Revista Internacional de Investigación e Innovación Tecnológica

Página principal: www.riiit.com.mx

Trading de criptomonedas mediante algoritmos de aprendizaje por refuerzo profundo y aprendizaje automático

Cryptocurrency trading using deep reinforcement learning and machine learning algorithms

Valdez-Rivera, W.[a], Saucedo-Zendejo, F.R.[a], Navarro-Acosta, J.A.[a]*, Musi-Gutiérrez, J.[b]

[a] Centro de Investigación en Matemáticas Aplicadas, Universidad Autónoma de Coahuila. Unidad Saltillo, 25020, Saltillo, Coahuila, México.
[b] Facultad de Sistemas, Universidad Autónoma de Coahuila. Unidad Saltillo, 25350, Arteaga, Coahuila, México.
wbaldo_valdez@uadec.edu.mx, fesaucedoz@uadec.edu.mx, alejandro.navarro@uadec.edu.mx*, jmusi@uadec.edu.mx

Innovación Tecnológica: Tecnología para la adopción y uso de criptomonedas.

Área de aplicación industrial: Transacciones bancarias, educación, entre otras.

Recibido: 05 septiembre 2023
Aceptado: 07 febrero 2024

Abstract

The aim of this article is to perform a comparison of the performance of machine learning and deep reinforcement learning algorithms for buying and selling cryptocurrencies based on price prediction. This is done by training and testing such models using daily data spanning the period from 2017-11-09 to 2022-05-01 and was downloaded from Yahoo finance page in their cryptocurrency section. It is observed that machine learning algorithms have high accuracy metrics in their predictions and execute quickly, although they do not result in high return on investment, nor do they surpass deep reinforcement learning in this regard.

Keywords: machine learning, cryptocurrencies, trading, reinforcement learning

Resumen

El objetivo de este artículo es realizar una comparación del desempeño de algoritmos de aprendizaje automático y de aprendizaje por refuerzo profundo para la compra y venta de criptomonedas basándose en la predicción de precios. Esto se lleva a cabo mediante el entrenamiento y prueba de dichos modelos usando información diaria que abarca el periodo del 2017-11-09 al 2022-05-01 y fue descargada de la página de Yahoo finance en su sección de criptomonedas. Se observa que los algoritmos de aprendizaje automático tienen métricas de precisión altas en sus predicciones y se ejecutan de forma rápida, aunque no resultan en alto retorno de la inversión, ni tampoco sobrepasan al aprendizaje por refuerzo profundo en este sentido.

Palabras clave: aprendizaje automático, criptomonedas, trading, aprendizaje por refuerzo.

I. Introducción

Las criptomonedas son un medio de cambio que se utiliza para realizar transacciones financieras, estas comparten tres características comunes: descentralización, pseudoanonimato y transparencia. Están descentralizadas en el sentido de que, en lugar de estar gobernados por una sola institución, se administran a través de una red de igual a igual, la mayoría de las cuales deben acordar qué transacciones son válidas. La seguridad en sus transacciones está basada en su criptografía, de donde heredan su nombre de criptomonedas (Trozze, 2022). Actualmente la criptomoneda con el mayor valor de capitalización, más establecida y estudiada es el Bitcoin. El uso del Bitcoin ha crecido vastamente en los últimos años como un activo financiero más que solo como una herramienta para transacciones (Sabry et al., 2020). En la actualidad existen más de 10 mil monedas y hasta 20,325 mercados para la transacción de estas (Soni, 2021). Para el 3 de noviembre del 2021 la capitalización total del mercado alcanzo un nuevo récord histórico de $2.8 billones de dólares, con casi $1 billón de dólares agregados en poco más de un mes. Por lo tanto, el número de instituciones financieras que incluyen criptomonedas en sus portafolios de inversión ha ido en aumento en los años recientes. Un ejemplo de esta tendencia es el caso de Indonesia, donde el ministerio de comercio reporta que de 2020 a 2021 se incrementó en un 50% el número de personas invirtiendo en criptomonedas (Restuputri et al., 2023). En el ámbito académico, se ha observado que aproximadamente un 85% de los artículos de investigación existentes sobre el trading de criptomonedas han sido escritos apenas desde el 2018, siendo el trading la compra y venta de activos cotizados (acciones, divisas y futuros) con el fin de obtener ganancias (Bancomer, 2021). En este sentido, para llevar a cabo la optimización dinámica de un portafolio de inversión se ha demostrado que las redes neuronales recurrentes (RNN) y las redes de gran memoria de corto plazo (LSTM) muestran un mejor desempeño que la media móvil integrada autorregresiva (ARIMA) (McNally et al., 2018). Para este propósito también se propuso un multi agente basado en una red Double Q - Learning, y en una red de Double Dueling Q - Learning con el cual se obtuvieron mejores resultados con respecto a la técnica de comprar y mantener (Buy and Hold) en términos de acumulación de ganancias (Patel, 2018). En (Kolla, 2020) se hizo uso de RNN con LSTM mostrando que el precio de los activos podía ser predicho con una alta precisión y los resultados de la predicción se compararon con regresión lineal y RNN usando células LSTM. En general, las aproximaciones con aprendizaje

