Proyecto: Análisis y Predicción de ETFs de EEUU

Innovación en Formación Profesional

Autor/es: Ivan López Tomàs y Nelson William Morales

Tutor/es: Yuandry Noa Gálvez, Antonio Guil y Anthony Torres

Repositorio: https://github.com/nwmorales/ETF_Predictions

Resumen

El presente proyecto aborda el desafío del análisis y la predicción del comportamiento de los Fondos Cotizados en Bolsa (ETFs) del mercado estadounidense, caracterizado por grandes volúmenes de datos y dinámicas complejas. El objetivo principal fue diseñar, implementar y evaluar un sistema integral basado en tecnologías Big Data e Inteligencia Artificial para la ingesta, procesamiento, almacenamiento y modelado predictivo de datos de ETFs.

La metodología implicó la adquisición de datos históricos de ETFs desde fuentes públicas como Kaggle y la API de Yahoo Finance. Se desarrolló una arquitectura de datos robusta utilizando Docker y Docker Compose para la contenerización y gestión de servicios como Apache NiFi, encargado del flujo de ingesta y transformación de datos; Apache HDFS, como sistema de almacenamiento masivo y persistente para el histórico de datos; y Apache Cassandra, como base de datos NoSQL optimizada para la ingesta rápida y el acceso eficiente a datos para análisis y predicción. Sobre esta infraestructura, se plantea la aplicación y evaluación de diversos modelos de predicción, incluyendo Regresión Lineal, Prophet, Redes Neuronales y algoritmos de Clasificación, con análisis complementarios realizados en Orange Datamining y visualización de resultados mediante Power BI, y graficaciones de los modelos en google collab.

Entre los principales resultados, se destaca la exitosa implementación y validación funcional de la arquitectura Big Data propuesta, la cual demostró su capacidad para gestar, transformar y almacenar correctamente los datos de ETFs. Las pruebas iniciales de modelado en Orange Datamining con validación cruzada indicaron que el modelo **Red Neuronal** obtuvo un rendimiento prometedor, aunque se observaron valores de R^2 excepcionalmente altos que requerirán una discusión detallada. Los modelos de Prophet desarrollados en Python alcanzaron una precisión del 58.23% en la comparación entre los valores predichos por nuestro modelo y los datos reales extraídos mediante la API

Se concluye que la integración de tecnologías Big Data y técnicas de Inteligencia Artificial proporciona un marco potente y escalable para el análisis avanzado de ETFs. El sistema desarrollado sienta una base sólida para futuras investigaciones y la aplicación de modelos más sofisticados, con potencial para generar insights valiosos en el ámbito de la inversión financiera.

Palabras Clave: ETFs, Predicción Financiera, Big Data, Inteligencia Artificial, Apache NiFi, Apache Cassandra, Apache Hadoop, Docker, Machine Learning, Series Temporales.

Abstract

This project addresses the challenge of analyzing and predicting the behavior of Exchange-Traded Funds (ETFs) in the US market, characterized by large volumes of data and complex dynamics. The main objective was to design, implement, and evaluate a comprehensive system based on Big Data and Artificial Intelligence technologies for the ingestion, processing, storage, and predictive modeling of ETF data.

The methodology involved the acquisition of historical ETF data from public sources such as Kaggle and the Yahoo Finance API. A robust data architecture was developed using Docker and Docker Compose for the containerization and management of services such as Apache NiFi, responsible for the data ingestion and transformation pipeline; Apache HDFS, as a massive and persistent storage system for historical data; and Apache Cassandra, as a NoSQL database optimized for rapid ingestion and efficient data access for analysis and prediction. Based on this infrastructure, the application and evaluation of various prediction models is proposed, including Linear Regression, Prophet, Neural Networks, and Classification algorithms, with complementary analyses performed in Orange Datamining and results visualization using Power BI, with model graphs in Google Collab
.
Among the main results, the successful implementation and functional validation of the proposed Big Data architecture stands out, demonstrating its ability to successfully ingest, transform, and store ETF data. Initial modeling tests in Orange Datamining with cross-validation indicated that the Neural Network model performed promisingly, although exceptionally high R^2 values were observed, which will require a detailed discussion. Prophet models developed in Python achieved an accuracy of 58.23% when comparing the values predicted by our model with the real data extracted through the API.

It is concluded that the integration of Big Data technologies and Artificial Intelligence techniques provides a powerful and scalable framework for advanced ETF analysis. The developed system lays a solid foundation for future research and the application of more sophisticated models, with the potential to generate valuable insights in the field of financial investment.

Keywords: ETFs, Financial Forecasting, Big Data, Artificial Intelligence, Apache NiFi, Apache Cassandra, Apache Hadoop, Docker, Machine Learning, Time Series.

1. Introducción

Los Fondos Cotizados en Bolsa (ETFs) se han consolidado como uno de los instrumentos de inversión más dinámicos y de mayor crecimiento en los mercados financieros globales. Su flexibilidad, diversificación y accesibilidad los han convertido en una opción preferente tanto para inversores institucionales como minoristas, particularmente en el robusto mercado estadounidense.

El análisis y la predicción del comportamiento de los activos financieros continúan siendo un desafío central y un área de intensa investigación. La capacidad de anticipar movimientos de mercado o identificar patrones subyacentes es crucial para la optimización de estrategias de inversión y la gestión de riesgos.

La Inteligencia Artificial y las tecnologías Big Data en la última década han abierto nuevas fronteras en este campo. Estas herramientas ofrecen capacidades sin precedentes para procesar y analizar la vasta cantidad de datos generados por los mercados financieros, identificar relaciones no lineales complejas y desarrollar modelos predictivos con un potencial de precisión superior a los enfoques tradicionales.

1.1. Motivación

Los mercados financieros, y en particular los ETFs, ofrecen oportunidades de inversión, pero su comportamiento es complejo y volátil. Predecir sus movimientos es un desafío constante. Este proyecto surge de la necesidad de:

* Aplicar técnicas de ciencia de datos para entender mejor el comportamiento de los ETFs.
* Explorar la capacidad predictiva de modelos de Machine Learning sobre datos financieros históricos.
* Desarrollar una metodología sistemática para la selección de activos (basada en rendimiento pasado) y la posterior modelización.
* Generar herramientas o insights que *potencialmente* podrían ayudar a fundamentar estrategias de inversión (con las debidas precauciones).

1.2. Objetivos del proyecto

- Este proyecto se centra en el análisis de Fondos Cotizados (ETFs) relevantes del mercado estadounidense. El objetivo principal es identificar los ETFs con mejor rendimiento histórico reciente y utilizar sus datos para construir modelos predictivos que intenten anticipar futuros movimientos del mercado. La finalidad última es explorar la viabilidad de utilizar estos modelos como apoyo en la toma de decisiones de inversión.
- El flujo de trabajo abarca desde la obtención y limpieza de datos históricos de ETFs hasta el entrenamiento y evaluación de modelos de Machine Learning para la predicción de precios o tendencias.

- Adquirir y preprocesar conjuntos de datos históricos relevantes de una selección de ETFs estadounidenses, utilizando fuentes como Kaggle y la API de Yahoo Finance.
- Realizar un análisis exploratorio exhaustivo de los datos recopilados para identificar características, tendencias y patrones preliminares, empleando herramientas como Pandas en el entorno de Google Colab.
- Diseñar e implementar una arquitectura de Big Data escalable, utilizando Docker para la contenerización, Apache NiFi para la ingesta y el flujo de datos, Apache Cassandra para el almacenamiento de series temporales y Apache Hadoop para el procesamiento y almacenamiento masivo.
- Desarrollar, entrenar y evaluar el rendimiento de modelos de regresión, incluyendo Regresión Lineal y Prophet, para la predicción de la evolución de los precios de los ETFs.
- Implementar y evaluar Redes Neuronales LSTM, para el modelado y predicción del comportamiento de los ETFs.
- Visualizar los datos, los resultados de los análisis y las predicciones de los modelos de manera interactiva y comprensible utilizando Power BI y complementar el análisis con técnicas de minería de datos mediante Orange Datamining.

2. Conceptos Fundamentales y Tecnologías Aplicadas

2.1. Fondos Cotizados en Bolsa (ETFs)

Un Fondo Cotizado en Bolsa (ETF, por sus siglas en inglés Exchange-Traded Fund) es un vehículo de inversión colectiva cuyas participaciones se negocian en mercados secundarios de valores, al igual que las acciones. Su objetivo principal suele ser replicar el comportamiento de un determinado índice bursátil, sector económico, materia prima o una cesta diversificada de activos...

La versatilidad de los ETFs se refleja en su amplia tipología, que incluye desde los tradicionales ETFs indexados que siguen índices como el S&P 500 o el NASDAQ 100, hasta ETFs sectoriales, temáticos, de renta fija, e incluso de gestión activa, aunque estos últimos representan una porción menor del mercado...

2.2. Modelos de Predicción Aplicados

2.2.1. Regresión Lineal

¿Qué es un modelo de predicción basado en una Regresión Lineal?

Un modelo de Regresión Lineal es un algoritmo fundamental en estadística y aprendizaje automático que busca modelar la relación lineal entre una variable dependiente (en tu caso, el precio futuro del ETF) y una o más variables independientes o predictoras (que podrían ser precios históricos del ETF, volúmenes de negociación, indicadores macroeconómicos, etc.).

El objetivo del modelo de regresión lineal es encontrar los valores de los coeficientes que mejor se ajusten a los datos históricos, minimizando la suma de los errores al cuadrado (método de mínimos cuadrados ordinarios - OLS). Una vez entrenado el modelo con datos

Históricos, se pueden utilizar los coeficientes aprendidos para predecir valores futuros de la variable dependiente basándose en los valores futuros de las variables predictoras.

¿Hasta dónde llega la utilidad del modelo de Regresión Lineal en la predicción de valores de ETFs?

En la predicción de valores futuros de ETFs, un modelo de regresión lineal puede utilizarse de diversas maneras, dependiendo de las variables predictoras que se incluyan:

* Predicción basada en precios históricos: La forma más simple sería utilizar precios históricos del propio ETF como variable predictora (por ejemplo, el precio de ayer para predecir el precio de hoy, o un promedio de precios pasados). Esto se asemeja a algunos enfoques básicos de análisis técnico.
  Como es nuestro modelo.

* Incorporación de otras variables financieras: Se pueden incluir otras variables financieras que se consideren relevantes para el precio del ETF, como:

- Índices de referencia: El rendimiento del índice subyacente del ETF (por ejemplo, el SP 500 para un ETF que lo rastrea).

- Precios de activos relacionados: Los precios de acciones individuales dentro del ETF o de otros ETFs relacionados.

- Volúmenes de negociación: El volumen de acciones del ETF negociadas.

- Tasas de interés: Las tasas de interés pueden influir en el atractivo de diferentes clases de activos.

- Tipos de cambio: Relevante para ETFs que invierten en activos extranjeros.

* Incorporación de indicadores macroeconómicos: Variables como el PIB, la inflación, las tasas de desempleo, los índices de confianza del consumidor, etc., podrían utilizarse para modelar tendencias a largo plazo o efectos cíclicos en el precio del ETF.

El modelo de regresión lineal busca identificar y cuantificar la relación lineal entre estas variables predictoras y el precio futuro del ETF. Una vez entrenado, se pueden ingresar los valores futuros estimados de las variables predictoras para obtener una predicción del precio del ETF.

Puntos fuertes de un modelo de Regresión Lineal en el contexto de la predicción de ETFs:

* Simplicidad e interpretabilidad: Los modelos de regresión lineal son relativamente fáciles de entender e interpretar. Los coeficientes (β) indican la dirección y la magnitud del efecto de cada variable predictora sobre la variable dependiente. Esto puede proporcionar información valiosa sobre los factores que influyen en el precio del ETF.

* Rapidez y eficiencia: El entrenamiento y la predicción con modelos de regresión lineal son computacionalmente eficientes, incluso con grandes conjuntos de datos.
  Base estadística sólida: La regresión lineal tiene una base estadística bien establecida, con métricas y pruebas para evaluar la bondad del ajuste del modelo y la significancia de las variables predictoras (por ejemplo, R-cuadrado, pruebas t, pruebas F).

* Flexibilidad en la inclusión de variables: Permite incorporar una amplia variedad de variables predictoras que se consideren relevantes para el precio del ETF.
  Puede capturar relaciones lineales: Si la relación entre las variables predictoras y el precio del ETF es aproximadamente lineal, la regresión lineal puede proporcionar buenas predicciones.

2.2.2. Redes Neuronales

¿Qué es una red neuronal?

Una red neuronal artificial (RNA) es un modelo computacional perteneciente al campo del aprendizaje automático, cuyo diseño se inspira en la estructura y funcionamiento del sistema nervioso biológico. Este tipo de arquitectura resulta especialmente eficaz para abordar problemas de alta complejidad que presentan limitaciones para los enfoques algorítmicos tradicionales, como es el caso del reconocimiento de imágenes, el procesamiento del lenguaje natural (PLN) o la traducción automática.

Las redes neuronales están conformadas por unidades elementales denominadas neuronas artificiales, las cuales se agrupan en capas (de entrada, ocultas y de salida) y se interconectan mediante enlaces ponderados. Cada neurona recibe un conjunto de señales de entrada, las procesa mediante una función de activación no lineal y propaga el resultado hacia las neuronas de la capa siguiente. Los pesos asignados a cada conexión cuantifican la influencia de una neurona sobre otra y constituyen los parámetros ajustables del modelo.

Durante la fase de entrenamiento, típicamente mediante algoritmos de optimización basados en retropropagación del error y descenso del gradiente, los pesos son modificados iterativamente con el objetivo de minimizar una función de coste definida sobre un conjunto de datos de entrenamiento. Este mecanismo de ajuste permite a la red neuronal aprender representaciones internas de los datos y generalizar patrones subyacentes, habilitando así su aplicación en tareas de predicción, clasificación o generación de contenido, entre otras.

En nuestro caso en concreto usamos una Red Neuronal Recurrente o RNN

¿Qué es un modelo de predicción basado en una Red Neuronal Recurrente (RNN)?

Una Red Neuronal Recurrente (RNN) es un tipo de red neuronal diseñada específicamente para procesar secuencias de datos, donde el orden y la dependencia temporal entre los elementos son importantes. A diferencia de las redes neuronales feedforward tradicionales, las RNN tienen conexiones que forman ciclos, lo que les permite mantener un estado interno (memoria) que captura información sobre las entradas pasadas en la secuencia.

En el contexto de la predicción de series temporales como los precios de ETFs, una RNN puede aprender patrones y dependencias a lo largo del tiempo.
La arquitectura básica de una RNN implica que la salida de una capa en un paso de tiempo se alimenta de nuevo a la misma capa en el siguiente paso de tiempo. Esto permite que la red "recuerde" información de pasos anteriores en la secuencia.

LSTM (Long Short-Term Memory): Una RNN con Memoria a Largo Plazo

Aunque las RNN teóricamente pueden aprender dependencias a largo plazo, en la práctica, las RNN simples sufren de un problema conocido como el "desvanecimiento del gradiente" (y el problema opuesto, la "explosión del gradiente"). Esto dificulta que la red aprenda y retenga información de pasos de tiempo muy lejanos en la secuencia.

Las redes LSTM son una arquitectura especial de RNN diseñada para superar estos problemas y capturar dependencias a largo plazo de manera más efectiva. Introducen un mecanismo llamado "celda de memoria" y "puertas" (gates) que controlan el flujo de información dentro de la red. Estas puertas utilizan funciones sigmoideas para producir valores entre 0 y 1, actuando como interruptores que modulan el flujo de información. La combinación de estas puertas permite a las LSTM aprender cuándo retener información importante durante largos períodos y cuándo descartar información irrelevante.

¿Para qué sirve un modelo basado en una Red Neuronal Recurrente (LSTM) en la predicción de valores de ETFs?

En la predicción de valores futuros de ETFs, un modelo LSTM puede ser muy útil por su capacidad para:

- Capturar dependencias temporales complejas: Los precios de los ETFs están influenciados por una compleja interacción de factores a lo largo del tiempo. Las LSTM pueden aprender patrones secuenciales que podrían ser difíciles de capturar con modelos lineales como la regresión lineal. Por ejemplo, podrían identificar patrones de volatilidad, tendencias a corto y mediano plazo, y posibles efectos de "momentum".
- Modelar la volatilidad: Aunque no modelan directamente la volatilidad como algunos modelos específicos (como los modelos GARCH), las LSTM pueden aprender patrones en la volatilidad histórica y, potencialmente, predecir cambios en la volatilidad futura.
- Manejar secuencias de entrada largas: La capacidad de las LSTM para retener información a largo plazo les permite procesar secuencias de precios históricos más extensas y capturar dependencias que se extienden durante períodos de tiempo más largos.
- Incorporar múltiples variables de entrada: Al igual que la regresión lineal, las LSTM pueden tomar múltiples series temporales como entrada (por ejemplo, precios históricos del ETF, volúmenes, precios de otros activos relacionados, indicadores técnicos) para realizar la predicción. La red puede aprender las complejas interrelaciones temporales entre estas variables y el precio futuro del ETF.

Puntos fuertes de un modelo basado en LSTM en el contexto de la predicción de ETFs:

* Capacidad para aprender dependencias a largo plazo: Esta es la principal ventaja de las LSTM sobre las RNN simples y los modelos tradicionales de series temporales. Pueden capturar patrones que se desarrollan durante períodos de tiempo extensos.
* Manejo de secuencias complejas: Las LSTM son capaces de modelar relaciones no lineales y dependencias temporales intrincadas que pueden existir en los mercados financieros.
* Adaptabilidad a diferentes patrones: La arquitectura flexible de las LSTM les permite adaptarse a una variedad de patrones y dinámicas en los datos del precio del ETF.
* Potencial para mejorar la precisión en mercados volátiles: En mercados con alta volatilidad y patrones complejos, las LSTM podrían superar a modelos más simples que asumen linealidad o dependencias temporales más cortas.

Diferencias y similitudes clave entre un modelo de regresión lineal y una red neuronal artificial (RNA)

Las Similitudes principales son las siguientes:

* Modelo paramétrico supervisado: Ambos enfoques pertenecen a la categoría de modelos de aprendizaje supervisado y requieren un conjunto de datos etiquetado para su entrenamiento. Además, son modelos paramétricos, es decir, aprenden una serie de parámetros (pesos) que definen la relación entre las variables de entrada y la salida.

* Función de salida basada en una combinación lineal de entradas: En su forma más básica, una neurona artificial realiza una combinación lineal de las entradas ponderadas por sus respectivos pesos, lo cual es conceptualmente equivalente a la formulación de la regresión lineal. En una red neuronal de una sola capa y sin funciones de activación no lineales, el comportamiento del modelo es efectivamente el de una regresión lineal.

* Optimización mediante descenso del gradiente: Ambos modelos pueden entrenarse mediante métodos de optimización basados en descenso del gradiente para minimizar una función de pérdida (por ejemplo, el error cuadrático medio).

Las diferencias principales por otro lado son:

* Capacidad de modelado no lineal: La regresión lineal modela únicamente relaciones lineales entre las variables de entrada y la salida. Por el contrario, las redes neuronales incorporan funciones de activación no lineales (como ReLU, sigmoide, tanh), lo que les permite aproximar funciones no lineales complejas y capturar relaciones de mayor nivel entre las variables.

* Arquitectura del modelo: La regresión lineal consiste en una única combinación lineal de entradas (una sola capa sin no linealidades), mientras que una red neuronal puede tener múltiples capas ocultas (modelo de red neuronal profunda o deep learning), lo que la convierte en un modelo jerárquico capaz de aprender representaciones de alto nivel.

