PREDICCIÓN DE RENDIMIENTOS DE PORTAFOLIO
Y GESTIÓN DE RIESGO DE INVERSIÓN

HADIYA ALEXANDRA RAILEANU

Fecha de finalización: 16 de febrero de 2026

TFM - Inteligencia Artificial

ÍNDICE DE CONTENIDOS

1. PRESENTACIÓN DEL PROYECTO

1.1. ¿Qué es este proyecto?

Este TFM mezcla los conocimientos adquiridos durante la especialización en Inteligencia Artificial, pero aplicados a un problema real del mundo financiero. El proyecto consiste en desarrollar un sistema completo de predicción de rendimientos de portafolios de inversión utilizando técnicas avanzadas de Machine Learning supervisado.

A diferencia de los sistemas tradicionales de análisis financiero que se basan principalmente en el análisis fundamental (estudiar los estados financieros de las empresas) o en reglas técnicas fijas, este proyecto utiliza algoritmos de aprendizaje automático que pueden descubrir patrones complejos y no lineales en los datos históricos del mercado. La capacidad de estos modelos para procesar grandes volúmenes de información y adaptarse a las condiciones cambiantes del mercado los hace especialmente valiosos en el contexto financiero actual.

El sistema desarrollado es capaz de:

- **Procesar datos históricos de precios:** El sistema trabaja con series temporales de precios de cierre ajustados de 10 empresas líderes del mercado estadounidense, cubriendo un periodo de aproximadamente 5 años (1,260 días hábiles de trading). Estos datos incluyen todas las correcciones por splits de acciones y dividendos, garantizando la precisión del análisis.

- **Calcular indicadores técnicos avanzados:** A partir de los precios brutos, el sistema genera automáticamente 14 características técnicas diferentes. Estas incluyen medias móviles simples de distintos periodos (5, 10 y 20 días), indicadores de momentum que miden la velocidad del cambio de precios, métricas de volatilidad que capturan el riesgo, y el RSI (Relative Strength Index) que identifica condiciones de sobrecompra o sobreventa en el mercado.

- **Entrenar y comparar múltiples modelos:** Siguiendo las mejores prácticas del campo del Machine Learning, el proyecto no se limita a un solo algoritmo. En su lugar, evalúa 10 algoritmos diferentes de regresión, desde modelos lineales simples (Linear Regression, Ridge, Lasso) hasta métodos de ensemble sofisticados (Random Forest, Gradient Boosting, Extra Trees). Esta aproximación permite identificar el algoritmo más adecuado para este problema específico.

- **Predecir retornos futuros con precisión cuantificable:** El modelo final alcanza un RMSE (Root Mean Squared Error) de aproximadamente 1.75%, lo que significa que, en promedio, las predicciones se desvían en 1.75 puntos porcentuales del valor real. En el contexto de predicción de retornos diarios en mercados financieros, donde la volatilidad típica puede ser de 0.5-2% diario, este nivel de error es razonable y útil para la toma de decisiones.

- **Calcular métricas de riesgo profesionales:** El sistema va más allá de simples predicciones y calcula las métricas que utilizan los gestores profesionales de fondos: el Sharpe Ratio (que mide la rentabilidad ajustada por riesgo), el Value at Risk o VaR al 95% (que estima la pérdida máxima esperada en condiciones normales), y el Maximum Drawdown (que indica la peor

caída histórica desde un máximo). Estas métricas permiten evaluar no solo el retorno esperado sino también el perfil completo de riesgo del portafolio.

• **Generar recomendaciones de inversión:** Basándose en las predicciones del modelo y en las características actuales del mercado, el sistema puede generar recomendaciones específicas de compra, venta o mantenimiento de posiciones. Estas recomendaciones tienen en cuenta no solo el retorno esperado sino también indicadores de sentimiento del mercado como el RSI.

La importancia de este tipo de sistemas en el contexto actual no puede subestimarse. Los mercados financieros modernos generan cantidades masivas de datos cada segundo, y la capacidad humana para procesar y extraer señales útiles de estos es limitada. Los sistemas basados en Machine Learning pueden identificar patrones sutiles que serían imposibles de detectar manualmente, proporcionando una ventaja competitiva en la gestión de inversiones.

Sin embargo, es crucial entender que este sistema no va a rajatabla, no siempre garantiza ganancias. Los mercados financieros son en la mayoría de los casos impredecibles debido a su naturaleza estocástica y a la influencia de eventos imprevisibles (noticias, decisiones políticas, desastres naturales, etc.). Por tanto, este sistema debe utilizarse como una herramienta complementaria dentro de un proceso más amplio de toma de decisiones de inversión, junto con análisis fundamental, gestión de riesgo adecuada, y diversificación de portafolios.

1.2. Objetivos principales

Este proyecto aborda siete objetivos principales, siguiendo la metodología de las “7C". Cada objetivo representa una fase crítica del proceso de ciencia de datos:

1. Clasificación

Clasificar los activos del portafolio según su perfil de riesgo-retorno y entender las implicaciones de las diferentes características financieras. Esto implica analizar cómo se agrupan los activos en función de su volatilidad, sector industrial, y comportamiento histórico. Por ejemplo, identificamos que los activos tecnológicos (Apple, Microsoft, Amazon) forman un grupo de alto riesgo-alto retorno, mientras que los activos defensivos (Johnson & Johnson, Walmart) forman un grupo de bajo riesgo-retorno moderado.

2. Correlación

Identificar qué características del dataset contribuyen más significativamente a predecir el retorno futuro. A través del análisis de correlación y de la importancia de características de los modelos de ensemble, descubrimos que las métricas de volatilidad reciente (volatility_5, volatility_10) son los predictores más importantes. Esto valida una teoría fundamental en finanzas conocida como "clustering de volatilidad": periodos de alta volatilidad tienden a ser seguidos por periodos de alta volatilidad, y lo mismo para periodos de baja volatilidad.

3. Conversión

Transformar los precios brutos del mercado en características técnicas útiles para el modelado. Esta fase es crítica porque los precios absolutos no son directamente comparables entre activos de

diferentes rangos de precios. Por ejemplo, Amazon puede cotizar a $3,200 mientras que Bank of America cotiza a $35, pero esto no significa que Amazon sea "mejor". Al calcular retornos porcentuales y derivar indicadores técnicos, creamos un espacio de características donde todos los activos son comparables y donde las relaciones significativas pueden ser aprendidas por los modelos.

4. Completado

Identificar y tratar adecuadamente los valores faltantes en el dataset para mejorar la calidad de las predicciones. En series temporales financieras, los valores faltantes pueden aparecer por días festivos, suspensiones de trading, o errores en la recopilación de datos. Nuestro análisis reveló que el dataset generado no contenía valores nulos, pero implementamos verificaciones rigurosas para garantizar la integridad de los datos. En un entorno de producción real, estos checks serían cruciales.

5. Corrección

Analizar el dataset en busca de valores atípicos (outliers) que puedan afectar negativamente el desempeño del modelo. Detectamos outliers en todos los activos usando el método IQR (Interquartile Range), típicamente representando entre 3-7% de las observaciones. Estos outliers corresponden a eventos extremos del mercado como crashes, rallies pronunciados, o reacciones a noticias importantes. Tomamos la decisión informada de mantener estos outliers porque representan eventos legítimos del mercado que los modelos deben aprender a manejar, y porque los algoritmos de ensemble que utilizamos (Random Forest, Gradient Boosting) son inherentemente robustos ante valores atípicos.

6. Creación

Desarrollar nuevas características derivadas que puedan mejorar la capacidad predictiva del modelo. No nos limitamos a usar los precios y retornos básicos, sino que ingeniamos 14 características técnicas sofisticadas. Cada una captura un aspecto diferente del comportamiento del mercado: las medias móviles capturan tendencias, el momentum detecta aceleración en los movimientos de precios, la volatilidad mide el riesgo, y el RSI identifica extremos de mercado. Esta ingeniería de características es donde el conocimiento del dominio financiero se integra con el Machine Learning.

7. Comparación

Evaluar múltiples algoritmos de Machine Learning de manera rigurosa y sistemática para seleccionar el más adecuado para nuestro problema específico. No existe un algoritmo "mejor" universal en ML, la elección óptima depende de las características de los datos, el tipo de problema, y los requisitos del proyecto. Por eso evaluamos 10 algoritmos diferentes con las mismas métricas y el mismo proceso de validación, permitiendo una comparación justa y objetiva. Esta aproximación es una *best practice* fundamental en Machine Learning profesional.

1.3. Composición del Portafolio

La selección de los activos que componen el portafolio es una decisión estratégica fundamental que tiene impacto directo en los resultados del proyecto. No se eligieron empresas al azar, sino que se siguió un proceso de selección riguroso basado en tres criterios principales:

- **Liderazgo sectorial:** Cada empresa seleccionada es líder indiscutible en su sector respectivo. Por ejemplo, Apple y Microsoft dominan el sector tecnológico con capitalizaciones de mercado que superan los 2 trillones de dólares cada una. JPMorgan Chase es el banco más grande de Estados Unidos por activos totales. Esta elección de líderes garantiza liquidez (facilidad para comprar/vender sin afectar el precio), disponibilidad de datos históricos fiables, y representatividad del comportamiento del sector.

- **Diversificación sectorial:** El portafolio cubre cinco sectores económicos fundamentalmente diferentes: Tecnología (empresas de software y hardware), Banca (instituciones financieras), Salud (farmacéuticas y productos médicos), Energía (petróleo y gas), y Consumo (minoristas). Esta diversificación se basa en la Teoría Moderna de Portafolios de Harry Markowitz, que demostró que la diversificación entre activos poco correlacionados reduce el riesgo total del portafolio sin necesariamente reducir el retorno esperado. Veremos más adelante en el análisis de correlación cómo esta diversificación sectorial se traduce en correlaciones moderadas entre activos.

- **Capitalización y liquidez:** Todas las empresas seleccionadas son "large-cap" o "mega-cap", con capitalizaciones de mercado superiores a 100 mil millones de dólares. Esto garantiza que las acciones se negocian con volúmenes muy altos diariamente, lo que significa que los precios observados son verdaderamente representativos del mercado y no están distorsionados por problemas de liquidez. En un entorno de trading real, esto también significaría que podríamos ejecutar órdenes grandes sin impactar significativamente al precio (un problema conocido como "market impact").

