Prototipo para predecir el precio de Bitcoin por medio de Machine Learning haciendo uso de despliegue continuo

José Fernando Herrera Salcedo

Proyecto de grado entregado para obtener el título de
Magister en Ingeniería de Software

Dirigida por
MSc Luis Gonzalo Noreña Agudelo

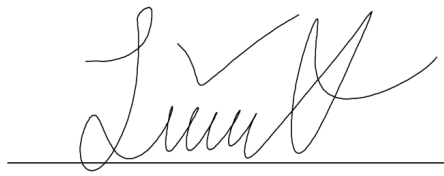
Pontificia Universidad Javeriana Cali
Facultad de Ingeniería y Ciencias
Maestría en Ingeniería de Software
Santiago de Cali
26 de mayo de 2025

Santiago de Cali, 26 de mayo de 2025.

Señores
Pontificia Universidad Javeriana Cali.
Ph.D. Luisa Rincón
Directora Maestría en Ingeniería de Software.
Cali.

Cordial Saludo.

Por medio de la presente hago constar que en mi calidad de director de trabajo de grado he revisado el proyecto titulado "Prototipo para predecir el precio de Bitcoin por medio de Machine Learning haciendo uso de despliegue continuo" realizado por el estudiante de Magister en Ingeniería de Software José Fernando Herrera Salcedo (cod: 8991552), el cual se encuentra terminado y considero que cumple con los requisitos para ser sustentado.
Atentamente,



MSc Luis Gonzalo Noreña Agudelo

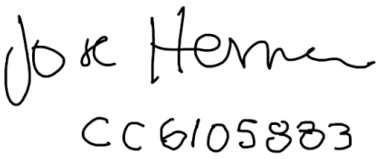
Santiago de Cali, 26 de mayo de 2025.

Señores
Pontificia Universidad Javeriana Cali
Ph.D. Luisa Rincón
Directora Maestría en Ingeniería de Software
Cali.

Cordial Saludo.

Me permito presentar a su consideración el proyecto de grado titulado "Prototipo para predecir el precio de Bitcoin por medio de Machine Learning haciendo uso de despliegue continuo" con el fin de cumplir con los requisitos exigidos por la Universidad y para que sea sometido a revisión del jurado y cumpla su aprobación, para conseguir posteriormente el título de Magister en Ingeniería de Software.

Atentamente,



José Fernando Herrera Salcedo
Código: 8991552

Ficha Resumen
Trabajo de Grado Maestría en Ingeniería de Software

TíTULO: Prototipo para predecir el precio de Bitcoin por medio de Machine Learning haciendo uso de despliegue continuo

1. Énfasis: Ingeniería de Software
2. Área de trabajo: Ingeniera de Software
3. Tipo de proyecto (Aplicado, Innovación, Investigación): Aplicado
4. Estudiante: José Fernando Herrera Salcedo
5. Correo electrónico: jjoseh1978@javerianacali.edu.co
6. Dirección y teléfono: Cra 62 # 9-123, Cali, 3504335744
7. Director: Luis Gonzalo Noreña Agudelo
8. Vinculación del director: Profesor Hora Cátedra
9. Correo electrónico del director: zalonore@gmail.com
10. Co-Director (Si aplica):
11. Grupo o empresa que lo avala (Si aplica):
12. Otros grupos o empresas:
13. Palabras clave(al menos 5): Bitcoin, Machine Learning, MLOps, Predicción, Despliegue Continuo
14. Fecha de inicio: 15 de febrero de 2025
15. Resumen:

    Este proyecto de grado aborda la volatilidad del precio de Bitcoin, que dificulta la toma de decisiones de inversión, mediante el desarrollo de un prototipo para su predicción utilizando Machine Learning (ML) y operaciones de ML (MLOps) para despliegue continuo. Dada la relevancia de Bitcoin y las dificultades que enfrentan los inversores, este trabajo busca ofrecer una solución tecnológica que apoye decisiones más informadas mediante predicciones actualizadas y transparentes. El objetivo general fue diseñar, implementar y evaluar una arquitectura MLOps en la nube (Azure) para automatizar el ciclo de vida de modelos de predicción de series temporales, lo que incluyó determinar variables relevantes (históricos de

precio/volumen de exchanges Bitstamp, Binance, Coinbase y métricas *on-chain* como cantidad movida, transacciones y dificultad de minado desde un nodo local), elaborar un conjunto de datos horario consolidado (inicialmente horario y luego agregado a frecuencia diaria para el entrenamiento de los modelos), desarrollar y comparar modelos LSTM y GRU, seleccionar el de mejor rendimiento, e implementar una aplicación web para visualizar los resultados. Como principales resultados, se implementó la arquitectura MLOps utilizando Azure Data Lake, Databricks, MLflow y Azure Container Apps; el modelo GRU fue seleccionado por su desempeño (MAPE promedio de 1,66 %), el cual se alcanzó al entrenar el modelo excluyendo las variables de la cadena; se desarrolló y desplegó una aplicación web con Streamlit mediante CI/CD (Docker, GitHub Actions, Azure Container Registry/Apps) que muestra la predicción diaria y métricas, y se logró la automatización del reentrenamiento diario del modelo. Las lecciones aprendidas destacan la importancia de la fuente y el preprocesamiento de datos (incluyendo transformaciones como diferencia logarítmica), la viabilidad y beneficios de implementar MLOps y CI/CD en la nube, y el desafío persistente que representa predecir con alta precisión la volatilidad inherente del mercado de Bitcoin.

Agradecimientos

Deseo expresar mi sincero agradecimiento a la Pontificia Universidad Javeriana Cali, y en particular a la Maestría en Ingeniería de Software, por brindar el entorno y los recursos que hicieron posible la culminación de este proyecto de grado y mi proceso de formación.

A mi familia, quiero dedicarle un profundo agradecimiento por su incondicional apoyo, ánimo constante y, sobre todo, por la enorme paciencia que demostraron a lo largo de este proceso académico. Su comprensión y aliento fueron mi mayor fortaleza en los momentos más demandantes.

De manera muy especial, agradezco a mi director MSc Luis Gonzalo Noreña Agudelo. Su guía experta, valiosos comentarios, disposición constante y rigurosidad académica fueron fundamentales para orientar el desarrollo de este trabajo y superar los desafíos encontrados.

Finalmente, completar este proyecto y la maestría ha sido una experiencia de inmenso crecimiento personal y profesional. El apoyo de mi director, el de mi familia, así como el brindado por la Pontificia Universidad Javeriana Cali y la Maestría en Ingeniería de Software, ha sido invaluable y crucial para alcanzar esta importante meta. A todos ustedes, mi más sincera gratitud.

Resumen

Este proyecto de grado enfrenta la volatilidad del precio de Bitcoin desarrollando un prototipo predictivo con Machine Learning (ML) y MLOps para despliegue continuo. Busca ofrecer una solución tecnológica para decisiones de inversión más informadas. El objetivo fue diseñar, implementar y evaluar una arquitectura MLOps en Azure para automatizar el ciclo de vida de modelos de predicción. Se determinaron variables relevantes (datos de *exchanges* y métricas *on-chain*), se elaboró un conjunto de datos diario, y se compararon modelos LSTM y GRU. Como resultados, se implementó la arquitectura MLOps (Azure Data Lake, Databricks, MLflow, Azure Container Apps); se seleccionó el modelo GRU (MAPE de 1,66 %) al excluir variables de la cadena; se desarrolló una aplicación web con Streamlit desplegada vía CI/CD, logrando el reentrenamiento diario automatizado. El trabajo destaca la importancia del preprocesamiento de datos y la viabilidad de MLOps.

Palabras Clave: Bitcoin, Machine Learning, MLOps, Predicción de Precios, Despliegue Continuo

Abstract

This degree project addresses Bitcoin's price volatility by developing a predictive prototype using Machine Learning (ML) and MLOps for continuous deployment. It aims to provide a technological solution for more informed investment decisions. The objective was to design, implement, and evaluate an MLOps architecture on Azure to automate the prediction model lifecycle. Relevant variables (exchange data and on-chain metrics) were determined, a daily dataset was created, and LSTM and GRU models were compared. Key results include the implemented MLOps architecture (Azure Data Lake, Databricks, MLflow, Azure Container Apps); the GRU model was selected (MAPE of 1,66 %) by excluding on-chain variables; a Streamlit web application was developed and deployed via CI/CD, achieving automated daily retraining. The work highlights the importance of data preprocessing and MLOps viability.LOps architecture and the performance of the selected model during the studied period.

Keywords: Bitcoin, Machine Learning, MLOps, Price Prediction, Continuous Deployment

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Definición del problema

Bitcoin nació con el objetivo de ser una moneda electrónica de pagos punto-a-punto (Nakamoto, 2008), sin necesidad de una entidad financiera que verifique la confianza. Esta moneda está basada en la tecnología *Blockchain*, donde se utilizan algoritmos de encriptación como mecanismo de protección de las transacciones, de ahí el origen del término criptomoneda.

Originalmente, Bitcoin empezó a comercializarse en foros de discusión, para que posteriormente pudiera ser comercializado en plataformas de intercambio de criptomonedas, también llamadas Exchanges de criptomonedas. Fue a través de los exchanges donde se popularizó el uso del Bitcoin, ya que estas plataformas permiten comprar criptomonedas a través de dinero fiduciario (XTB, 2022).

En un principio, este activo digital llamó la atención de particulares y, posteriormente, debido al buen comportamiento del precio, captó la atención de firmas institucionales como MicroStrategy o BlackRock (Lang and McGee, 2024). Un interés de los inversores es conocer o poder predecir el precio para encontrar los mejores momentos de compra y venta del activo. Buscando encontrar un buen precio, los inversores han utilizado métodos tales como estadística y, más recientemente, modelos de Machine Learning y Deep Learning (Zhang et al., 2024).

En un reporte realizado por Block Inc. (2022) se indica que el 51 % de los encuestados no invierten en Bitcoin debido a que no lo conocen bien. Además, el estudio de Auer et al. (2022) revela que 3/4 de los inversores en criptomonedas han perdido dinero. Una de sus causas es la de no poseer la habilidad de analizar el mercado (Beattie, 2014), como también el desconocimiento del comportamiento de este, ya que terminan vendiendo por debajo del precio de compra.

Actualmente, existen varias aplicaciones que permiten hacer la proyección del precio de manera gratuita o de pago, por ejemplo *coindex*, *30rates*, *lookintobitcoin*, *cryptopredictions* o *walletinvestor*, pero no indican qué se está usando para hacer la proyección del precio como tampoco indican el margen de error del precio proyectado respecto al precio real. Al revisar estas herramientas, tampoco se logra ver que logren predecir el precio de Bitcoin en temporalidades menores de un día.

Muchos inversores particulares no tienen la posibilidad de hacer análisis predictivos porque una parte de ellos realizan este tipo de inversiones como una fuente secundaria de ingresos. Contando con la falta de experiencia para determinar si es un buen momento de compra o venta.

1.1.1. Planteamiento del problema

Bitcoin se distingue como un activo escaso, estableciendo un suministro máximo de 21 millones de monedas, acompañado por un mecanismo de distribución claro y programado, que facilita la anticipación precisa del número de monedas que serán emitidas.

El Bitcoin, como activo digital y primera criptomoneda descentralizada, ha generado un interés creciente como inversión. Sin embargo, la naturaleza fluctuante de su precio presenta un reto significativo para aquellos inversores que buscan oportunidades de mercado. La habilidad de prever las variaciones en el precio del Bitcoin podría dar como resultado decisiones de inversión más informadas y potencialmente rentables.

Recientemente, Machine Learning ha emergido como un conjunto de técnicas prometedoras para el análisis del precio de Bitcoin. Estas técnicas tienen el potencial de identificar patrones complejos de grandes conjuntos de datos, lo que evidenció potencial para predecir el comportamiento del precio. Sin embargo, los resultados de estos modelos pueden verse comprometidos por diversas razones, como, por ejemplo, si no se mantienen actualizados con los datos más recientes, por datos faltantes o por la selección de características irrelevantes.

La presente investigación desarrolla la creación de un prototipo para predecir el precio de Bitcoin por medio de Machine Learning haciendo uso de despliegue continuo, con el objetivo de mejorar la capacidad de predicción del precio del Bitcoin.

1.1.2. Sistematización

- ¿Cómo predecir el precio del Bitcoin utilizando Machine Learning a través de un despliegue continuo?
- ¿Será que se puede predecir el precio para cinco minutos después, una hora después, un día después o una semana después?
- ¿La misma estrategia sería útil para predecir el precio a corto, mediano y largo plazo?
- ¿Cuáles son las variables que intervienen en la predicción del precio de Bitcoin?
- ¿Cuáles modelos de Machine Learning se podrían utilizar para hacer la predicción del precio?
- ¿Cuál metodología de despliegue continuo puede ser útil para analizar y predecir el precio de Bitcoin?

1.2. Objetivos del proyecto

1.2.1. Objetivo General

Diseñar, implementar y evaluar una arquitectura en la nube para la automatización del entrenamiento y redespliegue de un modelo de series temporales para la predicción del precio de Bitcoin, utilizando la metodología MLOps, así como diseñar e implementar una aplicación web en la nube que permita la visualización de la gráfica del precio de Bitcoin y su proyección.

1.2.2. Objetivos específicos

- Determinar las variables relevantes para la predicción del precio de Bitcoin y elaborar un dataset depurado y validado para el entrenamiento y evaluación del modelo.
- Diseñar, implementar y evaluar una arquitectura en la nube que permita la automatización del entrenamiento y redespliegue de un modelo de Machine Learning a través de MLOps registrando experimentos, métricas, artefactos y monitoreo.
- Desarrollar y entrenar dos modelos de Machine Learning distintos empleando técnicas de series temporales.
- Evaluar y comparar los resultados de los dos modelos. Seleccionar la técnica con rendimiento superior para la predicción del precio de Bitcoin.
- Diseñar e implementar una aplicación web de visualización del precio de Bitcoin y su proyección, y llevar a cabo la evaluación de dicha arquitectura.

1.3. Delimitaciones y alcances

El presente proyecto se enfoca en el diseño, implementación y evaluación de un prototipo para la predicción del precio de Bitcoin (BTC) utilizando técnicas de Machine Learning y una arquitectura MLOps en la nube. A continuación, se detallan los alcances y las delimitaciones del trabajo realizado:

1.3.1. Alcances

Modelo Predictivo: Se desarrollaron y evaluaron modelos de redes neuronales recurrentes (LSTM y GRU) para predecir el precio de Bitcoin a un paso adelante, considerando diversas variables. Se emplearon datos de precio y volumen de exchanges (Bitstamp, Binance y Coinbase) y métricas de la cadena de bloques (bitcoins movidos, transacciones y dificultad de minado), agregadas a frecuencia diaria para el entrenamiento. El modelo GRU, entrenado sin datos de la cadena y con un MAPE de 1,66 %, fue seleccionado para el prototipo.

Arquitectura MLOps: Se diseñó e implementó una arquitectura en la plataforma Azure que incluye Azure Data Lake Storage Gen2 para el almacenamiento de datos consolidados, Azure Databricks para el procesamiento y entrenamiento de modelos, MLflow para el registro y seguimiento de experimentos (parámetros, métricas como MAPE, y artefactos como modelos serializados y escaladores), y Azure Container Apps para el despliegue.

Automatización: Se implementó un proceso automatizado para el reentrenamiento diario del modelo predictivo seleccionado (GRU) y la actualización de la versión registrada en MLflow como 'Producción'. El seguimiento de experimentos con MLflow está integrado en el proceso de entrenamiento del modelo.

Visualización: Se desarrolló una aplicación web utilizando Streamlit que muestra la predicción del precio de Bitcoin para el día siguiente, la métrica clave del rendimiento del modelo (MAPE del

conjunto de prueba) y un gráfico comparativo de las predicciones frente a los precios históricos reales.

Despliegue Continuo (Aplicación Web): La aplicación web se despliega mediante un flujo de Integración Continua y Entrega Continua (CI/CD) utilizando Docker, GitHub Actions, Azure Container Registry y Azure Container Apps, asegurando la actualización automática de la interfaz tras cambios en su código fuente.

Evaluación: Se realizó una evaluación cuantitativa del modelo GRU seleccionado sobre un conjunto de prueba definido (1 de diciembre de 2024 al 9 de marzo de 2025), utilizando la métrica MAPE. Se verificó la funcionalidad de los elementos de la arquitectura MLOps y de la aplicación web desplegada.

1.3.2. Delimitaciones

Activo Financiero: El análisis y la predicción se centran exclusivamente en Bitcoin (BTC). No se incluyen otras criptomonedas ni activos financieros.

Horizonte de Predicción: El prototipo genera predicciones a corto plazo, específicamente para el día siguiente (una predicción diaria a un paso adelante). No se aborda la predicción a mediano o largo plazo (semanas, meses).

Modelos y Variables: La exploración de modelos se limitó a LSTM y GRU. No se experimentó con otras arquitecturas de Machine Learning o modelos estadísticos. El conjunto de variables de entrada se restringió a las fuentes de datos de *exchanges* y *blockchain* mencionadas; no se incorporaron otros factores como análisis de sentimiento, noticias o indicadores macroeconómicos.

Naturaleza del Prototipo: El sistema desarrollado es un prototipo funcional. No pretende ser una herramienta de inversión infalible ni ofrecer asesoramiento financiero. La precisión de las predicciones está sujeta a la volatilidad inherente del mercado de Bitcoin y a las limitaciones del modelo, que puede no anticipar movimientos abruptos o cambios de tendencia no reflejados en los datos históricos recientes.

Periodo de Evaluación: La evaluación del rendimiento del modelo se basa en un periodo histórico específico. El resultado obtenido de MAPE 1,66 % reflejan el desempeño en dicho periodo y no garantizan un rendimiento similar en el futuro bajo condiciones de mercado diferentes.

Robustez y Seguridad: Si bien se implementaron prácticas estándar en el desarrollo, aspectos avanzados de robustez, tolerancia a fallos a gran escala y auditorías de seguridad exhaustivas, propias de un sistema en producción comercial, quedan fuera del ámbito de este proyecto de maestría.

1.4. Justificación del trabajo de grado

El auge de las criptomonedas ha transformado el panorama financiero global, y dentro de este ecosistema emergente, Bitcoin se destaca como la criptomoneda con mayor capitalización de mercado, representando aproximadamente el 60 % del valor total del mercado cripto en noviembre de 2024 (Torpey, 2024). Sin embargo, entre muchas variables, la volatilidad inherente a Bitcoin dificulta la predicción de su precio, lo que representa un obstáculo para la toma de decisiones

informadas en este mercado. En este contexto, esta investigación aborda la creación de un sistema para la predicción del precio de Bitcoin que apoye la toma de decisiones de inversores y traders que operan con esta criptomoneda.