* Capacidad de generalización y expresividad: Las redes neuronales, debido a su arquitectura más compleja, tienen una mayor capacidad de representación (teóricamente, pueden aproximar cualquier función continua bajo ciertas condiciones, como establece el teorema de aproximación universal). Esto las hace más adecuadas para tareas como visión por computador o procesamiento de lenguaje, mientras que la regresión lineal se queda limitada a contextos donde las relaciones sean más simples y lineales.

* Requisitos computacionales y entrenamiento: Entrenar una red neuronal es computacionalmente más costoso y requiere técnicas adicionales como regularización (dropout, L2), optimizadores avanzados (Adam, RMSprop), y puede presentar problemas como el sobreajuste o el estancamiento del gradiente. En cambio, la regresión lineal tiene una solución analítica cerrada en muchos casos, y su entrenamiento es considerablemente más eficiente y estable.

* Interpretabilidad: La regresión lineal ofrece una interpretación directa y clara de los coeficientes como la influencia marginal de cada variable sobre la salida, lo cual es útil para análisis explicativo. Las redes neuronales, por su complejidad, se consideran modelos de caja negra, donde la interpretación de los pesos no es trivial y a menudo se requiere el uso de técnicas específicas de interpretabilidad (SHAP, LIME, saliency maps, etc.).

En la siguiente tabla se presenta una comparación entre el modelo de regresión lineal y una red neuronal artificial, destacando sus principales similitudes y diferencias en términos de capacidad de modelado, arquitectura, interpretabilidad y aplicaciones.

Comparativa entre Red Neuronal Artificial (RNA) y Regresión Lineal

Aspecto	*Regresión Lineal*	*Red Neuronal Artificial (RNA)*
Tipo de modelo	Supervisado, paramétrico	Supervisado, paramétrico
Relación entre variables	Lineal	Puede modelar relaciones no lineales
Arquitectura	Una sola capa (modelo plano)	Múltiples capas (modelo jerárquico, deep learning)
Función de salida	Combinación lineal de entradas	Combinación lineal seguida de funciones de activación no lineales
Capacidad de representación	Limitada a relaciones lineales	Capacidad universal de aproximación de funciones complejas
Método de entrenamiento	Solución analítica (cuando es posible) o descenso del gradiente	Descenso del gradiente con retropropagación, optimización iterativa
Computación y recursos	Requiere pocos recursos computacionales	Requiere mayor capacidad computacional (CPU/GPU)
Interpretabilidad	Alta: coeficientes directamente interpretables	Baja: modelo tipo “caja negra”; requiere técnicas específicas de interpretabilidad
Riesgo de sobreajuste	Bajo en general	Alto si no se regula adecuadamente
Ámbitos de aplicación	Problemas con relaciones lineales claras (predicción simple, análisis explicativo)	Tareas complejas como visión por computador, PLN, clasificación no lineal

2.2.3. Prophet

¿Qué es el modelo de predicción Prophet?

Prophet es un modelo de previsión desarrollado por Facebook (ahora Meta) y diseñado principalmente para series temporales con fuertes patrones de estacionalidad (como patrones diarios, semanales y anuales) y que pueden verse afectadas por eventos irregulares o festivos. A diferencia de algunos modelos tradicionales de series temporales como ARIMA, Prophet adopta un enfoque aditivo donde descompone la serie temporal en varios componentes:

* Tendencia (g(t)): Modela los cambios no periódicos en el valor de la serie temporal. Prophet implementa por defecto una tendencia lineal por partes, pero también permite una tendencia logística para modelar efectos de saturación.
* Estacionalidad (s(t)): Representa los patrones periódicos, como la estacionalidad diaria, semanal y anual. Prophet utiliza series de Fourier para modelar estos efectos, lo que le permite adaptarse a múltiples periodos de estacionalidad simultáneamente.
* Festivos (h(t)): Permite incorporar el impacto de eventos específicos y predecibles (como días festivos, lanzamientos de productos, etc.) que pueden afectar la serie temporal. El usuario debe proporcionar una lista de estos eventos y sus fechas.
* Término de error (Et): Representa el ruido aleatorio que no se explica por los otros componentes del modelo.

¿Para qué sirve Prophet en nuestro caso, enfocado a la predicción de valores de ETFs?

En el contexto específico de la predicción de valores futuros de acciones de ETFs, Prophet puede ser útil por varias razones:

1. Primeramente, este captura la estacionalidad: A que me refiero, aunque los mercados financieros no siempre presentan una estacionalidad tan marcada como, por ejemplo, las ventas minoristas, pueden existir patrones semanales (efectos de fin de semana) o incluso anuales sutiles relacionados con ciclos económicos o comportamiento de inversores. Prophet puede intentar identificar y modelar estos patrones.
2. Por otro lado, ofrece la opción de manejar las tendencias: Los precios de los ETFs obviamente tienen tendencias a largo plazo influenciadas por el rendimiento de los activos subyacentes, las condiciones económicas generales y el sentimiento del mercado. Prophet puede modelar estas tendencias, aunque su capacidad para predecir cambios bruscos en la tendencia (como los causados por eventos inesperados) es limitada.
3. Además, también incorpora en el modelo la variable de los eventos conocidos: Si bien es difícil predecir eventos futuros que impacten significativamente el mercado, Prophet permite incorporar el efecto de eventos pasados conocidos (como anuncios de políticas económicas, resultados empresariales importantes dentro del ETF, etc.) si se dispone de datos sobre su impacto. Esto puede ayudar a mejorar el ajuste del modelo a la historia.
4. Robustez y facilidad del uso de este modelo: Prophet es relativamente robusto ante datos faltantes y cambios en la varianza de la serie temporal. Además, su API en Python y R es bastante intuitiva, lo que facilita su implementación y ajuste, incluso para usuarios con menos experiencia en modelado de series temporales.
5. Otro gran punto a favor de este modelo frente a otros, es, la generación de intervalos de incertidumbre: Prophet proporciona intervalos de confianza para sus predicciones, lo que te da una idea del rango de posibles valores futuros y te ayuda a evaluar la incertidumbre asociada a la predicción.
6. Velocidad y escalabilidad es otro de los factores positivos que te presenta este modelo: Prophet está diseñado para manejar grandes conjuntos de datos y realizar predicciones de manera eficiente, lo cual puede ser útil si estás trabajando con muchos ETFs o con series temporales largas.

7. Y finalmente otra de las grandes virtudes que proporciona el prohet para la predicción de valores es, el manejo que te ofrece frente a los datos faltantes: Prophet puede manejar razonablemente bien los datos faltantes en la serie temporal sin necesidad de imputación previa.
   Sin embargo, es crucial tener en cuenta las limitaciones de Prophet al predecir valores de ETFs:

2.2.4. Modelos de Clasificación

¿Qué es un modelo de clasificación en el contexto de acciones de ETFs?

A diferencia de los modelos de regresión (como Prophet y la regresión lineal) y las redes neuronales recurrentes (como las LSTM) que hemos discutido para predecir valores numéricos futuros, un **modelo de clasificación** tiene como objetivo asignar una observación a una de varias categorías predefinidas.

En el contexto de acciones de ETFs, las tareas de clasificación podrían incluir:

- **Predicción de la dirección del precio:** Clasificar si el precio del ETF subirá o bajará (o se mantendrá relativamente igual) en un horizonte de tiempo específico (por ejemplo, al día siguiente, la próxima semana). Aquí las categorías serían "Subir", "Bajar" y posiblemente "Lateral".
- **Identificación de regímenes de mercado:** Clasificar el estado actual del mercado para un ETF en categorías como "Alcista", "Bajista", "Lateral" o "Volátil".
- **Detección de anomalías:** Clasificar transacciones o patrones de precios como "normales" o "anómalos" (lo que podría indicar oportunidades o riesgos).
- **Evaluación del riesgo:** Clasificar un ETF en categorías de riesgo (por ejemplo, "Bajo", "Medio", "Alto") basándose en sus características y el contexto del mercado.

El modelo de clasificación aprende a partir de datos históricos etiquetados, donde cada observación está asociada con una categoría conocida. Luego, utiliza este conocimiento para predecir la categoría de nuevas observaciones no vistas.

Modelos de Clasificación Comunes Aplicados a ETFs:

Existen muchos algoritmos de clasificación que podrían aplicarse a la predicción o categorización relacionada con ETFs. Algunos ejemplos incluyen:

- **Regresión Logística:** Aunque su nombre incluye "regresión", es un modelo lineal utilizado para problemas de clasificación binaria (dos categorías). Se puede extender para problemas multiclase. Modela la probabilidad de que una observación pertenezca a una categoría específica.
- **Máquinas de Vectores de Soporte (SVM):** Un algoritmo potente que busca encontrar el hiperplano óptimo que mejor separe las diferentes clases en el espacio de características.
- **Árboles de Decisión y Bosques Aleatorios (Random Forests):** Modelos basados en la división jerárquica del espacio de características para tomar decisiones de clasificación. Los bosques aleatorios son un conjunto de árboles de decisión que a menudo ofrecen mejor rendimiento y robustez.
- **Gradient Boosting (por ejemplo, XGBoost, LightGBM):** Algoritmos que construyen modelos de clasificación de forma aditiva, corrigiendo secuencialmente los errores de los modelos anteriores, lo que a menudo resulta en alta precisión.
- **Redes Neuronales Feedforward (Multilayer Perceptrons - MLP):** Redes neuronales básicas con una o más capas ocultas que pueden aprender relaciones complejas entre las características de entrada y las categorías de salida.
- **Redes Neuronales Recurrentes (RNNs) y LSTM para Clasificación:** Si bien las discutimos para la predicción de valores, también se pueden adaptar para tareas de clasificación de series temporales. Por ejemplo, una LSTM podría analizar una secuencia de precios históricos y clasificar si la tendencia general al final de la secuencia es alcista o bajista.

¿Para qué sirven los modelos de clasificación en el contexto de acciones de ETFs?

Los modelos de clasificación pueden ser valiosos en el análisis y la toma de decisiones relacionadas con ETFs de varias maneras:

- **Generación de señales de trading:** Clasificar la dirección probable del precio puede generar señales de compra o venta.
- **Gestión de riesgos:** Clasificar el nivel de riesgo de un ETF puede ayudar a los inversores a construir carteras acordes con su tolerancia al riesgo.
- **Detección de oportunidades:** Identificar regímenes de mercado específicos puede ayudar a los inversores a adaptar sus estrategias (por ejemplo, ser más agresivos en mercados alcistas).
- **Detección de anomalías:** Identificar patrones inusuales podría alertar sobre posibles problemas o ineficiencias del mercado.
- **Automatización de estrategias:** Las decisiones de clasificación pueden integrarse en sistemas de trading algorítmico para automatizar la toma de decisiones.

Puntos fuertes de los modelos de clasificación en el contexto de ETFs:

Los puntos fuertes varían según el modelo específico utilizado, pero en general incluyen:

- **Capacidad para modelar relaciones no lineales:** Muchos modelos de clasificación avanzados (como SVM, redes neuronales, gradient boosting) pueden capturar relaciones complejas y no lineales entre las características de entrada y las categorías de salida, lo que puede ser crucial en los mercados financieros.
- **Flexibilidad en la elección de características:** Se pueden utilizar diversas características como entrada, incluyendo precios históricos, indicadores técnicos, datos de volumen, e incluso datos macroeconómicos, para realizar la clasificación.
- **Generación de decisiones discretas:** La salida de un modelo de clasificación es una decisión clara sobre la categoría a la que pertenece una observación, lo que puede ser más directamente accionable que una predicción de valor continuo.
- **Evaluación con métricas específicas:** Los modelos de clasificación se pueden evaluar utilizando métricas diseñadas para este tipo de tareas, como precisión, recall, precisión (en inglés, precisión), F1-score, y la curva ROC AUC, que proporcionan una visión detallada del rendimiento del modelo.
- **Adaptabilidad a diferentes tareas:** El mismo conjunto de datos se puede utilizar para diferentes tareas de clasificación, dependiendo de cómo se definan las categorías objetivo.

2.3. Arquitecturas y Tecnologías Big Data

2.3.1. Apache Hadoop

El ecosistema Apache Hadoop representa una solución de código abierto fundamental en el mundo del Big Data, proporcionando herramientas para el almacenamiento y procesamiento distribuido de conjuntos de datos a gran escala. Dentro de este ecosistema, el Hadoop Distributed File System (HDFS) destaca como su componente principal de almacenamiento. HDFS es un sistema de archivos diseñado específicamente para almacenar archivos muy grandes de forma fiable a través de clústeres de máquinas estándar. Su arquitectura se basa en la división de los archivos en bloques de gran tamaño que se distribuyen y, crucialmente, se replican en múltiples nodos (DataNodes) del clúster. Esta replicación (cuyo factor es configurable) es la clave de su alta tolerancia a fallos, ya que asegura que la pérdida de un nodo individual no resulte en la pérdida de datos.

HDFS por sí mismo cumple un rol esencial como repositorio de datos masivo, escalable y relativamente económico. En muchas arquitecturas Big Data, HDFS actúa como el "Data Lake" o almacén central donde se depositan grandes cantidades de datos brutos o semi-procesados para su conservación a largo plazo y para servir como fuente para análisis posteriores o entrenamientos de modelos complejos

2.3.2. Apache Cassandra

Apache Cassandra es un sistema de gestión de bases de datos NoSQL distribuido y de código abierto, diseñado específicamente para manejar grandes volúmenes de datos con altos requisitos de disponibilidad y escalabilidad, operando sobre clústeres de hardware estándar. Su arquitectura distribuida, sin un único punto de fallo (peer-to-peer), y su mecanismo de replicación de datos le confieren una robusta tolerancia a fallos y la capacidad de mantener la alta disponibilidad del servicio incluso si fallan nodos individuales en el clúster.

Estas capacidades hacen que Cassandra sea particularmente valiosa en el sector financiero y en arquitecturas Big Data para casos de uso como:

- Almacenamiento y consulta eficiente de datos de series temporales: Ideal para precios de mercado, logs de transacciones, y otros datos indexados por tiempo donde se necesita tanto una ingesta rápida como un acceso veloz a rangos específicos
- Servir como base de datos operacional "caliente": Actuando como el almacén para los datos de acceso más frecuente o reciente, que alimentan aplicaciones, dashboards o procesos de predicción que requieren respuestas rápidas

2.3.3. Apache NiFi

Apache NiFi es una plataforma de software de código abierto, integrada en el ecosistema de la Apache Software Foundation, diseñada específicamente para automatizar y gestionar el flujo de datos (DataFlow) entre sistemas heterogéneos de forma fiable y escalable. Su principal característica distintiva es su interfaz gráfica de usuario basada en web, que permite a los usuarios diseñar visualmente complejos pipelines de datos mediante la conexión de "Procesadores". Estos procesadores son componentes modulares que realizan tareas específicas como la ingesta de datos desde diversas fuentes (ej., monitorizando directorios con GetFile o consumiendo APIs), la transformación de datos en ruta (ej., modificando esquemas o convirtiendo tipos de datos con UpdateRecord), y el enrutamiento inteligente de la información hacia uno o múltiples destinos (ej., PutHDFS, PutCassandraQL).

Por estas razones, NiFi se posiciona como un orquestador y preparador de datos muy eficaz en arquitecturas Big Data, especialmente en casos de uso que requieren:

- La integración de datos provenientes de múltiples fuentes con diferentes formatos o protocolos.
- La aplicación de transformaciones o limpiezas preliminares antes de que los datos lleguen a los sistemas de almacenamiento o análisis.
- La distribución fiable de datos a diferentes sistemas (ej., un data lake en HDFS para histórico y una base de datos NoSQL como Cassandra para acceso rápido).

2.3.4. Docker

Docker es una plataforma de código abierto que ha revolucionado la forma en que se desarrollan, distribuyen y ejecutan las aplicaciones mediante el uso de la tecnología de contenerización. La idea central es empaquetar una aplicación o servicio, junto con todas sus dependencias (librerías, binarios, archivos de configuración, etc.), en una unidad estandarizada y aislada denominada contenedor. Estos contenedores se crean a partir de imágenes (plantillas de solo lectura) y se ejecutan de manera consistente en cualquier máquina que tenga Docker instalado, independientemente del sistema operativo subyacente o de otras aplicaciones que se estén ejecutando. Para gestionar aplicaciones que constan de múltiples servicios interconectados (como una arquitectura Big Data), herramientas como Docker Compose permiten definir y ejecutar aplicaciones multi-contenedor a partir de un único archivo de configuración (docker-compose.yml).

Docker se ha convertido en una herramienta prácticamente indispensable en el desarrollo y despliegue de proyectos complejos de Big Data e Inteligencia Artificial por varias razones clave:

- Simplificación de la Configuración y el Despliegue: Montar entornos que involucran múltiples componentes de Big Data (como pueden ser clústeres de Hadoop, bases de datos NoSQL como Cassandra, herramientas de flujo como NiFi, etc.) puede ser un proceso manual complejo y propenso a errores. Docker y Docker Compose simplifican radicalmente este proceso, permitiendo levantar una arquitectura completa con todos sus servicios interconectados mediante unos pocos comandos
- Aislamiento de Servicios: Cada contenedor se ejecuta en su propio entorno aislado, con sus propios recursos (sistema de archivos, red, procesos), evitando conflictos de dependencias o puertos entre diferentes servicios que se ejecutan en la misma máquina host.
- Portabilidad: Las imágenes de Docker pueden construirse en una máquina y ejecutarse en cualquier otra (local, servidor, nube) que soporte Docker, facilitando la migración entre entornos.

2.4. Herramientas de Análisis y Visualización

2.4.1. Pandas

2.4.2. Power BI

Microsoft Power BI es un servicio de análisis de negocios que forma parte de la plataforma Power Platform de Microsoft, diseñado para proporcionar capacidades de inteligencia empresarial (Business Intelligence) y visualizaciones de datos interactivas. Power BI ofrece herramientas para la transformación y modelado de los mismos (a través de Power Query y DAX - Data Analysis Expressions), aunque su principal fortaleza reconocida radica en la creación de informes y dashboards visualmente atractivos y dinámicos.

La plataforma se destaca por su interfaz intuitiva que facilita la creación de una amplia gama de objetos visuales, como gráficos de líneas, barras, áreas, mapas, tablas, matrices y tarjetas de indicadores clave de rendimiento (KPIs).

2.4.3. Orange Data Mining

Orange Data Mining es una potente plataforma de software de código abierto, desarrollada en Python, que facilita el análisis de datos, la minería de datos y el aprendizaje automático (*Machine Learning*). Los usuarios construyen análisis complejos mediante la conexión de componentes visuales, conocidos como "widgets", en un lienzo de flujo de trabajo (*workflow*). Cada widget encapsula una operación específica, abarcando el ciclo de vida completo del análisis de datos: desde la carga de datos (*widget File*), la exploración visual interactiva (*widgets como Distributions, Correlations, Scatter Plot*), el preprocesamiento de datos (*incluyendo selección de características con Select Columns o transformaciones como codificación one-hot y escalado con Continuize*), la aplicación de una amplia gama de algoritmos de modelado (*regresión como Linear Regression, Tree, Random Forest, Gradient Boosting, así como clasificación y clustering*), hasta la evaluación rigurosa de modelos (*widget Test & Score con opciones como validación cruzada*) y el análisis detallado de resultados (*visualización de predicciones, importancia de variables con Rank, o inspección de modelos como con Tree Viewer*).