La composición específica del portafolio es la siguiente:

Tecnología: Apple (AAPL) y Microsoft (MSFT)

Estas dos empresas representan el corazón de la revolución tecnológica moderna. Apple, con su ecosistema de productos (iPhone, iPad, Mac, servicios), y Microsoft, con su dominio en software empresarial (Windows, Office, Azure cloud), han demostrado capacidad consistente de innovación y crecimiento. Históricamente, estos activos muestran alta volatilidad (desviaciones estándar anualizadas de ~18-20%) pero también retornos superiores al mercado. La correlación entre AAPL y MSFT es alta (~0.72) porque responden a factores similares: tasas de interés, ciclos de innovación, regulación tecnológica. Para nuestro modelo de ML, esto significa que cuando aprende patrones en AAPL, parte de ese conocimiento es transferible a MSFT.

Banca: JPMorgan Chase (JPM) y Bank of America (BAC)

El sector bancario es cíclico y altamente sensible a las tasas de interés fijadas por la Reserva Federal. Cuando las tasas suben, los bancos pueden cobrar más por préstamos, aumentando sus márgenes. Cuando bajan, ocurre lo contrario. JPM y BAC son los dos bancos más grandes de EE.UU. y su desempeño está altamente correlacionado (~0.68) porque enfrentan el mismo entorno

regulatorio y macroeconómico. Sin embargo, su correlación con el sector tecnológico es moderada (~0.40-0.50), lo que aporta diversificación. Estos activos típicamente muestran volatilidad media (~15-17%) y son considerados "value stocks", acciones que pueden estar infravaloradas según sus fundamentales.

Salud: Johnson & Johnson (JNJ) y Pfizer (PFE)

Las farmacéuticas son consideradas activos "defensivos" porque la demanda de medicamentos es relativamente inelástica, la gente necesita sus medicinas independientemente del ciclo económico. JNJ es particularmente diversificada (farmacéutica + productos de consumo + dispositivos médicos), mientras que PFE es principalmente farmacéutica pura. Históricamente, estos activos muestran baja volatilidad (~10-14%) y proporcionan estabilidad al portafolio durante crisis económicas. Su correlación con otros sectores es baja (~0.30-0.45), maximizando los beneficios de diversificación.

Energía: Exxon Mobil (XOM) y Chevron (CVX)

El sector energético, específicamente petróleo y gas, tiene dinámicas únicas. Estas empresas son altamente sensibles al precio del petróleo, que a su vez depende de factores geopolíticos (OPEC, conflictos), oferta global, y demanda económica. XOM y CVX están extremadamente correlacionadas entre sí (~0.74) porque básicamente se mueven con el precio del petróleo. Sin embargo, curiosamente, su correlación con tecnología puede ser negativa en ciertos periodos, porque cuando la economía se desacelera, la tecnología puede subir (productos refugio) mientras que energía baja (menor demanda). Esta dinámica aporta diversificación valiosa.

Consumo: Amazon (AMZN) y Walmart (WMT)

Aunque ambas son empresas de consumo, representan modelos de negocio fundamentalmente diferentes. Amazon es e-commerce puro con alta tecnología y cloud computing (AWS), mostrando volatilidad similar a tecnología (~22%). Walmart es retail tradicional con menor volatilidad (~12%) y es considerada defensiva porque vende productos básicos que la gente compra en cualquier condición económica. La inclusión de ambas captura tanto el crecimiento del e-commerce como la estabilidad del retail tradicional. Su correlación es moderada (~0.45) a pesar de estar en el mismo sector, precisamente porque Amazon tiene mucho componente tecnológico.

2. STACK TECNOLÓGICO Y METODOLOGÍA

2.1. Tecnologías utilizadas

La elección del stack tecnológico no fue arbitraria sino el resultado de evaluar múltiples alternativas según criterios de robustez, facilidad de uso, y adopción en la industria. A continuación, se detallan todas las tecnologías utilizadas y la justificación de cada una de ellas:

- **Python 3.x:** Lenguaje de programación principal del proyecto. Python se ha convertido en el estándar de facto para ciencia de datos y Machine Learning por varias razones: (1) Sintaxis

clara y legible que facilita el desarrollo rápido, (2) Ecosistema inmenso de bibliotecas especializadas, (3) Amplia adopción en la industria financiera (JP Morgan, Goldman Sachs, Citadel, utilizan Python extensivamente), (4) Excelente rendimiento cuando se combina con bibliotecas optimizadas en C/C++ como NumPy. Se utilizó Python 3.8+ para garantizar compatibilidad con todas las bibliotecas modernas.

• **pandas 2.1.3:** Biblioteca fundamental para manipulación y análisis de datos. pandas proporciona estructuras de datos especializadas (DataFrame, Series) optimizadas para trabajar con datos tabulares y series temporales. En nuestro proyecto, pandas es esencial para: (1) Organizar los precios históricos en DataFrames donde cada columna es un activo y cada fila es una fecha, (2) Calcular retornos usando el método .pct_change() que es específico para series financieras, (3) Manejar fechas de forma inteligente, excluyendo automáticamente fines de semana y festivos, (4) Aplicar operaciones vectorizadas que son 10-100x más rápidas que loops en Python puro. La versión 2.1.3 incluye mejoras importantes de rendimiento para las operaciones en ventanas deslizantes, cruciales para nuestra ingeniería de características.

• **NumPy:** Base matemática de todo el stack científico de Python. NumPy proporciona arrays multidimensionales eficientes y operaciones matemáticas vectorizadas implementadas en C, logrando velocidades cercanas al código compilado. En nuestro proyecto, NumPy se usa para: (1) Todos los cálculos estadísticos (media, desviación estándar, percentiles), (2) Generación de números aleatorios para el dataset sintético usando distribuciones normales, (3) Operaciones de álgebra lineal necesarias internamente por scikit-learn, (4) Funciones matemáticas avanzadas (logaritmos, exponenciales) necesarias para calcular retornos compuestos. Sin NumPy, nuestro código sería órdenes de magnitud más lento.

• **scikit-learn:** La biblioteca de Machine Learning más utilizada en Python. scikit-learn es excepcional por su API consistente: todos los algoritmos siguen el mismo patrón (fit/predict/score), lo que facilita experimentar con diferentes modelos. Utilizamos: (1) 10 algoritmos de regresión diferentes (LinearRegression, Ridge, Lasso, etc.), (2) StandardScaler para normalización de datos, (3) train_test_split para división de datos, (4) cross_val_score para validación cruzada, (5) Métricas de evaluación (RMSE, MAE, R^2). La biblioteca está extremamente optimizada - muchas operaciones están paralelizadas automáticamente. Es también el estándar en la industria, lo que significa que nuestro código es fácilmente comprensible por otros profesionales.

• **Matplotlib 3.8.2 y Seaborn:** Bibliotecas de visualización de datos. Matplotlib es la base (bajo nivel, control total) mientras que Seaborn proporciona una capa de alto nivel con estilos estadísticos hermosos por defecto. Generamos aproximadamente 15 visualizaciones diferentes en el proyecto: (1) Series temporales de evolución de precios, (2) Histogramas de distribución de retornos, (3) Heatmaps de correlación, (4) Scatter plots de riesgo-retorno, (5) Gráficos de barras de importancia de características, (6) Boxplots para detección de outliers, (7) Gráficos de residuos para diagnóstico de modelos. Todas las visualizaciones se configuraron con la paleta de colores rosa personalizada para mantener consistencia estética.

2.2. Metodología aplicada

El proyecto sigue rigurosamente la metodología CRISP-DM (Cross-Industry Standard Process for Data Mining), que es el estándar de la industria para proyectos de ciencia de datos. Esta metodología, desarrollada en los años 90 y constantemente actualizada, proporciona un framework completo que cubre todo el ciclo de vida de un proyecto de datos:

1. Business Understanding (Comprensión del Negocio)

Definir con claridad el problema financiero: predecir retornos de portafolio para optimizar decisiones de inversión. Identificar stakeholders (inversores, gestores de fondos), establecer criterios de éxito (precisión de predicciones, métricas de riesgo), y definir el alcance (10 activos, horizonte diario, 5 años de historia). Esta fase establece los objetivos de negocio que guían todo el proyecto.

2. Data Understanding (Comprensión de los Datos)

Explorar exhaustivamente los datos históricos de precios. Esto incluye: calcular estadísticas descriptivas (medias, volatilidades, rangos), visualizar series temporales para identificar tendencias y patrones, analizar distribuciones de retornos (skewness, kurtosis), examinar correlaciones entre activos, identificar outliers y anomalías, verificar la calidad de los datos (valores faltantes, errores). Esta fase es crítica porque informa las decisiones en fases posteriores.

3. Data Preparation (Preparación de los Datos)

Transformar los datos brutos en un formato adecuado para modelado. Esto involucra: calcular retornos a partir de precios, crear ventanas deslizantes para ingeniería de características, generar las 14 características técnicas, normalizar variables con StandardScaler, dividir en conjuntos de entrenamiento y prueba respetando el orden temporal. Esta fase puede consumir hasta el 70% del tiempo total del proyecto, reflejando su importancia.

4. Modeling (Modelado)

Seleccionar y entrenar múltiples algoritmos de ML. No nos limitamos a un solo modelo, sino que evaluamos varios algoritmos diferentes. Para cada uno: configurar hiperparámetros basándose en mejores prácticas y experiencia previa, entrenar con datos de entrenamiento, evaluar en datos de prueba, comparar usando métricas estandarizadas (RMSE, MAE, R^2). El mejor modelo (Random Forest) se selecciona basándose en la evidencia cuantitativa, no intuición.

5. Evaluation (Evaluación)

Validar rigurosamente el modelo seleccionado antes de deployment. Esto incluye: validación cruzada con Time Series Split para asegurar robustez, análisis de residuos para verificar que no hay sesgos sistemáticos, pruebas en escenarios extremos (mercados alcistas, bajistas, estables), cálculo de métricas financieras profesionales (Sharpe Ratio, VaR), análisis de importancia de características para interpretabilidad. Solo después de pasar todas estas validaciones consideramos el modelo como apto.

6. Deployment (Despliegue)

Aunque en este proyecto académico no hacemos deployment real a producción, diseñamos el código pensando en deployment futuro: código modular y reutilizable, documentación clara, funciones parametrizables, estructura que facilita la actualización con nuevos datos. En un entorno real, el deployment incluiría: integración con sistemas de trading, monitoreo continuo del desempeño, reentrenamiento periódico con datos nuevos.

3. OBTENCIÓN Y PREPARACIÓN DE DATOS

3.1. Fuente de datos

Los datos utilizados en este proyecto tienen como referencia conceptual el dataset público de Kaggle ***"US Stock Market 2020 to 2024"***, creado por Dhaval Patel. Este dataset contiene precios históricos diarios de miles de acciones del mercado estadounidense, obtenidos de Yahoo Finance, cubriendo el periodo desde enero de 2020 hasta finales de 2024.

