Predicciones de criptomonedas utilizando un modelo de prediccion basado en el algoritmo de bosque aleatorio

Estudiante:

Esteban Trochez Hernandez

Asesor:

Julian Andres Ayala Ruiz

Universidad Nacional Abierta y a Distancia – UNAD

Escuela de Ciencias Básicas, Tecnología e Ingeniería ECBTI

Ingeniería de Sistemas

2024

Resumen

Las criptomonedas han llevado a muchos inversores a intentar predecir el precio de las criptomonedas para intentar aumentar sus ganancias. Para animar a estos inversores, se proporciona una herramienta de predicción que les ayudará a tomar decisiones de inversión. En este proyecto se crearán modelos de predicción basados en el algoritmo de Bosque Aleatorio para realizar predicciones de los valores de las criptomonedas. Esta técnica propuesta ayuda a crear modelos predictivos del precio de cierre basados en características seleccionadas. Al realizar las predicciones, los resultados se evalúan a partir del conjunto de validación. La metodología que estructura este proyecto es la metodología CDIO, la cual es una herramienta para abordar problemas complejos en cuatro etapas: concepción, diseño, implementación y operación. Este artículo tiene como objetivo crear modelos de predicción basados en el algoritmo de Bosque Aleatorio para realizar predicciones precisas sobre la valoración a corto plazo de las criptomonedas Bitcoin y Ethereum. Los resultados de todos los modelos de predicción se comparan y luego se analizan para determinar cuál es el modelo de predicción más eficaz. Al final, después de aplicar el modelo seleccionado para la predicción, este modelo se comparará con los resultados del modelo CNN-LSTM mencionado en la revisión de la literatura para demostrar que el modelo propuesto es una mejora en términos de precisión de predicción.

Palabras clave: Criptomoneda, Random Forest, predicciones, CDIO, algoritmo.

Abstracto

Cryptocurrencies have led many investors to try to predict the price of cryptocurrencies in order to try and increase their profits. To encourage these investors, a prediction tool is provided to help them make investment decisions. In this project, prediction models based on the Random Forest algorithm will be created to make predictions of cryptocurrency values. This proposed technique helps to create predictive models of the closing price based on selected characteristics. When making predictions, the results are evaluated from the validation set. The methodology that structures this project is the CDIO methodology, which is a tool to address complex problems in four stages: conception, design, implementation and operation. This article aims to create prediction models based on the Random Forest algorithm to make accurate predictions about the short-term valuation of Bitcoin and Ethereum cryptocurrencies. The results of all prediction models are compared and then analyzed to determine which is the most effective prediction model. In the end, after applying the model selected for prediction, this model will be compared with the results of the CNN-LSTM model mentioned in the literature review to demonstrate that the proposed model is an improvement in terms of prediction accuracy.

Keywords: Cryptocurrency, Random Forest, predictions, CDIO, algorithm.

Tabla de Contenido

Lista de Tablas

Lista de Figuras

Lista de Apendices

Introducción

El mundo del dinero y las finanzas se está transformando ante nuestros ojos. Los clientes exigen servicios financieros modernos, nuevos canales de entrega y activos digitalizados en esta era digital. Las innovaciones resultantes cambian el paradigma, generan diferentes conductos para que el capital monetario esté disponible y cambian las transacciones financieras que ocurren en todo el mundo (Hileman & Rauchs, 2017).

Las cifras de criptomonedas están creciendo y la atención del público en general y de los inversores se está volviendo hacia ellas (Raiborn & Sivitanides, 2015). Dado su perfil en los medios y su fuerte aumento de valor, Bitcoin ha recibido un interés significativo por parte de los medios, los fondos de cobertura y el público y, junto con la escasez de literatura sobre Bitcoin, en la última década ha visto un aumento considerable en el interés académico. investigación sobre el tema de Bitcoin y otras criptomonedas (Swankie, 2019). Este proyecto pretende contribuir a la literatura proporcionando una herramienta de pronóstico que incluye ciertas variables para predecir el precio de Ethereum y Bitcoin a través de un modelo de predicción basado en Bosque Aleatorio de Aprendizaje Automático.

Un momento notable en el mercado de las criptomonedas es el "boom de las criptomonedas en 2017" (Reiff, 2020). Aunque su uso aumentó, la investigación en el área y el comercio de estas monedas permanecen en una zona gris debido a su volatilidad, lo que dificulta hacer predicciones precisas. Además, las criptomonedas están descentralizadas, disponibles las 24 horas del día, los 7 días de la semana y sin supervisión. Todas estas condiciones empeoran la posibilidad de mejorar su predicción.

El desarrollo de un algoritmo para predecir los precios de las monedas digitales altamente volátiles presenta una oportunidad para generar márgenes de beneficio lucrativos. Sería posible

obtener cientos de por ciento de ganancias en cuestión de semanas, incluso si el valor de esa criptomoneda disminuye (McCoy y Rahimi, 2020).

En este proyecto, el modelo de predicción basado en el algoritmo de Bosque Aleatorio predecirá los precios de Bitcoin y Ethereum como representante de las criptomonedas más populares y volátiles del mercado. Esperamos que estas dos criptomonedas también puedan ayudar a ofrecer un buen predictor para otras criptomonedas.

Las herramientas utilizadas para este proyecto son Google Colaboratory y Python porque Google Colaboratory tiene un tiempo de ejecución completamente configurado para aprendizaje profundo y acceso gratuito a una GPU robusta (Carneiro et al., 2018). Python es el lenguaje de programación elegido porque la amplia selección de bibliotecas y marcos de aprendizaje automático de Python simplifica el proceso de desarrollo y acorta el tiempo de desarrollo. La sintaxis simple y la legibilidad de Python fomentan la prueba rápida de algoritmos complejos y hacen que el lenguaje sea accesible para los no programadores (Protasiewicz, 2018).

Al final, los resultados obtenidos se compararán con el modelo CNN-LSTM de (Livieris et al., 2021). Esta comparación tiene como objetivo demostrar si el modelo propuesto es una mejora en términos de precisión de predicción.

Justificacion

La economía global y el mercado de criptomonedas han evolucionado de forma significativa en los últimos años. Las criptomonedas, lideradas por Bitcoin y Ethereum, siguen teniendo un papel destacado en la economía digital, pero su volatilidad es muy sensible a factores macroeconómicos, como las políticas monetarias y las tasas de interés, así como a factores políticos y sociales. En 2024, la regulación de las criptomonedas es un tema candente en diversas economías, y los cambios legislativos en países influyentes, junto con eventos económicos globales, pueden desencadenar variaciones abruptas en sus precios (Coinbase, 2024; ManTech Publications, 2024).

En este contexto, aunque se reconoce que estos factores macroeconómicos y políticos intervienen en la volatilidad del criptomercado, este trabajo se enfocará en la construcción de modelos de predicción específicos basados en algoritmos de aprendizaje automático, particularmente el algoritmo de Random Forest. Este enfoque permite captar patrones en los datos de mercado, minimizando el impacto de factores impredecibles y centrándose en la estructura interna de precios de criptomonedas como Bitcoin y Ethereum. Dada la complejidad y la volatilidad del criptomercado, el algoritmo de Random Forest ofrece ventajas al manejar grandes volúmenes de datos y múltiples variables, adaptándose bien a la naturaleza caótica del mercado (FundCount, 2024).

Este trabajo se sustenta en la realidad actual del criptomercado: el interés de grandes inversionistas ha aumentado, especialmente desde la entrada de instituciones como BlackRock y JPMorgan, lo que proporciona una base de confianza para otros actores del mercado (Coinbase, 2024). Sin embargo, el mercado sigue siendo extremadamente volátil y puede cambiar drásticamente en cuestión de días. Aunque Bitcoin y Ethereum dominan la atención y los

estudios, existen miles de altcoins que reciben menos investigación, lo cual crea una oportunidad para explorar su comportamiento y patrones de precios.

Nuestra contribución en este proyecto radica en construir un modelo predictivo que, mediante el uso del algoritmo de Random Forest, pueda ofrecer una perspectiva sobre el comportamiento de estas criptomonedas. Se pretende analizar Bitcoin y Ethereum como criptomonedas representativas abordando así el desafío de un mercado especulativo y dinámico (ManTech Publications, 2024; FundCount, 2024).

Finalmente, este trabajo es de especial relevancia para mi formación, ya que integra dos áreas en auge: el aprendizaje automático y las criptomonedas. Aunque los programas académicos actuales incluyen temas de inteligencia artificial, la práctica de aprendizaje automático aún es limitada. Este proyecto representa una oportunidad para aplicar algoritmos avanzados y contribuir al conocimiento en un área de relevancia actual y de interés creciente.

Objetivos

Objetivo General

Crear un modelo de predicción basado en el algoritmo de Bosque Aleatorio para pronosticar los precio de las dos criptomonedas más conocidas (Bitcoin y Ethereum) que estarían disponibles en GitHub para que los inversores puedan utilizarlas más tarde para sus predicciones de criptomonedas.

Objetivos Específicos

Adquirir datos históricos limpios y confiables de los precios de apertura, máximo, mínimo y cierre de Bitcoin y Ethereum, para entrenar un conjunto de modelos con los datos históricos a través de Google Colaboratory y código Python, de modo que el modelo mejor propuesto entregue resultados adecuados en la etapa de validación.

Entrenar el conjunto de modelos de predicción creados con los datos históricos a través de Google Collaboratory y código Python de manera que el mejor modelo entregue resultados adecuados en la etapa de validación.

Adaptar y optimizar un modelo de predicción basado en el algoritmo de Bosque Aleatorio para pronosticar con precisión los precios de Bitcoin y Ethereum, logrando una tasa de error baja y un porcentaje de exactitud del modelo superior al 95%.

Comparar los resultados con el modelo CNN-LSTM mencionado en la revisión de la literatura con el fin de demostrar si el modelo propuesto es una mejora en términos de precisión de predicción y en tasa de errores.

Subir el código fuente creado en Google Collaboratory a Github para que los inversores y grupos de investigación puedan cargar datos históricos y obtener pronósticos actualizados de los

precios de Bitcoin y Ethereum utilizando el modelo de predicción creado basado en el Bosque Aleatorio.

Alcance del Proyecto

Este proyecto se centra específicamente en la adaptación de un modelo de predicción basado en el algoritmo de Bosque Aleatorio para pronosticar los precios de Bitcoin y Ethereum, dos de las criptomonedas más prominentes y comercializadas en el mercado. A pesar de la extensa variedad de criptomonedas disponibles, hemos elegido estas dos debido a su importancia y liquidez en el ecosistema financiero actual.

En cuanto al conjunto de datos, nos limitaremos a utilizar información histórica exclusiva de Bitcoin y Ethereum. Esta decisión se basa en la necesidad de mantener la coherencia y la relevancia del proyecto, evitando la complejidad que podría surgir al incluir datos de otras criptomonedas.

La implementación técnica del proyecto se llevará a cabo en Google Collaboratory utilizando Python. Esto significa que las soluciones y resultados que presentaremos estarán estrechamente vinculados a estas herramientas y tecnologías específicas. Por tanto, no se considerarán otras plataformas o lenguajes de programación en el desarrollo y presentación de este proyecto. El código utilizado para entrenar ambos modelos se proporciona en el Apéndice C.

Marco de Referencia

Marco Conceptual

Bosque Aleatorio

El Bosque Aleatorio (también llamado Random Forest o RF) es un algoritmo de aprendizaje conjunto introducido por (Breiman, 2001) para impulsar el método del árbol de decisión. RF se originó a partir del algoritmo de ensacado (Breiman, 1996) y constaba de varios árboles de decisión independientes entre sí (Liaw y Wiener, 2002). RF se emplea a menudo en problemas de regresión y clasificación. Se generan aleatoriamente diferentes muestras de

conjuntos de entrenamiento a partir del conjunto de entrenamiento inicial, empleando un método de muestreo bootstrap en su operación. Luego se llevan a cabo procesos de árboles de decisión (también llamados Decision Trees o DT) en los nuevos conjuntos de entrenamiento (Chen et al., 2018) (Hefner et al., 2014).

Aunque el algoritmo de RF generalmente produce mejores resultados que los algoritmos DT, el algoritmo de RF requiere una cantidad sustancial de datos etiquetados para proporcionar resultados precisos (Oskar et al., 2006). Además, este tipo de clasificador tiene varias ventajas: se requieren pocos parámetros de entrada, el algoritmo es resistente al sobre ajuste y la varianza disminuye al aumentar el número de árboles sin generar sesgos (Meidan et al., 2017).

Evaluacion del Modelo – Errores

RMSE (Error Cuadrático Medio). La varianza muestral de las diferencias entre los valores observados y anticipados, también denominada residual, se conoce como RMSE. Se calcula matemáticamente mediante la siguiente fórmula:

Figura 1

Fórmula RMSE

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

Fuente. https://medium.com/usf-msds/choosing-the-right-metric-for-machine-learning-models-part-1-a99d7d7414e4

MAE (Mean Absolute Error/Error Absoluto medio). Se calcula la diferencia media absoluta (MAE) entre los valores observados y esperados. Dado que el MAE es una puntuación lineal, cada diferencia individual tiene el mismo peso en la media. Por ejemplo, la diferencia entre 0 y 10 será el doble que entre 0 y cinco. Se calcula matemáticamente mediante la siguiente fórmula:

Figura 2

Fórmula MAE

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

Fuente. https://medium.com/usf-msds/choosing-the-right-metric-for-machine-learning-models-part-1-a99d7d7414e4

R cuadrado (R^2) y R cuadrado ajustado. El R cuadrado y el R cuadrado modificado proporcionan información sobre qué tan bien ciertas variables independientes describen la variación de sus variables dependientes y se utilizan ampliamente en educación. Matemáticamente, R Cuadrado está dado por:

Figura 3

Fórmula R^2

$$\hat{R}^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2} = 1 - \frac{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

Fuente. https://medium.com/usf-msds/choosing-the-right-metric-for-machine-learning-models-part-1-a99d7d7414e4

MAPE (Mean Absolut Percentage Error/Error Porcentual Absoluto Medio). El R cuadrado y el R cuadrado modificado proporcionan información sobre qué tan bien ciertas variables independientes describen la variación de sus variables dependientes y se utilizan ampliamente en educación. Matemáticamente, R Cuadrado está dado por:

Figura 4

Fórmula MAPE

$$\mathrm{M} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

Fuente. https://vedexcel.com/how-to-calculate-mape-in-python/

Hiperparámetros

N_Estimators: int, default=100. Este parámetro es la cantidad de árboles dentro del bosque.

Max_Depth: int, default=None. Esto se refiere a la mayor profundidad del árbol. Los nodos se dispersan hasta que cada hoja sea pura, o si se selecciona Ninguno, hasta que haya menos muestras que las muestras min_samples_split.

Min_Samples_Split: int or float, default=2. Para separar un nodo interno se requiere un mínimo de muestras que es la cantidad mínima si es int y si es flotante, entonces la división mínima de muestras es una fracción y la cantidad mínima de muestras por división es ceil (división mínima de muestras * n muestras).

Min_Samples_Leaf: int or float, default=1. Son las cuántas muestras que se requieren en el nodo hoja. Solo se considerará un punto de división a cualquier profundidad cuando proporcione un mínimo de muestras de entrenamiento min_samples_leaf en las ramas izquierda

y derecha. Esto puede conducir a la nivelación del modelo, particularmente en regresión. Si es int, el número mínimo debe considerarse min_samples_leaf. El número mínimo de muestras para cada nodo se indica mediante ceil (min_samples_leaf * n_samples) si es un flotante (Sklearn.Ensemble.RandomForestRegressor — Scikit-Learn Documentation 0.24.2, 2021).