profundo son bastante comunes, y se tienen casos de estudio con Deep Q Learning (DQN) (Gu et al., 2016) y Deep Boltzmann Machine (DBM) (Salakhutdinov and Hinton, 2009). Por otro lado, existen otras aproximaciones para el manejo de un portafolio de inversión. En (Alessandretti et al., 2018) se aplicaron árboles de decisión para la potenciación del gradiente (XGBoost) y una red LSTM en un portafolio de criptomonedas, donde ambas generaron ganancias en todo el período de prueba. Por su parte (Jiang and Liang, 2017) aplicó una política de gradiente determinista usando una función de recompensa directa para resolver el problema del manejo del portafolio donde se obtuvieron buenos resultados en términos de ganancia acumulada. En otro aporte (Liang et al., 2018) se propone adaptar tres versiones del aprendizaje por refuerzo basado en una política de gradiente profunda determinista (DDPG), política de optimización proximal (PPO) y política del gradiente (PG) para el manejo dinámico de un portafolio donde se obtuvieron buenos resultados en la predicción de valores de activos. En otra aproximación (Bu and Cho, 2018) se propuso una combinación de una doble neurona Q y aprendizaje no supervisado pre-entrenado con una máquina de Boltzmann profunda (DBM) para generar y mejorar la función Q óptima con el cual se obtuvieron las más altas ganancias en comparación con los modelos de aprendizaje profundo existentes (LSTM, CNN, MLP). En cuanto al aprendizaje automático clásico existen una variedad de aproximaciones y técnicas utilizadas en el trading que han logrado buenos resultados. En un estudio (Guo and Tuckfield, 2020) se aplicó procesamiento de lenguaje natural utilizando Naive Bayes, Random Forest comparándolo contra el desempeño de redes CNN y LSTM para predecir precios de acciones y de índices donde se concluye que el grado de precisión de las redes solo fue ligeramente mejor. Como detalle final, se muestra que el uso de series de tiempo con aprendizaje automático (Sen and Mehtab, 2020) muestra ser de las mejores aproximaciones para tratar el aspecto del trading financiero. Sin embargo, las comparaciones entre el desempeño de distintas técnicas usualmente se realizan entre técnicas de deep learning contra los algoritmos de aprendizaje automático. La comparación entre el aprendizaje por refuerzo y los enfoques antes mencionados es una situación no abordada en la literatura investigada hasta el momento. Por lo tanto, el objetivo de este artículo es estudiar estas técnicas contra las de aprendizaje automático, para tratar de determinar cuál enfoque presenta un mejor desempeño (retorno de inversión), las posibles causas de estos resultados y la conveniencia de aplicar uno u otro enfoque, con lo cual se brindará información importante y original para esta área de estudio. La estructura de este articulo es como sigue: La sección II presenta las técnicas computacionales y el equipo utilizado para este estudio, la Sección III brinda una descripción de la metodología experimental empleada en este artículo, la sección IV presenta los resultados obtenidos y su discusión correspondiente, mientras que algunas conclusiones son presentadas en la sección V.

II. Técnicas y equipo

La ejecución del código se desarrolló en una computadora con sistema operativo Windows 10 de 64 bits, procesador Intel(R) Core (TM) i5-9400 CPU de 2.9 GHz con 16 GB de RAM. Los algoritmos fueron implementados en el lenguaje de programación Python (lenguaje de propósito general) haciendo uso de las siguientes librerías:

- Tensorflow: librería de computación matemática, que ejecuta gráficos de flujo de forma eficiente. Estos están formados por operaciones matemáticas representadas sobre nodos, y cuya

entrada y salida es un vector multidimensional o tensor de datos.

- Keras: librería de redes neuronales artificiales de código abierto, y debido a que las redes neurales son un tipo particular de gráfico de flujo de datos, TensorFlow y Keras combinan perfectamente.
- Scikit Learn: librería que acopla algoritmos de clasificación, regresión, clustering y reducción de dimensionalidad. Además, se usa en conjunto con otras librerías como NumPy, SciPy y Matplotlib.

La base de datos utilizada en esta investigación se descargó de Yahoo! Finance, la cual cuenta con una sección especial para las criptomonedas (https://finance.yahoo.com/crypto/). Dicha base de datos cuenta con la información diaria de diferentes criptomonedas, por lo que en términos técnicos dicha información corresponde a una serie de tiempo.

Aprendizaje automático

El aprendizaje automático (Machine Learning) es una rama de la inteligencia artificial, y también puede considerarse un subcampo de ciencias de la computación que basa su estudio en el uso de algoritmos y datos con el objetivo de desarrollar esquemas que imiten la manera en la que aprenden los seres humanos (IBM, 2020). Fue definido en 1950 por el pionero Arthur Samuel como la automatización y mejora del proceso de aprendizaje de las computadoras basadas en sus experiencias sin ser explícitamente programadas para ello, asemejándose al aprendizaje humano y animal (Fradkov, 2020).

Los algoritmos de aprendizaje automático más comunes son:

- **Regresión Ridge (RegR):** La regresión lineal es la técnica estadística más simple y mayormente usada para modelos predictivos. Esta, brinda una ecuación donde las características son variables independientes y nuestra variable objetivo es dependiente de estas características **Ec. (1)** (Shubham, 2017).

$$y = X\beta + \epsilon \qquad \textbf{(Ec. 1)}$$

Donde X son las variables, β son los parámetros de dichas variables y ϵ es error distribuido de forma normal. Los parámetros β son desconocidos y hay que aproximarlos utilizando la técnica de mínimos cuadrados.
Básicamente se busca una recta que minimiza la suma de los residuales cuadrados de las distintas observaciones con respecto a ella, considerando varianzas y sesgos bajos.