Las principales ventajas de este enfoque visual e integrado, especialmente relevantes en el ámbito del análisis de datos y la IA, incluyen:

- Accesibilidad y Curva de Aprendizaje: Facilita la comprensión y aplicación de técnicas complejas tanto para usuarios nuevos como para expertos que buscan rapidez.
- Rapidez de Prototipado y Experimentación: Permite construir, modificar y comparar diferentes flujos de trabajo de preprocesamiento y modelado de forma muy ágil, conectando y desconectando widgets.
- Interactividad y Comprensión: La capacidad de inspeccionar los datos o los resultados intermedios en casi cualquier punto del flujo de trabajo fomenta una comprensión más profunda del proceso y de los datos mismo

3. Metodología y Diseño del Sistema

3.1. Adquisición de Datos

3.1.1. Fuente de Datos: Kaggle

Una de las fuentes primarias de datos para nuestro proyecto fue la plataforma Kaggle. Específicamente, se utilizó el conjunto de datos "Mutual Funds and ETFs", aportado por el usuario Stefano Leone (stefanoleone992). Dentro de este conjunto de datos más amplio, el análisis y las predicciones desarrolladas en este trabajo se centran exclusivamente en la información contenida en el archivo ETF prices.csv.

El archivo ETF prices.csv contiene precios históricos diarios y volumen de negociación para una extensa colección de Fondos Cotizados en Bolsa (ETFs). Si bien el dataset general también incluye fondos mutuos, este archivo se enfoca específicamente en ETFs que, dado el contexto del conjunto de datos, están predominantemente listados y negociados en mercados de Estados Unidos (EEUU).

Las columnas principales identificadas y utilizadas de este archivo para el análisis son las siguientes:

- **fund_symbol**: Símbolo o ticker del ETF, que actúa como identificador único en el mercado.
- **price_date**: Fecha específica, en formato *AAAA-MM-DD*, a la que corresponden los datos de la fila.
- **open**: Precio de apertura del ETF en la jornada bursátil de *price_date*.
- **high**: Precio máximo alcanzado por el ETF durante la sesión de *price_date*.
- **low**: Precio mínimo alcanzado por el ETF durante la sesión de *price_date*.
- **close**: Precio del ETF al cierre de la jornada bursátil de *price_date*.
- **adj_close**: Precio de cierre ajustado. Este valor es fundamental para el análisis de rendimientos, ya que se corrige para reflejar eventos corporativos como el pago de dividendos y desdoblamientos de acciones (splits), ofreciendo una representación más precisa de la rentabilidad real de la inversión.
- **volume**: Número total de acciones del ETF negociadas durante la sesión en price_date, siendo un indicador clave de la liquidez del activo.

Los datos contenidos en el archivo ETF prices.csv cubren un extenso periodo histórico, abarcando desde el 29 de enero de 1993 hasta el 27 de marzo de 2024. Es relevante destacar que no todos los ETFs presentan datos para la totalidad de este intervalo, ya que la disponibilidad de su historial depende de la fecha de creación o liquidación de cada fondo específico.

La obtención del archivo ETF prices.csv se realizó mediante la descarga manual directa desde la página del dataset "Mutual Funds and ETFs" en la plataforma Kaggle, accesible en https://www.kaggle.com/datasets/stefanoleone992/mutual-funds-and-etfs/data.

Una observación inicial del archivo ETF prices.csv revela su considerable tamaño, con aproximadamente
1.295.490 millones de filas (registros) según la información proporcionada en Kaggle. Este volumen de datos sugiere la inclusión de un número significativo de ETFs diferentes.

Asimismo, la variabilidad en la longitud de las series temporales históricas para cada ETF fue una característica anticipada y considerada para la posterior fase de preprocesamiento y selección de datos para el modelado. También se tuvo en cuenta la posible presencia de valores ausentes (NaNs) en algunas columnas, un aspecto común en datos financieros que requiere un tratamiento específico.

3.1.2. Fuente de Datos: API de Yahoo Finance

Tras una revisión y filtrado inicial de nuestro conjunto de datos, y después de realizar las primeras pruebas con nuestros modelos de predicción, identificamos varias limitaciones significativas. Principalmente, observamos que:

1. Los registros históricos de los diferentes activos no comenzaban en la misma fecha, lo que dificultaba un análisis comparativo homogéneo.
2. Algunos ETFs presentaban valores ausentes (NaN) en sus series temporales.
3. Crucialmente, los datos disponibles no eran actuales, con una fecha de corte el **30 de noviembre de 2021**.

Estos defectos en nuestro dataset, comprometían la actualidad y la veracidad de la información, afectando directamente la potencial precisión y relevancia de nuestros modelos predictivos.

Es por eso que para subsanar estas limitaciones y asegurar la robustez de nuestro análisis, procedimos a actualizar y enriquecer nuestro conjunto de datos. Decidimos emplear la **API de Yahoo Finance**, una fuente reconocida y accesible para obtener datos financieros históricos y actualizados. El objetivo era adquirir información reciente para los ETFs que habíamos preseleccionado como más prometedores para nuestro estudio de predicción, asegurando además la consistencia en el rango temporal.

A continuación, os mostramos el código en Python utilizado para interactuar con la API de Yahoo Finance y descargar los datos necesarios:

```
import yfinance as yf
import pandas as pd
from datetime import datetime

# --- Configuración ---
# Lista de Tickers de los ETFs
tickers = ['SPY', 'QLD', 'RXL', 'USD', 'ROM', 'UCC', 'TNA',
'TECL', 'MIDU', 'UPRO', 'DRN',
               'UDOW', 'TQQQ', 'UMDD', 'SOXL', 'RETL', 'CURE',
'SMH', 'FRAK', 'AMER']

# Fechas de inicio y fin
start_date = "2021-11-01"
end_date = "2025-03-01"

# --- Descarga de Datos ---
```

```
print(f"Descargando datos para: {', '.join(tickers)}")
print(f"Rango de fechas: {start_date} a {end_date}")

data = pd.DataFrame() # Inicializar como DataFrame vacío
try:
    # Descargar los datos históricos
    data = yf.download(tickers, start=start_date, end=end_date,
progress=True)

    # --- Verificaciones Robustas ---
    if data.empty:
        print("\nError: No se pudieron descargar datos. El
DataFrame está vacío.")
        print("Verifica los tickers, las fechas o tu conexión a
internet.")
    # yfinance para múltiples tickers devuelve un MultiIndex en
las columnas.

    # Verificamos si 'Adj Close' está en el primer nivel de las
columnas.
    elif 'Adj Close' not in data.columns.get_level_values(0):
         print("\nError: La columna 'Adj Close' no se encontró en
los datos descargados.")
         print("La estructura de datos devuelta puede ser
inesperada.")
         print("Columnas disponibles:", data.columns)
         # Puedes descomentar la siguiente línea para ver todo el
dataframe devuelto y depurar
         # print("\nDataFrame completo devuelto por yfinance:\n",
data)
    else:
        # --- Extracción de 'Adj Close' ---
        # Ahora es seguro acceder a 'Adj Close'
        adj_close_data = data['Adj Close']

        # Eliminar filas donde *todos* los valores son NaN
        adj_close_data = adj_close_data.dropna(how='all')

        # Verificar si quedó algo después de eliminar NaNs
        if adj_close_data.empty:
            print("\nAdvertencia: Después de eliminar filas sin
datos, el DataFrame resultante está vacío.")
        else:
            # --- Mostrar Resultados ---
```

```
            print("\n--- Datos de Precios de Cierre Ajustados
('Adj Close') ---")


except Exception as e:
    # Captura otros posibles errores (conexión, etc.)
    print(f"\nOcurrió un error inesperado durante la descarga o
procesamiento: {e}")

print("\n--- Proceso Finalizado ---")
```

3.2. Preprocesamiento y Análisis Exploratorio de Datos (EDA)

En este apartado de la memoria, el objetivo principal es el de explicar cómo hemos realizado la limpieza y preparación de los datos de los 20 ETFs más importantes de EE. UU. Para los análisis subsecuentes, y al mismo tiempo, extraer conocimientos iniciales sobre sus características, distribuciones y relaciones mediante técnicas de análisis exploratorio.

3.2.1. Entorno y Herramientas de Preprocesamiento

Para llevar a cabo el preprocesamiento y el EDA, se ha utilizado un entorno de desarrollo basado en **Python**, ejecutado a través de un **Jupyter Notebook**. Las principales bibliotecas de Python empleadas son:

- **Pandas:** Fundamental para la manipulación y análisis de datos, especialmente para trabajar con DataFrames (estructuras de datos tabulares). Se usa para cargar el dataset, inspeccionar datos, limpiar, transformar y realizar agregaciones.
- **NumPy:** Para operaciones numéricas eficientes, especialmente con arrays. Aunque Pandas se basa en NumPy, a veces se utiliza directamente para cálculos específicos.
- **Matplotlib:** Biblioteca base para la creación de visualizaciones estáticas, animadas e interactivas en Python.
- **Seaborn:** Construida sobre Matplotlib, proporciona una interfaz de alto nivel para dibujar gráficos estadísticos atractivos e informativos. Es especialmente útil para visualizar distribuciones, relaciones y comparaciones.
- **Datetime (módulo de Python):** Para trabajar con fechas y horas, crucial para la columna **price_date**.

El flujo de trabajo se centra en cargar el archivo **ETF_cohortes.csv** en un DataFrame de Pandas y luego aplicar secuencialmente diversas operaciones de limpieza y análisis.

3.2.2. Proceso de Limpieza de Datos

La limpieza de datos es un paso crítico para asegurar la calidad y fiabilidad del análisis.

Carga de Datos:

Se carga el archivo **ETF_cohortes.csv** en un DataFrame de Pandas. Se realiza una inspección inicial utilizando **data.head(), data.info(), data.shape** y **data.isnull().sum()** para entender la estructura, los tipos de datos, el volumen de información y la presencia de valores ausentes.

Manejo de Tipos de Datos:

La columna **price_date**, inicialmente cargada como tipo **objet** (cadena de texto), se convierte correctamente al tipo **datetime** utilizando **pd.to_datetime(data['price_date'])**. Esto es esencial para cualquier análisis basado en series temporales.

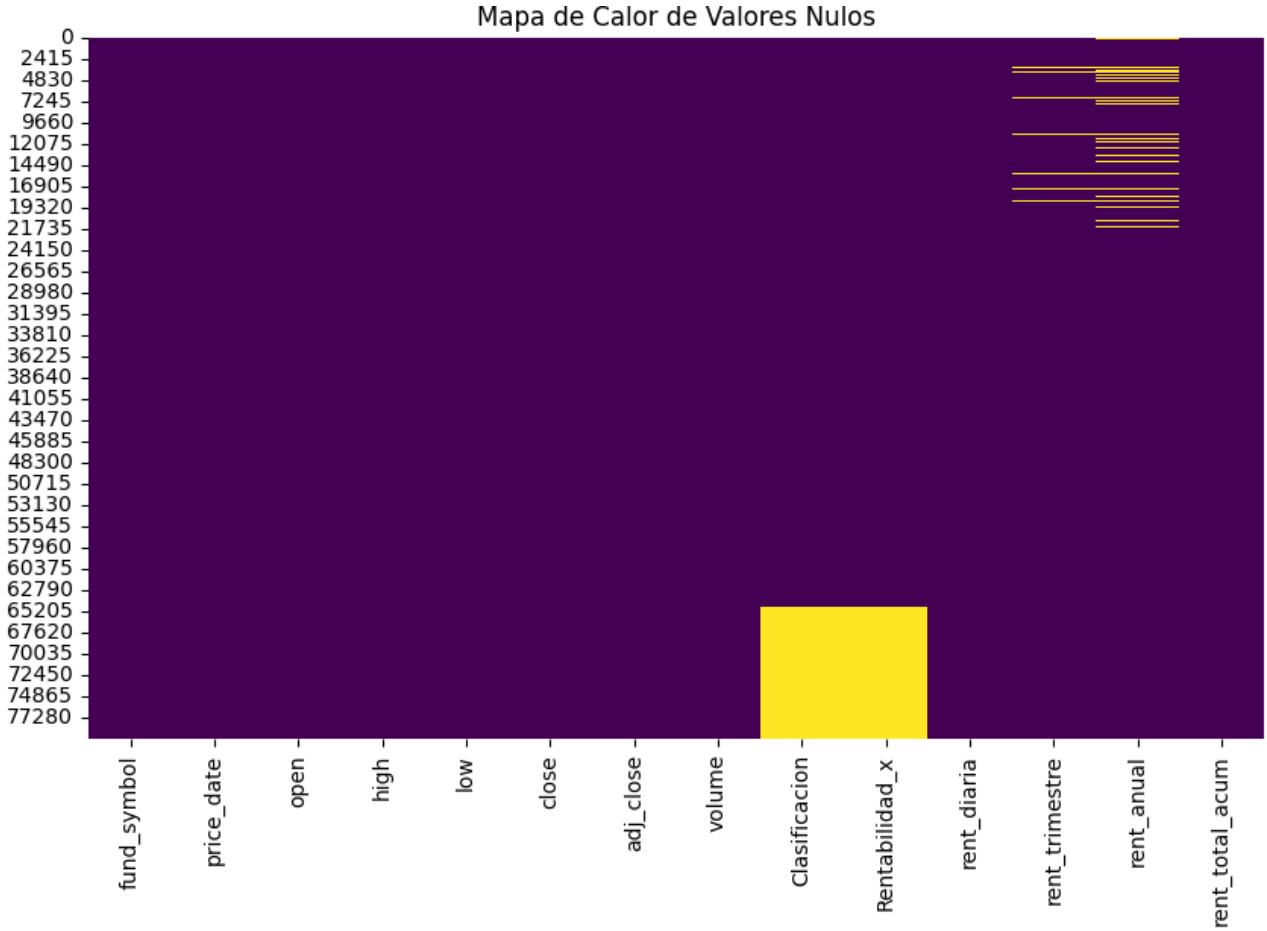
Las columnas **fund_symbol** y **Clasificación** se convierten al tipo **category** usando **astype('category')**. Esto puede optimizar el uso de memoria y es semánticamente correcto, ya que representan variables categóricas con un conjunto limitado de valores.

Manejo de Valores Ausentes (NaN):

Se identifica que las columnas **Clasificación, Rentabilidad_x, rent_diaria, rent_trimestre, rent_anual,** y **rent_total_acum** para revisar si contienen valores nulos.

Se creó un nuevo DataFrame, **df_limpio**, eliminando las filas donde la columna Clasificación contenía valores nulos. Esta operación resultó en la exclusión de dichas filas del conjunto de datos.

Las columnas de rentabilidad (Rentabilidad_x, rent_diaria, rent_trimestre, rent_anual y rent_total_acum) aún presentan valores nulos tras la eliminación basada en la columna Clasificación. Estos valores ausentes en las rentabilidades podrían deberse a la falta de datos históricos necesarios para realizar los cálculos en los periodos iniciales. Es necesario considerar estos valores nulos para análisis posteriores o aplicar técnicas de imputación según los requerimientos del modelo.



Extracción de Componentes de Fecha:

Se crean nuevas columnas para el mes (**month**) y el año (**year**) extrayéndolas de la columna **price_date** ya convertida. Esto facilita análisis agrupados por periodos temporales.

Inspección Final Post-Limpieza:

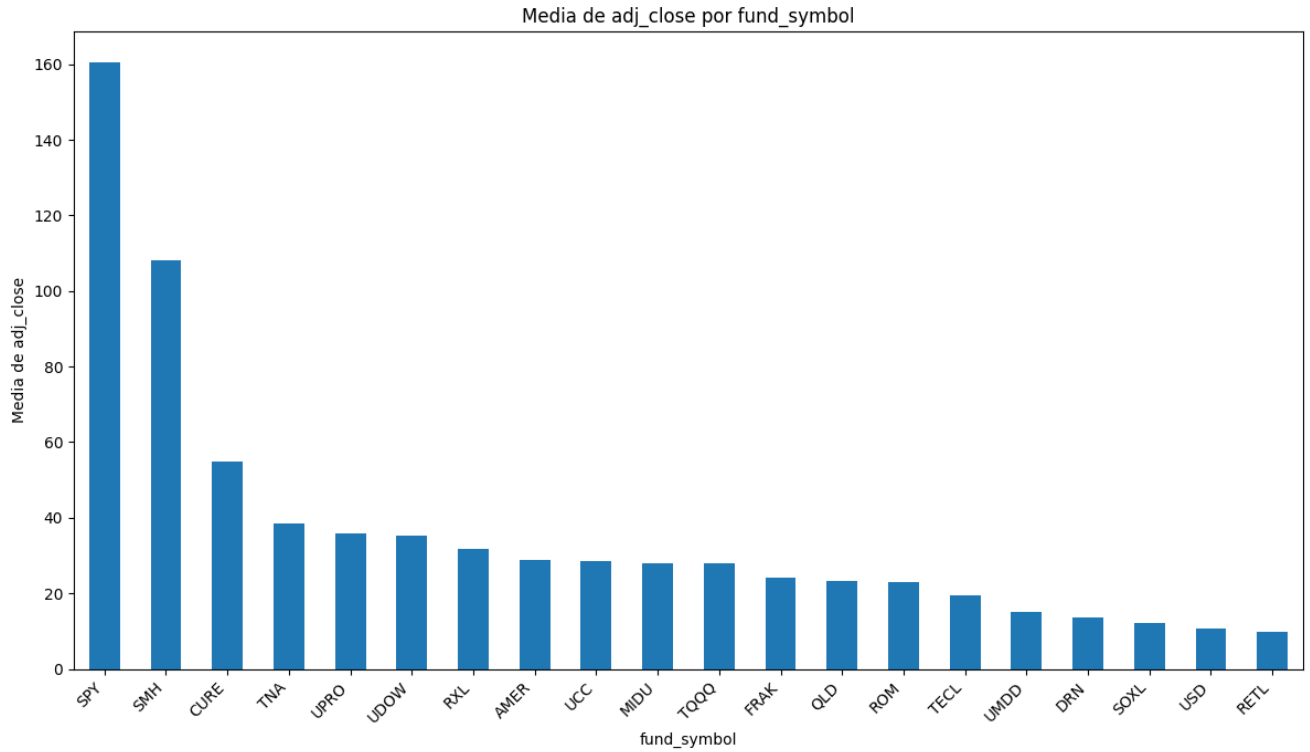
Se vuelve a verificar la información del DataFrame limpio **(df_limpio.info(), df_limpio.isnull().sum())** para confirmar los cambios y el estado actual de los valores ausentes.

3.2.3. Análisis Exploratorio de Datos (EDA)

Una vez que los datos están relativamente limpios, se procede a explorarlos para descubrir patrones, tendencias, anomalías y relaciones. Nuestro EDA abarca:

Estadísticas Descriptivas:

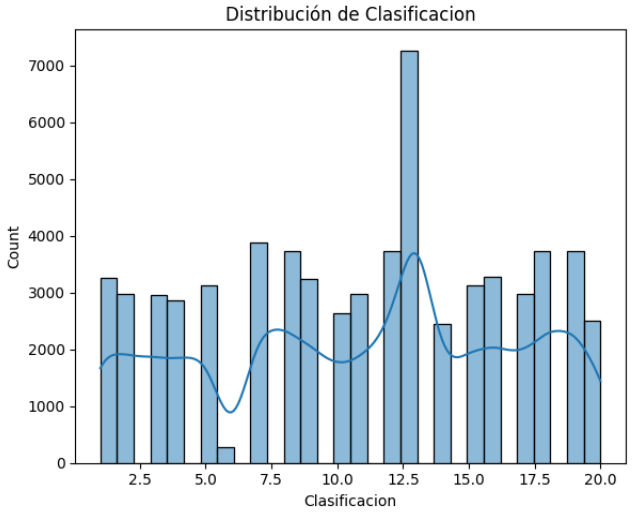
Se utiliza **df_limpio.describe() y df_limpio.describe(include='category')** para obtener un resumen estadístico de las columnas numéricas (media, desviación estándar, mínimo, máximo, cuartiles) y categóricas (conteos, valores únicos, frecuencia del más común). Esto proporciona una visión general de la distribución y escala de cada variable.

