Universidad Pontificia de Comillas

Diseño de un altcoin vinculado al USD basado en Deep Reinforcement Learning (IA)

Clave: 201600782
Autor: Álvaro Villadangos del Río
Tutor: José Portela González

MADRID | 7 de marzo de 2021

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGO

Introducción

En 2021 Bitcoin ha alcanzado una capitalización de mercado de 938,39B $ superando a JP Morgan, el banco con mayor capitalización de mercado del mundo (CoinGecko, 08). Bitcoin se ha convertido en una alternativa como valor refugio frente a opciones más tradicionales como el oro. Los factores principales han sido la pandemia mundial y la incertidumbre sobre las políticas monetarias de los bancos centrales como la Reserva Federal de Estados Unidos.

En los últimos años hemos estado viendo como medios de comunicación han fracasado en su labor informativa, grandes capitales han sido capaces de manipular el mercado de valores a su antojo y los bancos han abusado de su posición privilegiada provocando crisis financieras como la de 2008. El factor común es que los que tenían la sartén por el mango han salido ganando y el resto de la sociedad ha tenido que pagar las facturas quisieran o no.

Hasta el momento quienes generaban el problema también tenían la solución. Sin embargo, gracias a internet se está llevando un proceso de democratización real frente a los grandes poderes más tradicionales. Podemos ver como las redes sociales están reemplazando en su labor informativa a periódicos y televisiones haciendo públicos problemas y contenidos que estarían prohibidos mostrar en televisión. Por ejemplo, el presunto asesinato de George Floyd a manos de un policía dando lugar al movimiento mundial Black Lives Matter (Poidevin, 2020). En el sector bursátil, hemos visto como Wall Street bets, un foro de Reddit de inversores domésticos, conseguía aumentar un 400% las acciones de Game Stop haciendo perder a Merrill Lynch, con una posición en corto, alrededor de 5B$ que es equivalente a su capitalización de mercado (Sánchez, 2021). Gracias a internet ni el Estado puede controlar la información difundida, ni la SEC (Comisión de Bolsa y Valores de Estados Unidos) puede controlar al pequeño inversor.

En 2010, una persona anónima que responde al seudónimo de Satoshi Nakamoto se propone democratizar el dinero creando una tecnología novedosa y descentralizada de código abierto llamada blockchain. Su propuesta consistía en crear un sistema que escapara del control de los gobiernos, fuera totalmente democrático, anónimo y a su vez transparente. Esta filosofía nace como respuesta a los bancos centrales y gobiernos que manipulan el dinero de los individuos mediante los tipos de interés y empobrecen a los ahorradores con la inflación (Moreno, 2020). Además, con las tarjetas de crédito y la

penalización del dinero efectivo es imposible no dejar rastro de lo que hacemos o dejamos de hacer facilitando así los bancos el control sobre los individuos y empresas. Este medio de pago, que escapa al control de los gobiernos, ya está empezando a irrumpir en el día a día de personas y empresas. Plataformas de pago como PayPal ya permiten hacer transacciones con activos digitales y empresas como el club de futbol DUX realizó el primer fichaje con bitcoin en el mercado invernal de la temporada 2020-2021 (Marca, 2021). Por otro lado, gobiernos como el de España tratan de regular las monedas digitales para su control sin mucho éxito ya que no hay nada ni nadie que responda por bitcoin y además escapa al ámbito territorial que limita a los estados.

Junto con internet el acceso y generación a información se ha multiplicado exponencialmente y junto con ello el progreso en investigación. Progresivamente el exceso de información se ha convertido en un problema cada vez más complejo de abordar. Como respuesta a esta necesidad surge el campo del Big Data para el análisis masivo de datos y técnicas de decisión inteligente como el Machine Learning o la Inteligencia Artificial. Dando lugar a resultados impresionantes como los algoritmos de trading o resolviendo problemas como el del plegado de proteínas que llevaba 100 años planteado (Senior, 2020).

Estas técnicas son un arma de doble filo y ya se ha visto como la falta de regulación sobre la protección de datos y privacidad ha dado lugar a la manipulación de las masas. En el caso de Cambridge Analytics, se utilizaron estas técnicas para realizar análisis de sentimiento segmentando a la población de Reino Unido con el fin de manipular el voto británico a favor de la salida de Unión Europea y las elecciones presidenciales americanas de 2016 (News, 2019). También, gigantes de la información como Facebook y Google han sido múltiples veces sancionados por la vulneración del derecho a la intimidad y privacidad de sus usuarios.

