la selección por torneo para elegir padres para el cruce.

Tabla 10. Configuración de parámetros de GA

La mejora media sobre los 7 activos financieros:

- RMSE medio GA: **0.120** vs RMSE medio de selección aleatoria: **0.140** (–**14.5** % de error)

- R^2 medio GA: **0.503** vs R^2 medio de selección aleatoria: **0.309** (+**63.0** % en calidad de predictibilidad)

En todos los casos el método GA produce modelos con menor error **RMSE** y mayor capacidad explicativa **R^2** que la selección de variables aleatoria, lo que refuerza la utilidad del algoritmo genético para la selección.

En la Figura 33 se muestra la evolución de las métricas R^2 y RMSE a lo largo de las generaciones del algoritmo genético (GA) aplicado a la acción de NVDA. Tras una fase inicial de mejora rápida, ambas métricas tienden a estabilizarse a partir de la generación 60, lo que indica una convergencia efectiva del proceso evolutivo.

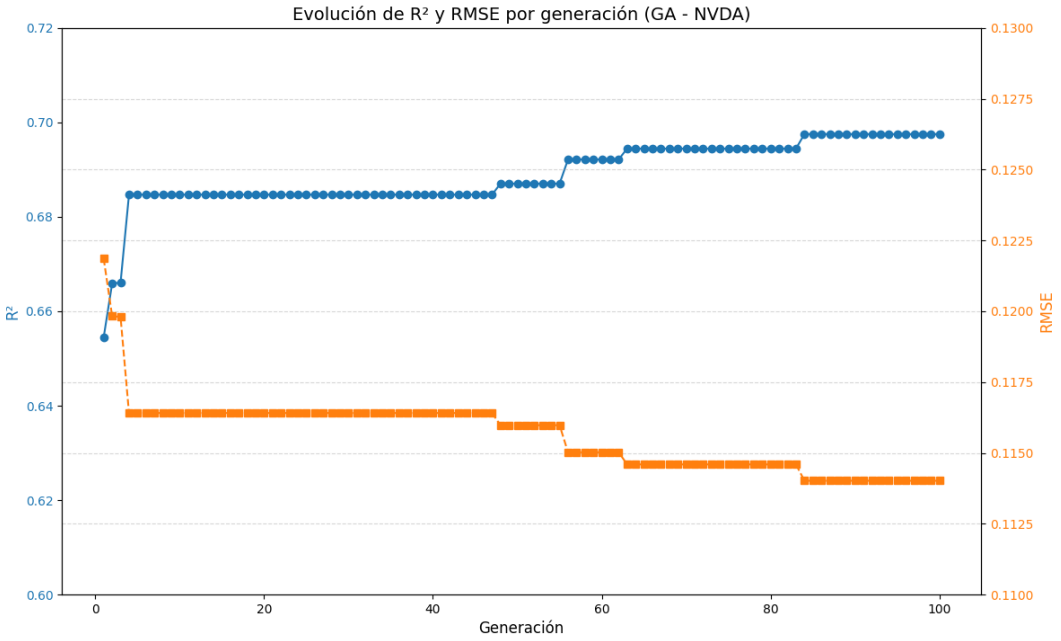


Figura 33. Evolución de métricas durante selección de features con GA para NVDA

El modelo final no solo ha logrado reducir la dimensionalidad, seleccionando únicamente 25 características de las 275 originales, sino que también ha conseguido mejorar ligeramente las métricas de rendimiento. Comparando con las métricas promediadas de utilizar todas las 275 características (véase Tabla 9) dónde se obtuvo un R^2 medio de 0,61 y un RMSE medio de 0,1 tras aplicar la selección genética, se han mejorado ligeramente: R^2 de 0,69 y el RMSE de 0,1.

Este resultado adquiere especial relevancia porque evidencia que no se trata únicamente de afinar métricas, sino de controlar la complejidad del modelo (pasar de **275** a **25** características), evitar el sobreajuste, y abrir la puerta al uso eficiente de otros algoritmos que son más sensibles a la cantidad de variables de entrada.

4.2.16 Perceptrón Multicapa (MLP)

Los perceptrones multicapa (MLP, Multi-Layer Perceptron) constituyen uno de los modelos neuronales más antiguos y ampliamente utilizados en la literatura de ML. Desde hace décadas se han aplicado a la predicción de series temporales financieras, demostrando su capacidad para aproximar relaciones no lineales y detectar patrones complejos.

Diversos trabajos han validado su eficacia en este ámbito, como el de Namdari y Durrani [23], que emplean un MLP para predecir tendencias bursátiles de corto plazo, o el de Peng et al. [2], donde se utilizan redes multicapa en combinación con indicadores técnicos para predecir la dirección de los precios.

Un perceptrón multicapa (MLP) está formado por neuronas organizadas en capas densamente conectadas (véase Figura 34 y Figura 35)**.** Cada neurona combina sus entradas ponderadas mediante pesos w con un sesgo (b o bias) y aplica una función de activación f

$$h = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

La propagación hacia adelante se formula capa a capa como:

$$h^{(l)} = f\left(W^{(l)} h^{(l-1)} + b^{(l)}\right), \qquad \hat{y} = g\left(W^{(L)} h^{(L-1)} + b^{(L)}\right)$$

donde f representa la función de activación (ReLU, sigmoide, tangente hiperbólica tanh, etc.) aplicada elemento a elemento al vector de la capa oculta, y g corresponde a la función de salida. El modelo se entrena minimizando una función de pérdida, en este caso el error cuadrático medio (MSE)**:**

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

La optimización de los parámetros se lleva a cabo mediante propagación hacia atrás (backpropagation), que aplica la regla de la cadena para calcular gradientes de la pérdida respecto a cada peso y sesgo. Dichos gradientes se actualizan con algoritmos como descenso estocástico del gradiente (SGD) o Adam, ajustando los parámetros (pesos w) paso a paso hasta minimizar el error.

En este trabajo, el perceptrón multicapa se incorpora como uno de los modelos de referencia para evaluar su desempeño dentro del marco de la selección genética de características. Los resultados obtenidos con el MLP se comparan posteriormente con otros algoritmos de ML como por ejemplo XGBoost Regressor, ElasticNet, Ridge, Lasso o SVR, lo que permite valorar de forma conjunta la robustez y eficacia de cada enfoque bajo las mismas condiciones experimentales

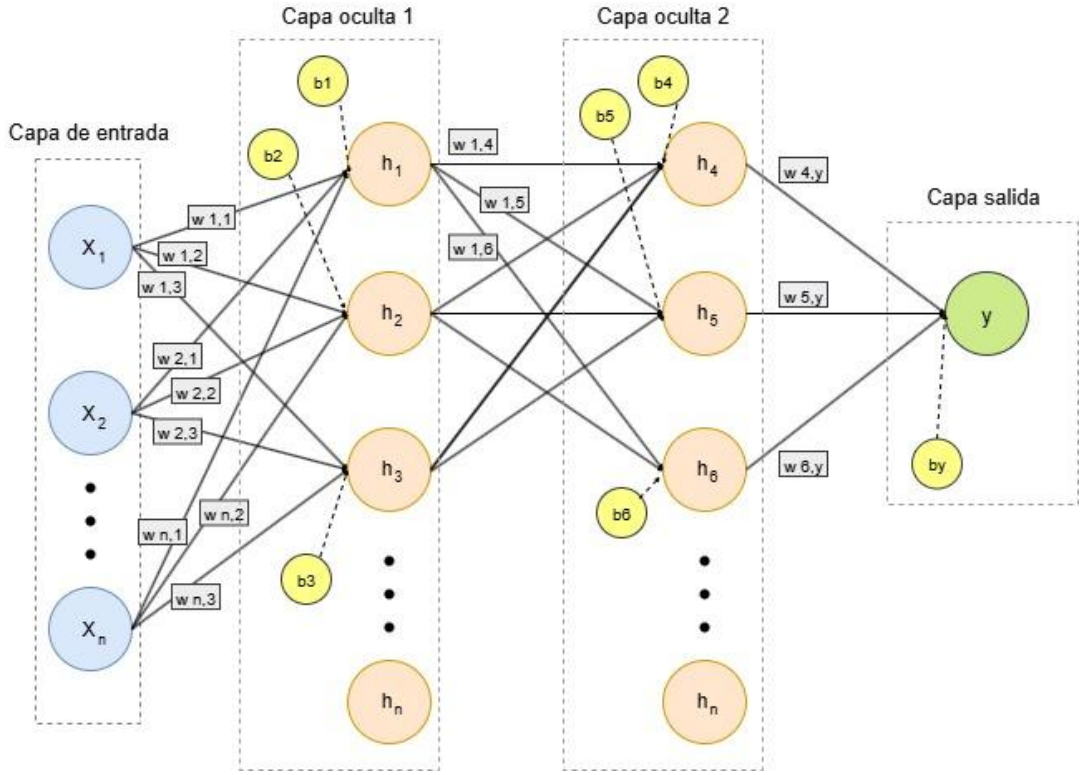


Figura 34. Esquema de red neuronal perceptrón multicapa (MLP).

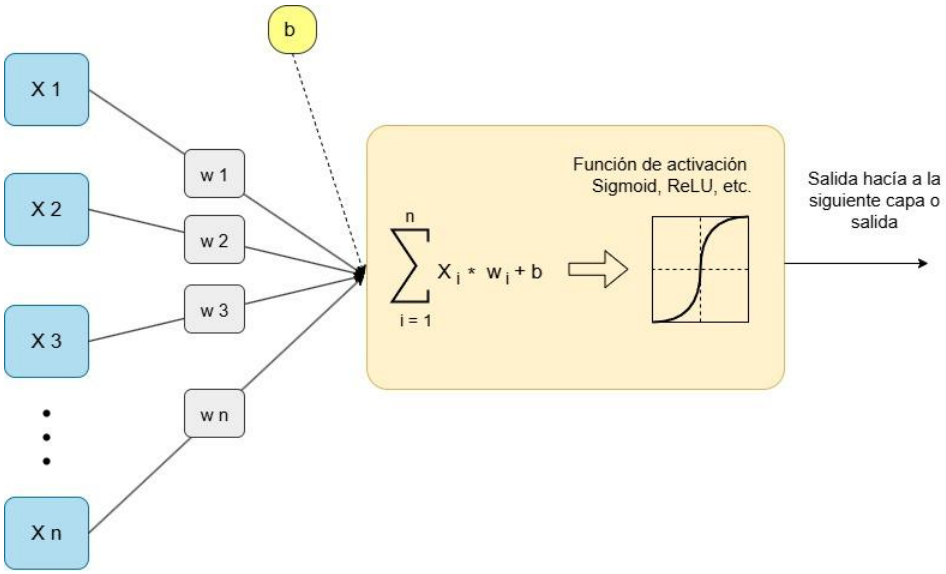


Figura 35. Estructura de una unidad (neurona) de MLP.

4.2.17 Evaluación de distintos modelos ML

Como se muestra en Figura 36 y Figura 37, además de XGBoost Regressor, se ha llevado a cabo un experimento para lograr una comparativa con distintos modelos disponibles en scikit-learn, tales como: ElasticNet, Ridge, Lasso, SVR, Regresión Lineal clásica y MLP (Perceptrón Multicapa) cada uno con **25 características** de entrada seleccionadas por GA. Esta diversidad de modelos permite evaluar no solo el rendimiento predictivo puro, sino también la robustez de la selección genética de características bajo diferentes supuestos y arquitecturas.

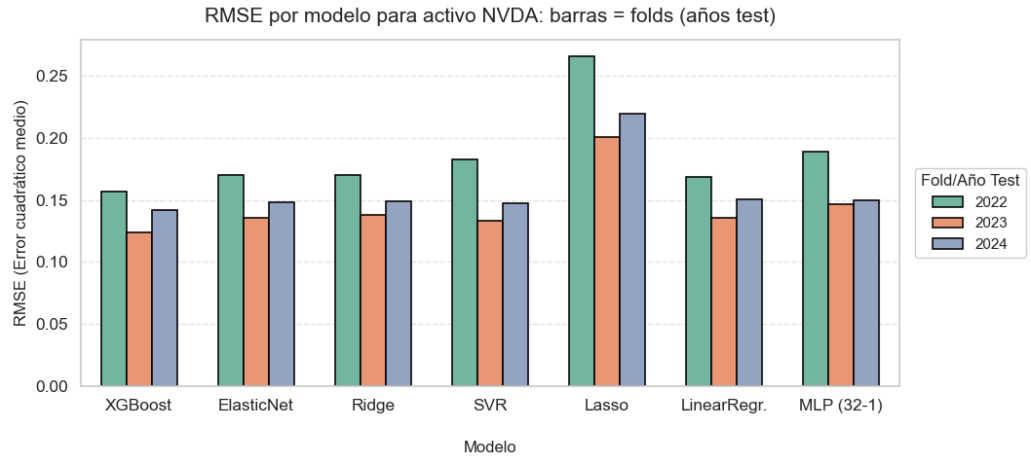


Figura 36. Comparativa de RMSE utilizando varios modelos ML con GA para el activo NVDA

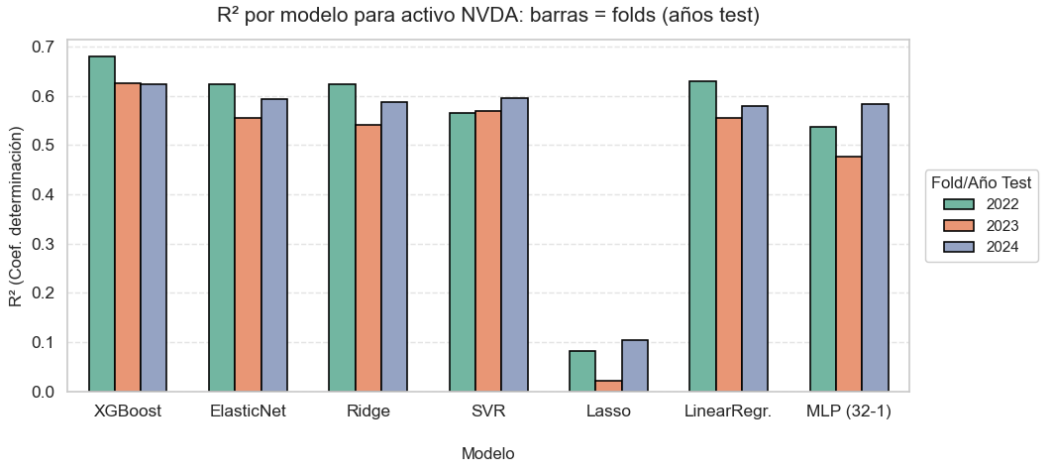


Figura 37. Comparativa de R2 utilizando varios modelos ML con GA para el activo NVDA

De la misma manera que en el experimento anterior (véase 4.2.15) se ha empleado una **validación cruzada temporal tipo K-Fold**, adaptada a la naturaleza secuencial de las series financieras. Cada fold se ha definido agrupando periodos consecutivos de entrenamiento y test, de acuerdo con la siguiente configuración:

Fold	Años de entrenamiento de modelo GA	Años de test de modelo GA	Año de validación final (crossval)
1	2010–2019	2020, 2021	2022
2	2010–2020	2021, 2022	2023
3	2010–2021	2022, 2023	2024

Tabla 11. Validación cruzada (K-Fold) cronológica al testear los modelos ML con GA

La configuración de los parámetros de algoritmo GA en este experimento (número de individuos, probabilidad de mutación, número de individuos de élite, etc.) es idéntica a la que se utiliza en 4.2.15 (véase Tabla 10).

Cabe destacar que, en la mayoría de los modelos evaluados, se han utilizado los parámetros por defecto proporcionados por scikit-learn, sin realizar ningún ajuste específico de hiperparámetros ni optimización avanzada de arquitectura.

El objetivo principal de este experimento ha sido **obtener una visión comparativa general** sobre el rendimiento relativo de distintas familias de modelos tras la reducción de dimensionalidad genética (25 características seleccionadas por GA), tomando como caso de estudio únicamente el activo **NVDA**. Por tanto, los resultados aquí mostrados no deben interpretarse como un ranking absoluto de rendimiento entre modelos, sino como una referencia exploratoria sobre la robustez de cada aproximación bajo condiciones homogéneas y realistas.

Tal y como indican, R. Namdari y T. S. Durrani et al. (2021) [23], el desempeño de las redes neuronales multicapa (MLP) en la predicción bursátil depende en gran medida de la arquitectura seleccionada y correcto ajuste de sus hiperparámetros, como el número de capas, neuronas y funciones de activación, lo que puede traducirse en diferencias notables de precisión entre configuraciones. Por ello es previsible que, mediante una búsqueda sistemática de hiperparámetros, modelos como el MLP o incluso otros algoritmos podrían alcanzar o superar el rendimiento de XGBoost Regressor configurado por defecto.

4.2.18 Impacto del modelo y la arquitectura hardware en la velocidad de entrenamiento del GA

Para cuantificar el efecto del modelo predictivo y el hardware utilizado, se ha realizado un test de rendimiento comparando el tiempo de entrenamiento de una generación completa del algoritmo genético bajo diferentes configuraciones (CPU multinúcleo, GPU y modelos base), así como distintos tamaños de dataset de entrenamiento **de 1.000 a 10.000 filas** (véase Figura 38).

En particular, la evaluación de modelo MLP (véase 4.2.16) presentó un reto significativo: la implementación estándar de **MLPRegressor** en scikit-learn no soporta paralelización ni entrenamiento en GPU, lo que imposibilita un uso eficiente de los recursos computacionales en escenarios intensivos como los que requiere el algoritmo genético.

Los resultados experimentales demuestran que el entrenamiento de una sola generación del GA (25 individuos, dataset de 10.000 filas y 25 características) con MLPRegressor de scikit-learn sobre CPU requiere aproximadamente **25 segundos** por generación (véase Figura 38). Si se deseara ejecutar un ciclo completo de optimización genética de 60 generaciones, el tiempo total superaría **25 minutos** (60 × 25 s = 1.500 s), lo que resulta impráctico incluso con hardware de altas prestaciones.

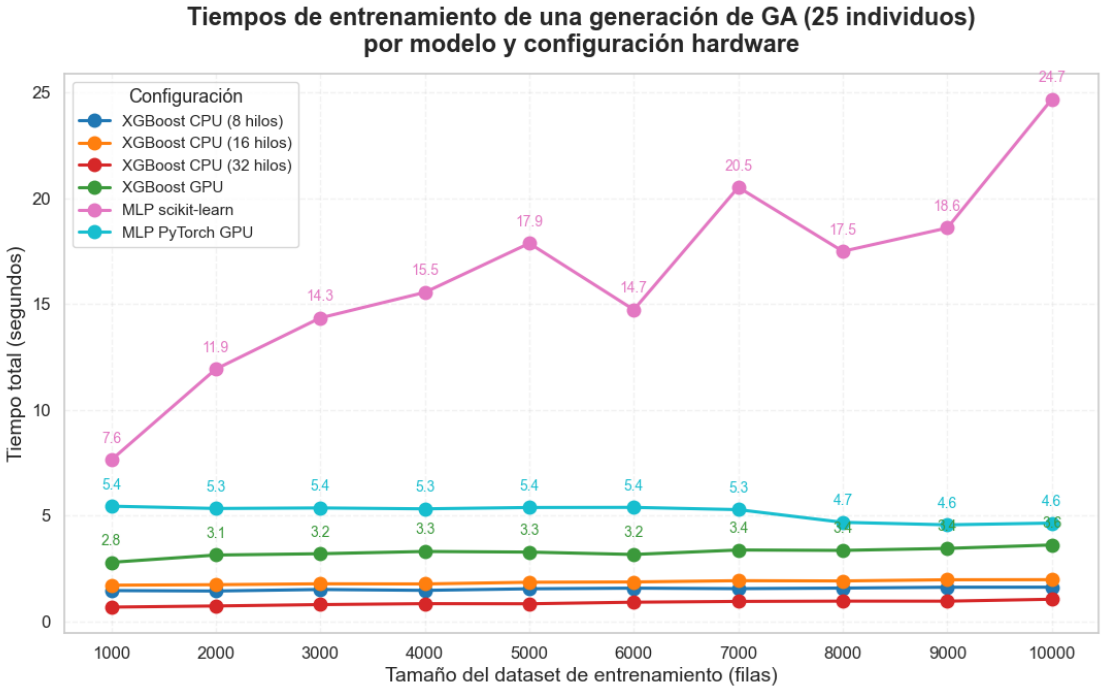


Figura 38 Latencias de entrenamiento GA por modelo y hardware

Para solventar este cuello de botella, se ha desarrollado e integrado un **MLP personalizado basado en PyTorch** (véase Anexo 10), específicamente diseñado para aprovechar la aceleración por GPU mediante **CUDA**. Esta arquitectura consiste en una sola capa oculta de 32 neuronas y una neurona de salida, con función de activación sigmoide en ambas capas. Se ha elegido la regularización **dropout** (20%) para mitigar el sobreajuste.

Además, se aplica penalización L2 (con weight_decay=0.0003 en el optimizador Adam) como regularización adicional sobre los pesos del modelo. El entrenamiento se realiza optimizando el error cuadrático medio (MSE) como función de pérdida. La tasa de aprendizaje utilizada es de 0.001.

Gracias a esta optimización, una generación completa del GA puede evaluarse en apenas **3,6 segundos** usando una GPU NVIDIA RTX 2080 Super. De este modo, entrenar 60 generaciones requiere tan solo unos **3 minutos y 36 segundos** (60 × 3,6s = 216 s), lo que permite realizar experimentos exhaustivos en tiempos razonables y sin renunciar a la rigurosidad metodológica.

De forma complementaria, el algoritmo **XGBoost Regressor** demostró un rendimiento excepcional tanto en CPU multi-hilo como en GPU, logrando tiempos por generación inferiores a **2 segundos** incluso con los datasets más grandes del experimento (10.000 filas). Resulta llamativo que, en este caso, la CPU multi-hilo (32 hilos) supera en velocidad a la GPU, que necesita unos **3,6 segundos** por generación.

Este resultado se explica porque, al trabajar con volúmenes de datos relativamente pequeños, la sobrecarga de transferir datos entre la memoria RAM y la memoria de la GPU para cada individuo del GA puede compensar la ventaja del paralelismo de la tarjeta gráfica. Además, los procesadores modernos cuentan con frecuencias de reloj mucho más altas que las GPUs (5,7 GHz vs 1,65 GHz) y aprovechan mejor la caché interna y la paralelización por hilos, especialmente en tareas que requieren lanzar muchos entrenamientos secuenciales y no saturan por completo la GPU.

Especificaciones del sistema de pruebas:

CPU: Intel Core i9-14900K (24 núcleos, 32 hilos, frecuencia máxima - 5,7 GHz)

Memoria RAM: 64 GB DDR5 a 4800 MHz, 6000 MT/s = 6.000 millones de transferencias por segundo, por canal.

GPU: NVIDIA RTX 2080 Super RAM: 8 GB, Núcleos CUDA: 3072 Arquitectura: Turing TU104,

En conclusión, se priorizó el uso de XGBoost para los experimentos principales, aunque se mantuvieron las pruebas con MLP-PyTorch para comparar robustez y viabilidad de arquitecturas neuronales más avanzadas. Como muestra la Figura 39, incluso con datasets de hasta 80.000 filas, el entrenamiento de una generación completa del algoritmo genético se mantiene en márgenes razonables para XGBoost Regressor, tanto en CPU multinúcleo como en GPU, y para MLP PyTorch sobre GPU.

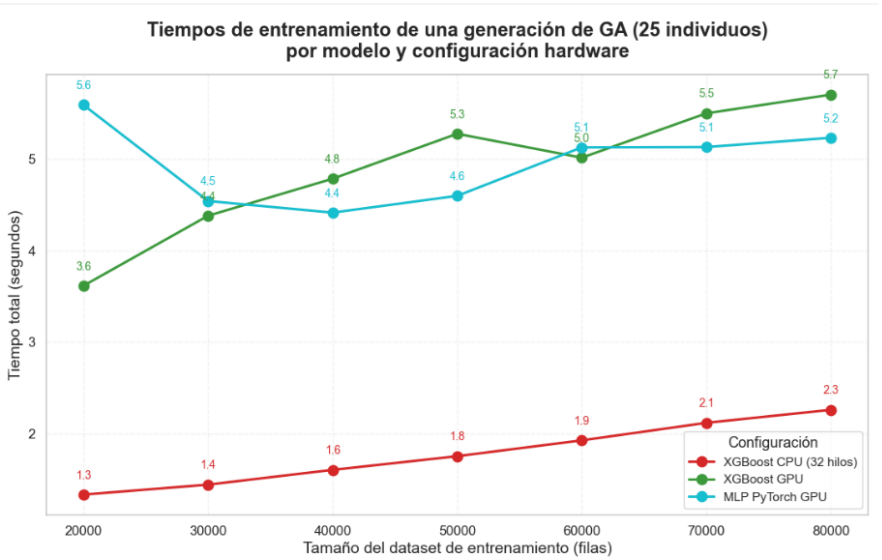


Figura 39 Latencias de entrenamiento de modelos con GA sobre datasets más grandes

4.2.19 Análisis de la influencia del número de variables y tamaño de dataset en la fase GA sobre la capacidad predictiva fuera de muestra

Como se muestra en la Figura 40, se ha analizado cómo influye el número de variables seleccionadas y el tamaño del conjunto de datos en la capacidad predictiva de modelo XGBoost Regressor, promediando las métricas obtenidas para los **7 activos financieros**: AAPL, MSFT, NVDA, CAT, NKE, REGN y BKNG.