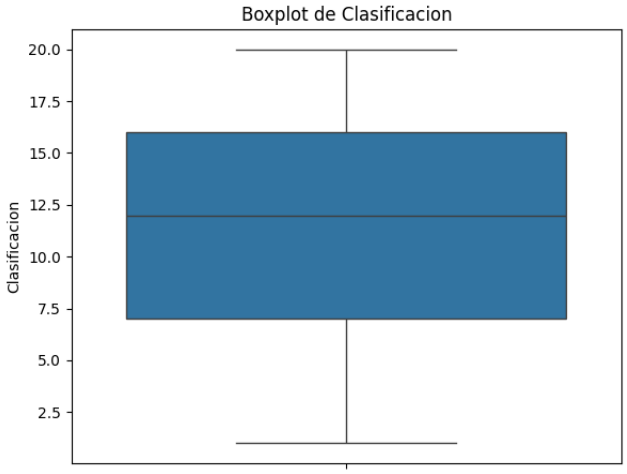


Análisis de Distribuciones:

Variables Numéricas: Se generan histogramas **(.hist())** para visualizar la distribución de las rentabilidades **(rent_diaria, rent_trimestre, rent_anual, rent_total_acum)**. Esto ayuda a entender si las rentabilidades siguen una distribución normal, si están sesgadas, o si presentan colas pesadas (eventos extremos).

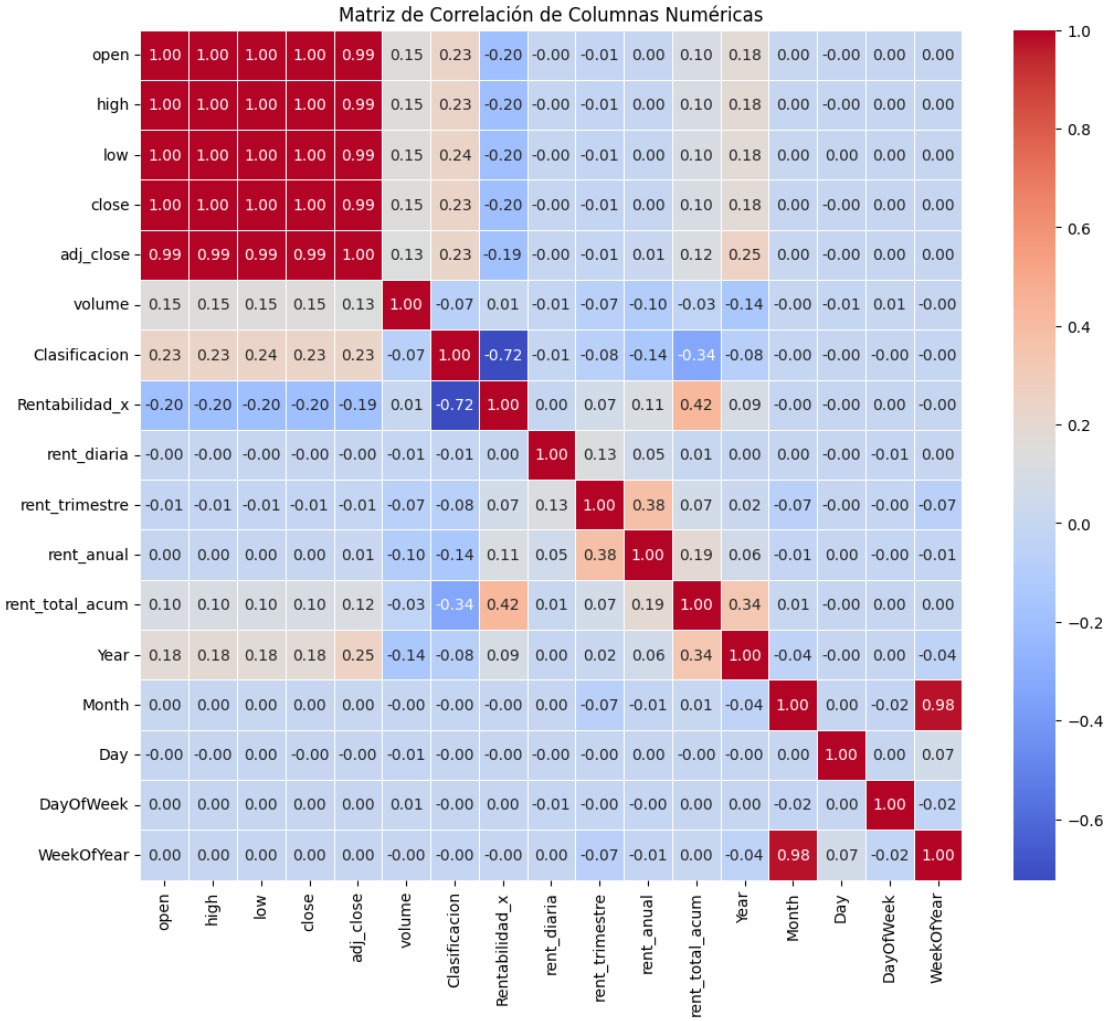
Variables Categóricas: Se utilizan gráficos de conteo **(seaborn.countplot())** para **fund_symbol y Clasificación** para visualizar la frecuencia de cada categoría. Esto muestra cuántos registros hay por ETF y por clasificación, y si hay desbalance entre clases.





Análisis de Correlación:

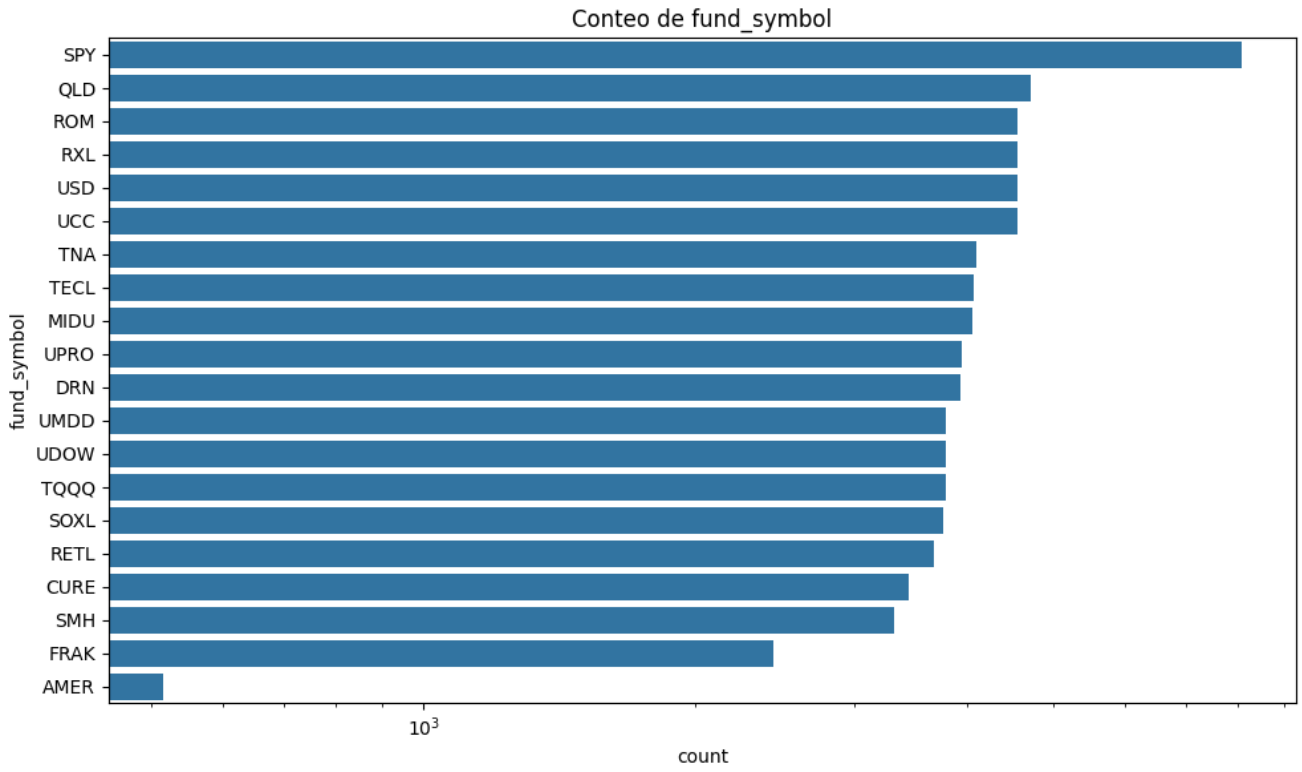
Se calcula la matriz de correlación para las variables numéricas **(df_limpio.corr(numeric_only=True)).** Esta matriz se visualiza mediante un mapa de calor **(seaborn.heatmap())**, es la imagen que hemos adjuntado anteriormente.



Análisis de Series Temporales:

Precios Ajustados (adj_close): Se realizan gráficos de línea de **adj_close** a lo largo del tiempo (**price_date**) para ETFs específicos (ej. 'SPY') y para múltiples ETFs agrupados por **Clasificación**. Esto permite observar tendencias a largo plazo, volatilidad y comparar el rendimiento visual entre diferentes activos o categorías.

Volumen (volume): Similarmente, se visualiza el volumen de negociación a lo largo del tiempo, lo que puede indicar periodos de alta o baja actividad del mercado.



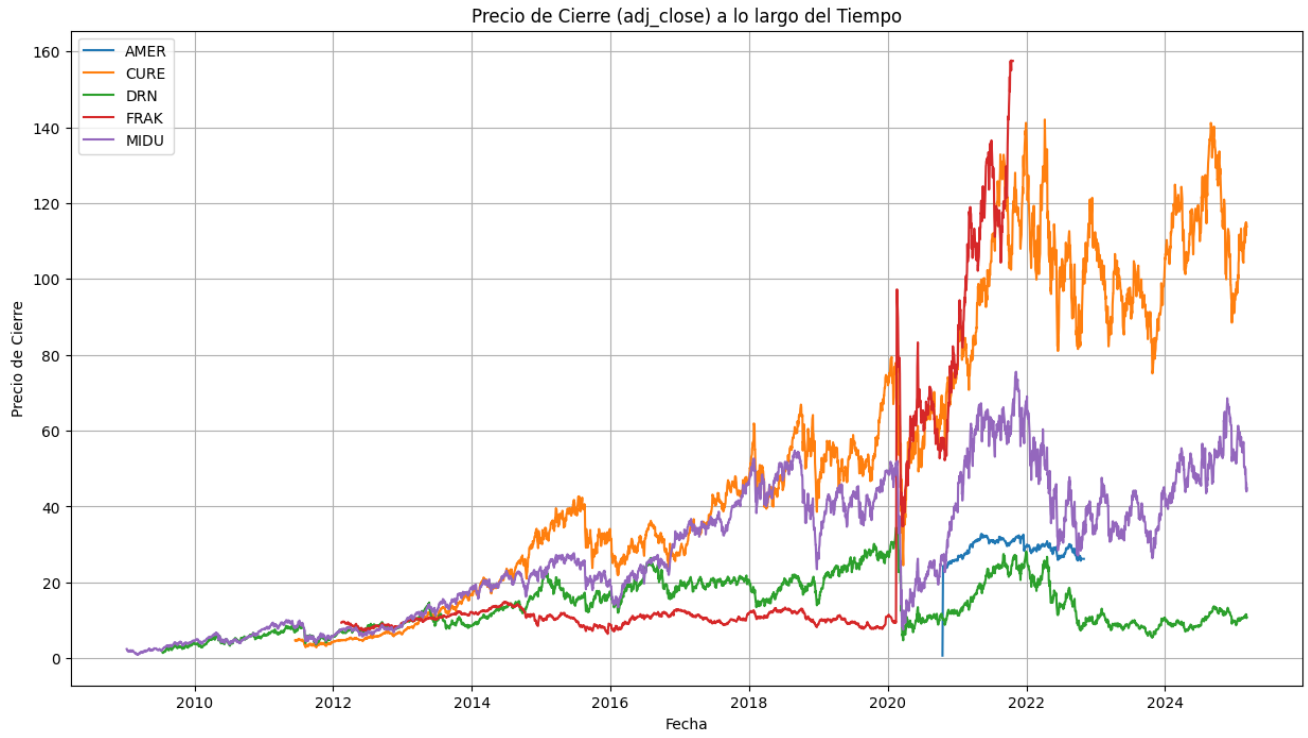
Análisis Comparativo por Categorías:

Box Plots (Diagramas de Caja): Se utilizan extensivamente para comparar la distribución de variables numéricas (como **adj_close, rent_diaria, rent_trimestre, rent_anual**) entre diferentes **Clasificación** de ETFs. Los box plots son excelentes para visualizar medianas, rangos intercuartílicos y posibles outliers por categoría. Por ejemplo, se compara la rentabilidad diaria media entre las distintas clasificaciones.

Agregaciones: Se realizan cálculos agrupados **(.groupby())** para obtener métricas como la media de **rent_diaria** por **Clasificación**, y luego se visualizan estos resultados (ej. con **plot(kind='bar')**).

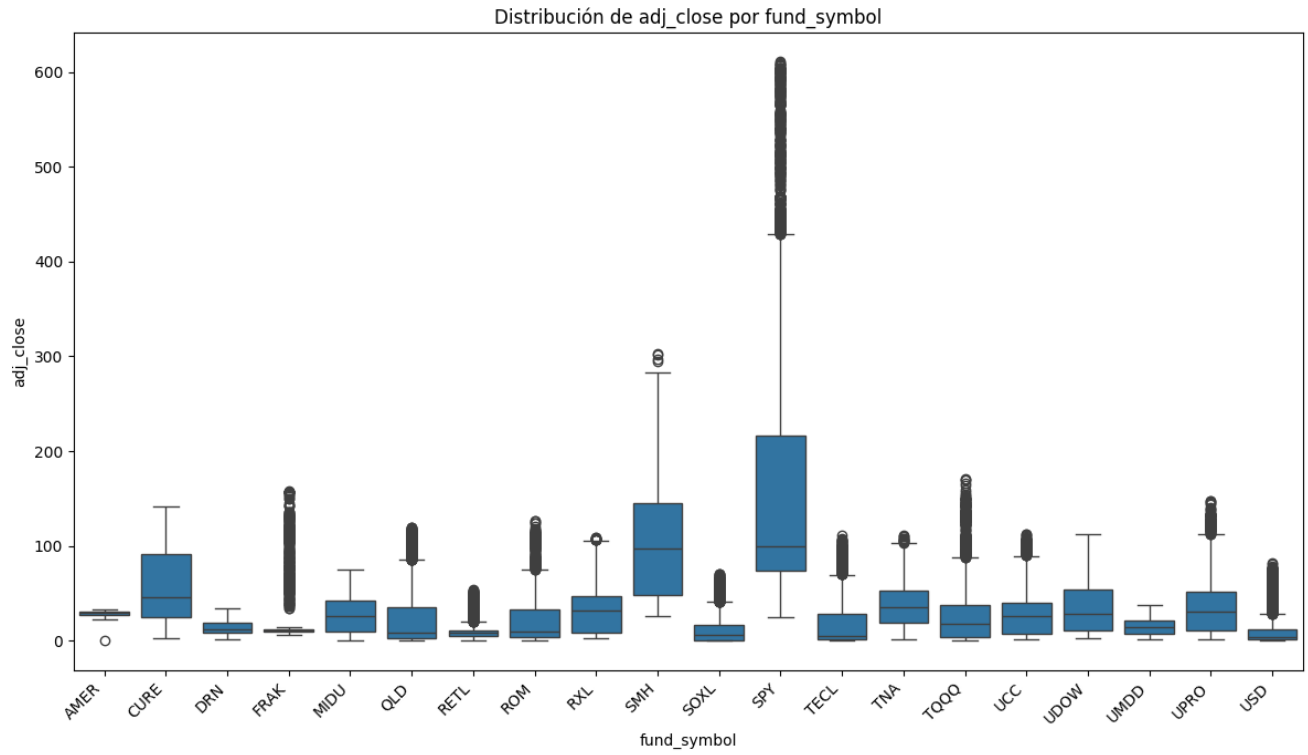
Análisis Específico por ETF:

Se filtran datos para ETFs individuales (como 'SPY') para realizar análisis más detallados de su comportamiento histórico en precios y volumen.



Identificación de Outliers (Visual):

A través de los box plots e histogramas, se pueden identificar visualmente valores atípicos en las rentabilidades o precios, que podrían requerir una investigación más profunda o un tratamiento especial en fases posteriores de modelado.



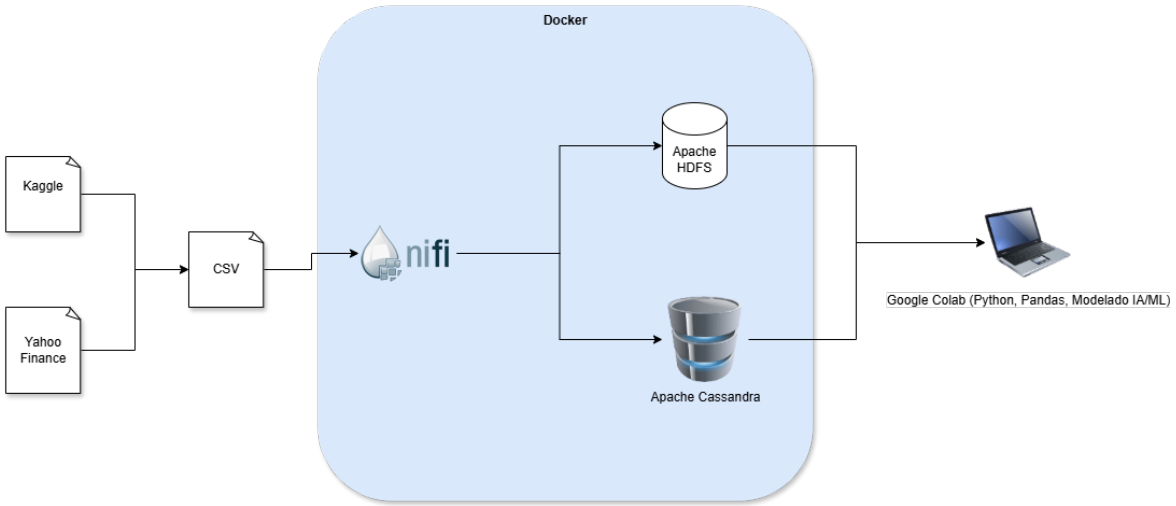
3.3. Diseño e Implementación de la Arquitectura Big Data

3.3.1. Diagrama de Arquitectura General del Sistema

La arquitectura general del sistema diseñado e implementado para el análisis y predicción de ETFs. El flujo de datos comienza con la obtención de información histórica de ETFs desde dos fuentes principales: un conjunto de datos en formato CSV proveniente de la plataforma Kaggle y datos actualizados obtenidos mediante la API de Yahoo Finance. Estos conjuntos de datos son consolidados y unificados en un único dataset en formato CSV, el cual se almacena en una carpeta designada del sistema de archivos.

Posteriormente, Apache NiFi monitoriza esta carpeta para ingerir automáticamente el archivo CSV unificado. Una vez ingerido, NiFi se encarga de aplicar las transformaciones necesarias a los datos y de distribuirlos a los dos sistemas de almacenamiento principales de la arquitectura: Apache HDFS, que actúa como el repositorio histórico masivo, y Apache Cassandra, que funciona como la base de datos NoSQL operacional para acceso rápido.