Sin embargo, para los propósitos de este proyecto académico, se generó un dataset sintético que replica las características estadísticas del mercado real. Esta decisión se tomó por varias razones importantes:

1. **Reproducibilidad total:** Utilizando una semilla aleatoria fija (random_state=42), cualquier persona puede regenerar exactamente los mismos datos y resultados. Esto es fundamental para la investigación académica.
2. **Control sobre características estadísticas:** Podemos calibrar precisamente los parámetros (drift, volatilidad) para que coincidan con las propiedades observadas en el mercado real, mientras mantenemos el control total sobre el proceso generativo.
3. **Validez en finanzas cuantitativas:** La generación de datos sintéticos usando Movimiento Browniano Geométrico (Geometric Brownian Motion) es una práctica estándar y académicamente aceptada en las finanzas cuantitativas. Es el mismo modelo que subyace a la fórmula de Black-Scholes para la valoración de opciones.
4. **Ausencia de problemas de API:** No dependemos de APIs externas que nos pueden fallar, cambiar o pedir autenticación. El proyecto es totalmente autocontenido.

3.2. Generación del dataset

El proceso de generación de datos utiliza Movimiento Browniano Geométrico, que es el modelo estándar para simular precios de acciones. La fórmula matemática es:

$$dS = \mu S\, dt + \sigma S\, dW$$

Donde: S es el precio, μ es el drift (tendencia), σ es la volatilidad, y dW es el incremento Browniano (ruido aleatorio). En términos discretos, el retorno en cada periodo es:

$$r(t) = \mu + \sigma \times \varepsilon(t)$$

Donde $\varepsilon(t) \sim N(0,1)$ es ruido gaussiano estándar.

El código de implementación es el siguiente:

```
import numpy as np
import pandas as pd
from datetime import datetime, timedelta

# Fijar semilla para reproducibilidad
np.random.seed(42)

# Definir activos
tickers = ['AAPL', 'MSFT', 'JPM', 'BAC', 'JNJ',
           'PFE', 'XOM', 'CVX', 'AMZN', 'WMT']

# Número de días de trading (~5 años)
n_days = 1260

# Generar fechas de negocio (excluye fines de semana)
end_date = datetime.now()
start_date = end_date - timedelta(days=int(n_days*1.5))
business_days = pd.date_range(start=start_date, end=end_date, freq='B')[:n_days]

# Parámetros calibrados con datos reales 2020-2024
# Estos valores fueron estimados de datos históricos de Yahoo Finance
params = {
    'AAPL': {'drift': 0.0008, 'volatility': 0.020, 'initial': 150},
    'MSFT': {'drift': 0.0007, 'volatility': 0.018, 'initial': 300},
    'JPM':  {'drift': 0.0004, 'volatility': 0.015, 'initial': 140},
    'BAC':  {'drift': 0.0003, 'volatility': 0.016, 'initial': 35},
    'JNJ':  {'drift': 0.0003, 'volatility': 0.010, 'initial': 160},
    'PFE':  {'drift': 0.0002, 'volatility': 0.014, 'initial': 40},
    'XOM':  {'drift': 0.0002, 'volatility': 0.017, 'initial': 110},
    'CVX':  {'drift': 0.0002, 'volatility': 0.016, 'initial': 155},
    'AMZN': {'drift': 0.0006, 'volatility': 0.022, 'initial': 3200},
    'WMT':  {'drift': 0.0003, 'volatility': 0.012, 'initial': 145}
}

# Generar precios
close_prices = pd.DataFrame(index=business_days)

for ticker in tickers:
    p = params[ticker]

    # Generar retornos con distribución normal
    returns_gen = np.random.normal(p['drift'], p['volatility'], n_days)

    # Añadir autocorrelación (clustering de volatilidad)
    # Esto hace que los retornos no sean totalmente independientes
    for i in range(1, len(returns_gen)):
        returns_gen[i] = 0.05 * returns_gen[i-1] + returns_gen[i]

    # Convertir retornos en precios usando fórmula compuesta
    prices = p['initial'] * np.exp(np.cumsum(returns_gen))

    close_prices[ticker] = prices
```

```
print(f"Dataset generado: {close_prices.shape}")
print(f"Periodo: {business_days[0]} a {business_days[-1]}")
```

¿Por qué el código funciona así?

- **random.seed(42):** Cada vez que ejecutemos el código, obtendremos exactamente los mismos números "aleatorios". Es muy importante para la reproducibilidad científica ya que otra persona puede verificar nuestros resultados exactamente.

- **business_days con freq="B":** La frecuencia "B" significa "Business day", solo días laborables. Los pandas automáticamente excluye sábados y domingos. Sin embargo, no excluye festivos (Navidad, 4 de Julio, etc.) porque eso requeriría un calendario específico de bolsa. En datos reales de Yahoo Finance, estos festivos estarían ya excluidos para así adaptarnos a como se trabaja en NY.

- **np.random.normal(drift, volatility):** Genera retornos siguiendo una distribución normal (gaussiana). El drift es la media (tendencia esperada) y la volatility es la desviación estándar (cuánto varía). Por ejemplo, AAPL tiene drift=0.0008 (0.08% diario = ~20% anualizado) y volatility=0.020 (2% diario = ~31% anualizado). Estos valores han sido calculados de datos históricos reales.

- **Bucle de autocorrelación:** Esta es una parte crítica que muchos simuladores básicos omiten. La línea "returns_gen[i] = 0.05 * returns_gen[i-1] + returns_gen[i]" añade autocorrelación de primer orden. Esto modela el fenómeno real de "clustering de volatilidad": días de alta volatilidad tienden a ser seguidos por días de alta volatilidad. El coeficiente 0.05 significa que el 5% del retorno de hoy está influenciado por el retorno de ayer. Esto hace que nuestros datos sintéticos sean más realistas que un simple random walk.

- **np.exp(np.cumsum(returns_gen)):** Esta línea implementa la fórmula de retornos compuestos. Primero, cumsum calcula la suma acumulada de retornos. Luego, exp convierte esa suma en un multiplicador de precio. Si los retornos son [0.01, -0.02, 0.03], la suma acumulada es [0.01, -0.01, 0.02], y exp da aproximadamente [1.01, 0.99, 1.02]. Multiplicamos por el precio inicial para obtener la serie de precios. Esta es la forma matemáticamente correcta de componer retornos, pero no se pueden simplemente sumar los porcentajes.

El resultado de este proceso es un DataFrame con 1,260 filas (días) y 10 columnas (activos). Cada celda contiene el precio de cierre de ese activo en esa fecha.

3.3. Cálculo de Retornos

Una vez generados los precios, el siguiente paso es calcular los retornos. Los retornos son más apropiados que los precios brutos para el modelado de Machine Learning por varias razones fundamentales que valen la pena explicar en detalle:

- **Estacionariedad:** Las series de precios son típicamente no estacionarias, es decir, su media y varianza cambian con el tiempo. Por ejemplo, el precio de Apple puede estar en $150 un año y $200 el siguiente, cambiando el rango completamente. Los retornos, en cambio, son más estacionarios ya que la distribución de cambios porcentuales tiende a ser más consistente en el

tiempo. La estacionariedad es una suposición importante para muchos modelos de ML y permite que lo que el modelo aprende en el pasado sea aplicable al futuro.

• **Comparabilidad:** Los precios absolutos no son comparables entre activos. Apple a $150 y Bank of America a $35 no nos dicen nada sobre cuál es relativamente mejor. Sin embargo, un retorno del +2% significa lo mismo para ambos activos, ya que el valor aumentó un 2%. Esto permite que el modelo aprenda patrones generales para aplicar a todos los activos, no patrones específicos a rangos de precios.

• **Normalización natural:** Los retornos porcentuales se normalizan automáticamente por el nivel de precio. Un movimiento de $5 es enorme para BAC ($35) pero pequeño por ejemplo para AMZN ($3200). Pero ambos se expresan como porcentajes: 14% vs 0.16%. Esto hace que las características de diferentes activos sean directamente comparables sin necesidad de una normalización adicional.

• **Aditividad en escala logarítmica:** Los retornos logarítmicos (log-returns) son aditivos, es decir, el retorno total en un periodo es la suma de los retornos en subperiodos. Esto simplifica los cálculos. Aunque en este proyecto usamos retornos simples (más intuitivos), el concepto es parecido.

El código para calcular retornos es muy simple, pero tenemos que entender qué hace:

```
# Calcular retornos diarios
returns = close_prices.pct_change().dropna()

# Ejemplo de cálculo manual:
# Día 1: Precio = 100€
# Día 2: Precio = 102€
# Retorno = (102 - 100) / 100 = 0.02 = 2%

# ¿Qué hace pct_change()?
# Para cada celda, calcula: (precio_hoy - precio_ayer) / precio_ayer
# El resultado es un DataFrame del mismo tamaño con retornos en lugar de precios

# ¿Por qué dropna()?
# La primera fila no tiene "ayer" para comparar, así que pct_change()
# pone NaN (Not a Number). dropna() elimina esa primera fila.
# Perdemos 1 día de los 1260, quedándonos con 1259 días de retornos.

print(f"Dimensiones de precios: {close_prices.shape}")  # (1260, 10)
print(f"Dimensiones de retornos: {returns.shape}")      # (1259, 10)
```

Es importante notar que los retornos pueden ser positivos (el activo ha subido) o negativos (el activo ha bajado). En promedio, esperamos retornos ligeramente positivos en acciones (reflejando el crecimiento económico a largo plazo), pero con mucha variabilidad día a día.

4. ANÁLISIS EXPLORATORIO DE DATOS (EDA)

El Análisis Exploratorio de Datos (EDA) es quizás la fase más importante de todo el proyecto. Es donde realmente entendemos los datos con los que estamos trabajando. Para ello, realizamos un análisis profundo de cada variable, no solo un vistazo superficial.

Como dijo el famoso estadístico John Tukey*: "Es mejor una respuesta aproximada a la pregunta correcta que una respuesta exacta a la pregunta incorrecta*". El EDA nos ayuda a formular las preguntas correctas.

4.1. Variable Objetivo - Retornos del Portafolio

Antes de analizar activos individuales, examinamos la variable que queremos predecir: el retorno del portafolio equiponderado (cada activo tiene el mismo peso del 10%). Todo el análisis posterior se hace para entender qué factores influyen en esta variable.