- **Random Forest (RF):** Es un algoritmo tanto de clasificación como de regresión. Se basa en la construcción de árboles de decisión en distintas muestras de datos. Toma el voto mayoritario hecho por la predicción de los árboles en el caso de la clasificación y el valor promedio en caso de una regresión (Sruthi, 2021).
En general se utilizan técnicas de ensamble, es decir, combinar varios modelos para hacer predicciones en lugar de utilizar un modelo individual. Los tipos de ensamble son (Vadapalli, 2020).
Bagging: Son un grupo de modelos también conocidos como embolsado cuyo objetivo es incrementar la precisión y la estabilidad del aprendizaje en donde los agentes aprenden en paralelo unos de otros de forma independiente y posteriormente se determina el resultado más votado o promedio.

Boosting: Es también un grupo de modelos, pero funciona de manera diferente a Bagging. En este grupo de modelos el aprendizaje se da de forma adaptativa y secuencial con el objetivo de mejorar secuencialmente las predicciones de los metalagoritmos.

- **XGBoost (XGB):** Extreme Gradient Boosting es un algoritmo de aprendizaje automático para clasificación y regresión basado en árboles de decisión que utiliza un marco de refuerzo de gradiente (Gradient Boosting) (Chen and Guestrin, 2016). El objetivo principal de Gradient Boost es minimizar la función de pérdida añadiendo predictores con mal desempeño mediante un algoritmo de optimización de descenso de gradiente. Tiene tres componentes principales (GreatLearning, 2020):
  Función de perdida: Cumple el papel de estimar cual es el mejor modelo para las predicciones utilizando los datos proporcionados.
  Predictores de mal desempeño: Árboles de decisión que clasifican los datos de peor manera que si fuera de forma aleatoria.
  Modelo aditivo: Proceso iterativo y secuencial donde se agregan árboles de decisión. Cada iteración, en principio, debe de reducir el valor de la función de perdida. Se agrega una cantidad de árboles hasta que la perdida es muy baja o ya no existe una mejora.

Aprendizaje por refuerzo

El uso de técnicas de aprendizaje por refuerzo se ha extendido a diversas áreas de la ciencia. Este funciona como una intersección entre varias áreas de estudio como las ciencias computacionales, la ingeniería, el aprendizaje automático, entre otras. El problema en general que resuelve este aprendizaje es el de la toma de decisiones de forma óptima. Los componentes básicos de este aprendizaje son conocidos como agente y entorno. El agente básicamente es un programa cuyo objetivo es la toma de decisiones para alcanzar una meta. Mientras que el entorno es el problema por resolver mediante las decisiones del agente. Existen además elementos adicionales a considerar (Torres, 2021):

- **Estado**: el conjunto de variables que representan al entorno es conocido como espacio de estados (space state). Un estado es un valor particular de ese espacio de variables. Usualmente el agente no tiene disponible todo el espacio de estados, entonces al estado que puede observar se le llama observación.
- **Acción y función de transición:** En cada estado, el entorno pone a disposición un conjunto de acciones (actions space) entre las que el agente puede elegir una utilizando distintos criterios. El agente influye en el medio ambiente a través de estas acciones. El entorno puede cambiar de estado como respuesta a la acción del agente. La función responsable de este mapeo se llama función de transición. El entorno también puede proporcionar una señal de recompensa como respuesta. A esto se le llama función de recompensa. El conjunto de funciones de transición y recompensa se denomina modelo del entorno (Morales, 2019).
- **Recompensa:** Existe una tarea u objetivo definido en un entorno, este puede proporcionar una señal de recompensa (reward) al agente como respuesta a sus acciones. Esta es un mecanismo de incentivo que le dice al agente qué es correcto y qué está mal usando recompensa y castigo. La idea

principal es que los agentes maximicen las recompensas totales (Zhang, 2021).

Las interacciones entre el entorno y el agente se pueden prolongar durante varios ciclos. A cada ciclo se le conoce como iteración. Al conjunto de un estado, acción, recompensa por la acción y estado subsecuente en cada iteración hasta el final se le llama experiencia y se representa como $s_t, a_t, r_{t+1}, s_{t+1}$ (Torres, 2021). La tarea que el agente que está tratando de resolver puede tener un final o puede ser continua. Cuando se tiene un final se le llama tarea episódica (episodic tasks), cuando no lo tienen son tareas continuas (continuous tasks), y a la secuencia de iteraciones hasta el final de una tarea episódica se le llama episodio (episode). Cuando no se tiene un fin en la tarea, la secuencia de iteraciones se le conoce como trayectoria que se representa del siguiente modo:

$$\tau = (s_0, a_0, r_1, s_1, a_1, r_2, s_2, \cdots, a_H, r_{H+1}, s_{H+1}) \quad \textbf{(Ec. 2)}$$

Donde H es la longitud de la trayectoria conocida como horizonte. En general, los agentes requieren varios episodios para aprender la forma de resolver una tarea. A la suma de las recompensas a través de un episodio se le conoce como retorno (return), valor que debe de ser maximizado por el agente ya que representa el aumento en términos relativos de la inversión inicial.