Este TFG contribuye a la literatura del campo de investigación de *Reinforcement Learning* y *Blockchain*. Se estudiará la posibilidad de combinar estas dos tecnologías para generar una moneda digital que se autorregule para que sus usuarios no se vean perjudicados por la especulación. Mediante una Inteligencia Artificial, que será parte del blockchain, se ajustará el número de monedas en circulación para regular su precio de una forma estable y sin riesgo. Existen formas de alcanzar esta estabilidad, pero utilizando colaterales como divisas, otras monedas digitales o materias primas. La mayor desventaja de dichos métodos es que desvirtúan la esencia de la tecnología blockchain donde el

principio rector es que no haya ningún respaldo físico que represente su valor, sino que sea la propia comunidad quien decida su valor. Por ejemplo, un activo digital respaldado por oro, en caso de desplomarse su precio dicho activo se vería arrastrado perjudicando a su comunidad. Dotando de inteligencia a una moneda podemos conseguir dicha estabilidad sin que dependa de factores externos. Estamos hablando de una moneda digital con un banco central que no puede estar sujeto a influencias ni intereses particulares.

Por último, la investigación planteará un modelo de predicción por aprendizaje basado en técnicas de Deep Learning y Reinforcement Learning con el objetivo de demostrar que el activo digital propuesto es capaz de estabilizar el precio de un activo tan volátil como el bitcoin a través de una simulación.

El resto del documento se estructura de la siguiente forma. El capítulo 1, contiene el marco teórico de la investigación, la tecnología blockchain y la creación de la Inteligencia Artificial. El capítulo 2, presenta el diseño del altcoin y todas las modificaciones hechas. En el capítulo 3 se analizan los resultados obtenidos. En el capítulo 4 discutiremos los méritos del blockchain, la Inteligencia artificial junto con los problemas que se planteen y las limitaciones que hemos tenido para la ejecución de este estudio.

Capítulo 1: Marco teórico

Este capítulo establece la posibilidad teórica de realizar la investigación propuesta. Para ello, el capítulo se centra en los 6 conceptos más importantes de la investigación e intenta responder a una serie de preguntas. Los 6 conceptos y las preguntas son las siguientes:

- Blockchain: ¿Qué es Blockchain? ¿Cómo funciona? ¿Qué beneficios tiene?
- Banco Central: ¿Qué es y cuáles son sus funciones?
- Coins y stablecoin: ¿Qué es una moneda digital y qué alternativas han surgido?
- Machine Learning e Inteligencia Artificial: ¿Cómo aprenden las máquinas y qué tipos de problemas resuelven?
- Deep Learning y redes neuronales: ¿Para qué sirven? ¿En qué consiste el aprendizaje profundo?
- Reinforcement Learning: ¿Cómo funciona? ¿Por qué es capaz de adaptarse a problemas? ¿Por qué tiene sentido utilizarlo para este problema?

Una vez establecida la posibilidad teórica de realizar el proyecto, dentro del mismo capítulo también se expondrá las investigaciones relacionadas más relevantes.

En línea con la introducción estos conceptos, todos ellos muy técnicos y novedosos, será necesario entenderlos para tener una imagen fiel del objetivo inicial que es crear una stablecoin. El blockchain es la tecnología que programaremos con un API REST en la que se realizarán las transacciones y será la base de todo el proyecto. Primero, deberemos entender cómo funciona el sistema monetario tradicional para replicar lo que es un banco central de forma descentralizada en el blockchain. A continuación, deberemos entender que tipos de monedas hay y que función cumple cada una de ellas para comprender como funciona el negocio y que estrategias se siguen para que un proyecto sea exitoso. Entendida la parte monetaria se pondrá en contexto de forma breve el ámbito de la inteligencia artificial y sus campos, Machine Learning y Deep Learning, para explicar finalmente la técnica de Reinforcement Learning en la que se basará la inteligencia artificial para estabilizar la moneda como si fuera un banco central.