El proceso realiza un filtrado supervisado por correlación de Pearson: se eliminan las variables cuya correlación con el target excede **0,6** siguiendo la estrategia de Saidi et al. [24]. Esta medida previene el sobreajuste y evita que el modelo base sus predicciones únicamente en características excesivamente correlacionadas con la variable objetivo, favoreciendo así la selección de características que aporten información más genuina y diversa.

El pipeline experimental se desarrolla en dos etapas principales:

- **Entrenamiento y validación interna con GA**: Para cada combinación de número de características y tamaño de muestra, el algoritmo genético selecciona el subconjunto óptimo de variables, entrenando en los años 2010–2019 y validando en los años 2020–2022.
- **Test fuera de muestra (out-of-sample)**: El modelo definitivo, construido con las variables seleccionadas, se entrena hasta 2022 y se evalúa sobre los datos de 2023–2024, que no se han utilizado en ninguna fase previa. Las métricas promediadas para los 7 stocks R^2, MSE, RMSE y MAE obtenidas en este test final (véase Figura 40) permiten comparar objetivamente el rendimiento de cada configuración.

Los resultados evidencian que el mejor rendimiento se logra seleccionando entre aproximadamente **12** y **35** variables y utilizando entre **30.000** y **50.000** muestras para entrenar el modelo. Añadir más características o aumentar el tamaño del dataset por encima de esos valores no proporciona mejoras significativas e incluso puede degradar la capacidad predictiva fuera de muestra.

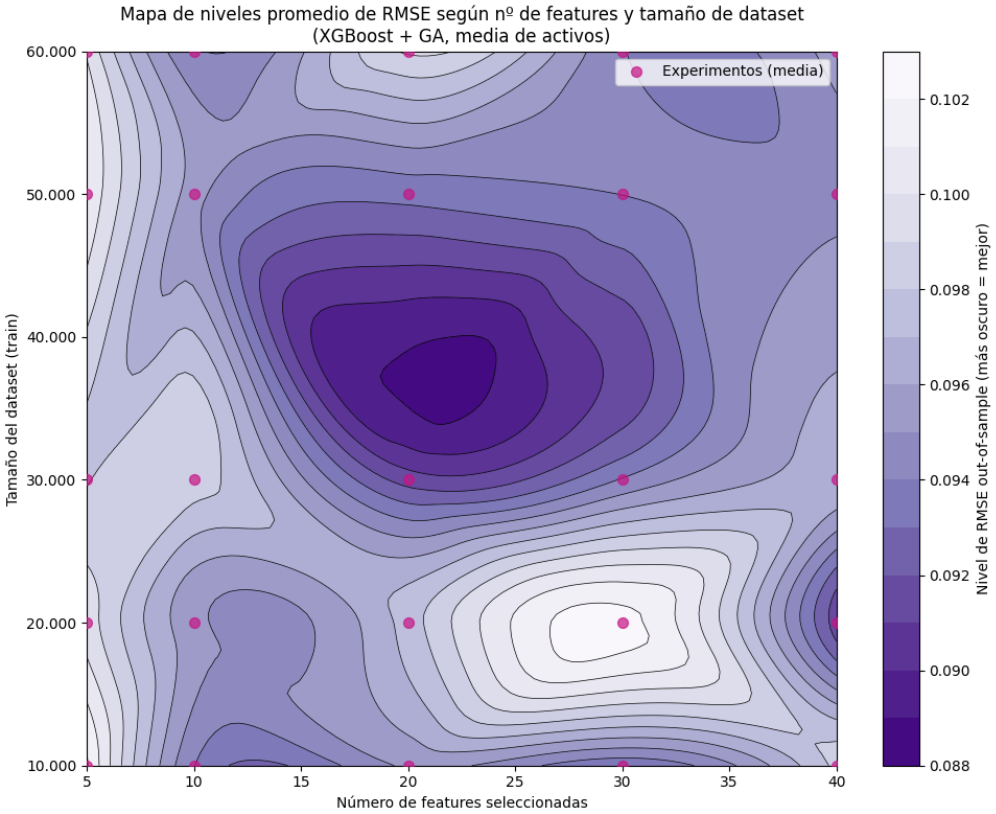


Figura 40. RMSE medio de modelo GA en función de tamaño del dataset

4.2.20 Análisis de características más influyentes en selección GA

Para identificar las variables más relevantes en la predicción de tendencias de precios, independientemente de condiciones de contexto como: tamaño del dataset, el activo financiero, el número de features a seleccionar, años de test y la semilla de aleatoriedad, se diseñó un experimento exhaustivo.

El procedimiento consistió en aplicar el GA sobre todos los stocks de la cartera diversificada de **25 activos** seleccionados previamente (véase 4.2.6), utilizando como variable objetivo el ángulo de tendencia **TARGET_TREND_ANG_15_5** con horizonte de predicción de 5 días. Las ejecuciones se realizaron bajo distintas condiciones de test, abarcando múltiples ventanas temporales te test ([2018–2019], [2018–2020], [2018–2021]) y repitiendo el proceso para cada activo.

En total se han ejecutado **75 selecciones** completas de features mediante GA dónde en cada iteración, se seleccionó aleatoriamente el número de variables a seleccionar entre **18** y **35**, así como el tamaño de dataset de entrenamiento (de **30.000** a **50.000** muestras) y la semilla de aleatoriedad. El modelo base del fitness fue un **XGBRegressor** optimizado con RMSE y la configuración interna del GA se fijó en 25 individuos por población, 20 generaciones, elitismo del 40% y tasa de mutación del 0,5.

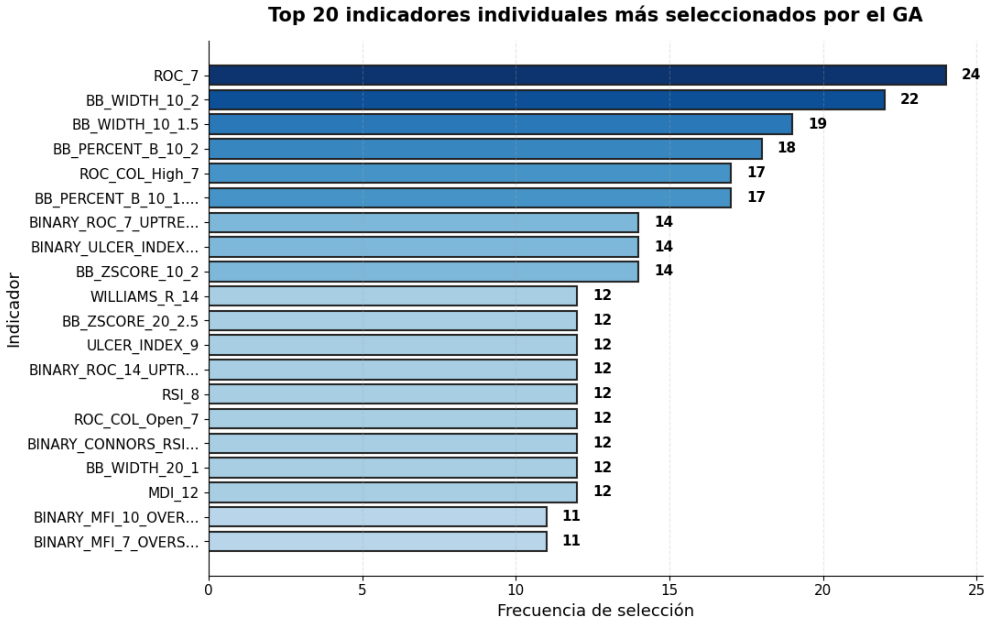


Figura 41. Top 20 indicadores técnicos seleccionados por GA

El análisis global de frecuencias revela que ciertos indicadores emergen sistemáticamente como los más frecuentes seleccionados por el algoritmo genético (véase Figura 41).

- **ROC_7** (Rate of Change a 7 periodos) - tasa de cambio, es un oscilador de momentum que mide la velocidad del cambio en el precio fue seleccionado en 24 ejecuciones.
- **BB_WIDTH_10_2 y BB_WIDTH_10_1.5** Ancho de las Bandas de Bollinger siendo el indicador técnico que representa la volatilidad relativa del precio respecto a una media móvil con 10 periodos, 2 y 1.5 desviaciones, con 22 y 19 apariciones respectivamente.
- Diversos indicadores derivados de Bandas de Bollinger (**BB_PERCENT_B_10_2, BB_PERCENT_B_10_1.5, BB_ZSCORE_10_2**, etc.).
- Señales binarias de tendencia o reversión como **BINARY_ROC_7_UPTREND** y **BINARY_ULCER_INDEX_14_UP** - Indicador técnico que mide el riesgo a la baja de un activo financiero, evaluando la profundidad y duración de sus caídas de precio.
- Medidas de fuerza relativa como **RSI_8 y WILLIAMS_R_14**

Estos resultados reflejan una preferencia clara del GA por indicadores de momento (miden la velocidad y la fuerza con la que cambian los precios), volatilidad y reversión de corto plazo, especialmente aquellos que capturan expansiones de volatilidad local y cambios bruscos en la dinámica del precio. Esto sugiere la capacidad de algoritmo genético para identificar de forma consistente los indicadores que realmente aportan valor en la predicción de tendencias de corto plazo.

4.2.21 Selección de características y ajuste de hiperparámetros

Conforme varios estudios, uno de ellos Chalvatzis y Hristu-Varsakelis (2025) [25], se considera preferible **entrenar un modelo predictor por activo** en vez de un único modelo para varios activos, dada la heterogeneidad entre series; por ello, aquí se adopta un enfoque multi-modelo con selección de características y ajuste por activo.

Basándose en los hallazgos de experimentos anteriores se diseñó un flujo de trabajo que consta de **tres etapas** para entrenar modelos específicos por cada activo financiero y generar predicciones estrictamente fuera de muestra, a partir de las cuales se aplicarán las estrategias de trading.

En primer lugar, se realiza la selección de características por activo mediante un algoritmo genético en configuración wrapper, utilizando modelo **XGBoost Regressor** como función de aptitud. (véase Figura 42). Desde un universo inicial de **275 características** de dataset completo se seleccionan 25 para cada modelo de activo financiero, aplicando train test split cronológico y reduciendo la dimensionalidad. Los parámetros de GA son idénticos a los de experimentos anteriores (véase Tabla 10)

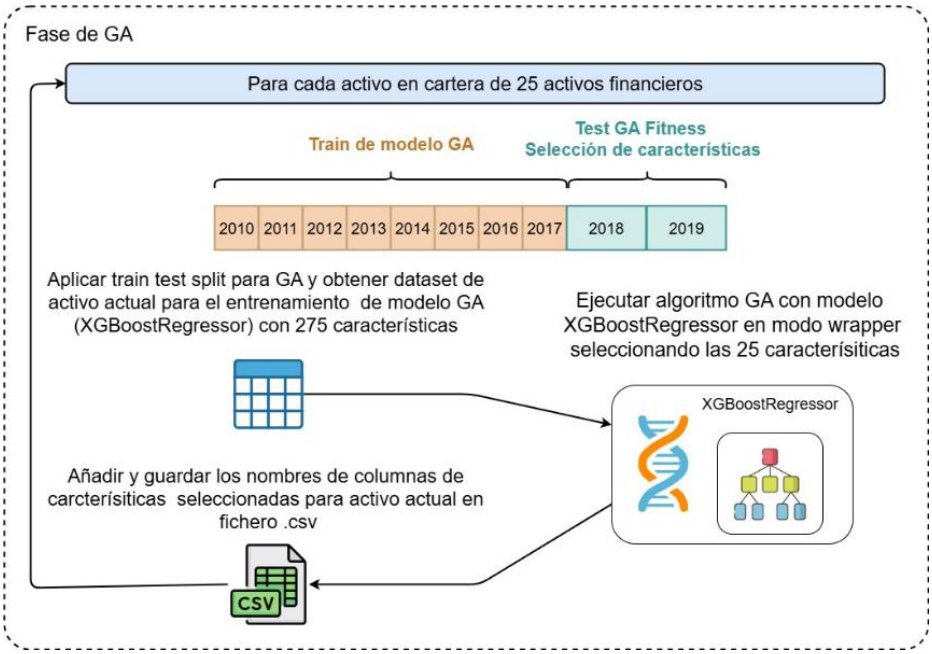


Figura 42. Fase de selección de características con GA

En segundo lugar, como se muestra en Figura 43, con ese subconjunto fijado, se lleva a cabo el ajuste de hiperparámetros de cada modelo mediante **RandomizedSearchCV** con validación cruzada temporal deslizante. La descripción hiperparámetros a ajustar se muestra en la Tabla 12.

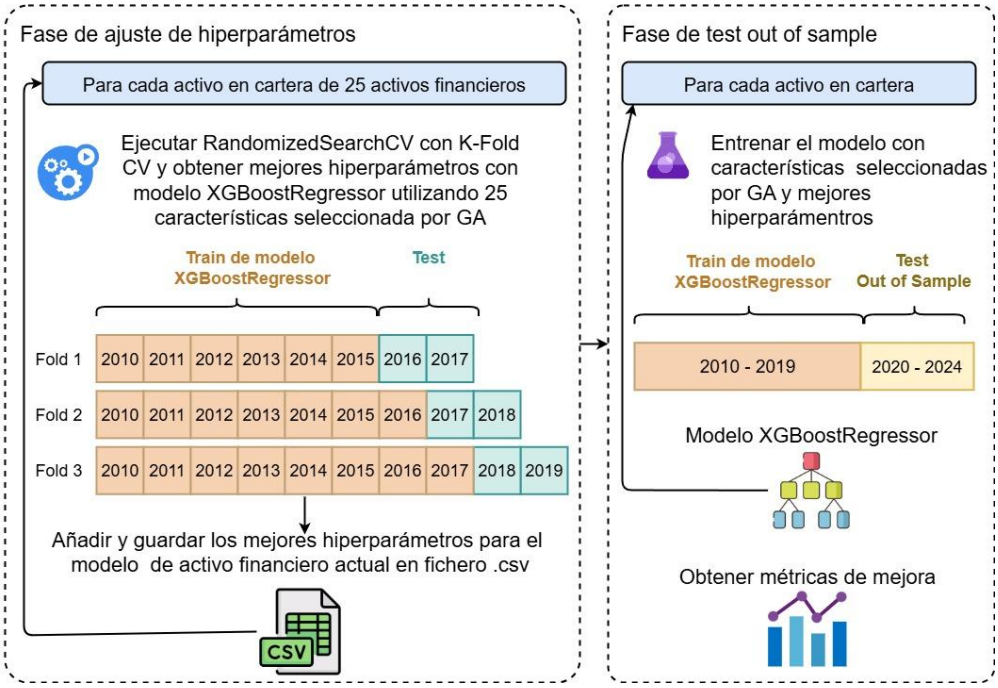


Figura 43. Fase de ajuste de hiperparámetros de modelos y test

Finalmente, para cada activo, se entrena el modelo definitivo con el histórico precedente y se evalúa sobre el período out-of-sample (años 2020 - 2024).

Como parte de diseño se guardan en ficheros .csv por activo tanto las características seleccionadas como los hiperparámetros óptimos. Esta decisión aporta reproducibilidad y permite ensayar distintas estrategias de trading y variantes de backtesting sin repetir los procesos costosos de selección y tuning.

Hiperparámetro	Valores	Descripción
n_estimators	50, 100, 150, 200, 250	Número de árboles de decisión que se construyen secuencialmente. Más árboles suelen mejorar el ajuste, pero incrementan el riesgo de sobreajuste y el tiempo de entrenamiento.
max_depth	3, 4, 5, 6	Profundidad máxima de cada árbol. Valores mayores permiten capturar interacciones más complejas, pero aumentan el riesgo de sobreajuste.
learning_rate	0.01, 0.03, 0.05, 0.10	Tasa de aprendizaje o “eta”. Escala la contribución de cada árbol; valores pequeños requieren más árboles, pero tienden a mejorar la generalización.

subsample	0.70, 0.85, 1.00	Porcentaje de muestras de entrenamiento usadas en cada árbol. Valores menores a 1 introducen aleatoriedad y ayudan a evitar sobreajuste.
colsample_bytree	0.70, 0.85, 1.00	Porcentaje de características usadas en la construcción de cada árbol. Favorece la diversidad de los árboles y reduce sobreajuste.

Tabla 12. Descripción de hiperparámetros

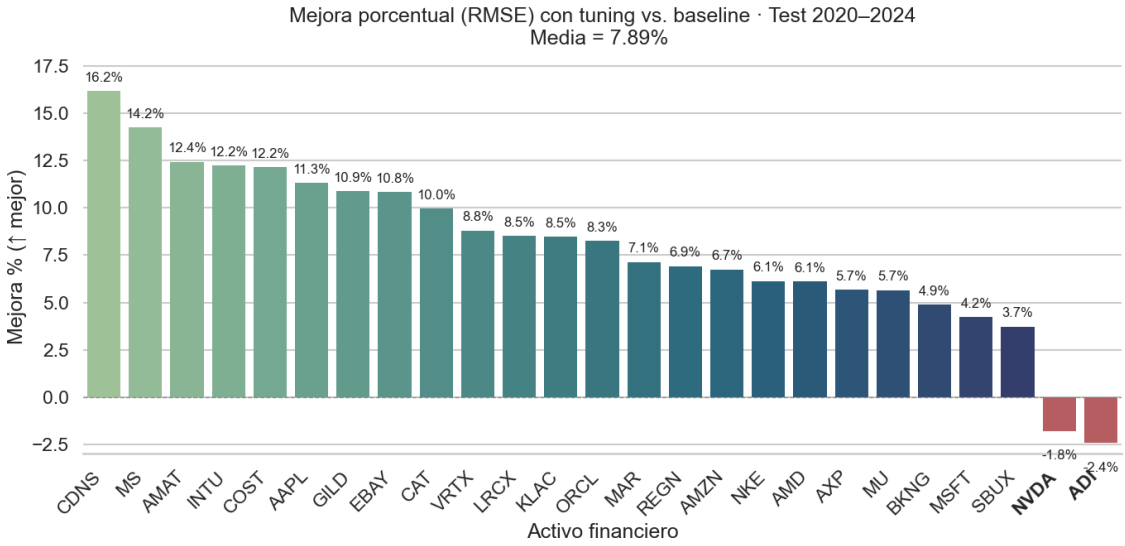


Figura 44. Mejora porcentual del RMSE tras el ajuste de hiperparámetros

Tras aplicar el ajuste de hiperparámetros se ha hecho la evaluación de la mejora porcentual en RMSE del modelo ajustado frente al modelo sin ajustar, manteniendo las 25 características seleccionadas por GA previamente y sobre el periodo out of sample, años 2020–2024.

Como se muestra en Figura 44, el ajuste reduce el **RMSE** en promedio un **7,89%** a nivel de cartera. La mejora es generalizada (23 de 25 activos), con máximos en CDNS (+16,2%), MS (+14,2%), AMAT (+12,4%), INTU (+12,2%) y COST/AAPL (~+12%). En el extremo opuesto, se observan ligeros empeoramientos en NVDA (−1,6%) y ADI (−2,4%), dentro de un margen acotado.

El ajuste de hiperparámetros aportó ganancia sistemática y robusta en error de predicción fuera de muestra, con variabilidad moderada entre activos. Dado el balance coste/beneficio, se adoptan los hiperparámetros ajustados por activo (guardados en ficheros .csv) como configuración por defecto para los experimentos posteriores y para las pruebas de backtesting de las estrategias de trading. En el Anexo 11 se puede consultar la implementación completa de este flujo de trabajo.

4.3 Recursos requeridos

El desarrollo del proyecto requirió un conjunto de recursos materiales, técnicos y formativos que permitieron su ejecución en condiciones óptimas. A continuación, se enumeran los principales:

- **Estación de trabajo de alto rendimiento (Intel i9, 64 GB RAM, GPU RTX 2080 Super)**: equipo principal para el desarrollo de los experimentos, entrenamientos de modelos, pruebas de *feature engineering* y ejecuciones intensivas del algoritmo genético.
- **Portátil HP Laptop 15-fd0082ns**: utilizado para documentación, redacción de la memoria y desarrollo de código auxiliar que no requería pruebas computacionales voluminosas.
- **Monitor LG 32'' 4K**: pantalla de apoyo para análisis simultáneo de resultados, visualización de métricas y elaboración de gráficos.
- **Microsoft Office Professional Plus 2019**: procesador de textos y hojas de cálculo empleados para la redacción de la memoria, elaboración de tablas y gráficos.
- **Bibliografía académica (Elsevier/ScienceDirect)**: artículos adquiridos para reforzar el marco teórico y la validación metodológica.
- **Cursos online (Udemy)**: formación complementaria en *machine learning*, trading cuantitativo y Python aplicada en fases iniciales.
- **Libros especializados (~15 títulos)**: literatura técnica de referencia en ML, trading y Python, entre los que destacan: *Python Data Science Handbook* (O'Reilly), *Deep Learning con Keras y PyTorch*, *Machine Learning for Algorithmic Trading* (Packt), *Python for Data Analysis* (O'Reilly), y *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow* (O'Reilly).
- **Librerías Python (scikit-learn, PyTorch, XGBoost, pandas, numpy, etc.)**: base del desarrollo, modelado y análisis.
- **Visual Studio Code, Google Colab, GitHub, draw.io**: herramientas para la programación, pruebas distribuidas, control de versiones y diagramación.
- **Datos históricos bursátiles (Yahoo Finance vía yfinance)**: fuente principal para la construcción del dataset y la simulación de backtesting.
- **Electricidad**: consumo derivado del uso intensivo de la estación de trabajo durante los experimentos.
- **Coworking**: espacio de trabajo complementario empleado para tutorías y sesiones de concentración.
- **Desplazamientos locales**: transporte asociado a reuniones, tutorías y acceso a recursos externos.

4.4 Presupuesto

Tipo de coste	Concepto	Valor	Comentarios
Recursos humanos	Científico de Datos (300 h × 25 €/h)	7.500 €	Modelado, feature engineering y validación experimental. Dedicación parcial ≈25 h/semana durante 3 meses.
	Ingeniero de Datos (300 h × 22 €/h)	6.600 €	Desarrollo de pipelines, ETL, EDA, gestión de datos y optimización. ≈25 h/semana durante 3 meses.
Material (equipo técnico)	Estación de trabajo de alto rendimiento (Intel i9, 64 GB RAM, SSD 1 TB, GPU RTX 2080 Super)	2.000 €	Equipo principal de pruebas y GA.
	Portátil HP Laptop 15-fd0082ns (Intel Core i5, 16 GB RAM, SSD 512 GB)	800 €	Equipo auxiliar para desarrollo y movilidad.
	Monitor LG 32'' 4K	400 €	Pantalla de apoyo para visualización de resultados.
Software y formación	Microsoft Office Professional Plus 2019	30 €	Licencia de productividad.
	Bibliografía académica (Elsevier/ScienceDirect)	29 €	Artículos adquiridos para el marco teórico.
	Cursos online Udemy (5 cursos × 20 €)	100 €	Formación en ML, trading y Python.
	Libros especializados (~15 títulos × 40 €)	600 €	Literatura técnica en ML, trading y Data Science.
	Librerías Python (scikit-learn, PyTorch, XGBoost, pandas, numpy, etc.)	0 €	Herramientas open source.
	Visual Studio Code, Google Colab, GitHub, draw.io	0 €	Software educativo y gratuito.

Tipo de coste	Concepto	Valor	Comentarios
Gastos indirectos	Electricidad (3 meses intensivos de uso)	150 €	Estimación de consumo energético adicional.
	Coworking (3 meses × 50 €/mes)	150 €	Espacio de trabajo compartido utilizado para el desarrollo del proyecto, reuniones, etc.
	Desplazamientos locales (tutorías, biblioteca, coworking)	100 €	Transporte urbano y pequeños gastos asociados.
TOTAL ESTIMADO		≈ 18.500 €	Valor redondeado del presupuesto global del proyecto.

Tabla 13 Presupuesto de proyecto

4.5 Viabilidad

El proyecto resulta viable técnica y económicamente. Con un coste teórico estimado de ≈18.500 € (véase Tabla 13), principalmente asociado a recursos humanos, se ha logrado desarrollar un sistema de predicción de tendencias bursátiles empleando recursos propios del autor y herramientas de libre acceso. El hardware de altas prestaciones utilizado fue de propiedad personal, aunque se incluyó en el presupuesto como coste teórico, garantizando la ejecución sin necesidad de adquirir nueva infraestructura ni contratar servicios externos de computación.

En el plano técnico, el sistema ha demostrado ser reproducible, escalable y flexible. La infraestructura disponible permitió ejecutar experimentos de gran volumen (backtesting, algoritmos genéticos y redes neuronales) y adaptarse a nuevos datos o modelos sin costes adicionales significativos. Al estar implementado en Python, el pipeline resulta fácilmente integrable en entornos de producción de machine learning, plenamente compatibles con librerías estándar como scikit-learn, pandas o PyTorch, lo que asegura su vigencia y portabilidad a futuro. En cuanto a la sostenibilidad, el pipeline diseñado puede emplearse como soporte en la toma de decisiones de inversión y como sistema de predicción automatizada de tendencias, cuya eficiencia se validó en los experimentos de backtesting. Su modularidad facilita ampliar el número de activos analizados, incorporar nuevos indicadores técnicos o arquitecturas de machine learning, e integrarse con relativa facilidad en plataformas de trading o entornos de análisis cuantitativo.

En conclusión, el proyecto no solo es viable en el corto plazo, sino que constituye una base sólida para aplicaciones futuras, con capacidad de generar valor real en la toma de decisiones financieras al aportar un sistema eficiente, replicable y adaptable a distintos escenarios del mercado.