Ajuste de hiperparámetros

Búsqueda manual. Sus criterios se utilizan en la búsqueda manual de parámetros; esto se hace basándonos en el uso de nuestros criterios o experiencias pasadas con métodos de aprendizaje automático.

En comparación con otros métodos, el método de búsqueda manual suele ser el método menos eficaz. Evaluamos varias combinaciones de hiperparámetros en un esfuerzo por identificar los resultados óptimos. No siempre es fácil encontrar los parámetros ideales del modelo mediante una búsqueda manual. Aunque a menudo se respetan los juicios de los expertos, no son necesariamente exactos. Como se demuestra en este trabajo con la cantidad de árboles en el modelo, los bucles se pueden usar para evaluar un parámetro en Python.

Búsqueda Aleatorio. Se utiliza una configuración aleatoria de hiperparámetros para construir, entrenar y probar una colección de parámetros con varias posibilidades. Cuando se emplean combinaciones arbitrarias de parámetros, se utiliza un número predeterminado de repeticiones para muestrear los parámetros en lugar de probar cada parámetro.

La validación cruzada es esencial para las operaciones de optimización de hiperparámetros. De esta manera, podemos evitar algunos hiperparámetros que funcionan increíblemente bien con datos de entrenamiento pero terriblemente con pruebas.

Es más eficiente que la búsqueda manual en términos tanto de eficiencia como de duración de cálculo (Suenaga, 2018).

Búsqueda con Grid Search. Para un conjunto seleccionado manualmente, se lleva a cabo una búsqueda exhaustiva de hiperparámetros y se examina cada combinación.

Este método es muy costoso desde el punto de vista computacional debido a su largo cálculo, pero garantiza la mejor combinación de conjuntos posible (Feurer & Hutter, 2019).

¿Qué es una Vela?

En el análisis técnico del mercado de criptomonedas y otros activos financieros, una "vela" o "candlestick" es una representación gráfica que muestra el comportamiento del precio de un activo en un intervalo de tiempo específico. Cada vela contiene cuatro datos clave: el precio de apertura, el precio de cierre, el máximo y el mínimo alcanzados en el período. Esta visualización es fundamental en el análisis de la acción del precio y es ampliamente utilizada para identificar tendencias y patrones que ayuden a los inversores a prever posibles movimientos futuros del mercado (Investopedia, 2023; CFA Institute, 2023).

La estructura de cada vela se compone de dos elementos principales: el "cuerpo" y las "mechas" o "sombras". El cuerpo de la vela refleja la diferencia entre el precio de apertura y el de cierre del activo durante el período especificado. Cuando el precio de cierre supera al de apertura, el cuerpo de la vela suele aparecer en verde o blanco, indicando una tendencia alcista. En cambio, si el precio de cierre es menor que el de apertura, el cuerpo de la vela se representa en rojo o negro, señalando una tendencia bajista. Las mechas o sombras, que sobresalen en la parte superior e inferior del cuerpo de la vela, representan los puntos máximo y mínimo alcanzados, mostrando así la volatilidad y el rango de precios del activo en ese intervalo (Murphy, 1999; Nison, 2001).

Este tipo de análisis, conocido como análisis de velas japonesas, tiene sus raíces en el mercado japonés de arroz del siglo XVIII y fue popularizado en los mercados financieros

occidentales por Steve Nison en su obra *Japanese Candlestick Charting Techniques* (Nison, 2001). A través de patrones de velas, como el "martillo" (hammer) o el "doji," los analistas pueden interpretar el comportamiento del mercado y anticipar posibles cambios de tendencia, lo que es especialmente útil en mercados volátiles como el de las criptomonedas, donde los cambios de precio son abruptos y significativos (Nison, 2001; Murphy, 1999).

Criptomonedas

¿Qué es la Criptomoneda?. Las criptomonedas son activos digitales que emplean técnicas criptográficas para asegurar sus transacciones y regular la emisión de nuevas unidades. Desde la aparición de Bitcoin en 2009, han cambiado la percepción y uso de los medios de intercambio y las inversiones. Estos activos se sustentan en tecnologías de registro distribuido, especialmente la blockchain, la cual mantiene un historial de transacciones transparente y descentralizado (Nakamoto, 2008).

Un rasgo característico de las criptomonedas es su descentralización. A diferencia de las monedas tradicionales emitidas y supervisadas por los gobiernos, las criptomonedas operan en redes de igual a igual (peer-to-peer), sin intervención de autoridades centrales. Esto les otorga una mayor resistencia a la censura y manipulación, además de facilitar el acceso financiero a quienes no cuentan con servicios bancarios convencionales (Catalini & Gans, 2016; Zohar, 2015).

En cuanto a su uso, las criptomonedas tienen aplicaciones diversas, que abarcan desde la transferencia de valor hasta contratos inteligentes que ejecutan acuerdos automáticamente cuando se cumplen ciertas condiciones. Estas funcionalidades han acelerado la adopción de criptomonedas a nivel global. No obstante, la volatilidad de estos activos y su empleo en

actividades ilícitas han impulsado la atención regulatoria y el debate sobre su futuro en el sistema financiero mundial (Foley, Karlsen, & Putniņš, 2019).

A medida que avanza la tecnología y la regulación, el futuro de las criptomonedas sigue siendo incierto, pero su impacto en la economía digital es innegable. Las criptomonedas están desafiando las nociones tradicionales de dinero y ofreciendo nuevas oportunidades para la inversión y la innovación en los servicios financieros (Catalini & Gans, 2016; Zohar, 2015).

¿Qué es Bitcoin?. Bitcoin, creada en 2009 por un individuo o grupo bajo el nombre de Satoshi Nakamoto, es la primera criptomoneda y aún mantiene un rol central en el ámbito financiero mundial. Funciona como un sistema de efectivo digital que permite transacciones directas entre personas sin la intervención de intermediarios, asegurando la integridad y privacidad de las transacciones mediante el uso de criptografía (Nakamoto, 2008).

La tecnología subyacente de Bitcoin, la blockchain, continúa siendo un elemento innovador. Este registro descentralizado permite verificar transacciones de manera transparente y segura, lo que ha llevado a un aumento en la adopción de Bitcoin no solo como medio de intercambio, sino también como una reserva de valor (Antonopoulos, 2014; Zohar, 2015). Además, la creciente aceptación institucional, con empresas y fondos de inversión que ahora consideran a Bitcoin como parte de sus carteras, ha contribuido a establecer su legitimidad en el mercado financiero global (Coinbase Institutional, 2024).

Sin embargo, el entorno regulatorio de Bitcoin ha sido objeto de un creciente escrutinio. A medida que más países implementan regulaciones para los activos digitales, Bitcoin enfrenta desafíos en términos de cumplimiento y adopción. Por ejemplo, en 2024, regulaciones más estrictas en Europa y Estados Unidos están llevando a debates sobre la necesidad de equilibrar la innovación con la protección del consumidor (Foley, Karlsen, & Putniņš, 2019). Esto ha

generado un interés renovado en las implicaciones de la regulación de criptomonedas y su impacto en el futuro de Bitcoin y otros activos digitales.

A pesar de los desafíos, Bitcoin sigue siendo un líder indiscutible en el ámbito de las criptomonedas, marcando el rumbo hacia un futuro donde la tecnología blockchain y los activos digitales podrían redefinir el sistema financiero mundial (Catalini & Gans, 2016).

¿Qué es Ethereum?. Ethereum es una plataforma blockchain descentralizada que permite desarrollar contratos inteligentes y aplicaciones descentralizadas (dApps) sin la intervención de intermediarios. Lanzada en 2015 por Vitalik Buterin, Ethereum ha ampliado el uso de la tecnología blockchain al hacer posible la ejecución de transacciones automatizadas y complejas (Buterin, 2013).

Su criptomoneda nativa, Ether (ETH), se utiliza para pagar tarifas de transacción en la red. A diferencia de Bitcoin, Ethereum no cuenta con un límite de suministro, lo cual afecta su comportamiento en el mercado. En 2023, ETH ha experimentado un notable crecimiento debido al aumento en la adopción de dApps y al desarrollo del ecosistema de finanzas descentralizadas (DeFi) (CoinMarketCap, 2024; DappRadar, 2023).

Sin embargo, Ethereum enfrenta desafíos, como la escalabilidad y las altas tarifas de transacción. La transición a Ethereum 2.0, que implementa un mecanismo de prueba de participación (PoS), busca abordar estos problemas y mejorar la eficiencia de la red (Kahn, 2021; Ethereum Foundation, 2024).

A medida que la regulación en torno a las criptomonedas se intensifica, el futuro de Ethereum dependerá de su capacidad para adaptarse a las normativas y mantener su posición como líder en el espacio de blockchain (Foley et al., 2019).

Marco Jurídico

El marco jurídico constituye un conjunto de disposiciones legales esenciales para regir el despliegue y la utilización del modelo de aprendizaje automático, asegurando el cumplimiento legal, la protección de los derechos de propiedad intelectual y la promoción de prácticas éticas.

Protección de la propiedad intelectual

Derechos de Autor. Se reconoce la titularidad de los derechos de autor del creador sobre el modelo de aprendizaje automático y los materiales asociados, otorgándole la exclusividad para su reproducción, distribución y comunicación pública.

Patentes. Se evalúa la posibilidad de proteger mediante patentes los aspectos innovadores del modelo o sus algoritmos, garantizando al creador el derecho exclusivo de explotación durante un período determinado.

Licencia. Se establecen los términos y condiciones para la concesión de licencias del modelo a terceros usuarios, definiendo las limitaciones de uso, las regalías y los derechos de sublicencia, con el propósito de regular su difusión y utilización.

Responsabilidad e Indemnización

Limitación de Responsabilidad. Se especifican las limitaciones a la responsabilidad del creador por los daños derivados del uso del modelo, salvo en casos de conducta dolosa o negligencia grave, protegiendo así sus intereses frente a posibles reclamaciones.

Indemnización. Se establece la obligación de los usuarios de indemnizar al creador por cualquier reclamación, pérdida o responsabilidad surgida del uso del modelo, garantizando una compensación adecuada por los perjuicios causados.

Uso justo y consideraciones éticas

Directrices Éticas. Se establecen principios éticos que deben regir el desarrollo y la implementación del modelo, tales como la equidad, la transparencia y la responsabilidad social, promoviendo así su utilización ética y responsable.

Política de Uso Justo. Se definen normas y criterios para el uso legal y ético del modelo, con el fin de prevenir prácticas abusivas o discriminatorias, garantizando así su aplicación en consonancia con los principios de justicia y equidad.

Marco Tecnológico

La importancia del marco tecnológico radica en la necesidad de establecer una estructura sólida que facilite el despliegue y uso del modelo en Google Colab por parte de los usuarios, garantizando así su eficacia y accesibilidad.

Arquitectura y Componentes del Modelo

Descripción del Modelo. Se proporciona una explicación detallada de la arquitectura del modelo, los algoritmos utilizados y las metodologías implementadas en el proceso de aprendizaje automático, brindando una comprensión clara de su funcionamiento.

Componentes y Dependencias. Se identifican las bibliotecas de software, herramientas y dependencias externas necesarias para el funcionamiento óptimo del modelo en Google Colab, asegurando su correcta integración y ejecución.

Interfaz de Usuario e Interacción

Diseño de Interfaz de Usuario. Se establecen pautas de diseño para crear una interfaz intuitiva y fácil de usar que permita a los usuarios interactuar con el modelo de aprendizaje automático dentro de Google Colab de manera eficiente y efectiva.

Formatos de Entrada y Salida. Se definen los formatos de datos de entrada aceptados por el modelo y los formatos de salida generados por el mismo, garantizando la compatibilidad con las entradas del usuario y las salidas deseadas.

Implementación y Accesibilidad del Modelo

Estrategias de Implementación. Se discuten diversas opciones y estrategias de implementación para hacer que el modelo de aprendizaje automático sea accesible para los usuarios en Google Colab, teniendo en cuenta consideraciones de rendimiento, escalabilidad y seguridad.

Uso Compartido y Colaboración: Se proporcionan pautas para compartir el modelo con otros usuarios, fomentando el desarrollo colaborativo y la colaboración dentro de la comunidad de Google Colab, lo que permite una mayor difusión y utilización del modelo.

Este marco tecnológico sirve como guía para desarrolladores y profesionales que buscan implementar modelos de aprendizaje automático en Google Colab, ofreciendo orientación sobre arquitectura, integración, implementación, optimización del rendimiento y mecanismos de soporte para garantizar una experiencia de usuario fluida y productiva.

Marco Legal

La creciente importancia de las criptomonedas en los mercados globales requiere una comprensión integral de las implicaciones legales que rodean su uso y los datos empleados en el modelado predictivo. En esta sección se establece la importancia de abordar las preocupaciones regulatorias y éticas en el desarrollo de dichos modelos.

Entorno regulatorio para las criptomonedas

Descripción general de las regulaciones de criptomonedas. El panorama regulatorio de las criptomonedas varía ampliamente según las jurisdicciones. En los Estados Unidos, los organismos reguladores como la Comisión de Bolsa y Valores (SEC) han comenzado a clasificar ciertas criptomonedas como valores, lo que lleva a regulaciones más estrictas. Por el contrario, otras jurisdicciones, como El Salvador, han adoptado políticas más indulgentes, reconociendo a Bitcoin como moneda de curso legal. Comprender estos marcos regulatorios es crucial para garantizar el cumplimiento y mitigar los riesgos legales en la implementación del modelo.

Estatus legal de Bitcoin y Ethereum. Bitcoin y Ethereum a menudo se clasifican como productos básicos según la ley de EE. UU., que los somete a las regulaciones de la Comisión de Comercio de Futuros de Productos Básicos (CFTC). Esta clasificación afecta a las prácticas comerciales y a la protección de los inversores. La clasificación legal influye en la forma en que se pueden recopilar, analizar y comunicar los datos en el contexto de la predicción de sus precios.

Leyes de privacidad y protección de datos

Reglamento General de Protección de Datos (RGPD). El RGPD impone requisitos estrictos sobre la recopilación, el procesamiento y el almacenamiento de datos dentro de la Unión Europea. Todos los datos personales utilizados en el entrenamiento del modelo deben cumplir con los principios del RGPD, incluida la minimización de datos, la limitación de la finalidad y la obtención del consentimiento explícito del usuario. La anonimización de los datos es esencial para evitar violaciones de los derechos individuales de privacidad.

Gestión de Riesgos y Responsabilidad

Responsabilidad legal por las predicciones. Las imprecisiones en las predicciones de precios pueden provocar pérdidas financieras para los inversores. Establecer exenciones de responsabilidad claras y advertencias de riesgo junto con los resultados del modelo puede ayudar a gestionar las preocupaciones de responsabilidad. Comprender los estándares legales para el asesoramiento financiero es fundamental para mitigar posibles reclamaciones.

Mitigación de riesgos. La implementación de estrategias de gestión de riesgos, como la realización de pruebas retrospectivas del modelo con datos históricos y la incorporación de actualizaciones periódicas basadas en las condiciones del mercado, puede reducir el riesgo de bajo rendimiento y las ramificaciones legales asociadas.

Marco Teorico

En esta sección, resumimos algunas estrategias y algoritmos de aprendizaje automático que funcionaron en modelos de predicción. La lista de artículos y algoritmos de aprendizaje automático explorados se encuentra en la Tabla 1. Proporcionamos su año, tipo de publicación y técnica de aprendizaje automático.