1.1. Blockchain

Un blockchain o cadena de bloques es una lista de registros en continuo crecimiento, llamados bloques, que se conectan y aseguran usando criptografía.

Un bloque, contiene información (Data), el hash del bloque anterior (Prev. Hash) y el hash del bloque actual (Hash). El primer bloque es conocido como el *genesis block*, contiene la emisión inicial de monedas o *Inicial Coin Offering* (ICO), y nunca desaparecerá del blockchain ya que es el primer bloque de la cadena. Su particularidad es que al no haber un bloque que le preceda no tendrá un hash anterior. El segundo bloque se unirá al primero incluyendo el hash anterior en él y generará un nuevo hash para que se pueda unir a este bloque otro bloque posterior. En caso de no coincidir el hash anterior con el hash del bloque anterior no podrá unirse el bloque a la cadena. Por ello decimos que los bloques se encuentran unidos criptográficamente.

Como podemos ver en la siguiente imagen encontramos un bloque donde Kirill envía a Hadelin 500 hadcoins, realiza una compra por 100 hadcoins en Ebay y envía a Joe 70 hadcoins. En este caso la información que contiene el bloque son transacciones, pero Kirill también podría haber enviado una imagen, un video o incluso un sistema operativo. Mientras lo que se quiera enviar sean datos puede enviarse lo que sea.

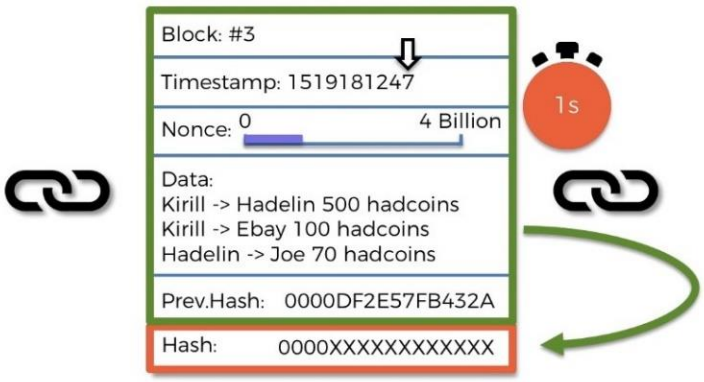

Figura 1: Ejemplo de un bloque de una cadena

Este bloque representa el tercer bloque del blockchain, que esta minado en el segundo 1519181247 a contar desde el 1 de enero de 1970. El hash del bloque anterior que sirve de unión al que estaría uniéndose es el 0000DF2E57FB432A y el nuevo hash que se está tratando de minar aún se desconoce 0000XXXXXXXXXXX. El motivo por el que comienzan por '0000' es para dificultar la tarea al generar un nuevo bloque. Cada 0 hace que sea 10 veces más complicado minar un bloque. Una vez encontrado el hash se valida el bloque por la comunidad y es hecho público en la red.

1.1.1. Hash: SHA256

El hash, necesario para la unión de los bloques en la cadena, funciona como una huella digital y cada bloque tiene su huella. Es posible que dos bloques coincidan con la misma huella, pero la probabilidad es muy pequeña. El SHA256 genera una huella digital para los documentos que se han añadido al bloque. Este algoritmo fue generado por la NSA y es completamente público, pero no indagaremos mucho sobre él en este TFG.

El nombre viene de Secure Hash Algorithm y 256 es el número de bits que ocupa en memoria. El hash siempre ocupa 64 caracteres que consisten en dígitos y letras haciendo el hash hexadecimal (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F). Es importante que este algoritmo no solo encripta documentos de texto sino para cualquier documento digital como videos, fotos, sistemas operativos, …

Hay 5 requisitos para los algoritmos hash:

- Una dirección, nunca se puede volver del hash al documento. No se puede recuperar un documento desde el hash.
- Determinístico, si utilizamos el algoritmo hash con el mismo documento de nuevo debemos generar exactamente el mismo hash.
- Computación rápida.
- Efecto avalancha, si se realiza un cambio por mínimo que sea en el documento se genera un hash significativamente diferente.
- Debe tolerar colisiones, el total de caracteres está limitado a 64 bits y así que la cantidad de combinaciones que se pueden generar está limitada. Por lo tanto, habrá colisiones entre bloques con el mismo hash, pero es muy poco probable. El principal problema es que si alguien fuera capaz de generar artificialmente estas colisiones sería capaz de modificar los documentos del bloque.