Q – Learning

Q-learning es una técnica de aprendizaje automático que pertenece a la familia de algoritmos de aprendizaje por refuerzo y basados en valor. Los algoritmos basados en valor actualizan la función de valor en función de una ecuación, en particular, la ecuación de Bellman (Para detalles ver (Lucarelli and Borrotti, 2020)). Este algoritmo basa su aprendizaje en el seguimiento de una serie de normas que le indican a los agentes las acciones que deberían ser tomadas bajo determinadas circunstancias, sin requerir adaptaciones para tratar con problemas con recompensas o con transiciones estocásticas, o modelos del entorno. La función Q usa la ecuación de Bellman y toma dos valores de entrada, los estados s y las acciones a. A grandes rasgos los pasos a seguir por Q – learning son:

1. **Inicializar la tabla Q**: con columnas correspondientes a acciones, mientras que las filas corresponden a los diferentes estados (Fig. 1).
2. **Elegir una acción:** determinar una acción a realizar.
3. **Realizar la acción:** los pasos 2 y 3 son realizados de forma indefinida hasta que el entrenamiento se detiene. Una acción a se elige en un estado s basado en la tabla Q, al inicio todos los valores deben de ser 0. Después se va a actualizando según la ecuación de Bellman.
4. **Medir la recompensa:** después de realizada una acción, se observa un resultado y se obtiene una recompensa.
5. **Actualizar la tabla Q:** Esto se hace actualizando la función Q, maximizándola (**Ec. (3)**). En este caso, la función Q regresa la recompensa futura esperada de dicha acción en el estado actual:

$$Q(s, a)_{nuevo} = Q(s, a)_{actual} + \alpha[R(s, a) + \gamma(max)Q^j(s^j, a^j) - Q(s, a)_{actual}] \quad \textbf{(Ec. 3)}$$

Donde α es la taza de aprendizaje, con valores entre 0 y 1. Implica que tanto se actualizan los valores Q, si es cercano a 0 entonces nunca se actualizan y no se aprende nada, si es cercano a 1 entonces los valores se actualizan rápido y el aprendizaje es alto (DeepRobotics, 2016).

Deep Q – Learning (DQN)

El algoritmo Q - Learning funciona de forma adecuada cuando el entorno no es muy complejo, y la cantidad de estados y acciones es limitada. Pero cuando dichos valores aumentan, el uso de este algoritmo se vuelve poco conveniente. Con esto en mente, el algoritmo Deep Q-Network o DQN fue desarrollado (Mnih et al., 2015). Este algoritmo constituye una combinación de redes neuronales profundas (Deep Neural Networks) con el algoritmo Q – learning, en el que se utiliza una red neuronal en la aproximación de la función Q con el objetivo de descartar el uso de una tabla para su representación.

Las redes neuronales artificiales (RNA) imitan el funcionamiento de las neuronas cerebrales de los seres vivos, que trabajan y están conectadas entre sí. La cuales, con entrenamiento y experiencia, crean y refuerzan conexiones para aprender algo (Julian, 2016). Una red neuronal sencilla se compone de tres capas, los valores de entrada (input layer), la capa escondida (hidden layer), y la capa de salida (output layer). En la primera capa se encuentran los valores de entrada en la forma de un vector columna de tamaño n. La última capa tiene todos los valores de salida posibles con valores entre 0 y 1 representando que tan probable es que cada valor sea cierto. Y finalmente se tiene la capa media, la capa escondida, aquí se realizan las transformaciones de los valores iniciales para dar con el resultado de la capa final. En ocasiones cuando el problema a resolver es complejo, se requiere el uso de una RNA con una estructura más compleja, es decir, una RNA con una cantidad grande de capas ocultas (Fig. 1).

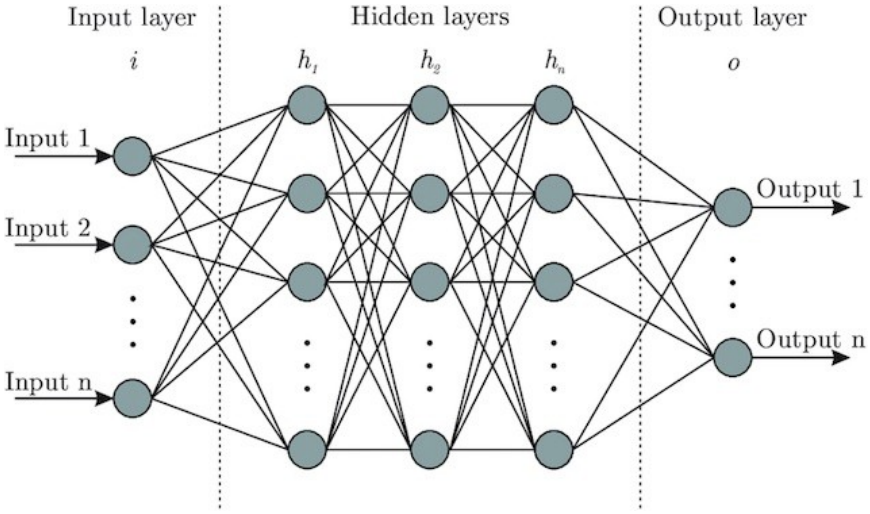

Figura 1. Esquema de neurona con n capas ocultas (Shukla, 2019).

Estas redes son conocidas como Redes Neuronales Profundas. En general, el aprendizaje profundo se considera como una colección de estrategias usando las propias redes neuronales para la resolución de problemas del aprendizaje automático (Para detalles ver (Torres, 2020)).