4.6 Resultados del proyecto

4.6.1 Evaluación de modelos predictivos con métricas de acierto direccional

En anteriores fases de proyecto se obtuvieron resultados satisfactorios en métricas de regresión como R^2, RMSE o MAE, lo que demuestra la capacidad de los modelos para aproximar la variable objetivo en términos estadísticos. Sin embargo, para evaluar su utilidad desde la perspectiva de un inversor es necesario comprobar si las predicciones se traducen en movimientos direccionales explotables en el mercado.

En esta línea, Costantini et al. (2016) evidencian que, en modelos de predicción de tipo de cambio, la evaluación basada solo en errores de predicción es insuficiente, siendo más relevante la **exactitud direccional** y la rentabilidad de las señales derivadas [26]. Siguiendo esta motivación, se introduce la métrica de *Acierto Direccional durante la Ventana con Umbral Fijo (DA)*, concebida como una adaptación práctica de la precisión direccional clásica.

El procedimiento es el siguiente: el modelo genera un score continuo entre 0 y 1, que en nuestro caso corresponde a la proyección del ángulo de la tendencia normalizado (véase 4.2.8). Esta variable objetivo se construyó de forma que:

- Un valor **0.5** indica **ausencia de tendencia** (plano).
- Valores cercanos a **1.0** representan una pendiente positiva fuerte, equivalente a un ángulo de aproximadamente **+75°** (tendencia alcista pronunciada).
- Valores cercanos a **0.0** representan una pendiente negativa fuerte, equivalente a **–75°** (tendencia bajista pronunciada).

A partir de este score se aplican dos **umbrales de decisión fijos** para transformar la salida continua en una señal discreta:

Si el score ≥ **0.6**, se interpreta como predicción de **subida (UP)**.

Si el score ≤ **0.4**, se interpreta como predicción de **bajada (DOWN)**.

Valores intermedios (0.4 < score < 0.6) se consideran **zona muerta**, donde no se emite señal para evitar decisiones en contextos de indecisión.

El criterio de acierto se define así: dado un precio de entrada Pt , la predicción es correcta si, en alguno de los cinco días siguientes, el precio de cierre se mueve en la dirección indicada y supera un umbral fijo de **±1,0 %.**

- Una señal de **subida (UP)** es acierto si el máximo alcanza al menos $Pt \times (1 + 0{,}01)$
- Una de **bajada (DOWN)** es acierto si el mínimo desciende hasta $Pt \times (1 - 0{,}01)$.
- Si el modelo emite una señal **UP** y no se cumple la condición anterior, se considera **fallo (falso positivo)**.

- Si el modelo emite una señal **DOWN** y no se cumple la condición correspondiente, se considera **fallo (falso negativo)**.
- Cuando el score del modelo se sitúa en la **zona muerta** (0,4 < score < 0,60), **no se evalúa la predicción**, ya que no se emite señal.

Del análisis de las variaciones a cinco días (véase Tabla 14) se aprecia que los movimientos típicos de los activos se sitúan en un rango de ≈ 1–2 % a 4–8 % (Q1–Q3), mientras que los extremos van desde ±0,2 % hasta ±14 %. La elección de un umbral reducido mínimo del 1 % para validar una predicción se justifica plenamente: descarta ruidos pequeños y, a la vez, permite capturar la mayoría de movimientos significativos reflejados en la tabla.

La variación acumulada en un horizonte de 5 días se define como:

$$|R_{t,5}| = \left|\frac{Close_{t+5} - Close_t}{Close_t}\right|$$

donde $(Close_t)$ es el precio de cierre en el día $día(t)$.

Activo	Media	Mediana	STD	Mínimo (5%)	1er cuartil (Q1)	3er cuartil (Q3)	Máximo (95%)
NVDA	5,62%	4,46%	4,61%	0,35%	2,19%	8,09%	14,71%
MSFT	2,87%	2,37%	2,33%	0,22%	1,17%	4,01%	7,40%
MU	4,88%	3,90%	4,15%	0,33%	1,85%	6,71%	12,95%
AAPL	3,13%	2,45%	2,65%	0,21%	1,23%	4,38%	8,13%

Tabla 14. Variaciones acumuladas de precio cierre a 5 días (2020–2024)

Se ha analizado la exactitud direccional de modelos entrenados con datos de 4 activos financieros: NVDA, MSFT, MU y AAPL dónde la metodología se planteó bajo un esquema estrictamente fuera de muestra (*out-of-sample*). Cada modelo se entrenó de manera independiente en el período **2010–2019**, incluyendo la selección de variables mediante algoritmo genético y el ajuste de hiperparámetros (véase 4.2.21). La evaluación de la precisión direccional se realizó en el intervalo **2020–2024**, reservado exclusivamente para test, garantizando resultados no sesgados por procesos de optimización y más próximos al comportamiento real de mercado.

Los resultados presentados en las matrices de confusión (Figura 45) y en los **informes de clasificación** (Tabla 15) reflejan un comportamiento realista y coherente con la naturaleza de los mercados financieros. Se observa que las métricas varían entre activos, lo que evidencia que la dificultad de predicción no es homogénea y depende en gran medida de la dinámica propia de cada activo financiero.

En **NVDA** se obtienen los resultados más consistentes (accuracy ≈ 0.72), lo que muestra cierta fiabilidad en la detección de tendencias alcistas. **AAPL** y **MU** se sitúan en un rango intermedio (≈

0.64 y ≈ 0.63), mientras que **MSFT** destaca por la dificultad de identificar caídas (recall DOWN bajo). En conjunto, aunque las métricas varían entre activos, la accuracy global entre **0.63 y 0.72** indica un rendimiento por encima del azar, suficiente para plantear su explotación práctica en escenarios de inversión real, aun reflejando la dificultad intrínseca de los mercados financieros.

Aunque los modelos **superan el azar** (accuracy > 0,5)**,** no obstante, **no alcanzan** muy buenos niveles como por ejemplo > 0,8 lo que confirma la dificultad de la predicción direccional en mercados financieros a medio y corto plazo.

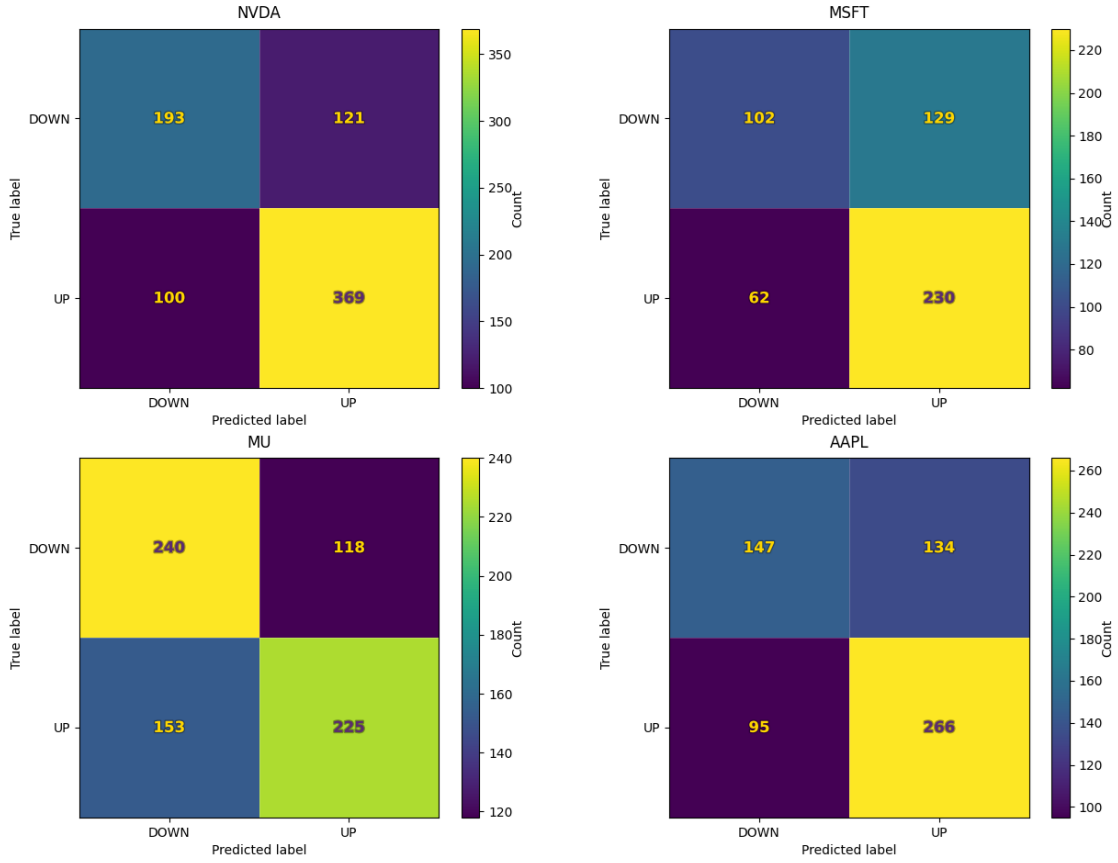


Figura 45. Matrices de confusión de los 4 activos analizados

Activo	Precision (DOWN)	Recall (DOWN)	F1 (DOWN)	Precision (UP)	Recall (UP)	F1 (UP)	Accuracy	Macro avg F1
NVDA	0.659	0.615	0.636	0.753	0.787	0.770	0.718	0.703
MSFT	0.622	0.442	0.516	0.641	0.788	0.707	0.635	0.612
MU	0.611	0.670	0.639	0.656	0.595	0.624	0.632	0.632
AAPL	0.607	0.523	0.562	0.665	0.737	0.699	0.643	0.631

Tabla 15. Informe de clasificación de 4 activos analizados

En conclusión, la estrategia propuesta logra niveles de exactitud que capturan patrones direccionales explotables en varios de los activos considerados. Las fórmulas y descripciones detalladas de las métricas de clasificación empleadas en este análisis se incluyen en el Anexo 12.

4.6.2 Simulación de compra venta de acciones usando las predicciones de modelos

La estrategia de simulación de compra y venta de acciones (backtesting) se aplica sobre nuestra cartera de 25 activos financieros, asignando a cada uno un capital inicial de 10.000 USD. Cada activo mantiene una curva de capital independiente, lo que permite analizar la evolución individual y la agregada de la cartera.

Dentro de este marco, se incorporan reglas explícitas de **stop-loss** (el límite de pérdida) y **take-profit** (beneficio objetivo) para la gestión de las operaciones. Su uso no responde a una decisión arbitraria, sino a la práctica consolidada en la literatura de trading algorítmico: Vezeris, Kyrgos y Schinas [27] muestran que diferentes configuraciones de TP/SL influyen de manera decisiva en la rentabilidad y en la estabilidad de la curva de capital, mientras que Kaminski y Lo [28] evidencian que, aunque en un *random walk* los stop-loss reducen el retorno esperado, en entornos con momentum o correlación serial (como las tendencias de corto plazo modeladas en este trabajo) estos mecanismos contribuyen a mejorar el perfil riesgo-retorno y a limitar pérdidas extremas.

Las señales predictivas se generan en el período fuera de muestra (2020–2024), a partir de modelos **XGBoost Regressor** entrenados individualmente para cada activo sobre el periodo 2010–2019 con características seleccionadas por el algoritmo genético y parámetros óptimos cargados desde ficheros *.csv* previamente calculados (véase 4.2.21).

Reglas operativas (por activo):

- **Estrategia solo-largo:** únicamente se permite abrir posiciones de compra; no se realizan ventas en corto.
- **Señal predictiva ($\hat{y}_t$):** valor continuo estimado por el modelo (ángulo de tendencia normalizado).
- **Umbral de entrada:** se fija en **0,51**, ligeramente superior al 0,5 y aunque este valor puede introducir ruido, dicho ruido queda controlado por las reglas de gestión de la operación (Stop Loss y Take Profit).
- **Tamaño de la posición:** en cada entrada se invierte la totalidad del capital disponible para ese activo:

$$n^{\circ}\ \text{títulos} = \frac{\text{Capital actual}}{\text{Precio de entrada}}$$

- Se mantiene **una única posición simultánea por activo**.

Condiciones de cierre

- **Take Profit (TP):** El objetivo de beneficio se fija en un +1,5 % respecto al precio de entrada:

$$TP = \text{Precio de entrada} \times (1 + 0.015)$$

  Si el máximo intradiario alcanza este valor en los 5 días posteriores a la apertura, la posición se cierra con ganancia.

- **Stop Loss (SL):** El límite de pérdida se fija en **–3 %** respecto al precio de entrada:

$$SL = \text{Precio de entrada} \times (1 - 0.03)$$

  Si el mínimo intradiario alcanza este valor, la posición se liquida inmediatamente con pérdida controlada.

- **Cierre por tiempo (T+5)**: Si tras 5 días no se ha alcanzado ni TP ni SL, la posición se liquida al precio de mercado del quinto día.

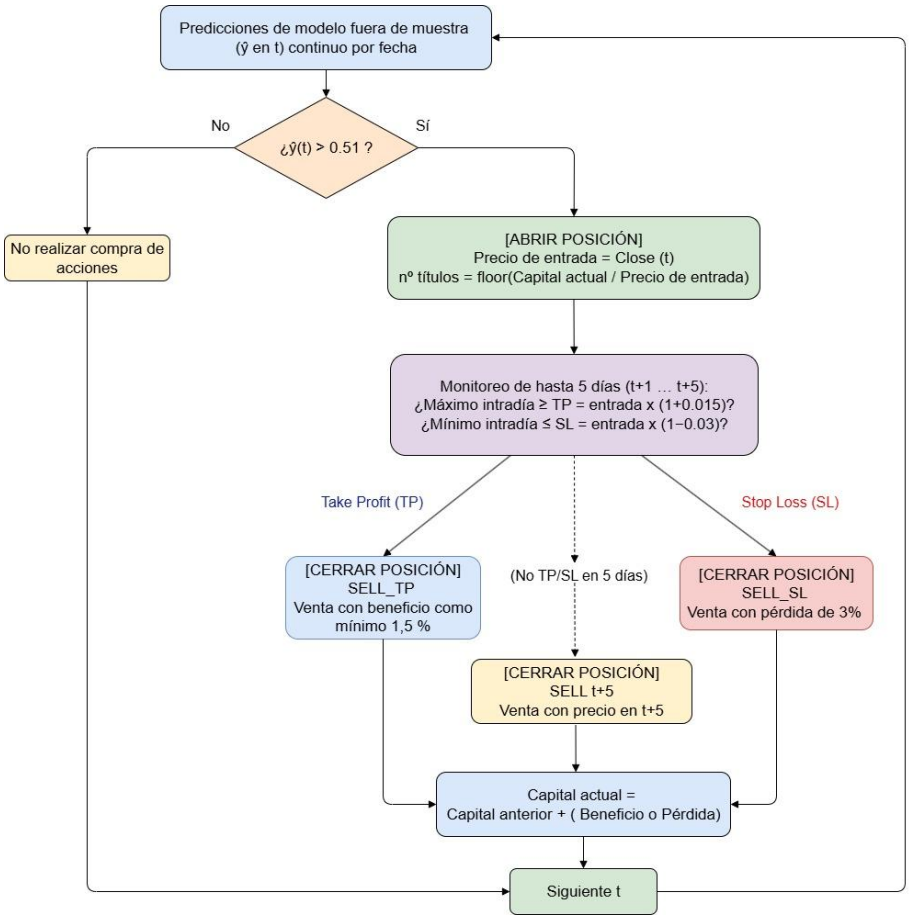


Figura 46. Flujo de simulación de tradinng

Como se muestra en Figura 46, cuando no hay posición, es decir no hay acciones compradas, se consulta la predicción de modelo fuera de muestra para el día actual y, si esta supera umbral 0,51 (señal de tendencia alcista), entonces se compra al precio **cierre** con el capital disponible de activo actual.

Luego se **gestionan 5 sesiones**: cada día se comprueba si se alcanza el **take profit** (+1,5%) o el **stop loss** (−3%); si no ocurre, las acciones se venden al precio cierre del quinto día. Finalmente se actualiza el capital acumulado (capital actual), se registra la operación y se vuelve a evaluar la predicción de modelo para el siguiente día.

En la Figura 47 se muestra la evolución del capital en el backtesting de NVDA. El modelo identifica con acierto tramos alcistas, como entre el 05-02 y el 25-02 del 2020, donde las compras permiten capturar la subida hasta 7,8 $.

En cambio, durante la tendencia bajista de mediados de marzo (10-03 a 20-03) evita en gran parte abrir posiciones, protegiéndose de caídas hacia 5,2 $. Aunque existen señales erróneas, la curva de capital evidencia que estas son **limitadas** y que, en términos generales, el modelo logra un rendimiento consistente y una capacidad predictiva sólida incluso en escenarios de cierta volatilidad.

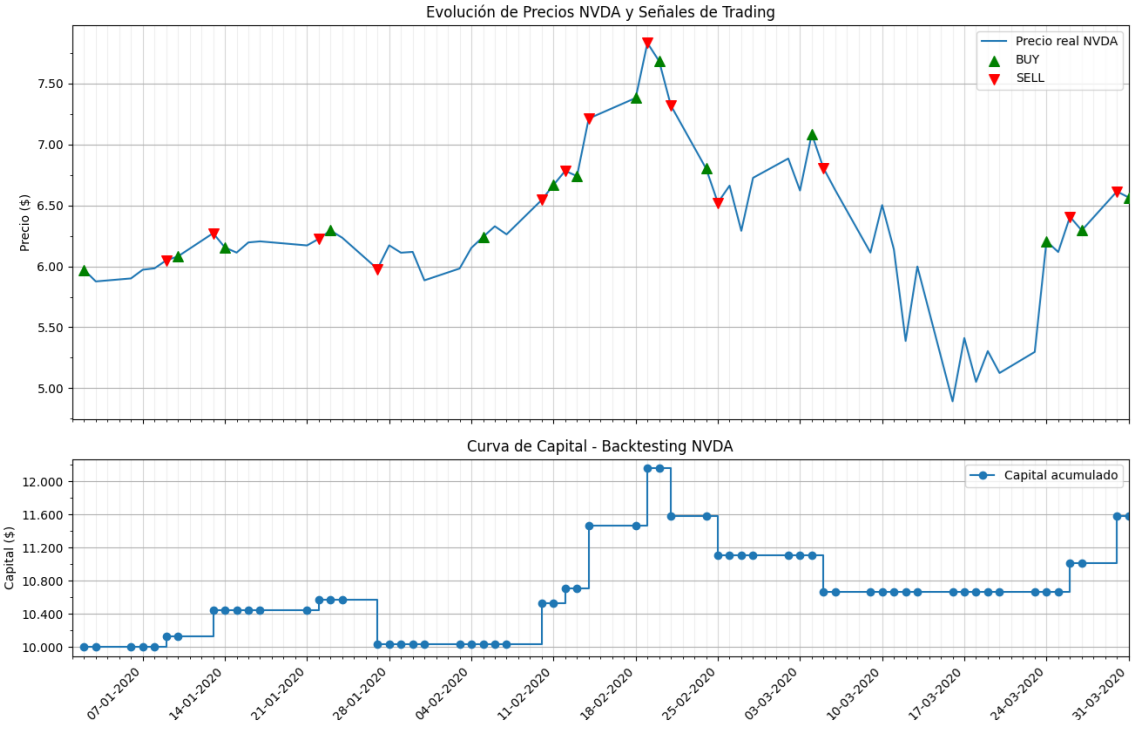


Figura 47. Backtesting de NVDA: Señales de compra-venta y crecimiento del capital.

En la Figura 48 se muestra la evolución mensual del capital durante el backtesting para los 25 activos de la cartera entre 2020 y 2024. La regresión lineal asociada a cada serie indica que la mayoría presentan una tendencia de crecimiento positivo y sostenido.

Hay muy pocos activos con caídas relevantes por debajo de **8.000 $** (−20 % del capital inicial), lo que refleja una adecuada preservación del capital. Incluso en aquellos con fases laterales o bajistas, el balance a 60 meses es mayoritariamente ascendente, lo que confirma la solidez del modelo y su capacidad para generar retornos consistentes en distintos activos.

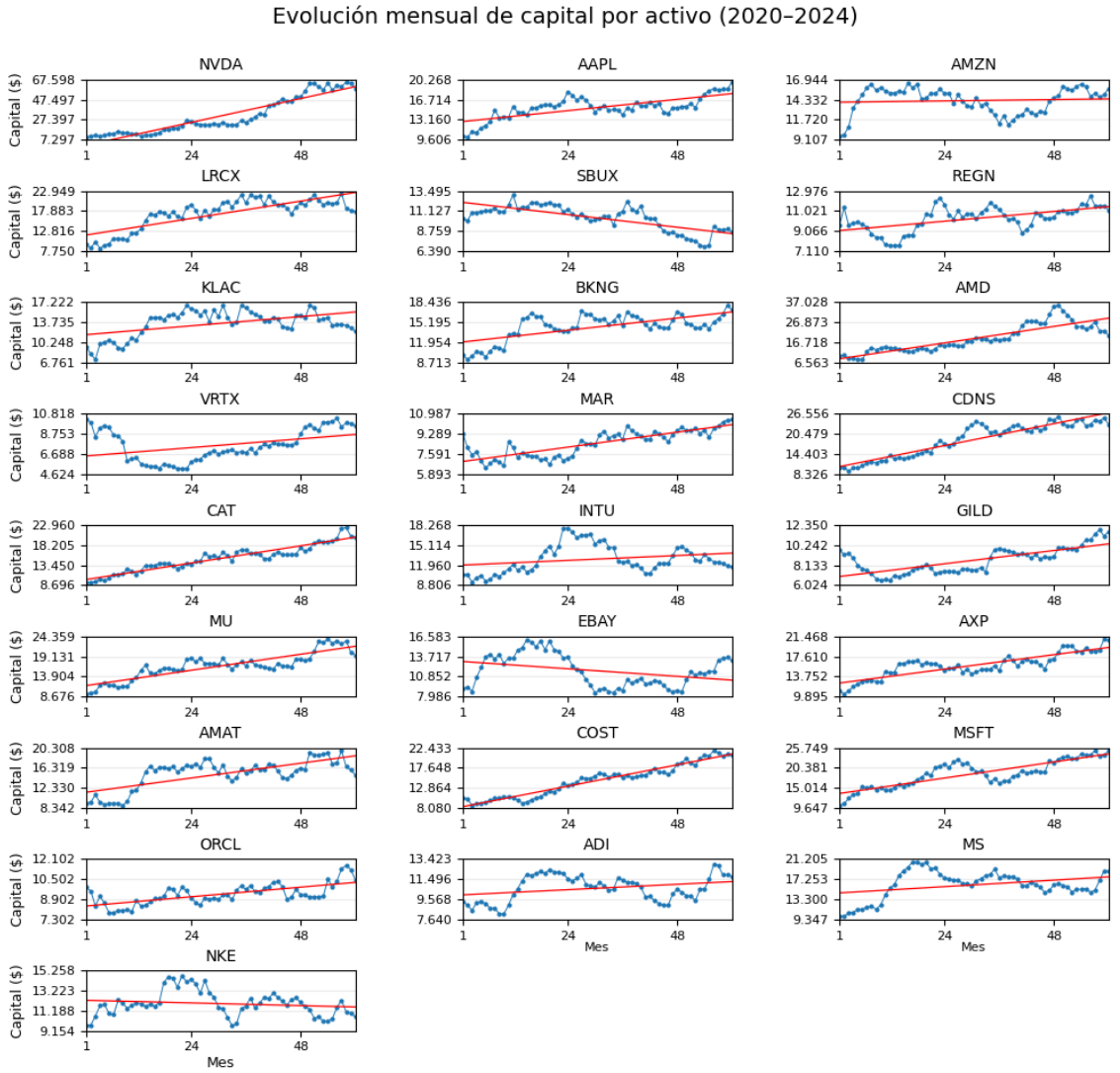


Figura 48. Backtesting: Evolución mensual de capital por activo en la cartera.

En la Figura 49 se muestra la evolución del capital agregado de la cartera compuesta por 25 activos en el periodo 2020–2024. El capital inicial de 250.000 $ creció hasta aproximarse a los 450.000 $, lo que supone una revalorización sustancial.

Si bien la tendencia global es claramente ascendente, resulta especialmente relevante analizar el comportamiento en 2022, un año marcado por inflación en máximos de cuatro décadas, subidas agresivas de tipos de interés por parte de la Reserva Federal y la incertidumbre derivada

de la guerra en Ucrania. Estos factores provocaron retrocesos notables en los principales índices bursátiles, con caídas cercanas al −19 % en el S&P 500 y superiores al −30 % en el Nasdaq.

En contraste, nuestra cartera experimentó únicamente un retroceso del **−5 %** a lo largo del año, lo que demuestra una notable capacidad de resiliencia y confirma la robustez del modelo aplicado incluso en escenarios de mercado altamente adversos.