Tabla 1

Lista de artículos por año, tipo y técnica de aprendizaje automático

Autores	Año	Tipo	Tecnica
D. Shah and K. Zhang	2014	Documento de sesión	Regresión bayesiana
S. Colianni, S. Rosales and M. Signorotti	2015	Informe de investigación	Regresion Logistica, Naïve Bayes,

			Máquinas de Vectores de Soporte
C. Lamon, E. Nielsen and E. Redondo	2016	Informe de investigación	Regresion Logistica, Naïve Bayes, Máquinas de Vectores de Soporte
H.S. Yin and R. Vatrapu	2017	Documento de sesión	Bosques aleatorios, bosques extremadamente aleatorios, embolsado, aumento de gradiente
S. McNally, J. Roche, and S. Caton	2018	Documento de sesión	Red neuronal recurrente, memoria a corto plazo
Z. Jiang and J. Liang	2018	Documento de sesión	Red neuronal convolucional
L. Alessandretti, A. ElBahrawy, L. M. Aiello and A. Baronchelli	2018	Documento de Trabajo	Memoria larga a corto plazo, aumento extremo de gradiente
T. R. Li, A. S. Chamrajnagar, X.	2018	Documento de Trabajo	Aumento de gradiente extremo

R. Fong, N. R. Rizik and F. Fu			
W. Chen, Z. Zheng, J. Cui, E. Ngai, P. Zheng, and Y. Zhou	2018	Documento de sesión	Aumento de gradiente extremo
Julien Chevallier, Dominique Guégan and Stéphane Goutte	2021	Documento de sesión	Red Neuronal Artificial, Máquina de Vectores de Soporte, Bosque Aleatorio, K Vecinos Más Cercanos, AdaBoost, Regresión de Ridge
Livieris, Ioannis E. Kiriakidou, Niki Stavroyiannis, Stavros Pintelas, Panagiotis	2021	Documento de sesión	Red neuronal convolucional

Fuente. Elaboracion Propia

Regresión Bayesiana

(Shah & Zhang, 2014) utilizaron el método bayesiano en su estudio para predecir las variaciones de valor de Bitcoin y construir una estrategia rentable de comercio de criptomonedas.

Su estrategia casi duplica la inversión en una cartera de Bitcoin en menos de sesenta días, una vez que se compara con información comercial real de los intercambios de criptomonedas. En comparación con la regresión bayesiana, Random Forest es más versátil para predecir los precios de las criptomonedas.

Un modelo basado en Random Forest maneja relaciones complejas y no lineales y no depende de suposiciones previas sólidas, lo que lo hace mejor para capturar la imprevisibilidad de los mercados de criptomonedas. La regresión bayesiana proporciona información probabilística y funciona bien con patrones lineales más simples, pero puede tener dificultades con la volatilidad del mercado y la dinámica no lineal. Random Forest ofrece mayor flexibilidad y precisión, aunque la regresión bayesiana es más interpretable y proporciona estimaciones de confianza para las predicciones.

Naïve Bayes

(Colianni et al., 2015) informaron sobre la posibilidad de distinguir diez movimientos del valor de Bitcoin basándose en el análisis de sentimiento de Twitter. Bernoulli Naive Bayes obtuvo el mejor rendimiento de todos los algoritmos de clasificación de texto al lograr una precisión diaria del 95,00 % y una precisión hora a hora del 76,23 %. (Lamon et al., 2016) realizaron un análisis más detallado al incluir conocimiento de los titulares de las noticias diarias y agregar otra criptomoneda (Ethereum) a su modelo. El clasificador Bernoulli Naive Bayes proporcionó las mejores predicciones de precios de Ethereum en este proyecto. El modelo fue muy eficaz a la hora de predecir aumentos de precios, con un 75,8% de los días predichos correctamente. La precisión de la predicción de las caídas de precios fue mucho menor, de solo el 16,1%.

Un modelo basado en Random Forest es mejor para predecir los precios de las criptomonedas que Naive Bayes porque maneja relaciones complejas y no lineales e interacciones de características, que son comunes en los mercados financieros. Ofrece mayor precisión y es más resistente al ruido y la volatilidad, mientras que Naive Bayes simplifica demasiado al suponer independencia de las características.

Bosque Aleatorio

(Yin y Vatrapu (2017) realizaron un estudio para estimar la proporción de entidades cibercriminales dentro del ecosistema Bitcoin. La lista de los cuatro clasificadores principales en su proyecto es (clasificadores de bosque aleatorio, bosque extremadamente aleatorio, ensacado y potenciador de gradiente) según la precisión de la validación cruzada (77.38%, 76.47%, 78.46%, 80.76% respectivamente).

Red Neuronal Recurrente

(McNally et al., 2018) utilizaron redes de memoria a corto plazo (LSTM) para predecir los movimientos de Bitcoin. Su análisis muestra que los LSTM están listos para alcanzar una precisión de clasificación del 52% al predecir la dirección a largo plazo de los precios de Bitcoin. (Alessandretti et al., 2018) también emplearon redes LSTM para analizar datos diarios de 1681 criptomonedas y desarrollaron un modelo de predicción para cada criptomoneda. Luego les dio el poder para planificar una estrategia comercial que supere los puntos de referencia promedio.

Un modelo basado en Random Forest es más eficiente e interpretable que las redes neuronales recurrentes (RNN) para predecir los precios de las criptomonedas. Si bien las RNN son mejores para capturar patrones dependientes del tiempo, requieren una cantidad significativa de datos y recursos computacionales, y pueden ser difíciles de entrenar. Random Forest, por otro

lado, maneja bien las relaciones no lineales, es más rápido de implementar y menos propenso al sobreajuste, aunque carece de la capacidad de modelar datos secuenciales como las RNN.

Aumento de Gradiente

Tres estudios de análisis diferentes han utilizado el aumento de gradiente y técnicas conectadas, como el aumento de gradiente extremo. Estos estudios desarrollan modelos predictivos relacionados con los valores de las criptomonedas (Alessandretti et al., 2018; Li et al., 2018) y observan esquemas Ponzi en el mercado de Ethereum (Chen et al., 2018).

Al comparar un modelo basado en Random Forest con Gradient Boosting para predecir los precios de las criptomonedas, Gradient Boosting suele ofrecer una mayor precisión al centrarse en reducir los errores mediante modelos mejorados secuencialmente. Sin embargo, puede ser más lento de entrenar y más sensible al sobreajuste, especialmente en mercados volátiles como el de las criptomonedas. Random Forest, aunque suele ser menos preciso, es más rápido, más fácil de ajustar y más resistente al ruido. Es una mejor opción cuando la interpretabilidad, la velocidad y la estabilidad son prioridades, mientras que Gradient Boosting se favorece por su mayor precisión en modelos bien ajustados.

Red Neuronal Convolucional

(Jiang & Liang, 2018) presentan una red neuronal convolucional sin modelo. El modelo utiliza el conocimiento del historial de valores de una colección de 220 criptomonedas diferentes. El rendimiento de su modelo supera a tres puntos de referencia y tres algoritmos de gestión de cartera diferentes. Con los modelos de red de memoria a corto plazo (LSTM) utilizados en (Livieris et al., 2021), el modelo fusiona los datos procesados obtenidos de los vectores de salida de las capas LSTM y los procesa aún más para realizar la predicción final. La precisión obtenida

en este estudio con bitcoin oscila entre el 51,88% y el 55,03% y entre el 49,57% y el 51,51% para Ethereum.

En comparación con las redes neuronales convolucionales (CNN), un modelo basado en Random Forest es más sencillo y eficiente para predecir los precios de las criptomonedas. Las CNN se destacan en el reconocimiento de patrones dentro de datos espaciales, como imágenes, pero su aplicación a series temporales financieras es menos intuitiva y requiere una adaptación especializada. Random Forest, por otro lado, es más adecuado para datos tabulares estructurados, ya que ofrece una implementación más rápida y una interpretación más sencilla. Si bien las CNN pueden capturar patrones ocultos y complejos cuando se adaptan a datos financieros, Random Forest suele ser más confiable y más fácil de aplicar sin una personalización extensa del modelo.

Metodologia

Explicación de la Metodología

Este proyecto tiene como objetivo crear un modelo de predicción basado en el algoritmo de Bosque Aleatorio para realizar predicciones precisas de la valoración a corto de las dos criptomonedas más conocidas (Bitcoin y Ethereum). Para ello, la metodología CDIO da estructura al proyecto.

La metodología CDIO fue seleccionada para este proyecto porque es un marco educativo innovador para formar a la próxima generación de líderes en ingeniería. La industria se beneficia porque CDIO produce ingenieros que tienen el conocimiento, el talento y la experiencia que necesita. Los educadores están interesados porque el plan de estudios CDIO constituye una base para la planificación curricular y la evaluación basada en resultados que es universalmente adaptable para todas las escuelas de ingeniería. Y los estudiantes están entusiasmados porque se gradúan con una colección única de experiencias personales, interpersonales y de creación de

sistemas que les permite sobresalir en equipos de ingeniería reales y crear nuevos productos y sistemas (CDIO, 2010).

Etapas de la Metodología

Concepción

Sugerimos la predicción de una colección de criptomonedas en la etapa de concepción. En un breve período de tiempo, los valores previstos de las criptomonedas se compararán con los precios reales mediante un conjunto de validación. Junto a las herramientas y métodos a emplear, también se especifican las etapas y fases que se llevarán a cabo. Cada paso necesita tener información sobre los recursos a utilizar, la duración y las variables de entrada y salida. Una vez finalizada la primera fase, se lleva a cabo una evaluación de los métodos y recursos.

Diseño

Recopilación y Procesamiento de Datos. Tenemos que decidir qué datos son requeridos para lograr el objetivo planteado. Es necesario seleccionar datos pertinentes y brindar una explicación para el proceso de selección. Obtener suficientes datos requiere saber qué proporciona cada variable y confirmar su precisión.

Después de completar estos procesos, es fundamental realizar una limpieza de datos, que implica utilizar la cantidad adecuada de datos y eliminar cualquier información potencialmente falsa o vacía de la estructura de datos. En esta etapa, se agregan datos que podrían ser cruciales para el modelo de predicción, como procedimientos que mejoran nuestra comprensión y ayudan a construir el modelo de predicción o la evaluación de si se necesitan más datos de los que no tenemos actualmente.

La combinación de tablas a menudo puede conducir a un proceso de comprensión más ágil y eficaz. Necesitamos examinar estas modificaciones y determinar si mejorarán la técnica del Bosque Aleatorio cuando la usemos nuevamente.

Exploración de Datos. Para la toma de decisiones de minería de datos, esta fase es crucial. Se requiere un análisis cuantitativo de todos los datos recopilados para determinar los valores históricos más altos, más bajos, medias, variaciones, etc.

Los números recopilados nos proporcionarán información precisa sobre la estabilidad del modelo de predicción. Ahora hemos identificado posibles valores atípicos, que debemos investigar más a fondo en caso de que incluyan errores. Numerosos programas realizan análisis estadísticos y simplifican todos estos datos, lo cual es crucial para la aplicación precisa del modelo de predicción.

Obtención de áreas de Formación y Validación. Para llevar a cabo tanto el entrenamiento como la validación de los modelos predictivos que se desarrollarán más adelante, los datos se dividen en dos subconjuntos en este paso.

Implementación

Construcción de los Modelos Manuales de Predicción. En este paso, los datos investigados se utilizan para elegir la técnica a utilizar. Se pueden utilizar numerosos enfoques, pero en función de los datos disponibles, se debe elegir el más eficaz. Es muy útil poder utilizar múltiples procedimientos y comparar los resultados. Por ejemplo, emplear Random Forest para lograr nuestros objetivos.

Antes de crear los modelos de predicción, debemos considerar nuestra estrategia de evaluación. En numerosos métodos, los datos se extraen de la base de datos y se utilizan en dos partes: una para entrenar el modelo y la otra para evaluar su eficacia. Si se necesita más de un

modelo para encontrar el que mejor cumpla con el objetivo previsto, entonces se pueden construir más.

En esta etapa de desarrollo de modelos de predicción, algunos modelos se construyen utilizando herramientas para la optimización de parámetros, mientras que otros se construyen utilizando una variedad de criterios (basados en la reducción del error de predicción).

Cabe mencionar de que en la fase de desarrollo solo se mostrara el procedimiento realizado con los datos de ETHUSD1d, ya que la cantidad de árboles, profundidad y otros hiperparámetros utilizados varía dependiendo de los datos utilizados, por lo tanto, se explicara las diferencias de los hiperparámetros utilizados para ETHUSD 1d y BTCUSD 1d en la sección de resultados.

Explotación y validación de los Modelos Manuales de Predicción. Cada modelo manual de predicción creado se investiga y se verifica su eficacia en este paso. Es fundamental comprobar que los modelos evaluados funcionan correctamente, seguido de una selección previa de los modelos de predicción que mejores resultados producen.

En este punto, también se aclaran las deficiencias de cada modelo que se descubrieron durante el análisis de cada modelo. Al realizar un análisis exhaustivo e integral, es imperativo evaluar si esta actividad se alinea con los objetivos principales establecidos en la primera fase. Estos defectos pueden ayudar indirectamente a determinar si es necesario repetir alguna actividad desde que fracasó el desarrollo del modelo.

En este paso de la evaluación es donde se revelan, aunque sutilmente, las deficiencias de cada modelo, incluido si pudimos omitir un algoritmo y si tuvimos que rehacer una tarea debido a un error en la construcción del modelo de predicción.

Después de conseguir los resultados del análisis de explotación y validación de los modelos manuales de predicción, se elige el modelo definitivo de búsqueda manual y se nombrará el modelo “modified_final”.

Construcción e implementación de los modelos de búsqueda aleatoria y Grid Search. Además de elegir las variables más cruciales o aplicar ingeniería de variables (manipularlas), en muchos métodos de aprendizaje automático, el uso de datos de entrenamiento adicionales es una forma típica de mejorar los modelos. En este caso, se utilizará la búsqueda aleatoria y Grid_Search para intentar una búsqueda autónoma de hiperparámetros.

Operación

En esta etapa de operación, una vez realizado el proceso de análisis, se utiliza para la predicción el modelo con la puntuación de precisión más alta y las mejores estadísticas de error.

Al finalizar el proyecto se debe redactar un informe final que presente los resultados. Debe ser sintetizado y resumido.

En este punto, es fundamental evaluar si todo salió exactamente según lo planeado y, en caso contrario, identificar las áreas que aún requieren mejoras y por qué.

Desarollo

Ejecución del Desarrollo

El par Ethereum - dólar estadounidense se utilizará como ejemplo para demostrar el proceso. Aunque se utilizan varios parámetros, el proceso es relativamente similar, por lo que los hallazgos que se presentarán no cubrirán el procedimiento completo para evitar repeticiones.

Desarrollo de la Etapa de Concepción

Hoy en día, se puede anticipar con bastante confianza el precio de cualquier instrumento financiero, como Bitcoin o las acciones de Google, gracias al aprendizaje automático y a las

estadísticas. Sin embargo, los expertos en Big Data todavía están trabajando en algunas estrategias para aumentar la precisión de los modelos predictivos, de modo que los inversores puedan utilizar plenamente la herramienta.

Por lo tanto, para abordar la incertidumbre financiera, es necesaria construir y evaluar un modelo de predicción basado en el algoritmo de Random Forest. Sugerimos predecir un conjunto de monedas virtuales en esta investigación. Utilizando un conjunto de validación, se creará un pronóstico de los valores de las criptomonedas y luego, durante un breve período de tiempo, se comparará con los precios reales.

Para realizar la predicción vamos a considerar la vela de 1 día para obtener los precios de las criptomonedas de Ethereum y Bitcoin.