En DQN, los estados son los valores de entrada de la neurona, y los resultados son los valores Q de todas las acciones posibles (Gupta, 2018). Los pasos para el desarrollo del algoritmo son los siguientes (Choudary, 2019):

1. Toda la experiencia es guardada.

2. La siguiente acción es determinada por el valor máximo arrojado por la red neuronal.
3. Una vez que se tiene la función objetivo (ecuación de Bellman para DQL):

$$Q(s, a) = R(s, a) + \gamma(max_{a'})Q(s^j, a^j) \quad \textbf{(Ec. 4)}$$

Se calcula la función de perdida (loss function):

$$L = E[(R(s, a) + \gamma(max_{a'})Q(s^j, a^j) - Q(s, a))^2] \quad \textbf{(Ec. 5)}$$

Que será la función por minimizar con el algoritmo de descenso del gradiente estocástico (Stochastic Gradient Descendent).

Una vez que se ha calculado la perdida entre los valores objetivos Q y los valores Q actuales, se puede hacer retropropagación en la red y actualizar los valores de sesgo b y los pesos W (Youngson, 2020).

III. Métodos experimentales

Para el entrenamiento de los algoritmos de aprendizaje automático, se dividió el conjunto de datos en conjuntos de entrenamiento-validación que abarca el periodo 2017-11-09 al 2021-08- 01 obteniendo 1360 registros, mientras que el conjunto de prueba comprende el periodo 2021-08-02 al 2022-05-01 es decir, 273 registros. Posteriormente se utilizaron modelos autorregresivos para pronósticos de series de tiempo. Siendo la autorregresión un modelo de series de tiempo que utiliza observaciones de pasos de tiempo anteriores como valores de entrada a un modelo de regresión para la predicción del valor en el siguiente paso de tiempo (Brownlee, 2021). Se consideraron ventanas de 10 días (lags) antes de la fecha a predecir el valor de cierre de la moneda durante todo el intervalo de datos de entrenamiento. Además, para evaluar el desempeño de los algoritmos, se hace uso del backtesting sobre el conjunto entrenamiento-validación. La idea detrás de este método es evaluar la precisión de un modelo de pronóstico utilizando datos históricos existentes. El proceso es iterativo y se repite en varias fechas presentes de los datos históricos. Esto es útil para evitar resultados sesgados debido al sobre entrenamiento de los modelos. Los modelos aplicados para la autorregresión fueron la regresión Ridge (RegR), Random Forest (RF) y XGBoost (XGB). Los hiperparámetros de estos modelos se eligieron usando Grid Search (GS). GS es el proceso de ajustar los hiperparámetros para mejorar el desempeño de un modelo determinado, para detalles ver (Lutins, 2017). Para empezar el proceso de trading se consideran únicamente 3 acciones: comprar, vender y mantener. Para este proceso se inicia con la cantidad de 350 dólares de presupuesto para el programa que sigue tres reglas:

1. Si la predicción obtenida es mayor al precio actual de la moneda, y se tiene el dinero suficiente, entonces se compra. Se le resta el precio de la moneda al presupuesto y aumenta en 1 el inventario (que empieza en ceros).
2. Si la predicción obtenida es menor al precio actual de la moneda, y se tiene alguna moneda en el inventario, entonces se vende. Se agrega el precio de venta al presupuesto y disminuye en 1 el inventario.
3. Si ya no se tiene dinero ni monedas en el inventario se acaba la simulación.

Este proceso se realiza con las ventanas de 10 días durante todo el período de prueba (273 días). Durante toda la simulación se mide el tiempo de ejecución, el retorno de inversión (ROI) y el error porcentual cuadrático medio (MAPE). Por lo tanto, se esperan obtener valores altos de ROI con niveles bajos de

MAPE. Esto nos indica que tanto hemos aumentado en términos relativos la inversión inicial. Para garantizar la validez de los resultados la experimentación se realiza 20 veces y se considera el promedio de los valores.

Configuración experimental para Aprendizaje por refuerzo profundo

Se define a el agente que será el encargado de llevar las transacciones de los activos. Los atributos para el DQN fueron:

- **Tamaño de estados**: Ventana de tiempo para predicciones.
- **Acciones:** Número de acciones a tomar (comprar, vender y mantener).
- **Inventario**: Cantidad de monedas disponibles.
- **Gamma**: Factor de descuento de la ecuación de Bellman.
- **Épsilon**: política codiciosa inicial (porcentaje de veces que se toma una decisión aleatoria).
- **Épsilon final**: Valor final de épsilon.
- **Decaimiento de épsilon**: Tasa de decaimiento del valor (que tanto porcentaje del valor original de conserva tras cada iteración).

Después se describe la arquitectura de la red con un modelo secuencial:

- **Primera capa**: Densamente conectada con 128 neuronas. Función de activación: Relu.
- **Segunda capa**: Densamente conectada con 64 neuronas. Función de activación: Relu.
- **Tercera capa**: Densamente conectada con 32 neuronas. Función de activación: Relu.
- **Cuarta capa**: Capa de activación de salida con la misma cantidad de neuronas que acciones posibles del agente. Función de activación: Lineal.

Se compila el modelo con la función de pérdida del MAPE utilizando el optimizador del gradiente descendiente estocástico (SGD) con una tasa de aprendizaje (learning rate) de 0.001. Se cargan los datos de BNB (2019-01-01 al 2022-05-01) que se utilizarán para entrenar a la red. Con una función se crean los estados que alimentan a la red y los elementos son los siguientes:

- **Dinero**: Inversión inicial.
- **Data**: Datos de los precios de cierre diarios de la moneda a analizar.
- **Timestep**: Fecha en la cual se quiere hacer una predicción.
- **Window size**: Tamaño de la ventana de tiempo (cantidad de días previos a la fecha de predicción para la construcción de estados).