Los datos almacenados en HDFS y Cassandra son luego accedidos y utilizados por un entorno de análisis y modelado basado en Google Colab, donde se desarrollan, entrenan y evalúan los modelos de predicción de Inteligencia Artificial. Finalmente, los resultados de estos modelos, así como los análisis exploratorios, se visualizan e investigan de forma interactiva mediante las herramientas Power BI y Orange Datamining. Es importante destacar que los componentes clave de la infraestructura de datos (NiFi, HDFS y Cassandra) se gestionan y ejecutan como servicios contenerizados a través de Docker y Docker Compose, asegurando la portabilidad y reproducibilidad del entorno.



3.3.2. Contenerización con Docker

Para la gestión del entorno de desarrollo y el despliegue de los diversos servicios que componen la arquitectura Big Data de este proyecto, se adoptó la tecnología de contenerización mediante Docker, con la orquestación de los servicios multi-contenedor gestionada a través de Docker Compose. Esta elección se fundamentó en la necesidad de simplificar la instalación, asegurar el aislamiento de los servicios, y garantizar la reproducibilidad y portabilidad del entorno completo.

El sistema se definió mediante un archivo **docker-compose.yml** (**Archivo Docker**), el cual describe los siguientes servicios principales, sus configuraciones, dependencias y mecanismos para la persistencia de datos:

- Red Personalizada (*etf-net*): Se definió una red de tipo *bridge* denominada *etf-net*

para permitir la comunicación interna entre todos los contenedores de la arquitectura de forma aislada y controlada.

- Servicio Apache Cassandra (*cassandra*):
    - Se utilizó la imagen oficial *cassandra:4.1*
    - Se expuso el puerto *9042* para la conexión mediante CQLSH y desde Apache NiFi.
    - Se configuró un volumen *cassandra_data* para persistir los datos de la base de datos en */var/lib/cassandra* dentro del contenedor, asegurando que la información no se pierda al detener o reiniciar el servicio.
    - Se establecieron variables de entorno para la configuración del clúster, como *CASSANDRA_CLUSTER_NAME=ETFCluster*.
    - Se incluyó un *healthcheck* para verificar la operatividad del servicio antes de que otros servicios dependientes iniciaran.
- Servicios HDFS (*Hadoop Distributed File System*):
    - NameNode (*namenode*): Se empleó la imagen *bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8*. Se mapeó un volumen (*hdfs_namenode*) para la persistencia de los metadatos del sistema de archivos en */hadoop/dfs/name*. Se configuraron variables de entorno cruciales como *CORE_CONF_fs_defaultFS=hdfs://namenode:9000* (definiendo el URI del sistema de archivos) y *HDFS_CONF_dfs_replication=2* (indicando la replicación de datos en dos DataNodes). Se expusieron los puertos *9870* (para la interfaz web del NameNode) y *9000* (para el acceso al HDFS).
- DataNodes (*datanode1*, *datanode2*): Se utilizaron dos instancias de DataNode con la imagen *bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8*. Cada DataNode se configuró con su propio volumen persistente (*hdfs_datanode1*, *hdfs_datanode2*) para almacenar los bloques de datos en */hadoop/dfs/data*. Se establecieron dependencias (*depends_on*) para asegurar que los DataNodes iniciaran después de que el NameNode estuviera saludable, y se configuró la variable *SERVICE_PRECONDITION* para esperar la disponibilidad de la UI del NameNode.
- Servicio Apache NiFi (*nifi*):
- Se utilizó la imagen apache/nifi:1.28.0.
- Se expuso el puerto *8443* para el acceso a la interfaz web de NiFi (HTTPS).
- Se configuró para ejecutarse en modo standalone (*NIFI_CLUSTER_IS_NODE=false*) y sin dependencia de Zookeeper (*NIFI_ZK_CONNECT_STRING=*).
- Se definieron múltiples volúmenes (*nifi_conf*, *nifi_database_repo*, *nifi_flowfile_repo*, *nifi_content_repo*, *nifi_provenance_repo*, *nifi_state*, *nifi_logs*) para asegurar la persistencia de toda la configuración, los flujos, los repositorios de datos y los logs de NiFi.
- Crucialmente, se mapearon dos volúmenes desde el sistema anfitrión (host) hacia el contenedor de NiFi:
    - *./etf_data_input:/data_in*: Para permitir a NiFi acceder al archivo CSV de entrada (*ETF_cohortes_final_preprocesado*.csv) ubicado en el directorio *etf_data_input* del host.
    - *./config/hadoop/core-site.xml:/opt/nifi/nifi-current/conf/core-site.xml*: Para proporcionar a NiFi la configuración necesaria para conectarse a HDFS.
- Se establecieron dependencias para que NiFi iniciará después de que los servicios de Cassandra y HDFS (NameNode y DataNodes) estuvieran operativos y saludables.

3.3.3. Flujo de Datos con Apache NiFi

Apache NiFi fue la herramienta seleccionada para la orquestación del flujo de datos (ETL - Extract, Transform, Load) en este proyecto. Su función principal fue ingestar el archivo CSV preprocesado, transformar su contenido al formato Apache Avro para una mayor eficiencia y tipado de datos, y finalmente distribuir estos datos en paralelo tanto al sistema de almacenamiento HDFS como a la base de datos Apache Cassandra.

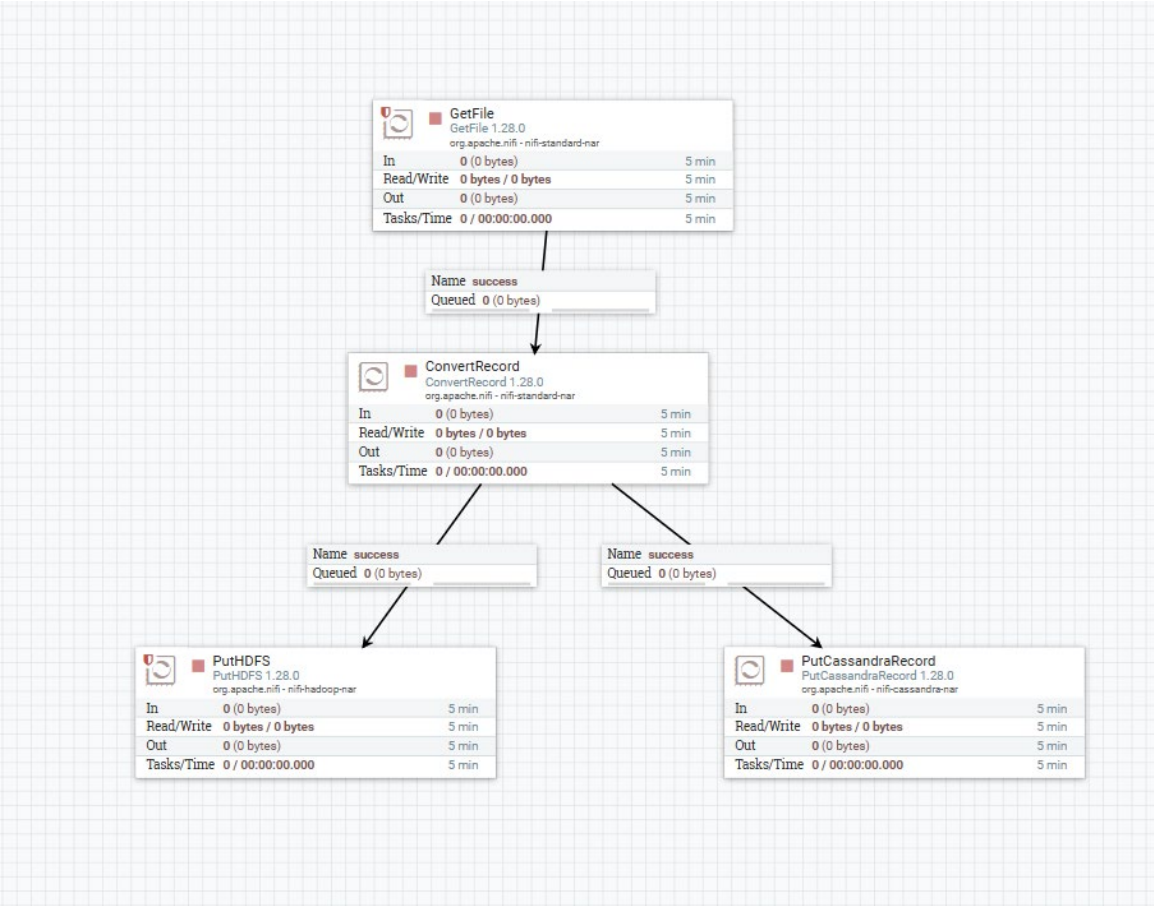
3.3.3.1. Configuración de Controller Services Esenciales

Para el correcto funcionamiento de los procesadores y la adecuada interpretación y escritura de los datos, se configuraron varios Controller Services cruciales dentro de NiFi:

- **CSVReader**: Este servicio se configuró para interpretar los datos del archivo CSV de entrada. La configuración clave incluyó:
  - Schema Access Strategy: Use Schema Text.
  - Schema Text: Se proporcionó un esquema Avro explícito para definir la estructura y los tipos de datos de cada columna del CSV. Un aspecto importante de este esquema fue la definición del campo price_date como {"type": "int", "logicalType": "date"}, para su correcta interpretación.
  - Date Format: Se especificó yyyy-MM-dd para asegurar que el CSVReader parseara correctamente las cadenas de fecha del CSV al tipo lógico date de Avro.
  - Treat First Line as Header: Establecido a true.
- **AvroRecordSetWriter**: Responsable de escribir los datos en el formato binario de Apache Avro.
  - Schema Access Strategy: Inherit Record Schema, para utilizar el esquema definido por el CSVReader.
  - Schema Write Strategy: Embed Avro Schema, lo que permite que cada FlowFile Avro resultante sea auto-descriptivo al incluir su esquema.
- **CassandraSessionProvider**: Para gestionar y proporcionar las conexiones al clúster de Apache Cassandra.
  - Cassandra Contact Points: Se configuró con el nombre del servicio y puerto de Cassandra dentro de la red Docker (cassandra:9042).
  - Keyspace: Se especificó el keyspace de destino, etf_data.
  - Client Auth: Se estableció en NONE, ya que no se configuró autenticación SSL para Cassandra en este entorno de desarrollo.
- **AvroReader**: Utilizado por el procesador PutCassandraRecord para leer los registros en formato Avro antes de su inserción en Cassandra.
  - Schema Access Strategy: Use Embedded Avro Schema, para interpretar el esquema contenido en los FlowFiles Avro.

3.3.3.2. Diseño y Configuración de Procesadores del Flujo Principal

El pipeline de datos en NiFi se construyó con la siguiente secuencia de procesadores principales:



- GetFile:
  - Propósito: Ingestar el archivo ETF_cohortes_final_preprocesado.csv.
  - Configuración Clave:
    - Input Directory: /data_in (ruta dentro del contenedor NiFi, mapeada a ./etf_data_input/ en el host Docker).
    - File Filter: ETF_cohortes_final_preprocesado\.csv para asegurar que solo se procese este archivo específico.
    - Keep Source File: Se mantuvo en true durante las fases de desarrollo y prueba para facilitar la depuración y reprocesamiento sin necesidad de volver a colocar el archivo.
- ConvertRecord:
  - Propósito: Transformar los datos del formato CSV al formato Avro.
  - Configuración Clave:
    - Record Reader: Se asignó el CSVReader configurado previamente.
    - Record Writer: Se asignó el AvroRecordSetWriter configurado previamente.
- PutCassandraRecord:
  - Propósito: Escribir los registros (en formato Avro, interpretados por su Record Reader) en la tabla designada de Apache Cassandra. Se optó por este procesador en lugar de PutCassandraQL debido a [mencionar brevemente el desafío de visibilidad de propiedades si se quiere, o simplemente indicar que es más adecuado para flujos basados en registros].

    - Configuración Clave:
        - Record Reader: Se asignó el AvroReader configurado.
        - Cassandra Connection Provider: Se asignó el CassandraSessionProvider.
        - Keyspace: etf_data.
        - Table Name: etf_quotes
        - Las relaciones failure se configuraron para auto-terminarse tras varios intentos o en caso de fallo persistente para simplificar el flujo en este contexto.
- PutHDFS:
    - Propósito: Almacenar una copia de los archivos Avro (provenientes de ConvertRecord) en el Hadoop Distributed File System (HDFS).
    - Configuración Clave:
        - Hadoop Configuration Resources: Se especificó la ruta al archivo core-site.xml (montado en el contenedor NiFi desde ./config/hadoop/core-site.xml en el host) que contenía la propiedad fs.defaultFS = hdfs://namenode:9000.
        - Directory: El directorio destino en HDFS (ej. /etf_project/data_avro/).
        - File Name: Se utilizó una expresión de NiFi para generar nombres de archivo únicos y evitar colisiones, incorporando un timestamp (${filename}_${now():format('yyyyMMddHHmmssSSS')}.avro)
        - Las relaciones failure y success se configuraron para auto-terminarse.

3.3.3.3. Estructura Lógica del Flujo

El flujo de datos se diseñó de la siguiente manera:

- El procesador GetFile ingesta el archivo CSV. Su cola de salida success se conecta al procesador ConvertRecord.
- El procesador ConvertRecord transforma los datos a Avro. Su cola de salida success se bifurca y se conecta a dos procesadores de forma paralela:
    - PutCassandraRecord
    - PutHDFS

3.3.3.4. Gestión de Archivos de Configuración Externos

Para el correcto funcionamiento del flujo, fue necesario montar ciertos archivos de configuración del host Docker en el contenedor de NiFi mediante volúmenes. Específicamente, el archivo core-site.xml de Hadoop (ubicado en ./config/hadoop/core-site.xml en el host) fue esencial para la configuración del procesador PutHDFS. El archivo CSV de entrada también se accedió a través de un volumen montado (./etf_data_input/ en el host).

```
mi_proyecto_etf/
├── docker-compose.yml
│
├── etf_data_input/
│   └── ETF_cohortes_final_preprocesado.csv
│
└── config/
    └── hadoop/
        └── core-site.xml
```

3.3.4. Almacenamiento con Apache Cassandra

Para el almacenamiento de los datos de ETFs que requieren acceso rápido y frecuente, particularmente para alimentar los procesos de predicción o visualizaciones interactivas, se seleccionó Apache Cassandra. Su capacidad para manejar altas tasas de ingesta de datos (provenientes del flujo de Apache NiFi) y su optimización para lecturas de baja latencia basadas en la clave primaria fueron determinantes para esta elección. Cassandra actúa en esta arquitectura como la base de datos operacional "caliente", complementando el rol de HDFS como almacén histórico masivo.

La configuración de Cassandra se realizó de la siguiente manera:

- Keyspace: Se creó un keyspace denominado `etf_data` para agrupar todas las tablas relacionadas con el proyecto.

```
CREATE KEYSPACE etf_data WITH replication = {'class':
'SimpleStrategy', 'replication_factor': '1'}  AND
durable_writes = true;
```

- Tabla de Cotizaciones (ETFs): Se diseñó una tabla principal para almacenar los datos históricos de precios y volumen. Asumiendo el nombre provisional `etf_quotes`, su esquema se definió para reflejar las columnas del dataset procesado por NiFi, con especial atención al tipo de dato de `price_date`. Un ejemplo de la estructura de la tabla sería:

```
CREATE TABLE IF NOT EXISTS etf_quotes (
    fund_symbol TEXT,
    price_date TIMESTAMP,
    open DOUBLE,
    high DOUBLE,
    low DOUBLE,
    close DOUBLE,
    adj_close DOUBLE,
    volume BIGINT,
    clasificacion INT,
    rentabilidad_x DOUBLE,
    rent_diaria DOUBLE,
    rent_trimestre DOUBLE,
    rent_anual DOUBLE,
    rent_total_acum DOUBLE,
```

```
    PRIMARY KEY ((fund_symbol), price_date)
) WITH CLUSTERING ORDER BY (price_date DESC);
```

3.3.5. Almacenamiento y Procesamiento con Apache Hadoop

Para el almacenamiento masivo y a largo plazo de la totalidad de los datos históricos de ETFs recopilados y procesados, se implementó el Hadoop Distributed File System (HDFS), componente del ecosistema Apache Hadoop . En esta arquitectura, HDFS actúa como el data lake principal o archivo histórico robusto, destinado a conservar el conjunto completo de información para futuros análisis batch, reentrenamiento de modelos con historiales extensos o cualquier exploración de datos a gran escala.

La configuración del clúster HDFS, gestionada a través de Docker Compose, consistió en un nodo NameNode y dos nodos DataNode. Los aspectos clave de su configuración y uso fueron:

- **Configuración del NameNode**: El NameNode se configuró con el URI del sistema de archivos `fs.defaultFS=hdfs://namenode:9000`, el cual fue referenciado por los DataNodes y por Apache NiFi para las operaciones de escritura.
- **Replicación de Datos:** Para asegurar la tolerancia a fallos y la durabilidad de los datos, se estableció un factor de replicación (`dfs.replication`) de 2. Esto significa que cada bloque de datos almacenado en HDFS fue replicado en ambos DataNodes disponibles, protegiendo la información contra la posible falla de un nodo individual.
- **Ingesta de Datos desde NiFi**: Los datos de ETFs, una vez transformados al formato Apache Avro por el procesador `ConvertRecord` en Apache NiFi, eran enviados al HDFS mediante el procesador `PutHDFS`.
    - **Formato de Almacenamiento**: Los datos se almacenaron en HDFS en formato Avro. Este formato fue seleccionado por ser un formato de serialización de datos compacto, rápido y que incluye el esquema junto con los datos, facilitando su posterior lectura e interpretación por diversas herramientas de procesamiento.
    - **Estructura de Directorios**: Los archivos Avro se depositaron en un directorio designado dentro de HDFS, por ejemplo, `/etf_project/data_avro/`.
    - **Nomenclatura de Archivos**: Para asegurar la unicidad y evitar la sobrescritura accidental, los archivos Avro se nombraron utilizando una expresión en NiFi que incluía el nombre original y un timestamp, como por ejemplo `${filename}_${now():format('yyyyMMddHHmmssSSS')}.avro`.
    - **Configuración de core-site.xml (Archivo core-site)**: El procesador `PutHDFS` en NiFi se configuró con la ruta al archivo core-site.xml (montado en el contenedor de NiFi), que contenía la directiva `fs.defaultFS` necesaria para la conexión con el NameNode de HDFS

.

La elección de HDFS como el sistema para el almacenamiento histórico se basa en su probada capacidad para manejar petabytes de datos de manera escalable y económica. Su optimización para lecturas secuenciales de grandes bloques de datos lo hace ideal como fuente para herramientas de procesamiento masivo (como Apache Spark, si se utilizara en futuras extensiones) que se benefician del acceso a la totalidad del histórico para entrenar modelos de Machine Learning o realizar análisis retrospectivos profundos.