Desarrollo de la Etapa de Diseño

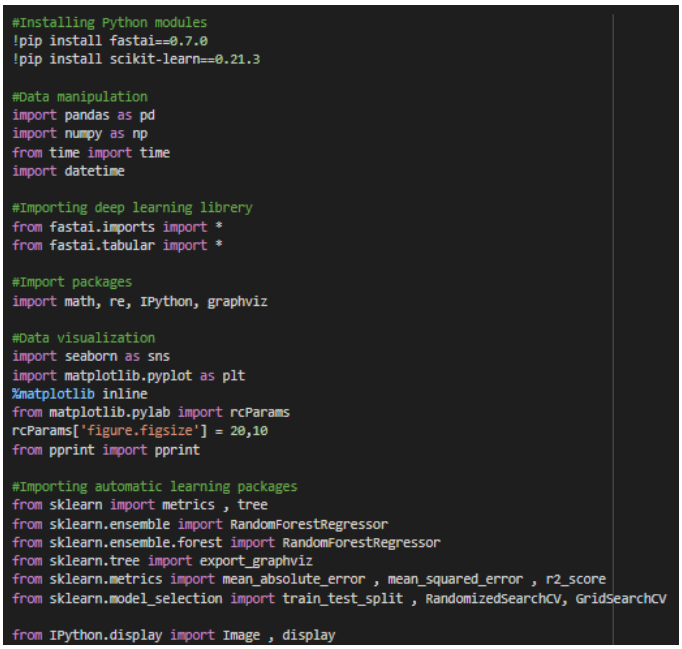
Recopilación y Manejo de Datos

Instalación e importación de la biblioteca Python. Primero, es necesario importar las bibliotecas que se utilizarán en Python. Como se comentó anteriormente, este paso se realiza desde Google Colaboratory. Una vez descargado e instalado, se importan las bibliotecas a utilizar.

Las librerías que se muestran después de esto se importaron para obtener los datos de criptomonedas necesarios, crear un modelo y examinar sus resultados.

Figura 5

Librerías Utilizadas para la construcción de los modelos predictivos



```
#Installing Python modules
!pip install fastai==0.7.0
!pip install scikit-learn==0.21.3

#Data manipulation
import pandas as pd
import numpy as np
from time import time
import datetime

#Importing deep learning librery
from fastai.imports import *
from fastai.tabular import *

#Import packages
import math, re, IPython, graphviz

#Data visualization
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib.pylab import rcParams
rcParams['figure.figsize'] = 20,10
from pprint import pprint

#Importing automatic learning packages
from sklearn import metrics , tree
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble.forest import RandomForestRegressor
from sklearn.tree import export_graphviz
from sklearn.metrics import mean_absolute_error , mean_squared_error , r2_score
from sklearn.model_selection import train_test_split , RandomizedSearchCV, GridSearchCV

from IPython.display import Image , display
```

Fuente. Elaboración Propia

Recopilación de Datos. Actualmente, se puede acceder a datos históricos en formato CSV (valores separados por comas) para algunas criptomonedas seleccionadas. Cada archivo incluye los datos de precios de cada criptomoneda. El valor al utilizar ambas criptomonedas se muestra en dólares estadounidenses.

Los datos del 12 de febrero del 2020 al 19 de enero del 2024 para la vela de 1 día de Ethereum y Bitcoin se obtuvieron a través de la página de Kaggle (Kaggle permite a los usuarios buscar y publicar conjuntos de datos).

Procesamiento de Datos. Los archivos CSV que se descargaron y almacenaron previamente se cargan utilizando la biblioteca Pandas y el código Python que sigue. Además, se crea una nueva columna que contiene información sobre el precio de cierre de la próxima vela.

Figura 6

La recopilación de datos para el par ETH-dólar por hora

```
df = pd.read_csv("ETH-USD (12-2-20:19-1-24).csv")
df.index = df['timestamp']
df['y']=df['Close']
```

Fuente. Elaboración Propia

El marco de datos generado se ve a continuación, donde se insertó la nueva columna y se copiaron los datos del archivo CSV. Los índices ya no son numéricos porque cada entrada ha sido renombrada con su fecha y período de tiempo real.

8 columnas y 1438 filas componen cada archivo de precios que se recibe una vez al día; Una columna adicional y exactamente la misma cantidad de filas estarán presentes en el marco de datos. El comando que sigue verifica que los datos se ingresaron correctamente.

Figura 7

Tamaño del marco de datos (filas, columnas)

```
print(df.info())
print('\n')
print("Número de muestras:", df.shape[0])
print("Número de variables:", df.shape[1])
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1438 entries, 2/12/20 to 1/19/24
Data columns (total 8 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   timestamp  1438 non-null   object
 1   Open       1438 non-null   float64
 2   High       1438 non-null   float64
 3   Low        1438 non-null   float64
 4   Close      1438 non-null   float64
 5   Adj Close  1438 non-null   float64
 6   Volume     1438 non-null   int64
 7   y          1438 non-null   float64
dtypes: float64(6), int64(1), object(1)
memory usage: 101.1+ KB
None


Número de muestras: 1438
Número de variables: 8
```

Fuente: Elaboración Propia

Como se mencionó anteriormente, los marcos de datos recopilan datos históricos de varias fechas en función de la vela. Algunos de ellos no tienen relación con nuestra investigación. La estructura de datos generada se compone de:

- Timestamp: el momento en que comenzó el intervalo de tiempo.
- Open: el precio en el que comienza cada criptomoneda dentro del período de tiempo especificado.
- High: la cantidad que representa el precio más alto de cada criptomoneda dentro del rango dado.
- Low: el importe del precio mínimo de cada criptomoneda dentro del rango predeterminado.

- Close: el monto al que se cierra cada criptomoneda dentro del período de tiempo determinado.
- Adj Close: es el precio de cierre ajustado de una accion o active financiero.
- Volume: es la cantidad de un activo invertido durante un período de tiempo predeterminado.
- y: precio de cierre de la vela más uno. Utilizando los atributos de la vela n, se producirá una predicción y se compararán los resultados reales.

Selección de Variables. Tanto las variables independientes como las dependientes se han seleccionado utilizando código Python. Dado que el resultado está vinculado al número de variables asociadas con el precio de OHLC, la selección de las variables independientes se realizó únicamente sobre la base de la intuición. No obstante, el párrafo siguiente ilustra el análisis realizado de las relaciones.

La variable dependiente que requiere predicción es el precio de cierre de la vela n+1, a menudo conocida como objetivo de producción. En otras palabras, nuestro objetivo es investigar el precio del intervalo siguiente utilizando variables independientes de la vela n. Esta variable está reunida en un marco de datos de una sola columna al que nos referiremos como y_df.

Las variables independientes de Open, Close, High y Low de la vela n son los atributos que se utilizaron para crear la predicción. Todo está reunido en un marco de datos al que nos referiremos como X_df. Se han encontrado correlaciones bajas entre el resto de los atributos y la variable objetivo en los modelos de las distintas criptomonedas: volumen, número de operaciones, etc.

Se pensó que sería más eficiente utilizar Python para gestionar las estructuras de datos y producir un archivo CSV que contenga toda la información posible.

Figura 8

Selección de Variables y Creación de un Nuevo Marco de Datos

```
df_small = df
features = ['Open', 'High', 'Low', 'Close']

X_df = df_small[features]
y_df = df_small['y']
```

Fuente. Elaboración Propia

Para crear un modelo más eficaz, se examinarán los factores más significativos en un esfuerzo por determinar qué variables independientes tienen el mayor impacto en el modelo de predicción.

Correlación de variables. Se muestra la correlación entre las variables. Se debe recordar que la variable "ignorar" se eliminó porque no proporcionaba al modelo ninguna información reveladora.

Figura 9

Correlación de Pearson

	open	high	low	close	Volume ETH	Volume USD	y
open	1.000000	0.998359	0.995446	0.996012	0.161654	0.672002	0.996012
high	0.998359	1.000000	0.995101	0.998298	0.170887	0.686361	0.998298
low	0.995446	0.995101	1.000000	0.997042	0.138174	0.626964	0.997042
close	0.996012	0.998298	0.997042	1.000000	0.158693	0.665275	1.000000
Volume ETH	0.161654	0.170887	0.138174	0.158693	1.000000	0.511953	0.158693
Volume USD	0.672002	0.686361	0.626964	0.665275	0.511953	1.000000	0.665275
y	0.996012	0.998298	0.997042	1.000000	0.158693	0.665275	1.000000

Fuente. Elaboración Propia

Existe una aceleración constante y positiva entre las variables de apertura, máximo, mínimo y cierre. Por el contrario, esperábamos que la variable objetivo dependiera más del

volumen. Por esta razón excluiremos esta característica final en el proceso de la creación del modelo. Se obtuvo la siguiente correlacion:

- **Correlación de y – volumen: 0.1586.**

Exploración de Datos

Ahora sería un buen momento para analizar la información recopilada. Seguiremos usando el mismo archivo CSV que hace referencia al par ETH/USD como punto de referencia para aclarar. En el siguiente análisis, el número de muestras, los promedios, las desviaciones de la media, los máximos y mínimos y los percentiles se muestran en columnas.

Figura 10

Valores Estadísticos de las Variables Cuantitativas

	Open	High	Low	Close	Adj Close	Volume	y
count	1438.000000	1438.000000	1438.000000	1438.000000	1438.000000	1.438000e+03	1438.000000
mean	1769.261990	1818.757827	1715.188179	1770.570489	1770.570489	1.620229e+10	1770.570489
std	1087.985131	1120.083983	1050.717559	1086.848838	1086.848838	1.034877e+10	1086.848838
min	110.406784	116.021622	95.184303	110.605873	110.605873	2.081626e+09	110.605873
25%	1168.572876	1203.884979	1094.777130	1170.523285	1170.523285	8.867603e+09	1170.523285
50%	1725.129944	1772.583618	1674.309571	1726.336121	1726.336121	1.421193e+10	1726.336121
75%	2339.297180	2400.433350	2238.290893	2342.927063	2342.927063	2.040704e+10	2342.927063
max	4810.071289	4891.704590	4718.039063	4812.087402	4812.087402	8.448291e+10	4812.087402

Fuente. Elaboración Propia

Podemos observar en la figura anterior que un valor anormal o impar a priori no existe, por lo que no sería necesario examinar nuestros archivos en busca de errores. Es evidente que el precio más alto es de $4891.70, mientras que el más bajo es de $95.18. Además, el marco de datos generado tiene todas las filas que se anticipó que estarían allí y no tiene valores nulos.

Obtención de áreas de Formación y Validación

Los datos de entrenamiento y los datos de prueba son los dos subconjuntos de nuestros datos que se utilizan para resolver problemas de regresión utilizando el algoritmo de Bosque Aleatorio El proceso de creación del modelo puede comenzar después de importar las bibliotecas y de haber revisado y editado minuciosamente el archivo. En un nuevo marco de datos, seleccionamos solo las funciones: Open(Apertura), High(Alto), Low(Bajo) y Close(Cerrar).

Una vez más se producirán dos marcos de datos, uno con los atributos necesarios para entrenar el modelo y otro con la variable que se debe predecir para evaluar el rendimiento de los modelos mostrados.

Luego dividimos nuestros datos usando las utilidades de Python. Luego dividimos nuestros datos usando las utilidades de Python. La siguiente figura muestra el código que se utilizó:

Figura 11

Creación de Conjuntos de Entrenamiento y Validación

```
X_train, X_val, y_train, y_val = train_test_split(X_df, y_df, test_size=0.2, random_state=42, shuffle=False)
```

Fuente. Elaboración Propia

La figura adjunta ilustra cómo están organizados los conjuntos de entrenamiento y validación. En particular, los conjuntos generados por las cuatro variables independientes tienen cuatro columnas como se predijo, mientras que los conjuntos de variables que se utilizarán para la validación y el entrenamiento tienen solo una columna.

Figura 12

Estructura de Conjuntos

```
print('Training Features Shape:', X_train.shape)
print('Training Labels Shape:', y_train.shape)
print('Testing Features Shape:', X_val.shape)
print('Testing Labels Shape:', y_val.shape)
```

```
Training Features Shape: (1150, 4)
Training Labels Shape: (1150,)
Testing Features Shape: (288, 4)
Testing Labels Shape: (288,)
```

Fuente. Elaboración Propia

La función train_test_split() divide el conjunto de datos original en dos conjuntos ordenados: el conjunto de entrenamiento, que constituye el 80% del total, y el conjunto de prueba más reciente, que constituye el 20% restante.

La función se puede repetir simplemente configurando el parámetro random_state, que establece la semilla del generador de números aleatorios.

Desarrollo de la Etapa de Implementación

Construcción e Implementación de los Modelos Manuales de Predicción

Después de generar los conjuntos de entrenamiento y validación, utilizamos la búsqueda manual para construir modelos de manuales de predicción. Inicialmente, se construirán modelos y luego se modificarán sus hiperparámetros en un esfuerzo para mejorarlos. Después de eso, se emplearán métodos para optimizar los hiperparámetros.

Antes de crear cualquier modelo de predicción, se construyeron 2 funciones, print_score y rmse. Estas funciones nos permiten determinar el RMSE y verificar los valores de R2 en ambos conjuntos. Los errores y parámetros se mostrarán juntos en el capítulo de resultados, pero

primero se proporcionarán algunos resultados y las modificaciones realizadas a cada modelo para aclarar el proceso que se siguió.

Figura 13

Funciones para mostrar los resultados del modelo

```
[20] #Functions to evaluate models (RMSE and R² of training and validation set)

     def rmse(x,y): return math.sqrt(((x-y)**2).mean())
     def print_score(m):
         res = [rmse(m.predict(X_train), y_train),
                rmse(m.predict(X_val), y_val),
                     m.score(X_train, y_train),
                     m.score(X_val, y_val)]
         print(f"RMSE training set: {res[0]} \nRMSE validation set: {res[1]} \nR² training set: {res[2]} \nR² validation set: {res[3]}")
```

Fuente. Elaboración Propia

default_model. Con la excepción del parámetro random_state, que elimina la aleatorización cada vez que se ejecuta el modelo, el primer modelo se genera con valores predeterminados. El modelo se entrena después de su construcción. La construcción y el entrenamiento del modelo se muestran en la figura adjunta.

Figura 14

Entrenamiento y Construccion del Default_model ETHUSD

```
default_model = RandomForestRegressor(random_state=42)
default_model.fit(X_train,y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor(random_state=42)
```

Fuente. Elaboración Propia

Podemos adquirir el coeficiente de determinación y el error RMSE utilizando la función print_score().

Se adjunta una captura de pantalla de los hallazgos para poder visualizarlo; sin embargo, los resultados de todos los modelos sugeridos eventualmente se mostrarán en un marco de datos para que pueda comparar todos los modelos de prediccion creados.

Figura 15

Resultados del Default_model ETHUSD

```
print_score(default_model)

RMSE training set: 2.6823846608294173
RMSE validation set: 2.587552811550278
R² training set: 0.9999950606068664
R² validation set: 0.9998810105030357
```

Fuente. Elaboración Propia

modified_trees_leaf_model. Se presenta un modelo con parametros que fueron ajustados manualmente. Se realizarán las siguientes modificaciones a la configuración predeterminada del modelo:

- N_estimators = 60. Contiene sesenta árboles de decisión. Se cree que debería mejorarse porque es menos que el modelo predeterminado (100).
- Min_samples_leaf = 25. La profundidad del árbol disminuye a medida que aumenta el número de muestras necesarias para estar en el último nodo. Los árboles serán más cortos y comunicarán menos información, lo que los hará menos precisos a menor profundidad.