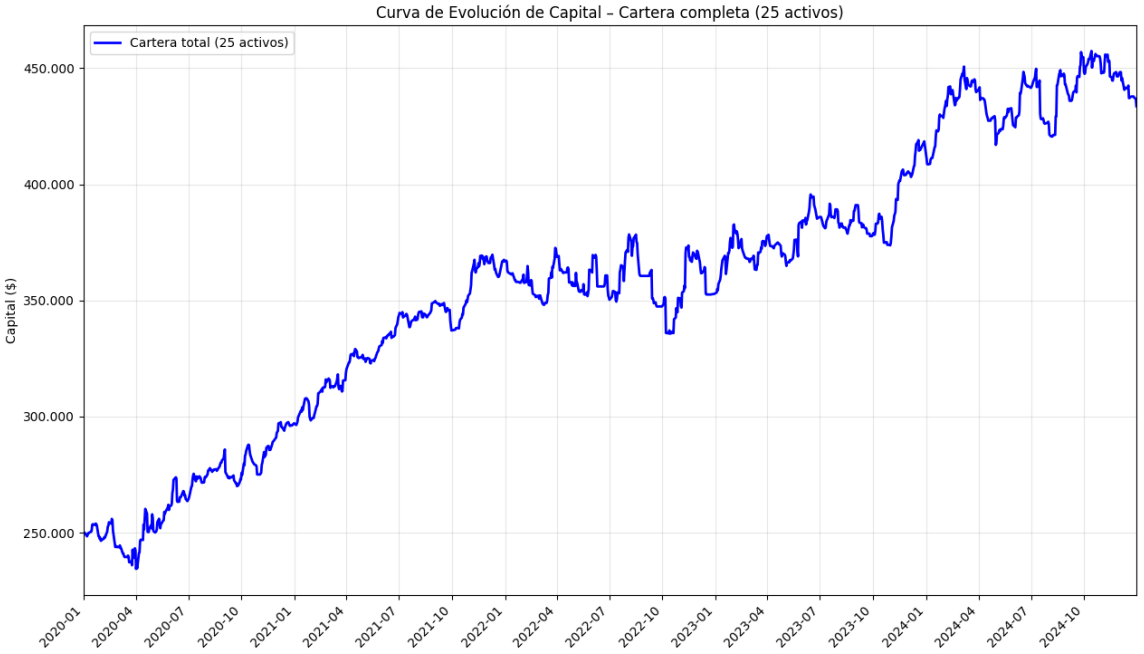


Figura 49. Backtesting: Evolución de capital global de cartera (25 activos).

Año	Capital inicial $	Capital final $	Ganancia $	Trades	Ganadores (%)	Payoff Ratio	Expectancy ($)	Drawdown máx ($)	ROI anual
2020	250.000	294.313,13	44.313,13	1.187	62,30%	0,74	38,39	20.855,24	17,73%
2021	294.313,13	361.927,72	67.614,58	933	64,60%	0,8	73,44	11.593,74	22,97%
2022	361.927,72	358.027,28	-3.900,44	903	57,70%	0,74	-8,58	41.847,26	-1,08%
2023	358.027,28	411.638,13	53.610,85	953	60,70%	0,8	65,71	20.975,02	14,97%
2024	411.638,13	434.102,06	22.463,92	928	59,50%	0,81	16,02	32.383,31	5,46%

Tabla 16. Métricas anuales de desempeño de la cartera en backtesting.

La Tabla 16 resume el desempeño de la cartera en el periodo 2020–2024. El capital inicial de 250.000 $ alcanzó 434.102 $, con una tasa de crecimiento anual compuesta (CAGR) del **11,68 %**. El número de operaciones se mantuvo cercano a 1.000 por año, con un porcentaje de acierto medio del **61 %**.

El ratio beneficio/pérdida (payoff) osciló entre 0,74 y 0,81, reflejando una relación estable entre ganancias y pérdidas. La rentabilidad esperada por operación (expectancy) fue positiva en todos los ejercicios salvo en 2022 (−8,58 $), coincidiendo con la caída de los índices bursátiles.

El máximo retroceso de capital (drawdown) anual osciló entre 11.000 $ y 42.000 $, según la volatilidad de cada ejercicio. Incluso en el peor caso (2022), la reducción fue limitada a un 8,58 % del capital, sin comprometer la trayectoria ascendente de la cartera ni su capacidad de recuperación.

El **ROI anual** se movió entre −1,08 % y +22,97 %, lo que evidencia un rendimiento acumulado consistente pese a un único año negativo. Las fórmulas utilizadas para el cálculo de estas métricas se encuentran recogidas en el Anexo 13

4.6.3 Impacto de variable objetivo sobre las métricas de simulación de trading

Hasta este punto, todas las pruebas anteriores se han realizado empleando una única definición de variable objetivo: TARGET_TREND_ANG_15_5, donde la pendiente de tendencia futura de precios se calcula con una ventana de 15 días (W) y un horizonte de predicción de 5 días (H).

Para profundizar en el efecto que ejerce esta definición sobre los resultados, se amplió el estudio repitiendo el proceso completo de selección de variables mediante algoritmo genético, optimización de hiperparámetros y backtesting de trading, pero seleccionando distintas configuraciones de la variable objetivo.

Se ha empleado la nomenclatura **W{ventana}H{horizonte}** donde:

- **W**: número de observaciones incluidas en la ventana de regresión lineal para estimar el ángulo de tendencia.
- **H**: horizonte en días que determina la amplitud futura considerada en la definición del target.

De esta manera, se evaluaron distintas combinaciones de (W, H), manteniendo constantes todos los demás parámetros de trading, configuración de algoritmo genético y ajuste de hiperparámetros así mismo como el periodo de test fuera de muestra (años 2020 -2024).

Como se muestra en Figura 50, Figura 51 y Tabla 17, los resultados evidencian que la selección de la variable objetivo tiene un impacto decisivo en la rentabilidad y estabilidad de la estrategia. En particular, se observa que al introducir horizontes más largos (H ≥ 10), los modelos generan señales más consistentes y el capital acumulado aumenta de manera significativa.

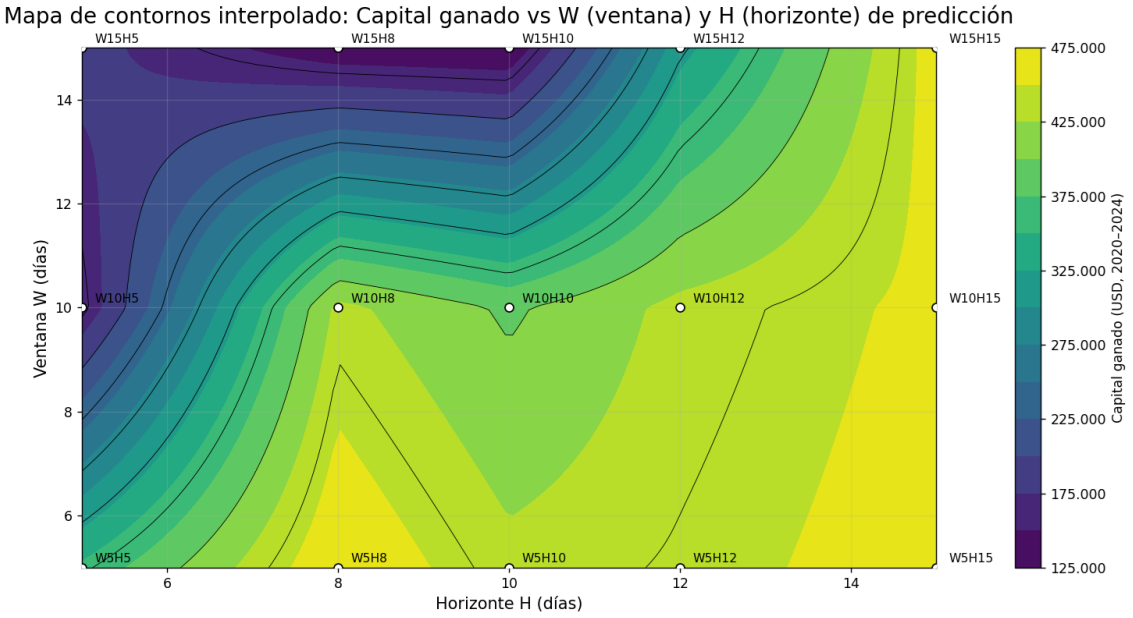


Figura 50. Capital ganado en función de configuración de variable objetivo (2020-2024).

Los resultados muestran un patrón claro: a medida que el horizonte H aumenta, el capital final y la rentabilidad acumulada crecen de forma significativa (véase Figura 50).

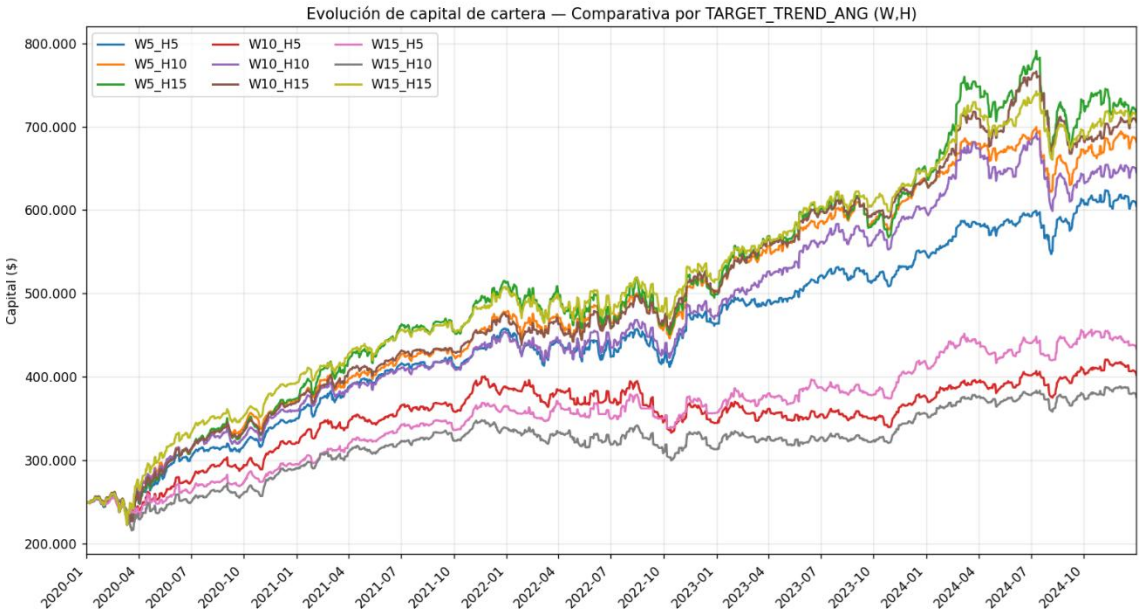


Figura 51. Evolución de capital acumulado de cartera en función de variable objetivo.

Variable Objetivo	Capital inicial $	Capital Final $	Capital ganado $	ROI %	Max DD $	Max DD %	% Gana - dores	Expec- tancy $	Payoff ratio	CAGR %
W5H8	250.000	721.365	471.365	188,55%	57.751	8,70%	62,90%	79,91	0,77	21,97%
W5H15	250.000	711.026	461.026	184,41%	117.283	14,70%	62,30%	70,37	0,74	23,65%
W15H15	250.000	711.082	461.082	184,43%	82.923	11,30%	63,10%	92,55	0,76	23,00%
W10H15	250.000	705.554	455.554	182,22%	99.436	13,00%	62,90%	96,66	0,77	23,08%
W5H12	250.000	692.013	442.013	176,81%	110.259	14,70%	62,80%	81,77	0,74	21,85%
W10H12	250.000	682.147	432.147	172,86%	109.246	14,40%	62,80%	82,93	0,75	22,25%
W10H8	250.000	681.769	431.769	172,71%	66.536	9,80%	62,40%	77,56	0,78	22,24%
W5H10	250.000	681.021	431.021	172,41%	58.236	8,50%	63,40%	81,96	0,75	22,34%
W10H10	250.000	645.429	395.429	158,17%	93.170	13,50%	62,60%	79,13	0,76	20,90%
W5H5	250.000	604.998	354.998	141,99%	63.845	16,20%	59,90%	26,59	0,76	19,69%
W15H12	250.000	564.851	314.851	125,94%	67.546	12,00%	61,70%	60,25	0,75	18,36%
W15H5	250.000	434.102	184.102	73,64%	42.729	11,30%	60,80%	37,34	0,76	11,65%
W15H8	250.000	382.100	132.100	52,84%	45.335	12,70%	60,30%	29,45	0,76	9,73%
W10H5	250.000	402.230	152.230	60,89%	67.754	16,90%	60,00%	27,6	0,76	9,99%
W15H10	250.000	375.923	125.923	50,37%	61.145	14,40%	60,40%	28,41	0,75	9,42%

Tabla 17 Métricas de rendimiento de XGBoost Regressor en función de variable objetivo

Los mejores desempeños se observan en **W5H15** (ROI 184,41 %, CAGR 23,65 %), **W15H15** (ROI 184,43 %, CAGR 23,00 %) y **W10H15** (ROI 182,22 %, CAGR 23,08 %), todos ellos con horizontes largos de 15 días. Estas configuraciones alcanzan capitales finales en el rango de 690.000–722.000 $, casi triplicando el capital inicial en 5 años.

En contraste, los horizontes más cortos (H=5 o H=10) muestran ganancias notablemente menores. Por ejemplo, W15H5 (ROI 73,64 %) o W15H10 (ROI 50,37 %) apenas duplican el capital, y presentan además CAGR en torno al 10–12%, muy por debajo de las configuraciones con horizontes amplios.

El porcentaje de operaciones ganadoras se mantiene bastante estable en el rango de 60–63% independientemente de la configuración, lo que indica que el mayor ROI no proviene de un aumento en la proporción de aciertos, sino de que las operaciones ganadoras son más amplias y consistentes cuando la variable objetivo incorpora mayor información futura.

Como se muestra en Figura 52 a diferencia de la rentabilidad, el comportamiento del riesgo (medido como Max Drawdown %) no sigue un patrón monotónico en función de tamaño de horizonte o ventana. Su promedio se mantiene entre el 11% y el 14%, sin evidenciar una escalada proporcional al horizonte. Esto sugiere que aumentar horizonte de predicción amplifica la

rentabilidad, pero no necesariamente incrementa el riesgo de forma directa, lo que refuerza la robustez del modelo.

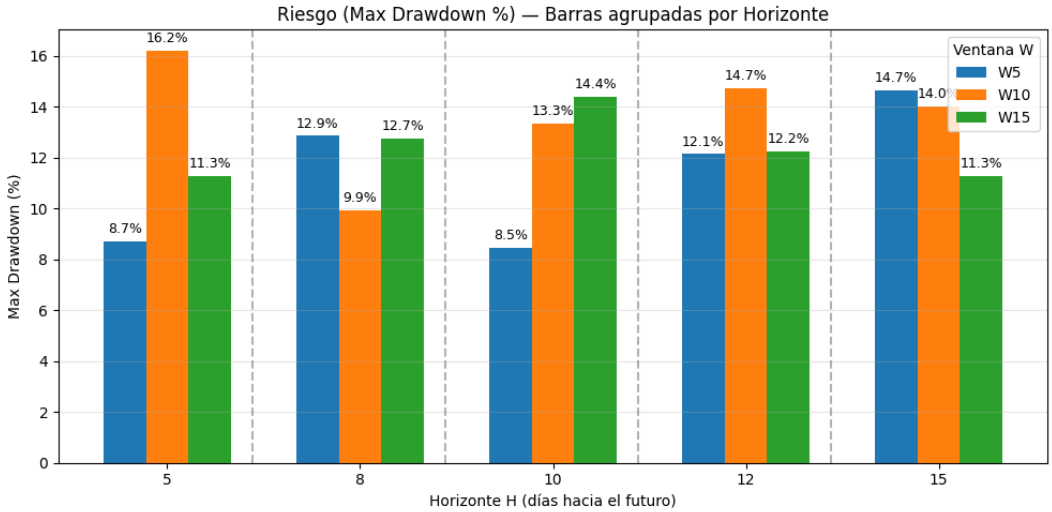


Figura 52. Riesgo en función de variable objetivo.

4.6.4 Evaluación comparativa entre el sistema propuesto y modelos ARIMA clásicos

Con el objetivo de establecer un punto de comparación sólido frente a los modelos ML desarrollados en este trabajo, se implementó un pipeline de predicción basado en el modelo **ARIMA** (*Autoregressive Integrated Moving Average*). Este enfoque, formalizado originalmente por Box y Jenkins [29], constituye uno de los pilares de la modelización de series temporales y ha sido aplicado extensamente en economía y finanzas durante más de cinco décadas. Al día de hoy continúa siendo utilizado en la predicción de series temporales bursátiles [30].

El modelo ARIMA se denota como ARIMA (p, d, q), donde:

- p indica el número de términos autorregresivos (AR),
- d corresponde al grado de diferenciación aplicado para garantizar estacionariedad,
- q representa el número de rezagos en la media móvil (MA).

Su formulación general puede expresarse como:

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t$$

Donde: y_t es el valor de la serie en el instante t, ϕ_i y θ_j son parámetros estimados, y ε_t corresponde al término de error (ruido blanco). Cuando se aplica diferenciación de orden d, la serie se transforma como:

$$yt' = (1 - B)^d y_t$$

Donde: B es el operador de rezago, de forma que $By_t = y_{t-1}$

En nuestro caso, se entrenó un modelo **ARIMA (1,1,0)** por cada uno de los 25 activos de la cartera, utilizando exclusivamente los **precios de cierre** como variable de entrada. De este modo, a diferencia de los modelos de nuestro sistema propuesto que incorporan cientos de indicadores técnicos, ARIMA se centra únicamente en la dinámica interna de la serie.

El entrenamiento se llevó a cabo en el periodo **2010–2019**, mientras que las predicciones se extendieron de forma continua para el intervalo **2020–2024**, sin recalibraciones intermedias, asegurando así una comparación justa con el criterio de rango de entrenamiento y test.

Las señales de trading se derivaron directamente del signo de las predicciones: si la estimación a cinco días vista superaba el precio real, se generaba señal de **compra**. Para garantizar igualdad de condiciones, el backtesting se realizó bajo las mismas condiciones que en nuestros experimentos anteriores (véase 4.6.2), incluyendo la variable objetivo con horizonte de 5 días.

Se aplicaron idénticas reglas de control del riesgo (*take profit* 1,5 %, *stop loss* 3 % y paciencia máxima de 5 días), asignando 10.000 USD por activo y consolidando todas las operaciones de 2020–2024 en un log maestro.

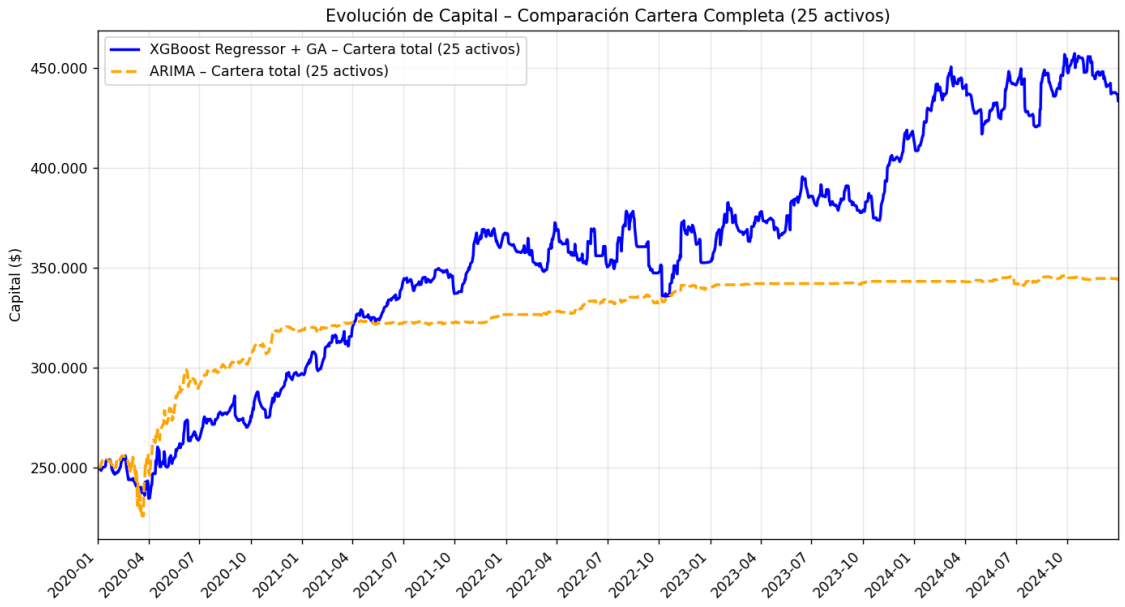


Figura 53. Evolución de capital de cartera, ARIMA vs XGBoost Regressor + GA.

La Figura 53 muestra la evolución de capital de la cartera completa utilizando el modelo ARIMA frente a nuestro sistema basado en XGBoost Regressor + GA con la variable objetivo configurada como ventana 15 y horizonte 5.

Se observa que, en los primeros meses del periodo de prueba, ARIMA llega a superar al modelo propuesto, reflejando su capacidad de capturar patrones lineales de corto plazo. Sin embargo, su desempeño se degrada progresivamente, hasta quedar rezagado frente a nuestro enfoque.

La causa principal de esta degradación radica en la naturaleza del propio ARIMA. Al estar construido sobre componentes autorregresivos y de medias móviles, depende de los valores **absolutos** de la serie y de su **estacionariedad**. Esto provoca un drift progresivo en las predicciones, que a su vez desemboca en una **falta de señales** de entrada y una notable disminución en el número de trades a lo largo del tiempo, como se aprecia en la Tabla 18.

Se observa cómo, tras un 2020 con 801 operaciones y un ROI anual del 27,6 %, la actividad operativa cae de forma abrupta: en 2021 apenas se registran 109 trades y en 2023 solo 14, con un ROI residual del 0,85 %. Aunque en 2024 se recupera levemente el número de operaciones (57), el rendimiento se mantiene muy bajo (0,29 %). La tasa de crecimiento anual compuesta (CAGR) alcanzada por ARIMA fue del **6,61 %**.

En consecuencia, la rentabilidad acumulada de ARIMA se estabiliza y deja de crecer, lo que explica la trayectoria plana de la curva de capital en los últimos años del periodo.

Año	Capital inicial $	Capital final $	Ganancia $	Drawdown máx ($)	Trades	Ganado-res (%)	Payoff Ratio	Expec-tancy ($)	ROI %
2020	250.000,00	318.981,87	68.981,87	28.491,44	801	64,20 %	0,8	86,12	27,59
2021	318.981,87	326.606,60	7.624,73	2.330,99	109	61,50 %	1,01	67,15	2,39
2022	326.606,60	340.323,01	13.716,41	4.573,35	119	67,20 %	0,87	109,21	4,20
2023	340.323,01	343.207,00	2.884,00	686,48	14	85,70 %	1,3	257,48	0,85
2024	343.207,00	344.200,79	993,78	5.168,27	57	63,20 %	0,61	6,41	0,29

Tabla 18. Métricas de rendimiento de cartera (ARIMA)

En este experimento, se decidió no reentrenar ARIMA para asegurar condiciones comparables y consistentes a lo largo de 2020–2024.

En contraste, nuestro sistema basado en predicciones de modelos **XGBoost Regressor** entrenado con señales de indicadores no requiere reentrenamientos continuos, ya que se entrena sobre **indicadores técnicos derivados de las tendencias de precios** en lugar de valores absolutos. Este planteamiento lo hace más robusto frente a cambios de escala en los precios y evita el deterioro observado en ARIMA. Desde esta perspectiva, el modelo propuesto se presenta como una alternativa superior, al ofrecer estabilidad predictiva sin necesidad de recalibración frecuente.

4.6.5 Desempeño del sistema propuesto frente a diseños clásicos basados en MLP

El perceptrón multicapa (MLP), (véase 4.2.16), se incluye como uno de los modelos de referencia más utilizados en experimentos de predicción bursátil. Para su evaluación se implementó un test especifico a partir de series OHLCV (precios diarios de: apertura, máximos, cierre, mínimos y volumen correspondiente), en el que se construyeron muestras mediante una ventana deslizante L de **30 días** y un horizonte H de **5 días**.

Así, para cada instante t, el modelo recibe como entrada los precios y volúmenes de los últimos treinta días y estima el valor de cierre a cinco días vista.

$$X_t = [OHLCV_{t-L}, OHLCV_{t-L+1}, \ldots, OHLCV_{t-1}], \qquad y_t = Close_{t+H}$$

Los datos fueron normalizados mediante StandardScaler, ajustado únicamente con el conjunto de entrenamiento para evitar fuga de información. El entrenamiento se realizó por activo, empleando el periodo 2010–2019 como conjunto de aprendizaje y 2020–2024 como conjunto de prueba fuera de muestra. Las predicciones $\hat{y}_t$ se transformaron en señales de trading comparándolas con el precio actual $Close_t$

$$signal\ t\ = \begin{cases} 0{,}6, si\ \ \hat{y}_t > Close_t \\ 0{,}4\ \ en\ caso\ contrario \end{cases}$$

De esta forma, el MLP genera una instrucción binaria (comprar o no comprar), que se evalúa en el backtester bajo las **mismas condiciones de prueba** que el resto de modelos: mismo horizonte, mismo periodo out-of-sample y mismos parámetros de trading (stop-loss, take-profit y capital inicial).