El uso de Machine Learning para predecir precios en mercados financieros ha demostrado resultados prometedores. Por ejemplo, en *Forecasting Cryptocurrency Prices Using Ensembles-Based Machine Learning Approach* (Derbentsev et al., 2020) se reporta un MAPE (Mean Absolute Percentage Error) de 2,96 % en la predicción de precios de criptomonedas. No obstante, la precisión en la predicción del precio de Bitcoin sigue siendo un desafío, como se evidencia en *A comparison between ARIMA, LSTM, and GRU for time series forecasting* (Yamak et al., 2019), donde se señala que la proyección del precio de Bitcoin no fue precisa, posiblemente debido a la limitada cantidad de datos utilizados.

Este trabajo logra superar estas limitaciones mediante el desarrollo de un prototipo de sistema en la nube, alojado en Azure, para la predicción del precio del Bitcoin basado en Machine Learning. El sistema se diferencia por la integración de un conjunto de datos extenso y actualizado, que se obtiene de la API de tres *exchanges* y abarca el periodo aproximado de diez años, incluyendo variables como precio de apertura, cierre, máximo, mínimo, volumen, como también datos de la cadena; una arquitectura automatizada con CI/CD que permite el reentrenamiento y despliegue continuo del modelo GRU, y una aplicación web que facilita el acceso a las predicciones.

La adopción de una arquitectura en la nube proporciona la escalabilidad y disponibilidad necesarias para soportar un alto número de usuarios concurrentes. Además, la integración de prácticas de CI/CD, como se describe en *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation* (Humble and Farley, 2010), incrementa la eficiencia operativa y reduce los riesgos asociados al desarrollo y despliegue de software, permitiendo una respuesta rápida a los cambios en el mercado y asegurando que el sistema se mantenga actualizado y preciso ante la dinámica cambiante del mercado.

Al desarrollar un sistema de predicción del precio de Bitcoin, es importante establecer expectativas realistas en cuanto a la precisión del modelo. Aunque predecir el futuro es un desafío, especialmente en este mercado dado por su volatilidad, podemos tomar como referencia el artículo *Bitcoin Cryptocurrency Price Prediction Using IFA-BiLSTM* que reporta un MAPE de 3,48 % y una precisión de 96,52 % (Pratama, 2024). Sin embargo, este rendimiento se basa en datos históricos y puede no ser representativo del comportamiento futuro del mercado.

En conclusión, esta investigación contribuye al área de la economía digital y las finanzas mediante la creación de un sistema predictivo para el valor de Bitcoin. El sistema desarrollado, basado en Machine Learning, una arquitectura en la nube sólida y prácticas de CI/CD, aporta en la toma de decisiones de inversores, *traders* y empresas en el mercado de criptomonedas, ofreciendo una ventaja en un entorno dinámico y volátil.

1.5. Metodología de la investigación

Para la gestión del presente proyecto, desarrollado por una única persona, se adoptó un enfoque inspirado en los principios de la metodología ágil Scrum. Esta aproximación facilitó una ejecución

organizada, iterativa y adaptable a los hallazgos y desafíos surgidos. La planificación y el seguimiento de las tareas se centralizaron en un tablero de Jira configurado al estilo Scrum, donde cada objetivo principal del proyecto se definió como una épica.

- **Estructura del Equipo y Roles (Adaptada al desarrollo individual):**
  - **Product Owner:** José Herrera. Responsable de definir los objetivos de la investigación, priorizar las tareas (Product Backlog) y aceptar o rechazar los resultados de cada iteración (Sprint).
  - **Scrum Master (facilitador del proceso individual):** José Herrera. Responsable de adaptar y seguir el proceso inspirado en Scrum, remover obstáculos y garantizar la adherencia a las prácticas ágiles seleccionadas.
  - **Equipo de Desarrollo (individual):** José Herrera. Responsable de realizar todas las tareas de investigación y desarrollo.
- **Adaptación de los Eventos de Scrum para la Gestión Individual:**
  - **Sprint:** Se establecieron Sprints con una duración fija de dos semanas, cada uno enfocado en la generación de un incremento funcional y tangible del producto.
  - **Planificación del Sprint:** Al comenzar cada Sprint, se efectuaba una planificación personal para determinar las labores a ejecutar, seleccionando actividades del backlog del producto que estuvieran asociadas a una Épica concreta.
  - **Seguimiento Continuo (análogo al Daily Scrum):** En lugar de una reunión formal diaria, se efectuó un seguimiento personal y continuo del progreso, identificando y abordando impedimentos de manera proactiva.
  - **Revisión del Sprint:** Al finalizar cada Sprint, se revisó el incremento de producto generado para evaluar el avance con respecto a los objetivos de la iteración y del proyecto en general.
  - **Retrospectiva del Sprint (reflexión personal):** Tras finalizar cada iteración, se llevaba a cabo una reflexión individual orientada a identificar oportunidades de optimización en el proceso de trabajo personal y en la planificación de los Sprints subsiguientes.
- **Artefactos Scrum y Gestión en Jira:**
  - **Product Backlog:** Listado ordenado por prioridad de todas las funcionalidades, tareas de investigación, mejoras y correcciones requeridas para el proyecto. Este backlog se gestionó en Jira, donde cada objetivo específico del proyecto se constituyó como una épica, la cual contenía las tareas asociadas (ver Figura 1.1).
  - **Sprint Backlog:** Conjunto de tareas del Product Backlog seleccionadas para ser desarrolladas durante un Sprint específico, gestionadas dentro del tablero de Jira.

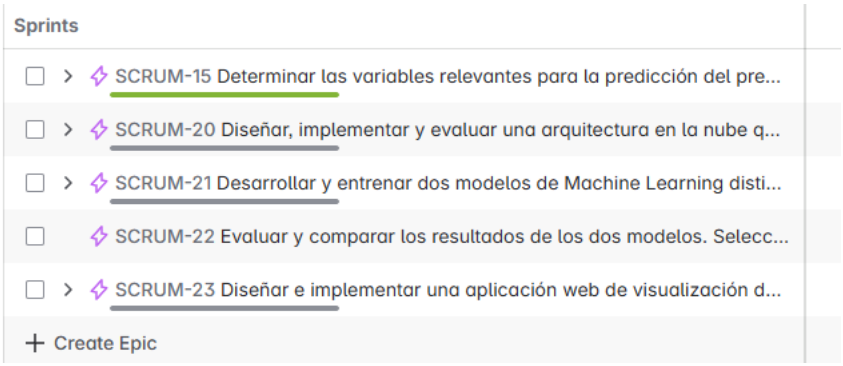


Figura 1.1: Tablero de Sprints en Jira mostrando objetivos del proyecto organizados como Épicas (ej. SCRUM-15). (Elaboración propia)

- **Incremento de Producto:** Fue la acumulación de todos los elementos del Product Backlog que se habían completado durante el Sprint actual y todos los Sprints previos, resultando en la evolución del prototipo.

A partir de la adaptación de los principios Scrum para la gestión del proyecto, este enfoque se aplicó para organizar el diseño y la implementación de los pipelines de MLOps. La metodología MLOps, a su vez, se concentró específicamente en las fases del ciclo de vida del modelo de Machine Learning.

1. **Diseño e Implementación de Pipelines de MLOps con Principios Scrum:**
   - **Planificación del Pipeline con Principios Scrum y Jira:**
     - Se utilizaron los principios de Scrum y el tablero de Jira para organizar la planificación, diseño e implementación de los pipelines de MLOps. Se definieron los requisitos y el alcance mediante épicas (objetivos del proyecto) y tareas específicas en el *backlog* de producto en Jira.
     - Se trabajó en iteraciones (Sprints) para abordar los elementos del *backlog*, con el objetivo de construir progresivamente el pipeline para la experimentación y operacionalización de modelos de Machine Learning, enfocándose en las tareas de la Épica correspondiente a cada Sprint.
   - **Desarrollo Iterativo de Pipelines:**
     - Durante cada Sprint, el esfuerzo se concentró en las tareas asociadas a la Épica (objetivo específico) seleccionada para esa iteración. Estas tareas contribuyeron al desarrollo progresivo o la mejora de los componentes del pipeline de MLOps, así como a su integración con las herramientas y servicios en la nube cuando correspondía al objetivo del Sprint.
     - Al final de cada Sprint, la revisión y ajuste de los avances permitieron adaptar el trabajo a los cambios o nuevos requisitos surgidos durante el desarrollo.

2. **Metodología MLOps:**
   - **Experimentación y Desarrollo del Modelo:**
     - Siguiendo los principios de MLOps, se gestionó la experimentación de modelos de Machine Learning, empleando técnicas de versionado y evaluación para asegurar la calidad del modelo desde las primeras etapas.
   - **Validación y Evaluación de Modelos:**
     - Se realizó la evaluación del rendimiento de los modelos como parte del proceso MLOps, utilizando métricas estándar (ej. MAPE, RMSE) sobre conjuntos de datos de prueba para verificar su precisión. Los resultados de estas evaluaciones fueron registrados y considerados antes de la designación de un modelo para producción.
3. **Integración Continua y Despliegue Continuo (CI/CD):**
   - Se utilizó un pipeline para automatizar parte del ciclo de vida del modelo, incluyendo el reentrenamiento y el registro de nuevas versiones, así como el despliegue de la aplicación web asociada en un entorno de nube.
   - Se aseguró que el pipeline incluyera pasos para el reentrenamiento y actualización de modelos, aprovechando la escalabilidad y flexibilidad de la nube.
4. **Monitoreo y Operaciones en Producción:**
   - Se establecieron prácticas de monitoreo del rendimiento del modelo y mantenimiento operacional para asegurar su fiabilidad en el entorno de producción.
   - Se utilizaron las capacidades de las herramientas en la nube y MLflow para el seguimiento de métricas clave del modelo.
5. **Revisión y Mejora Continua (Adaptación de Scrum y MLOps):**
   - **Revisión del Sprint y Feedback (Autoevaluación):**
     - Al finalizar cada Sprint, se revisaron los avances concernientes a la Épica (objetivo específico) abordada, evaluando la contribución al desarrollo general del proyecto y, específicamente, al pipeline de MLOps cuando era aplicable. El feedback provino principalmente de la autoevaluación y la contrastación con los objetivos planteados y el asesoramiento del director del proyecto.
     - Las reflexiones personales (análogas a las retrospectivas) al final de los Sprints sirvieron para detectar posibilidades de mejora en el proceso de desarrollo individual.

1.6. Resultados obtenidos

Como culminación de este proyecto de grado, se alcanzaron los objetivos planteados, materializándose en los siguientes resultados principales:

- **Dataset Consolidado y Variables Relevantes:** Se determinaron e integraron variables clave para la predicción, incluyendo datos históricos de precio (apertura, máximo, mínimo, cierre) y volumen de los exchanges Bitstamp, Binance y Coinbase, junto con métricas *on-chain* (cantidad de bitcoins movidos, número de transacciones y dificultad de minado) extraídas de un nodo Bitcoin local. Esta información se procesó para generar un dataset consolidado, inicialmente a frecuencia horaria y posteriormente agregado a una frecuencia diaria para el entrenamiento de los modelos, almacenado en Azure Data Lake Storage Gen2.
- **Arquitectura MLOps Funcional:** Se diseñó e implementó una arquitectura MLOps en la plataforma Azure, utilizando Azure Data Lake Storage Gen2, Azure Databricks para el procesamiento y entrenamiento, MLflow para el registro y seguimiento de experimentos (parámetros, métricas y artefactos como modelos y escaladores), y Azure Container Apps como entorno de despliegue. Crucialmente, se logró la automatización del reentrenamiento diario del modelo seleccionado, actualizando la versión designada como 'Producción' en MLflow.
- **Modelo Predictivo GRU Seleccionado:** Se desarrollaron y entrenaron dos modelos de redes neuronales recurrentes para series temporales (LSTM y GRU) que utilizan múltiples variables de entrada.. Tras una evaluación comparativa que incluyó el análisis del impacto de los datos de la cadena, el modelo GRU, utilizando exclusivamente datos históricos de precio y volumen de exchanges, fue seleccionado por demostrar un rendimiento predictivo superior. En el conjunto de prueba definido (1 de diciembre de 2024 al 9 de marzo de 2025), este modelo obtuvo un Error Porcentual Absoluto Medio (MAPE) promedio de 1,66 %.
- **Aplicación Web de Visualización Desplegada:** Se diseñó e implementó una aplicación web interactiva utilizando Streamlit. Esta aplicación visualiza la predicción diaria del precio de Bitcoin, métricas clave del rendimiento del modelo y una gráfica comparativa con precios históricos. La aplicación fue desplegada en Azure Container Apps mediante un flujo de Integración Continua y Entrega Continua (CI/CD) configurado con Docker, GitHub Actions y Azure Container Registry, asegurando su disponibilidad y actualización automática.

1.7. Cláusula de exención de responsabilidad

Es importante señalar que los resultados obtenidos por el modelo predictivo deben interpretarse dentro del contexto de la volatilidad inherente al mercado de Bitcoin. La capacidad de predicción de este modelo, aunque respaldada por metodologías de Machine Learning y MLOps, no garantiza la precisión absoluta ni asegura ganancias o pérdidas en inversiones reales. Los resultados presentados en este estudio tienen fines académicos y no constituyen asesoramiento financiero.

Capítulo 2

Marco de referencia

2.1. Marco Teórico

2.1.1. Fundamentos de Bitcoin

2.1.1.1. Tecnología *blockchain*

En el contexto de Bitcoin, la tecnología *blockchain* funciona como un registro digital distribuido que anota todas las operaciones realizadas con esta criptomoneda. Las transacciones individuales son convalidadas por miembros de la red mediante un procedimiento denominado minería (Antonopoulos and Harding, 2023) y, una vez que obtienen confirmación, estas operaciones se consolidan en agrupaciones denominadas "bloques". Dichos bloques se organizan en una secuencia temporal, enlazándose cada uno con su predecesor mediante un *hash* criptográfico, lo que origina el concepto de *blockchain* o cadena de bloques.

Los bloques en la red Bitcoin consisten en conjuntos de transacciones ya validadas, los cuales se integran a la cadena de bloques mediante enlaces criptográficos en intervalos de aproximadamente diez minutos (The Investopedia Team, 2023). Este mecanismo da como resultado un historial de todas las operaciones registradas en la cadena, que es inalterable y de acceso público, como se ilustra en la Figura 2.1.

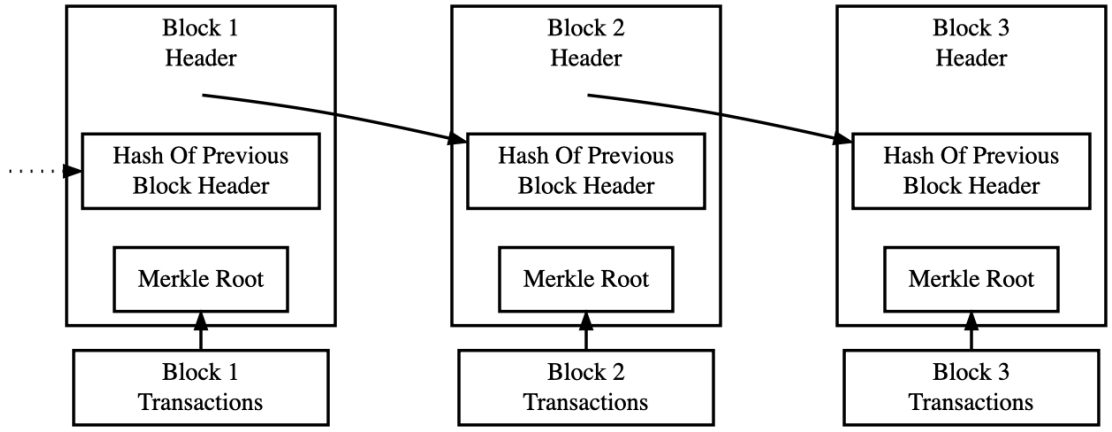


Figura 2.1: *Blockchain* de Bitcoin simplificada. (Bitcoin.org, 2020a)

Cada bloque contiene (Bitcoin.org, 2020b):

- **Cabecera del bloque:**
  - **Versión:** Especifica la versión del protocolo usada para generar el bloque.
  - **Enlace hash al bloque precedente:** Una referencia criptográfica (hash) al bloque anterior en la cadena, lo que asegura la continuidad y evita la alteración de los bloques anteriores.
  - **Raíz de Merkle:** Un hash que representa todas las transacciones incluidas en el bloque. La raíz de Merkle es el resultado final de un árbol de Merkle, que es una estructura utilizada para verificar eficientemente la integridad de las transacciones en el bloque.
  - **Marca de tiempo (timestamp):** La hora en la que se añadió el bloque a la cadena.
  - **nBits:** Un campo que representa la dificultad actual de la prueba de trabajo requerida para minar un nuevo bloque.
  - **Nonce:** Un valor arbitrario que se utiliza en la minería de criptomonedas. Los mineros cambian el nonce hasta que el hash del bloque cumple con los requisitos de dificultad de la red.
- **Cuerpo del bloque:**
  - **Contador de transacciones:** Un número que indica la cantidad de transacciones incluidas en el bloque.
  - **Transacciones:** Una lista de todas las transacciones confirmadas durante el período de tiempo que cubre el bloque. Cada transacción incluye detalles como las direcciones de origen y destino, la cantidad de Bitcoin transferida y las firmas digitales que verifican la autenticidad de la transacción.

Los bloques son creados por los mineros de Bitcoin, que utilizan la potencia de cálculo para resolver un problema criptográfico complejo (prueba de trabajo) (Nakamoto, 2008). El primer minero en resolver el problema y validar todas las transacciones del bloque es recompensado con una cantidad fija de bitcoins (la recompensa por bloque) más las comisiones de transacción incluidas en ese bloque (Nakamoto, 2008). Una vez que un bloque es minado y añadido a la cadena, se comparte con todos los nodos de la red para su validación y se convierte en parte del registro público e inmutable de la *blockchain*.

El *halving* de Bitcoin es un mecanismo automatizado de ajuste de la emisión incorporado en el algoritmo del protocolo de Bitcoin, ver Figura 2.2, diseñado para controlar la inflación de esta criptomoneda, limitando su oferta. Este evento se produce cada 210.000 bloques minados, lo que equivale aproximadamente cada cuatro años. En cada halving, la recompensa de los mineros de Bitcoin por bloque minado se divide a la mitad.