Figura 16

Entrenamiento, Construcción y Resultados del Modelo modified_trees_leaf_model ETHUSD

```
modified_trees_leaf_model = RandomForestRegressor(n_estimators=60, min_samples_leaf=25,  random_state=42
modified_trees_leaf_model.fit(X_train, y_train)
```

```
RMSE training set: 58.410946971110285
RMSE validation set: 32.203808571599055
R² training set: 0.997657821585607
R² validation set: 0.9815691800529122
```

Fuente. Elaboración Propia

En realidad, el uso de estos valores hace que el error en el conjunto de validación sea mucho peor que el default_model. Se descubre que el modelo de predicción no mejora al aumentar el parámetro min_samples_leaf.

modified_depth_model. Sugerimos crear un nuevo modelo utilizando una cantidad similar de árboles que el anterior modelo de prediccion con una profundidad que consideremos apropiada en primer lugar después de concluir que la profundidad es significativa para nuestro modelo. Para este modelo se emplea los siguientes parametros:

- N_estimators= 55. No se utilizan tanto como el modelo anterior.
- Max_depth = 20. Creemos que el error mejorará significativamente al aumentar la profundidad.

De forma predeterminada, se utilizan los argumentos restantes.

Figura 17

Entrenamiento, Construcción y Resultados del modelo Modified_depth_model BTCUSD

```
modified_depth_model = RandomForestRegressor(n_estimators=55, max_depth=20,  random_state=42)
modified_depth_model.fit(X_train, y_train)
```

```
RMSE training set: 2.6932029260847035
RMSE validation set: 2.5987711270671543
R² training set: 0.9999950206846129
R² validation set: 0.9998799765103337
```

Fuente. Elaboración Propia

Al aumentar la profundidad y excluir la opción min_samples_leaf, el error del conjunto de validación disminuye en comparación con el modified_tree_leafs_model y el default_model.

En un esfuerzo por maximizar los resultados, sugerimos aumentar el número de árboles y reducir la profundidad para el siguiente modelo.

modifed_depth_and_trees_model. Teniendo en cuenta lo encontrado en el modelo anterior se emplean los siguientes parametros:

- N_estimators = 85. Aunque implique mayores costes computacionales, se consiguen mejoras.
- Max_depth = 18. Se reduce la profundidad para compensar por la alta cantidad de árboles y se consiguen mejoras.

De forma predeterminada, se utilizan los argumentos restantes.

Figura 18

Entrenamiento, Construccion y resultados del modelo Modifed_depth_and_trees_model

ETHUSD

```
modified_depth_and_trees_model = RandomForestRegressor(n_estimators=85, max_depth=18,  random_state=42)
modified_depth_and_trees_model.fit(X_train, y_train)
```

```
RMSE training set: 2.6591677079273013
RMSE validation set: 2.522848728916314
R² training set: 0.9999951457410922
R² validation set: 0.9998868869780402
```

Fuente. Elaboración Propia

Creemos que estas modificaciones mejoran el modelo porque en realidad resultan en una disminución del error RMSE y una mejora en el coeficiente de determinación.

Explotación y validación de Modelos Manuales de Predicción

Teniendo en cuenta que los parámetros de profundidad y la cantidad de árboles fueron los componentes más importantes y centrales en muchos de los modelos, se recomienda realizar una búsqueda manual de los parámetros óptimos.

Variando el número de árboles. Se ha creado una función de Python que informa los errores en el modelo al cambiar la cantidad de árboles para evaluar la forma en que funciona el modelo de predicción en relación con el parámetro de cantidad de árboles. Este método construye el bucle y produce y entrena los modelos instantáneamente.

Consideraremos todos los parámetros predeterminados, con excepción de la cantidad de árboles, que varía automáticamente de 10 a 10 en cada modelo, con un máximo de 500 árboles.

Se implementa la función reg_acc, la cual muestra los errores MAE, MAPE, RMSE, eficiencia y coeficiente de determinación R2 del conjunto de validación.

Figura 19

Función Reg_acc

```
# Return MAE, MRSE, R²
def reg_acc(y_true, y_pre):
    return_var = []
    from math import sqrt
    rmse = sqrt(mean_squared_error(y_true,y_pre))
    return_var.append(rmse)
    print ("RMSE: ",rmse )
    r2 = r2_score(y_true,y_pre)
    return_var.append(r2)
    print ("R²: ",r2 )
    mae = mean_absolute_error(y_true,y_pre)
    return_var.append(mae)
    print ('MAE: ',mae)

    if 0 in y_true :
        print("MAPE can't be calculated")
        return_var.append(0)
    else :
        mape = round(np.mean(np.abs((y_true - y_pre)/y_true))*100,4)
        print ('MAPE :', mape)
        print('============================')
        print('Model Accuracy(%) :', 100-mape)
        print('============================')
        return_var.append(mape)
        return_var.append(100-mape)
    return return_var
```

Fuente. Elaboración Propia

Figura 20

Creación del modelo de bucle

```
#Looking for the best nº estimators. Parameters by default

random_tree = [i*10 for i in range(1,50)]
rmse , r_sq , mae , mape = [],[],[],[]
for tree_size in random_tree:
    print('Tree Size:', tree_size)
    model = RandomForestRegressor(random_state=42, n_estimators=int(tree_size)).fit(X_train,y_train)
    model.fit(X_train, y_train)
    estimation = model.predict(X_val)
    result = reg_acc(y_val,estimation)
    rmse.append(result[0])
    r_sq.append(result[1])
    mae.append(result[2])
    mape.append(result[3])
```

Fuente. Elaboración Propia

Encontramos que, a partir de los 70 árboles, se empieza a estabilizar el modelo en términos de coeficiente de determinación y en errores bajo estos parámetros establecidos.

De todos los árboles examinados, se encontró que el modelo que tiene 90 árboles tiene un R2 más alto y un RMSE más bajo. Como se muestra abajo:

Figura 21

El resultado de la cantidad de árboles más eficientes

```
Tree Size: 90
RMSE:  2.5397144234370335
R²:  0.9998853695613124
MAE:  1.7332019560955803
MAPE : 0.0867
============================
Model Accuracy(%) : 99.9133
============================
```

Fuente. Elaboración Propia

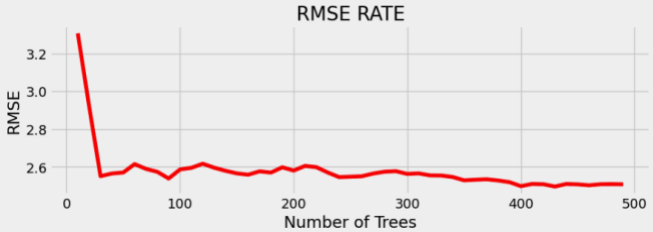
Con lo anterior se encontró que la cantidad de árboles utilizados es particularmente significativa para realizar las predicciones. Aunque se sabe que la precisión de un modelo predictivo aumenta con la cantidad de árboles utilizados, cada modelo tiene un umbral por encima del cual se estabiliza su resultado, por lo que no es necesario agregar más árboles (como

lo ilustran las imágenes adjuntas). El aumento de precisión de predicción se empieza a volver insignificante y es necesario tener en cuenta los gastos computacionales asociados. Con respecto al error RMSE:

Figura 22

Función número de árboles - RMSE

```
plt.figure(figsize=(10,3))
plt.title('RMSE RATE')
plt.xlabel('Number of Trees')
plt.ylabel('RMSE')
sns.lineplot(x=random_tree,y=rmse, color='red')
```



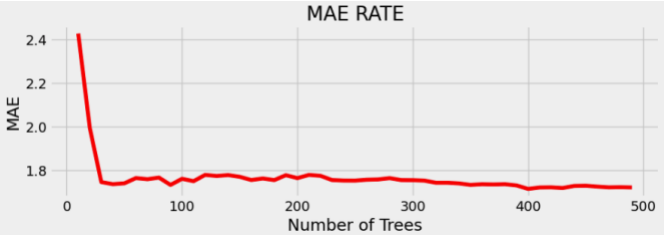
Fuente. Elaboración Propia

Con respecto al error MAE:

Figura 23

Function No. trees - MAE.

```
plt.figure(figsize=(10,3))
plt.title('MAE RATE')
plt.xlabel('Number of Trees')
plt.ylabel('MAE')
sns.lineplot(x=random_tree,y=mae, color='red')
```



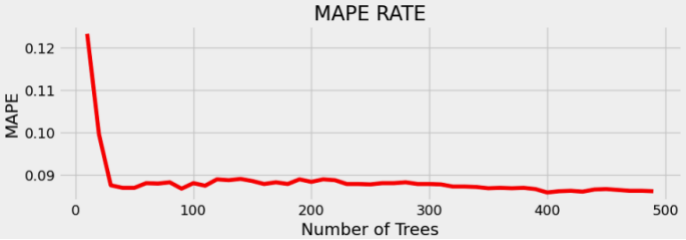
Fuente. Elaboración Propia

Con respecto al error MAPE:

Figura 24

Function No. trees - MAPE

```
plt.figure(figsize=(10,3))
plt.title('MAPE RATE')
plt.xlabel('Number of Trees')
plt.ylabel('MAPE')
sns.lineplot(x=random_tree,y=mape, color='red')
```



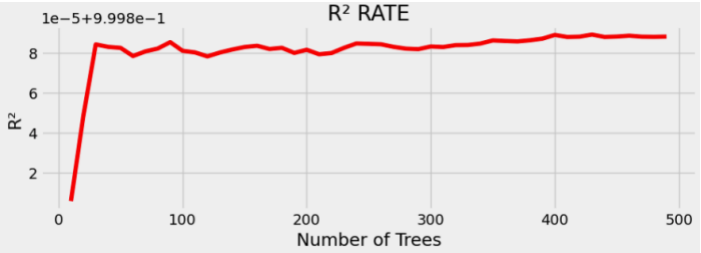
Fuente. Elaboración Propia.

Con respecto al error R2:

Figura 25

Function No. trees – R2

```
plt.figure(figsize=(10,3))
plt.title('R² RATE')
plt.xlabel('Number of Trees')
plt.ylabel('R²')
sns.lineplot(x=random_tree,y=r_sq, color='red')
```



Fuente. Elaboración Propia

Hay cambios notables entre 70 y 180 árboles, lo que indica que tendríamos un mejor resultado si se utiliza una cantidad de árboles entre ese rango. Podemos ver en los gráficos anteriores que el coeficiente de determinación máximo y los errores RMSE, MAE y MAPE más bajos se lograrán con 90 árboles.

Es importante señalar que el tiempo de cálculo está aumentando y los resultados son notablemente comparables después de 70 árboles. Por lo tanto, para el modelo de predicción ETHUSD 1d y BTCUSD 1d, se decide seleccionar los 90 árboles para el modelo modificado final("modified_final"), ya que es la cantidad de árboles más precisa entre el rango de 70 y 180 árboles.

Variar la profundidad de los árboles. Se ha desarrollado una metodología muy similar al estudio realizado con la cantidad de árboles para evaluar el rendimiento del modelo de predicción respecto a su parámetro de mayor profundidad.

Cada iteración ha implicado la creación y entrenamiento de modelos. Los resultados de los 90 árboles que se recolectaron previamente se contarán en un esfuerzo por optimizar la profundidad y la cantidad de árboles en términos de precisión.

Más adelante intentaremos emplear una variedad de técnicas para realizar una búsqueda más profunda con más parámetros que podrían ser cruciales para el modelo.

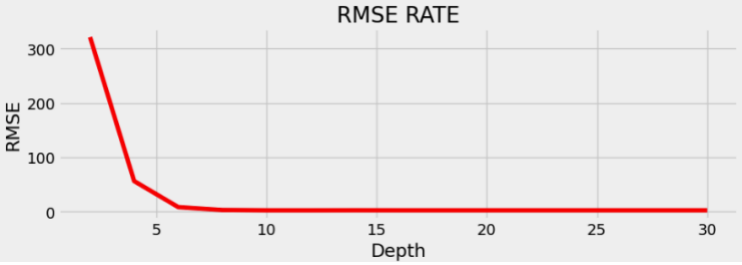
Como se indicó anteriormente, todos los parámetros que se dejan vacíos tienen asignado su valor predeterminado. Para ello, se construye un bucle para examinar cómo varían los resultados en función de la mayor profundidad permitida.

Para el conjunto de validación, emplearemos la misma función reg_acc que produce los errores RMSE, MAE, MAPE y coeficiente de determinación R2.

Anticipamos resultados de menor calidad con menos profundidad de antemano. Después de la llamada a la función, observamos lo siguiente:

Figura 26

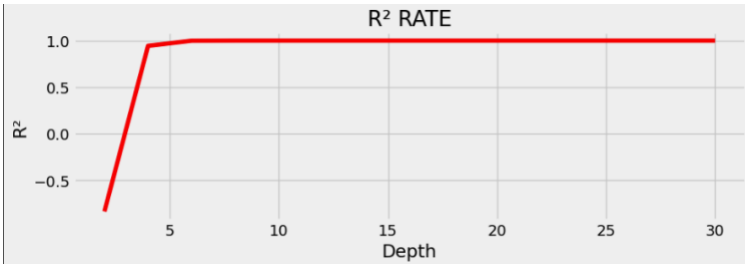
RMSE de la Función de profundidad



Fuente. Elaboración Propia

Figura 27

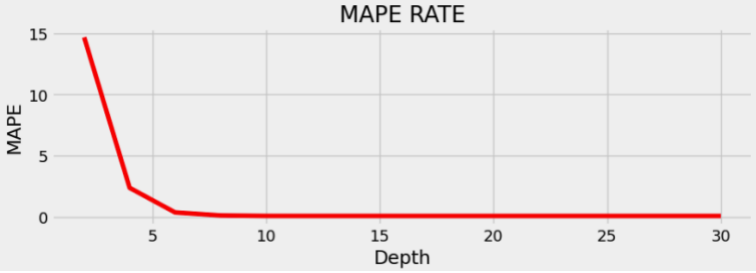
R2 de la Función de profundidad



Fuente. Elaboración Propia

Figura 28

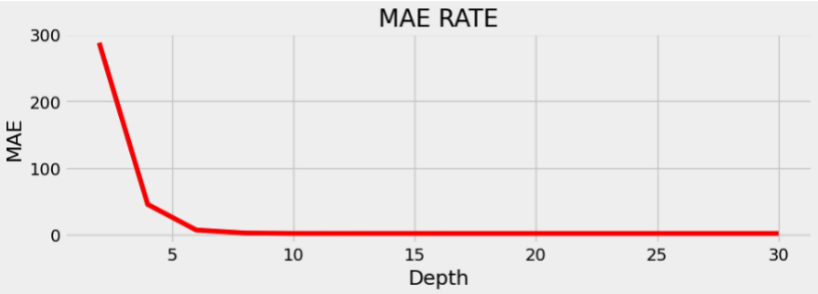
MAPE de la Función de profundidad



Fuente. Elaboración Propia

Figura 29

MAE de la Función de profundidad



Fuente. Elaboración Propia

Entre todas las opciones examinadas, la de profundidad 12 tenía el RMSE más bajo y el R2 más alto. Como se muestra en la siguiente figura:

Figura 30

Resultado de la Profundidad más Eficiente

```
Depth Size: 16
RMSE:  2.5397144234370335
R²:  0.9998853695613124
MAE:  1.7332019560955803
MAPE : 0.0867
============================
Model Accuracy(%) : 99.9133
============================
```

Fuente. Elaboración Propia

La eficiencia del modelo aumenta marginalmente por lo mencionado anteriormente.