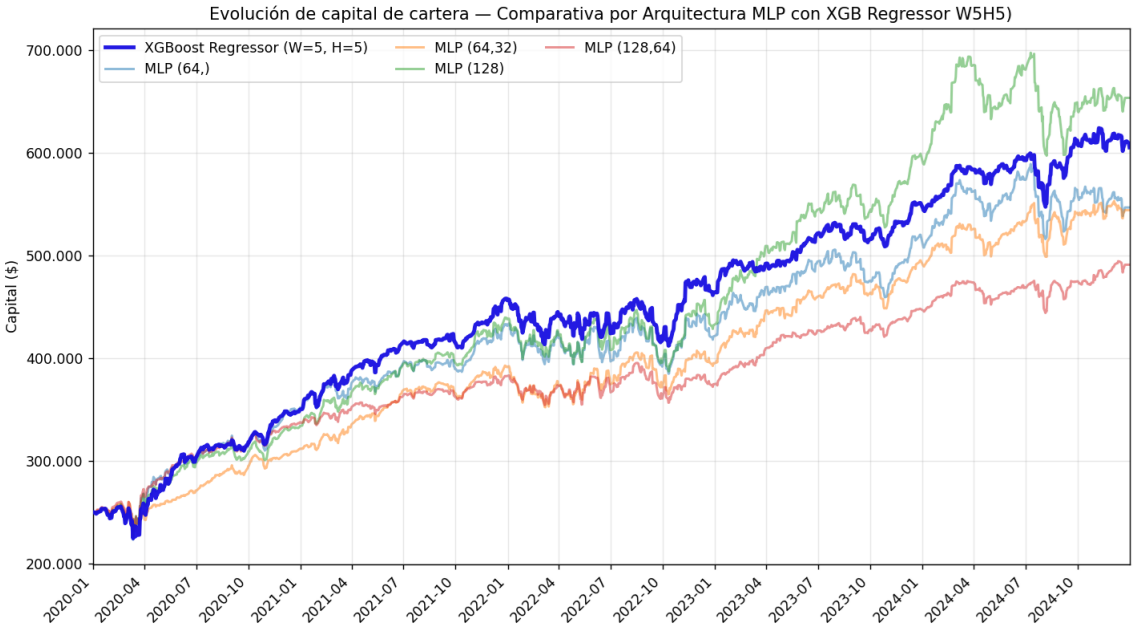


Figura 54. Evolución del capital de la cartera por arquitectura MLP y modelo XGB (W5H5)

La Figura 54 muestra la evolución del capital de la cartera comparando distintas arquitecturas MLP frente al sistema propuesto, basado en modelos XGBoost Regressor entrenados individualmente por acción con horizonte W=5, H=5. Todas las curvas siguen un patrón similar de crecimiento y retrocesos en línea con los ciclos del mercado entre 2020 y 2024, aunque con diferencias claras en el nivel final alcanzado.

Destaca que el MLP con **una sola capa oculta compuesta por 128 unidades (neuronas)** logra superar ligeramente la rentabilidad del sistema propuesto, mientras que otros diseños más complejos, como el de dos capas ocultas con 128 y 64 unidades cada una, quedan por detrás, lo que evidencia la fuerte sensibilidad de este tipo de redes neuronales artificiales a su configuración interna.

Arq. MLP	Capital inicial $	Capital Final $	Capital ganado $	ROI %	Max DD $	Max DD %	Gana-dores %	Expec-tancy $	Payoff ratio	CAGR %
128	250.000	653.224	403.224	161,3	100.212	14,4	61,4	75,44	0,78	21,3
64	250.000	546.547	296.547	118,6	73.130	12,4	60,8	51,21	0,76	17
64,32	250.000	543.946	293.946	117,6	52.355	9,5	61,7	60,81	0,77	16,9
128,64	250.000	490.845	240.845	96,3	38.651	9,8	62,1	56,94	0,76	14,5
32	250.000	381.923	131.923	52,8	27.684	8,4	61,4	42,99	0,75	8,9

Tabla 19. Métricas de rendimiento de cartera en función de arquitectura de MLP

La Tabla 19 detalla las métricas asociadas a cada arquitectura. El ROI oscila entre un **52,8 %** y un **161,3 %**, y el CAGR entre un **8,9 %** y un **21,3 %**, reflejando la dispersión de resultados entre configuraciones. El drawdown máximo varía entre **8,4 %** y **14,4 %**, mientras que el porcentaje de operaciones ganadoras se mantiene estable en torno al **61 %** en todos los casos. La expectativa y el payoff ratio confirman que, aunque todas las arquitecturas operan con un comportamiento general similar, la rentabilidad depende de manera crítica de cómo se ajuste la red.

En contraste, el sistema propuesto, basado en un ensemble de árboles de decisión optimizados mediante Extreme Gradient Boosting (XGBoost) con seleccón automatica de caracteríasticas mediante algorinmo genético, ofrece resultados competitivos sin necesidad de definir número de capas, neuronas, funciones de activación, dropout, etc.

Esto lo convierte en una opción más manejable y robusta, donde la mejora del rendimiento proviene de la agregación iterativa y ponderada de múltiples árboles en un marco ensemble, y no de una arquitectura neuronal difícil de calibrar.

4.6.6 Comparativas y métricas finales

En la Figura 55 se compara la evolución de capital durante backtesting según distintas configuraciones. En MLP, la variación proviene de la arquitectura de la red y sin embargo en XGBoost Regressor, de la definición de la variable objetivo, donde horizontes largos (H=15) generan trayectorias distintas a horizontes cortos (H=5).

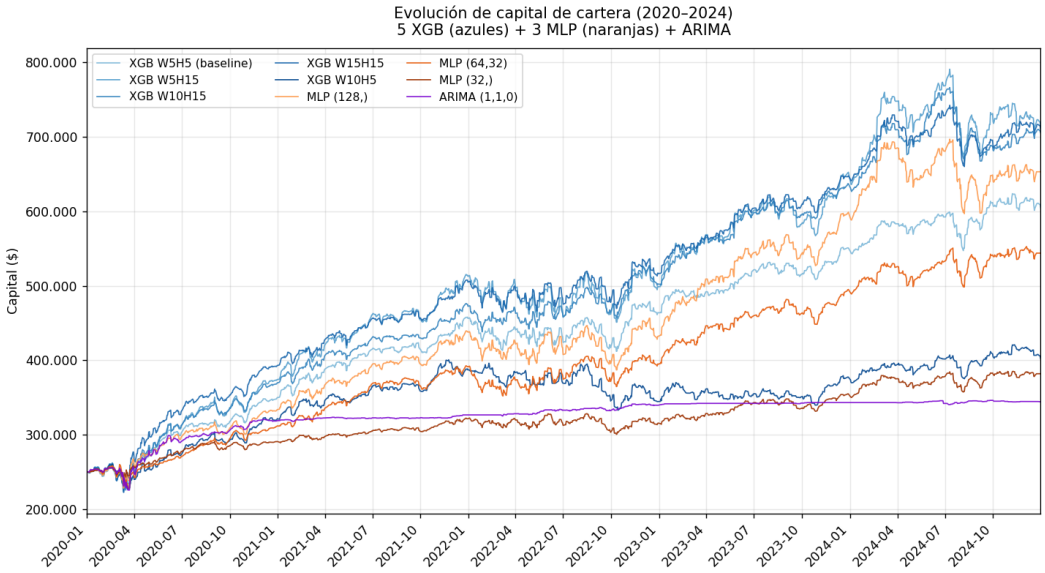


Figura 55. Evolución de capital acumulado en función de modelo su configuración.

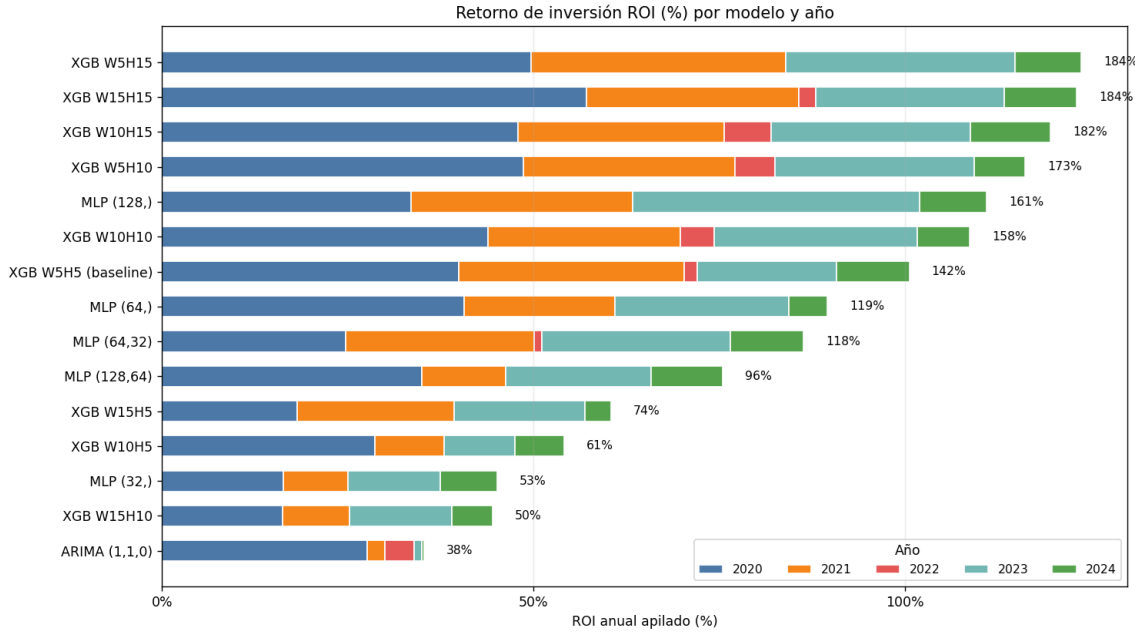


Figura 56. ROI % Anual en función de modelo y configuración compuesto por años.

La Figura 56 muestra el ROI anual apilado por modelo y año (2020–2024). Se aprecia un patrón ligado al comportamiento del mercado: en años alcistas los modelos alcanzan rentabilidades elevadas, mientras que, en 2022, cuando los mercados estuvieron a la baja, ninguno consiguió predicciones que llevaran a buenos resultados. En XGBoost destacan los horizontes largos (H=15) con los mejores rendimientos acumulados, y en MLP se evidencia la influencia de la arquitectura.

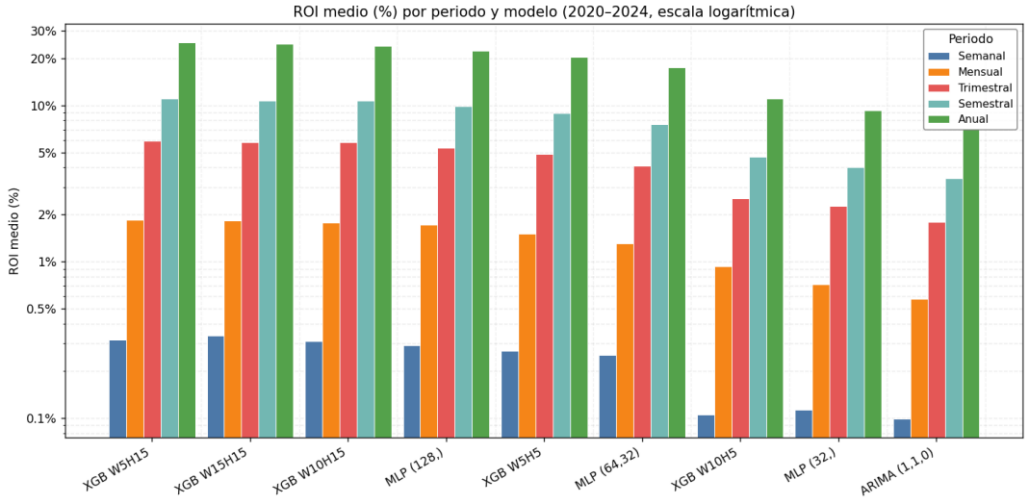


Figura 57. Retorno de inversión % medio en función de modelo y periodo.

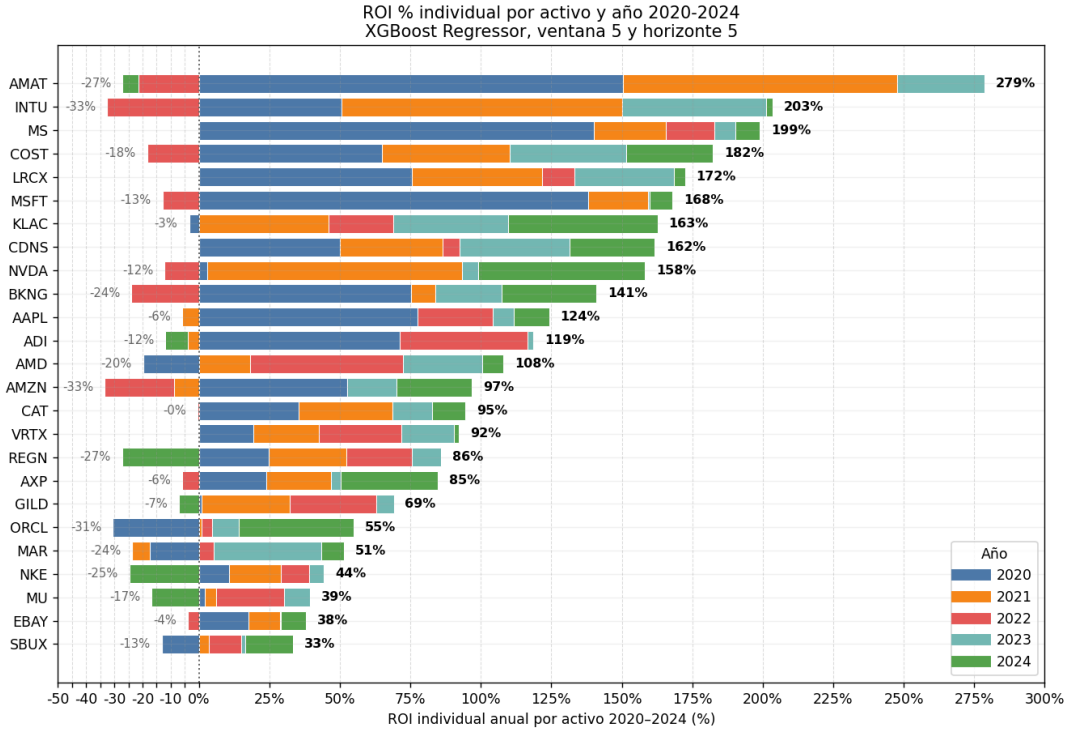


Figura 58. Retorno de inversión % para cada activo financiero apilado por años.

El gráfico en la Figura 58 refleja el ROI anual por activo en el periodo 2020–2024. Aunque se observan pérdidas puntuales en ciertos ejercicios como **AMAT en 2022 (−27 %)** o **INTU en 2021 (−33 %)**, estas caídas quedan ampliamente compensadas por los rendimientos en años posteriores, alcanzando acumulados superiores al 200 % en ambos casos. En contraste, valores como SBUX o MU muestran aportaciones finales más modestas, cercanas al 30–40 %.

No obstante, la **gran masa de resultados positivos relevantes** se concentra entre un 50 % y un 182 % de retorno acumulado, donde se agrupan numerosos activos (COST, MS, NVDA, KLAC, CDNS, entre otros), aportando de forma consistente al crecimiento de la cartera. Este patrón indica que las ganancias no son fruto del azar ni de episodios aislados, sino que responden a una contribución recurrente y diversificada de la mayoría de los valores.

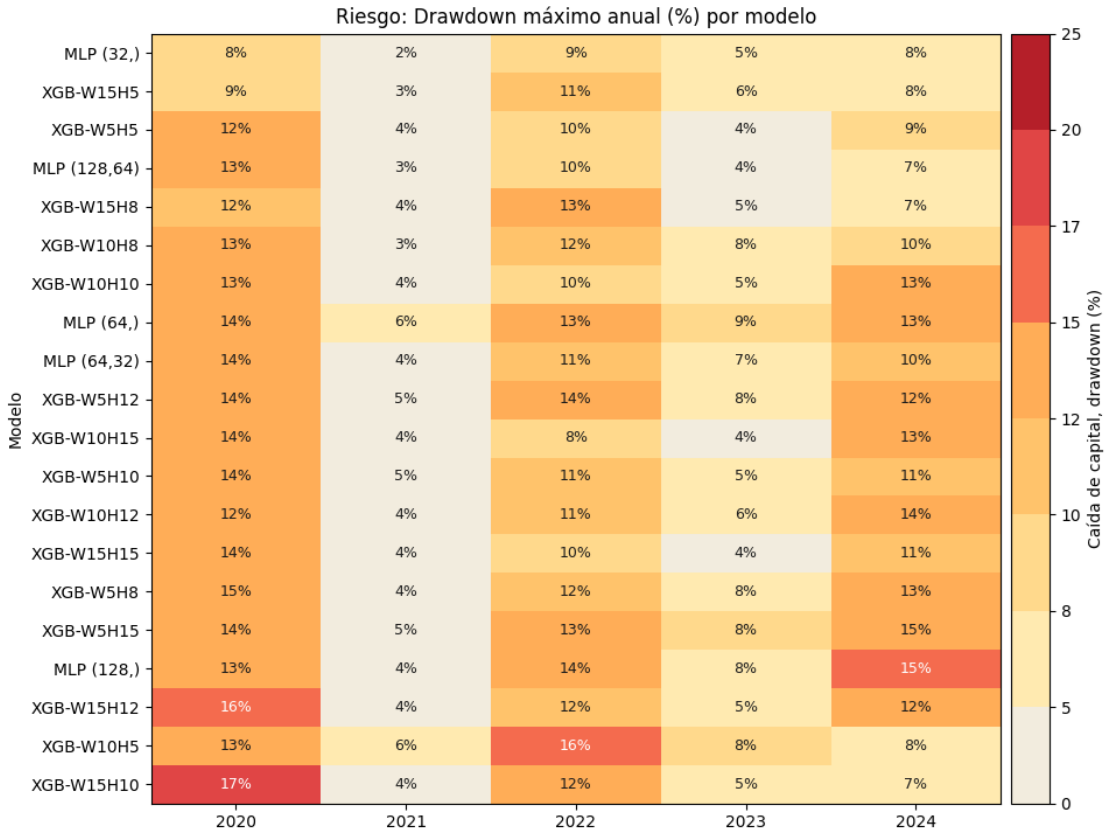


Figura 59. Riesgos: caída de capital de cartera anual en función de modelo

El mapa de calor en Figura 59 refleja el **riesgo asumido por cada modelo y configuración** en el periodo 2020–2024, medido a través del drawdown máximo anual (% caída de capital anual). Los tonos claros corresponden a niveles de caída de capital reducidos (<10 %), mientras que los tonos naranjas y rojos destacan los episodios de mayor estrés (>15 %).

Se aprecia que, en general, los modelos mantienen niveles de riesgo contenidos, aunque ciertas configuraciones (p. ej., XGB-W15H10 en 2020) alcanzan caídas cercanas al 17 %.

Modelo	Capital inicial $	Capital final $	Capital ganado $	ROI %	Max DD $	Max DD %	Trades	% Gana-dores	Expec-tancy $	Payoff ratio	CAGR %
XGB-W5H8	250K	721.365	471.365	188,55%	96.377	12,85%	5.681	62,65%	83	0,76	23,62%
XGB-W5H15	250K	711.026	461.026	184,41%	116.914	14,78%	6.724	61,97%	69	0,74	23,27%
XGB-W15H15	250K	710.819	460.819	184,33%	82.401	11,10%	4.855	63,32%	95	0,75	23,26%
XGB-W10H15	250K	705.554	455.554	182,22%	99.436	12,98%	4.713	62,89%	97	0,77	23,08%
XGB-W5H12	250K	692.013	442.013	176,81%	91.721	12,24%	6.248	62,29%	71	0,75	22,60%
XGB-W10H12	250K	682.147	432.147	172,86%	109.246	14,42%	5.211	62,79%	83	0,75	22,25%
XGB-W5H10	250K	681.909	431.909	172,76%	77.940	11,14%	5.240	63,40%	82	0,74	22,24%
XGB-W10H8	250K	681.769	431.769	172,71%	66.536	9,77%	5.567	62,39%	78	0,78	22,24%
MLP (128,)	250K	653.224	403.224	161,29%	100.212	14,38%	5.345	61,40%	75	0,78	21,30%
XGB-W10H10	250K	645.429	395.429	158,17%	93.170	13,46%	4.997	62,56%	79	0,76	20,90%
XGB-W5H5	250K	604.998	354.998	142,00%	52.070	8,69%	5.285	62,65%	67	0,76	19,35%
XGB-W15H12	250K	564.851	314.851	125,94%	70.662	12,08%	5.505	62,03%	57	0,75	17,72%
MLP (64,)	250K	546.547	296.547	118,62%	73.130	12,42%	5.791	60,84%	51	0,76	17,03%
MLP (64,32)	250K	543.946	293.946	117,58%	52.355	9,50%	4.834	61,73%	61	0,77	16,91%
MLP (128,64)	250K	490.846	240.846	96,34%	38.651	9,78%	4.230	62,06%	57	0,76	14,52%
XGB-W15H5	250K	434.102	184.102	73,64%	40.728	10,74%	4.904	61,05%	38	0,76	11,68%
XGB-W10H5	250K	402.230	152.230	60,89%	67.754	16,91%	5.516	60,01%	28	0,76	9,99%
MLP (32,)	250K	381.924	131.924	52,77%	27.684	8,44%	3.069	61,36%	43	0,75	8,89%
XGB-W15H8	250K	380.300	130.300	52,12%	47.427	13,51%	5.014	59,97%	26	0,76	8,76%
XGB-W15H10	250K	375.924	125.924	50,37%	48.133	13,84%	4.995	60,10%	25	0,75	8,51%

Tabla 20. Métricas de rendimiento de cartera (2020-2024) en función de modelo.

La Tabla 20 resume el desempeño de los distintos modelos evaluados en el periodo 2020–2024, calculado sobre una cartera inicial de 250.000 $. Los resultados muestran una clara superioridad de los modelos basados en XGBoost Regressor con ventanas amplias (por ejemplo, **XGB-W5H8** con un ROI del 188,55 % y un CAGR del 23,62 %), que encabezan el ranking en términos de rentabilidad. En contraste, algunos modelos presentan un rendimiento más modesto, como **XGB-W15H10**, cuyo ROI apenas alcanza el 50,37 %.

En general, la mayoría de configuraciones mantienen un **drawdown máximo anual controlado entre el 9 % y el 14 %**, lo que evidencia una relación riesgo–beneficio equilibrada. La rentabilidad no se explica únicamente por azar, ya que el porcentaje de operaciones ganadoras se mantiene estable en torno al **60–63 %** para casi todos los casos.

Estos resultados refuerzan la robustez del enfoque propuesto, evidenciando que, a pesar de ligeras variaciones en payoff ratio o expectancy, la tendencia global es claramente positiva y consistente en los modelos mejor posicionados.

Capítulo 5. DISCUSIÓN

Los resultados obtenidos permiten valorar tanto las fortalezas como las limitaciones del sistema propuesto. En primer lugar, se ha evidenciado que la combinación de técnicas de feature engineering avanzado, selección de características mediante algoritmos genéticos y modelos de machine learning como XGBoost Regressor ofrece un rendimiento superior al de métodos tradicionales basados únicamente en series de precios (Figura 53, Tabla 18). Este hallazgo confirma lo señalado en la literatura reciente acerca de la importancia de integrar indicadores técnicos y procesos de selección automatizada.

No obstante, la mejora observada no es uniforme ni garantiza un éxito absoluto. En determinados periodos de alta volatilidad, como 2022, el rendimiento de los modelos se vio reducido (Tabla 16), lo que refleja la sensibilidad del enfoque ante cambios bruscos de régimen de mercado. Además, aunque los MLP fueron probados con distintas arquitecturas, su rendimiento resultó menos estable frente a XGBoost Regressor (Figura 54, Tabla 19).

Esto se explica en parte porque su configuración fue más simple —basada en precios OHLCV con ventana móvil fija dónde no se ha realizado ni selección evolutiva de características ni un ajuste exhaustivo de hiperparámetros. Como consecuencia, el modelo no logró explotar todo su potencial y mostró una mayor irregularidad frente al sistema propuesto con XGBoost y GA.

Desde un punto de vista práctico, el backtesting indica que las señales generadas podrían resultar útiles para estrategias de corto plazo (Figura 49). Sin embargo, estos resultados deben interpretarse con cautela: el sistema no incorpora costes de transacción, slippage ni limitaciones de liquidez, factores que en un entorno real reducirían la rentabilidad estimada. Igualmente, el coste computacional del algoritmo genético plantea dudas sobre su escalabilidad a universos de activos más amplios (Figura 38 y Figura 39).

En definitiva, el trabajo demuestra que es posible mejorar la capacidad predictiva respecto a enfoques clásicos, pero también revela que la aplicabilidad práctica depende de ajustar el sistema a las condiciones reales del mercado y de complementar el modelo con mecanismos de gestión de riesgo. El sistema aporta un marco metodológico sólido (Figura 7), aunque no debe interpretarse como una solución definitiva sino como un paso intermedio hacia modelos más integrales y adaptativos.

Por último, la comparación entre configuraciones heterogéneas como EDA + XGBoost + GA frente a MLP con OHLCV o ARIMA sin reajuste periódico, debe tomarse con cautela. El rendimiento no depende solo del algoritmo, sino también del preprocesado, la definición de variable objetivo, la selección de características, el ajuste de hiperparámetros o la arquitectura de modelo. En este trabajo se comprobó que aspectos como el tamaño del dataset, el número de variables seleccionadas por GA, la formulación del ángulo de tendencia y la optimización de parámetros (o el reajuste de ARIMA cada cierto tiempo) afectan de forma decisiva a los resultados. Esto confirma que la evaluación debe centrarse en todo el pipeline de modelado y no en la simple comparación entre algoritmos.