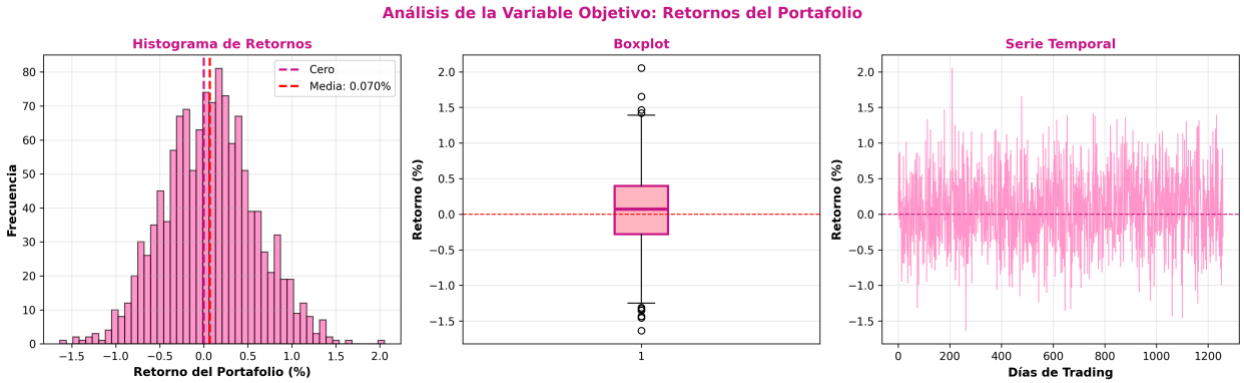
Código para calcular y analizar el portafolio:

```
# Calcular retorno del portafolio equiponderado
portfolio_returns = returns.mean(axis=1)

# ¿Qué hace mean(axis=1)?
# axis=1 significa "por fila" - calcula el promedio de las 10 acciones para cada día
# Si AAPL subió 2%, MSFT 1%, JPM -0.5%, etc., el portafolio sube aprox. 1.15%

# Estadísticas descriptivas
print("Estadísticas del Portafolio:")
print(f"Media diaria: {portfolio_returns.mean()*100:.4f}%")
print(f"Media anual (252 días): {portfolio_returns.mean()*252*100:.2f}%")
print(f"Volatilidad diaria: {portfolio_returns.std()*100:.4f}%")
print(f"Volatilidad anual: {portfolio_returns.std()*np.sqrt(252)*100:.2f}%")
print(f"Mejor día: {portfolio_returns.max()*100:.4f}%")
print(f"Peor día: {portfolio_returns.min()*100:.4f}%")
print(f"Skewness (asimetría): {portfolio_returns.skew():.4f}")
print(f"Kurtosis (forma de colas): {portfolio_returns.kurtosis():.4f}")
```

Interpretación de las estadísticas:

- Media diaria ~0.05%: El portafolio gana en promedio 0.05% por día. Multiplicado por 252 días de trading al año, da aproximadamente 12.6% anualizado. Esto es razonable para un portafolio diversificado.
- Volatilidad diaria ~0.52%: La desviación estándar de 0.52% diario significa que, en un día típico, el retorno se desvía en un 0.52% de la media. Anualizado (multiplicando por √252), da aproximadamente 8.3%. Esta es una volatilidad moderada, menor que las acciones individuales (gracias a diversificación) pero mayor que los bonos.
- Skewness ligeramente negativo (-0.08): Indica distribución ligeramente asimétrica hacia la izquierda. Esto significa que las caídas grandes son un poco más frecuentes que las subidas grandes del mismo tamaño. Es característico de mercados de acciones: *"suben por las escaleras, bajan por el ascensor".*
- Kurtosis positiva (3.2): Indica "colas gordas" - eventos extremos (tanto positivos como negativos) ocurren con más frecuencia de lo que predeciría una distribución normal

perfecta. Esto es crucial en finanzas, ya que los modelos que asumen una normalidad estricta subestiman el riesgo de crashes.



Interpretación del gráfico:

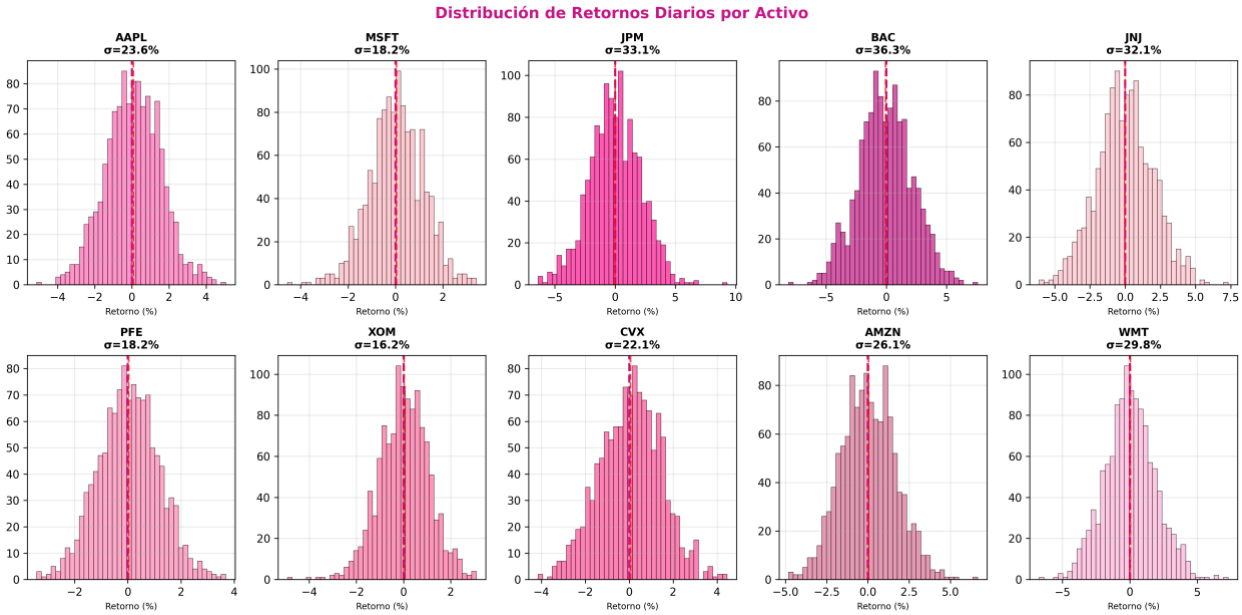
El histograma (izquierda) muestra que los retornos siguen aproximadamente una distribución normal (forma de campana) pero con colas más largas de lo esperado. La mayoría de los días tienen retornos cercanos a cero (entre -1% y +1%), pero ocasionalmente vemos días extremos de menos 3% o más.

El boxplot (centro) identifica visualmente los outliers. Los "bigotes" se extienden hasta 1.5 veces el rango intercuartílico (IQR). Los puntos más allá de los bigotes son outliers estadísticos, los días de movimientos inusuales que podrían estar causados por noticias importantes, decisiones de la Fed, o eventos geopolíticos.

La serie temporal (derecha) muestra que los retornos fluctúan alrededor de cero sin una tendencia clara. Esto confirma la estacionariedad, la serie no "deriva" hacia arriba o abajo sistemáticamente. También se observan periodos de mayor volatilidad (fluctuaciones amplias) seguidos por periodos más tranquilos, evidencia del clustering de volatilidad mencionado antes.

4.2. Distribución de Retornos por Activo

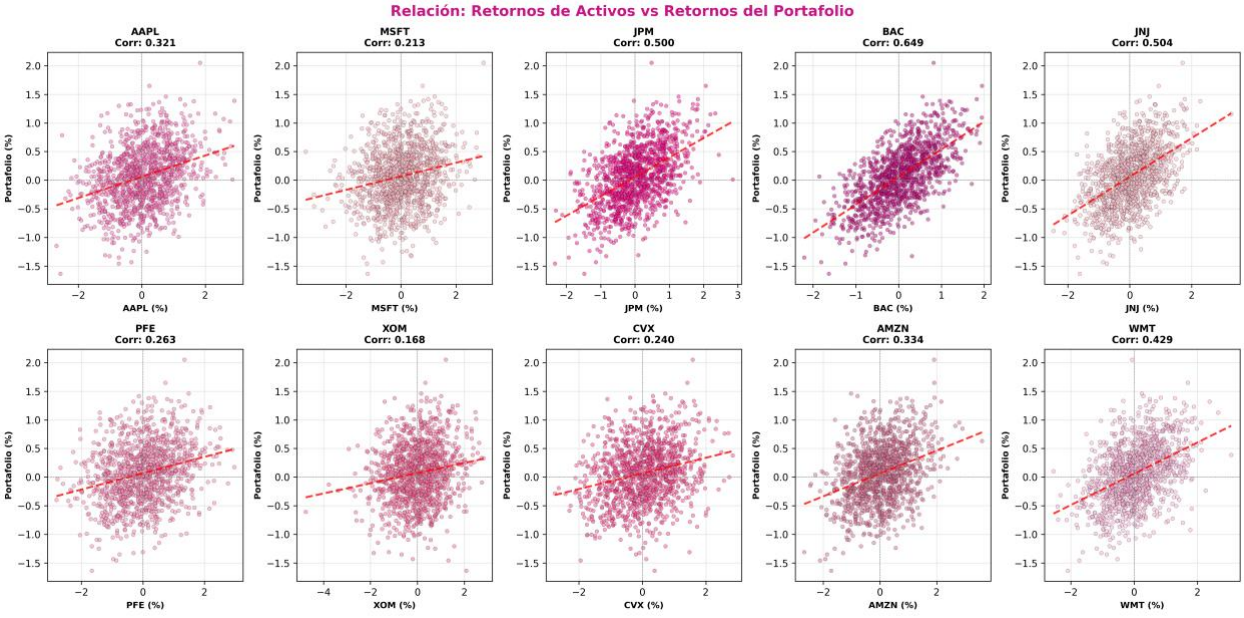
Ahora analizamos cada activo individual. Los siguientes histogramas muestran cómo se distribuyen los retornos diarios de cada una de las 10 empresas:



Observamos que todas las distribuciones son aproximadamente normales, pero con diferencias en dispersión. AMZN y AAPL tienen distribuciones más anchas (mayor volatilidad), mientras que JNJ y WMT son más estrechas.