1.1.2. Descentralización P2P

Como comentábamos antes, los bloques están criptográficamente conectados y es así como se forma la cadena. De esta forma es como se añaden las transacciones a la cadena. Sin embargo, la inmutabilidad de la cadena se origina a medida que se añaden bloques porque al modificar un bloque en un punto dejará de encajar con el posterior. Si se quiere modificar un bloque también habrá que modificar el hash del resto de bloques de la cadena convirtiendo el fraude extremadamente difícil y costoso.

En un sistema descentralizado tenemos idealmente un número de computadores interconectados al mismo tiempo. El blockchain está copiado en todos estos ordenadores con todas las transacciones y todos los datos continuamente actualizándose.

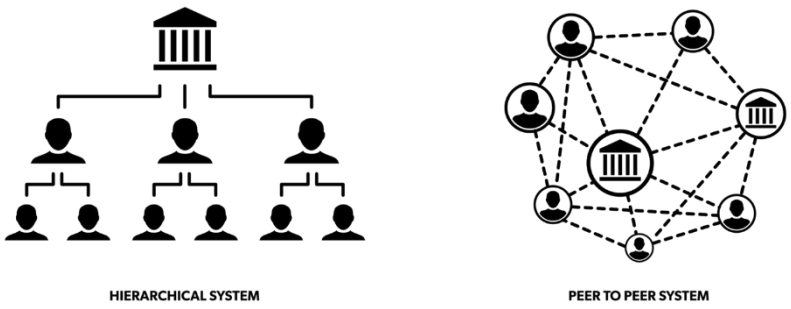

Figura 2: Comparación sistema centralizado y descentralizado

Cuando un bloque es añadido esto es comunicado al resto de computadores y validado por ellos. Posteriormente se añaden más bloques y mientras pasa este tiempo el blockchain comprueba constantemente la cadena. En el caso de darse un ataque al blockchain y conseguir el supuesto dificilísimo de reemplazar fraudulentamente una cadena entera desde un punto determinado, el resto de computadores identificarán que hay una irregularidad en esa cadena y reemplazarán la cadena fraudulenta por la cadena original. Por lo tanto, para modificar una cadena de forma fraudulenta no solo hay que reemplazar todos los bloques posteriores al bloque atacado, sino que además necesitaría atacar a más de la mitad de los computadores que participan en la red del blockchain al mismo tiempo. Cuanto más grande sea la red más segura será el blockchain. Así, es como conseguimos confiar en una tecnología donde nadie confía en nadie.

1.1.3. Minería

La minería de criptomonedas se puede definir como el conjunto de procesos necesarios para poder realizar validaciones, así como el procesamiento de las transacciones de una criptomoneda dentro de un blockchain o una cadena de bloques.

En un bloque tenemos distintos apartados:

- El número del bloque en la cadena.

- Datos, donde se recogen las transacciones realizadas con una moneda propia del blockchain.
- Hash anterior, el hash del bloque anterior que une este bloque con el anterior.
- Nonce o *Number use only once.*
- Hash, es el hash propio de este bloque. Para generar este hash el algoritmo de encriptado tiene en cuenta el Data, el Nonce y el hash anterior para generar el hash. Este proceso apenas toma unos instantes.

Para minar un bloque el minero deberá proponer bloques distintos al algoritmo con el fin de generar un hash que el blockchain acepte. Sin embargo, el número de bloque no se puede modificar por cuestiones de orden en la cadena, el hash anterior es necesario mantenerlo para que el bloque pueda conectarse a la cadena y los datos no pueden ser modificados porque rompería con la inmutabilidad de la cadena. Por todo esto, se introduce el nonce. El nonce es un apartado del bloque donde puede introducir cualquier número tantas veces quiera dando como resultado diferentes hashes. Es una forma de darle margen de maniobra a los mineros en su carrera por ser el primero en añadir el próximo bloque de la cadena.

Hasta el momento sabemos que podemos generar distintos hashes variando el número del nonce. Partimos de que el hash no es más que un número hexadecimal como hemos explicado ya anteriormente. De todos los hashes posibles sabemos que el más pequeño que sería sesenta y cuatro “0” s hasta el más grande sería sesenta y cuatro “F” s.