Capítulo 6. CONCLUSIONES

6.1 Conclusiones del trabajo

Los resultados indican que el rendimiento del enfoque con ingenería de características + GA + XGBoost es condicional a la configuración del target (W, H); por tanto, **no hay superioridad global** frente a métodos tradicionales. En la **línea base** (W=15, H=5) el **MLP** presenta mejores métricas (Tabla 20). En cambio, con **W=5, H=8** el sistema con XGBoost supera al MLP (Tabla 20, Figura 51). En resumen, la ventaja depende de la calibración del target, arquitectura de modelo, más que del algoritmo por sí solo.

Cada modelo aporta ventajas y limitaciones. XGBoost Regressor, con feature engineering y selección evolutiva de variables, mostró estabilidad sin reajustes frecuentes. ARIMA exige actualización periódica de parámetros, y los MLP requieren un ajuste de su arquitectura y hiperparametros muy exhaustiva.

En cualquier caso, los objetivos se han cumplido: se ha desarrollado un sistema automatizado que, sin ser necesariamente el mejor en términos absolutos, ha demostrado en backtesting capacidad para generar señales de inversión y, en la mayoría de escenarios, beneficios en la cartera (Figura 47, Figura 55, Figura 58 y Tabla 20).

En suma, la eficacia de un sistema de predicción financiera no depende de un único algoritmo, sino del ajuste integral de modelado entero: desde la ingeniería de características hasta la validación y el tratamiento adaptativo de los modelos.

6.2 Conclusiones personales

Al inicio del proyecto tenía la convicción de que el sistema basado en: EDA automático, ingeniería de características avanzada + GA + XGBoost, sería claramente superior a cualquier alternativa. Las métricas mejoraban de forma progresiva: primero al comprobar la eficiencia de GA, después con la selección del tamaño del dataset y, por último, con el ajuste de hiperparámetros**.** Sin embargo, los experimentos mostraron otra realidad: al **aumentar los horizontes temporales**, aunque los beneficios crecieron, un **MLP mucho más simple**, entrenado solo con precios OHLCV**,** sin GA ni tuning, llegó a superar a mi sistema “todopoderoso”.

Esta experiencia me permitió comprender que, en *machine learning* financiero, **no existen soluciones definitivas**. Los resultados dependen del diseño completo del sistema: preprocesado y modelado del dataset**,** ingeniería de características**,** elección de algoritmos/arquitecturas y **condiciones de backtesting**, entre otros factores. Además, están condicionados por el régimen de mercado: en fases alcistas surgen más oportunidades de *trading*, mientras que en bajistas el sistema apenas amortigua el impacto (como en 2022). En lo personal, el aprendizaje es claro: más allá de los algoritmos, la clave está en **mantener una visión crítica y flexible**, aceptando que modelos sencillos pueden sorprender frente a enfoques más sofisticados.

Capítulo 7. FUTURAS LÍNEAS DE TRABAJO

El sistema desarrollado abre diversas posibilidades de investigación y mejora. Una de las más inmediatas consiste en perfeccionar el la logica de compra y venta de activos (acciones), que en esta versión de simulación se utilizó un umbral fijo de señal de compra de 0,51, stop loss del 3 % y take profit del 1,5 %. Aunque este esquema ha permitido comparar de manera homogénea distintos activos y configuraciones (Figura 46, Figura 49), en un escenario real convendría que estos parámetros se adaptaran dinámicamente al régimen del activo y del mercado. Por ejemplo, podrían ajustarse en función de métricas como volatilidad actual, riesgo asumido, ROC (ratio de cambio de precios) promedio o ATR (promedio de rango de cambio de precios verdadero), lo que permitiría que el sistema gestionara de forma más eficiente escenarios de alta o baja variabilidad.

Otra línea de mejora se relaciona con la extensión y diversificación del dataset. El entrenamiento de modelos se basó principalmente en datos desde 2010 (Figura 43), lo que deja fuera periodos críticos como la crisis de 2008. Incluir señales de indicadores basadas en series históricas más amplias permitiría evaluar la estabilidad de los modelos en ciclos de mercado más extremos y entrenar algoritmos más resilientes. Asimismo, convendría revisar el uso de un umbral fijo en la variable objetivo: como se observó en algunos histogramas (Figura 19), la distribución de los targets puede presentar sesgos o deriva, lo que provoca que el modelo pierda precisión. Una alternativa futura sería calcular el umbral de compra de forma adaptativa, en función de la media y desviación de la variable objetivo en cada periodo.

En cuanto a la gestión del capital, el backtester actual (4.6.2) asigna un capital inicial fijo de 10.000 USD por activo, pero en cada operación se invertía todo el capital disponible para activo en cuestión, lo que generaba una exposición total en cada entrada. Este enfoque fue útil para la comparación en igualdad de condiciones, pero se aleja de la operativa real.

En un escenario profesional, tanto la asignación inicial como la fracción invertida en cada compra dependen de métricas dinámicas como volatilidad, riesgo relativo, ATR (promedio de rango verdadero de precios) o rentabilidad reciente. Futuras versiones deberían incorporar un sistema adaptativo, capaz de decidir no solo cuánto capital destinar a cada activo, sino también qué porcentaje mantener en reserva, optimizando así la gestión del riesgo de la cartera.

Desde un punto de vista metodológico, sería interesante explorar modelos en ensemble específicos por activo. En lugar de un único predictor, se podrían entrenar varios modelos con diferentes targets (por ejemplo, horizontes cortos y largos) y combinar sus predicciones mediante esquemas de votación o ponderación. Esto aumentaría la estabilidad de las señales y reduciría la dependencia de una sola configuración.

Finalmente, para llegar a conclusiones finales más acercadas a la realidad es imprescindible añadir costes de transacciones y comisiones a la lógica de backtester. Dado que estos factores afectan de forma notable a las estrategias de alta rotación, su inclusión es clave para evaluar con mayor realismo la viabilidad práctica del sistema.

Capítulo 8. REFERENCIAS

[1] A. Elder, *Trading for a Living: Psychology, Trading Tactics, Money Management*. New York: Wiley, 1993.

[2] Y. Peng, P. H. M. Albuquerque, H. Kimura, and C. A. P. B. Saavedra, "Feature selection and deep neural networks for stock price direction forecasting using technical analysis indicators," *Machine Learning with Applications*, vol. 6, p. 100060, 2021. https://doi.org/10.1016/j.mlwa.2021.100060

[3] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018. https://doi.org/10.1016/j.ejor.2017.11.054

[4] Y. Gao, R. Wang, & E. Zhou, "Stock Prediction Based on Optimized LSTM and GRU Models," Scientific Programming, vol. 2021, Article ID 4055281, 2021. https://doi.org/10.1155/2021/4055281

[5] G. Ji, J. Yu, K. Hu, J. Xie, and X. Ji, "An adaptive feature selection schema using improved technical indicators for predicting stock price movements," *Expert Systems with Applications*, vol. 202, p. 116941, 2022. https://doi.org/10.1016/j.eswa.2022.116941

[6] A. S. Paramita, "A comparative study of feature selection techniques in machine learning for predicting stock market trends," *Journal of Applied Data Sciences*, vol. 4, no. 3, 2023. https://www.researchgate.net/publication/373988499_A_Comparative_Study_of_Feature_Selection_Techniques_in_Machine_Learning_for_Predicting_Stock_Market_Trends

[7] F. Moodi and A. J. Rafsanjani, "Evaluation of feature selection performance for identification of best effective technical indicators on stock market price prediction," *arXiv preprint*, arXiv:2310.09903, 2023. https://arxiv.org/abs/2310.09903

[8] C. A. Sagaceta-Mejía, E. A. Trejo-Ramírez, and A. J. Rivera-Rodríguez, "An Intelligent Approach for Predicting Stock Market Movements in Emerging Markets Using Technical Indicators," *Computational Economics and Applications*, vol. 3, no. 1, pp. 23–41, 2024. https://www.researchgate.net/publication/379260374_An_Intelligent_Approach_for_Predicting_Stock_Market_Movements_in_Emerging_Markets_Using_Optimized_Technical_Indicators_and_Neural_Networks

[9] Zouaghia, Z., Kodia Aouina, Z., & Ben Said, L. (2023). *Stock Movement Prediction Based On Technical Indicators Applying Hybrid Machine Learning Models*. En *2023 International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE. https://www.researchgate.net/publication/375986509_Stock_Movement_Prediction_Based_On_Technical_Indicators_Applying_Hybrid_Machine_Learning_Models

[10] E. F. Fama, "Efficient Capital Markets: A Review of Theory and Empirical Work," *The Journal of Finance*, vol. 25, no. 2, pp. 383-417, 1970.

[11] Pabuccu, H., & Barbu, A. "Feature selection with annealing for forecasting financial time series." Financial Innovation, 2024. https://jfin-swufe.springeropen.com/articles/10.1186/s40854-024-00617-3

[12] Masini, R. P., Medeiros, M. C., & Mendes, E. F. "Machine Learning Advances for Time Series Forecasting." arXiv, 2020. https://arxiv.org/abs/2012.12802

[13] Leung, T., & Zhao, T. "Financial Time Series Analysis and Forecasting with HHT Feature Generation and Machine Learning." arXiv, 2021. https://ideas.repec.org/p/arx/papers/2105.10871.html

[14] Abraham, R., El Samad, M., Bakhach, A. M., El-Chaarani, H., Sardouk, A., El Nemar, S., & Jaber, D. (2022). Forecasting a Stock Trend Using Genetic Algorithm and Random Forest. *Journal of Risk and Financial Management*, 15(5), 188. https://www.mdpi.com/1911-8074/15/5/188

[15] Sen, J. (2022). *A Comparative Study on the Sharpe Ratio, Sortino Ratio, and Calmar Ratio in Portfolio Optimization*. Disponible en researchgate: https://www.researchgate.net/publication/366517929_A_Comparative_Study_on_the_Sharpe_Ratio_Sortino_Ratio_and_Calmar_Ratio_in_Portfolio_Optimization

[16] SSRN Working Paper (2015). *Common Metrics for Performance Evaluation: Overview of Popular Performance Ratios*. Disponible en SSRN: https://papers.ssrn.com/sol3/Delivery.cfm/SSRN_ID2662054_code2424270.pdf?abstractid=2662054

[17] Mostafavi, S. M., & Hooman, A. R. (2025). Key technical indicators for stock market prediction. *Machine Learning with Applications, 20*, 100631. https://doi.org/10.1016/j.mlwa.2025.100631

[18] Chang, S.-H., Hsu, C.-W., Li, H.-Y., Zeng, W.-S., & Ho, J.-M. (2021). *Short-term stock price-trend prediction using meta-learning*. En *2021 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8). Disponible en: https://arxiv.org/abs/2105.13599

[19] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794. [Online]. Available: https://arxiv.org/abs/1603.02754

[20] XGBoost Developers. (2024). *XGBoost Documentation*. Disponible en: https://xgboost.readthedocs.io/en/stable/

[21] Zhila Yaseen Taha, Abdulhady Abas Abdullah, Tarik A. Rashid. (2021). *Optimizing Feature Selection with Genetic Algorithms: A Review of Methods and Applications.* Disponible en: https://www.researchgate.net/publication/384267727_Optimizing_Feature_Selection_with_Genetic_Algorithms_A_Review_of_Methods_and_Applications

[22] Kocyigit, E., Korkmaz, M., Sahingoz, O. K., & Diri, B. (2024). *Enhanced Feature Selection Using Genetic Algorithm for Machine-Learning-Based Phishing URL Detection*. *Applied Sciences*, 14(14), 6081. Disponible en: https://www.researchgate.net/publication/382234296_Enhanced_Feature_Selection_Using_Genetic_Algorithm_for_Machine-Learning-Based_Phishing_URL_Detection

[23] R. Namdari and T. S. Durrani, "A multilayer feedforward perceptron model in neural networks for predicting stock market short-term trends," *Operations Research Forum*, vol. 2, no. 3, pp. 1–19, 2021. Disponible en : https://link.springer.com/article/10.1007/s43069-021-00071-2

[24] R. Saidi, N. Ben Amor, and K. Ben Abdessalem, "Hybrid Feature Selection Method Based on the Genetic Algorithm and Pearson Correlation Coefficient," International Journal of Computer Applications, vol. 182, no. 37, pp. 6–12, 2019. [Online]. Disponible en: https://www.researchgate.net/publication/329513778_Hybrid_Feature_Selection_Method_Based_on_the_Genetic_Algorithm_and_Pearson_Correlation_Coefficient

[25] C. Chalvatzis and D. Hristu-Varsakelis, "Max-one selection of equity prediction models for portfolio construction," *Engineering Applications of Artificial Intelligence*, vol. 153, Art. no. 110694, Aug. 2025. Disponible en: https://www.sciencedirect.com/science/article/pii/S0952197625006943

[26] M. Costantini, M. Marcellino, and C. Schumacher, "Forecasting Errors, Directional Accuracy and Profitability of Currency Trading Models," International Institute for Applied Systems Analysis (IIASA), 2016. Disponible en: https://pure.iiasa.ac.at/id/eprint/12332/1/Forecasting Errors%2C%20Directional%20Accuracy%20and%20Profitability%20of%20Currency%20Trading.pdf

[27] D. Vezeris, T. Kyrgos y C. Schinas, "Take Profit and Stop Loss Trading Strategies Comparison in Combination with an MACD Trading System," *J. Risk Financial Manag.*, vol. 11, no. 3, p. 56, 2018. Disponible en: https://www.mdpi.com/1911-8074/11/3/56

[28] K. M. Kaminski y A. W. Lo, "When Do Stop-Loss Rules Stop Losses?," *J. Financial Markets*, vol. 18, pp. 234–254, 2014 (versiones de trabajo: SSRN/MIT). Disponible en : https://dspace.mit.edu/bitstream/handle/1721.1/114876/Lo_When%20Do%20Stop-Loss.pdf

[29] G. E. P. Box y G. M. Jenkins, *Time Series Analysis: Forecasting and Control*. San Francisco, CA, EE. UU.: Holden-Day, 1970.

[30] N. Marisetty, "Forecasting Selected International Stock Indices Returns by Using ARIMA Model," Working Paper, REVA Business School, REVA University, 18 Jul. 2024. Disponible en: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4898668

Capítulo 9. ANEXOS

9.1 Anexo 1. Lista completa de stocks (activos financieros).

Símbolo	Nombre de la empresa	Sector	Cap. mercado (billones USD)
AAPL	Apple Inc.	Electrónica de consumo	3,15
ADI	Analog Devices, Inc.	Semiconductores	0,11
ADP	Automatic Data Processing, Inc.	Software - Aplicaciones	0,13
AMAT	Applied Materials, Inc.	Equipos y materiales semiconductores	0,14
AMD	Advanced Micro Devices, Inc.	Semiconductores	0,18
AMGN	Amgen Inc.	Fabricantes de medicamentos - General	0,15
AMZN	Amazon.com, Inc.	Venta minorista por Internet	2,21
AXP	American Express Company	Servicios de crédito	0,21
BKNG	Booking Holdings Inc.	Servicios de viajes	0,17
CAT	Caterpillar Inc.	Maquinaria agrícola y pesada	0,16
CDNS	Cadence Design Systems, Inc.	Software - Aplicaciones	0,09
CMCSA	Comcast Corporation	Servicios de telecomunicaciones	0,13
COST	Costco Wholesale Corporation	Grandes almacenes	0,45
CSCO	Cisco Systems, Inc.	Equipos de comunicación	0,25
CVX	Chevron Corporation	Petróleo y gas integrados	0,25
DIS	The Walt Disney Company	Entretenimiento	0,2
EBAY	eBay Inc.	Venta minorista por Internet	0,03
GE	General Electric Company	Aeroespacial y defensa	0,23
GILD	Gilead Sciences, Inc.	Fabricantes de medicamentos - General	0,13
HON	Honeywell International Inc.	Conglomerados	0,14
IBM	International Business Machines Corp.	Servicios de tecnología informática	0,23
INTC	Intel Corporation	Semiconductores	0,1
INTU	Intuit Inc.	Software - Aplicaciones	0,19
JNJ	Johnson & Johnson	Fabricantes de medicamentos - General	0,37
KLAC	KLA Corporation	Equipos y materiales semiconductores	0,1
KO	The Coca-Cola Company	Bebidas no alcohólicas	0,3
LRCX	Lam Research Corporation	Equipos y materiales semiconductores	0,11
MAR	Marriott International, Inc.	Hoteles	0,07
MCD	McDonald's Corporation	Restaurantes	0,22
MCHP	Microchip Technology Incorporated	Semiconductores	0,03
MMM	3M Company	Conglomerados	0,08
MRK	Merck & Co., Inc.	Fabricantes de medicamentos - General	0,2
MS	Morgan Stanley	Mercados de capital	0,2
MSFT	Microsoft Corporation	Software - Infraestructura	3,34
MU	Micron Technology, Inc.	Semiconductores	0,1
NKE	NIKE, Inc.	Calzado y accesorios	0,09
NVDA	NVIDIA Corporation	Semiconductores	3
ORCL	Oracle Corporation	Software - Infraestructura	0,44

ORLY	O'Reilly Automotive, Inc.	Repuestos de automóviles	0,08
PEP	PepsiCo, Inc.	Bebidas no alcohólicas	0,18
PFE	Pfizer Inc.	Fabricantes de medicamentos - General	0,13
PG	The Procter & Gamble Company	Productos para el hogar y personales	0,38
QCOM	QUALCOMM Incorporated	Semiconductores	0,17
REGN	Regeneron Pharmaceuticals, Inc.	Biotecnología	0,06
SBUX	Starbucks Corporation	Restaurantes	0,1
T	AT&T Inc.	Servicios de telecomunicaciones	0,19
TXN	Texas Instruments Incorporated	Semiconductores	0,17
VRTX	Vertex Pharmaceuticals Incorporated	Biotecnología	0,11
WMT	Walmart Inc.	Grandes almacenes	0,77
XOM	Exxon Mobil Corporation	Petróleo y gas integrados	0,47

9.2 Anexo 2. Detalle técnico del Ratio Sharpe

El *Ratio Sharpe* cuantifica la rentabilidad adicional obtenida por cada unidad de riesgo asumida. Es un estándar para comparar activos o carteras ajustando por su volatilidad. Se define como:

$$\text{Sharpe} = \frac{\bar{r} - r_f}{\sigma}$$

Donde:

- $\bar{r} =$ rentabilidad media del activo
- $r_{f\,=}$ tasa libre de riesgo anual (en este estudio se ha utilizado un 2%)
- σ = volatilidad de los retornos

9.3 Anexo 3. Cálculo de retornos, volatilidad y taza de riesgo

Cálculo de retornos

Los **retornos diarios** se calculan de dos formas habituales:

Retorno aritmético:

$$r_t = \frac{(P_t - P_{t-1})}{P_{t-1}}$$

Donde P_t , P_{t-1} son precios de activo (stock) en día t y día anterior t-1 o t-n dónde n es tamaño de periodo del retorno.

Retorno logarítmico:

$$r_t = log\left(\frac{P_t}{P_{t-1}}\right)$$

Volatilidad anualizada. La volatilidad diaria se estima como la desviación estándar de los retornos diarios:

$$\sigma_{diaria} = \sqrt{\frac{1}{N-1}\sum_{t=1}^{N}(r_t - \bar{r})^2}$$

Para obtener la volatilidad anualizada, se multiplica por la raíz cuadrada del número de sesiones bursátiles anuales (≈252):

$$\sigma_{anual} = \sigma_{diaria} \times \sqrt{252}$$

Ajuste de la tasa libre de riesgo. La tasa libre de riesgo anual r_f se convierte a base diaria para la fórmula, dividiéndola por 252:

$$r_{f,\text{diaria}} = \frac{r_f}{252}$$

9.4 Anexo 4. Grupos de indicadores técnicos.

Familia	Indicador	Fórmula	Descripción
Tendencia	EMA	$EMA_t = \alpha \cdot P_t + (1-\alpha) \cdot EMA_{t-1}$	Media móvil exponencial
Tendencia	MACD	$MACD = EMA_{\text{fast}} - EMA_{\text{slow}}$	Diferencia entre EMAs
Tendencia	MACD Señal	Señal $= EMA_{\text{signal}}(\text{MACD})$	EMA del MACD
Tendencia	MACD Histograma	Histograma $= MACD -$ Signal	MACD menos Señal
Tendencia	ADX	$ADX = EMA(\text{DMI})$	Fuerza de la tendencia
Tendencia	DI+, DI-	DI^+, DI^-	Dirección movimiento
Momento	RSI	$RSI = 100 - \frac{100}{1+RS}$	Índice de fuerza relativa
Momento	Stochastic %K	$\%K = 100 \cdot \frac{P_{\text{close}} - P_{\text{low},n}}{P_{\text{high},n} - P_{\text{low},n}}$	Oscilador rápido

Momento	Stochastic %D	$\%D = EMA_3(\%K)$	Media exponencial de %K
Momento	Williams %R	$Williams\ \%R = -100 \cdot \frac{P_{\text{high},n} - P_{\text{close}}}{P_{\text{high},n} - P_{\text{low},n}}$	Extremos de precio
Momento	Momentum	$\text{Momentum}_n = P_{\text{close}} - P_{\text{close},n}$	Diferencia de cierre
Momento	CCI	$CCI = \frac{TP - EMA(TP)}{0.015 \cdot MAD}$	Desviación precio típico
Volatilidad	ATR	$ATR = EMA(\text{True Range})$	Rango verdadero medio
Volatilidad	Bollinger Bands	$\text{BB}_{\text{upper/lower}} = EMA_n \pm k \cdot \sigma_n$	Bandas de volatilidad
Volatilidad	Percent B	$\%B = \frac{P_{\text{close}} - \text{BB}_{\text{lower}}}{\text{BB}_{\text{upper}} - \text{BB}_{\text{lower}}}$	Posición respecto a bandas
Volatilidad	Z-score	$\text{Z-score} = \frac{P_{\text{close}} - EMA_n}{\sigma_n}$	Desviación estándar
Volatilidad	Bollinger Width	$\text{Bollinger Width} = \frac{\text{BB}_{\text{upper}} - \text{BB}_{\text{lower}}}{EMA_n}$	Anchura de bandas
Volumen	OBV	$OBV_t = OBV_{t-1} + V_t$ si $P_t > P_{t-1}$ $OBV_t = OBV_{t-1} - V_t$ si $P_t < P_{t-1}$ $OBV_t = OBV_{t-1}$ si $P_t = P_{t-1}$	Volumen en balance
Volumen	ADL	$ADL_t = ADL_{t-1} + \left(\frac{(P_{\text{close}} - P_{\text{low}}) - (P_{\text{high}} - P_{\text{close}})}{P_{\text{high}} - P_{\text{low}}}\right) \cdot V_t$	Acumulación/distribución
Volumen	Chaikin Oscillator	$\text{CHO} = EMA_3(ADL) - EMA_{10}(ADL)$	Oscilador Chaikin
Otros	ROC	$ROC_n = 100 \cdot \frac{P_t - P_{t-n}}{P_{t-n}}$	Tasa de cambio

Otros	TEMA	$TEMA = 3 \cdot EMA - 3 \cdot EMA(EMA) + EMA\big(EMA(EMA)\big)$	Triple EMA
Otros	SAR	$SAR_{t+1} = SAR_t + \alpha \cdot (EP - SAR_t)$	Parabolic SAR
Avanzados	Connors RSI	$\text{Connors RSI} = \frac{RSI_{n_1} + RSI_{\text{streak}} + \text{PercentRank}_{n_3}}{3}$	Connors RSI
Avanzados	Ultimate Oscillator	$UO = 100 \cdot \frac{4 \cdot \text{AVG}_7 + 2 \cdot \text{AVG}_{14} + \text{AVG}_{28}}{4 + 2 + 1}$	Ultimate Oscillator
Avanzados	Ulcer Index	$\text{UlcerIndex}_n = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{P_{\text{close},i} - \max(P_{\text{close},1:i})}{\max(P_{\text{close},1:i})} \times 100\right)^2}$	Índice de Ulcer
Targets	Ángulo universal tendencia	$\theta = \arctan\left(\frac{m}{P_t \cdot r}\right)$	Ángulo normalizado de pendiente de regresión lineal dentro de ventana móvil.