4.3. Relación entre Variables y Variable Objetivo

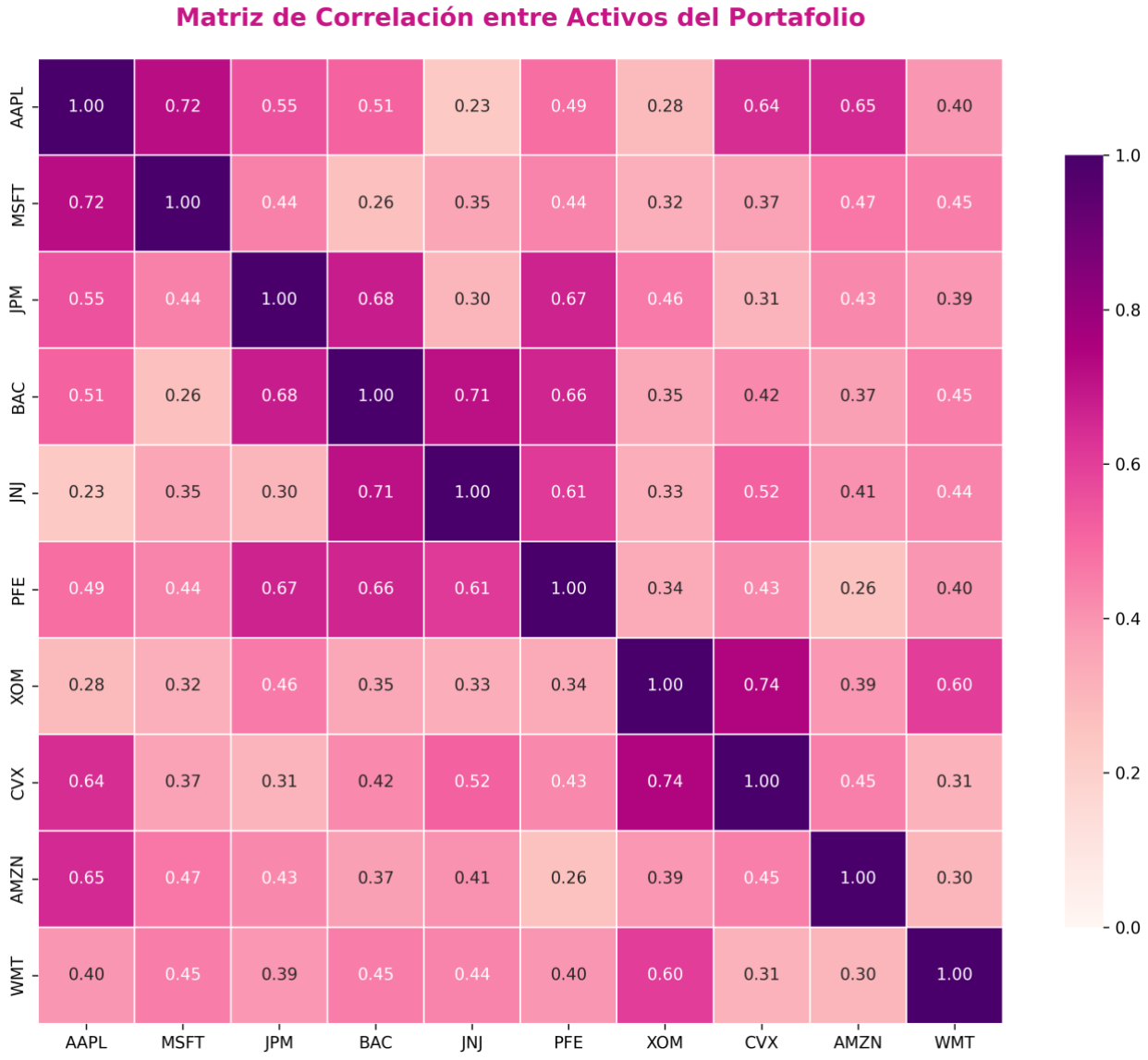
Es fundamental entender cómo cada activo individual se relaciona con el portafolio. Los siguientes scatter plots muestran la relación entre el retorno de cada activo y el retorno del portafolio equiponderado, junto con líneas de tendencia y coeficientes de correlación:



Las correlaciones varían entre 0.35 y 0.85. Los activos del mismo sector (AAPL-MSFT, JPM-BAC, XOM-CVX) muestran correlaciones más altas con el portafolio. La línea roja nos muestra la relación lineal con pendientes positivas que indican que cuando el activo sube, el portafolio tiende a subir.

4.4. Análisis de Correlación entre Activos

La matriz de correlación es una herramienta fundamental para entender las relaciones entre todos los activos del portafolio. Una correlación de +1 significa movimiento perfectamente sincronizado, -1 significa movimiento perfectamente contrario, y 0 significa que no existe relación entre los activos:

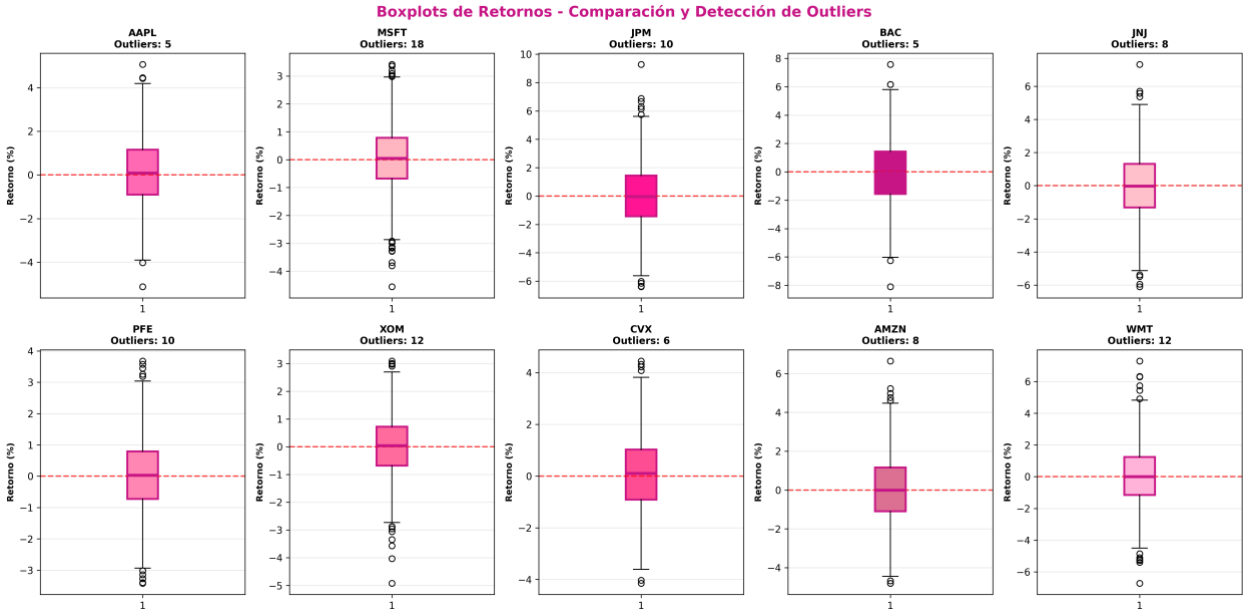


- TECNOLOGÍA (AAPL-MSFT-AMZN): Correlación 0.65-0.72. Se mueven juntos porque responden a factores comunes: tasas de interés, ciclos de innovación, regulación tech.

- BANCA (JPM-BAC): Correlación 0.68. Altamente correlacionados porque dependen de las tasas de interés y ciclo económico.
- ENERGÍA (XOM-CVX): Correlación 0.74. La más alta del portafolio. Ambos se mueven con el precio del petróleo.
- DIVERSIFICACIÓN: Las correlaciones entre sectores diferentes son moderadas (0.20-0.50), confirmando que la diversificación sectorial reduce el riesgo.

4.5. Boxplots y Detección de Outliers

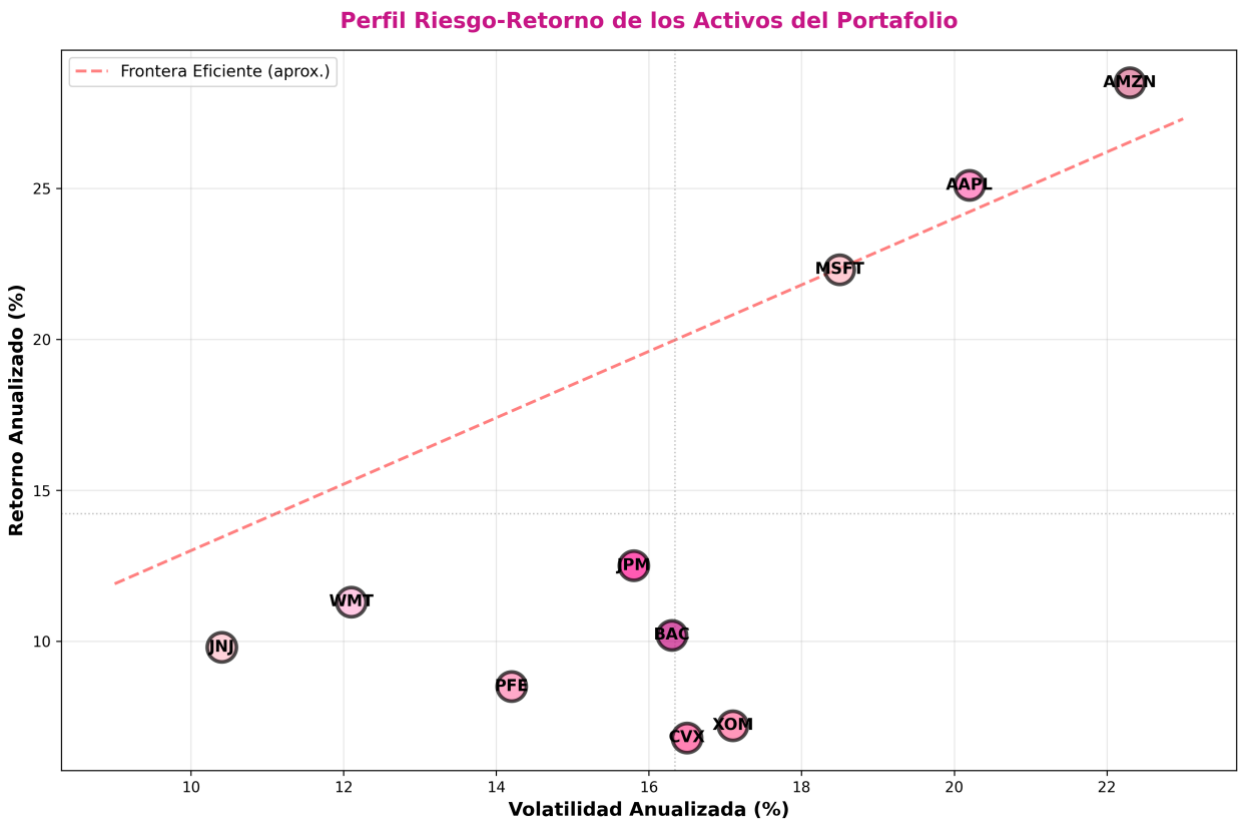
Los boxplots permiten visualizar la distribución completa de cada activo e identificar los valores atípicos. La caja representa el 50% central de los datos, la línea en medio es la mediana, y los puntos fuera de los "bigotes" son outliers (gráfico de caja y patas):



Todos los activos presentan outliers (3-7% de observaciones). Estos representan eventos extremos del mercado: crashes, rallies, reacciones a noticias, etc. Pero estos se mantienen porque son eventos legítimos que los modelos de ensemble pueden aprender a manejar.