Se define la ventana de tiempo de 10 días, el número de episodios (o epochs) como 20 (cantidad de veces que se recorre todo el data set), el batch size de 32 (datos a analizar antes de retropropagar y ajustar los pesos de la red) y una inversión inicial de 100 dólares. Con esto finalmente se crea el objeto agente para llegar a la parte del bucle de entrenamiento. Se crean los estados correspondientes con las siguientes condiciones para la toma de decisiones:

1. Si la acción seleccionada por la red es 0, entonces se mantiene la posición (no se compra ni se vende nada).
2. Si la acción seleccionada por la red es 1, entonces se compra la moneda. Se añade al inventario y se resta su valor al dinero disponible.
3. Si la acción seleccionada por la red es 2, entonces se vende (considerando que el inventario no está vacío). Para el precio de venta se utiliza el último valor del día y se vende a ese precio. Se agrega el precio de venta al dinero disponible y se le resta una unidad al inventario. La recompensa que se le

da al agente es la diferencia entre el valor del portafolio actual y el inicial.
4. Si ya no hay dinero ni acciones se acaba el episodio.
5. Si se llega al último día del set de datos se acaba el episodio.

IV. Discusión de Resultados

El primer algoritmo de ML que se puso a prueba es la Regresión Ridge, con hiperparámetros sin ajuste, es decir sin el uso de Grid Search. La Figura 2 muestra las predicciones que dicho algoritmo brinda en comparación con los valores reales de la serie de tiempo (datos de prueba).

Figura 2. Predicciones de la regresión Ridge sin Grid Search.

Las métricas que se obtuvieron fueron una R cuadrado de 0.626 y un MAPE de 8.84. Posteriormente se aplica el ajuste de hiperparámetros considerando distintas combinaciones entre los lags y lambda. Para lambda se consideraron 10 valores entre 10^{-3} y 10^{3} separados de forma uniforme en una escala logarítmica, para los lags se consideraron distintos valores entre 1 y 20. Los parámetros obtenidos fueron: Lambda de 0.001 y Lags igual a 1. Alcanzando así métricas de R cuadrado de 0.947 y un MAPE de 3.04. Es decir, se lograron obtener mejores métricas. La Figura 3 muestra la mejora en las predicciones del modelo.

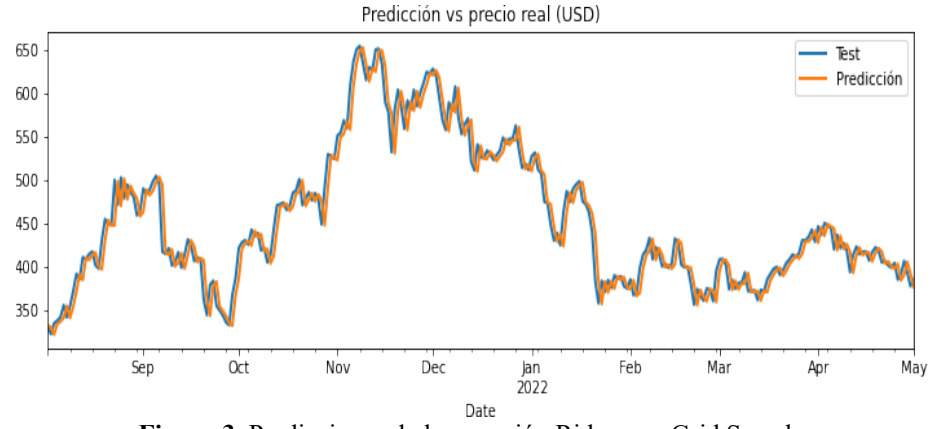

Figura 3. Predicciones de la regresión Ridge con Grid Search.

Este proceso se ejecuta 20 veces con el fin de corroborar el desempeño del modelo aplicando una estrategia de compra y venta y se obtienen los resultados promedio de tiempo de ejecución de 1.64 segundos, un ROI de 1.4 y MAPE de 3.04. Este mismo proceso se sigue para los algoritmos Random Forest y XGBoost. Por compacidad la Tabla 1 muestra los resultados de los tres algoritmos de ML al realizar las predicciones, es decir, los hiperparámetros con y sin ajuste (HSA y HCA respectivamente), Así como las mejores métricas obtenidas.

Tabla 1. Resultados de los algoritmos.

Técnica		HSA	HCA	Métricas	
RF	# de árboles	100	500	Tiempo	180.33 s
	Prof. Max.	sin	5	MAPE	4.95
	Lags	1	10	ROI	1.38
XGB	# de árboles	100	100	Tiempo	7.64 s
	Prof. Max.	4	4	MAPE	5.75
	Lags	10	10	ROI	0.96
RegR	Lambda	1	0.001	Tiempo	1.64 s
	Lags	10	1	MAPE	3.04
				ROI	1.4
DQN	Configuración descrita en sección III	No aplica		Tiempo	3417 s
				MAPE	1.75
				ROI	60

Elaboración propia.

Como se puede observar los retornos de inversión mayores los brindan los algoritmos de RF y RegR, lo cual es coherente al observar las métricas MAPE de estas técnicas. Sus valores indican que dichos algoritmos tienen una capacidad similar de predicción sobre la serie de tiempo.