Cabe señalar que el modelo que se utiliza es un ejemplo y se creó para el par Ethereum – Dólar estadounidense, o ETH/USDT. Además, los parámetros pueden cambiar.

Cabe mencionar que, a partir de la profundidad de 12, los resultados se empiezan a estabilizar y el tiempo de cálculo es cada vez mayor después de pasar esa profundidad. Por lo

tanto, se decide elegir la profundidad de 16, ya que es la profundidad con más precisión para el modelo de predicción de ETHUSD 1d.

Se selecciona un modelo con 90 árboles y una profundidad de 16 como modelo de búsqueda manual final ("modified_final") para ETHUSD 1d en un esfuerzo para aumentar el coeficiente de determinación y reducir la tasa de errores. Esta decisión se tomó a la luz de los hallazgos y conclusiones extraídas en el punto anterior.

Construcción e implementación de los modelos de búsqueda aleatoria y Grid Search

Teniendo en cuenta los parámetros utilizados con los modelos manuales de predicción, se utilizará la búsqueda aleatoria y Grid_Search para intentar una búsqueda autónoma de hiperparámetros.

Ajustar Hiperparámetros. Aunque Scikit-Learn implementa una serie de parámetros hiperpredeterminados, esto no significa que sean perfectos. Cuando se trata de ajustar un modelo, el aprendizaje automático se parece más a la ingeniería de prueba y error porque es difícil pronosticar qué hiperparámetros funcionarán mejor.

Los hiperparámetros se ajustan a lo largo de iteraciones mediante validación cruzada K-Fold y parámetros de modelo adicionales.

Una vez contrastados, se elige el más exitoso para entrenarlo utilizando el conjunto de entrenamiento y evaluarlo utilizando el conjunto de prueba. Afortunadamente, Scikit-Learn permite realizar el ajuste del modelo K-Fold CV automáticamente, lo cual es más rápido.

Podemos definir una colección de hiperparámetros y ejecutar la validación cruzada K-Fold en cada conjunto de valores utilizando la función RandomizedSearchCV de Scikit-Learn.

Buscar con Búsqueda Aleatoria. Para la busqueda aleatoria, definimos un conjunto de parámetros. Con cada iteración, el algoritmo elegirá una nueva combinación.

Ahora tenemos más control sobre más aspectos que antes. Anteriormente, se pensaba que muchos estaban en predeterminado.

Figura 31

Parámetros para la búsqueda aleatoria

```
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 10, stop = 500)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt', None]
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 50, num = 45)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 4, 6, 8, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4, 6, 8, 10, 12, 14, 16, 20, 25, 30]
# Method of selecting samples for training each tree
bootstrap = [True]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
print(random_grid)
```

Fuente. Elaboración Propia

Sin embargo, el beneficio de la búsqueda aleatoria es que elegimos números al azar de un amplio rango en lugar de probar todas las opciones.

A continuación, se desarrolla un modelo sobre el cual se utilizará y entrenará el método sugerido.

Figura 32

Construcción del Algoritmo de Búsqueda Aleatoria

```
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestRegressor(random_state = 42)
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available cores
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid, n_iter = 100, scoring='neg_mean_squared_error',
                               cv = 3, verbose=2, random_state=42, n_jobs=-1,
                               return_train_score=False)
# Fit the random search model
rf_random.fit(X_train, y_train)
```

Fuente. Elaboración Propia.

Cabe resaltar la importancia de los parámetros n_iter, que indica el número de iteraciones a realizar, y cv, que indica el número de divisiones para la validación cruzada. Si bien la búsqueda de alternativas crece con el número de iteraciones, la probabilidad de sobre ajuste disminuye al aumentar el CV, aunque a costa de tiempos de cálculo significativamente más largos.

Además, el parámetro de puntuación es importante, ya que, en este ejemplo, queremos alcanzar el RMSE más bajo. Además, el parámetro predeterminado return_train_score = False se conserva porque estamos interesados en la evaluación de errores en el conjunto de validación. De lo contrario, habría un aumento significativo en el gasto computacional. Los mejores parámetros obtenidos con este algoritmo se presentan a continuación.

Figure 33

Mejores Parámetros Obtenidos por el Algoritmo de Búsqueda Aleatoria

```
{'n_estimators': 60,
 'min_samples_split': 4,
 'min_samples_leaf': 2,
 'max_features': 'sqrt',
 'max_depth': 20,
 'bootstrap': True}
```

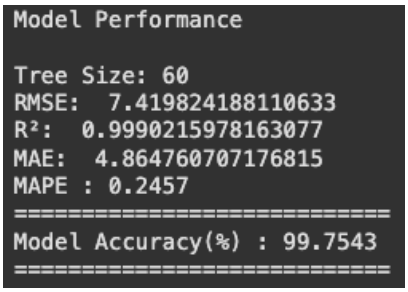
Fuente. Elaboración Propia.

Se garantizará que se pueda acceder a una alternativa utilizando bootstrap = True.

Actualmente, el modelo se prueba utilizando los parámetros ideales tras la aplicación de la técnica de búsqueda aleatoria. Best_random es el nombre de este nuevo modelo, que presenta los mejores parámetros actualmente en uso.

Figura 34

Comportamiento del Modelo con Mejores Parámetros luego de Aplicar la Búsqueda Aleatoria



Fuente. Elaboración Propia

Validación Cruzada con Grid_Search. Podemos disminuir el rango de cada hiperparámetro mediante la búsqueda aleatoria. Ahora que sabemos dónde concentrar nuestros parámetros, podemos experimentar con otras configuraciones. Para hacer esto se utiliza GridSearchCV, un método que evalúa cada combinación que especificamos. Se reevaluarán diferentes parámetros para emplear GridSearchCV. El cálculo lleva mucho más tiempo.

El primer modelo creado con el metodo GridSearchCV se llamará “first_grid”.

Figura 35

Construcción del modelo First_Grid

```
from sklearn.model_selection import GridSearchCV
# Create the parameter grid based on the results of random search
param_grid = {
    'bootstrap': [True],
    'max_depth': [10,12,14,16],
    'min_samples_leaf': [1,3,5],
    'min_samples_split': [2,4,6],
    'n_estimators': [65,75,85]
}

# Create a base model
rf = RandomForestRegressor(random_state = 42)

# Instantiate the grid search model
grid_search = GridSearchCV(estimator = rf, param_grid = param_grid,
                           cv = 3, n_jobs = -1, verbose = 2, return_train_score=False, scoring='neg_mean_
```

Fuente. Elaboración Propia

Después de construir y ejecutar el modelo anterior, podemos ver los parámetros óptimos del modelo a continuación.

Figura 36

Mejores Parámetros Después de la Búsqueda de first_grid

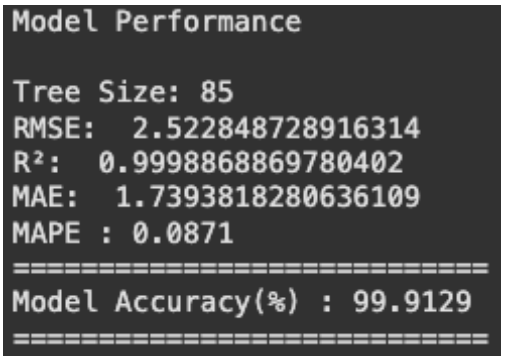
```
{'bootstrap': True,
 'max_depth': 16,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'n_estimators': 85}
```

Fuente. Elaboración Propia

Utilizando estos parámetros se evalúa el modelo en términos de precisión y en tasa de errores a continuación.

Figura 37

Comportamiento del Modelo first_grid



Fuente. Elaboración Propia

Esta búsqueda más exhaustiva arrojó mejores resultados que el modelo de búsqueda aleatoria. Por lo tanto, se llevará a cabo una segunda búsqueda de Grid con diferentes parámetros para ver si hay una mejora significativa. Nos referimos a este modelo como second_grid.

Figura 38

Construcción del modelo Second_Grid

```
param_grid = {
    'bootstrap': [True],
    'max_depth': [11,13,15,16],
    'min_samples_leaf': [1,3,5],
    'min_samples_split': [2,4,6],
    'n_estimators': [60,70,80]
}

# Create a base model
rf = RandomForestRegressor(random_state = 42)

# Instantiate the grid search model
grid_search_final = GridSearchCV(estimator = rf, param_grid = param_grid,
                                 cv = 3, n_jobs = -1, verbose = 2,scoring='neg_mean_squared_error', return_train_score=False)

grid_search_final.fit(X_train, y_train);
```

Fuente. Elaboración Propia

Figura 39

Mejores parámetros después de la búsqueda de second_grid

```
{'bootstrap': True,
 'max_depth': 15,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'n_estimators': 60}
```

Fuente. Elaboración Propia

Una vez más, se realiza una evaluación del modelo sugerido tras la selección de los parámetros óptimos.

Figura 40

Comportamiento del modelo second_grid

```
Tree Size: 60
RMSE:  2.615625087364154
R²:  0.9998784146716402
MAE:  1.7644423618055025
MAPE : 0.088
============================
Model Accuracy(%) : 99.912
============================
```

Fuente. Elaboración Propia

Como se evidencia anteriormente, el ajuste de hiperparámetros en la segunda búsqueda de grid bajó el nivel de precisión y aumentó la tasa de errores para el modelo de predicción de second_grid. Por lo tanto, second_grid será el último modelo de precisión que se construirá para este proyecto aplicado.

Desarrollo de la Etapa de Operacion

De los ocho modelos de predicción que se desarrollaron, en esta etapa se selecciona el mejor modelo. Para tomar esta decisión, primero se muestra una tabla que contiene todos los

resultados combinados de los modelos. El método evaluate_model() se desarrolló para mostrar los resultados.

Luego de la selección del modelo óptimo, los precios se proyectarán a partir del conjunto del 20% que se seleccionó y se contrastarán con los precios de cierre reales del conjunto de validación. El objetivo de este paso es presentar un gráfico que compare los precios de cierre reales y previstos del conjunto de validación. Este gráfico muestra los costos proyectados en naranja y los precios reales en azul.

Después, se muestra una tabla con las primeras cinco y últimas cinco filas del precio de cierre actual y previsto. La idea de esta tabla es mostrar una visualización más detallada de qué tan cerca están los precios reales y previstos.

Por último, se muestra la diferencia promedio entre los precios de cierre reales y previstos.

Resultados

Resultados en la Etapa de Concepción

Se acordó ejecutar un procedimiento de predicción para las criptomonedas Ethereum y Bitcoin en la primera fase. Se utiliza un conjunto de validación para pronosticar los valores de bitcoin, que posteriormente se comparan con los precios reales durante un breve período de tiempo.

Para esta etapa se eligieron las dos criptomonedas más populares (Bitcoin y Ethereum). Estas criptomonedas son populares, no son muy nuevas y ofrecen un desafío de predicción, ya que sus precios fluctúan constantemente.

Resultados en la Etapa de Diseño

Para la etapa de diseño, se diseñaron cinco modelos de búsqueda manual para buscar manualmente un rango de parámetros óptimos que podemos usar para los modelos de búsqueda automática. Luego se diseñaron tres modelos de búsqueda automática para proporcionar los mejores parámetros y resultados posibles. Todos los modelos de predicción fueron diseñados basándose en el algoritmo de bosque aleatorio.

Resultados en la Etapa de Implementación

En esta tercera etapa, construimos e implementamos los modelos de búsqueda manual y búsqueda automática para ETH/USD y BTC/USD.

Los resultados de cada modelo implementado para el par de ETH/USD son:

- default_model: Se utilizaron los parámetros por defecto. Se obtuvo una precisión del 99.9120%.
- modified_trees_leaf_model: Se encuentra que el nodo final debe incluir por lo menos 25 muestras y se utiliza una menor cantidad de árboles(60). Esto indica que para volver a dividir el nodo del árbol se necesitan 25 observaciones. El aumento de este número hace que la profundidad del árbol disminuya, lo cual hace que esta cantidad de muestras sea perjudicial en el modelo sugerido (aumentan los errores). Los nodos crecen más profundamente hasta que las hojas son puras o contienen menos información. Se obtuvo una precisión del 98.661%, lo cual es menor que el default_model.
- modified_depth_model: Como se indicó anteriormente, se cree que profundizar puede aumentar la precisión del modelo de predicción; por lo tanto, se propuso un modelo con una cantidad similar de árboles(55) pero con una profundidad mayor (20). Las

métricas de evaluación han mejorado significativamente respecto al modelo anterior y con respecto al primer modelo propuesto. En consecuencia, se deduce que, hasta cierto punto, un árbol más profundo normalmente produce mejores resultados. Se obtuvo una precisión de 99.9125%, lo cual es mayor que el modified_trees_leaf_model y que el default_model.

- modified_depth_and_trees_model: El recuento de árboles del bosque aumenta (85) y se baja la profundidad (18) para compensar por el aumento de árboles. Como era de esperar, agregar más árboles ayuda a mejorar los resultados, pero es importante recordar que después de una cierta cantidad de árboles, los resultados comienzan a estabilizarse. Debido a esto, es mejor no emplear más de lo necesario para evitar costos computacionales innecesarios. Se obtuvo una precisión del 99.9129%, lo cual es mayor que todos los modelos anteriores.
- modified_final: Este modelo de predicción es extremadamente significativo debido a la minuciosa búsqueda que se realizó. El proceso de emplear bucles para encontrar los valores de árbol y profundidad a partir de los cuales se alcanza un nivel particular de estabilidad. Al utilizar 90 árboles y una profundidad de 16, se encontró que, la tasa de errores disminuye y el coeficiente de determinación del modelo mejora. Este modelo obtuvo una precisión del 99.9133%, lo cual es superior a todos los modelos manuales creados anteriormente.
- best_random: Se logran resultados más bajos cuando se realiza una búsqueda de validación cruzada en la configuración de un conjunto de parámetros. Este modelo obtuvo una precisión de 99.7543% lo cual es menor a todos los modelos manuales creados anteriormente.

- first_grid: Se utiliza un nuevo conjunto de parámetros y la búsqueda se ejecuta una vez más, intentando todas las combinaciones posibles. La búsqueda muestra el par de 85 árboles y una profundidad de 16, como los mejores parámetros del conjunto. Con esto se evidencia que no se realizó una búsqueda suficiente para encontrar los verdaderos mejores parámetros. Los resultados son muy similares: no son tan buenos como el modelo final modificado, pero siguen siendo mejores o iguales que los demás modelos. Actualmente, se cree que podemos estar muy cerca de la configuración óptima de hiperparámetros. Este modelo obtuvo una precisión de 99.9129%, lo cual es menor que el modified_final pero mejor o igual que el resto de modelos.
- second_grid: Una vez más se realiza una búsqueda, analizando todas las posibilidades con una serie distinta de parámetros. Las medidas de evaluación no muestran una mejora comparada con el modelo first_grid después de cambiar el conjunto y el rango de los parámetros. La búsqueda muestra el par de 60 árboles y una profundidad de 15, como los mejores parámetros del conjunto. Con esto se evidencia que no se realizó una búsqueda suficiente para encontrar los verdaderos mejores parámetros. Este modelo obtuvo una precisión de 99.9120%, lo cual es menor que el modelo first_grid y el modified_final.

Los resultados de cada modelo implementado para el par de BTC/USD son:

- default_model: Se utilizaron los parametros por defecto. Se obtuvo una precision de 99.9133%.
- modified_trees_leaf_model: Se encuentra que el nodo final debe incluir por lo menos 25 muestras. Esto indica que para volver a dividir el nodo del árbol se necesitan 25 observaciones. El aumento de este número hace que la profundidad del árbol

disminuya, lo cual hace que esta cantidad de muestras sea perjudicial en el modelo sugerido (aumentan los errores). Los nodos crecen más profundamente hasta que las hojas son puras o contienen menos información. Se obtuvo una precisión de 98.9871% lo cual es menor que el default_model.