9.5 Anexo 5. Código Python de importación de datos históricos de stocks

Programa Python **Download_Historical_Data.py** Realiza la descarga de datos históricos con movimientos de precios diarios de 50 stocks desde Yahoo Finance, adicionalmente realiza imputación de valores faltantes con estrategia **median**.

```
import pandas as pd
import yfinance as yf
import numpy as np
from sklearn.impute import KNNImputer

def apply_etl(df, ohlcv_strategy='median', max_null_pct=0.10, knn_neighbors=5):
    """
    Imputador parametrizable para valores nulos en las columnas OHLCV.

    Estrategias disponibles:
        - 'median': Rellena valores faltantes con la mediana de cada columna.
        - 'average': Rellena valores faltantes con la media aritmética.
        - 'last_known': Propaga el último valor conocido (forward-fill + backward-
          fill).
        - 'moda' / 'most_frequent': Utiliza el valor más frecuente (moda).
        - 'knn_imputation': Imputación avanzada con KNNImputer (requiere sklearn).

        El parámetro max_null_pct permite fijar un umbral máximo de filas con nulos
        por símbolo. Si lo supera, el proceso para ese símbolo y lo ignora (control de
        calidad).

        Al final, los precios se redondean a 5 decimales y el volumen se convierte a
        entero.
    """
    ohlcv = ["Open", "High", "Low", "Close", "Volume"]

    # --- Control de calidad: verificamos % de filas con valores nulos ---
    total = len(df)
    filas_con_nulos = df[ohlcv].isnull().any(axis=1).sum()
    null_pct = filas_con_nulos / total

    # Si no hay filas con nulos, pasamos al paso de Redondeo y Formatos de ETL
    if (filas_con_nulos > 0):

        # Si el porcentaje de filas con nulos supera el umbral, detenemos el proceso
        # para este símbolo para evitar problemas de calidad en el dataset final.
        if null_pct > max_null_pct:
            raise ValueError(
                f"Demasiados valores faltantes en {df['Symbol'].iloc[0]}: "
                f"{null_pct:.1%} de las filas tienen algún nulo en OHLCV. "
                f"Proceso detenido para este símbolo por calidad insuficiente."
            )

        # --- Imputación robusta según estrategia seleccionada ---
        if ohlcv_strategy == 'median':
            # Rellenar NaN con la mediana de cada columna
            for col in ohlcv:
                med = df[col].median()
                df[col] = df[col].fillna(med)

        elif ohlcv_strategy == 'average':
            # Rellenar NaN con la media aritmética de cada columna
```

```
            for col in ohlcv:
                avg = df[col].mean()
                df[col] = df[col].fillna(avg)

        elif ohlcv_strategy == 'last_known':
            # Forward-fill y backward-fill: el valor válido más próximo
            df[ohlcv] = df[ohlcv].fillna(method='ffill').fillna(method='bfill')

        elif ohlcv_strategy == 'moda' or ohlcv_strategy == 'most_frequent':
            # Moda (valor más frecuente), útil para volumen o casos discretos
            for col in ohlcv:
                moda = df[col].mode()
                if not moda.empty:
                    df[col] = df[col].fillna(moda[0])
                else:
                    df[col] = df[col].fillna(0)  # fallback para columnas vacías

        elif ohlcv_strategy == 'knn_imputation':
            # Imputación KNN: usa el patrón de los datos vecinos
            imputer = KNNImputer(n_neighbors=knn_neighbors)
            df[ohlcv] = imputer.fit_transform(df[ohlcv])

        else:
            raise ValueError(f"Estrategia '{ohlcv_strategy}' no reconocida.")

    # Redondear precios a 5 decimales (precisión suficiente para análisis financiero)
    for col in ["Open", "High", "Low", "Close"]:
        df[col] = df[col].round(5)

    # Convertimos Volumen a entero (por integridad de datos y almacenamiento)
    df["Volume"] = df["Volume"].astype(int)

    return df

# ---------- CARGA DE DATOS PRINCIPAL ---------- #
symbols = [ "AAPL", "ADI", "ADP", "AMAT", "AMD", "AMGN", "AMZN", "AXP", "BKNG",
            "CAT", "CDNS", "CMCSA", "COST", "CSCO", "CVX", "DIS", "EBAY", "GE",
            "GILD", "HON", "IBM", "INTC", "INTU", "JNJ", "KLAC", "KO", "LRCX",
            "MAR", "MCD", "MCHP", "MMM", "MRK", "MS", "MSFT", "MU", "NKE", "NVDA",
            "ORCL", "ORLY", "PEP", "PFE", "PG", "QCOM", "REGN", "SBUX", "T", "TXN",
            "VRTX", "WMT", "XOM", "^GSPC"]

start_date = "1999-07-01"
end_date = pd.Timestamp.today().strftime("%Y-%m-%d")
all_data = []

for symbol in symbols:

    print(f"Descargando {symbol} ...")
    df = yf.download(symbol, start=start_date, end=end_date, progress=False,
    auto_adjust=True)

    # Si columnas son multinivel (MultiIndex), eliminamos el nivel extra
    if isinstance(df.columns, pd.MultiIndex):
        df.columns = df.columns.get_level_values(0)

    # --- Control crítico de ausencia de datos ---
    if df.empty:
```

```
            # Detenemos el proceso porque no se admite un dataset incompleto.
            raise ValueError(
                f"ERROR: No se encontraron datos para el símbolo '{symbol}'. "
                f"Proceso detenido para evitar inconsistencias y problemas posteriores en
                  el análisis multi-stock."
            )

        # --- Renombramos la columna de fecha  eliminando el índice actual ---
        df = df.reset_index().rename(columns={"Date": "Fecha"})
        df["Symbol"] = symbol
        df = df[["Fecha", "Symbol", "Open", "Close", "High", "Low", "Volume"]]

        # --- ETL: Imputación parametrizada de valores nulos en OHLCV ---
        # Aplicamos el ETL con la estrategia de imputación de NaN seleccionada
        df = apply_etl(
            df,
            ohlcv_strategy='median',  # La estrategia para experimentos: 'median',
               'average', 'moda', 'knn_imputation'

            max_null_pct=0.10,  # Máximo 10% de filas con nulos permitidas
            knn_neighbors=5     # Número de vecinos para KNNImputer
        )

        # --- Añadimos el DataFrame procesado a la lista de todos los datos ---
        all_data.append(df)

# --- Concatenación final ---
final_df = pd.concat(all_data, axis=0)

# --- Ordenamos por símbolo y fecha para consistencia ---
final_df = final_df.sort_values(["Symbol", "Fecha"]).reset_index(drop=True)

# --- Guardamos datos en CSV (formato estándar para ETL financieros) ---
final_df.to_csv("Data/AllStocksHistoricalData.csv", sep=";", index=False)
```

9.6 Anexo 6. Creación de dataset completo

Fragmento de clase **Feature_Generator.py**, representa código Python de cálculo algunos indicadores como: EMA, RSI y MACD.

```
import pandas as pd
import numpy as np
from pathlib import Path
from concurrent.futures import ProcessPoolExecutor
from sklearn.linear_model import LinearRegression
from stock_indicators import indicators
from stock_indicators import Quote
import traceback
import warnings

# ----------------------------
# Clase FeatureGenerator
# Genera datos de características para análisis financiero
# Genera señales de indicadores técnicos como EMA, RSI , etc
# ----------------------------
```

```
class Feature_Generator:
    def __init__(self, df, date_col='Fecha'):
        self.df = df.copy()
        self.date_col = date_col
        self._quotes = self._build_quotes()

    def _build_quotes(self):
        fechas = np.array(pd.to_datetime(self.df[self.date_col]).dt.to_pydatetime())
        opens = self.df['Open'].values
        highs = self.df['High'].values
        lows = self.df['Low'].values
        closes = self.df['Close'].values
        volumes = self.df['Volume'].values

        return [
            Quote(
                date=fechas[i],
                open=opens[i],
                high=highs[i],
                low=lows[i],
                close=closes[i],
                volume=volumes[i]
                )
                for i in range(len(closes))
            ]

    def add_ema(self, periods_ema=[10, 20, 50]):
        new_cols = {}
        for p_ema in periods_ema:
            ema_result = indicators.get_ema(self._quotes, p_ema)
            ema = np.array([x.ema if x.ema is not None else np.nan for x in
            ema_result])
            new_cols[f'EMA_{p_ema}'] = ema
        if new_cols:
            self.df = pd.concat([self.df, pd.DataFrame(new_cols,
            index=self.df.index)], axis=1)

        return self


    def add_rsi(self, periods_rsi=[14]):
        new_cols = {}
        for p_rsi in periods_rsi:

            rsi_result = indicators.get_rsi(self._quotes, p_rsi)
            rsi = np.array([x.rsi if x.rsi is not None else np.nan for x in
            rsi_result])

            new_cols[f'RSI_{p_rsi}'] = rsi

        if new_cols:
            self.df = pd.concat([self.df, pd.DataFrame(new_cols,
            index=self.df.index)], axis=1)
        return self

    def add_macd(self, fast_slow_signals=[[12, 26, 9]]):

        new_cols = {}
        for fast, slow, signal in fast_slow_signals:
```

```
            macd_result = indicators.get_macd(self._quotes, fast, slow, signal)
            new_cols[f'MACD_{fast}_{slow}_{signal}'] = [x.macd if x.macd is not None
            else np.nan for x in macd_result]

            new_cols[f'MACD_SIGNAL_{fast}_{slow}_{signal}'] = [x.signal if x.signal
            is not None else np.nan for x in macd_result]

            new_cols[f'MACD_HISTOGRAM_{fast}_{slow}_{signal}'] = [x.histogram if
            x.histogram is not None else np.nan for x in macd_result]

        if new_cols:
            self.df = pd.concat([self.df, pd.DataFrame(new_cols,
            index=self.df.index)], axis=1)

        return self
```

Fragmento de clase **Build_Dataset_All_Features.py**, encargada de orquestar la generación de dataset con todos juegos de indicadores y las variables objetivo. A continuación, se muestra fragmento de Python de ordenes de crear columnas con señales de indicadores calculados por **Feature_Generator.py**

```
import pandas as pd
import numpy as np
from pathlib import Path
from concurrent.futures import ProcessPoolExecutor
from stock_indicators import indicators
from stock_indicators import Quote
from Feature_Generator import Feature_Generator
import traceback
import os
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
import time

# ---------------------------
# Función parcial para ejecución en paralelo
# ---------------------------
def procesar_tupla(tupla):
    symbol, df_sub = tupla
    return procesar_symbol(df_sub, symbol)

# ---------------------------
# Función para procesar cada símbolo
# ---------------------------
def procesar_symbol(df_sub, symbol ):
    try:

        df_sub = df_sub.sort_values(by="Fecha")
        fg = Feature_Generator(df_sub, date_col="Fecha")

        # EMA con periodos comunes
        ema_periods = [5, 7, 10, 12, 14]

        # RSI con periodos comunes
        rsi_periods = [8, 10, 12, 14]
```

```
# MACD con los conjuntos de parámetros (Fast, Slow, Signal)
macd_periods = [[12, 26, 9], [7, 18, 9], [10, 20, 7]]

# Stochastic Oscillator con los conjuntos de parámetros (K, D, Smooth)
stochastic_periods = [[14, 3, 3]]

# Williams %R con periodos comunes
williams_periods = [10, 14, 17]

# CCI con periodos estándar
cci_periods = [9, 12, 14, 15, 20]

# Rate of change con periodos estándar
roc_periods = [1, 2, 3, 5, 7, 10, 12, 14]
```

Segmento de creación de columnas correspondiente a distintos indicadores técnicos de mercado que hace uso de clase **Feature_Generator**.

```
# Añadimos todas las características al generador
fg.add_ema(ema_periods)
fg.add_rsi(rsi_periods)
fg.add_macd(macd_periods)
fg.add_stochastic(stochastic_periods)
fg.add_williams_r(williams_periods)
fg.add_cci(cci_periods)
fg.add_roc(roc_periods)
fg.add_roc_on_columns(periods=roc_periods, columns=["Open", "High", "Low"])
fg.add_adx(adx_periods)
```

9.7 Anexo 7. Fórmulas de estandarización y transformación de características

Transformación logarítmica

Reduce el sesgo en variables estrictamente positivas con asimetría fuerte. Útil para escalas exponenciales o colas largas a la derecha. Para variables estrictamente positivas (X > 0) con un sesgo muy fuerte.

$$X' = \log(1 + X)$$

Transformación Box-Cox

Aplica una transformación de potencia a variables estrictamente positivas para normalizar distribuciones con sesgo o curtosis moderada/alta. El parámetro λ se ajusta automáticamente para maximizar la normalidad.

$$X' = \frac{X^{\lambda} - 1}{\lambda},\ \lambda \neq 0; \quad X' = log(X)\ ,\ \lambda = 0$$

Transformación Yeo-Johnson

Es una transformación de potencia generalizada que permite trabajar tanto con valores positivos como negativos. El parámetro λ se optimiza automáticamente para normalizar la variable.

Para X ≥ 0, λ ≠ 0:

$$X' = \frac{(X + 1)^{\lambda} - 1}{\lambda}$$

Para X ≥ 0, λ = 0:

$$X' = \log(X + 1)$$

Para X < 0, λ ≠ 2:

$$X' = -\frac{(-X + 1)^{2-\lambda} - 1}{2 - \lambda}$$

Para X < 0, λ = 2:

$$X' = -\log(-X + 1)$$

Transformación Robust Scaler

El escalado robusto transforma cada variable restando su mediana y dividiendo por el rango intercuartílico (IQR), definido como la diferencia entre el tercer cuartil (Q_3) y el primer cuartil (Q_1):

$$X' = \frac{X - Mediana(X)}{IQR(X)}$$

(IQR es el rango intercuartílico: cuartil Q3 menos cuartil Q1)

Transformación Standard Scaler

La estandarización global transforma cada variable para que tenga media cero y varianza uno. Para cada observación, se resta la media de la variable y se divide entre su desviación estándar:

$$X' = \frac{x_i - \mu}{\sigma}$$

Donde:

x_i es la variable, μ es la media de la variable y σ es la desviación estándar de la variable.

9.8 Anexo 8. Estandarización y transformación de características automática

Clase Python encargada de llevar a cabo la transformación automática de características según sus estadísticas como: sesgo, curtosis, valores anómalos, etc.

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import PowerTransformer, RobustScaler, StandardScaler,
FunctionTransformer
from scipy.stats import kurtosis, skew

class AdaptiveFeatureNormalizer:
    """
    Transformador adaptativo de variables numéricas basado en skewness, kurtosis y
    outliers.
    Selecciona automáticamente la mejor transformación para cada feature:
       - Aplica log-transform (log1p) si el sesgo es muy fuerte y la variable es
         estrictamente positiva.
       - Aplica Yeo-Johnson si el sesgo es muy fuerte y existen valores negativos.
       - Aplica Box-Cox si el sesgo es moderado y todos los valores son positivos.
       - Aplica Yeo-Johnson si el sesgo es moderado y hay valores negativos.
       - Si la variable es aproximadamente simétrica y sin colas largas, aplica
       RobustScaler.
    Finalmente, aplica StandardScaler global (media 0, varianza 1).

    El proceso se realiza *stock por stock* para evitar que se mezclen distribuciones
    de distintos activos.

    La selección es totalmente automática: sólo se usan propiedades estadísticas de
    cada serie.
    """
    def __init__(self, skew_threshold=0.8, log_skew_threshold=2.0,
       kurtosis_threshold=4.0):

        # Umbral de skewness (sesgos) para considerar transformación logarítmica
        self.skew_threshold = skew_threshold

        # Umbral de skewness para considerar transformación logarítmica fuerte
        # (log_skew_threshold > 2 indica un sesgo muy fuerte)
        self.log_skew_threshold = log_skew_threshold

        # Achacamiento o apuntamiento de la distribución.
        # Umbral de kurtosis para considerar colas largas
        # (kurtosis > 3 indica colas más largas que una distribución normal)

        self.kurtosis_threshold = kurtosis_threshold
        # Diccionario para almacenar transformadores (objetos como) por columna
        self.transformers = {}
        self.final_scaler = None

    def fit(self, X):

        #reseteamos el diccionario de transformadores
        self.transformers = {}
        X_tr = X.copy()
```

```
        for col in X.columns:

            # Calculamos skewness, kurtosis y outliers
            skew_val = series.skew()
            kurt_val = kurtosis(series, fisher=True)
            q1, q3 = np.percentile(series, [25, 75])
            iqr = q3 - q1

            # Contamos outliers usando el criterio de 3*IQR
            outliers = ((series < q1 - 3*iqr) | (series > q3 + 3*iqr)).sum()

            # Selección automática de transformación
            # El sesgo es muy fuerte?
            if abs(skew_val) > self.log_skew_threshold:
                if (series > 0).all():
                    # Positivo (todos los valores son positivos)
                    transformer = FunctionTransformer(np.log1p, validate=True)
                else:
                    # Existen valores negativos
                    transformer = PowerTransformer(method='yeo-johnson',
                     standardize=False)

            # El sesgo es moderado
            elif abs(skew_val) > self.skew_threshold or abs(kurt_val) >
              self.kurtosis_threshold
              or outliers > 0:

                if (series > 0).all():
                    # Positivo (todos los valores son positivos)
                    transformer = PowerTransformer(method='box-cox',
                     standardize=False)
                else:
                    transformer = PowerTransformer(method='yeo-johnson',
                     standardize=False)
            else:
                # Aproximadamente simétrico y sin colas largas
                transformer = RobustScaler()

            transformer.fit(series.values.reshape(-1, 1))
            self.transformers[col] = transformer
            X_tr[col] = transformer.transform(X[[col]].values)

        # Estandarización global final (media 0, varianza 1)
        self.final_scaler = StandardScaler()
        self.final_scaler.fit(X_tr.values)
        return self

    # Transformación de los datos
    # Aplica las transformaciones definidas a partir de los datos de entrada
    def transform(self, X):
        X_tr = X.copy()

        # Aplicamos las transformaciones creadas en fit
        for col, transformer in self.transformers.items():
            X_tr[col] = transformer.transform(X[[col]].values)

        # Estandarización final y devolución del DataFrame transformado
        X_scaled = pd.DataFrame(
            self.final_scaler.transform(X_tr.values),
            columns=X_tr.columns, index=X_tr.index
        )
        return X_scaled
```

```
    # Método fit_transform que combina fit y transform en una sola llamada
    def fit_transform(self, X):
        return self.fit(X).transform(X)

# =============== Punto de entrada vamos stock por stock ===============

#Leemos el dataset completo (con todos los sombolos y sus indicadores)
df = pd.read_csv("DATA/Dataset_All_Features.csv", parse_dates=['Fecha'], sep=';')

# --- Selección de columnas de indicadores técnicos ---
# Excluimos las columnas de EMA, TARGET, BIN, Fecha, Symbol  y OCHLV
# Las columnas de indicadores son aquellas que no empiezan por 'EMA_', 'TARGET_',
'BIN_' y no son OCHLV
# y no son de tipo objeto (dtype != 'O')
indicadores = [
    col for col in df.columns
    if (
        not col.startswith('EMA_') and
        not col.startswith('TARGET_') and
        not col.startswith('BIN_') and
        col not in ['Fecha', 'Symbol', 'Open', 'Close', 'High', 'Low', 'Volume']
        and df[col].dtype != 'O'
    )
]

# --- Procesamos stock por stock para evitar el solapamiento de transformación y
escalado de indicadores entre distintos stocks ---
dfs_final = []
for symbol, df_sym in df.groupby('Symbol'):

    # Seleccionar solo las columnas de indicadores para el símbolo actual
    df_indicadores = df_sym[indicadores].copy()

    auto_tf = AdaptiveFeatureNormalizer()
    df_indicadores_trans = auto_tf.fit_transform(df_indicadores)

    # Volvemos a montar el dataframe completo
    df_resultado = df_sym.copy()

    # Pasamos las columnas de indicadores transformadas al dataframe original
    df_resultado[indicadores] = df_indicadores_trans

    # Añadimos el DataFrame transformado a la lista de resultados (un dataframe por
    símbolo)
    dfs_final.append(df_resultado)

df_total = pd.concat(dfs_final, ignore_index=True)

# --- Volcamos el dataset completo transformado ---
df_total.to_csv("DATA/Dataset_All_Features_Transformado.csv", sep=';', index=False)

print(" Dataset completo guardado en DATA/Dataset_All_Features_Transformado.csv\n"
      "Incluye: fechas, Symbol, OCHLV, binarias, targets y features de indicadores
transformados.")
```

9.9 Anexo 9. Algoritmo genético de selección de características

Segmentos clave de código fuente Python **GA_Feature_Selection.py** que realiza la selección genética de características.

Función de evaluación de individuos conforme parámetros como: modelo, conjunto de características de un individuo, datos para entrenar y testear el modelo.

```
def fitness(self, individual, return_metrics=False):
    """
    Aquí estamos evaluando un individuo: probamos su subconjunto de features
```

```
en el modelo correspondiente y devolvemos la métrica principal.
El fitness mide la calidad del subconjunto de variables elegido por este
individuo, usando el modelo y métrica definidos (ElasticNet, Ridge, XGBRegressor,
MLP...).
"""
# Si el individuo no tiene ninguna feature activa, devolvemos la peor métrica
posible.
# Esto penaliza soluciones que "no seleccionan nada" y fuerza a usar algún input.
if np.sum(individual) == 0:
   resultado = (np.inf, np.inf, np.inf, -np.inf)
   return resultado if return_metrics else resultado[self.metric_index]

# Elegimos solo las columnas (features) activas según el cromosoma del individuo.
# idx es un array (máscara) de índices donde el individuo tiene 1 (features
  activas).
# Esto nos permite seleccionar dinámicamente las columnas de X_train y X_test.
idx = np.where(individual == 1)[0]
X = self.X_train[:, idx]
X_test = self.X_test[:, idx]
y = self.y_train
y_test = self.y_test

try:
    # Entrenamos y predecimos con el modelo solicitado. Cada uno se maneja
      diferente:
    if self.fitness_model == "ElasticNet":

        # Si queremos escalar el target (solo útil en algunos casos, sobre todo
          para ElasticNet)
        if self.scale_target:
            scaler = MinMaxScaler()
            # Aquí añadimos una dimensión a y/y_test, escalamos y luego
            devolvemos a vector 1D.
            y_train_s = scaler.fit_transform(y[:, None]).flatten()
            y_test_s = scaler.transform(y_test[:, None]).flatten()
        else:
            y_train_s = y
            y_test_s = y_test

        # Entrenamos el modelo ElasticNet con parámetros fijos (se puede tunear)
        model = ElasticNet(alpha=0.01, l1_ratio=0.5, max_iter=1000,
        model.fit(X, y_train_s)
        y_pred = model.predict(X_test)

        # Si escalamos, deshacemos el escalado para poder comparar contra el y
        original.
        if self.scale_target:
            y_pred = scaler.inverse_transform(y_pred[:, None]).flatten()
            y_test = scaler.inverse_transform(y_test_s[:, None]).flatten()

    elif self.fitness_model == "Ridge":
        # Modelo Ridge: similar, pero sin escalado del target por defecto.
        model = Ridge(alpha=1.0, random_state=self.random_state)
        model.fit(X, y)
        y_pred = model.predict(X_test)

    elif self.fitness_model == "XGBRegressor":
        # Modelo XGBoost: potente para relaciones no lineales y selección de
        features.
```

```
            model = XGBRegressor(n_estimators=60, max_depth=4, n_jobs=32,
                                 random_state=self.random_state, verbosity=0)
            model.fit(X, y)
            y_pred = model.predict(X_test)
```

Funciones de selección de individuos para la siguiente generación, cruce de cromosomas de individuos y la mutación

```
def select(self, scores):
        """
        Aquí elegimos los individuos que pasan a la próxima generación:
        Primero, guardamos los élite y luego rellenamos el resto con torneos.
        """
        if self.fitness_metric == "r2":
            indices_ordenados = np.argsort(scores)[::-1]  # De mayor a menor
        else:
            indices_ordenados = np.argsort(scores)  # De menor a mayor

        elite = [self.population[i] for i in indices_ordenados[:self.elite]]
        rest = []

        # Aquí jugamos torneos para llenar el resto de la población
        while len(rest) < self.n_pop - self.elite:
            candidates = np.random.choice(self.n_pop, self.tournament_size,
                                          replace=False)
            if self.fitness_metric == "r2":
                best_idx = candidates[np.argmax(scores[candidates])]
            else:
                best_idx = candidates[np.argmin(scores[candidates])]
            rest.append(self.population[best_idx].copy())
        return elite + rest


    def crossover(self, parent1, parent2):
        """
        Aquí estamos mezclando los genes de dos padres, eligiendo aleatoriamente
        cada feature de uno u otro. Así sale el hijo.
        """
        # Máscara binaria aleatoria para mezclar genes de los padres (crossover)
        mask = np.random.randint(0, 2, self.n_features)
        child = (parent1 & mask) | (parent2 & ~mask)
        return self.enforce_active_limits(child)


    def mutate(self, individual):
        """
        Aquí revisamos cada feature y, con cierta probabilidad, la cambiamos de
        estado. Es la chispa de variabilidad genética.
        """
        mutated = individual.copy()

        # Aquí recorremos cada feature y aplicamos la mutación si toca
        for i in range(self.n_features):
            if random.random() < self.mut_prob:
                # Convertimos el bit: si estaba activo, lo apagamos; si no, lo
                 encendemos.
```

```
                if mutated[i] == 1:
                    mutated[i] = 0
                else:
                    mutated[i] = 1

            # Nos aseguramos de que el individuo sigue cumpliendo los límites de
            features activas.

            # Si no, lo ajustamos para que tenga el número correcto de features
            activas.
            return self.enforce_active_limits(mutated)
```

9.10 Anexo 10. Implementación de MLP con soporte de GPU

Segmento de código Python de clase **GA_Feature_Selection** (función fitness) dónde se implementa una red neuronal MLP utilizando librería PyTorch (https://pytorch.org/). El modelo se define dinámicamente según el número de variables de entrada (`X.shape[1]`) seleccionadas por cada cromosoma, y consta de una capa oculta con 32 neuronas (activación Sigmoid), regularización con `nn.Dropout(0.2)`, y una capa de salida `nn.Linear(32, 1)` con función de activación sigmoide.