El acceso programático o para análisis a la información contenida en este libro contable se puede lograr a través de distintas metodologías. Estas incluyen el uso de las API ofrecidas por proveedores de servicios especializados en datos *on-chain*, o bien, mediante la operación directa de un nodo de la red Bitcoin (utilizando software como Bitcoin Core). Operar un nodo local permite consultas

Halving Date	Block Number	Block Subsidy	BTC / Day	Total Supply	% Mined
3 Jan 2009	0	50	7 200	0	0%
28 Nov 2012	210 000	25	3 600	10 500 000	50%
9 Jul 2016	420 000	12,5	1 800	15 750 000	75%
11 May 2020	630 000	6,25	900	18 375 000	88%
2024	840 000	3,125	450	19 687 500	94%
2028	1 050 000	1,5625	225	20 343 750	97%
2032	1 260 000	0,78125	112,5	20 671 875	98,4%
2036	1 470 000	0,390625	56,25	20 835 937,5	99,2%
2040	1 680 000	0,1953125	28,125	20 917 968,75	99,6%
2044	1 890 000	0,09765625	14,0625	20 958 984,375	99,8%
2048	2 100 000	0,04882813	7,03125	20 979 492,1875	99,9%
2052	2 310 000	0,02441406	3,515625	20 989 746,09375	99,95%
2056	2 520 000	0,01220703	1,7578125	20 994 873,046875	99,98%
2060	2 730 000	0,00610352	0,87890625	20 997 436,5234375	99,988%
2064	2 940 000	0,00305176	0,43945313	20 998 718,2617188	99,994%
2068	3 150 000	0,00152588	0,21972656	20 999 359,1308594	99,997%
2072	3 360 000	0,00076294	0,10986328	20 999 679,5654297	99,998%
2076	3 570 000	0,00038147	0,05493164	20 999 839,7827148	99,9992%
2080	3 780 000	0,00019073	0,02746582	20 999 919,8913574	99,9996%
2084	3 990 000	0,00009537	0,01373291	20 999 959,9456787	99,9998%
2088	4 200 000	0,00004768	0,00686646	20 999 979,9728394	99,999905%
2092	4 410 000	0,00002384	0,00343323	20 999 989,9864197	99,999952%
2096	4 620 000	0,00001192	0,00171661	20 999 994,9932098	99,999976%
2100	4 830 000	0,00000596	0,00085831	20 999 997,4966049	99,999988%
2104	5 040 000	0,00000298	0,00042915	20 999 998,7483025	99,999994%
2108	5 250 000	0,00000149	0,00021458	20 999 999,3741512	99,999997%
2112	5 460 000	0,00000075	0,00010729	20 999 999,6870756	99,999999%
2116	5 670 000	0,00000037	0,00005364	20 999 999,8435378	99,9999993%
2120	5 880 000	0,00000019	0,00002682	20 999 999,9217689	99,9999996%
2124	6 090 000	0,00000009	0,00001341	20 999 999,9608844	99,9999998%
2128	6 300 000	0,00000005	0,00000671	20 999 999,9804422	99,99999992%
2132	6 510 000	0,00000002	0,00000335	20 999 999,9902211	99,99999997%
2136	6 720 000	0,00000001	0,00000168	20 999 999,9951106	99,99999999%
2140	6 930 000	0,00000000	0	20 999 999,9975553	100%

Figura 2.2: Emisión de bitcoins en cada *halving* (Soni, 2024).

directas a la cadena vía interfaces como RPC (*Remote Procedure Call*), aunque esta alternativa demanda recursos considerables de almacenamiento y tiempo para la sincronización inicial.

2.1.1.2. Bitcoin como Activo de Inversión

El mercado de Bitcoin es global y opera las 24 horas del día. Esto se debe a la naturaleza descentralizada de Bitcoin. Como activo digital, Bitcoin se ha convertido en un foco de inversión tanto para particulares como para instituciones financieras, quienes buscan diversificar sus carteras. No obstante, su alta volatilidad presenta desafíos significativos para la predicción de su precio y la toma de decisiones de inversión (Baur et al., 2018). Este proyecto se centra en abordar dicha volatilidad mediante la predicción de su precio.

2.1.2. Machine Learning

2.1.2.1. Principios básicos de Machine Learning

La disciplina del *Machine Learning* consiste en desarrollar sistemas que, basándose en datos, adquieren conocimiento y perfeccionan su habilidad para realizar predicciones (Serrano, 2021). Mediante el uso de modelos matemáticos, esta técnica analiza datos para discernir patrones, lo que a su vez faculta a los sistemas computacionales para llevar a cabo tareas determinadas. En el presente proyecto, se ha optado por la modalidad de aprendizaje supervisado.

- **Aprendizaje supervisado:** Se realiza el entrenamiento del modelo con datos etiquetados, es decir, cada entrada de datos viene acompañada de la salida correcta. El objetivo es aprender una función que logre encontrar las salidas correctas dependiendo de las entradas (Géron, 2022).

2.1.2.2. Evaluación y validación de modelos predictivos en Machine Learning

Son procesos para determinar el rendimiento y la fiabilidad de un modelo al realizar predicciones sobre datos no vistos previamente (Géron, 2022). Estos procesos ayudan a asegurar que el modelo generaliza bien y que sus predicciones son precisas.

- **Evaluación de Modelos:** La evaluación de modelos cuantifica su desempeño usando métricas pertinentes al problema. Para regresión (ej. predicción de precios), RMSE y MSE miden la magnitud del error (Géron, 2022), y el MAPE indica el error como porcentaje del valor real (Lazzeri, 2020). Todas estas métricas se calculan con datos de prueba no vistos durante el entrenamiento.
- **Validación de Modelos:** La validación de modelos se refiere al proceso que asegura que el modelo funciona bien con datos no usados previamente. Una técnica fundamental es:
  - **División de Datos:** Los datos se separan en conjuntos de entrenamiento (el modelo se entrena con estos datos), validación (usada para ajustar los hiperparámetros del modelo) y prueba (usada para evaluar el desempeño del modelo) (de Prado, 2018).

  Consideraciones:

  - **Sobreajuste (Overfitting):** El sobreajuste implica que un modelo se vuelve incapaz de hacer predicciones precisas con datos diferentes a los de entrenamiento. Esto se evidencia cuando el rendimiento en el grupo de entrenamiento es mucho mejor que en el grupo de prueba (Géron, 2022).
  - **Subajuste (Underfitting):** Pasa cuando un modelo es simple y no logra capturar la complejidad oculta de los datos, dando como resultado un desempeño deficiente tanto en el entrenamiento como en los datos de prueba (Géron, 2022).

- **Selección de Modelo:** La validación y evaluación de modelos contribuyen a la selección del modelo más adecuado entre un conjunto de modelos candidatos, basado en su rendimiento (Géron, 2022).
- **Ajuste de Hiperparámetros:** Los hiperparámetros son configuraciones externas al modelo que deben ser definidas por el practicante y pueden tener un gran impacto en el rendimiento del modelo (Géron, 2022).

2.1.3. Análisis de precio

2.1.3.1. Análisis a través de herramientas gráficas

Una de las herramientas más utilizadas actualmente para hacer análisis de precio de Bitcoin es TradingView, como se ilustra en la Figura 2.3.

Figura 2.3: Precio de Bitcoin en TradingView. (Elaboración propia)

TradingView es una plataforma ampliamente utilizada para el análisis técnico de diversos activos financieros, como criptomonedas, divisas, acciones, índices y materias primas. Esta plataforma se basa en el uso de indicadores técnicos, que son herramientas matemáticas y estadísticas que se aplican a datos históricos de precios y/o volúmenes para identificar patrones y tendencias (Murphy, 1999). Un ejemplo común es la media móvil simple (SMA), la cual calcula el promedio aritmético del valor de un activo durante un período determinado (Huang and Petukhina, 2022).

Los datos que se visualizan en esta aplicación, como se mencionaba en el párrafo anterior, son el histórico del activo. Estos datos provienen de diferentes *exchanges* o brókeres, que son las plataformas que permiten hacer el intercambio del activo a una moneda fíat seleccionada, por ejemplo: el Bitcoin versus el Dólar, el S&P 500 versus el Dólar, el oro versus el Dólar, el Euro versus el Dólar.

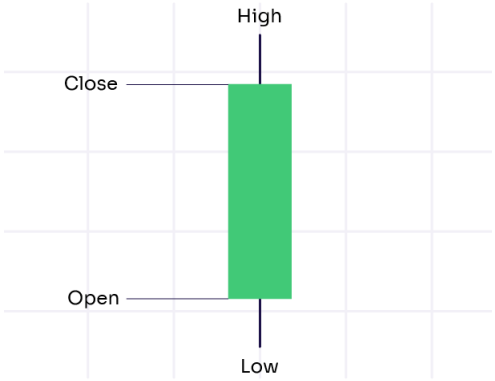


Figura 2.4: Candlestick (Mogal, 2024).

Date	Open	High	Low	Close	Adj Close	Volume
2022-01-01	46311.746094	47827.312500	46288.484375	47686.812500	47686.812500	24582667004
2022-01-02	47680.925781	47881.406250	46856.937500	47345.218750	47345.218750	27951569547
2022-01-03	47343.542969	47510.726562	45835.964844	46458.117188	46458.117188	33071628362
2022-01-04	46458.851562	47406.546875	45752.464844	45897.574219	45897.574219	42494677905
2022-01-05	45899.359375	46929.046875	42798.222656	43569.003906	43569.003906	36851084859
2022-01-06	43565.511719	43748.718750	42645.539062	43160.929688	43160.929688	30208048289
2022-01-07	43153.570312	43153.570312	41077.445312	41557.902344	41557.902344	84196607520
2022-01-08	41561.464844	42228.941406	40672.277344	41733.941406	41733.941406	28066355845

Figura 2.5: Datos OHLC para Bitcoin en temporalidad diaria. (Elaboración propia)

Para analizar precios de criptomonedas, se priorizan datos de *exchanges* con alto volumen, indicativo de liquidez y representación fiel del mercado (Danial, 2023). Estos *exchanges* ofrecen API que herramientas como TradingView utilizan para visualizar y analizar información (Zherlitsyn, 2024). Los datos suelen mostrarse como velas (*candlesticks*, ver Figura 2.4) en formato OHLC (Apertura, Máximo, Mínimo, Cierre, ver Figura 2.5), representando la actividad del precio en periodos ajustables (minutos a meses) según la necesidad del análisis (Jansen, 2020).

2.1.3.2. Conceptos de series temporales y su relevancia en la predicción financiera

Las series temporales son secuencias de datos que se recogen en puntos específicos a lo largo del tiempo, ordenados cronológicamente y, por lo general, espaciados a intervalos iguales o regulares. En el ámbito financiero, las series temporales son particularmente relevantes para la predicción de precios. Tres conceptos importantes son:

- **Tendencia:** La tendencia describe el patrón de comportamiento general de los datos en una secuencia temporal a lo largo de un periodo (Peixeiro, 2022). En el contexto financiero, determinar la tendencia de un activo contribuye a la predicción de su conducta en el largo plazo. Frecuentemente, la validez de los modelos de series temporales exige que la serie sea estacionaria, lo que significa que su media y varianza no deben cambiar sistemáticamente con el tiempo. Alcanzar esta estacionariedad puede requerir transformaciones para eliminar o mitigar las tendencias existentes (Jansen, 2020).
- **Volatilidad:** Mide cuánto varían o se dispersan los rendimientos de un activo financiero y es una indicación del riesgo (Hayes, 2003). La volatilidad es un factor central en la predicción financiera y es una característica prominente del precio de Bitcoin que este proyecto busca modelar.
- **Ruido:** En el contexto de series temporales, el ruido es la variabilidad aleatoria en los datos que no se puede explicar por el modelo de la serie temporal (la señal). Aunque los modelos buscan extraer la señal predecible, la presencia de ruido es una de las razones por las que las predicciones nunca son perfectas, especialmente en mercados financieros complejos (Peixeiro, 2022).

2.1.3.3. Técnicas de preprocesamiento y transformación de datos temporales

El preprocesamiento y la transformación de datos temporales son pasos cruciales en el análisis de series temporales. Estos pasos son esenciales para preparar los datos para el análisis posterior con Machine Learning (Peixeiro, 2022). Las técnicas relevantes para este proyecto incluyen:

- **Limpieza de datos:**
  - Corrección de errores y valores faltantes, utilizando métodos como la imputación (interpolación, etc.) (Little and Rubin, 2019).
- **Normalización o estandarización:**
  - Escalado de los datos para que tengan un rango específico, como 0 a 1 (normalización Min-Max) (Han et al., 2011).
  - Aplicación de la estandarización (o transformación z-score) a los datos, con el fin de obtener una media de 0 y una desviación estándar de 1 en la distribución resultante (James et al., 2023).

- **Transformaciones matemáticas:**
  - Aplicación de funciones como la logarítmica para estabilizar la varianza o normalizar la distribución de los datos (Box et al., 2015). Estas transformaciones pueden preparar la serie para análisis posteriores.
- **Diferenciación:**
  - Cálculo de las diferencias entre observaciones consecutivas ($y'_t = y_t - y_{t-1}$) para eliminar tendencias o patrones estacionales y así obtener una serie estacionaria (Palma, 2016).
  - Una variante común, especialmente en series financieras, es la **diferencia logarítmica** ($y'_t = \log(y_t) - \log(y_{t-1})$). Esta técnica combina los beneficios de la diferenciación (para la estacionariedad) con las propiedades de la transformación logarítmica (para estabilizar la varianza). Adicionalmente, el resultado se aproxima a la tasa de retorno porcentual, facilitando a menudo la interpretación y el modelado (Huang and Petukhina, 2022).
- **Agregación o remuestreo:**
  - Cambio de la granularidad de los datos, por ejemplo, de datos horarios a datos diarios, para simplificar el análisis o para alinear series temporales con diferentes frecuencias (Hyndman and Athanasopoulos, 2018).
- **Separación de series temporales:**
  - Separar la serie temporal en conjuntos de entrenamiento y prueba (y opcionalmente validación), respetando la secuencia temporal de los datos, evitando la contaminación con información futura (Hyndman and Athanasopoulos, 2018).

2.1.3.4. Modelos predictivos de Machine Learning empleados en series temporales

Existen múltiples algoritmos de *Machine Learning* aplicables al análisis de series temporales en finanzas. La presente investigación se centra específicamente en:

- **Algoritmos de Machine Learning para series temporales:**
  - **Redes Neuronales Recurrentes (RNN):** Se consideran especialmente las arquitecturas LSTM (Memoria a Corto y Largo Plazo) (Peixeiro, 2022) y GRU (Unidad Recurrente Cerrada) (Joseph and Tackes, 2024). Estos modelos son idóneos para capturar dependencias a largo plazo en series temporales, una característica relevante para la predicción de precios.

2.1.4. MLOps

En la actualidad, MLOps se está consolidando como una tendencia en el desarrollo, implementación y gestión de Machine Learning (ver Figura 2.6).

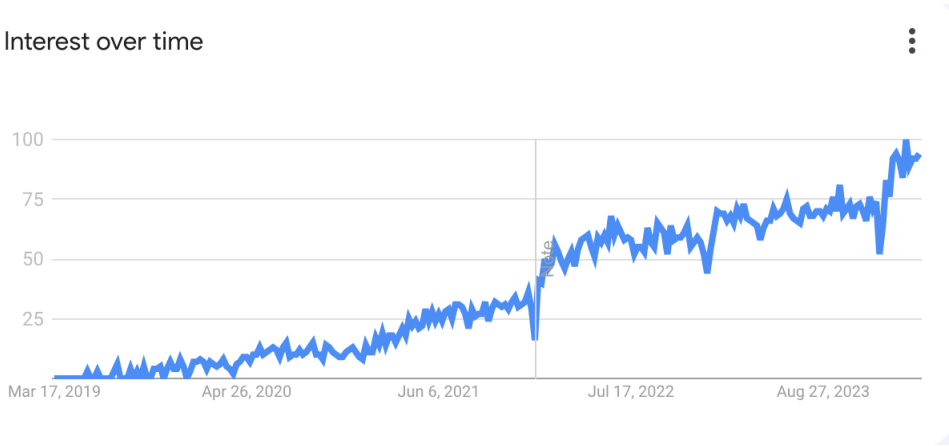


Figura 2.6: Tendencia de MLOps por Google Trends (Pardalis, 2023).

MLOps (Machine Learning Operations) es un conjunto de herramientas y buenas prácticas para usar Machine Learning en producción (Huyen, 2022). La Figura 2.7 ilustra un diagrama a alto nivel de lo que se compone MLOps.

- ML: se refiere a Machine Learning, en relación con los datos y el modelo.
- DEV: se refiere a la etapa de desarrollo de un modelo.
- OPS: hace referencia a la operación para la puesta a producción de un modelo.

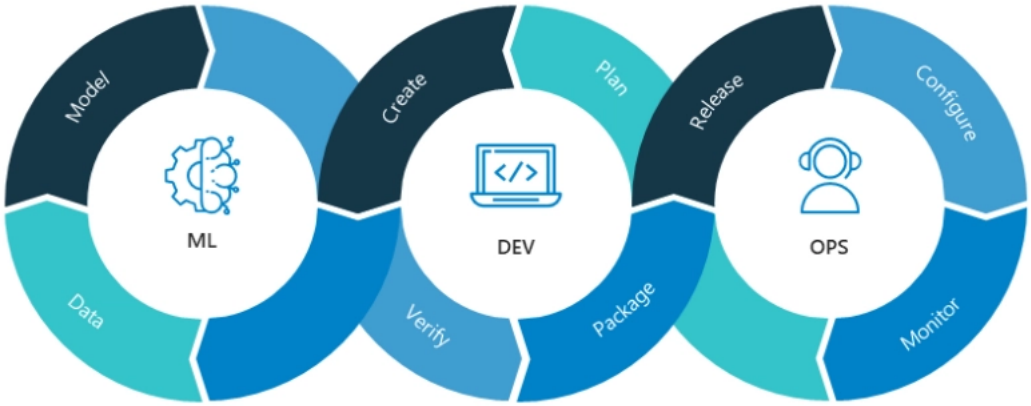


Figura 2.7: Diagrama de MLOps (Merritt, 2020).

El flujo de trabajo de MLOps (ver Figura 2.8) se desglosa de la siguiente manera:

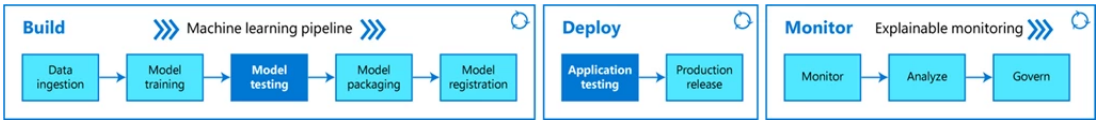


Figura 2.8: Diagrama del flujo de trabajo de MLOps (Higuchi, 2022).