- modified_depth_model: Como se indicó anteriormente, se cree que profundizar puede aumentar la precisión del modelo de predicción; por lo tanto, se propuso un modelo con un número similar de árboles pero una profundidad mayor (20). Las métricas de evaluación han mejorado significativamente respecto al modelo anterior. En consecuencia, se deduce que, hasta cierto punto, un árbol más profundo normalmente produce mejores resultados. Se obtuvo una precisión del 99.9023%, lo cual es mayor que el modified_trees_leaf_model, pero menor que default_model.
- modified_depth_and_trees_model: El recuento de árboles del bosque aumenta (25) y se baja la profundidad (15) para compensar por el aumento de árboles. Se encontró que se utilizaron suficientes árboles para ver una mejora con respecto a los modelos anteriores. Se obtuvo una precisión de 99.9076%, lo cual es mejor que el modified_depth_model y modified_trees_leaf_model, pero menor que el default_model.
- modified_final: Es considerado extremadamente significativo debido a la minuciosa búsqueda que se realizó. El proceso de emplear bucles para encontrar los valores de árbol y profundidad a partir de los cuales se alcanza un nivel particular de estabilidad. Al utilizar 50 árboles y una profundidad de 14, se encontró que la tasa de errores disminuye y el coeficiente de determinación del modelo mejora. Este modelo obtuvo

una precisión de 99.9162% lo cual es superior a todos los modelos manuales creados anteriormente.

- best_random: Se logran resultados más bajos cuando se realiza una búsqueda de validación cruzada en la configuración de un conjunto de parámetros. Este modelo obtuvo una precisión del 99.7907%, lo cual es menor a casi todos los modelos manuales creados anteriormente.
- first_grid: Se utiliza un nuevo conjunto de parámetros y la búsqueda se ejecuta una vez más, intentando todas las combinaciones posibles. La búsqueda muestra el par de 25 árboles y una profundidad de 15, como los mejores parámetros del conjunto. Con esto se evidencia que no se realizó una búsqueda suficiente para encontrar los verdaderos mejores parámetros. Los resultados son menores: no son tan buenos como el modified_final. Actualmente, se cree que podemos estar muy cerca de la configuración óptima de hiperparámetros. Este modelo obtuvo una precisión de 99.9014%, lo cual es menor que el modified_final.
- second_grid: Una vez más se realiza una búsqueda, analizando todas las posibilidades con una serie distinta de parámetros. La búsqueda muestra un par de 30 árboles y una profundidad de 8, como los mejores parámetros del conjunto. Con esto se evidencia que no se realizó una búsqueda suficiente para encontrar los verdaderos mejores parámetros. Las medidas de evaluación muestran que este modelo no mejora comparada con el modelo first_grid después de cambiar el conjunto y el rango de los parámetros. Este modelo obtuvo una precisión de 99.8885%, lo cual es mejor que el modelo best_random pero menor que el resto de modelos.

Resultados en la Etapa de Operacion

Para cada criptomoneda (Bitcoin y Ethereum) investigada se han propuesto ocho modelos, con los ajustes necesarios. En esta etapa se explicará el modelo de predicción seleccionado para la predicción de ETHUSD 1d y BTCUSD 1d junto a sus resultados.

Para la cuarta etapa, en la vela de 1 día de Ethereum y la vela de 1 día de Bitcoin, las Tablas 1 y 2 muestran que el mejor modelo de predicción es modified_final. Este modelo de predicción es el modelo seleccionado para la predicción en ambas criptomonedas y este modelo también tiene mejores estadísticas de error que el modelo de predicción presentado en (Livieris et al., 2021).

Tabla 2

Resultados de los Modelos Propuestos ETHUSD 1d

	model	n_features	n_trees	max_depth	min_samples_leaf	min_samples_split	RMSE	R²	MAE	MAPE	accuracy	time
0	default_model	4	100	NaN	1	2	2.587553	0.999881	1.761195	0.0880	99.9120	0.442469
1	modified_trees_leaf_model	4	60	NaN	25	2	32.203809	0.981569	26.298152	1.3386	98.6614	0.146052
2	modified_depth_model	4	55	20.0	1	2	2.598771	0.999880	1.753074	0.0875	99.9125	0.273849
3	modified_depth_and_trees_model	4	85	18.0	1	2	2.522849	0.999887	1.739382	0.0871	99.9129	0.372242
4	modified_final	4	90	16.0	1	2	2.539714	0.999885	1.733202	0.0867	99.9133	0.440722
5	best_random	4	60	20.0	2	4	7.419824	0.999022	4.864761	0.2457	99.7543	0.233937
6	first_grid	4	85	16.0	1	2	2.522849	0.999887	1.739382	0.0871	99.9129	0.364484
7	second_grid	4	60	15.0	1	2	2.615625	0.999878	1.764442	0.0880	99.9120	0.262479

Fuente. Elaboración Propia

Tabla 3

Resultados de los modelos propuestos BTCUSD 1d

	model	n_features	n_trees	max_depth	min_samples_leaf	min_samples_split	RMSE	R²	MAE	MAPE	accuracy	time
0	default_model	4	100	NaN	1	2	41.755089	0.999961	31.199591	0.0867	99.9133	0.697945
1	modified_trees_leaf_model	4	20	NaN	25	2	436.852501	0.995731	358.160633	1.0129	98.9871	0.105745
2	modified_depth_model	4	15	20.0	1	2	48.537792	0.999947	34.746020	0.0977	99.9023	0.094240
3	modified_depth_and_trees_model	4	25	15.0	1	2	45.855860	0.999953	32.888149	0.0924	99.9076	0.158379
4	modified_final	4	50	14.0	1	2	41.005286	0.999962	30.116346	0.0838	99.9162	0.305637
5	best_random	4	240	35.0	2	4	100.418603	0.999774	77.378851	0.2093	99.7907	0.976083
6	first_grid	4	25	15.0	2	6	45.409310	0.999954	34.819337	0.0986	99.9014	0.124853
7	second_grid	4	30	8.0	1	2	49.039495	0.999946	39.523144	0.1115	99.8885	0.148959

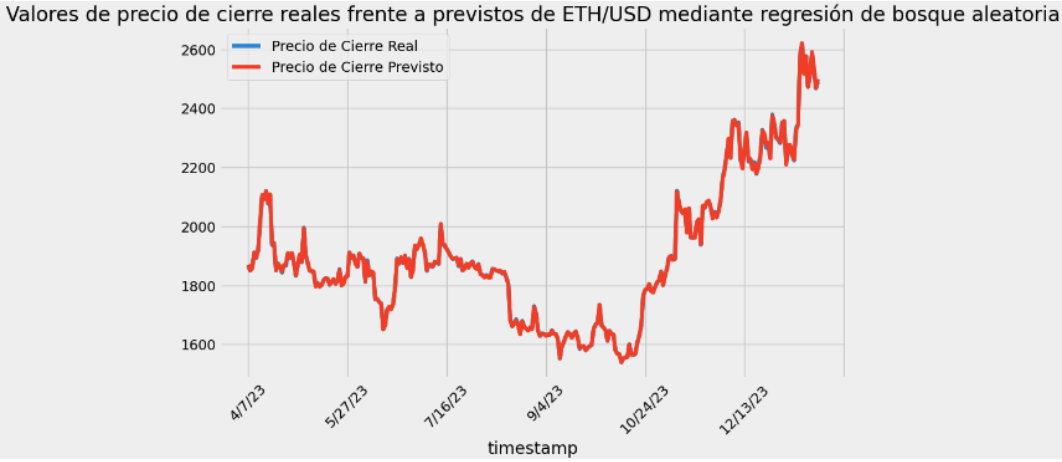
Fuente. Elaboración Propia

Con las tablas anteriores también se evidencia que el valor más alto de R2 lo alcanza el modelo modified_final con 0,999885 para Ethereum y 0.999962 para Bitcoin. El modelo de predicción modified_final tiene también los errores RMSE, MAE y MAPE más bajos para Ethereum y Bitcoin.

Luego, utilizando los modelos modified_final, en la figura 41 y 42 se muestra que los valores reales se aproximan y que no existen valores atípicos significativos.

Figura 41

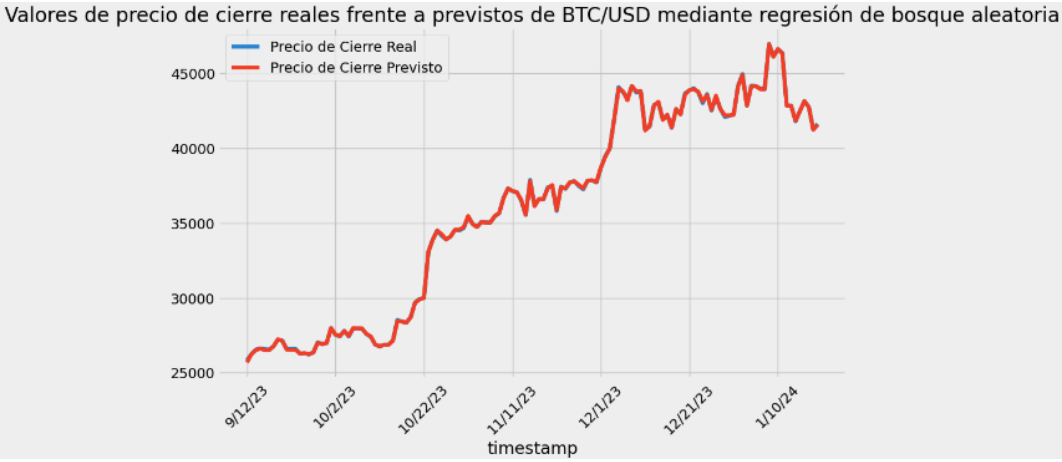
Gráfico de los precios de cierre reales y previstos de ETHUSD 1d



Fuente. Elaboración Propia

Figura 42

Gráfico de los precios de cierre reales y previstos de BTCUSD 1d



Fuente. Elaboración Propia

Con las figuras 43 y 44 podemos observar tablas con información más detallada en las que vemos la primera y las últimas cinco filas de los precios de cierre reales y previstos. Con estas cifras, detectamos que los precios reales y previstos se aproximan.

Figura 43

Precios de cierre reales y previstos de ETHUSD 1d

timestamp	Close	Precio de Cierre Real	Precio de Cierre Previsto
4/7/23	1865.636108	1865.636108	1870.311035
4/8/23	1849.498169	1849.498169	1849.909271
4/9/23	1859.387817	1859.387817	1857.143255
4/10/23	1911.207520	1911.207520	1911.038972
4/11/23	1892.189697	1892.189697	1894.485734
...	...	...	...
1/15/24	2511.363770	2511.363770	2510.672181
1/16/24	2587.691162	2587.691162	2590.377485
1/17/24	2528.369385	2528.369385	2539.390098
1/18/24	2467.018799	2467.018799	2468.479667
1/19/24	2489.498535	2489.498535	2498.842703

Fuente. Elaboración Propia

Figura 44

Precios de cierre reales y previstos de BTCUSD 1d

timestamp	close	Precio de Cierre Real	Precio de Cierre Previsto
9/12/23	25833.34375	25833.34375	25680.211249
9/13/23	26228.32422	26228.32422	26231.035428
9/14/23	26539.67383	26539.67383	26501.635742
9/15/23	26608.69336	26608.69336	26612.797228
9/16/23	26568.28125	26568.28125	26502.792149
...	...	...	...
1/15/24	42511.96875	42511.96875	42483.581561
1/16/24	43154.94531	43154.94531	43152.271565
1/17/24	42742.65234	42742.65234	42746.036406
1/18/24	41262.05859	41262.05859	41228.179532
1/19/24	41618.40625	41618.40625	41574.780233

Fuente. Elaboración Propia

Lo anterior muestra que el modelo de predicción modified_final no tiene valores atípicos considerables, lo que lo convierte en un modelo de predicción muy consistente. La diferencia promedio entre los precios de cierre reales y previstos se muestra en las siguientes figuras:

Figura 45

Promedio de los precios de cierre reales y previstos de ETHUSD 1d

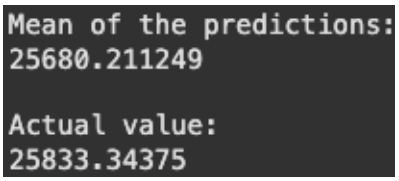
```
Mean of the predictions:
1870.3110345892858

Actual value:
1865.636108
```

Fuente. Elaboración Propia

Figura 46

Promedio de los precios de cierre reales y previstos de BTCUSD 1d



Fuente. Elaboración Propia

Teniendo en cuenta las figuras anteriores, podemos observar que con la vela de 1 día de Ethereum, el precio previsto muestra una diferencia promedio de 4.68 $ en comparación con el valor real, y la vela de 1 día de Bitcoin muestra una diferencia promedio de 153.13 $ en comparación con el valor real.

Ahora, comparando nuestro modelo de predicción con el modelo de predicción encontrado en (Livieris et al., 2021), podemos observar que al predecir Ethereum y Bitcoin, nuestro modelo presenta un RMSE, MAE y MAPE más bajo, y un error R2 más alto.

Conclusiones

Conclusiones de la Etapa de Concepción

En el presente trabajo se analizó el mercado de las criptomonedas con el fin de utilizar un modelo basado en la técnica de bosque aleatorio para anticipar el precio de cierre. En esta etapa se realizó un análisis basado en la popularidad del mercado y la dificultad de predicción. Se eligieron para este estudio las criptomonedas Ethereum y Bitcoin.

Conclusiones de la Etapa de Diseño

Para la etapa de diseño, se diseñaron cinco modelos de búsqueda manual para buscar manualmente un rango de parámetros óptimos que podemos usar para los modelos de búsqueda automática. Luego se diseñaron tres modelos de búsqueda automática para proporcionar los mejores parámetros y resultados posibles. Todos los modelos de predicción fueron diseñados basándose en el algoritmo de bosque aleatorio.

Conclusiones de la Etapa de Implementación

En la tercera etapa las conclusiones para los modelos de búsqueda manual fueron que la acción de aumentar el parámetro min_samples_leaf perjudica el modelo de predicción, ya que tener menos profundidad no beneficia al modelo, y tener un mayor número de árboles y una mayor profundidad ayuda a la precisión del modelo. Con estas conclusiones se logró ajustar el modelo modified_final para que diera resultados de alta precisión, buscando con parámetros que cumplieran los requisitos de estas conclusiones. Se esperaba lograr resultados aún mejores con los modelos de búsqueda automática, pero se evidenció que estos modelos automáticos no estaban realizando una búsqueda completa para encontrar los verdaderos parámetros más óptimos por el tiempo limitado de búsqueda debido a los altos tiempos computacionales.

Conclusiones de la Etapa de Operacion

Para la cuarta etapa, tanto en Ethereum como en Bitcoin, el modelo seleccionado fue el modified_final para ETH/USD y BTC/USD, porque presentó el valor más alto de R2 y el valor más bajo para los errores RMSE, MAE y MAPE. Tener un error MAE y RMSE más bajo implica una mayor precisión de un modelo de regresión. Además, se considera deseable un valor más alto de R2 (Chugh, 2020).

El lado positivo es que al realizar la prediccion final con el modelo modified_final, no se encontraron valores atípicos considerables, lo que lo convierte en un modelo de prediccion consistente para ETH/USD y BTC/USD.