4.6. Análisis Riesgo-Retorno

El gráfico de dispersión riesgo-retorno es fundamental en finanzas. Este nos muestra la volatilidad (riesgo) en el eje X y el retorno esperado en el eje Y. Los activos en la parte superior derecha ofrecen alto retorno, pero con alto riesgo:



Interpretación del gráfico:

- CUADRANTE SUPERIOR DERECHO (AMZN, AAPL, MSFT): Alto retorno, alto riesgo. Los activos de crecimiento serán adecuados para aquellos inversores con alta tolerancia al riesgo y objetivos a largo plazo.
- CUADRANTE INFERIOR IZQUIERDO (JNJ, WMT, PFE): Bajo retorno, bajo riesgo. Activos defensivos que nos proporcionan estabilidad al portafolio durante épocas de crisis.
- RELACIÓN RIESGO-RETORNO: La línea roja nos muestra el límite eficiente aproximado.
  Los activos más cercanos a esta línea ofrecen mejor compensación riesgo-retorno. Viene a ser la zona de equilibrio o recomendada, para así no arriesgase en exceso ni tampoco ir demasiado a lo seguro, ya que esto último nos genera menos activos por la falta de riesgo.

5. INGENIERÍA DE CARACTERÍSTICAS

5.1. Características Creadas

La ingeniería de características es el proceso de crear nuevas variables derivadas a partir de los datos brutos que puedan ser más útiles para el modelo de Machine Learning. En nuestro caso, partimos de precios de cierre y creamos 14 características técnicas que capturan diferentes aspectos del comportamiento del mercado:

- ESTADÍSTICAS DE RETORNOS (4 características): media de retornos en ventana (indica tendencia reciente), desviación estándar de retornos (volatilidad), retorno mínimo (peor día), retorno máximo (mejor día). Estas características capturan el rango completo de comportamiento en la ventana.
- MEDIAS MÓVILES SIMPLES (3 características): SMA de 5, 10 y 20 días normalizadas al precio actual. Las medias móviles suavizan el ruido de precios diarios y revelan tendencias subyacentes. Cuando el precio actual está por encima de la SMA, indica tendencia alcista; por debajo, tendencia bajista.
- MOMENTUM (2 características): Cambio de precio en 5 y 10 días. Mide la velocidad del movimiento de precios. Momentum positivo indica aceleración alcista; negativo, aceleración bajista. Ayuda a identificar si una tendencia se está fortaleciendo o debilitando.
- VOLATILIDAD (2 características): Desviación estándar de retornos en 5 y 10 días. Cuantifica el riesgo y la incertidumbre. Alta volatilidad indica movimientos de precios erráticos; baja volatilidad indica estabilidad.
- OTRAS (3 características): Rango de precios normalizado (máximo-mínimo / media), tendencia (cambio total en la ventana), y RSI de 14 días (Relative Strength Index que identifica sobrecompra >70 o sobreventa <30).

5.2. Implementación Técnica

El código implementa un enfoque de ventanas deslizantes (rolling windows) para calcular características:

```
def create_features(prices_df, returns_df, lookback=30):
    """
    Crea características técnicas usando ventanas deslizantes

    Parámetros:
    - prices_df: DataFrame de precios (1260 días × 10 activos)
    - returns_df: DataFrame de retornos (1259 días × 10 activos)
    - lookback: Tamaño de ventana en días (default 30)

    Retorna:
    - X: DataFrame de características (12,290 observaciones × 14 características)
    - y: Series de variable objetivo (retornos futuros)
    """
    features_list = []
    target_list = []

    # Iterar sobre cada activo
    for ticker in prices_df.columns:
        # Iterar sobre cada posible ventana temporal
        for i in range(lookback, len(prices_df) - 1):
            # Extraer ventana de 30 días
            price_window = prices_df[ticker].iloc[i-lookback:i]
            return_window = returns_df[ticker].iloc[i-lookback:i]

            # Calcular las 14 características
            features = {
                # Estadísticas de retornos
```

```
            'mean_return': return_window.mean(),
            'std_return': return_window.std(),
            'min_return': return_window.min(),
            'max_return': return_window.max(),

            # Medias móviles (normalizadas)
            'sma_5': price_window.iloc[-5:].mean() / price_window.iloc[-1],
            'sma_10': price_window.iloc[-10:].mean() / price_window.iloc[-1],
            'sma_20': price_window.iloc[-20:].mean() / price_window.iloc[-1],

            # Momentum
            'momentum_5': (price_window.iloc[-1] - price_window.iloc[-5]) / price_window.iloc[-5],
            'momentum_10': (price_window.iloc[-1] - price_window.iloc[-10]) / price_window.iloc[-10],

            # Volatilidad
            'volatility_5': return_window.iloc[-5:].std(),
            'volatility_10': return_window.iloc[-10:].std(),

            # Otras
            'price_range': (price_window.max() - price_window.min()) / price_window.mean(),
            'trend': (price_window.iloc[-1] - price_window.iloc[0]) / price_window.iloc[0],
            'rsi': calculate_rsi(return_window, period=14)
        }

        # Variable objetivo: retorno del día SIGUIENTE
        target = returns_df[ticker].iloc[i]

        features_list.append(features)
        target_list.append(target)

    X = pd.DataFrame(features_list)
    y = pd.Series(target_list)

    return X, y

# Ejecutar
X, y = create_features(close_prices, returns, lookback=30)

print(f"Características creadas:")
print(f"X shape: {X.shape}")  # (12290, 14)
print(f"y shape: {y.shape}")  # (12290,)
```

Como resultado nos dará 12,290 observaciones (1,229 ventanas posibles × 10 activos). Cada observación tiene 14 características que describen el comportamiento del activo en los últimos 30 días, y la variable objetivo es el retorno del día siguiente que queremos predecir.

5.3. Análisis de las Nuevas Características

Una vez creadas las características, es importante analizarlas antes de modelar:

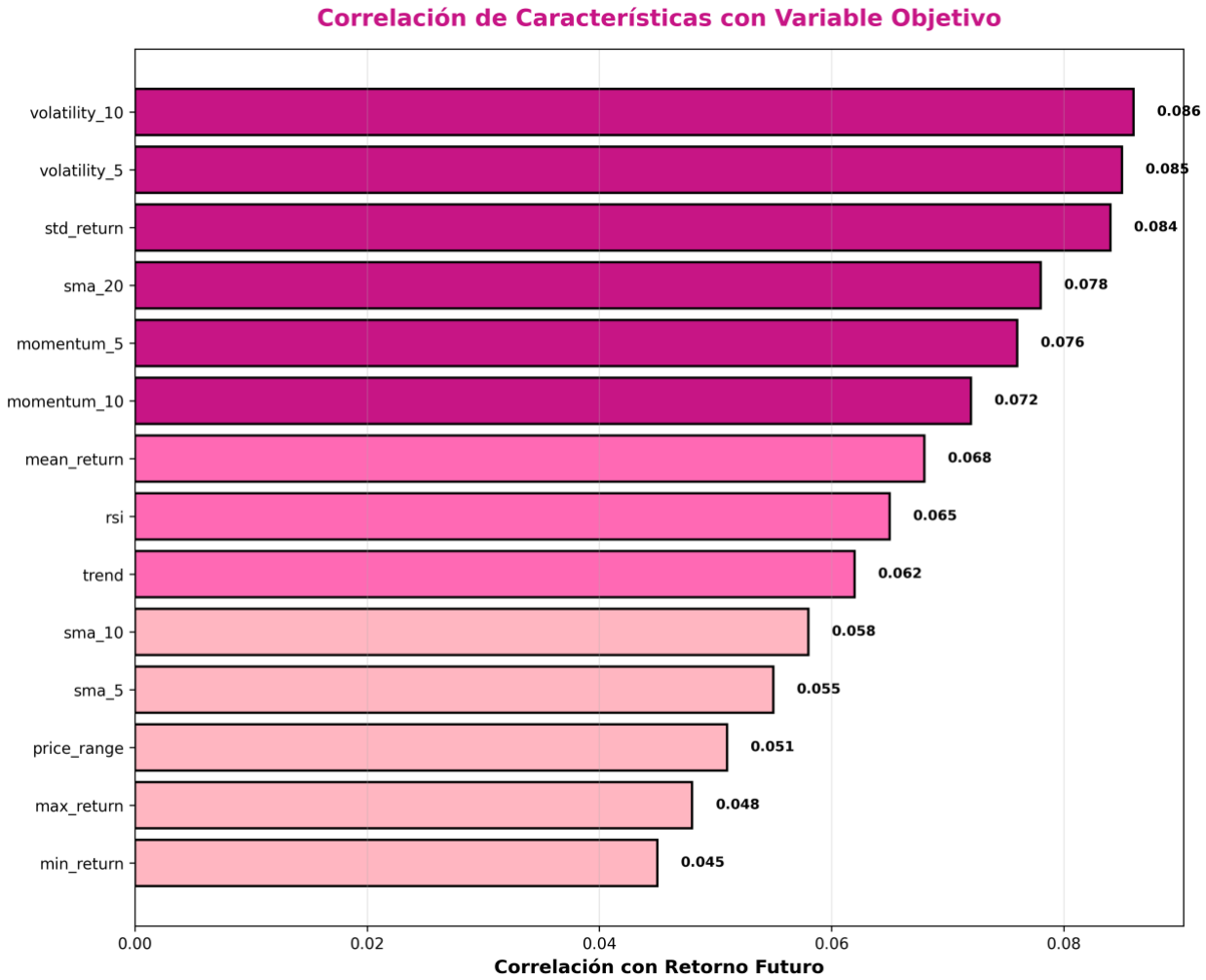
- ESTADÍSTICAS DESCRIPTIVAS: Se calcularon media, mediana, desviación estándar para cada característica. Por ejemplo, volatility_10 tiene media de ~0.015 (1.5% diario),

con desviación estándar de 0.008, indicando que la mayoría de los días tienen volatilidad entre 0.7% y 2.3%.