- Construcción
  - **Ingesta de datos:** La ingesta de datos consiste en la recolección y adquisición de los datos. Esto puede incluir actividades como la obtención de datos de diferentes fuentes, como archivos, bases de datos, streams en tiempo real, API, entre otros (Raj, 2021). También implica la limpieza y preparación de datos para asegurarse de que sean de alta calidad y estén en el formato adecuado para el procesamiento posterior.
  - **Entrenamiento del modelo:** Consiste en ajustar los parámetros de un algoritmo de Machine Learning mediante el uso de datos de entrenamiento (Raj, 2021). Durante este proceso, el modelo "aprende" a partir de los datos proporcionados, encontrando patrones y relaciones que luego puede utilizar para hacer predicciones o tomar decisiones.
  - **Prueba del modelo:** Tras el entrenamiento del modelo, es fundamental probar su rendimiento para asegurarse de que está haciendo predicciones precisas. La prueba del modelo implica evaluarlo contra datos de prueba que no se utilizaron durante el entrenamiento (Raj, 2021). Esto proporciona una estimación de cómo el modelo funcionará en situaciones del mundo real. Se utilizan métricas de rendimiento como la precisión, la recuperación, el área bajo la curva ROC, entre otras, para evaluar la calidad del modelo.
  - **Empaquetado del modelo:** Esto se refiere al proceso de poner el modelo entrenado y probado en un formato que facilita su despliegue en un entorno de producción. Esto puede implicar la creación de un contenedor (por ejemplo, usando Docker) (Raj, 2021) que incluya el modelo, junto con todas las bibliotecas y dependencias necesarias para ejecutarlo. El empaquetado garantiza que el modelo se pueda trasladar fácilmente entre diferentes entornos y sistemas sin problemas de compatibilidad.
  - **Registro del modelo:** El registro del modelo es como una biblioteca o repositorio donde se almacenan diferentes versiones de los modelos entrenados (Raj, 2021). Permite a los equipos de ML llevar un seguimiento de los distintos modelos que se han creado, incluyendo información sobre su rendimiento, los datos utilizados para entrenarlos y los hiperparámetros seleccionados. Un registro de modelos facilita la administración del ciclo de vida de los modelos, incluyendo funciones como la auditoría, la replicación y el *rollback* a versiones previas en caso necesario.
- Despliegue
  - **Prueba de la aplicación:** En MLOps se refiere a la validación de la aplicación completa que incluye el modelo de Machine Learning. Esto no sólo implica asegurarse de que

el modelo haga predicciones precisas, sino que la aplicación funcione correctamente en su conjunto (Raj, 2021). Esto incluye pruebas de integración para verificar cómo interactúa el modelo con otros componentes de la aplicación, pruebas de rendimiento para asegurarse de que la aplicación puede manejar la carga esperada, y pruebas de seguridad para proteger contra vulnerabilidades.

- **Despliegue en producción:** El despliegue en producción es el proceso de llevar un modelo de Machine Learning a un entorno de producción donde pueda ser utilizado por los usuarios finales o sistemas (Raj, 2021). Esto implica no poner el modelo en funcionamiento, sino establecer la infraestructura necesaria para que funcione de manera eficiente, como servicios web o en la nube, y asegurarse de que se integra correctamente con otros sistemas empresariales.

- Monitoreo

  - **Monitoreo:** El monitoreo continuo del rendimiento y la salud operativa es vital para los modelos en producción. Este proceso permite detectar problemas como la *deriva de datos*, que ocurre cuando las características de los nuevos datos difieren de aquellos usados en el entrenamiento, afectando la efectividad. La identificación de estas problemáticas usualmente requiere un reentrenamiento del modelo con datos actualizados para mantener su precisión, un ciclo que se puede automatizar (Raj, 2021; Huyen, 2022; Kreuzberger et al., 2023a).
  - **Análisis:** El análisis en MLOps es el proceso de examinar y comprender el rendimiento y el comportamiento de los modelos (Raj, 2021). Esto puede implicar el análisis de las entradas y salidas del modelo, la interpretación de las métricas de rendimiento, y la realización de diagnósticos para identificar causas de problemas o áreas para mejorar. El análisis también puede referirse a la experimentación y al análisis estadístico que se realiza para mejorar los modelos.
  - **Gobernanza:** La gobernanza en MLOps incluye las políticas, los procesos y las medidas que se utilizan para controlar y gestionar los sistemas de Machine Learning (Raj, 2021). Esto abarca desde el cumplimiento de normas de la privacidad y seguridad de datos hasta la gestión de versiones de modelos y la adopción de decisiones éticas sobre el uso de algoritmos de ML. La gobernanza asegura que los modelos se utilicen de manera responsable, transparente y que se mantengan los estándares de calidad a lo largo del tiempo.

2.2. Estado del Arte

En el artículo *Forecasting bitcoin volatility: exploring the potential of deep learning* (Pratas et al., 2023) concluyen que para un modelo de una sola variable en temporalidad diaria, los modelos ARCH/GARCH tienen un buen desempeño, y entre los de deep learning MLP (Multi Layer Perceptron), RNN y LSTM, MLP es el que mejor se desempeña. Además, concluyen que en una

futura investigación se podrían considerar modelos multivariables, entre los datos de la cadena y el sentimiento del mercado.

En otro artículo, *The Long Short-Term Memory (LSTM) Model Combines with Technical Analysis to Forecast Cryptocurrency Prices.* (Fu and Ismail, 2023) analizan la combinación de LSTM con indicadores técnicos para predecir el precio de Bitcoin para un conjunto de datos en temporalidad diaria y encuentran que el rendimiento de la predicción mejora cuando se integran algunos indicadores técnicos, ya que reducen el error y aumentan la precisión. Argumentan que no todos los indicadores técnicos son igualmente útiles para la predicción.

En un blog de *Neptune.ai* (Zvorničanin, 2025) muestra un ejemplo de una implementación de un pipeline de MLOps para hacer predicciones en series de tiempo sobre la plataforma Neptune.ai.

En el artículo *Machine Learning Operations (MLOps): Overview, Definition, and Architecture* (Kreuzberger et al., 2023b) se presenta otra propuesta de una arquitectura para una implementación de MLOps. También destaca los desafíos operativos en la implementación de MLOps, que incluyen la necesidad de una automatización robusta debido a la complejidad de los entornos de producción de Machine Learning, la importancia de una fuerte gobernanza y control de versiones para la gran cantidad de artefactos generados, y la dificultad de resolver problemas de soporte debido a la cantidad de componentes involucrados en un sistema MLOps.

En este blog de Binance (Binance, 2022) muestran una arquitectura de MLOps de Machine Learning de extremo a extremo en tiempo real. Describen los diferentes componentes y tecnologías involucradas en este diseño y los beneficios de usar MLOps.

En el artículo *Real-Time Data Architecture Patterns* (Garcia, 2023) menciona algunos de los patrones de arquitectura de datos en tiempo real más utilizados, como lo son: *Batch Processing* para análisis en intervalos definidos, *Stream Processing* para flujos de datos continuos, *Lambda Architecture* que combina procesamiento en tiempo real y por lotes, *Kappa Architecture* centrada en flujos continuos, *Event-Driven Architecture* basada en eventos, y Microservices Architecture para servicios modulares. En este otro artículo *Real-Time Big Data Architecture for Processing Cryptocurrency and Social Media Data: A Clustering Approach Based on k-Means* (Barradas et al., 2022) hacen el uso de arquitectura Kappa para procesar en tiempo real los datos.

Dentro del ámbito de MLOps, herramientas como MLflow han surgido como soluciones populares para gestionar el ciclo de vida completo del desarrollo de modelos. MLflow permite el seguimiento de experimentos, la reproducción de resultados, el empaquetado de modelos y su despliegue en diversas plataformas (Lauchande, 2021).

Plataformas como Databricks proporcionan un entorno unificado para el ciclo de vida de modelos de Machine Learning, facilitando la colaboración entre ingenieros y científicos de datos. Databricks, con su integración con herramientas como MLflow y su capacidad para escalar recursos de computación, se posiciona como una solución robusta para la construcción de arquitecturas MLOps (Palacio, 2021).

Este proyecto contribuye al estado del arte mediante la evaluación de modelos de Machine Learning en al menos dos temporalidades distintas y la exploración de un enfoque para la predicción del precio de Bitcoin utilizando múltiples variables. Adicionalmente, se implementará una arquitectura basada en MLOps para facilitar el despliegue, monitoreo continuo y actualización de los modelos en

un entorno de producción. La construcción de un prototipo funcional que integre estas metodologías representa un avance práctico significativo, ya que no solo se enfoca en mejorar la precisión de las predicciones, sino que asegura su aplicabilidad, mantenimiento y adaptabilidad en el tiempo frente a la naturaleza dinámica del mercado de criptomonedas.

2.3. Resumen del capítulo

Este capítulo estableció el contexto teórico y el estado del arte relevantes para el proyecto.

Se inició con los fundamentos de Bitcoin, donde se describió la tecnología *blockchain* como un libro contable digital descentralizado. Se detalló la estructura de los bloques (cabecera y cuerpo), el proceso de minería, el mecanismo de halving para controlar la oferta, y los métodos para obtener el acceso a los datos de la cadena (API o nodo local). Se introdujo Bitcoin como un activo de inversión cuya volatilidad justificó el presente estudio.

A continuación, se introdujo el Machine Learning, se explicaron sus principios básicos y se enfocó en el aprendizaje supervisado, que fue la categoría utilizada en el proyecto. Se abordó la evaluación de modelos de regresión mediante métricas como RMSE, MSE y MAPE sobre conjuntos de prueba, y la validación mediante la división de datos, donde se mencionaron los conceptos de sobreajuste y subajuste.

La sección de Análisis de Precio describió herramientas gráficas como TradingView y el uso de datos OHLC provenientes de *exchanges*. Se definieron conceptos relevantes de series temporales como tendencia, volatilidad y ruido. Se enumeraron técnicas de preprocesamiento de datos temporales utilizadas en el proyecto: limpieza de valores faltantes, normalización/estandarización, transformación logarítmica, diferenciación (simple y logarítmica), agregación y separación de series. Finalmente, se presentaron los modelos predictivos de Machine Learning seleccionados para series temporales, específicamente las Redes Neuronales Recurrentes LSTM y GRU.

Posteriormente, se explicó MLOps como un conjunto de prácticas para la operacionalización de Machine Learning. Se describió su flujo de trabajo típico, que incluyó las fases de construcción (ingesta de datos, entrenamiento, prueba, empaquetado, registro del modelo), despliegue (prueba de aplicación, puesta en producción) y monitoreo (seguimiento de rendimiento, detección de deriva de datos, análisis y gobernanza). Se mencionaron estrategias para el reentrenamiento.

Finalmente, el Estado del Arte revisó investigaciones previas sobre predicción de volatilidad y precio de Bitcoin usando diversos modelos. Se referenciaron ejemplos de implementaciones de pipelines MLOps, arquitecturas propuestas, desafíos operativos, patrones de arquitectura de datos en tiempo real y herramientas como MLflow y Databricks. Se posicionó este proyecto indicando que evaluó modelos LSTM y GRU, utilizó un enfoque multivariable e implementó una arquitectura MLOps.

Capítulo 3

Desarrollo del Proyecto

En este capítulo se describe la ejecución técnica del proyecto, cuya estructura y desarrollo se basaron en los objetivos específicos previamente establecidos. Para la dirección de este proceso, se adoptó la metodología ágil Scrum, utilizando la herramienta Jira para la monitorización y el control de las tareas en un tablero dedicado. Los apartados siguientes detallan las actividades llevadas a cabo y los resultados conseguidos para cada uno de los objetivos planteados.

La selección de las herramientas tecnológicas para este proyecto se basó en criterios de robustez, disponibilidad de documentación y accesibilidad. Se optó por la plataforma en la nube Microsoft Azure por su amplia adopción a nivel global, lo que asegura una documentación extensa y soporte comunitario. Adicionalmente, se dispuso de créditos en esta plataforma, lo que facilitó el uso de sus servicios.

Para el desarrollo de software se emplearon lenguajes como C# y Python, elegidos por la abundante documentación y la disponibilidad de librerías para el procesamiento de datos y aprendizaje automático. La implementación de los modelos se realizó con Keras, mientras que la gestión del ciclo de vida y la experimentación se apoyó en Azure Databricks y MLflow, herramientas reconocidas en el ámbito de MLOps.

3.1. Determinar las variables relevantes para la predicción del precio de Bitcoin y elaborar un dataset depurado y validado para el entrenamiento y evaluación del modelo

Para identificar las variables relevantes en la predicción del precio de Bitcoin mediante un modelo de Machine Learning, se evaluó la utilidad de la información disponible en la *blockchain*. El estudio de Adjei (2019) sobre los determinantes de los retornos de Bitcoin concluye que la dificultad de minado y el tamaño del bloque, que impactan negativamente los retornos, son predictores robustos de estos. Con el fin de acceder a estos datos, se consideraron proveedores de API que ofrecieran información de la cadena de bloques, identificándose tres servicios potenciales: Glassnode, CryptoQuant y Blockchair.

Entre los servicios de API evaluados, Glassnode proporciona análisis de la *blockchain* y acceso a datos mediante una API, lo cual requiere una suscripción profesional con un costo mensual de $833 USD (facturación anual). De forma análoga, CryptoQuant ofrece funcionalidades similares con una suscripción anual que asciende a $799 USD mensuales para el acceso a su API. Estas opciones, si bien robustas, representaban un costo considerable.

Se exploró también Blockchair, un servicio que permite el acceso a datos de la *blockchain* con funcionalidades de análisis más limitadas, mediante versiones gratuitas y de pago. No obstante, su versión gratuita, con un tope de 1000 llamadas diarias a la API, se consideró insuficiente para el volumen de la *blockchain* de Bitcoin (aproximadamente 830 000 bloques en ese momento). Pruebas subsiguientes con una suscripción de pago de Blockchair también revelaron limitaciones, incluyendo interrupciones inesperadas del servicio, lo que finalmente motivó la búsqueda de una alternativa más controlable y transparente.

Dado que los servicios de API mencionados acceden en última instancia a la *blockchain*, se optó por la instalación de un nodo local de Bitcoin Core. Esta decisión, si bien otorgaba control total y acceso directo a los datos mediante su interfaz RPC, introdujo desafíos iniciales de ingeniería de software. En primer lugar, el considerable tamaño de la *blockchain* de Bitcoin, que alcanzó 1,2 terabytes durante el proyecto, exigió la adquisición de una unidad de almacenamiento de disco de alta capacidad. En segundo lugar, la sincronización inicial completa del nodo con la red Bitcoin representó un proceso extenso, requiriendo aproximadamente cinco días consecutivos de operación continua, un factor temporal crítico en la planificación del proyecto.

Una vez completada la descarga y sincronización, se exploró la información accesible vía RPC. Se determinó mediante pruebas con Python la viabilidad de obtener datos relevantes como la cantidad de bitcoins movidos por bloque, el número de transacciones, la dificultad de minado, las marcas de tiempo y las direcciones de los participantes.

La transformación de estos datos de la *blockchain* a un formato de series temporales con intervalos fijos, requerido por los modelos de Machine Learning, constituyó el siguiente desafío ingenieril. El volumen de aproximadamente 830 000 bloques hizo que los scripts iniciales desarrollados en Python para la extracción y conversión tuvieran tiempos de procesamiento excesivamente largos. Para optimizar el rendimiento, se reimplementó esta lógica en C#, un lenguaje compilado. Aunque esta migración representó una mejora, el procesamiento secuencial de la totalidad de los bloques en C# aún consumía una cantidad considerable de horas.

La solución definitiva para alcanzar una eficiencia adecuada involucró la optimización avanzada del proceso en C# mediante el diseño y puesta en marcha de una estrategia de particionamiento de datos y paralelismo, ver Figura 3.1. Los bloques se dividieron en lotes que fueron procesados de forma concurrente, aprovechando todos los núcleos disponibles del procesador. Esta optimización de ingeniería de software fue fundamental para reducir el tiempo de la primera carga y procesamiento completo de la información relevante de la *blockchain* a unas cuantas horas.

Un desafío de ingeniería de software fue la selección de una estrategia de persistencia intermedia para el volumen de información extraída de cada bloque. Aunque se evaluó MongoDB por su flexibilidad de esquema, las pruebas de carga indicaron que esta opción se convertía en un cuello de botella para la persistencia de los datos, debido a la sobrecarga en las operaciones de escritura.

Para superar esta limitación y gestionar los datos eficientemente, se implementó un proceso de transformación de la información extraída de la *blockchain*. Este proceso, esquematizado en la Figura 3.2, consta de dos etapas secuenciales que progresivamente transforman los datos hasta generar un conjunto final agregado, con estructura temporal uniforme para el análisis.

```
Parallel.For(lastBlock, blockCount + 1, options, () => {
        var threadCredentials = new RPCCredentialString {
            UserPassword = new NetworkCredential(rpcUser, rpcPassword),
            Server = rpcServer
        };

        var threadClient = new RPCClient(threadCredentials, Network.Main);

        return threadClient;
    }, (i :int, loopState, rpc) => {

        try {

            var block = rpc.GetBlock(i);
            Task.Run(async () => {
```

Figura 3.1: Fragmento del código C# usando paralelismo. (Elaboración propia)

La primera etapa consiste en la lectura de información de cada bloque mediante la interfaz RPC del nodo. Para optimizar el rendimiento de la escritura y facilitar la manipulación subsiguiente, estos datos se almacenan inicialmente en archivos de texto plano individuales, utilizando el formato CSV.

En la segunda etapa, una aplicación adicional procesa los archivos CSV individuales generados previamente. Esta consolida y agrega la información para producir un conjunto de datos con una temporalidad de una hora. Dicha agregación horaria fue necesaria porque el tiempo de procesamiento por bloque individual podía variar, y se requería una estructura temporal regular para los análisis posteriores.

Un ejemplo de la estructura de estos archivos CSV resultantes, agregados con temporalidad horaria, se presenta en la Figura 3.3. La figura ilustra la estructura definitiva del conjunto de datos, el cual constituye el fundamento para las etapas posteriores del proyecto.