3.4. Desarrollo y Entrenamiento de Modelos de Predicción

3.5. Uso de Herramientas de Visualización y Minería

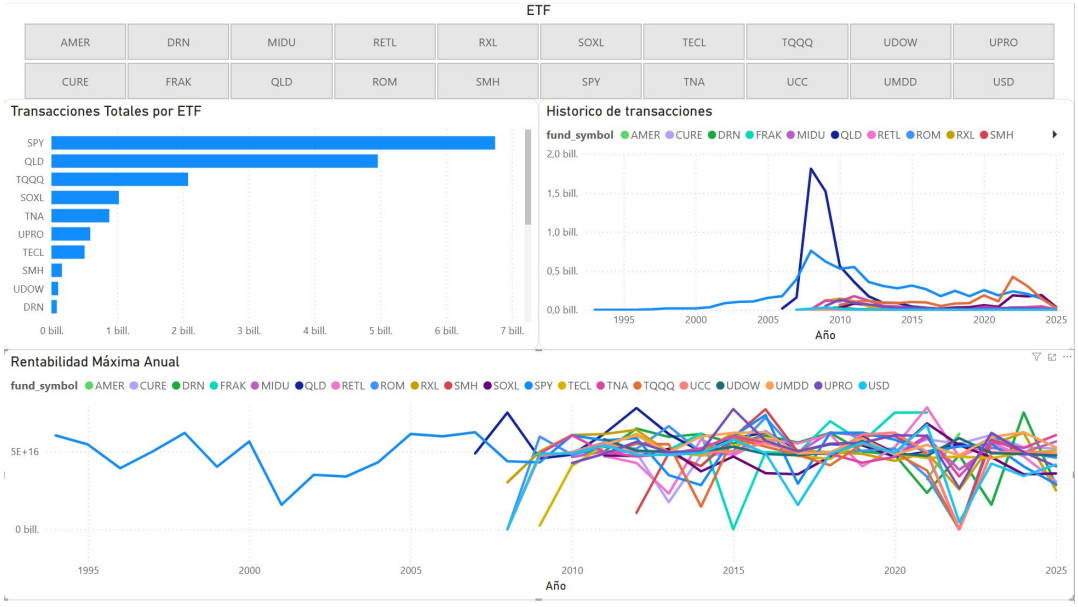
3.5.1. Visualización con Power BI

Para la exploración visual de los datos y la presentación de los hallazgos clave del proyecto, se utilizó Microsoft Power BI. Esta herramienta se conectó directamente al archivo *ETF_cohortes_final_preprocesado.csv*, el cual contiene los datos de los ETFs ya limpios y preparados tras la fase de preprocesamiento.

Se diseñó un informe en Power BI con el objetivo de ofrecer una visión general y la capacidad de explorar aspectos específicos del comportamiento de los ETFs. Este informe incluyó las siguientes visualizaciones principales:

- Gráfico "Transacciones Totales por ETF":
  - Propósito: Mostrar una comparativa del volumen total de negociación acumulado para cada ETF presente en el dataset, permitiendo identificar rápidamente los ETFs con mayor o menor actividad transaccional.
- Gráfico "Histórico de Transacciones":
  - Propósito: Visualizar la evolución de la actividad transaccional o de los precios a lo largo del tiempo para los ETFs seleccionados.
- Gráfico "Rentabilidad Máxima Anual":
  - Propósito: Identificar y comparar el rendimiento anual más alto alcanzado por los diferentes ETFs o por el conjunto de ETFs a lo largo de los años.

Para facilitar la exploración y el análisis focalizado, el informe de Power BI se dotó de funcionalidad interactiva. Específicamente, se implementaron botones de filtro basados en la columna *fund_symbol*. Esto permite al usuario seleccionar uno o varios ETFs de interés, y todos los gráficos del informe se actualizan dinámicamente para reflejar únicamente la información correspondiente a los ETFs seleccionados, permitiendo un análisis comparativo y detallado de forma intuitiva.



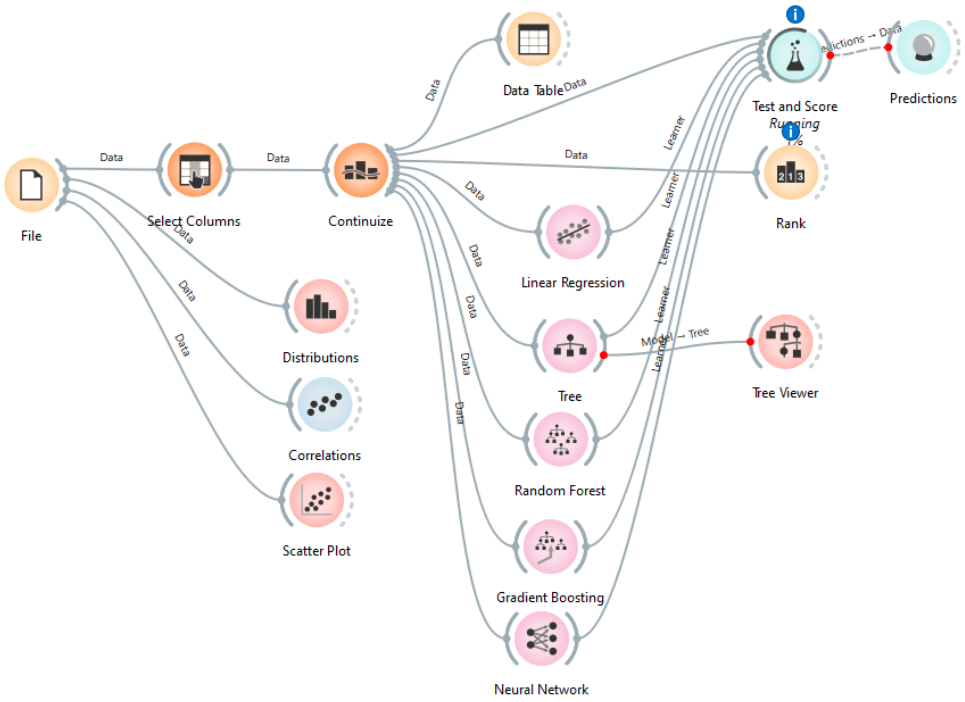
3.5.2. Minería de Datos con Orange Datamining

Además de las implementaciones de modelos en Python, se utilizó la plataforma Orange Datamining para realizar análisis complementarios y explorar visualmente el proceso de modelado predictivo. Orange fue seleccionado por su interfaz gráfica intuitiva, que permite la construcción de flujos de trabajo de machine learning mediante la conexión de widgets, facilitando el prototipado rápido y la experimentación sin necesidad de codificación extensiva. El enfoque principal en Orange fue la implementación de un flujo de trabajo de regresión para predecir el valor numérico de la variable adj_close de los ETFs.

El proceso metodológico seguido en Orange se detalla a continuación:

- **Carga de Datos**: El flujo de trabajo comenzó con el widget *File*, mediante el cual se cargó el archivo CSV (*ETF_cohortes_final_preprocesado.csv*) que contiene los datos históricos de los ETFs.
- **Exploración Inicial de Datos**: Una vez cargados los datos, se emplearon widgets de visualización como *Distributions*, *Correlations* y *Scatter Plot* conectados directamente al widget File. Esta etapa permitió una comprensión inicial de las características de los datos, la visualización de las distribuciones de las variables, la identificación de correlaciones relevantes (por ejemplo, entre las diferentes cotizaciones y el volumen, o entre las características y la variable objetivo) y la búsqueda de patrones visuales preliminares.
- **Selección y Definición de Variables**: Se utilizó el widget *Select Columns* para definir explícitamente los roles de cada columna del dataset. La variable *adj_close* fue designada como la Variable Objetivo (Target) a predecir. Un conjunto de columnas, incluyendo *open*, *high*, *low*, *close*, *volume*, *fund_symbol* y *Clasificacion*, se establecieron como las Características (Features) de entrada para los modelos de regresión. Otras variables (como *price_date*, *Rentabilidad*, u otras que no fueran a usarse directamente como predictores) se marcaron como Metadatos o se excluyeron para evitar problemas como la fuga de datos (data leakage).
- **Preprocesamiento Centralizado de Datos**: Todas las características seleccionadas se pasaron al widget *Continuize*. Este widget fue un paso clave para preparar los datos para los algoritmos de regresión, realizando dos tareas principales:
    - Codificación de Variables Categóricas: Las variables categóricas como *fund_symbol* y *Clasificacion* fueron transformadas utilizando la técnica de One-Hot Encoding, convirtiéndolas en múltiples columnas numéricas binarias (0/1).
    - Escalado de Variables Numéricas: Las variables numéricas continuas (*open*, *high*, *low*, *close*, *volume*) fueron escaladas (mediante las opciones de *Standardize* o *Normalize* dentro del widget) para que sus valores tuvieran rangos comparables, una práctica beneficiosa para el rendimiento de muchos algoritmos de regresión.
- **Modelado y Entrenamiento**: La salida de datos preprocesados del widget *Continuize* se conectó como entrada a varios widgets de algoritmos de regresión. Los modelos específicos que se probaron y compararon incluyeron: *Linear Regression*, *Tree* , *Random Forest*, *Neural Network* y *Gradient Boosting*.
- **Evaluación Robusta de Modelos**: Para evaluar y comparar el rendimiento de los diferentes modelos de regresión, se utilizó el widget Test & Score como componente central.
    - Los datos preprocesados (provenientes de *Continuize*) se enviaron directamente al widget *Test & Score*.
    - Se configuró para utilizar la técnica de Validación Cruzada (Cross-Validation), específicamente con 10 folds (o el número de folds que hayas usado). Este

método proporciona una estimación más robusta del rendimiento del modelo que una simple división entrenamiento/prueba, ya que todos los datos se utilizan tanto para entrenar como para probar el modelo en diferentes iteraciones.
    - Todos los modelos de regresión (Learners) implementados (*Linear Regression*, *Tree*, etc.) se conectaron al widget *Test & Score*. Esto permitió una comparación directa de su rendimiento bajo las mismas condiciones de validación y utilizando un conjunto común de métricas de regresión, como el Error Cuadrático Medio (RMSE), el Error Absoluto Medio (MAE) y el Coeficiente de Determinación (R^2).
- **Análisis de Resultados y Predicciones**:
    - Los resultados numéricos de la evaluación (las métricas para cada modelo) se analizaron directamente en la tabla proporcionada por el widget *Test & Score* para identificar los modelos con mejor desempeño predictivo.
    - Se empleó el widget Rank (conectado a la salida de *Continuize*) para evaluar la importancia relativa de las diferentes características de entrada (incluyendo aquellas generadas por el One-Hot Encoding) en la predicción de *adj_close*.
    - Para modelos interpretables como los árboles de regresión, se utilizó el widget *Tree Viewer* para visualizar la estructura del árbol aprendido y entender las reglas de decisión generadas.



3.6. Desafíos Técnicos y Soluciones Implementadas

Durante el diseño e implementación de la arquitectura Big Data y el flujo de procesamiento de datos, se presentaron diversos desafíos técnicos que requirieron investigación, experimentación y ajustes en el enfoque inicial. A continuación, se describen los más significativos y las soluciones adoptadas:

3.6.1. Problemas con el Procesador *PutCassandraQL* en Apache NiFi

- **Desafío**: Durante la configuración del flujo para la ingesta de datos en Apache Cassandra, se encontraron dificultades persistentes con el procesador *PutCassandraQL* en varias versiones probadas de NiFi (incluyendo 1.24.0 y la finalmente utilizada 1.28.0). Específicamente, el procesador no mostraba en la interfaz gráfica propiedades esenciales para su configuración, como "Statement Type" o "Table Name", lo que impedía su uso directo para realizar operaciones de inserción de manera sencilla.
- **Solución**: Como alternativa y solución a este inconveniente, se optó por utilizar el procesador *PutCassandraRecord*. Este procesador está diseñado para trabajar con flujos basados en registros (como los FlowFiles en formato Avro generados por el procesador *ConvertRecord*) y permitió una configuración más directa y funcional para la inserción de datos en Cassandra, utilizando el *CassandraSessionProvider* y el *AvroReader* configurados previamente.

3.6.2. Manejo de Tipos de Dato de Fecha para la Ingesta en Cassandra

- **Desafío**: Un obstáculo recurrente fue la correcta inserción de la columna *price_date* en Apache Cassandra. Se produjeron múltiples errores del tipo *InvalidQueryException*: *Expected 4 byte long for date (X)* al intentar insertar los datos, indicando una incompatibilidad entre el formato de fecha que NiFi estaba enviando y el que Cassandra esperaba para su tipo *DATE* o *TIMESTAMP*.
- **Solución**: Tras varias iteraciones y pruebas, la solución que permitió una ingesta exitosa y la correcta representación de la fecha en Cassandra implicó una configuración coordinada:
    - En el *CSVReader* de NiFi, se definió el *Schema Text* para la columna *price_date* como *{"type": "int", "logicalType": "date"}* y se especificó el *Date Format* como *yyyy-MM-dd*. Esto permitió a NiFi interpretar correctamente la cadena de fecha del CSV y manejarla internamente de una forma que resultaba en un valor numérico.
    - En la tabla de Apache Cassandra, la columna *price_date* se definió con el tipo de dato *TIMESTAMP*. Este tipo en Cassandra es capaz de aceptar representaciones numéricas de fechas, lo que lo hizo compatible con el formato de 8 bytes que NiFi enviaba tras la interpretación del *logicalType: date* de Avro.

3.6.3. Error de Configuración en el Procesador PutHDFS

- **Desafío**: Al configurar el procesador *PutHDFS* para escribir los archivos Avro en HDFS, se encontró un error que indicaba "Specified Resource is a DIRECTORY" para la propiedad *Hadoop Configuration Resources*, a pesar de que se estaba apuntando a la ruta del archivo *core-site.xml*.
- **Solución**: Se determinó que el error ocurría si el archivo *core-site.xml* no existía físicamente en la ruta esperada del host Docker antes de que los contenedores fueran iniciados con *docker-compose up -d*. La solución consistió en asegurar que el archivo *core-site.xml* (con la configuración *fs.defaultFS = hdfs://namenode:9000*) estuviera correctamente ubicado en la ruta del host (*./config/hadoop/core-site.xml*) que se mapeaba al volumen del contenedor de NiFi previo al inicio de la pila de servicios.

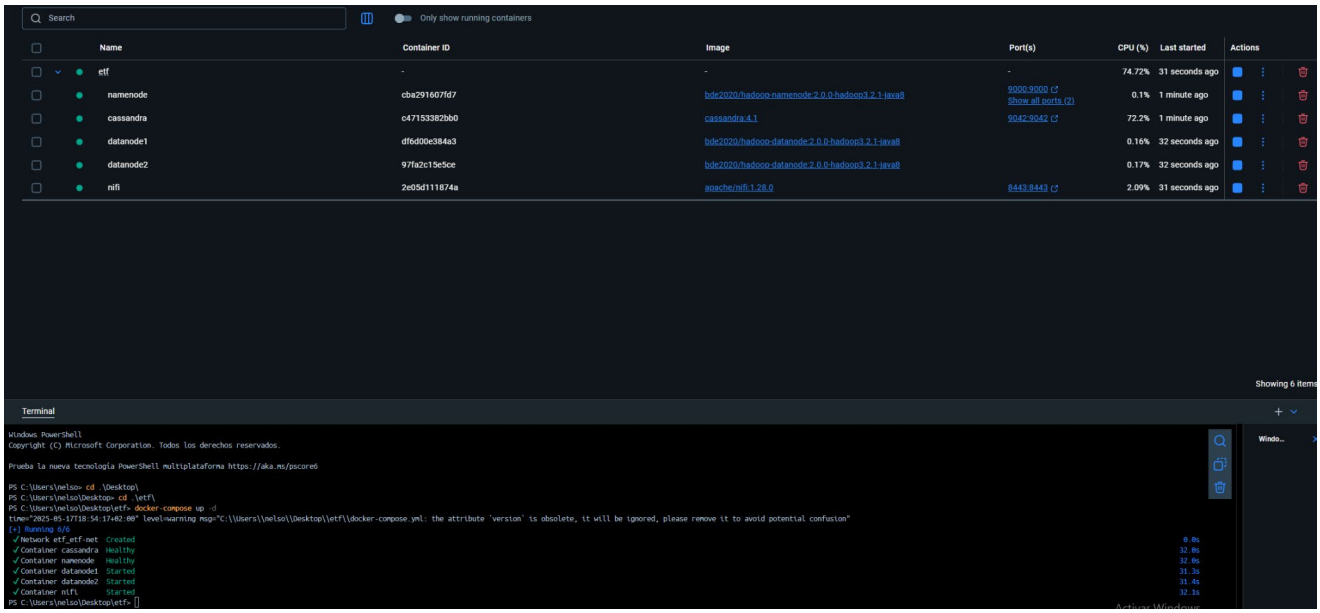
4. Resultados y Pruebas

4.1. Resultados de la Implementación y Pruebas de la Arquitectura Big Data

Tras la implementación de los componentes y flujos descritos en la metodología (Capítulo 3), se procedió a la ejecución y verificación de la arquitectura Big Data. Los resultados de estas pruebas confirmaron el correcto funcionamiento y la integración de las tecnologías seleccionadas.

4.1.1. Funcionamiento del Entorno Contenerizado (Docker y Docker Compose)

La arquitectura de servicios, gestionada mediante Docker Compose, demostró ser robusta y estable. La ejecución del comando *docker-compose up -d* resultó en el levantamiento coordinado y exitoso de todos los contenedores definidos: Apache NiFi, el NameNode y los dos DataNodes de HDFS, y el nodo de Apache Cassandra. Se verificó que los servicios se iniciaron sin errores y que la comunicación entre ellos, a través de la red Docker interna *etf-net*, se estableció correctamente, lo cual fue evidenciado por la capacidad de NiFi para conectarse y enviar datos tanto a HDFS como a Cassandra. La configuración de volúmenes para la persistencia de datos también funcionó según lo esperado, manteniendo la información y las configuraciones de los servicios tras detener y reiniciar los contenedores.



4.1.2. Operación del Flujo de Ingesta con Apache NiFi

El flujo de datos diseñado en Apache NiFi para la ingesta, transformación y distribución del archivo *ETF_cohortes_final_preprocesado.csv* se ejecutó de manera satisfactoria. Se observó a través de la interfaz de usuario de NiFi que:

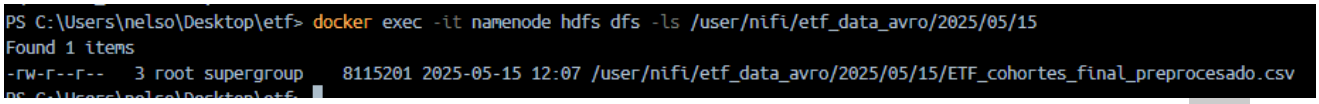
- El procesador *GetFile* identificó y recogió correctamente el archivo CSV del directorio de entrada especificado (*/data_in*).
- El procesador *ConvertRecord* transformó los datos del formato CSV al formato Apache Avro, aplicando las definiciones de esquema y conversiones de tipo de dato configuradas.
- Posteriormente, el flujo bifurcado distribuyó exitosamente los datos procesados en formato Avro a los dos destinos paralelos:
    - Los datos fueron escritos en Apache HDFS mediante el procesador *PutHDFS*.

- Los datos fueron ingeridos en la tabla correspondiente de Apache Cassandra mediante el procesador *PutCassandraRecord*. El proceso completo se llevó a cabo sin errores reportados por NiFi y se pudo monitorizar el tránsito de los FlowFiles a través de cada etapa del pipeline.

4.1.3. Verificación del Almacenamiento de Datos (Apache HDFS y Apache Cassandra)

Se realizaron comprobaciones en ambos sistemas de almacenamiento para confirmar la correcta recepción y persistencia de los datos:

- En Apache HDFS: Mediante la ejecución de comandos del sistema de archivos de Hadoop (*hdfs dfs -ls /etf_project/data_avro/*), se constató la presencia de los archivos en formato Avro en el directorio de destino especificado. Se verificó que los nombres de los archivos correspondían al patrón definido en NiFi y que. La configuración de replicación de datos (factor de 2) se mantuvo activa, asegurando la distribución de los bloques de datos entre los dos DataNodes.