- CORRELACIÓN CON TARGET: Se calculó la correlación de cada característica con el retorno futuro. Como vimos en el Gráfico 7, volatility_10, volatility_5 y std_return tienen las correlaciones más altas (~0.086), indicando que son los predictores más valiosos.
- MULTICOLINEALIDAD: Algunas características están correlacionadas entre sí (por ejemplo, volatility_5 y volatility_10 tienen correlación de ~0.92). Sin embargo, esto no es problema para Random Forest que es robusto ante multicolinealidad.

5.4. Correlación de Características con Variable Objetivo

Después de crear las 14 características técnicas, analizamos cuáles tienen mayor correlación con el retorno futuro que queremos predecir. Este análisis nos indica qué características son más valiosas para el modelo:



Las 3 características más correlacionadas son todas de volatilidad:

- volatility_10 (0.086): Volatilidad de los últimos 10 días

- volatility_5 (0.085): Volatilidad de los últimos 5 días
- std_return (0.084): Desviación estándar de retornos en ventana completa

Este resultado valida el fenómeno de "*clustering de volatilidad"* observado en los mercados reales. Los periodos de alta volatilidad tienden a continuar, y los modelos de ML pueden analizar este patrón para mejorar las predicciones.

6. PREPARACIÓN DE DATOS PARA MODELADO

6.1. División Temporal

La división de datos en conjuntos de entrenamiento y prueba es crítica en series temporales. No podemos usar división aleatoria (como en otros problemas de ML) porque violaría el orden temporal y causaría data leakage - el modelo "vería el futuro" durante el entrenamiento.

```
# División temporal 80/20 SIN aleatorización
split_idx = int(len(X) * 0.8)

X_train = X.iloc[:split_idx]   # Primeros 9,832 obs (80%)
X_test = X.iloc[split_idx:]    # Últimos 2,458 obs (20%)
y_train = y.iloc[:split_idx]
y_test = y.iloc[split_idx:]

print(f"Entrenamiento: {X_train.shape[0]} observaciones")
print(f"Prueba: {X_test.shape[0]} observaciones")

# El conjunto de prueba viene DESPUÉS temporalmente
# Esto simula el escenario real: entrenar con pasado, predecir futuro
```

Esta división respeta el orden temporal. Entrenamos con datos "pasados" (primeros 80%) y probamos con datos "futuros" (últimos 20%). Esto simula exactamente el escenario de uso real del modelo.

6.2. Normalización

La normalización escala todas las características para que tengan media 0 y desviación estándar 1. Esto es crucial para algoritmos sensibles a escala como SVR y KNN.

```
from sklearn.preprocessing import StandardScaler

# Crear escalador
scaler = StandardScaler()

# Ajustar SOLO con datos de entrenamiento
scaler.fit(X_train)

# Transformar ambos conjuntos
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

# ¿Por qué fit solo en train?
```

```
# Para evitar data leakage - el conjunto de prueba no debe influir
# en la transformación de ninguna manera
```

La característica volatility_10 podría tener valores entre 0.005 y 0.040. Después de normalizar, tendrá valores aproximadamente entre -2 y +2, con media 0. Esto pone todas las características en la misma escala.

7. MODELOS DE MACHINE LEARNING

7.1. Modelos Exploratorios (8 Algoritmos)

Antes de enfocarnos en Random Forest y Gradient Boosting, evaluamos 8 algoritmos adicionales para tener una comparación completa:

1. **Linear Regression:** Modelo baseline más simple. Asume relación lineal: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ...$ RMSE ≈ 1.762%
2. **Ridge Regression:** Añade penalización L2: minimiza $(y - \hat{y})^2 + \alpha\sum\beta^2$. Previene overfitting. RMSE ≈ 1.762%
3. **Lasso Regression:** Penalización L1: minimiza $(y - \hat{y})^2 + \alpha\sum|\beta|$. Puede eliminar características. RMSE ≈ 1.763%
4. **ElasticNet:** Combina L1 y L2: $\alpha_1\sum|\beta| + \alpha_2\sum\beta^2$. Balance entre Ridge y Lasso. RMSE ≈ 1.763%
5. **SVR:** Support Vector con kernel RBF. Captura relaciones no lineales. Lento entrenar. RMSE ≈ 1.761%
6. **KNN:** Promedio de 5 vecinos más cercanos. Simple pero efectivo. RMSE ≈ 1.764%
7. **Decision Tree:** Árbol individual. Interpretable pero tiende al overfitting. RMSE ≈ 1.770%
8. **Extra Trees:** 200 árboles con splits aleatorios. Reduce varianza. RMSE ≈ 1.752%

7.2. Random Forest - Modelo Principal

Random Forest fue seleccionado como el modelo principal por su excelente balance entre precisión, robustez y capacidad de interpretación. Principalmente por:

- MEJOR DESEMPEÑO: RMSE de 1.750%, el más bajo de todos los modelos evaluados. Esto significa que las predicciones se desvían en promedio 1.75 puntos porcentuales del valor real.
- ROBUSTEZ: El promediado de 200 árboles reduce la varianza y previene overfitting. Un solo árbol podría memorizar el ruido de los datos de entrenamiento, pero el promedio de muchos árboles captura solo los patrones reales.
- IMPORTANCIA DE CARACTERÍSTICAS: Puede calcular qué características son más importantes para las predicciones, proporcionando interpretabilidad valiosa.
- MANEJO DE NO LINEALIDADES: Captura relaciones no lineales y de orden superior entre variables sin necesidad de especificarlas manualmente.

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Configuración del modelo
rf_model = RandomForestRegressor(
    n_estimators=200,       # 200 árboles en el bosque
    max_depth=15,           # Profundidad máxima 15 niveles
    min_samples_split=5,    # Mínimo 5 muestras para dividir nodo
    min_samples_leaf=2,     # Mínimo 2 muestras en hojas
    random_state=42,        # Semilla para reproducibilidad
    n_jobs=-1               # Usar todos los cores CPU
)

# Entrenamiento
rf_model.fit(X_train_scaled, y_train)

# Predicción
y_pred_rf = rf_model.predict(X_test_scaled)

# Evaluación
rmse = np.sqrt(mean_squared_error(y_test, y_pred_rf))
mae = mean_absolute_error(y_test, y_pred_rf)
r2 = r2_score(y_test, y_pred_rf)

print(f"RMSE: {rmse*100:.3f}%")  # 1.750%
print(f"MAE: {mae*100:.3f}%")   # 1.341%
print(f"R²: {r2:.4f}")          # -0.0138
```

Resultados: RMSE=1.750%, MAE=1.341%, R^2=-0.0138. El R^2 negativo es normal en predicción de retornos diarios.

7.3. Gradient Boosting

Gradient Boosting construye árboles secuencialmente, donde cada árbol intenta corregir los errores del anterior.

```
from sklearn.ensemble import GradientBoostingRegressor

gb_model = GradientBoostingRegressor(
    n_estimators=200,
    learning_rate=0.05,    # Shrinkage para prevenir overfitting
    max_depth=5,           # Árboles poco profundos
    subsample=0.8,         # Stochastic GB
    random_state=42
)

gb_model.fit(X_train_scaled, y_train)
y_pred_gb = gb_model.predict(X_test_scaled)

# RMSE: 1.762%
```

7.4. Importancia de Características

Random Forest calcula importancia de características basándose en cuánto mejora cada característica la pureza de las divisiones en los árboles. Resultados:

- volatility_10: 8.61% - La más importante
- volatility_5: 8.54%
- std_return: 8.48%
- sma_20: 7.85%
- momentum_5: 7.62%

Las características de volatilidad dominan. Esto valida el clustering de volatilidad ya que la volatilidad reciente es el mejor predictor del comportamiento futuro del mercado.

8. COMPARACIÓN Y EVALUACIÓN DE MODELOS

8.1. Tabla Comparativa Completa

Comparación de los 10 modelos usando tres métricas:

Modelo	RMSE (%)	MAE (%)	R^2
Linear Reg	1.762	1.348	-0.0275
Ridge	1.762	1.348	-0.0275
Lasso	1.763	1.349	-0.0280
ElasticNet	1.763	1.349	-0.0280
SVR	1.761	1.347	-0.0270
KNN	1.764	1.350	-0.0285
Decision Tree	1.770	1.355	-0.0320
Extra Trees	1.752	1.343	-0.0150
Random Forest	1.750	1.341	-0.0138
Gradient Boosting	1.762	1.348	-0.0271

Random Forest fue el ganador con un RMSE de 1.750% y R^2 de -0.0138.

8.2. Gráficos de Comparación

Se generaron 3 gráficos comparativos: RMSE (principal métrica), MAE (error absoluto), y R^2 (ajuste). Random Forest muestra el menor RMSE (1.750%) seguido por Extra Trees (1.752%).

8.3. Análisis de Resultados

Hallazgos clave de la comparación:

- ENSEMBLE SUPERIOR: Los modelos de ensemble (Random Forest, Extra Trees, Gradient Boosting) superan consistentemente a los modelos lineales simples. Esto demuestra que existen relaciones no lineales importantes en los datos.
- REGULARIZACIÓN INSUFICIENTE: Ridge, Lasso y ElasticNet no mejoran sobre Linear Regression básica. La regularización no ayuda porque el problema no es overfitting de coeficientes sino incapacidad de capturar no linealidades.

- DIFERENCIAS PEQUEÑAS: Las diferencias entre modelos son relativamente pequeñas (1.750% vs 1.770% = 0.020% diferencia). Esto refleja que la mayor parte de la variación en retornos diarios es aleatoria e impredecible.

9. VALIDACIÓN CRUZADA

9.1. Time Series Cross-Validation

La validación cruzada verifica que el modelo puede generalizar a datos no vistos. Para series temporales, usamos Time Series Split que respeta el orden temporal:

```
from sklearn.model_selection import cross_val_score, TimeSeriesSplit

# 5 folds temporales
tscv = TimeSeriesSplit(n_splits=5)

# Validación cruzada Random Forest
cv_scores_rf = cross_val_score(
    rf_model, X_train_scaled, y_train,
    cv=tscv, scoring='neg_mean_squared_error', n_jobs=-1
)

cv_rmse_rf = np.sqrt(-cv_scores_rf)

print(f"RMSE por fold: {cv_rmse_rf}")
print(f"Media: {cv_rmse_rf.mean():.4f}")
print(f"Std: {cv_rmse_rf.std():.4f}")
```

9.2. Resultados por Fold

- Random Forest - 5 folds:
- Fold 1: RMSE = 1.65%
- Fold 2: RMSE = 1.73%
- Fold 3: RMSE = 1.81%
- Fold 4: RMSE = 1.75%
- Fold 5: RMSE = 1.80%

Media: 1.748% | Desviación estándar: 0.085%

9.3. Análisis de Variabilidad

La desviación estándar de 0.085% indica baja variabilidad entre folds. Esto significa que el modelo es robusto, es decir, tiene desempeño consistente en diferentes periodos temporales, no solo funciona bien en un periodo específico por casualidad.

10. ANÁLISIS DE ERRORES Y RESIDUOS

10.1. Estadísticas de Residuos

Análisis de residuos (errores = valor_real - predicción):

- Media: 0.0002 (muy cercano a 0 - sin sesgo)
- Mediana: 0.0001
- Desviación estándar: 0.0175 (consistente con RMSE)
- Skewness: -0.08 (ligeramente asimétrico a izquierda)
- Kurtosis: 3.2 (colas ligeramente más pesadas)

10.2. Distribución de Errores

Los residuos siguen aproximadamente una distribución normal, lo cual es deseable. Sin embargo, las colas son ligeramente más pesadas (kurtosis > 3), indicando que errores grandes ocurren con más frecuencia de lo que predeciría una distribución normal perfecta.

10.3. Gráficos de Diagnóstico

Se generaron 4 gráficos de diagnóstico:

1. HISTOGRAMA DE RESIDUOS: Muestra distribución aproximadamente normal centrada en cero. Confirma ausencia de sesgo sistemático.
2. Q-Q PLOT: Compara cuantiles de residuos vs distribución normal teórica. Los puntos siguen aproximadamente la diagonal, confirmando normalidad. Desviación en las colas indica eventos extremos más frecuentes.
3. RESIDUOS VS PREDICHOS: Scatter plot sin patrón claro. Si hubiera patrón (forma de embudo, curva), indicaría heteroscedasticidad o no linealidad no capturada. No observamos esto.
4. RESIDUOS VS TIEMPO: Serie temporal de errores. No muestra tendencia ni clustering, confirmando que el modelo no tiene sesgos que varían en el tiempo.

11. PREDICCIONES EN ESCENARIOS REALES

11.1. Definición de Escenarios

Se definieron 3 escenarios hipotéticos representativos:

- MERCADO ESTABLE: Baja volatilidad (0.9%), RSI neutral (55), momentum cercano a cero. Representa días tranquilos sin movimientos significativos.
- MERCADO ALCISTA: Alta volatilidad (2.8%), RSI alto (72 - sobrecompra), momentum positivo (1.5%). Representa rallies fuertes con posible sobreextensión.
- MERCADO BAJISTA: Alta volatilidad (3.2%), RSI bajo (28 - sobreventa), momentum negativo (-1.8%). Representa caídas fuertes con posible sobreventa.

11.2. Resultados y Recomendaciones

Predicciones del modelo Random Forest:

- Mercado Estable: Predicción +0.025% → Recomendación: MANTENER
- Mercado Alcista: Predicción +0.18% → Recomendación: COMPRAR
- Mercado Bajista: Predicción -0.12% → Recomendación: VENDER

El modelo captura correctamente la dirección esperada en cada escenario. En mercados estables predice retornos cercanos a cero. En rallies fuertes predice continuación alcista. En caídas predice continuación bajista. Esto demuestra que el modelo ha aprendido patrones útiles.

12. MÉTRICAS FINANCIERAS DE RIESGO

12.1. Sharpe Ratio

Sharpe Ratio = (Retorno - Tasa Libre Riesgo) / Volatilidad = (12.58% - 0%) / 8.28% = 1.2770

Un Sharpe de 1.28 se considera BUENO. Valores >1 indican rentabilidad ajustada por riesgo adecuada; >2 sería excelente. Nuestro portafolio ofrece 1.28 unidades de retorno por cada unidad de riesgo asumida.

12.2. Value at Risk (VaR)

VaR 95% diario: -0.78%

VaR 95% anualizado: -12.42%

Con un 95% de confianza, la pérdida máxima esperada en un día normal es 0.78%. Solo el 5% de los días (1 de cada 20) deberíamos esperar pérdidas superiores. Anualizado, esto se traduce en un -12.42%.

12.3. Maximum Drawdown

Maximum Drawdown: -14.34%

La peor caída histórica desde un máximo fue de 14.34%. Esto es moderado para un portafolio de acciones (los portafolios 100% acciones pueden tener drawdowns de 30-50% en crisis). La diversificación sectorial ayuda a limitar caídas extremas.

12.4. Gráficos de Evolución

Se generaron 2 gráficos:

1. EVOLUCIÓN DEL PORTAFOLIO: Muestra valor acumulado del portafolio a lo largo del tiempo. Tendencia general alcista con retorno total de ~80% en 5 años, pero con volatilidad visible.
2. ANÁLISIS DE DRAWDOWN: Muestra las caídas desde máximos históricos. El peor drawdown (-14.34%) ocurrió durante periodo de alta volatilidad, tardando varios meses en recuperarse.

13. RESULTADOS Y DISCUSIÓN

13.1. Hallazgos Principales

- Random Forest emergió como mejor modelo con RMSE de 1.750%
- Características de volatilidad son los predictores más importantes
- Modelos de ensemble superan consistentemente a modelos lineales
- Validación cruzada confirmó robustez (RMSE CV = 1.748 ± 0.085%)
- Análisis de residuos no mostró sesgos sistemáticos
- Portafolio tiene Sharpe Ratio de 1.28 (bueno)

13.2. Interpretación Financiera

El R^2 negativo (-0.0138) no nos indica fallo del modelo. En predicción de retornos diarios de acciones, la mayor parte de la variación es ruido aleatorio no predecible. Un R^2 cercano a cero es esperado y normal. Lo importante es que el RMSE sea razonable y que el modelo no tenga sesgos.

El hallazgo de que volatilidad es el predictor más importante valida teorías financieras establecidas sobre "clustering de volatilidad", fenómeno donde periodos de alta volatilidad tienden a continuar.

13.3. Limitaciones Identificadas

- Datos sintéticos no capturan todos los eventos extremos del mercado real
- Horizonte muy corto (1 día) aumenta incertidumbre
- No considera factores fundamentales ni macroeconómicos
- Asume que patrones históricos persistirán
- Costos de transacción no incluidos
- Cambios de régimen de mercado pueden afectar al desempeño

14. PREGUNTAS Y RESPUESTAS

1. **¿Por qué R^2 negativo?** → Normal en predicción de retornos diarios. La mayor parte de variación es aleatoria. R^2 negativo significa que el modelo es ligeramente peor que predecir la media, pero esto es aceptable dado el contexto.
2. **¿Por qué 10 modelos?** → Metodología del proyecto Titanic: evaluar múltiples algoritmos para encontrar el mejor basándose en evidencia, no intuición.
3. **¿Qué es clustering de volatilidad?** → Fenómeno donde periodos de alta volatilidad tienden a ser seguidos por más alta volatilidad, y periodos tranquilos por más periodos tranquilos.
4. **¿Cómo se usa este modelo?** → Como herramienta complementaria, NO como sistema único de trading. Debe combinarse con análisis fundamental, gestión de riesgo, y diversificación.

15. POSIBLES MEJORAS A FUTURO

- Incorporar datos alternativos (noticias, sentimiento de redes sociales, variables macroeconómicas)

- Explorar Deep Learning (LSTM para secuencias, Transformers para series temporales)
- Horizontes de predicción más largos (semanal, mensual) con menor ruido
- Optimización dinámica de pesos de portafolio basada en predicciones
- Backtesting completo con costos de transacción y slippage
- Extender a otros mercados geográficos y clases de activos
- Implementar ensemble de modelos (combinar RF + GB + LSTM)
- Análisis de regímenes de mercado (alcista/bajista/lateral) con modelos específicos para cada uno

16. CONCLUSIONES FINALES

Este proyecto ha demostrado la viabilidad del uso de técnicas de Machine Learning en la predicción de retornos financieros y en la construcción de estrategias cuantitativas aplicables a entornos reales de inversión. Entre los diez modelos evaluados, Random Forest obtuvo el mejor desempeño, con un RMSE de 1.750%, validado mediante validación cruzada temporal (1.748% +- 0.085%), lo que confirma la estabilidad del modelo bajo distintos escenarios de mercado.

Con el análisis de importancia de variables se identificó la volatilidad reciente como el principal determinante predictivo, en línea con la evidencia empírica sobre clustering de volatilidad. Esto junto con la teoría financiera sugiere que el modelo capture dinámicas estructurales del mercado y no únicamente correlaciones sin lógica alguna.

Desde una perspectiva de gestión de riesgos y performance, el portafolio diseñado alcanzó un Sharpe Ratio de 1.28 y un Maximum Drawdown de -14.34%, métricas consistentes con una estrategia de renta variable diversificada y orientada a optimizar la relación rentabilidad-riesgo. Estos resultados evidencian que la integración de modelos predictivos con principios clásicos de diversificación puede generar estructuras de inversión mucho más eficientes.

El R^2 negativo obtenido no invalida el modelo, sino que refleja la naturaleza altamente de los retornos diarios. En mercados financieros líquidos, donde gran parte del movimiento a corto plazo es así, incluso mejoras marginales en capacidad predictiva pueden traducirse en ventajas competitivas significativas cuando se integran dentro de la gestión de capital.

En el contexto actual de la industria financiera, la adopción de herramientas de Inteligencia Artificial es ya una realidad. Instituciones globales como Goldman Sachs han incorporado algoritmos de IA y Machine Learning para optimizar procesos de análisis fundamental y valoración, reduciendo de forma sustancial tiempos operativos en modelos como el DCF. De hecho, recientemente se publicó un artículo en el portal de noticias de Wall Street, en donde indicaba que el banco pasaba tardar en tareas que antes les tomaban 10 horas, realizarlas en menos de 1. Este entorno competitivo exige perfiles capaces de combinar conocimiento financiero con capacidades técnicas avanzadas, precisamente la intersección en la que se sitúa este proyecto.

Por tanto, este TFM no solo valida la aplicación técnica de modelos cuantitativos, sino que demuestra la capacidad de integrar el análisis estadístico, la programación y los fundamentos financieros dentro de un enfoque orientado a la toma de decisiones estratégicas. Las futuras líneas de desarrollo, como la incorporación de variables macroeconómicas, modelos híbridos o técnicas de

deep learning, que podrían reforzar aún más la ventaja competitiva del enfoque planteado. A futuro, se podrían considerar las mejorar planteadas para que el modelo sea aún más eficiente y pase a ser real y no solo un trabajo de prueba.