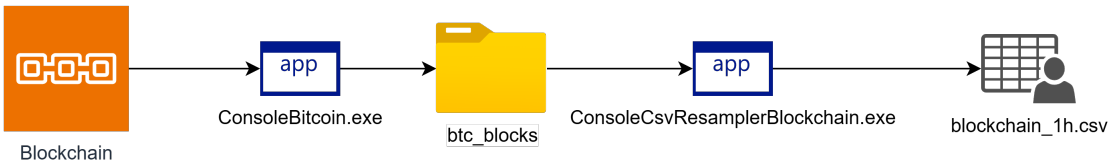


Figura 3.2: Transformación de la *blockchain* a serie temporal. (Elaboración propia)

Paralelamente, la información de precio y volumen de Bitcoin se obtuvo de las API de tres *exchanges*: Bitstamp, por su historial desde 2011; Binance, por ser el de mayor volumen transaccional global, y Coinbase, como segundo en volumen (CoinMarketCap, 2024). Estos datos, ya en formato de series temporales, se almacenaron en archivos CSV, como se observa en la Figura 3.4.

Un aspecto crucial en la recolección de datos de estas diversas fuentes (API de *exchanges* e información de la *blockchain*) es el manejo estandarizado de las marcas de tiempo. Estas se registran

A	B	C	D	E	F	G	H	I
initialblock	finalblock	date	initialdate	finaldate	value	qty	avg	difficulty
898440	898447	1748278800	1748261059	1748264389	23464.89401	17096	1.203849457	121658450774825
898448	898450	1748282400	1748265003	1748267951	17181.15796	11035	1.532114234	121658450774825
898451	898453	1748286000	1748269340	1748271416	34931.06727	10768	3.241859067	121658450774825
898454	898468	1748289600	1748272220	1748275040	55248.8587	31202	1.949010684	121658450774825
898469	898478	1748293200	1748275578	1748278199	28177.56823	28875	1.125269398	121658450774825
898479	898483	1748296800	1748278861	1748280953	15302.95201	14128	1.108548524	121658450774825
898484	898486	1748300400	1748282485	1748284838	18523.08969	9539	1.752961578	121658450774825
898487	898492	1748304000	1748286074	1748289188	23886.52614	15857	1.464486151	121658450774825
898493	898495	1748307600	1748290137	1748292929	17557.88786	10443	1.662925677	121658450774825
898496	898502	1748311200	1748293217	1748295659	16633.96437	14061	1.224424903	121658450774825
898503	898508	1748314800	1748297327	1748299840	16755.65159	15249	0.979521197	121658450774825
898509	898514	1748318400	1748300900	1748302321	8364.098364	11073	0.70247214	121658450774825

Figura 3.3: Ejemplo de la estructura de un archivo CSV con datos de la blockchain agregados con temporalidad horaria. (Elaboración propia)

y conservan en formato UTC para asegurar la coherencia temporal desde su origen y a lo largo de todo el procesamiento subsiguiente.

En la última etapa del preprocesamiento, se integraron los datos de la *blockchain* y de los tres *exchanges* en un único conjunto. Un script en Python automatizó esta consolidación (Figura 3.5), realizando transformaciones y la unión por marca de tiempo. Dicho script fue diseñado para preservar el estándar UTC de las marcas de tiempo, provenientes de las fuentes originales, garantizando así la integridad temporal del conjunto de datos unificado. Se aplicó también interpolación para valores faltantes.

Los datos consolidados, con sus marcas de tiempo estandarizadas en UTC, se cargaron finalmente en Azure Data Lake Storage Gen2. La implementación consideró la eficiencia en el manejo de volúmenes grandes y la necesidad de una estructura temporal homogénea. Aunque los datos se procesaron inicialmente a frecuencia horaria, se agregaron a una temporalidad diaria (manteniendo la referencia UTC) para el entrenamiento de los modelos de predicción y la generación de pronósticos diarios.

Análisis Visual y Estacionariedad de las Variables

La Figura 3.6 ilustra el comportamiento temporal de las variables seleccionadas para el análisis: 'Close', 'Volume', 'btc', 'number_transactions' y 'difficulty'. A continuación, se define cada una de estas variables:

- **'Close' (precio de cierre)**: Corresponde al último precio al que se negoció Bitcoin durante el intervalo de tiempo analizado (diario en este caso) en los *exchanges*.
- **'Volume' (volumen transado)**: Representa la cantidad total de Bitcoin que se comerció en los *exchanges* durante el periodo de tiempo especificado. Esta cifra indica el nivel de actividad y liquidez del mercado.

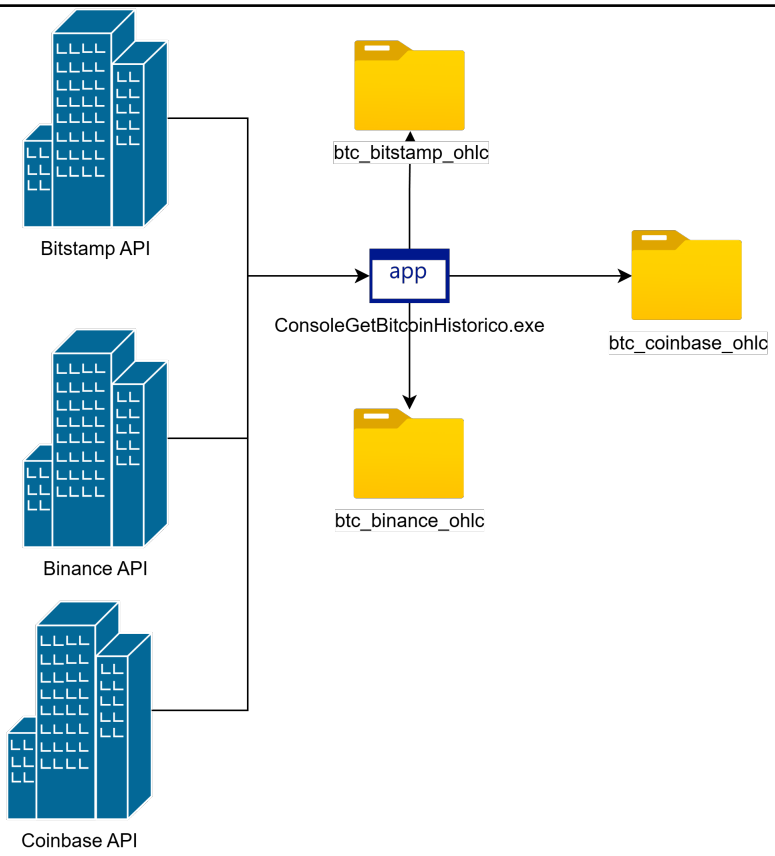


Figura 3.4: Obtención de los datos desde los *exchanges*. (Elaboración propia)

- **'btc' (cantidad de bitcoins movidos)**: Se refiere al total de bitcoins que fueron transferidos entre diferentes direcciones dentro de la *blockchain* en cada bloque y luego agregados al intervalo de tiempo correspondiente.

- **'number_transactions' (número de transacciones)**: Indica el número total de transacciones individuales que fueron registradas y validadas en la *blockchain* durante el intervalo de tiempo considerado.

- **'difficulty' (dificultad de la red)**: Es una medida que indica cuán difícil es encontrar un bloque nuevo en la *blockchain* de Bitcoin. Esta se ajusta periódicamente para mantener un tiempo promedio de generación de bloques.

En la visualización de la Figura 3.6, se observa que la serie 'Close' presenta una tendencia general con fluctuaciones notables. La variable 'difficulty' muestra un comportamiento de incrementos escalonados a lo largo del periodo observado. Por su parte, las series 'Volume', 'btc' y

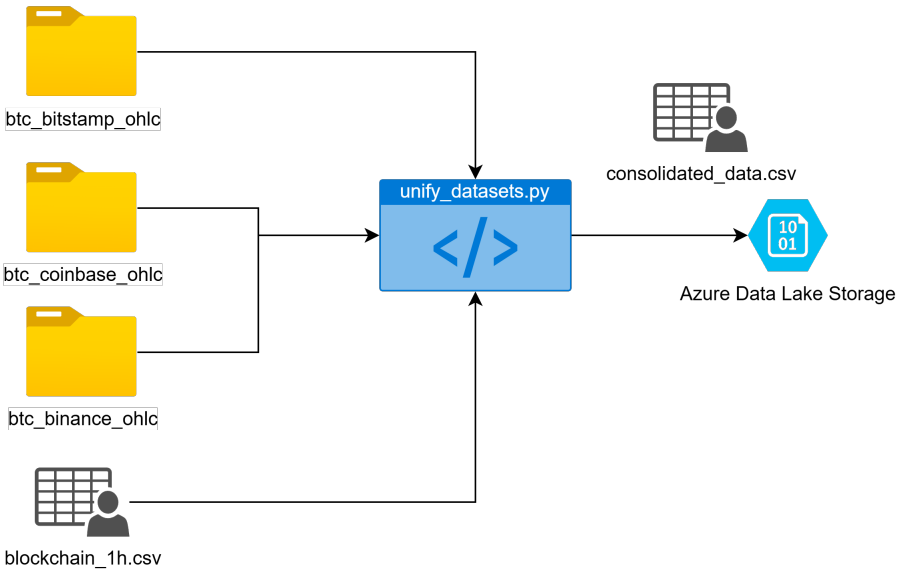


Figura 3.5: Unificación de los datasets de la *blockchain* y los *exchanges*. (Elaboración propia)

'number_transactions' exhiben una considerable variabilidad, donde 'number_transactions' parece indicar un ligero incremento hacia el final del intervalo graficado.

Se reconoce que la existencia de tendencias y la falta de estacionariedad en las series de tiempo pueden incidir en el desempeño y la fiabilidad de los modelos de predicción. En consecuencia, se evaluó la estacionariedad de las series utilizando la prueba de Dickey-Fuller Aumentada (ADF). Los valores p resultantes de este análisis para cada variable estudiada se consignan en la Tabla 3.1.

Close	**Volume**	**btc**	**number_transactions**	**difficulty**
0,9879	0,0370	0,0169	$1{,}377 \times 10^{-13}$	0,9962

Tabla 3.1: ADF valores p

Usualmente, un valor p que sobrepasa el nivel de significancia convencional de 0.05 lleva a la no refutación de la hipótesis nula, señalando así que la serie podría no ser estacionaria (Kaabar, 2024). Dichos resultados subrayan la importancia de transformar aquellas series que no presentan estacionariedad antes de iniciar la etapa de modelado.

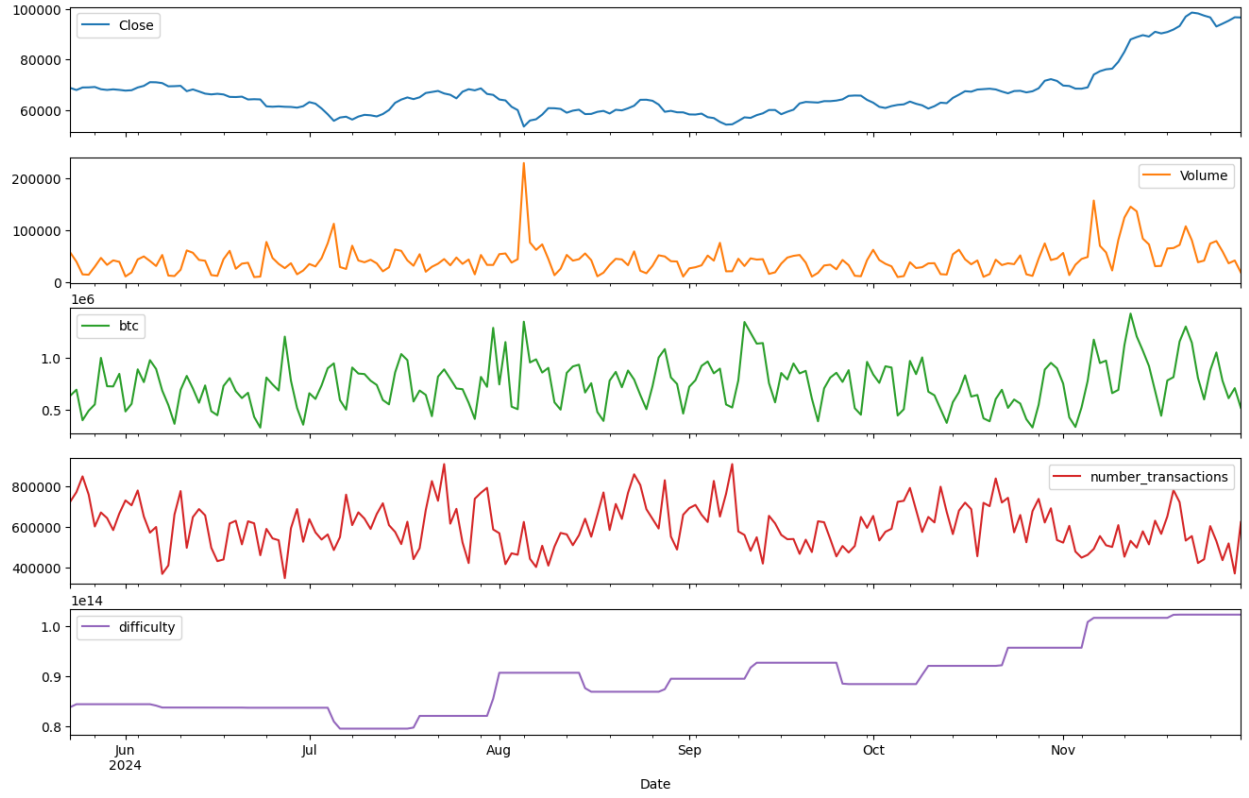


Figura 3.6: Comportamiento temporal de las variables relevantes. (Elaboración propia)

3.2. Diseñar, implementar y evaluar una arquitectura en la nube que permita la automatización del entrenamiento y redespliegue de un modelo de Machine Learning a través de MLOps registrando experimentos, métricas, artefactos y monitoreo.

Para cumplir con el objetivo de diseñar, implementar y evaluar una arquitectura en la nube que automatice el entrenamiento y redespliegue de un modelo de Machine Learning mediante MLOps, registrando experimentos, métricas, artefactos y monitoreo, se propuso la siguiente arquitectura de alto nivel, ilustrada en la Figura 3.7.

La arquitectura se fundamenta en la plataforma Azure y se articula a través de los siguientes componentes principales:

Fuentes de Datos: La información requerida para el entrenamiento del modelo se origina en dos fuentes primarias: la *blockchain* de Bitcoin y diversos *exchanges* de criptomonedas.

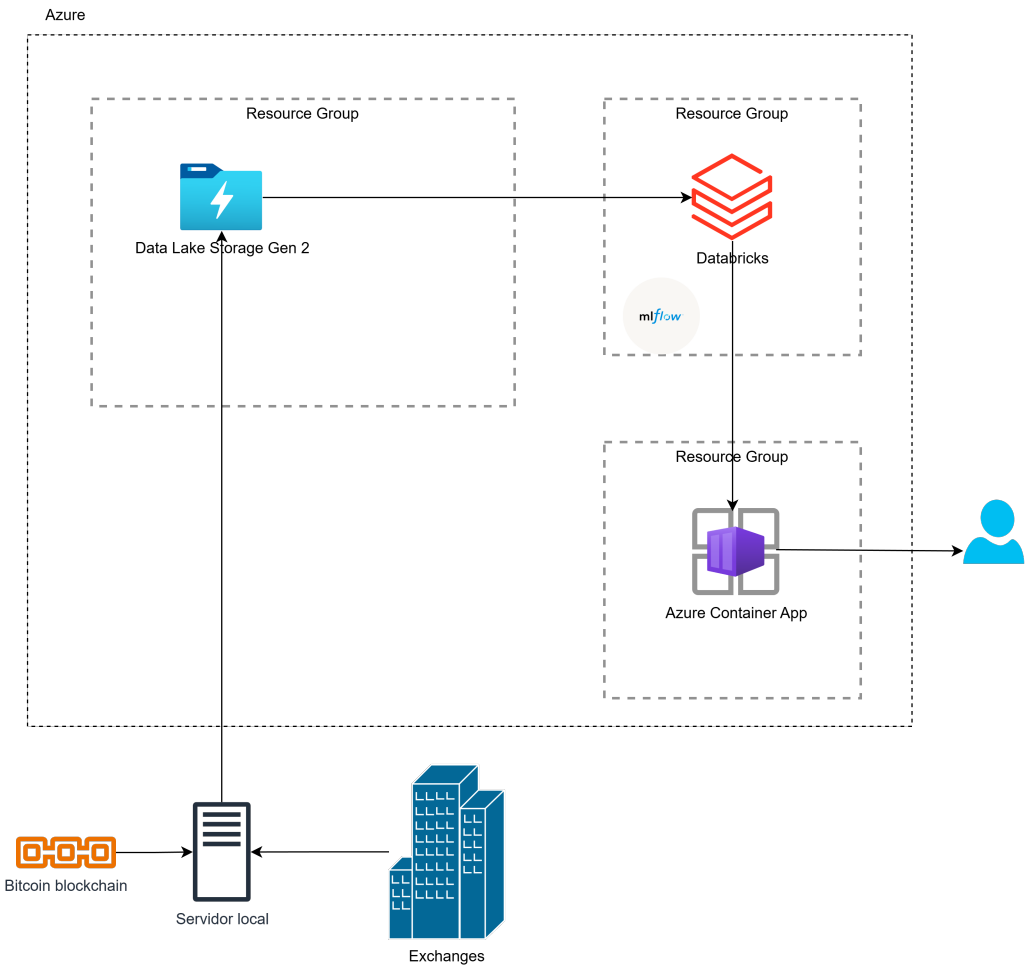


Figura 3.7: Arquitectura de alto nivel. (Elaboración propia)

Servidor Local: El servidor local es fundamental en el proceso de ingesta. Diariamente, se ejecutan de forma programada en este servidor los *scripts* que obtienen y consolidan la información de la *blockchain* y los *exchanges*, generando un archivo CSV unificado. Inmediatamente después de esta consolidación local, la carga de dicho archivo CSV hacia Azure Data Lake Storage Gen2 se ejecuta también de forma automatizada. Este proceso es gestionado por el Programador de Tareas de Windows en el servidor local, el cual inicia la copia de datos a las 00:01 cada día.

Azure Data Lake Storage Gen2: El archivo CSV, que es consolidado y actualizado horariamente en el servidor local, tiene su versión más reciente copiada diariamente para ser almacenada en Azure Data Lake Storage Gen2. Dicha solución de almacenamiento, caracterizada por su escalabilidad y elevado desempeño, hace posible el alojamiento de considerables cantidades de datos conservando su forma nativa, lo que simplifica su consulta para las fases subsiguientes de procesa-

miento y entrenamiento.