- En Apache Cassandra: A través de la interfaz de línea de comandos *cqlsh*, se ejecutaron sentencias *SELECT* sobre la tabla de destino en el keyspace *etf_data* (ej. *SELECT * FROM etf_data.etf_quotes LIMIT 10;* y *SELECT COUNT(*) FROM etf_data.etf_quotes;*). Estas consultas confirmaron que la tabla fue poblada correctamente con los datos de los ETFs, que el número de registros era el esperado [si lo contaste] y que la estructura de los datos, incluyendo los tipos de dato como *TIMESTAMP* para la columna *price_date*, se correspondía con el esquema definido.

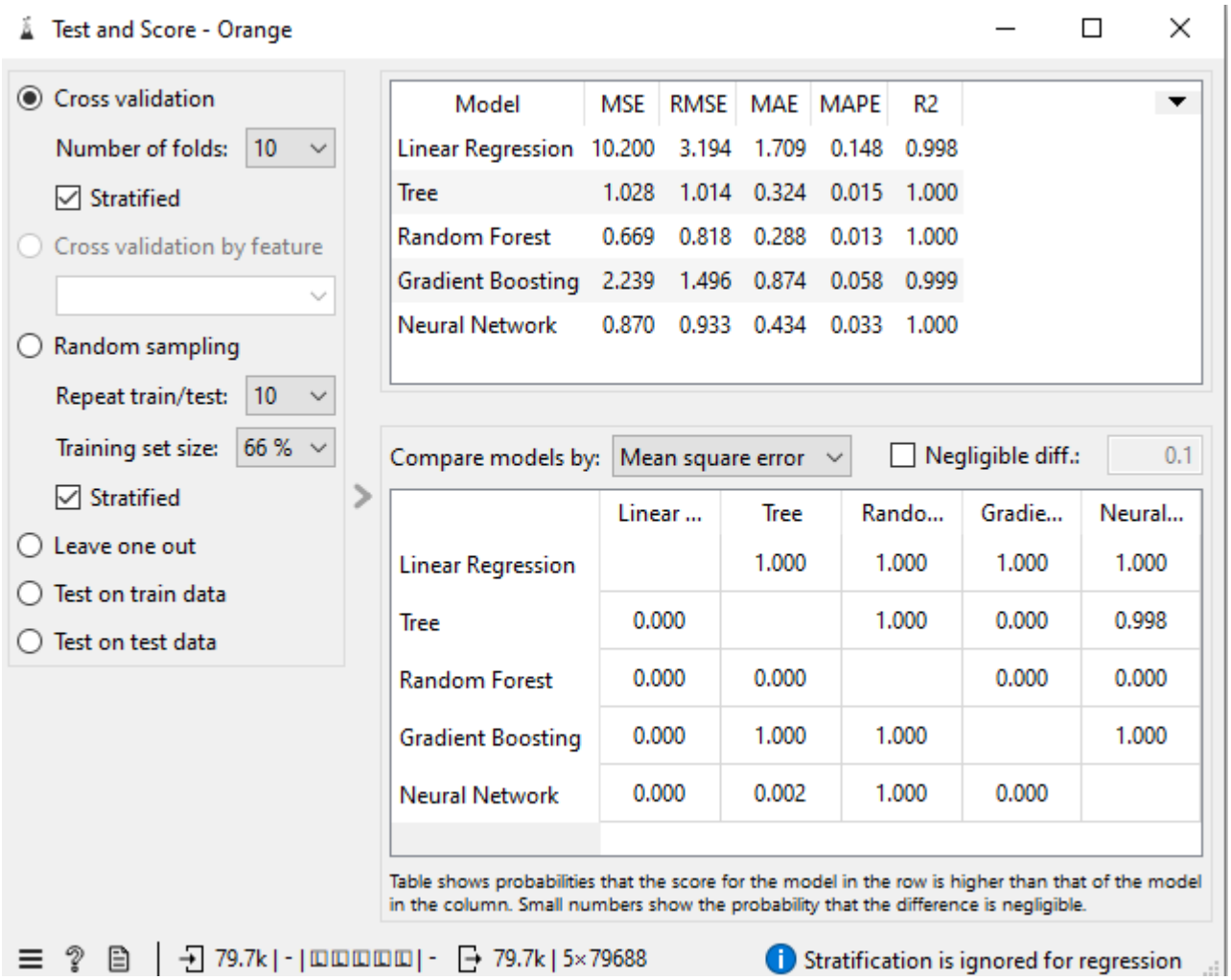
4.2. Resultados de Visualización y Minería de Datos

4.2.1. Resultados de la Evaluación de Modelos de Regresión en Orange Datamining

Para evaluar y comparar el rendimiento de diferentes algoritmos de regresión en la predicción de la variable adj_close de los ETFs, se utilizó el widget Test & Score de Orange Datamining. La evaluación se configuró empleando un esquema de validación cruzada (Cross-Validation) de 10 folds. Aunque la opción de estratificación estaba

marcada, se reconoce que esta es ignorada por Orange para tareas de regresión.

Se evaluaron cinco modelos de regresión: Regresión Lineal, Árbol de Decisión (Tree), Random Forest, Gradient Boosting y Red Neuronal (Neural Network). Las métricas clave de rendimiento registradas incluyeron el Error Cuadrático Medio (MSE), la Raíz del Error Cuadrático Medio (RMSE), el Error Absoluto Medio (MAE), el Error Porcentual Absoluto Medio (MAPE) y el Coeficiente de Determinación (R^2).



Del análisis de los resultados presentados en la imagen, se extraen las siguientes observaciones clave:

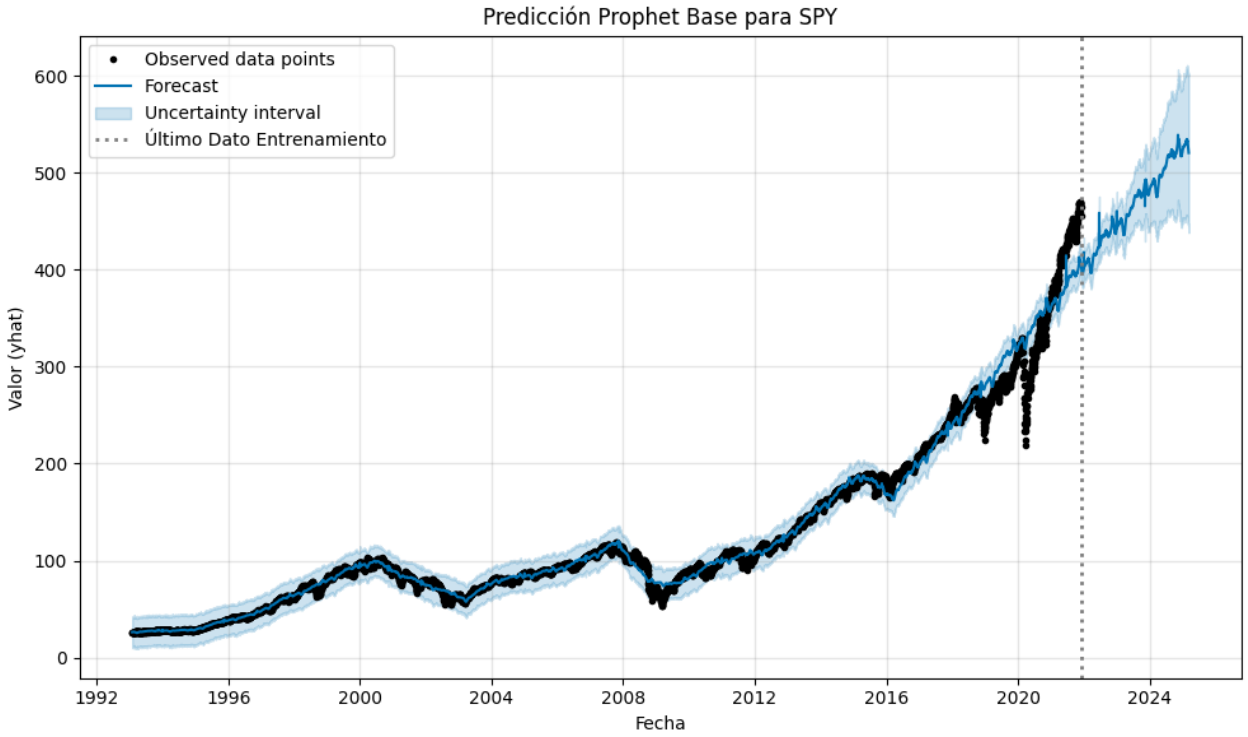
- El modelo **Random Forest** demostró ser el de mejor rendimiento general, alcanzando los menores valores en las métricas de error (RMSE: 0.818; MAE: 0.288; MAPE: 1.3%) y un Coeficiente de Determinación (R^2) de 1.000.
- La **Red Neuronal** se posicionó como el segundo mejor modelo, con un RMSE de 0.933, un MAE de 0.434 y también un R^2 de 1.000, mostrando un rendimiento muy cercano al de Random Forest.
- El modelo de **Árbol de Decisión** también exhibió un excelente desempeño, con un RMSE de 1.014, un MAE de 0.324 y un R^2 de 1.000.
- Los modelos de **Gradient Boosting** (RMSE: 1.496; R^2: 0.999) y, notablemente, la **Regresión Lineal** (RMSE: 3.194; R^2: 0.998) obtuvieron resultados inferiores en términos de error en comparación con los tres modelos anteriores, aunque sus valores de R^2 siguen siendo muy altos.

4.3. Resultados de los Modelos de Predicción

Análisis Detallado de los Resultados de Predicción con Prophet para el ETF (Exchange Traded Funds) SPY

Las siguientes visualizaciones son el resultado de aplicar modelos Prophet para predecir el comportamiento futuro del precio del ETF SPY, un índice de referencia del mercado bursátil estadounidense. Cada gráfico ofrece una perspectiva diferente sobre el rendimiento y las expectativas generadas por estos modelos.

Grafico Histórico y Predicción Futura para SPY



En este gráfico se muestra representado la evolución histórica del precio de cierre ajustado del ETF SPY, y de forma superpuesta, la predicción futura y el intervalo de incertidumbre correspondiente de nuestro modelo.

Datos Observados (Históricos): La dispersión de puntos negros representa la trayectoria histórica del precio de cierre ajustado del SPY. La escala logarítmica es crucial aquí, ya que transforma el crecimiento exponencial en una tendencia más lineal. Esto facilita la identificación de tasas de crecimiento consistentes a lo largo del tiempo. En este gráfico, la tendencia general alcista, característica del mercado de valores a largo plazo, se manifiesta como una pendiente positiva. Las fluctuaciones alrededor de esta tendencia reflejan la volatilidad inherente al mercado. Para tu proyecto, esta visualización inicial confirma la naturaleza no estacionaria de la serie temporal del precio, lo que justifica el uso de modelos como Prophet, capaces de capturar tendencias y estacionalidades.

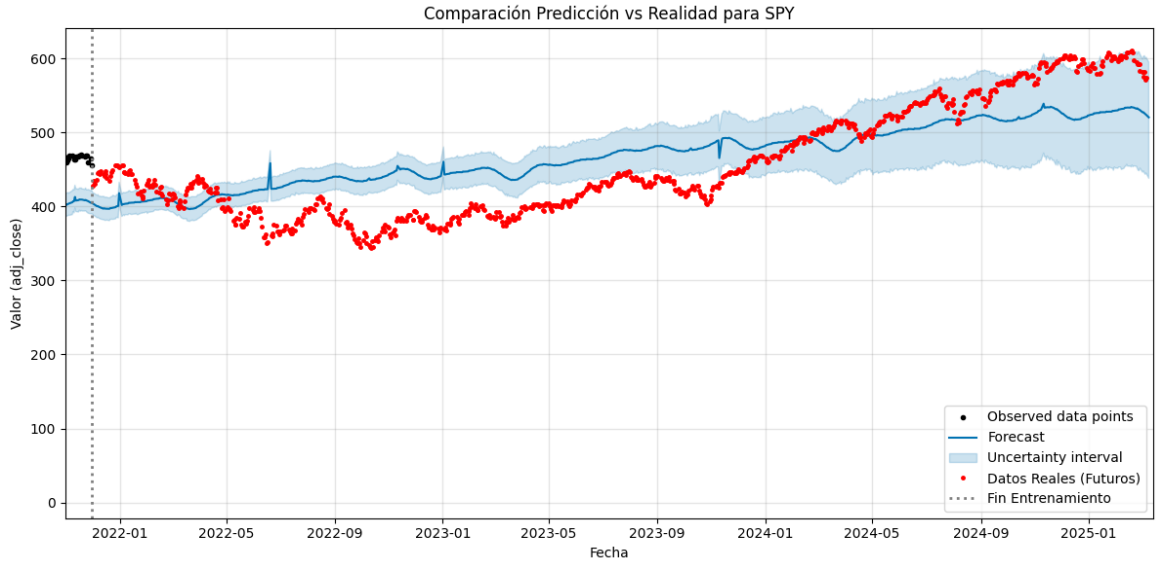
Predicción (yhat) a Largo Plazo: La línea azul continua representa la predicción del modelo Prophet extendiéndose hacia el futuro más allá de la fecha de fin del entrenamiento (marzo de 2025). La pendiente positiva de esta línea sugiere que el modelo anticipa una continuación del crecimiento histórico del SPY.Esta predicción constituye una hipótesis fundamental sobre la dirección futura del mercado, que deberá ser validada con datos reales posteriores, como mostraremos en los gráficos siguientes.

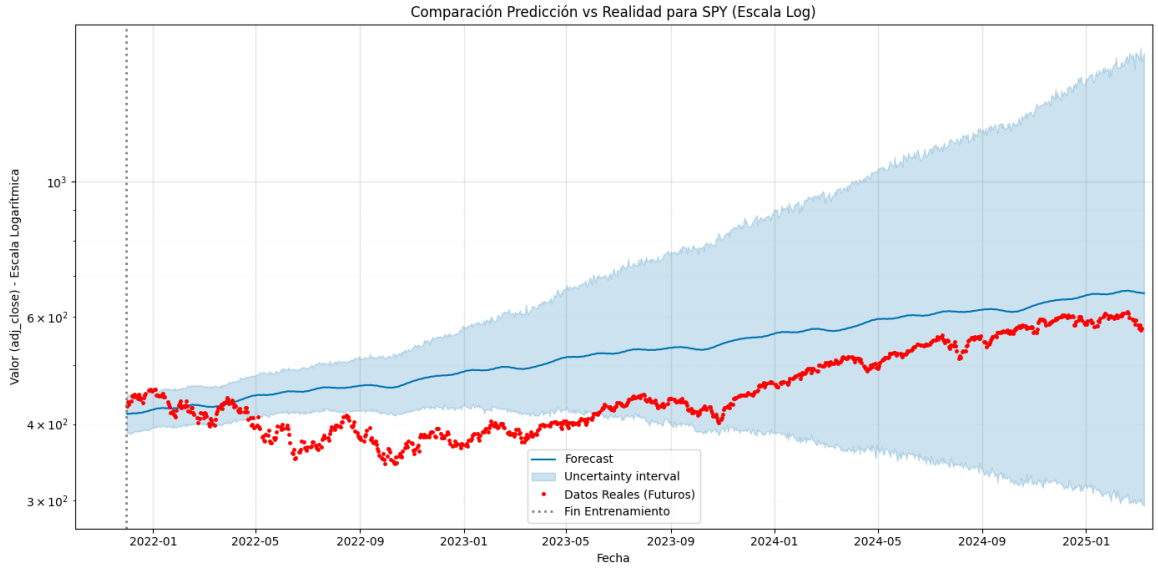
Intervalo de Incertidumbre Creciente: La banda sombreada en color celeste alrededor de la línea de predicción ilustra el intervalo de incertidumbre asociado con la predicción. Este intervalo se ensancha a

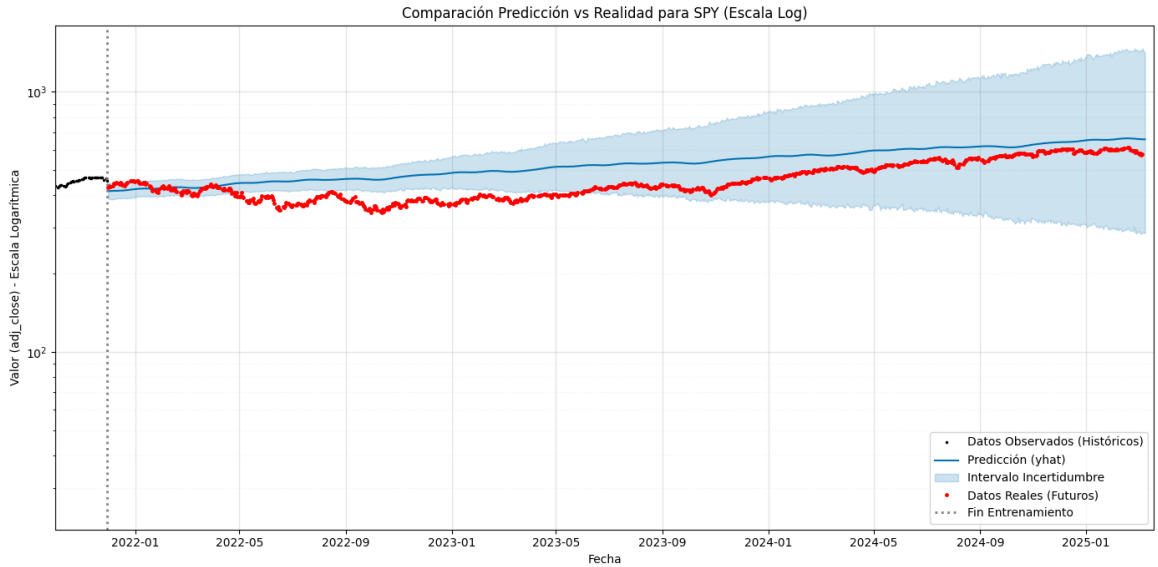
medida que el horizonte de predicción se extiende más en el futuro. Esta expansión es una característica inherente de las predicciones a largo plazo en sistemas complejos como los mercados financieros, donde la incertidumbre aumenta debido a la acumulación de posibles eventos imprevistos. Este intervalo de incertidumbre es vital para comprender el rango de posibles resultados y gestionar las expectativas sobre la precisión de la predicción a largo plazo. Un intervalo amplio sugiere una mayor probabilidad de desviaciones significativas del valor predicho.

Fin Entrenamiento (2025-03-07): La línea vertical punteada marca el punto temporal hasta el cual el modelo Prophet fue entrenado con datos históricos. Todo lo que se encuentra a la derecha de esta línea es la predicción generada por el modelo. Esta demarcación es crucial para entender qué parte de la visualización se basa en datos reales y cuál es una extrapolación del modelo.

Gráfico comparativo de Predicción vs Realidad para SPY en diferentes Escalas







Estos gráficos son cruciales para evaluar el rendimiento del modelo Prophet al comparar la predicción con los

datos reales observados después del período de entrenamiento, y se muestra en diversas escalas para mostrar de mejor manera el rendimiento de nuestro modelo frente a los valores reales extraídos mediante el proceso de extracción de nuevos datos.

Datos Observados (Históricos) en Escala Logarítmica: La dispersión de puntos negros representa la trayectoria histórica del precio de cierre ajustado del SPY. La escala logarítmica es crucial aquí, ya que transforma el crecimiento exponencial en una tendencia más lineal. Esto facilita la identificación de tasas de crecimiento consistentes a lo largo del tiempo. En este gráfico, la tendencia general alcista, característica del mercado de valores a largo plazo, se manifiesta como una pendiente positiva. Las fluctuaciones alrededor de esta tendencia reflejan la volatilidad inherente al mercado.