A continuación, se procede a implementar el enfoque DQN. El cual presenta un tiempo de ejecución bastante mayor a los algoritmos de ML, cerca del 3000% mayor en comparación con RegR. Sin embargo, este enfoque da un retorno de inversión mayor de alrededor de 40% en comparación con el mejor algoritmo de ML presentado en este trabajo.

Teniendo en cuenta los resultados de todos los algoritmos analizados se puede concluir que las técnicas de aprendizaje automático son estrategias rápidas y seguras para la implementación de algoritmos de trading automatizado de criptomonedas, mientras que los algoritmos de aprendizaje por refuerzo profundo presentan una alternativa atractiva para conseguir retornos de inversión mucho mayores, con un costo computacional y riesgos mayores. Por lo tanto, la elección de la técnica de aprendizaje para un algoritmo de trading automatizado debe estar basada en una estrategia de inversión bien establecida.

V. Conclusiones

Cómo se puede apreciar en los resultados, el mayor retorno, así como el menor tiempo de ejecución que se obtuvo con los algoritmos de ML los consiguió la regresión Ridge. Considerando estos resultados, entonces este algoritmo (de entre los implementados en las condiciones mostradas) resulta ideal para una estrategia moderada de trading, disminuyendo los riesgos lo máximo posible. Por otro lado, los valores elegidos utilizando la técnica del Grid Search no resultaron en una mejora significativa de los algoritmos, pero no se descarta su uso aplicando otros enfoques de ajuste de hiperparámetros.

En el caso del aprendizaje por refuerzo profundo se obtuvieron muy buenos resultados con la estrategia empleada. Se obtuvieron retornos de hasta 60 veces la inversión inicial. Sin embargo, el costo computacional empieza a hacerse presente en este caso, tomando hasta más de 3500 segundos (casi un día entero) en la ejecución de cada episodio. Si se requiriera ampliar el análisis a más monedas durante un tiempo mayor, y con una mayor cantidad de episodios, el costo computacional es un factor para tener en cuenta. El usar DQN representa una estrategia que en un inicio puede ser algo inestable (con retornos variados de la inversión) y que consume una gran cantidad de recursos, pero que representa un enfoque interesante de implementar debido a los grandes retornos.

Finalmente, todos los resultados anteriores muestran que los algoritmos de aprendizaje pueden ser aplicados exitosamente con fines productivos, específicamente en el sector de trading automatizado de criptomonedas. Sin embargo, estos resultados podrían servir de punto de partida en el análisis de técnicas de aprendizaje automático para el trading automatizado en otros sectores de la economía.

VI. Referencias

Alessandretti, L., ElBahrawy, A., Aiello, L. M., & Baronchelli, A. (2018). Anticipating cryptocurrency prices using machine learning. Complexity, 2018, 1-16. Accedido 07-03-2021.

Bancomer, B. (2021). 'trading'? Obtenido de https://www.bbva.com/es/que-es-trading-que-hace-falta-para-operar/. Accedido 21-02-2021.

Brownlee, J. (2017). Autoregression models for time series forecasting with python. Machine Learning Mastery, 2(01). Obtenido de https://machinelearningmastery.com/autoregressionmodels-time-series-forecasting-python/. Accedido 06-07-2022.

Bu, S. J., & Cho, S. B. (2018). Learning optimal Q-function using deep Boltzmann machine for reliable trading of cryptocurrency. In Intelligent Data Engineering and Automated Learning–IDEAL 2018: 19th International Conference, Madrid, Spain, November 21–23, 2018, Proceedings, Part I 19 (pp. 468-480). Springer International Publishing. Accedido 22-02-2021.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794). Accedido 01-12-2022.

Choudhary, A. (2019). A hands-on introduction to deep q-learning using openai gym in python. Analytics Vidhya. Obtenido de https://www.analyticsvidhya.com/blog/2019 / 04/introduction-deep-qlearning-python/. Accedido 27-05-2021.

DeepRobotics (2016). A comprehensive approach to reinforcement learning.

Obtenido de https://vmayoral.github.io/robots,/ai,/deep/learning,/rl,/reinforcement/learning tutorial/. Accedido 28-05-2021.

Fradkov, A. L. (2020). Early history of machine learning. IFAC-PapersOnLine, 53(2), 1385-1390.

GreatLearning (2020). Understanding xgboost algorithm I what is xgboost algorithm? Obtenido de https://www.mygreatlearning.com/blog/xgboostalgorithm/. Accedido 05-07-2022.

Gu, S., Lillicrap, T., Sutskever, I., & Levine, S. (2016). Continuous deep q-learning with model-based acceleration. In International conference on machine learning (pp. 2829-2838). PMLR. Accedido 24-02-2021.

Guo, J. and Tuckfield, B. (2020). News-based machine learning and deep learning methods for stock prediction. Journal of Physics: Conference Series, 1642:012014. Accedido 15-03-2022.

Gupta, S. (2018). Sentiment analysis: Concept, analysis and applications. Toward Data Science. Obtenido de https://towardsdatascience.com/sentiment-analysis-concept analysis-and-applications-6c94d6f58c17. Accedido 25-02-2021.

IBM (2020). Machine learning. Obtenido de https://www.ibm.com/cloud/learn/machine-learning. Accedido 12-04-2022.

Jiang, Z., & Liang, J. (2017). Cryptocurrency portfolio management with deep reinforcement learning. In 2017 Intelligent systems conference (IntelliSys) (pp. 905-913). IEEE. Accedido 22-02-2021.