Comparaciones Finales con el Modelo de Livieris et al., 2021

Al comparar el modelo seleccionado basado en *Random Forest* (modified_final) con el modelo de Livieris et al. (2021), se confirma que el modelo de predicción *modified_final* es más eficaz para predecir tanto Ethereum como Bitcoin por diversas razones.

En primer lugar, en cuanto a la arquitectura de los modelos**,** el modelo CNN-LSTM de Livieris et al. (2021) combina redes neuronales convolucionales (CNN) para la extracción de características con redes de memoria a largo y corto plazo (LSTM) para la predicción de secuencias. Esta arquitectura es particularmente beneficiosa cuando se dispone de grandes cantidades de datos. En contraste, el modelo seleccionado basado en *Random Forest* (modified_final) emplea un método de aprendizaje por conjuntos que crea varios árboles de decisión y agrega sus predicciones. Este modelo resultó ser generalmente más fácil de interpretar y requirió menos ajustes en comparación con los modelos de aprendizaje profundo, como el CNN-LSTM de Livieris et al. (2021).

En cuanto a las métricas de rendimiento**,** se utilizaron índices como RMSE, MAE, MAPE y R^2 para evaluar la precisión de los modelos de predicción. El modelo *modified_final,* basado en *Random Forest*, presentó valores más bajos de RMSE, MAE y MAPE, así como un R^2 más alto en comparación con el modelo CNN-LSTM (Livieris et al., 2021), lo que indica un mejor rendimiento en la predicción de los precios de las criptomonedas.

En relación con los requisitos de datos**,** el modelo CNN-LSTM de Livieris et al. (2021) requiere grandes cantidades de datos para entrenar de manera efectiva y evitar el sobreajuste, además de un preprocesamiento extenso para optimizar los formatos de entrada. Por otro lado, el modelo basado en *Random Forest* (modified_final) funcionó bien con conjuntos de datos más pequeños y es menos sensible al sobreajuste, lo que permitió utilizar un conjunto de datos más compacto y específico.

Finalmente, en términos de interpretabilidad**,** el modelo CNN-LSTM de Livieris et al. (2021) se considera una "caja negra", lo que dificulta la interpretación de la influencia de características específicas en las predicciones. En cambio, el modelo basado en *Random Forest* (modified_final) proporcionó puntuaciones de importancia de características, lo que facilitó la interpretación y comprensión de las variables que impulsan las predicciones.

En conclusión, nuestro modelo seleccionado basado en Random Forest(modified_final) tiene mejores resultados debido a la mejora de medición de errores, la simplicidad de la arquitectura del modelo, la baja cantidad de datos que se necesita para entrenar el modelo y en la fácil interpretabilidad de las variables para el entrenamiento del modelo. Esto indica que el modelo seleccionado basado en Random Forest(modified_final) es un modelo de predicción más preciso y que supone una mejora respecto al modelo CNN-LSTM (Livieris et al., 2021).

Recomendaciones

A partir de los resultados obtenidos en esta investigación, se pueden hacer diversas recomendaciones para la mejora del modelo desarrollado basado en *Random Forest*, así como para futuras investigaciones en el área de predicción de precios de criptomonedas. A continuación, se presentan algunas recomendaciones clave.

El modelo basado en Random Forest mostró un buen rendimiento en términos de precisión y facilidad de interpretación, pero siempre hay espacio para mejoras. Una recomendación importante sería realizar un ajuste más detallado de los hiperparámetros del modelo, como el número de árboles en el conjunto, la profundidad máxima de los árboles, el número mínimo de muestras para dividir un nodo y el número mínimo de muestras por hoja. Estos ajustes podrían optimizar la capacidad del modelo para generalizar y mejorar su rendimiento en predicciones futuras.

Aunque el modelo Random Forest fue efectivo con el conjunto de datos utilizado en este trabajo, ampliar el conjunto de datos podría mejorar aún más su precisión. Se recomienda incorporar más datos históricos de criptomonedas, lo que permitiría al modelo aprender de una mayor variedad de patrones y condiciones de mercado. Además, la inclusión de factores adicionales, como indicadores técnicos o información sobre eventos globales relevantes, podría enriquecer el modelo y aumentar su capacidad predictiva.

El preprocesamiento adecuado de los datos es fundamental para el rendimiento del modelo. Se sugiere mejorar el manejo de los valores faltantes utilizando técnicas más avanzadas, como la imputación basada en modelos o la interpolación, que podrían mejorar la calidad de los datos utilizados para entrenar el modelo. Además, aplicar métodos de escalado y normalización de los datos, como Min-Max Scaling o Standardization, podría ayudar a mejorar la convergencia

del modelo y evitar que los árboles de decisión se vean influenciados por características de gran escala.

El modelo Random Forest es conocido por su capacidad para proporcionar puntuaciones de importancia de las características, lo que facilita su interpretación. Sin embargo, en el caso de este trabajo, se recomienda seguir explorando la interpretación más detallada de cómo cada característica afecta las predicciones. Una opción podría ser el uso de técnicas de visualización para representar de manera más clara la importancia de las características y cómo estas influyen en las decisiones de los árboles de decisión, lo que aumentaría la comprensión del modelo y la confianza de los usuarios en sus resultados.

Dado que los precios de las criptomonedas son altamente volátiles y están sujetos a condiciones de mercado cambiantes, se recomienda la implementación de un sistema de aprendizaje continuo para el modelo. Esto implicaría reentrenar el modelo Random Forest de forma periódica con datos nuevos, lo que permitiría adaptarlo a las fluctuaciones y a los patrones emergentes del mercado. Además, un sistema de validación en tiempo real podría ser útil para ajustar el modelo de acuerdo con el rendimiento observado en condiciones del mercado actuales.

Aunque el modelo Random Forest mostró buenos resultados, sería interesante explorar variantes de este algoritmo, como Gradient Boosting Machines (GBM) o XGBoost, que han demostrado un rendimiento superior en ciertos casos. La experimentación con estos modelos podría revelar formas de mejorar la precisión de las predicciones y proporcionar una comparación directa que ayude a identificar el mejor enfoque para el análisis de precios de criptomonedas.

Referencias

Alessandretti, L., Elbahrawy, A., Aiello, L. M., & Baronchelli, A. (2018). Anticipating Cryptocurrency Prices Using Machine Learning. *Complexity*, *2018*. https://doi.org/10.1155/2018/8983590

Breiman, L. (1996). *Bagging Predictors*. *24*, 123–140.

Breiman, L. (2001). *Random Forests*. *45*, 5–32.

Buterin, V. (2013). *Ethereum White Paper*. Recuperado de https://ethereum.org/en/whitepaper/

Carneiro, T., Da Nobrega, R. V. M., Nepomuceno, T., Bian, G. Bin, De Albuquerque, V. H. C., & Filho, P. P. R. (2018). Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. *IEEE Access*, *6*, 61677–61685. https://doi.org/10.1109/ACCESS.2018.2874767

Catalini, C., & Gans, J. S. (2016). Some Simple Economics of the Blockchain. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.2874598

CDIO. (2010). The CDIO Initiative. *The CDIO Standards (with Customized Rubrics)*, *December*, 1–14. http://www.cdio.org

CFA Institute. (2023). *Technical Analysis: An Introduction to Chart Patterns*. Recuperado el 2 de noviembre de 2024, de https://cfainstitute.org

Chen, W., Zheng, Z., Cui, J., Ngai, E., Zheng, P., & Zhou, Y. (2018). *Detecting Ponzi Schemes on Ethereum*. 1409–1418. https://doi.org/10.1145/3178876.3186046

Chugh, A. (2020). *MAE, MSE, RMSE, Coefficient of Determination, Adjusted R Squared — Which Metric is Better? | by Akshita Chugh | Analytics Vidhya | Medium*. https://medium.com/analytics-vidhya/mae-mse-rmse-coefficient-of-determination-adjusted-r-squared-which-metric-is-better-cd0326a5697e

Coinbase. (2024). *Cryptocurrency Market Dynamics in 2024*. Coinbase. Recuperado el 2 de noviembre de 2024, de https://coinbase.com

Coinbase Institutional. (2024). *2024 Crypto Market Outlook*. Recuperado de https://www.coinbase.com

CoinMarketCap. (2024). *Ethereum Price History*. Recuperado de https://coinmarketcap.com/currencies/ethereum/

Colianni, S., Rosales, S., & Signorotti, M. (2015). Algorithmic Trading of Cryptocurrency Based on Twitter Sentiment Analysis. *CS229 Project*, 1–5. http://cs229.stanford.edu/proj2015/029_report.pdf

DappRadar. (2023). *DApp Industry Overview: 2023*. Recuperado de https://dappradar.com/

Ethereum Foundation. (2024). *Ethereum 2.0: The Future of Ethereum*. Recuperado de https://ethereum.org/en/eth2/

Feurer, M., & Hutter, F. (2019). *Hyperparameter Optimization*. 3–33. https://doi.org/10.1007/978-3-030-05318-5_1

Foley, S., Karlsen, J. R., & Putniņš, T. J. (2019). Sex, drugs, and bitcoin: How much illegal activity is financed through cryptocurrencies? *The Review of Financial Studies*, 32(5), 1798-1853. https://doi.org/10.1093/rfs/hhz015

FundCount. (2024). *Why Cryptocurrency is Here to Stay—A 2024 Update*. FundCount. Recuperado el 2 de noviembre de 2024, de https://fundcount.com

Hefner, J. T., Spradley, M. K., & Anderson, B. (2014). Ancestry assessment using random forest modeling. *Journal of Forensic Sciences*, *59*(3), 583–589. https://doi.org/10.1111/1556-4029.12402

Hileman, G., & Rauchs, M. (2017). 2017 Global Cryptocurrency Benchmarking Study. *SSRN Electronic Journal*, *44*(0). https://doi.org/10.2139/ssrn.2965436

Investopedia. (2023). *Candlestick*. Recuperado el 2 de noviembre de 2024, de https://investopedia.com

Jiang, Z., & Liang, J. (2018). Cryptocurrency portfolio management with deep reinforcement learning. *2017 Intelligent Systems Conference, IntelliSys 2017*, *2018-Janua*, 905–913. https://doi.org/10.1109/IntelliSys.2017.8324237

Kahn, C. M. (2021). The transition to Ethereum 2.0: Understanding the new consensus mechanism. *Journal of Blockchain Research*, 1(1), 12-22. https://doi.org/10.1016/j.jbr.2021.01.003

Lamon, C., Nielsen, E., & Redondo, E. (2016). Cryptocurrency Price Prediction Using News and Social Media Sentiment. *Pdfs.Semanticscholar.Org*, *25*, 96. http://cs229.stanford.edu/proj2017/final-reports/5237280.pdf%0Ahttps://pdfs.semanticscholar.org/c3b8/0de058596cee95beb20a2d087dbcf8be01ea.pdf

Li, T. R., Chamrajnagar, A. S., Fong, X. R., Rizik, N. R., & Fu, F. (2018). Sentiment-Based Prediction of Alternative Cryptocurrency Price Fluctuations Using Gradient Boosting Tree Model. *Frontiers in Physics*, *7*(JULY). https://arxiv.org/abs/1805.00558v1

Liaw, A., & Wiener, M. (2002). Classification and Regression by randomForest. *R News*, *2*(3), 18–22.

Livieris, I. E., Kiriakidou, N., Stavroyiannis, S., & Pintelas, P. (2021). An advanced CNN-LSTM model for cryptocurrency forecasting. *Electronics (Switzerland)*, *10*(3), 1–16. https://doi.org/10.3390/electronics10030287

ManTech Publications. (2024). *Machine Learning Models for Predicting Cryptocurrency Trends*. ManTech Publications. Recuperado el 2 de noviembre de 2024, de https://mantechpublications.com

McCoy, M., & Rahimi, S. (2020). Prediction of Highly Volatile Cryptocurrency Prices Using Social Media. *International Journal of Computational Intelligence and Applications*, *19*(4). https://doi.org/10.1142/S146902682050025X

McNally, S., Roche, J., & Caton, S. (2018). Predicting the Price of Bitcoin Using Machine Learning. *Proceedings - 26th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2018*, 339–343. https://doi.org/10.1109/PDP2018.2018.00060

Meidan, Y., Bohadana, M., Shabtai, A., Ochoa, M., Tippenhauer, N. O., Guarnizo, J. D., & Elovici, Y. (2017). *Detection of Unauthorized IoT Devices Using Machine Learning Techniques*. http://arxiv.org/abs/1709.04647

Murphy, J. J. (1999). *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*. New York Institute of Finance.

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. *Bitcoin.org*. https://bitcoin.org/bitcoin.pdf

Nison, S. (2001). *Japanese Candlestick Charting Techniques: A Contemporary Guide to the Ancient Investment Techniques of the Far East* (2nd ed.). New York Institute of Finance.

Oskar, G., Atli, B., & R., S. (2006). Random Forests for land cover classification. *Pattern Recognition Letters*, *27*(4), 294–300. https://doi.org/10.1016/J.PATREC.2005.08.011

Protasiewicz, J. (2018). *Python AI: Why Is Python So Good for Machine Learning?* https://www.netguru.com/blog/python-machine-learning

Rafael Carrion Perez. (2020). Prediccion de Precios de Criptomonedas Con Random Forrest.pdf

Raiborn, C., & Sivitanides, M. (2015). Accounting Issues Related to Bitcoins. *Journal of Corporate Accounting & Finance*, *26*(2), 25–34. https://doi.org/10.1002/JCAF.22016

Reiff, N. (2020). *Bitcoin vs. Ethereum: What's the Difference?* Investopedia. https://www.investopedia.com/articles/investing/031416/bitcoin-vs-ethereum-driven-different-purposes.asp

Shah, D., & Zhang, K. (2014). Bayesian regression and Bitcoin. *2014 52nd Annual Allerton Conference on Communication, Control, and Computing, Allerton 2014*, *2*(1), 409–414. https://doi.org/10.1109/ALLERTON.2014.7028484

sklearn.ensemble.RandomForestRegressor — scikit-learn 0.24.2 documentation. (2021). https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

Sun Yin, H., & Vatrapu, R. (2017). A first estimation of the proportion of cybercriminal entities in the bitcoin ecosystem using supervised machine learning. *Proceedings - 2017 IEEE International Conference on Big Data, Big Data 2017*, *2018-Janua*(December 2017), 3690–3699. https://doi.org/10.1109/BigData.2017.8258365

Swalin, A. (2018). *Choosing the Right Metric for Evaluating Machine Learning Models — Part 1 | by Alvira Swalin | USF-Data Science | Medium*. https://medium.com/usf-msds/choosing-the-right-metric-for-machine-learning-models-part-1-a99d7d7414e4

Swankie, G. (University of S. (2019). *Examining the Price Dynamics of the Cryptocurrency Market and Predicting Bitcoin Price Through the Application of Statistical Analysis and Deep Learning*. *August 2019*. https://doi.org/10.13140/RG.2.2.13654.40003

Suenaga, H. (2018). *Machine Learning 1: Lesson 1. My personal notes from machine learning… | by Hiromi Suenaga | Medium*. https://medium.com/@hiromi_suenaga/machine-learning-1-lesson-1-84a1dc2b5236

Zohar, A. (2015). Bitcoin: Under the Hood. *Communications of the ACM*, 58(9), 104-113. https://doi.org/10.1145/2701412

Apéndices

Apéndice A

El código completo utilizado para implementar y entrenar los modelos discutidos en estatesis está disponible en el siguiente enlace:

https://drive.google.com/drive/folders/1cbrP7ROvcwM0IqqMASNDI_R39e3vElR0?usp=sharing