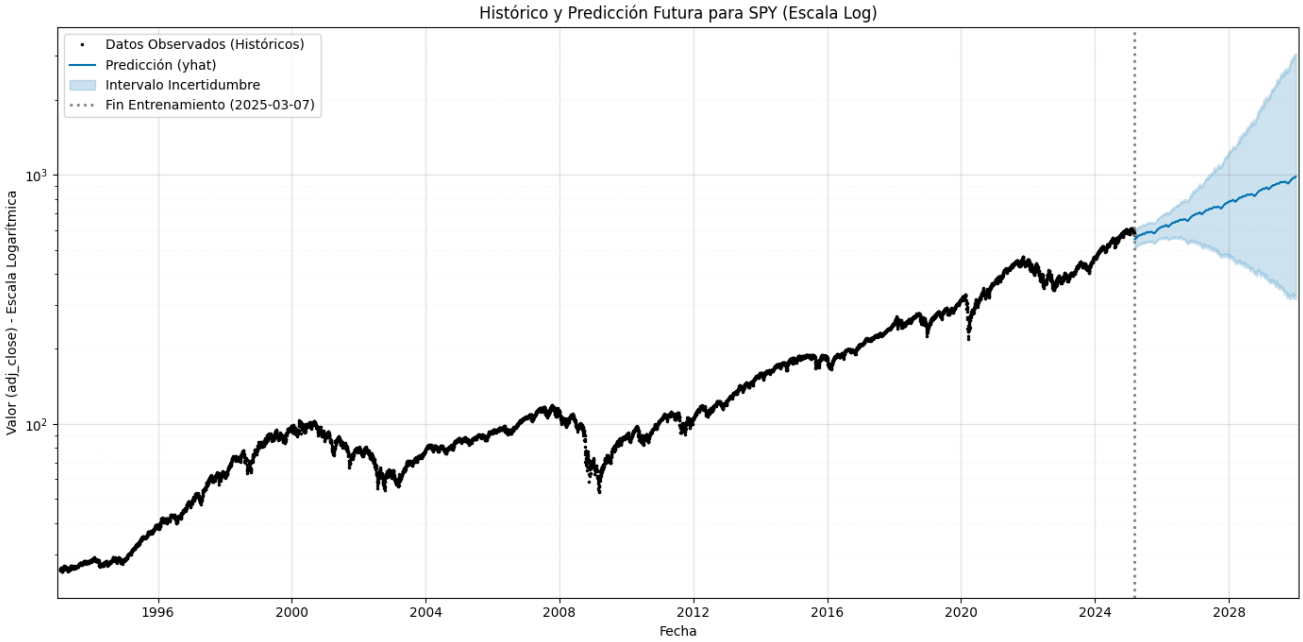
Predicción (yhat) en Escala Logarítmica: La línea azul continua muestra la predicción del modelo para el período posterior al entrenamiento.

Intervalo Incertidumbre en Escala Logarítmica: La banda celeste sombreada indica el rango de incertidumbre de la predicción.

Datos Reales (Futuros) en Escala Logarítmica: Los puntos rojos representan los precios reales del ETF SPY observados después de la fecha de fin del entrenamiento. La posición de estos puntos en relación con la línea de predicción (azul) y el intervalo de incertidumbre (celeste) es fundamental para evaluar la precisión del modelo.

Fin Entrenamiento: La línea vertical punteada separa el período de entrenamiento del período de evaluación de la predicción.

Gráfico de Análisis Detallado de la Predicción Futura basada en Prophet para SPY



Este gráfico ilustra la evolución histórica del precio de cierre ajustado del ETF SPY, junto con la predicción futura generada por mi modelo Prophet, todo ello visualizado en una escala logarítmica en el eje vertical.

- **Datos Observados (Históricos) en Escala Logarítmica:** Los puntos negros dispersos a lo largo del gráfico representan los valores históricos del precio de cierre ajustado del ETF SPY a través del tiempo. La utilización de una escala logarítmica es significativa, ya que transforma el crecimiento porcentual en una progresión lineal. Esto facilita la identificación de tendencias de crecimiento relativas a lo largo de extensos periodos. En la visualización, se aprecia claramente una tendencia alcista general del SPY a largo plazo, manifestada como una pendiente positiva en la nube de puntos. Las fluctuaciones alrededor de esta tendencia reflejan la volatilidad natural del mercado de valores.

- **Predicción (yhat) en Escala Logarítmica:** La línea azul continua que se extiende más allá de la línea vertical punteada representa la predicción del modelo Prophet para el precio futuro del SPY. La pendiente positiva de esta línea predictiva sugiere que el modelo anticipa una continuación de la tendencia de crecimiento histórico en términos porcentuales, con algunas fluctuaciones emergentes de forma no continua. Esta predicción constituye una expectativa clave sobre la trayectoria futura del SPY, que deberá ser contrastada con los datos reales a medida que se disponga de ellos, para poder cerciorarnos de que nuestro modelo es definitivamente preciso.

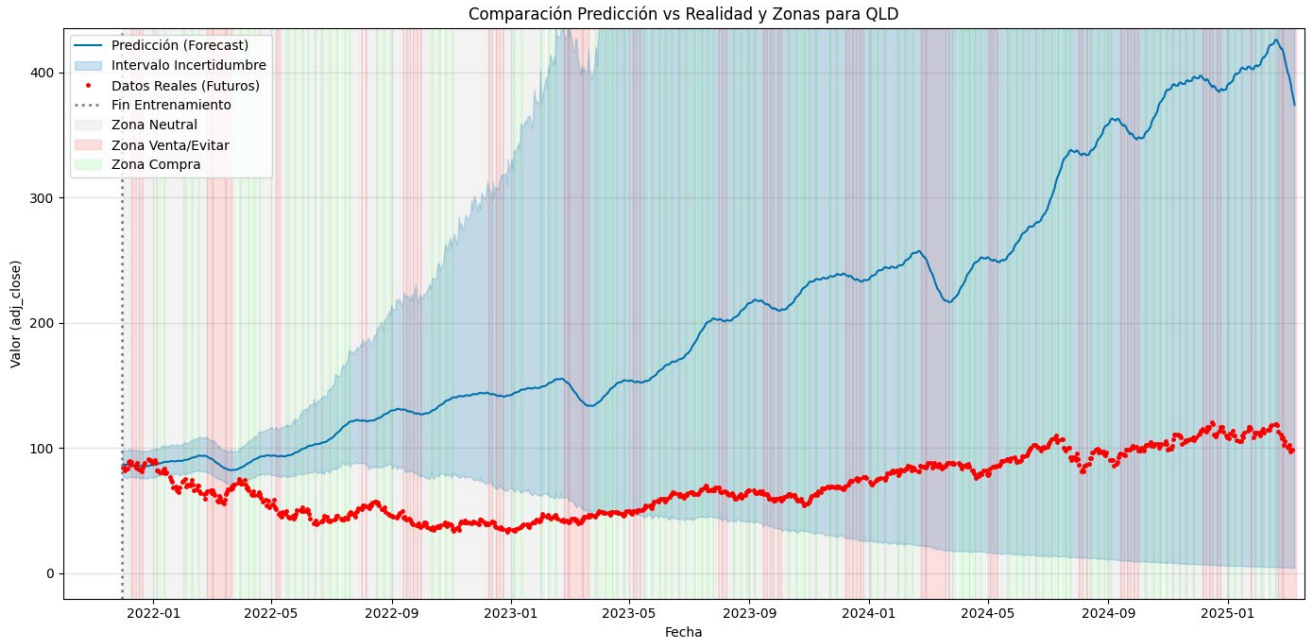
- **Intervalo de Incertidumbre en Escala Logarítmica:** La zona sombreada en color celeste que rodea la línea de predicción delimita el intervalo de incertidumbre asociado a la predicción. Este intervalo representa el rango dentro del cual es probable que se encuentre el precio real del SPY en el futuro, con un cierto nivel de confianza estadística. Se observa que este intervalo tiende a ensancharse a medida que el horizonte de predicción se extiende más allá de la fecha de fin del entrenamiento. Este aumento de la incertidumbre es un fenómeno común en las predicciones de series temporales financieras a largo plazo, debido a la creciente probabilidad de eventos imprevistos que pueden afectar el mercado. Para finalizar aclarar que un intervalo amplio indica una mayor incertidumbre en la predicción a largo plazo.

- **Fin Entrenamiento (2025-03-07):** La línea vertical punteada marca la fecha hasta la cual se utilizaron los datos históricos para entrenar el modelo Prophet (7 de marzo de 2025). Todo lo que se sitúa a la

derecha de esta línea corresponde a la predicción generada por el modelo para el futuro. Esta demarcación temporal es esencial para distinguir entre los datos reales utilizados para el aprendizaje del modelo y las proyecciones futuras.

Resultados modelo clasificación

Gráfico de Análisis de los Resultados del Modelo Basado en Zonas_PctChange para QLD



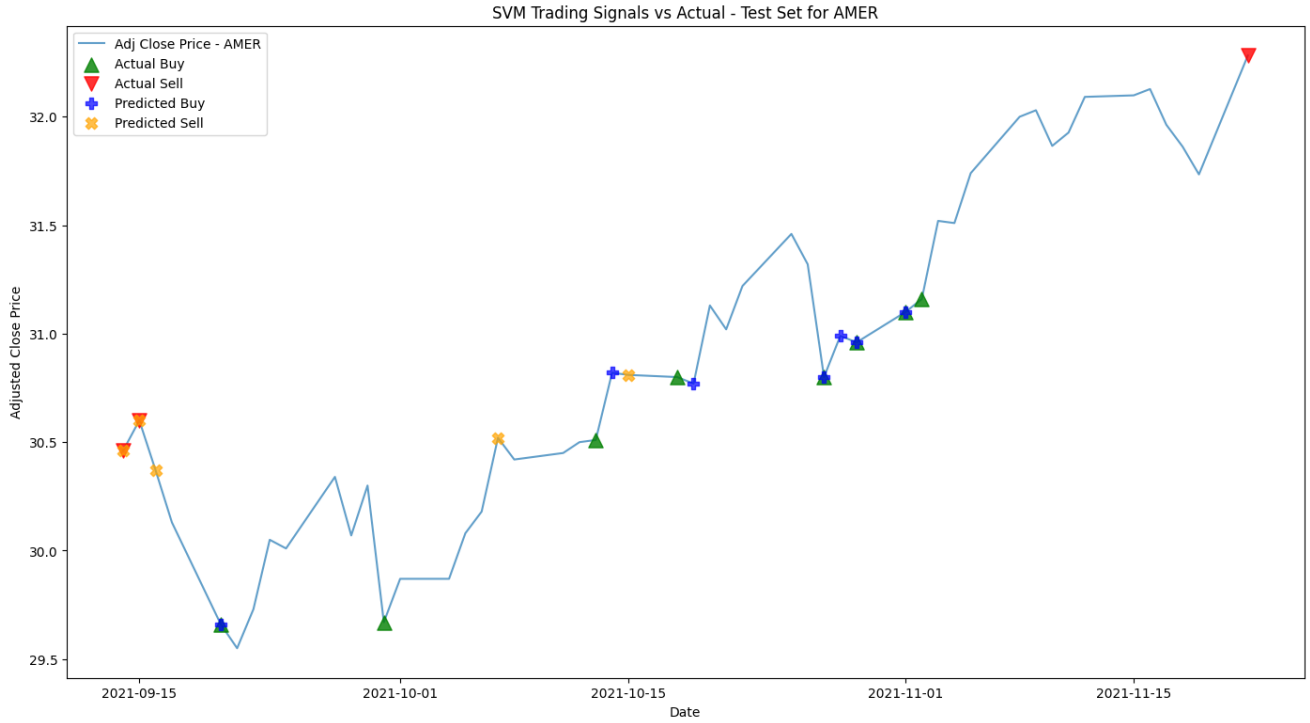
Este gráfico, muestra la gráfica de comparación entre la predicción y los datos reales que habíamos generado con nuestro modelo de predicción, junto con zonas definidas para la compra o venta de activos por los cambios porcentuales en el precio (Zona_PctChange), para el ETF QLD, utilizando una escala logarítmica en el eje vertical.

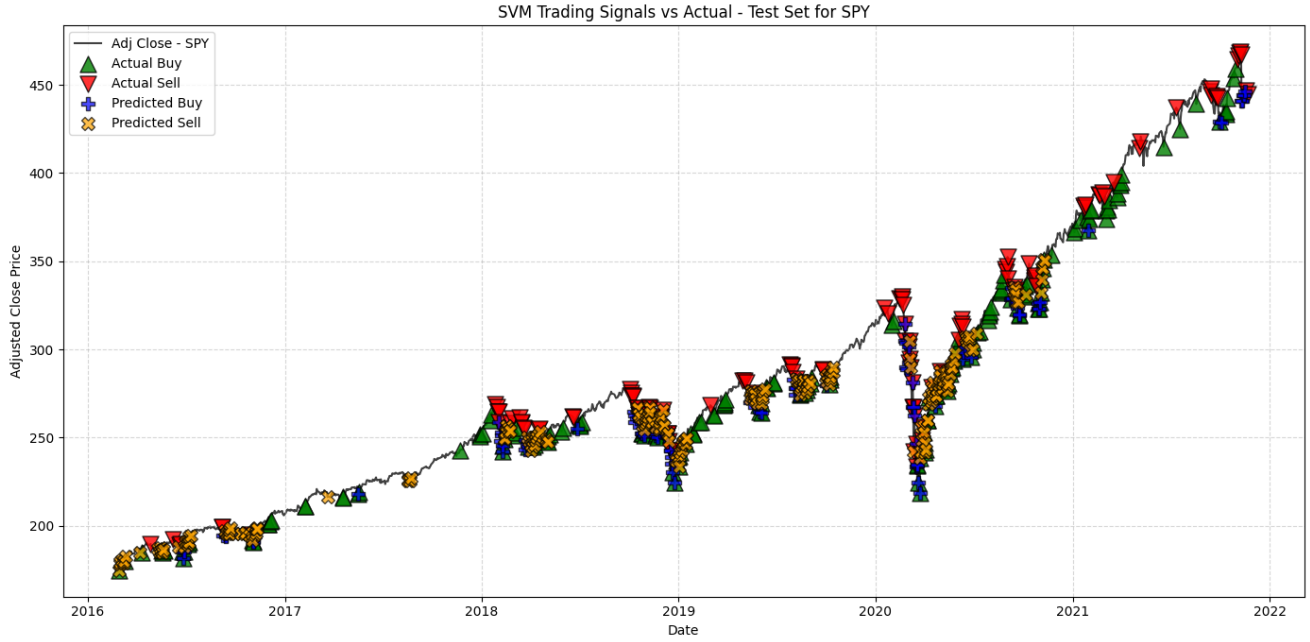
%Camb: Neutral (% Camb): Las áreas sombreadas en color gris vertical indican períodos donde los cambios porcentuales en el precio se consideraron dentro de un umbral "neutral" definido por la metodología Zona_PctChange. Durante estos períodos, el modelo no genera señales de trading demasiado claras.

%Camb: Venta (% Camb): Las áreas sombreadas en color rojo vertical señalan períodos donde los cambios porcentuales en el precio superan un umbral definido para indicar una posible señal de "Venta"

%Camb: Compra (% Camb): Las áreas sombreadas en color verde vertical identifican períodos donde los cambios porcentuales en el precio superan un umbral definido para indicar una posible señal de "Compra",

Gráfico de Análisis de los Resultados del Modelo SVM para SPY (Señales de Trading vs. Realidad)





En estos dos gráficos, se muestra la comparación de las señales de trading generadas por un modelo de Máquinas de Vectores de Soporte (SVM) con el precio real ajustado de cierre del ETF SPY en el conjunto de prueba.

Adj Close: La línea negra continua representa la evolución del precio de cierre ajustado de los 2 ETF a lo largo del tiempo en el conjunto de prueba.

Actual Buy (Triángulos Verdes): Los triángulos verdes marcados en el gráfico indican los momentos en el tiempo en el conjunto de prueba donde, retrospectivamente, habría sido una buena decisión "Comprar" el activo, basándose en movimientos de precio posteriores. Estos no son señales generadas por el modelo, sino una visualización de oportunidades de compra "reales" en el conjunto de prueba.

Actual Sell (Triángulos Rojos Invertidos): De manera similar, los triángulos rojos invertidos señalan los momentos en el conjunto de prueba donde habría sido una buena decisión "Vender" el activo, precediendo a caídas de precio. Nuevamente, estos representan oportunidades de venta "reales" en el conjunto de prueba.

Predicted Buy (Cruces Azules): Las cruces azules en el gráfico indican las señales de "Comprar" generadas por el modelo SVM en el conjunto de prueba. El objetivo es que estas señales coincidan con o precedan a los "Actual Buy" (triángulos verdes).

Predicted Sell (Equis Naranjas): Las equis naranjas representan las señales de "Vender" generadas por el modelo SVM en el conjunto de prueba. Idealmente, estas señales deberían coincidir con o preceder a los "Actual Sell" (triángulos rojos invertidos).

Este modelo, puede sernos de gran utilidad para:

- **Evaluaciones de la Precisión de las Señales:** El gráfico permite evaluar visualmente la precisión del modelo SVM al generar señales de trading

- **Identificación de Falsas Alarmas:** Observa la presencia de señales de "Predicted Buy" que no son seguidas por movimientos alcistas significativos o señales de "Predicted Sell" que no anticipan caídas de precio. Estas son "falsas alarmas" que generarían operaciones no rentables.

- **Latencia de las Señales:** Analiza si las señales predichas tienden a ocurrir demasiado tarde o demasiado pronto en relación con los movimientos reales del precio. Una señal tardía podría significar perderse parte del movimiento, mientras que una señal temprana podría resultar en una posición mantenida durante un período prolongado sin el movimiento esperado.

5. Conclusiones

En conclusión, con este proyecto hemos logrado superar la mayoría de los objetivos establecidos al enfrentar el reto de analizar y predecir el comportamiento de los Fondos Cotizados en Bolsa (ETFs) significativos en el mercado estadounidense. La etapa inicial se enfocó en reconocer los ETFs con mejor desempeño histórico reciente, para posteriormente desarrollar modelos predictivos que intentaran prever futuros cambios en el mercado. En este contexto, se obtuvieron y preprocesaron conjuntos de datos históricos significativos utilizando plataformas como Kaggle y la API de Yahoo Finance, llevando a cabo un análisis exploratorio detallado con herramientas como Pandas para descubrir características y patrones.

Un elemento clave del trabajo fue el diseño y la ejecución de una arquitectura Big Data escalable. El uso de Docker para la contenerización, la orquestación de datos con Apache NiFi, y el almacenamiento mediante Apache Cassandra para series temporales y Apache Hadoop para procesamiento a gran escala resultaron ser soluciones eficaces para gestionar el volumen y la complejidad de los datos financieros.

De igual manera, se logró el propósito de crear, capacitar y analizar el comportamiento de los mercados, abarcando Regresión Lineal, el uso e implementación de redes Neuronales LSTM y finalmente el modelo Prophet para la predicción. Los resultados obtenidos, aunque muestran espacios para mejorar y requieren un análisis más detallado, como se notó en la evaluación con Orange Datamining, confirman la implementación de estas técnicas de Machine Learning en el ámbito de los ETFs.

Finalmente, se utilizaron herramientas de visualización como Power BI y técnicas de minería de datos como Orange Datamining para exhibir los resultados y enriquecer el análisis. Aunque el objetivo principal de examinar la viabilidad de emplear estos modelos como un soporte directo en unas futuras decisiones de inversión, esta es un área que necesita validación en situaciones reales y factores adicionales, que lamentablemente no hemos sido capaces de obtener para este proyecto, aun así, este proyecto, establece las bases metodológicas y tecnológicas, alcanzando los objetivos de investigar y desarrollar las capacidades predictivas sobre ETFs estadounidenses utilizando un enfoque integral de Big Data e Inteligencia Artificial.