Juchli, M. (2018). Limit order placement optimization with deep reinforcement learning: Learning from patterns in cryptocurrency market data. Accedido 2402-2021.

Julián, G. (2016). Las redes neuronales: qué son y por qué están volviendo. Revista Xataka. Obtenido de https://www.xataka.com/robotica-e-ia/las-redes-neuronales-queson-y-porque-estan-volviendo. Accedido 20-05-2021.

Keller, A., & Scholz, M. (2019). Trading on cryptocurrency markets: Analyzing the behavior of bitcoin investors. Accedido 28-02-2021.

Kolla, B. (2020). Predicting crypto currency prices using machine learning and deep learning techniques. International Journal of Advanced Trends in Computer Science and Engineering, 9. Accedido 22-02-2021.

Liang, Z., Chen, H., Zhu, J., Jiang, K., & Li, Y. (2018). Adversarial deep reinforcement learning in portfolio management. arXiv preprint arXiv:1808.09940. Accedido 2202-2021.

Lucarelli, G., & Borrotti, M. (2019). A deep reinforcement learning approach for automated cryptocurrency trading. In Artificial Intelligence Applications and Innovations: 15th IFIP WG 12.5 International Conference, AIAI 2019, Hersonissos, Crete, Greece, May 24–26, 2019, Proceedings 15 (pp. 247-258). Springer International Publishing. Accedido 21-02-2021.

Lucarelli, G., & Borrotti, M. (2020). A deep Q-learning portfolio management framework for the cryptocurrency market. Neural Computing and Applications, 32, 17229-17244. Accedido 21-02-2021.

Lutins, E. (2017). Grid searching in machine learning: Quick explanation and python implementation. Medium,

Medium, 5. Obtenido de https://elutins.medium.com/gridsearching-in-machine-learning-quick-explanation-and-pythonimplementation-550552200596. Accedido 07-07-2022.

McNally, S., Roche, J., & Caton, S. (2018, March). Predicting the price of bitcoin using machine learning. In 2018 26th euromicro international conference on parallel, distributed and network-based processing (PDP) (pp. 339-343). IEEE. Accedido 22-02-2021.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A., Veness, J., Bellemare, M., Graves, A., Riedmiller, M., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. Nature, 518:529-33. Accedido 27-05-2021.

Morales, M. (2019). Deep Reinforcement Learning. Accedido 2-12-2021.

Patel, Y. (2018). Optimizing market making using multi-agent reinforcement learning. arXiv preprint arXiv:1812.10252. Accedido 22-02-2021.

Restuputri, D.P.; Refoera, F.B.; Masudin, I. Investigating Acceptance of Digital Asset and Crypto Investment Applications Based on the Use of Technology Model (UTAUT2). *FinTech* 2023, *2*, 388-413. https://doi.org/10.3390/fintech2030022

Sabry, F., Labda, W., Erbad, A., and Malluhi, Q. (2020). Cryptocurrencies and artificial intelligence: Challenges and opportunities. IEEE Access, 8: 175840 175858. Accedido 21-02-2021.

Salakhutdinov, R. and Hinton, G. (2009). Deep Boltzmann Machines, volume 5 of Proceedings of Machine Learning Research. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA. Accedido 24-02-2021.

Sen, J. (2022). Machine learning and deep learning in stock price prediction. Machine Learning in the Analysis and Forecasting of Financial Time Series, 29-67. Accedido 15-03-2022.

Shubham, J. (2017). A comprehensive beginners guide for linear, ridge and lasso regression in python and r. Obtenido de https://www.analyticsvidhya.com/blog/2017/06/a-comprehensive-guidefor-linear-ridge-and-lasso-regression/h2_5. Accedido 14-04-2022.

Shukla, L. (2019). Fundamentals of neural networks. Obtenido de https://wandb.ai/site/articles/fundamentals-of-neural-networks. Accedido 2105-2021.

Soni, S. (2021). Crypto rally: Total market cap hits new all-time high of 2,8t;1t added in just over a months time. Obtenido de https://www.financialexpress.com/market/crypto-rally-total-market-caphits-new-all-time-high-of-2-8t-1 t-added-in-just-over-a-months-time /2362504/. Accedido 25-11-2021.

Sruthi, E. (2021). Understanding random forest. Obtenido de https://www.analyticsvidhya.com/blog/2021/06/understanding-randomforest/. Accedido 05-07-2022.

Torres, J. (2020). A gentle introduction to deep reinforcement learning. Obtenido de https://towardsdatascience.com/drl-01-a-gentle-introduction-todeep-reinforcement-learning-405b79866bf4. Accedido 20-05-2021.

Torres, J. (2021). Introducción al aprendizaje por refuerzo profundo. Accedido 10-112021.

Trozze et al. (2022). Cryptocurrencies and future financial crime. Crime Science. https://doi.org/10.1186/s40163-021-00163-8.

Vadapalli, P. (2020). Bagging vs boosting in machine learning: Difference between bagging and boosting. Obtenido de https://www.upgrad.com/blog/bagging vs-boosting. Accedido 05-07-2022.

Youngson, M. (2020). Introduction to reinforcement learning. Obtenido de https://medium.com/swlh/introduction-to-reinforcement-learning63ff8923bd88. Accedido 27-05-2021.

Zhang, A. (2021). How to design a reinforcement learning reward function for a lunar lander. Obtenido de https://towardsdatascience.com/how-to-designreinforcement-learning-reward-function-for-a-lunar-lander-562a24c393f6. Accedido 2-12-2021.