Azure Databricks: Para la fase de entrenamiento del modelo de Machine Learning, se utiliza Azure Databricks. Esta plataforma de análisis basada en Apache Spark proporciona un entorno colaborativo para la ciencia de datos y la ingeniería de datos. En este componente, se ejecutan los *notebooks* que contienen el código para la preparación de los datos, el entrenamiento de los modelos (LSTM y GRU, como se describió previamente), y el seguimiento de los experimentos mediante MLflow. La Figura 3.8 ilustra la configuración específica del clúster de nodo único en Azure Databricks empleado para estas tareas computacionales, donde se destaca la versión del *runtime* (*15.4 LTS*) y el tipo de nodo seleccionado (*Standard_F4*, con 8 GB de memoria y 4 núcleos).

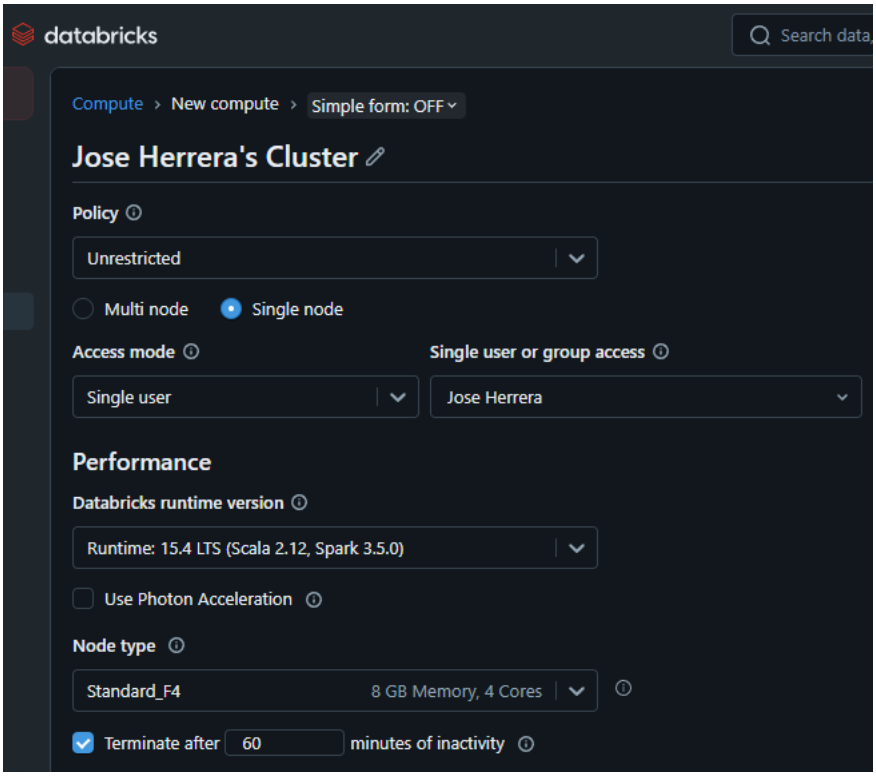


Figura 3.8: Configuración del clúster de nodo único 'Jose Herrera's Cluster' en Azure Databricks, detallando la versión del *runtime*, el tipo de nodo y otros parámetros relevantes para el entrenamiento de los modelos. (Elaboración propia)

MLflow: Dentro del entorno de Azure Databricks, se integra MLflow para gestionar el ciclo de vida de los experimentos de Machine Learning. MLflow se encarga del registro de los parámetros de cada ejecución, las métricas de rendimiento obtenidas durante el entrenamiento y la evaluación, los artefactos generados (como los modelos serializados y los escaladores), y el seguimiento de las

diferentes versiones de los modelos.

Azure Container Apps: Para la fase de despliegue del modelo entrenado, se emplea Azure Container Apps. Este servicio permite ejecutar aplicaciones en contenedores en un entorno sin servidor y altamente escalable. Dentro de Azure Container Apps reside una aplicación encargada de la visualización de la información generada por el modelo desplegado. Sin embargo, el diseño y la implementación detallada de esta aplicación de visualización se contemplarán en un objetivo posterior de este proyecto (referido en la Sección 3.5 en la página 39).

Monitoreo (Implícito): Si bien el diagrama de alto nivel no detalla un componente específico de monitoreo, la implementación de una arquitectura MLOps robusta requiere la integración de herramientas de monitoreo. Estas herramientas supervisan el rendimiento del modelo desplegado en Azure Container Apps, así como la infraestructura subyacente, permitiendo detectar posibles desviaciones en el rendimiento del modelo o problemas en la infraestructura, y activar alertas para su resolución.

Usuario Final: El usuario final accede a las predicciones del modelo a través de la aplicación web de visualización, la cual está alojada y expuesta por Azure Container Apps.

Esta arquitectura en la nube facilita la implementación de prácticas MLOps, permitiendo el seguimiento de los experimentos, la gestión de los artefactos y el monitoreo continuo del modelo desplegado. La disponibilidad de los datos actualizados para el ciclo de reentrenamiento diario del modelo se asegura mediante una secuencia coordinada y automatizada.

El proceso de actualización diaria comienza con la carga del archivo CSV consolidado desde el servidor local hacia Azure Data Lake Storage Gen2 a las 00:01 UTC, tarea gestionada por el Programador de Tareas de Windows. Seguidamente, a las 00:10 UTC, se activa automáticamente un *job* en Azure Databricks, denominado 'Job_maestria'. La configuración de este *job*, detallada en formato YAML (ver Figura 3.9), establece su ejecución diaria mediante la expresión cron quartz_cron_expression: 12 10 0 * * ?.

Dicho *job* orquesta dos tareas secuenciales: la primera, 'Task_gru_no_cadena', ejecuta el *notebook* '/Workspace/Users/jherrera/gru_no_cadena' para el entrenamiento del modelo GRU y el correspondiente registro de sus resultados en MLflow. La segunda tarea, 'Task_promove_production', que depende de la finalización de la anterior, ejecuta el *notebook* '/Workspace/Users/jherrera/promove_production' para registrar la nueva versión del modelo y transicionarla a la etapa de 'Producción' en MLflow. Ambas tareas utilizan el clúster existente especificado, y la ventana de nueve minutos entre la carga de datos y el inicio del *job* ha demostrado ser suficiente.

Para la automatización del ciclo de reentrenamiento, se optó por un proceso de ejecución diaria programada en lugar de mecanismos basados en eventos. Esta decisión se fundamentó en la eficiencia del proceso completo de actualización diaria. Este ciclo, que abarca la carga de datos desde el servidor local (iniciada a las 00:01 y con una ventana de nueve minutos para la transferencia hasta las 00:10) y el subsiguiente *job* de reentrenamiento en Azure Databricks (que comienza a las 00:10), típicamente concluye en su totalidad en menos de 30 minutos. Por ello, un enfoque de activación diaria basada en tiempo resultó suficiente y más simple de implementar que la gestión de *triggers* específicos.

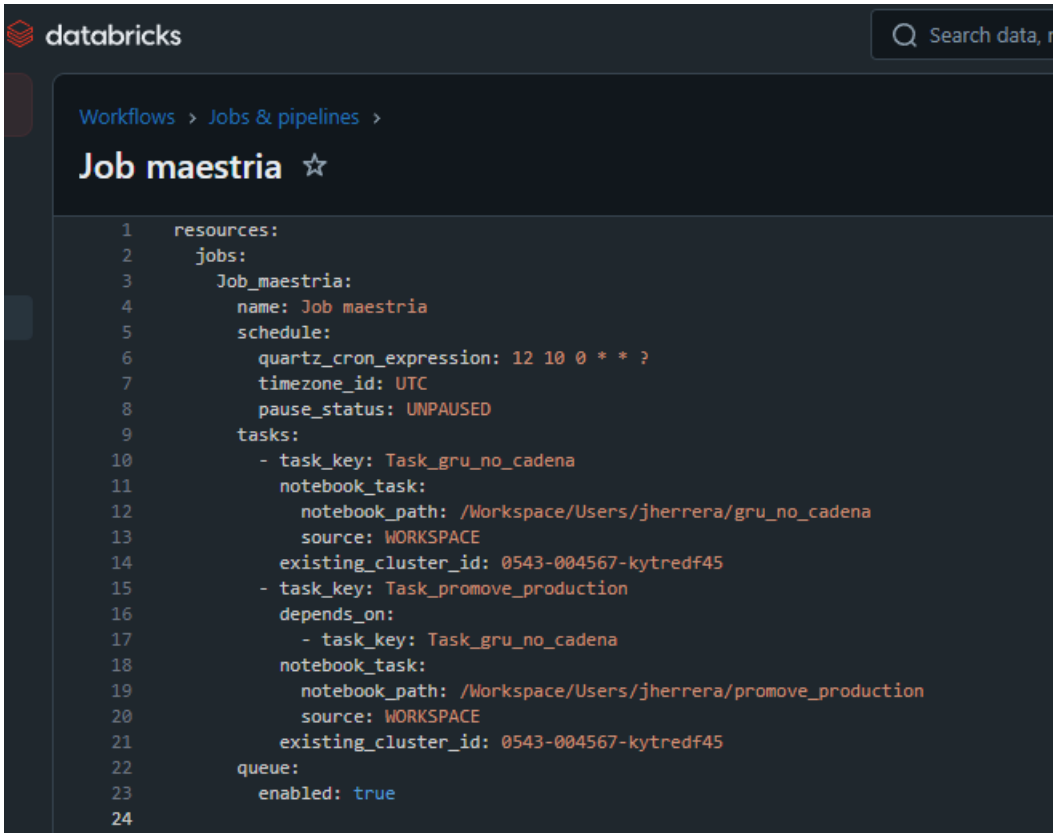


Figura 3.9: Configuración YAML del Job 'Job_maestria' en Azure Databricks, mostrando la programación cron y las tareas secuenciales para el entrenamiento y promoción del modelo. (Elaboración propia, basada en la interfaz de Databricks)

3.3. Desarrollar y entrenar dos modelos de Machine Learning distintos empleando técnicas de series temporales

Para abordar este objetivo, se implementaron dos arquitecturas de redes neuronales recurrentes: Long Short-Term Memory (LSTM) y Gated Recurrent Unit (GRU). La elección de estas arquitecturas se basa en su capacidad para gestionar dependencias a largo plazo en secuencias de datos, mitigando el problema del gradiente desvaneciente que puede afectar a redes recurrentes más simples (Lazzeri, 2020). Sus mecanismos internos de compuertas les permiten aprender qué información conservar o descartar a lo largo de la secuencia, una característica pertinente para modelar series temporales financieras. Ambos modelos se configuraron para utilizar múltiples variables de entrada y predecir un único paso futuro en la serie temporal del precio.

La implementación y entrenamiento de estos modelos se llevó a cabo utilizando la librería Keras con el *backend* de TensorFlow. Debido a las limitaciones de soporte de CUDA en las versiones recientes de TensorFlow para Windows, se optó por utilizar el Subsistema de Windows para Linux

(WSL) para aprovechar la capacidad de procesamiento de la Unidad de Procesamiento Gráfico (GPU) de la máquina local, lo que aceleró los tiempos de entrenamiento en comparación con el uso exclusivo de la Unidad Central de Procesamiento (CPU).

Los resultados de la prueba ADF (presentada en la sección 3.1) indicaron la no estacionariedad de series de entrada fundamentales para el modelo, como el precio de Bitcoin ('Close') y la dificultad de la red ('difficulty'). Esta situación orientó la necesidad de aplicar transformaciones para estabilizar dichas series antes del modelado.

En consecuencia, se aplicó una transformación de diferencia logarítmica a la serie del precio de Bitcoin y una diferenciación simple (de primer orden) a la de 'difficulty'. La primera transformación, común en series financieras, busca estabilizar la varianza, facilitar la estacionariedad y mejorar la interpretación (Huang and Petukhina, 2022). La segunda es una técnica estándar para remover tendencias y lograr estacionariedad en datos como 'difficulty' (Palma, 2016). Ambas transformaciones se realizaron para optimizar el procesamiento de los datos por los modelos LSTM y GRU.

El preprocesamiento tuvo como objetivo optimizar el aprendizaje de patrones en los modelos para así mejorar su capacidad predictiva. Para la fase de entrenamiento se emplearon datos históricos recientes, agregados a una frecuencia diaria a partir de un conjunto definido de observaciones horarias, los cuales se complementaron con conjuntos de validación y de prueba.

Aunque la intención original del proyecto (Sección 1.4) consideraba un historial de datos de aproximadamente diez años, la experimentación y optimización del reentrenamiento diario del prototipo indicaron que un periodo específico de 192 días era, de hecho, suficiente para el desarrollo y la evaluación de los modelos finales. La elección de este periodo de 192 días se definió buscando optimizar los costos del reentrenamiento diario en Azure y mantener la estabilidad de la arquitectura de los modelos LSTM y GRU, pues la experimentación evidenció que periodos más largos exigían ajustes en sus capas para sostener el rendimiento.

Modelo LSTM

Se diseñó un modelo LSTM secuencial compuesto por dos capas LSTM con 200 unidades cada una, intercaladas con capas de *Dropout* para mitigar el sobreajuste. La primera capa LSTM devolvió secuencias, mientras que la segunda no. Por último, una capa densa de una sola unidad produjo la predicción del precio. El modelo fue compilado mediante el optimizador Adam y la función de pérdida de Error Absoluto Medio (MAE), con la Raíz del Error Cuadrático Medio (RMSE) como métrica de evaluación durante el entrenamiento.

El proceso de entrenamiento y evaluación se gestionó utilizando la plataforma MLflow, que permitió registrar los parámetros de configuración del modelo, las métricas de rendimiento en los conjuntos de entrenamiento, validación y prueba, así como los artefactos generados, incluyendo los escaladores utilizados para la normalización de los datos y el modelo entrenado.

Para la evaluación del modelo, se invirtió la transformación de escalado y la diferenciación logarítmica aplicada a la serie temporal del precio, lo que permitió obtener las predicciones en la escala original y calcular métricas de error interpretables, como el RMSE, el MAE y el Error Porcentual Absoluto Medio (MAPE). Adicionalmente, se generaron visualizaciones contrastando

las proyecciones con los valores reales en el conjunto de prueba.

Finalmente, se realizó una predicción del precio para el día siguiente al último dato disponible en el conjunto de prueba, utilizando la última secuencia de datos preprocesados como entrada al modelo entrenado.

Modelo GRU

Se construyó un modelo GRU secuencial con una arquitectura similar al modelo LSTM, utilizando capas GRU en lugar de LSTM. El modelo incluyó dos capas GRU con 200 unidades cada una, con capas de Dropout para la regularización. La compilación, el entrenamiento y el seguimiento con MLflow se realizaron de manera análoga al modelo LSTM, utilizando los mismos optimizadores, función de pérdida y métricas.

La evaluación del modelo GRU también implicó la inversión de las transformaciones de escalado y diferenciación logarítmica para obtener las predicciones en la escala original y calcular las métricas de error correspondientes. Se generaron gráficos comparativos y se realizó la predicción del precio para el día siguiente, siguiendo el mismo procedimiento que para el modelo LSTM.

3.4. Evaluar y comparar los resultados de los dos modelos. Seleccionar la técnica con rendimiento superior para la predicción del precio de Bitcoin

En esta sección se efectúa la evaluación comparativa del desempeño predictivo de los modelos Long Short-Term Memory (LSTM) y Gated Recurrent Unit (GRU). El propósito es determinar qué arquitectura ofrece un rendimiento más ajustado para la predicción del precio de Bitcoin bajo las condiciones de este estudio.

La evaluación se realizó mediante la generación de predicciones diarias del precio de Bitcoin para el periodo del 1 de diciembre de 2024 al 9 de marzo de 2025. Para cada día, en este intervalo, se contrastó el precio predicho por cada modelo con el precio real observado. Se utilizó el Error Porcentual Absoluto Medio (MAPE), calculado diariamente y luego promediado sobre el periodo, como métrica principal de comparación.

Al seleccionar las variables para el entrenamiento de los modelos, se partió de la premisa de que el precio ('Close') y el volumen ('Volume') obtenidos de los *exchanges* son indicadores fundamentales que operan conjuntamente. Estudios recientes específicos para Bitcoin, como el de Moyo and Phiri (2023), confirman la existencia de una relación dinámica y a largo plazo entre su precio y volumen, e incluso describen un bucle de retroalimentación donde las variaciones en una variable pueden influir en la otra. Dada esta interdependencia documentada, estas dos variables de los *exchanges* se utilizaron como un conjunto base en las comparaciones principales.

La evaluación inicial examinó el rendimiento de los modelos al incorporar a este conjunto base (precio y volumen de los *exchanges*) las variables *on-chain* seleccionadas ('btc', 'number_transactions', 'difficulty'). La Tabla 3.2 presenta un extracto del rendimiento predictivo diario de los modelos LSTM y GRU bajo este primer escenario.

Fecha	Precio real	LSTM	LSTM MAPE %	GRU	GRU MAPE %
12/1/24	96 848	96 971	0,13	96 469	0,39
12/2/24	96 031	96 887	0,89	96 627	0,62
12/3/24	95 502	95 601	0,10	96 244	0,78
12/4/24	96 421	95 424	1,03	94 904	1,57
12/5/24	101 050	96 705	4,30	96 903	4,10
...	...	...	...	...	...
3/5/25	88 706	83 843	5,48	85 397	3,73
3/6/25	90 505	86 806	4,09	90 226	0,31
3/7/25	88 034	88 880	0,96	87 463	0,65
3/8/25	86 227	87 748	1,76	87 312	1,26
3/9/25	84 320	86 673	2,79	88 010	2,51

Tabla 3.2: Primeras y últimas 5 entradas del rendimiento predictivo diario de los modelos LSTM y GRU con datos de la cadena.

En esta configuración con datos de la cadena, el modelo LSTM obtuvo un MAPE promedio de 1,76 %, mientras que el modelo GRU registró un MAPE promedio de 1,86 %. Estos valores indican que el modelo LSTM (1,76 %) exhibió una precisión predictiva promedio ligeramente superior al modelo GRU (1,86 %) al utilizar dichas variables *on-chain*.

Posteriormente, se evaluó el rendimiento de los modelos bajo un segundo escenario, en el cual se excluyeron completamente las variables *on-chain*, utilizando únicamente el conjunto base de precio y volumen de los *exchanges*. La Tabla 3.3 detalla el comportamiento diario del MAPE en este escenario.

Fecha	Precio real	LSTM	LSTM MAPE %	GRU	GRU MAPE %
12/1/24	96 848	96 865	0,02	96 797	0,05
12/2/24	96 031	97 083	1,10	96 777	0,78
12/3/24	95 502	96 310	0,85	96 895	1,46
12/4/24	96 421	95 059	1,41	95 220	1,25
12/5/24	101 050	96 524	4,48	96 903	3,66
...	...	...	...	...	...
3/5/25	88 706	83 629	5,72	83 910	5,41
3/6/25	90 505	87 188	3,66	90 041	0,51
3/7/25	88 034	89 549	1,72	89 797	2,00
3/8/25	86 227	88 056	2,12	87 901	1,94
3/9/25	84 320	86 258	2,30	86 583	2,68

Tabla 3.3: Primeras y últimas 5 entradas del rendimiento predictivo diario de los modelos LSTM y GRU sin datos de la cadena.