```
import torch
import torch.nn as nn

elif self.fitness_model == "MLP-Torch":
    # Aquí usamos una red neuronal propia hecha en PyTorch, compatible con GPU
    torch.manual_seed(self.random_state)
    np.random.seed(self.random_state)

    # Si el número de features cambia, reconstruimos la arquitectura (dinámica).
    if (self.torch_mlp is None) or (X.shape[1] != self.torch_input_dim):
        self.torch_input_dim = X.shape[1]
        self.torch_mlp = nn.Sequential(
            nn.Linear(self.torch_input_dim, 32),
            nn.Sigmoid(),
            nn.Dropout(0.2),
            nn.Linear(32, 1),
            nn.Sigmoid()
            ).to(self.torch_device)

    # Aquí reseteamos los pesos para que no arrastre información de generaciones
    # anteriores.
    self.randomize_torch_weights()

    # Convertimos los datos a tensores y los subimos al dispositivo GPU.
    X_torch = torch.tensor(X, dtype=torch.float32).to(self.torch_device)
    y_torch = torch.tensor(y, dtype=torch.float32).view(-1, 1).to(self.torch_device)
    X_test_torch = torch.tensor(X_test, dtype=torch.float32).to(self.torch_device)

    # Optimizador Adam y función de pérdida MSE.
    optimizer = torch.optim.Adam(self.torch_mlp.parameters(), lr=0.001,
                                 weight_decay=0.0003)
    loss_fn = torch.nn.MSELoss()
```

```
        # Early stopping manual, igual que en sklearn.
        best_loss = float('inf')
        epochs_no_improve = 10
        n_iter_no_change = 10
        best_model_state = None

        # Entrenamiento por épocas (máximo 500).
        for epoch in range(500):
            self.torch_mlp.train()

            # PyTorch suma los gradientes por defecto! Lo limpiamos al inicio de cada
            época.
            optimizer.zero_grad()
            # Aquí hacemos la pasada hacia adelante y calculamos la pérdida.
            output = self.torch_mlp(X_torch)
            # Calculamos la pérdida y hacemos backpropagation.
            loss = loss_fn(output, y_torch)
            # Aquí hacemos el paso de optimización.
            loss.backward()
            # Actualizamos los pesos del modelo.
            optimizer.step()

            # Aquí guardamos la pérdida actual para early stopping.
            curr_loss = loss.item()

            # Guardamos el mejor estado del modelo (early stopping)
            # si la pérdida mejora, actualizamos el mejor estado.
            # Si no mejora, contamos cuántas épocas llevamos sin mejora.
            if curr_loss < best_loss - 1e-4:
                best_loss = curr_loss
                epochs_no_improve = 0
                best_model_state = self.torch_mlp.state_dict()
            else:
                epochs_no_improve += 1
                if epochs_no_improve >= n_iter_no_change:
                    break

        # Recuperamos el mejor estado (si lo hay)
        if best_model_state is not None:
            self.torch_mlp.load_state_dict(best_model_state)

        # Predicción final sobre el set de test.
        self.torch_mlp.eval()

        # Aquí hacemos la predicción y la pasamos a CPU para convertirla a numpy.
        with torch.no_grad():
            y_pred = self.torch_mlp(X_test_torch).cpu().numpy().flatten()

        # Calculamos todas las métricas de rendimiento (RMSE, MAE, MSE, R2).
        resultado = self.calcular_metricas(y_test, y_pred)

        # Según lo que nos pidan, devolvemos solo la métrica principal o el paquete
        # entero de métricas.
        return resultado if return_metrics else resultado[self.metric_index]
```

El modelo se despliega en GPU si está disponible (propiedad `self.torch_device`).

```
# Si elegimos 'MLP-Torch' como modelo, configuramos la red neuronal y la GPU (si
existe).
if self.fitness_model == "MLP-Torch":
    print(f"torch.cuda.is_available = {torch.cuda.is_available()}")
    if torch.cuda.is_available():
        print("Dispositivos CUDA disponibles:")
        print(torch.cuda.device_count())
        print(torch.cuda.current_device())
        print(torch.cuda.get_device_name())

    # Elegimos automáticamente GPU si hay disponible
    self.torch_device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    self.torch_mlp = None
    self.torch_input_dim = self.n_features

    # Definimos la arquitectura del MLP con PyTorch (capa oculta de 32, dropout,
    # etc.)
    self.torch_mlp = nn.Sequential(
        nn.Linear(self.torch_input_dim, 32),
        nn.Sigmoid(),
        nn.Dropout(0.2),
        nn.Linear(32, 1),
        nn.Sigmoid()
        ).to(self.torch_device)
```

9.11 Anexo 11. Código de selección GA y ajuste de hiperparámetros

Implementación de selección de características mediante GA y ajuste de hiperparámetros aplicado a cada activo financiero (stock) de nuestra cartera.

Código Python de selección de 25 características para cada activo a partir de dataset completo:

```
import os
import pandas as pd
import numpy as np
from xgboost import XGBRegressor
from GA.GA_Feature_Selection import GA_Feature_Selection

# CONFIGURACIÓN
SYMBOLS_TO_TEST = ['NVDA', 'AAPL', 'AMZN', 'LRCX', 'SBUX', 'REGN', 'KLAC', 'BKNG',
'AMD', 'VRTX', 'MAR', 'CDNS',  'CAT', 'INTU', 'GILD',  'MU', 'EBAY', 'AXP', 'AMAT',
'COST', 'MSFT', 'ORCL', 'ADI', 'MS', 'NKE']

TARGET_COL = 'TARGET_TREND_ANG_15_5'
N_FEATURES = 25

GA_TRAIN_YEARS = [2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017]
GA_TEST_YEARS =  [2018, 2019]

DATA_PATH = "DATA/Dataset_All_Features_Transformado.csv"
RESULTS_PATH = "Resultados/Features_Seleccionadas_GA.csv"

# FUNCIONES
def split_por_anios(df, years, symbol=None):
    df_split = df[df['Fecha'].dt.year.isin(years)]
    if symbol:
        df_split = df_split[df_split['Symbol'] == symbol]
    return df_split

# MAIN
if __name__ == "__main__":
```

```
# 1. Cargar dataset
df = pd.read_csv(DATA_PATH, parse_dates=["Fecha"], sep=';')
df = df.sort_values(['Symbol', 'Fecha']).reset_index(drop=True)

# 2. Determinar lista de features disponibles
cols_a_excluir = ['Fecha', 'Symbol', 'Open', 'Close', 'High', 'Low', 'AdjClose',
                  'Volume']
cols_a_excluir += [c for c in df.columns if c.startswith('EMA_')]
all_features = [c for c in df.columns if c not in cols_a_excluir and not
                c.startswith('TARGET_')]

# Lista para guardar resultados
filas_salida = []

# 3. Proceso para cada stock
for SYMBOL_TEST in SYMBOLS_TO_TEST:
    print(f"\n=== {SYMBOL_TEST} ===")
    print(f"Años GA train: {GA_TRAIN_YEARS}, años GA test: {GA_TEST_YEARS}")

    # Splits
    df_ga_train = split_por_anios(df, GA_TRAIN_YEARS)
    df_ga_test  = split_por_anios(df, GA_TEST_YEARS, SYMBOL_TEST)

    X_train_ga = df_ga_train[all_features]
    y_train_ga = df_ga_train[TARGET_COL]
    X_test_ga  = df_ga_test[all_features]
    y_test_ga  = df_ga_test[TARGET_COL]

    # GA Feature Selection
    ga = GA_Feature_Selection(
        X_train=X_train_ga,
        y_train=y_train_ga,
        X_test=X_test_ga,
        y_test=y_test_ga,
        feature_names=all_features,
        fitness_model='XGBRegressor',
        fitness_metric='rmse',
        n_pop=25,
        n_gen=20,
        elite=10,
        mut_prob=0.5,
        random_state=42,
        max_active=N_FEATURES,
        min_active=N_FEATURES,
        tournament_size=3
    )
    ga.fit(verbose=True)
    best_features = ga.get_best_features()

    print(f"Features seleccionadas ({len(best_features)}): {best_features}")

    # Guardar en lista
    filas_salida.append([SYMBOL_TEST, ", ".join(best_features)])

# 4. Guardar en CSV con pandas
df_out = pd.DataFrame(filas_salida, columns=['Stock', 'Features'])
df_out.to_csv(RESULTS_PATH, sep=';', index=False, encoding='utf-8')

print(f"\nOK -> guardado: {RESULTS_PATH}")
```

Fragmentos de código Python de fase ajuste de hipeparámetros óptimos mediante **RandomizedSearchCV** para cada activo financiero, utilizando las 25 características seleccionadas previamente:

```
import os
import numpy as np
import pandas as pd
from xgboost import XGBRegressor
from sklearn.model_selection import RandomizedSearchCV

# ---------------------- CONFIG ---------------------- #
DATASET_PATH  = "DATA/Dataset_All_Features_Transformado.csv"
FEATS_PATH    = "Resultados/Features_Seleccionadas_GA.csv"   # columnas:
Stock;Features (coma-separadas)
OUT_DIR       = "Resultados"
RES_PER_STOCK = os.path.join(OUT_DIR, "Hiperparametros_por_stock.csv")
RES_GLOBAL    = os.path.join(OUT_DIR, "Hiperparametros_globales_media.csv")

# Cartera de 25 símbolos
SYMBOLS_TO_TEST = [
    'NVDA','AAPL','AMZN','LRCX','SBUX','REGN','KLAC','BKNG','AMD','VRTX',
    'MAR','CDNS','CAT','INTU','GILD','MU','EBAY','AXP','AMAT','COST','MSFT',
    'ORCL','ADI','MS','NKE'
]

TARGET_COL   = "TARGET_TREND_ANG_15_5"
SCORING      = "neg_root_mean_squared_error"
N_ITER       = 100
N_JOBS       = -1
VERBOSE      = 1
RANDOM_STATE = 42

# Folds cronológicos
KFOLDS = [
    {"TRAIN": [2010, 2011, 2012, 2013, 2014, 2015],             "TEST": [2016, 2017]},
    {"TRAIN": [2010, 2011, 2012, 2013, 2014, 2015, 2016],       "TEST": [2017, 2018]},
    {"TRAIN": [2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017], "TEST": [2018, 2019]},
]

# Espacio de búsqueda
PARAM_DIST = {
    "n_estimators":  np.arange(50, 251, 50),
    "max_depth":     [3, 4, 5, 6],
    "learning_rate": [0.01, 0.03, 0.05, 0.1],
    "subsample":     [0.7, 0.85, 1.0],
    "colsample_bytree": [0.7, 0.85, 1.0],
}
INT_PARAMS = {"n_estimators", "max_depth"}

# ---------------------- MAIN ---------------------- #
def main():

    # Cargar dataset global (para poder hacer pooling)
    df = pd.read_csv(DATASET_PATH, sep=';', parse_dates=["Fecha"])
    df = df.sort_values(["Symbol", "Fecha"]).reset_index(drop=True)

    # Features seleccionadas por el GA por stock
    features_map = leer_features_por_stock(FEATS_PATH)
    filas_resultados = []
```

```
    for stock in SYMBOLS_TO_TEST:
        feats = features_map[stock]

        # X e y del DATASET con features seleccionadas por GA
        X = df[feats]
        y = df[TARGET_COL]

        # CV: train multi-stock / test solo stock objetivo
        cv_indices = construir_cv_indices_pooling(df, stock)

        # Definir el estimador base
        base_est = XGBRegressor(
            objective="reg:squarederror",
            random_state=RANDOM_STATE,
            n_jobs=N_JOBS,
            tree_method="hist",
        )

        rcv = RandomizedSearchCV(
            estimator=base_est,
            param_distributions=PARAM_DIST,
            n_iter=N_ITER,
            scoring=SCORING,
            cv=cv_indices,              # índices sobre df global
            n_jobs=N_JOBS,
            verbose=VERBOSE,
            random_state=RANDOM_STATE,
            refit=True,                 # refit en el mejor set
            return_train_score=False
        )

        print(f"\n>>> Buscando hiperparámetros para {stock} (feats={len(feats)})...")
        rcv.fit(X, y)
        best = rcv.best_params_
        best_score = rcv.best_score_  # será negativo (neg_root_mean_squared_error)
        print(f"{stock}: best {SCORING}={best_score:.6f} | params={best}")

        fila = {
            "Stock": stock,
            "best_score_neg_rmse": best_score,
            "best_rmse_pos": -best_score,  # por comodidad, RMSE positivo
        }
        for p in PARAM_DIST.keys():
            fila[p] = best.get(p, np.nan)
        filas_resultados.append(fila)

    # Resultados por stock
    df_params = pd.DataFrame(filas_resultados).sort_values("best_rmse_pos",
                            ascending=True).reset_index(drop=True)

    df_params.to_csv(RES_PER_STOCK, index=False, sep=';')
    print(f"\nGuardado resultados por stock: {RES_PER_STOCK}")

if __name__ == "__main__":
    main()
```

9.12 Anexo 12. Métricas de clasificación

En este trabajo se utilizan distintas métricas clásicas de clasificación, adaptadas al problema de predicción direccional en mercados financieros (por ejemplo, señales de **subida** o **bajada** de precios de activo en cuestión dentro de un horizonte de predicción). Todas ellas se definen a partir de los valores de la **matriz de confusión**, que distingue entre aciertos y errores del modelo:

TP (True Positive): el modelo predice *UP* y efectivamente ocurre *UP*.

TN (True Negative): el modelo predice *DOWN* y efectivamente ocurre *DOWN*.

FP (False Positive): el modelo predice *UP*, pero realmente ocurre *DOWN*.

FN (False Negative): el modelo predice *DOWN*, pero realmente ocurre *UP*.

Precisión (Precision)

La precisión mide el grado de acierto dentro de todas las veces que el modelo emitió una determinada señal. En otras palabras, indica qué proporción de las predicciones positivas de una clase fueron correctas.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- *Precision (UP):* de todas las veces que el modelo dijo *UP*, cuántas resultaron correctas.
- *Precision (DOWN):* de todas las veces que el modelo dijo *DOWN*, cuántas resultaron correctas.

Exhaustividad o Sensibilidad (Recall)

El recall mide la capacidad del modelo para identificar todos los casos reales de una clase. Refleja el porcentaje de aciertos sobre el total de casos verdaderos.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- *Recall (UP):* de todos los días que realmente fueron *UP*, cuántos detectó el modelo.
- *Recall (DOWN):* de todos los días que realmente fueron *DOWN*, cuántos detectó el modelo.

Medida F1 (F1-Score)

El F1 es la media armónica entre precisión y recall. Esta métrica es especialmente útil en mercados financieros porque penaliza modelos que tienden a favorecer una clase (por ejemplo, predecir *UP* constantemente) pero descuidan la otra.

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Exactitud global (Accuracy)

La exactitud mide el porcentaje de predicciones correctas sobre el total de observaciones.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

En este contexto refleja cuántos días acertó el modelo en la dirección del mercado, sin importar el balance entre subidas y bajadas.

F1 Macro Average

El F1 macro average es una métrica que resume el rendimiento global de un clasificador cuando existen dos o más clases. A diferencia de otras variantes como el **F1 ponderado (weighted avg)**, en el cálculo macro todas las clases reciben el mismo peso, independientemente del número de observaciones (soporte) que tengan.

Formalmente, para un problema binario con clases *DOWN* y *UP*:

$$F1_{\text{macro}} = \frac{F1_{\text{DOWN}} + F1_{\text{UP}}}{2}$$

donde cada $F1$ de la clase se obtiene como la media armónica de su precisión (Precision) y exhaustividad (Recall):

9.13 Anexo 13 Métricas de evaluación de simulación de compra – venta

El backtesting de la estrategia de trading se evaluó mediante un conjunto de métricas estandarizadas que permiten cuantificar tanto la rentabilidad como el riesgo del sistema. A continuación se describen cada una de ellas.

Rentabilidad anual (ROI anual, *Return on Investment*)

La rentabilidad anual mide el porcentaje de beneficio o pérdida obtenido en un año respecto al capital con el que se inició dicho período. Un ROI positivo indica que el capital se incrementó; uno negativo refleja una pérdida neta.

$$ROI_{anual} = \frac{\text{Capital final} - \text{Capital inicial}}{\text{Capital inicial}} \times 100$$

Tasa de aciertos (WinRate) %

La tasa de aciertos representa el porcentaje de operaciones que terminan con un beneficio respecto al total de operaciones realizadas. Es un indicador de frecuencia de éxito, pero no refleja por sí solo la rentabilidad, ya que no considera la magnitud de las ganancias o pérdidas.

$$WinRate = \frac{\text{Operaciones ganadoras}}{\text{Total de operaciones}} \times 100$$

Ganancia media / pérdida media:

$\overline{G}$, $\overline{P}$ Valores promedio de las operaciones positivas y negativas respectivamente.

Payoff Ratio Relación beneficio / pérdida

La relación beneficio/pérdida (Payoff Ratio) compara la ganancia media de las operaciones exitosas con la pérdida media de las operaciones fallidas. Un valor superior a 1 indica que, en promedio, las ganancias son mayores que las pérdidas.

$$\text{Payoff} = \frac{\overline{G}}{|\overline{P}|}$$

Donde : $\overline{G}$ es la ganancia media y $\overline{P}$ la pérdida media.

Compara la magnitud media de las ganancias frente a la de las pérdidas. Valores > 1 implican que las ganancias superan en media a las pérdidas.

Expectancy Beneficio esperado por operación:

La Expectancy refleja cuánto se espera ganar o perder de media en cada operación, combinando la frecuencia de aciertos (WinRate) con la magnitud de ganancias y pérdidas. Es una métrica clave porque resume en un solo valor la viabilidad económica del sistema.

$$Expectancy = \left(WinRate \times \overline{G}\right) - (1 - WinRate) \times \overline{P}$$

donde:

- WinRate = proporción de operaciones ganadoras (ej. 60 % = 0,6).
- $\overline{G}$ = ganancia media de las operaciones ganadoras.
- $\overline{P}$ = pérdida media de las operaciones perdedoras.

Si **Expectancy > 0**, la estrategia genera un beneficio esperado positivo por operación. Si **Expectancy < 0**, la estrategia destruye valor en promedio. El valor de expectancy está normalmente expresado en unidades monetarias.

Caída máxima de capital (*Maximum Drawdown*)

El drawdown máximo mide la mayor pérdida sufrida desde un máximo de capital hasta un mínimo posterior dentro de un período. Es un indicador del **riesgo máximo soportado** por la estrategia y permite evaluar la tolerancia de la cartera a pérdidas temporales.

$$DD_{max} = \max_{t}\left(\max_{s \le t} C\,apital(s) - Capital(t)\right)$$

Tasa compuesta de crecimiento anual (*CAGR, Compound Annual Growth Rate*)

El CAGR cuantifica la tasa de crecimiento anual que tendría el capital si se hubiera compuesto uniformemente a lo largo del período completo. Permite comparar estrategias con diferentes horizontes temporales y refleja el crecimiento sostenible a largo plazo.

$$CAGR = \left(\frac{Capital\ final}{Capital\ inicial}\right)^{\frac{1}{n}} - 1$$

donde n es el número de años del período.

9.14 Anexo 14 Repositorio de código fuente

Repositorio: https://github.com/MaxSheptyuk/TFM-Prediccion-Series-Financieras La Figura 60 muestra la raíz del proyecto y las carpetas principales.

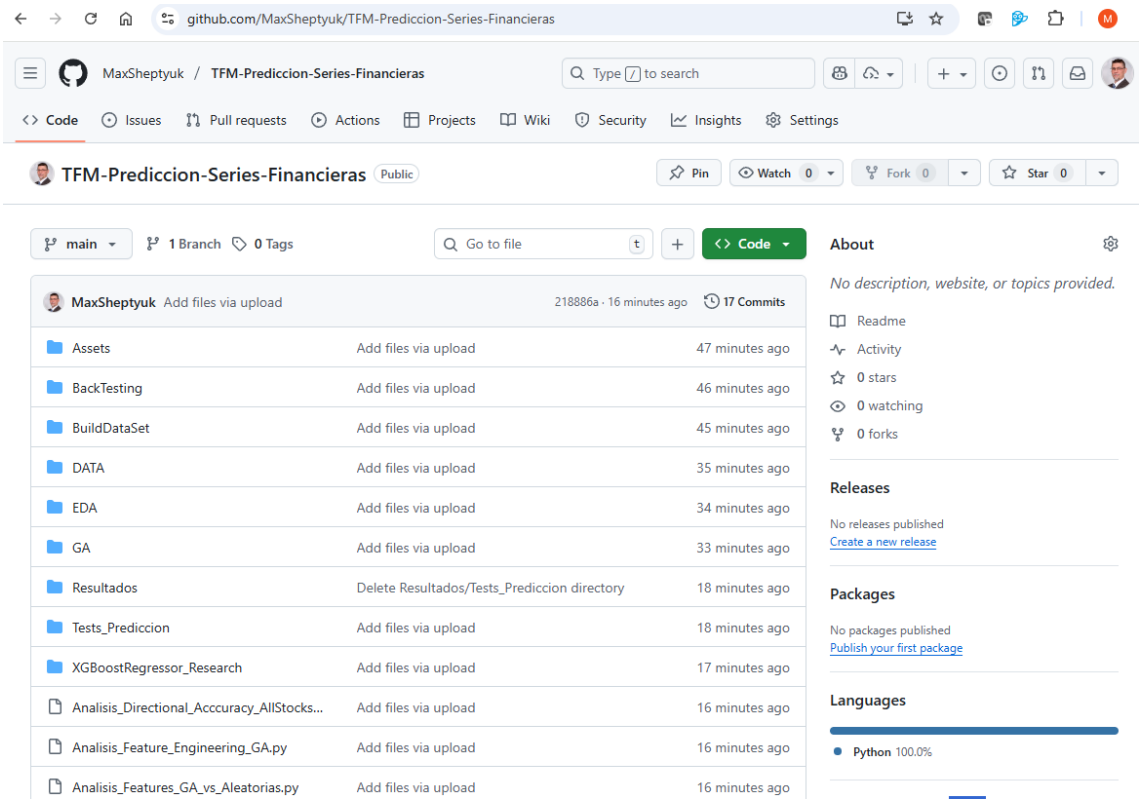


Figura 60. Repositorio de código fuente y descripción de ficheros clave.

Estructura del proyecto (resumen)

- BuildDataSet/ → Descarga histórica (yfinance), ETL e ingeniería de features.
- BackTesting/ → Utilidades comunes de backtesting (soporte a los pipelines).
- GA/ → Algoritmos Genéticos para selección de variables y análisis relacionados.
- Resultados/ → Salidas reproducibles (tablas, figuras, logs, métricas agregadas).
- Tests_Prediccion/ → Experimentos de predicción y validación (KFold, curvas, etc.).
- XGBoostRegressor_Research/ → Pruebas específicas con XGBoost.
- Assets/, EDA/, Tests_Prediccion/ → recursos, análisis exploratorio y tests.

Carpeta	Fichero	¿Qué hace? (resumen)
BuildDataSet	Download_Historical_Data.py	Descarga OHLCV (yfinance) y aplica ETL básico (imputación de nulos, control de % nulos, redondeos).
BuildDataSet	Build_Dataset_All_Features.py	Construye el **dataset final** por símbolo con el generador de indicadores; consolida CSV de trabajo.
Raíz	Feature_Generator.py	Genera **indicadores técnicos** masivos (EMA, RSI, MACD con *naming* de salidas, Stochastic, Williams %R, CCI, ADX/DI±, ATR, Bollinger, OBV, etc.).
Raíz	Adaptive_Feature_Normalizer.py	**Normalización adaptativa** según *skewness* (Power/Standard/MinMax), evitando *data leakage* (fit solo en train).
BuildDataSet	Comprobar_Coherencia_Dataset.py	Chequeos de calidad del dataset (nulos, duplicados, rangos, consistencia de columnas).
BuildDataSet	Build_Dataset_Auxiliar_Backtesting.py	Utilidades auxiliares para backtesting (p. ej., ATR simple para stops/trailing).
GA	GA_Feature_Selection.py	**Algoritmo Genético** para selección de *features*: población, fitness, torneo, *crossover*, mutación; compatible con ElasticNet/SVM/MLP/XGBoost.
BackTesting	Backtester.py	**Motor base** de backtesting: ejecución de señales, gestión de posiciones, métricas y logs de cartera.
BackTesting	BacktesterSignalExit.py	Backtesting con **salida gobernada por señal** (no solo SL/TP o ventana fija).
BackTesting	BacktesterCalidadML.py	Backtesting de **calidad de señal** (ventanas/umbrales, Spearman robusto, export ordenado).

Raíz (pipelines)	Pipeline_BackTesting_GAXGBoost.py	Pipeline principal: **XGBoost + GA** (señal, entrenamiento y backtesting).
Raíz (pipelines)	Pipeline_BackTesting_GAXGBoost_Calidad.py	Variante para evaluar **calidad de señal** con GA+XGB.
Raíz (pipelines)	Pipeline_BackTesting_XGB_AllFeatures.py	**Baseline** con XGBoost usando **todas** las *features* (sin GA).
Raíz (pipelines)	Pipeline_BackTesting_MLP_OHLC.py	Pipeline **MLP** (scikit-learn) sobre OHLC (+ *features* definidas).
Raíz (pipelines)	Pipeline_BackTesting_ARIMA.py	**Benchmark ARIMA** (modelo clásico de series temporales).
Raíz (pipelines)	Pipeline_Ajuste_Hiperparametros.py	**Tuning** con validación/cortes temporales; compara frente a baseline.

Tabla 21 Descripción breve de ficheros de repositorio y su función

9.15 Anexo 15 Python y sus librerías requeridas

=== Entorno Python ===

Python: 3.11.0 (CPython) Plataforma: Windows-10-10.0.26100-SP0

Paquete	Versión
pandas	2.2.3
numpy	1.26.4
scipy	1.15.2
scikit-learn	1.6.1
matplotlib	3.10.1
tqdm	4.67.1
python-dateutil	2.9.0.post0
xgboost	3.0.2
deap	1.4.3

Paquete	Versión
torch	2.5.1+cu121
torchvision	0.20.1+cu121
torchaudio	2.5.1+cu121
tensorflow	2.19.0
keras	3.9.2
yfinance	0.2.64
statsmodels	0.14.5
stock-indicators	1.3.5

Tabla 22. Python y las librerías requeridas.

[PÁGINA INTENCIONADAMENTE EN BLANCO]