La exclusión de los datos de la cadena resultó beneficiosa para el rendimiento de ambos modelos.

El MAPE promedio del modelo LSTM disminuyó de 1,76 % (con datos de la cadena) a 1,72 % (sin datos de la cadena). De manera más notable, el MAPE promedio del modelo GRU se redujo de 1,86 % (con datos de la cadena) a 1,66 % (sin datos de la cadena). Los hallazgos sugieren que, para el conjunto de variables y el periodo evaluado, la incorporación de las variables *on-chain* consideradas no se tradujo en una mejora del rendimiento predictivo.

De hecho, el modelo GRU entrenado sin datos de la cadena no solo mejoró su propio rendimiento de manera considerable, sino que también alcanzó el MAPE promedio más bajo de todas las configuraciones evaluadas (1,66 %). En esta configuración sin datos de la cadena, el modelo GRU superó al modelo LSTM, que obtuvo un MAPE de 1,72 %.

En consecuencia, la evaluación detallada en esta sección, utilizando el MAPE promedio como métrica principal, revela que el modelo GRU entrenado sin la inclusión de datos de la cadena (MAPE = 1,66 %) alcanzó el menor error predictivo. Este rendimiento fue superior al del modelo LSTM en la misma configuración sin datos (MAPE = 1,72 %), y ambos modelos sin datos *on-chain* superaron a sus respectivas versiones que sí los incorporaron. Estos hallazgos subrayan cómo la selección de características influye en el desempeño y apuntan al modelo GRU sin datos de la cadena como el de mayor precisión relativa dentro de este conjunto particular de experimentos.

El hallazgo de que la exclusión de las variables on-chain mejoró el rendimiento predictivo amerita una discusión. Es posible que los datos de la cadena, como la dificultad de minado, sean más efectivos para estimaciones a largo plazo que para la predicción diaria. Su influencia en la volatilidad a corto plazo podría ser menos directa en comparación con los datos de mercado, que reflejan la liquidez y el comportamiento del inversor de manera más inmediata.

Se debe considerar que si dentro de los datos de la cadena se hubiera agregado información sobre el crecimiento de la red (por ejemplo, número de direcciones activas), es factible que los resultados del modelo hubieran sido distintos. Futuras investigaciones podrían explorar estas variables para determinar su impacto en la predicción a corto plazo.

3.5. Diseñar e implementar una aplicación web de visualización del precio de Bitcoin y su proyección, y llevar a cabo la evaluación de dicha arquitectura

Se eligió GitHub para la gestión del código fuente y CI/CD por su robusto control de versiones (Git), facilitando la colaboración y el seguimiento de cambios. Sus GitHub Actions integradas permiten automatizar flujos de trabajo de compilación, prueba y despliegue directamente desde el repositorio. La amplia adopción de la plataforma también asegura extensa documentación y soporte comunitario, relevantes para el desarrollo del proyecto.

Para cumplir con este objetivo, así como llevar a cabo la evaluación de dicha arquitectura, se optó por utilizar la librería Streamlit. Esta herramienta de código abierto facilita la creación de interfaces web interactivas a partir de scripts de Python, lo que permitió desarrollar de manera eficiente un visualizador para la información generada por el modelo de predicción.

La aplicación web desarrollada con Streamlit presenta la predicción del precio de Bitcoin para

el próximo día, junto con métricas de rendimiento relevantes del modelo, como el RMSE (Raíz del Error Cuadrático Medio) calculado sobre los precios reconstruidos y el Test RMSE en escala de diferencia logarítmica. Adicionalmente, se incluye un gráfico que compara la predicción del precio con los precios reales históricos, proporcionando una representación visual del desempeño del modelo, ver Figura 3.10.

Figura 3.10: Interfaz de la aplicación web en Azure, mostrando la visualización de predicciones y métricas. El Run ID y los valores en la captura son ejemplos de una ejecución de desarrollo para ilustrar la funcionalidad. (Elaboración propia)

La configuración de la aplicación web y su despliegue se realizaron a través de un Dockerfile. El código fuente, tanto de esta aplicación web como de los *scripts* y aplicaciones en C# y Python desarrollados para la obtención y procesamiento de datos (detallados en la Sección 3.1), se encuentra gestionado en un repositorio de GitHub. Se implementó un *pipeline* de Integración Continua/Entrega Continua (CI/CD) utilizando GitHub Actions para la aplicación web. De esta manera, cada vez que se realiza un cambio y se sube (push) al repositorio de GitHub, se activa automáticamente un proceso de construcción de una nueva imagen de Docker, ver Figura 3.11.

Esta imagen de Docker resultante es posteriormente subida al registro de contenedores de Azure Container Registry. A su vez, Azure Container Apps, el servicio de despliegue de contenedores elegido para esta aplicación, se configura para monitorear el Azure Container Registry. Al detectarse una nueva imagen, Azure Container Apps automáticamente actualiza la aplicación web desplegada,

garantizando que siempre esté en ejecución la última versión del visualizador.

La decisión de utilizar Azure Container Apps se fundamentó en la capacidad de este servicio para ofrecer escalamiento automático en función de la demanda. Esta característica asegura que la aplicación web pueda manejar un número variable de usuarios y solicitudes sin comprometer su rendimiento o disponibilidad.

La arquitectura implementada permite una visualización accesible y dinámica de las predicciones del precio de Bitcoin generadas por el modelo de Machine Learning, facilitando la comprensión de los resultados y el seguimiento del desempeño del modelo a lo largo del tiempo. La automatización del despliegue a través de Docker, GitHub Actions y Azure Container Registry asegura un proceso eficiente y continuo de actualización de la aplicación web.

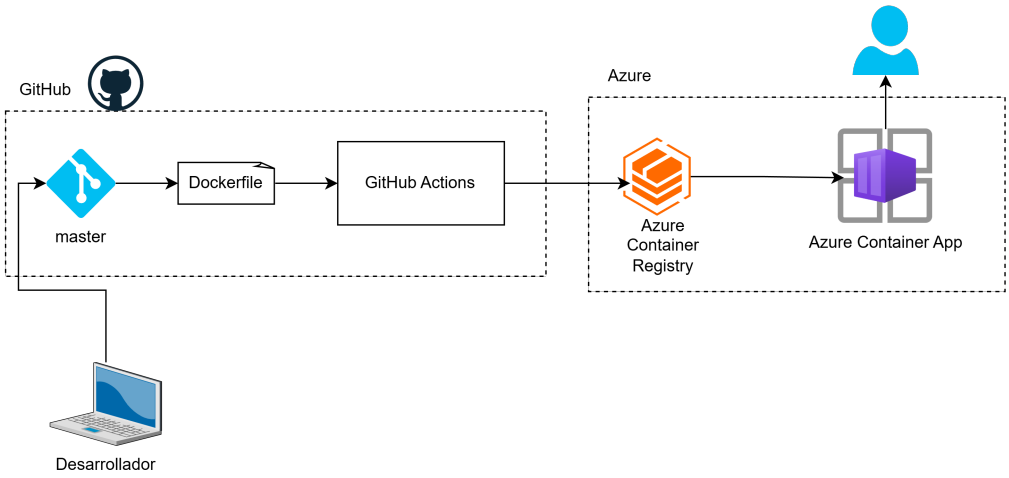


Figura 3.11: Arquitectura de despliegue CI/CD para la aplicación web. (Elaboración propia)

3.6. Resumen del capítulo

Este capítulo detalló el proceso integral de desarrollo llevado a cabo para el proyecto de predicción del precio de Bitcoin. Se inició con la fase de adquisición y preparación de datos, donde se exploraron diversas fuentes de información de la *blockchain* y *exchanges*, se superaron limitaciones mediante la instalación de un nodo Bitcoin local y se implementaron pipelines para la transformación de los datos brutos en series temporales horarias consolidadas, almacenadas finalmente en Azure Data Lake Storage.

Posteriormente, se abordó el diseño e implementación de una arquitectura MLOps en la nube de Azure, especificando los componentes clave como Azure Databricks, MLflow y Azure Container Apps, orientada a la automatización del entrenamiento, evaluación y despliegue de modelos de Machine Learning.

Posteriormente, se detalló el desarrollo y el entrenamiento de dos modelos de redes neuronales

recurrentes, GRU y LSTM, los cuales fueron implementados con Keras sobre TensorFlow, aprovechando la aceleración por GPU mediante WSL. Se detalló la preparación de los datos de entrada, la arquitectura de los modelos y el uso de MLflow para el seguimiento riguroso de experimentos.

La evaluación comparativa de los modelos entrenados, detallada en la Sección 3.4, se realizó utilizando la métrica MAPE sobre un periodo definido. Los resultados indicaron que el modelo GRU entrenado sin la inclusión de datos de la cadena (MAPE = 1,66 %) alcanzó el menor error predictivo. Este rendimiento fue superior al del modelo LSTM en la misma configuración (MAPE = 1,72 %) y a las versiones de ambos modelos que sí incorporaron datos de la cadena.

Finalmente, se presentó el diseño e implementación de una aplicación web para la visualización de las predicciones, desarrollada con Streamlit. Se describió su arquitectura de despliegue continuo (CI/CD) basada en Docker, GitHub Actions y Azure Container Apps, asegurando la escalabilidad y la actualización automática de la aplicación. El capítulo demostró así la consecución de cada objetivo específico a través de las distintas fases de desarrollo e implementación tecnológica.

Capítulo 4

Evaluación

4.1. Diseño de la evaluación

El diseño del proceso evaluativo se centró en la valoración de tres componentes esenciales del proyecto: el desempeño predictivo del modelo de Machine Learning seleccionado, la funcionalidad y efectividad de la arquitectura MLOps construida, y la operatividad de la aplicación web de visualización.

4.1.1. Evaluación del Modelo Predictivo

Se valorará de forma cuantitativa la aptitud predictiva del modelo GRU, escogido en la Sección 3.4 por su desempeño optimizado al excluir los datos de la cadena. Esta valoración se realizará empleando el conjunto de datos de prueba, que cubre el lapso del 1 de diciembre de 2024 al 9 de marzo de 2025. El Error Porcentual Absoluto Medio (MAPE) se utilizará como la métrica principal para medir el desempeño del modelo, lo que permitirá cuantificar la desviación porcentual promedio de sus predicciones con respecto a los precios efectivos de Bitcoin durante el intervalo de prueba.

4.1.2. Evaluación de la Arquitectura MLOps

La evaluación de la arquitectura MLOps implementada en Azure se centró en su funcionalidad y capacidad para automatizar el ciclo de vida del modelo. Los criterios fueron:

- Verificación de la ejecución automatizada del pipeline: Desde la ingesta y consolidación de datos en Azure Data Lake Storage Gen2 hasta el entrenamiento en Azure Databricks.
- Correcto registro de experimentos: Comprobación de que los parámetros, métricas y artefactos (modelo serializado, escaladores) fueran registrados adecuadamente en MLflow durante las ejecuciones de entrenamiento.
- Funcionalidad del despliegue (o capacidad de despliegue): Verificación de que los artefactos generados permitieran el empaquetado y potencial despliegue en Azure Container Apps (según lo diseñado).

La metodología consistió en ejecutar los componentes del pipeline y verificar los resultados y registros en las plataformas correspondientes (Azure Data Lake, Databricks/MLflow).

4.1.3. Evaluación de la Aplicación Web

La aplicación web de visualización desarrollada con Streamlit se evaluó mediante pruebas funcionales. Los criterios se centraron en verificar que la aplicación cumpliera con sus requisitos principales:

- Correcta visualización de la predicción del precio de Bitcoin para el día siguiente.
- Presentación clara de la métrica de rendimiento MAPE.
- Funcionalidad del gráfico comparativo entre precios históricos reales y predicciones.
- Accesibilidad y operatividad de la aplicación desplegada en Azure Container Apps a través de su URL pública.

La metodología se basó en la interacción directa con la aplicación desplegada, realizando pruebas para cada una de las funcionalidades descritas.

4.2. Resultados de la evaluación

Se exponen a continuación los resultados derivados de la implementación del diseño de evaluación descrito en la sección anterior.

4.2.1. Resultados del Modelo Predictivo

El análisis del rendimiento del modelo GRU, seleccionado en el capítulo anterior por su desempeño superior al excluir los datos de la cadena (Sección 3.4), se evaluó sobre el conjunto de prueba (1 de diciembre de 2024 al 9 de marzo de 2025). La métrica principal para esta evaluación fue el Error Porcentual Absoluto Medio (MAPE).

El modelo GRU, entrenado únicamente con datos de mercado y excluyendo las variables *on-chain*, obtuvo un MAPE promedio de 1,66 %. Este resultado, como se analizó en la sección anterior, sugiere que las variables *on-chain* estudiadas podrían ser más indicativas de tendencias a largo plazo, mientras que los datos de mercado resultaron más efectivos para la predicción diaria durante el periodo de prueba.

Este resultado debe interpretarse en el contexto de la alta volatilidad inherente al precio de Bitcoin. Un MAPE de 1,66 % se considera competitivo, especialmente si se compara con algunos *benchmarks* reportados en la literatura, como los MAPE del 2,96 % (Derbentsev et al., 2020) y 3,48 % (Pratama, 2024). Esto indica que el modelo GRU desarrollado ofrece una precisión adecuada para el desafío que representa la predicción del precio de este activo.

Adicionalmente, la Figura 4.1 presenta una comparación visual. En ella se observan los precios reales de Bitcoin (línea amarilla) junto con las predicciones generadas por el modelo GRU (línea verde) y, a modo comparativo, las del modelo LSTM (línea azul) para el mismo horizonte temporal.

El comportamiento general del modelo GRU, visible en la gráfica (Figura 4.1), muestra que sigue las tendencias del precio real. No obstante, al igual que con otros modelos basados en datos

históricos, se evidencian desafíos particulares. Por ejemplo, cuando la secuencia de precios reales utilizada como entrada para el modelo finaliza cerca de un máximo o mínimo local pronunciado, las predicciones subsiguientes pueden tardar en ajustarse a cambios de tendencia abruptos. El modelo tiende a ser influenciado por los patrones recientes en su ventana de datos de entrada.

En situaciones donde el mercado invierte su tendencia de manera súbita, se generan discrepancias entre la predicción y el precio real.

Para contextualizar este término, un análisis del conjunto de datos de evaluación (1 de diciembre de 2024 al 9 de marzo de 2025), presentados en el anexo resultsnocadena192dias.csv y visualizados en la Figura 4.1, revela que cuando las variaciones diarias absolutas del precio real de Bitcoin superaron aproximadamente el 3,0 %, el error MAPE del modelo GRU tendió a incrementarse de forma notable por encima de su promedio de 1,66 %.

Por ejemplo, el día 05/12/2024, dicho conjunto de datos muestra una variación absoluta del precio real del 4,80 %, asociada con un MAPE del GRU del 3,66 %. De forma similar, el 04/03/2025, una variación absoluta del 6,69 % en el precio conllevó un error del modelo GRU del 7,06 %.

Estos ejemplos, cuya dinámica puede observarse en la Figura 4.1 (particularmente en las fechas indicadas donde la línea de predicción GRU se desvía de la real), sugieren que, para este estudio, cambios diarios absolutos en el precio de esta magnitud o superiores pueden caracterizarse como 'súbitos', dado su impacto observable en la precisión del modelo.

Esto es atribuible a que los modelos, al estimar basándose en secuencias pasadas, tienen limitaciones inherentes para anticipar con alta precisión los puntos de inflexión no prefigurados en los datos de entrenamiento más inmediatos. Este comportamiento subraya la dificultad persistente de la predicción en mercados tan dinámicos y volátiles como el de Bitcoin, donde la información histórica reciente no siempre es un predictor infalible del comportamiento futuro inmediato. A pesar de ello, el modelo GRU seleccionado demuestra una capacidad de seguimiento de la tendencia general del precio.

4.2.2. Resultados de la Arquitectura MLOps

La ejecución y verificación de la arquitectura MLOps confirmaron su funcionalidad general:

- Se comprobó la correcta operación del flujo de datos: la información de las fuentes es extraída y preprocesada mediante programas desarrollados en C#, y posteriormente consolidada utilizando un *script* de Python en el servidor local. Luego, este conjunto de datos es cargado automáticamente a Azure Data Lake Storage Gen2, donde queda disponible para el entrenamiento de modelos. Este flujo asegura la actualización diaria de los datos en la nube.

- La ejecución de los procesos de entrenamiento en el entorno de Azure Databricks demostró el registro exitoso de los experimentos en MLflow. La inspección de dicha plataforma confirmó el correcto almacenamiento de hiperparámetros, la métrica de evaluación MAPE y los artefactos necesarios (como el modelo Keras serializado, los objetos escaladores y la firma del modelo), asegurando así la trazabilidad fundamental para las prácticas MLOps.

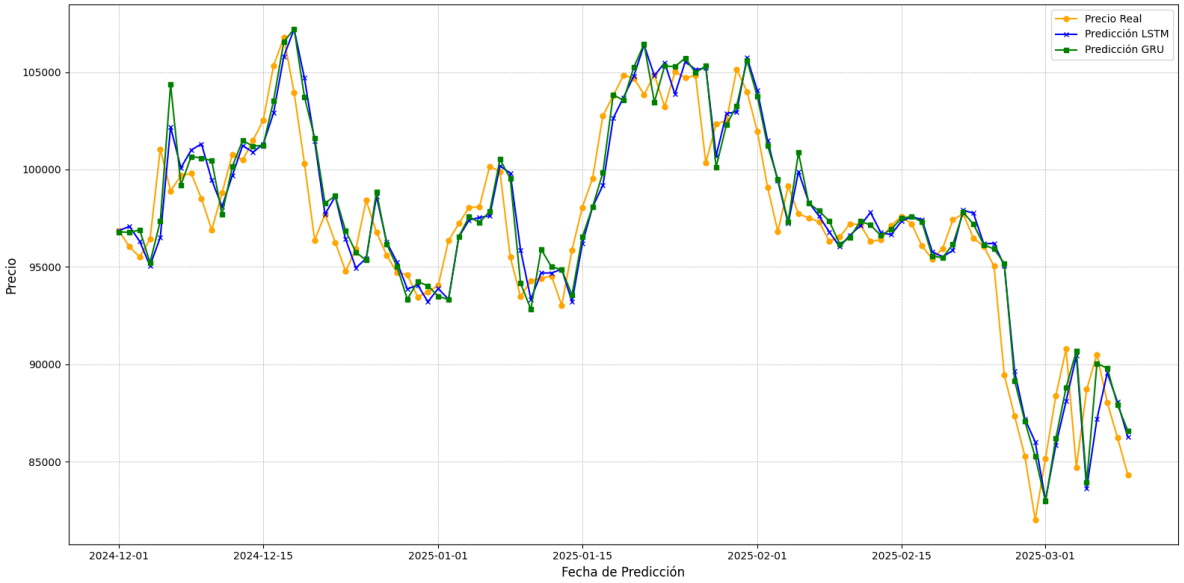


Figura 4.1: Comparación de precio real vs. predicciones de los modelos GRU y LSTM sin datos de la cadena. (Elaboración propia)

- Se validó el ciclo MLOps completo, asegurando la correcta actualización y disponibilidad del modelo, así como el formato adecuado de sus artefactos en MLflow. Un proceso automatizado diario, activado después de la medianoche, sincroniza los datos más recientes, reentrena el modelo y designa la nueva versión como 'Producción' en MLflow. La aplicación web Streamlit consulta dinámicamente esta versión para presentar siempre los resultados y predicciones más actuales.

  Este mecanismo implementa un ciclo automatizado eficaz para el mantenimiento y la actualización periódica del modelo predictivo, complementado por el pipeline CI/CD para el despliegue de la aplicación. La activación diaria programada del reentrenamiento, con carga de datos a las 00:01 y procesamiento a las 00:10, se consideró adecuada. La ejecución completa de estas etapas, usualmente inferior a 30 minutos, simplificó la orquestación del flujo de trabajo.

En resumen, los componentes de la arquitectura MLOps demostraron ser funcionales, permitiendo la gestión de datos, el entrenamiento trazable de modelos y la actualización automática diaria de la versión del modelo en producción visible a través de la aplicación web.

4.2.3. Resultados de la aplicación web

Las pruebas funcionales realizadas sobre la aplicación web desplegada en Azure Container Apps arrojaron resultados satisfactorios:

- La aplicación mostró correctamente la predicción numérica del precio para el día siguiente generada por el modelo GRU cargado.
- La métrica de rendimiento del modelo (MAPE) se presentó de forma clara en la interfaz.
- El gráfico interactivo que compara la serie temporal de precios reales con las predicciones (históricas o la del día siguiente) funcionó según lo esperado, permitiendo una evaluación visual del desempeño.
- La aplicación se mantuvo accesible y operativa a través de su URL pública durante el periodo de evaluación.

La herramienta Streamlit facilitó el desarrollo de una interfaz funcional y útil para la visualización de los resultados del modelo predictivo.

4.3. Resumen del capítulo

Este capítulo presentó el diseño y los hallazgos de la evaluación realizada a los componentes técnicos del proyecto: el modelo predictivo GRU seleccionado, la arquitectura MLOps implementada y la aplicación web de visualización. La estrategia de evaluación se centró en el análisis del Error Porcentual Absoluto Medio (MAPE) para el modelo, la verificación funcional de los procesos MLOps y pruebas operativas de la interfaz de usuario final.

Los resultados cuantitativos confirmaron un rendimiento competitivo del modelo GRU (entrenado sin datos de la cadena) en la predicción del precio de Bitcoin, alcanzando un MAPE promedio de 1,66 %. Este nivel de precisión se consideró relevante, aunque el análisis visual también subrayó las limitaciones ante cambios de tendencia abruptos.

La evaluación de la arquitectura MLOps verificó la correcta operación del registro de experimentos en MLflow y, fundamentalmente, la implementación exitosa de un ciclo automatizado que abarca desde la ingesta diaria de datos hasta el reentrenamiento y actualización del modelo en producción. Finalmente, las pruebas funcionales de la aplicación web desarrollada con Streamlit demostraron que cumple satisfactoriamente con los requisitos de visualización de predicciones, métricas y datos históricos, siendo accesible y operativa en su entorno de despliegue en Azure.

En conclusión, la evaluación realizada confirma que los componentes desarrollados son funcionales y operativos, logrando implementar una solución completa para la predicción del precio de Bitcoin que incluye un ciclo de actualización automatizado y una interfaz de visualización efectiva, cumpliendo así con los propósitos definidos en los objetivos del proyecto.

Capítulo 5

Conclusiones

5.1. Conclusiones

El presente trabajo culminó con el diseño, implementación y evaluación de un prototipo funcional para la predicción del precio de Bitcoin. Este sistema integra técnicas de Machine Learning con una arquitectura MLOps, y sus resultados permiten extraer conclusiones significativas para la predicción de activos volátiles.

Una conclusión fundamental surge del rendimiento del modelo GRU. Al operar exclusivamente con datos históricos de precios y volúmenes de los exchanges, que reflejan la dinámica del mercado a corto plazo, y excluir las métricas *on-chain* de Bitcoin, el modelo alcanzó un Error Porcentual Absoluto Medio (MAPE) de 1,66 %. Este hallazgo subraya la alta capacidad predictiva de los datos de mercado para el horizonte temporal estudiado.

Dicho resultado sugiere que, para el periodo y variables estudiadas, la incorporación de las métricas *on-chain* analizadas podría no ser indispensable para lograr predicciones competitivas. Cabe destacar que el MAPE de 1,66 % es inferior a los reportados en otros estudios de referencia, como 2,96 % (Derbentsev et al., 2020) y 3,48 % (Pratama, 2024), resaltando la efectividad del enfoque.

La implementación de la arquitectura MLOps en Azure demostró ser viable y aportó valor sustancial. Se estableció un sistema con capacidad para el reentrenamiento diario automatizado y la gestión continua del ciclo de vida del modelo, superando la predicción puntual. Esto representa un avance hacia sistemas predictivos más dinámicos y robustos, cruciales para mercados volátiles como el de las criptomonedas.

La operatividad de este flujo automatizado, desde la ingesta de datos hasta la actualización del modelo, sienta bases para herramientas predictivas más confiables y mantenibles. Esta capacidad de adaptación continua a la información reciente del mercado es un diferenciador clave.

El hecho de que el modelo GRU de mejor desempeño no requiriera datos intrínsecos a la cadena de Bitcoin abre perspectivas para la aplicabilidad de la arquitectura a otros activos financieros. Si la dinámica de precios puede capturarse primordialmente desde datos de mercado (OHLCV), el marco MLOps podría adaptarse para predecir precios en diversos instrumentos, ampliando el alcance de la solución.

Finalmente, aunque la predicción precisa del precio de Bitcoin sigue siendo un desafío por su volatilidad, este proyecto demuestra la robustez de combinar modelos de aprendizaje profundo con prácticas MLOps. Se aporta así una solución que no solo busca la precisión, sino que también asegura la relevancia continua de las predicciones mediante la automatización y el monitoreo, ofreciendo una herramienta valiosa para la toma de decisiones.

5.2. Trabajos futuros

Basado en las limitaciones y hallazgos de este proyecto, se proponen las siguientes líneas de investigación futura:

- **Ampliación del conjunto de variables:** Incorporar datos adicionales como análisis de sentimiento (redes sociales, noticias), indicadores macroeconómicos u otras métricas *on-chain*, para la mejora potencial de la capacidad predictiva del modelo.
- **Exploración de modelos alternativos:** Evaluar otras arquitecturas de Machine Learning y Deep Learning (ej. Transformers, CNN-LSTM) o modelos estadísticos (ej. ARIMA, GARCH), para comparar su rendimiento en la predicción del precio de Bitcoin.
- **Predicción a diferentes horizontes temporales:** Desarrollar modelos capaces de realizar predicciones a mediano o largo plazo (semanal, mensual), complementando la predicción diaria actual.
- **Automatización integral del pipeline ETL:** Implementar la automatización completa, incluyendo su posible operación en tiempo real, del pipeline de ingesta, transformación y carga de datos (ETL) desde las fuentes (blockchain, exchanges) hacia Azure Data Lake.
- **Extensión de la metodología a otros activos:** Adaptar y evaluar la arquitectura y metodología desarrolladas para la predicción de precios en otras criptomonedas o activos financieros diversos.
- **Mejora de la robustez ante la volatilidad:** Investigar y aplicar técnicas específicas para mejorar la predicción durante periodos de alta volatilidad o en puntos de inflexión del mercado.
- **Validación para entorno de producción:** Realizar pruebas de robustez, tolerancia a fallos, escalabilidad y seguridad, con el fin de preparar el prototipo para un posible despliegue en un entorno de producción comercial.
- **Optimización de la estrategia MLOps:** Explorar mecanismos de activación del reentrenamiento basados en eventos, como alternativa al enfoque temporal actual, y refinar las estrategias de monitoreo del modelo en producción.

5.3. Lecciones aprendidas

El desarrollo de este proyecto proporcionó diversas lecciones técnicas y metodológicas:

- **Adquisición de Datos:** Se constató la variabilidad en fiabilidad y las limitaciones (costo, cuotas) de las API de terceros para datos de la blockchain. Frente a esto, la implementación de un nodo Bitcoin local, si bien demanda tiempo y almacenamiento, ofrece un acceso directo y sin restricciones a los datos brutos.

- **Procesamiento de Datos:** La elección del lenguaje de programación puede impactar significativamente el rendimiento; C# demostró mayor eficiencia que Python para el procesamiento masivo inicial de bloques de Bitcoin. Asimismo, la agregación temporal (ej. horaria) es un paso necesario para adaptar datos de eventos discretos a formatos de series temporales.
- **Modelado de Series Temporales Financieras:** Transformaciones como la diferencia logarítmica pueden beneficiar el rendimiento de modelos LSTM y GRU. No obstante, predecir con alta precisión la volatilidad y los puntos de inflexión del mercado de Bitcoin constituye un desafío persistente, aun con modelos que capturan dependencias temporales.
- **Implementación de MLOps en la Nube:** Se validó la factibilidad y el valor de integrar servicios en la nube (Azure Data Lake, Databricks, Container Apps) con herramientas como MLflow para la gestión del ciclo de vida del modelo. MLflow resultó particularmente útil para la trazabilidad de experimentos, mientras que la automatización del reentrenamiento basada en tiempo fue eficiente para ciclos de corta duración.
- **Entorno de Desarrollo para Deep Learning:** El uso de WSL en Windows facilitó el acceso a la aceleración por GPU mediante librerías como TensorFlow, optimizando los tiempos de entrenamiento de los modelos.
- **Despliegue Continuo de Aplicaciones Web:** La combinación de Docker, GitHub Actions y servicios de registro y ejecución de contenedores en la nube (Azure Container Registry/Apps) simplifica la automatización del despliegue y la actualización de aplicaciones web.
- **Evaluación Integral de Modelos:** Es fundamental complementar las métricas cuantitativas de error (MAPE, RMSE, MAE) con análisis cualitativos y visualizaciones para una mejor comprensión del rendimiento y las limitaciones del modelo predictivo.

Bibliografía

Adjei, F. (2019). Determinants of bitcoin expected returns. *Journal of Finance and Economics*, 7(1):42–47.

Antonopoulos, A. M. and Harding, D. A. (2023). *Mastering bitcoin*. .ºReilly Media, Inc.".

Auer, R., Cornelli, G., Doerr, S., Frost, J., and Gambacorta, L. (2022). Crypto trading and bitcoin prices: evidence from a new database of retail adoption. Technical Report 1049, Bank for International Settlements.

Barradas, A., Tejeda-Gil, A., and Cantón-Croda, R.-M. (2022). Real-Time Big Data Architecture for Processing Cryptocurrency and Social Media Data: A Clustering Approach Based on k-Means. *Algorithms*, 15(5):140.

Baur, D. G., Hong, K., and Lee, A. D. (2018). Bitcoin: Medium of exchange or speculative assets? *Journal of International Financial Markets, Institutions and Money*, 54:177–189.

Beattie, A. (2014). 5 skills that traders need. *Investopedia*.

Binance (2022). Using mlops to build a real-time end-to-end machine learning pipeline. https://www.binance.com/en/blog/tech/using-mlops-to-build-a-realtime-endtoend-machine-learning-pipeline-3820048062346322706.

Bitcoin.org (2020a). Block Chain — Bitcoin. https://developer.bitcoin.org/devguide/block_chain.html.

Bitcoin.org (2020b). Block Chain — Bitcoin. https://developer.bitcoin.org/reference/block_chain.html.

Block Inc. (2022). 2022 bitcoin survey report: Knowledge and perceptions. https://block.xyz/inside/report-bitcoin-survey-2022.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time series analysis: Forecasting and control*. John Wiley & Sons.

CoinMarketCap (2024). Top cryptocurrency exchanges ranked by volume. https://coinmarketcap.com/rankings/exchanges/.

Danial, K. (2023). *Cryptocurrency investing for dummies*. John Wiley & Sons.

de Prado, M. L. (2018). *Advances in financial machine learning*. John Wiley & Sons.

Derbentsev, V., Datsenko, N., Babenko, V., Pushko, O., and Pursky, O. (2020). Forecasting cryptocurrency prices using ensembles-based machine learning approach. In *2020 IEEE International Conference on Problems of Infocommunications. Science and Technology*, pages 707–712. IEEE.

Fu, D. and Ismail, M. T. (2023). The long short-termmemory (lstm) model combines with technical analysis to forecast cryptocurrency prices. *Matematika*, 39(2):149 – 158.

Garcia, M. (2023). Real-Time data architecture patterns - DZone refcardz. https://dzone.com/refcardz/real-time-data-architecture-patterns.

Géron, A. (2022). *Hands-On machine learning with Scikit-Learn, Keras, and TensorFlow*. .ºReilly Media, Inc.".

Han, J., Kamber, M., and Pei, J. (2011). *Data mining: Concepts and techniques*. Elsevier.

Hayes, A. (2003). Volatility: Meaning in finance and how it works with stocks. *Investopedia*.

Higuchi, T. (2022). Mlops Blog Series Part 1: The art of testing machine learning systems using MLOps. https://azure.microsoft.com/en-us/blog/mlops-blog-series-part-1-the-art-of-testing-machine-learning-systems-using-mlops/.

Huang, C. and Petukhina, A. (2022). *Applied time series analysis and forecasting with python*. Springer Nature.

Humble, J. and Farley, D. (2010). *Continuous delivery: Reliable software releases through build, test, and deployment automation*. Pearson Education.

Huyen, C. (2022). *Designing Machine Learning Systems*. .ºReilly Media, Inc.".

Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: Principles and practice*. OTexts.

James, G., Witten, D., Hastie, T., Tibshirani, R., and Taylor, J. (2023). *An introduction to statistical learning: With applications in python*. Springer.

Jansen, S. (2020). *Machine learning for algorithmic trading : predictive models to extract signals from market and alternative data for systematic trading strategies with Python*. Packt Publishing.

Joseph, M. and Tackes, J. (2024). *Modern Time Series Forecasting with Python: Industry-ready machine learning and deep learning time series analysis with PyTorch and pandas*. Packt Publishing Ltd.

Kaabar, S. (2024). *Deep learning for finance*. .ºReilly Media, Inc.".

Kreuzberger, D., Kühl, N., and Hirschl, S. (2023a). Machine learning operations (mlops): Overview, definition, and architecture.

Kreuzberger, D., Kühl, N., and Hirschl, S. (2023b). Machine learning operations (mlops): Overview, definition, and architecture. *IEEE Access*, 11:31866–31879.

Lang, H. and McGee, S. (2024). How ETFs and institutions are driving the surge in Bitcoin prices. *Reuters*.

Lauchande, N. (2021). *Machine Learning Engineering with MLflow: Manage the end-to-end machine learning life cycle with MLflow*. Packt Publishing Ltd.

Lazzeri, F. (2020). *Machine learning for time series forecasting with python*. John Wiley & Sons.

Little, R. J. A. and Rubin, D. B. (2019). *Statistical analysis with missing data*. John Wiley & Sons.

Merritt, R. (2020). What is MLOps? https://blogs.nvidia.com/blog/what-is-mlops/.

Mogal, S. (2024). Ohlc meaning: How does it help you trade? https://www.venturasecurities.com/blog/ohlc-meaning-how-does-it-help-you-trade/.

Moyo, C. and Phiri, A. (2023). Re-examining bitcoin's price-volume relationship: A time-varying spectral analysis. *Journal of Risk and Financial Management*, 16(7):324.

Murphy, J. J. (1999). *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

Palacio, A. B. (2021). *Distributed Data Systems with Azure Databricks: Create, deploy, and manage enterprise data pipelines*. Packt Publishing Ltd.

Palma, W. (2016). *Time series analysis*. John Wiley & Sons.

Pardalis, K. (2023). Mlops is 98https://mlops.community/mlops-is-mostly-data-engineering/.

Peixeiro, M. (2022). *Time series forecasting in Python*. Simon and Schuster.

Pratama, A. (2024). Bitcoin cryptocurrency price prediction using ifa-bilstm. *2024 2nd International Conference on Technology Innovation and Its Applications (ICTIIA), Technology Innovation and Its Applications (ICTIIA), 2024 2nd International Conference on*, pages 1 – 6.

Pratas, T. E., Ramos, F. R., and Rubio, L. (2023). Forecasting bitcoin volatility: Exploring the potential of deep learning. *Eurasian Economic Review*, 13(2):285–305.

Raj, E. (2021). *Engineering MLOps: Rapidly build, test, and manage production-ready machine learning life cycles at scale*. Packt Publishing Ltd.

Serrano, L. (2021). *Grokking machine learning*. Simon and Schuster.

Soni, R. (2024). https://x.com/rajatsonifnance/status/1752359807095357764.

The Investopedia Team (2023). What is block time? What it measures, verification, and example. https://www.investopedia.com/terms/b/block-time-cryptocurrency.asp.

Torpey, K. (2024). Bitcoin's dominance of crypto market reaches highest level since 2021. https://www.investopedia.com/bitcoin-dominance-of-crypto-market-reaches-highest-level-since-2021-8744927.

XTB (2022). La historia del Bitcoin. https://www.xtb.com/es/educacion/la-historia-del-bitcoin.

Yamak, P. T., Yujian, L., and Gadosey, P. K. (2019). A comparison between ARIMA, LSTM, and GRU for time series forecasting. In *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, New York, NY, USA. ACM.

Zhang, J., Cai, K., and Wen, J. (2024). A survey of deep learning applications in cryptocurrency. *iScience*, 27(1):108509.

Zherlitsyn, D. (2024). *Python for Finance: Data analysis, financial modeling, and portfolio management (English Edition)*. BPB Publications.

Zvorničanin, E. (2025). Building mlops pipeline for time series prediction [tutorial]. https://neptune.ai/blog/mlops-pipeline-for-time-series-prediction-tutorial.