El algoritmo del blockchain establecerá un número objetivo para que los mineros alcancen ese hash. De esta forma cualquier hash que supere el target será rechazado por el sistema. Para que sea aceptado por el sistema debe estar por debajo de ese objetivo. Lo más importante de esto es que es completamente aleatorio para poner a prueba a los mineros.

El objetivo se establece requiriendo a los mineros que consigan un número de ceros (‘0000’) al comienzo del hash. De esta forma, cuantos más ceros se requieran al principio menor será la probabilidad de encontrar un hash válido. El Nonce que genera un hash válido es conocido como *golden nonce* porque tiene atribuida una recompensa.

Por ejemplo, un minero al generar un bloque añadirá unas transacciones (Kirill envía 10 hadcoins a Telepizza), el hash anterior (0000DF2E57FB432A), en un segundo concreto (1519181247) que insertado en el SHA256 obtendrá un hash candidato

(A78D0DF2E57FB432A). Como incumple la condición de empezar por '0000' el minero necesita probar otra vez modificando los datos. Uno de los inputs es el segundo en el que se minó el bloque por lo que cada segundo se genera un nuevo hash automáticamente para un mismo bloque de transacciones. Sin embargo, como no es eficiente para el minero esperar casi un segundo puede manipular el nonce. El nonce tiene un rango de 1 a 4 mil millones por lo que el minero tendrá 4 mil millones de oportunidades para añadir esa transacción a la cadena. En caso de agotar todas esas oportunidades del nonce en menos de segundo como último recurso puede probar con combinaciones de distintas transacciones dentro del bloque candidato hasta que el SHA256 devuelva un hash ganador.

El reto en una red descentralizada P2P se encuentra en saber a quién hacer caso, para ello se establece un protocolo de consenso. Si queremos que este protocolo funcione necesitamos que proteja la red frente a ataques y que si ocurre algún ataque en algún punto de la cadena tenga que realizar la tarea casi imposible de cambiar el resto de bloques en más de la mitad de los computadores.

En este caso nos preguntamos qué pasaría si se produce un ataque al final de la cadena al añadir un bloque malicioso. En una red de este tipo suele haber un lag o retardo entre los nodos que se encuentran muy lejos entre ellos. Podría pasar que dos nodos que se encuentran muy lejos minaran un bloque al mismo tiempo. En este caso la red entraría en conflicto para decidir que bloque se mantiene y cual se desecha para que la cadena pueda seguir creciendo. No se pueden mantener los dos bloques y dividir la recompensa porque los bloques pueden tener transacciones distintas.

Existen muchos tipos de protocolos de consenso como el Proof-of-Work (PoW) o el Proof-of-Stake (PoS). En este caso utilizaremos el Proof-of-Work porque es el que usa bitcoin y es el más común.

Proof-of-Work consiste en complicar la tarea de minado del hash exigiendo un número de "0"s al principio del hash. Esta competición de iteración conlleva una inversión en hardware y electricidad enormes para ser el primero en encontrar el *golden nonce*.

El minero añadirá a su cadena el bloque y recibirá una recompensa, que en Bitcoin son 12.5 bitcoins, y también las tasas asociadas a cada transacción. Así se incentiva a que los mineros jueguen limpio. En el caso de que se rechace su bloque porque han añadido

transacciones maliciosas o cualquier otra cosa fraudulenta no obtendrán la recompensa ni las tasas y no recuperarán la inversión realizada.

Antes de que el bloque minado se propague por todos los nodos se harán una serie de comprobaciones muy rigurosas. En caso de no cumplir con alguna de la larga lista de comprobaciones el bloque será rechazado.

Estos puzles criptográficos se caracterizan por ser difíciles de resolver, pero sencillos de verificar. Partimos del supuesto en que dos mineros minan un bloque exactamente al mismo tiempo, así que tenemos un conflicto en la red con dos cadenas que son diferentes. Asumiendo que fueron correctamente generados empezarán a propagarse por la red mensajes contradictorios entre los nodos.

En este supuesto nos encontramos con cadenas que compiten. Así que lo nodos optarán por esperar hasta que se genere una cadena más larga que la otra. Entonces la cadena que antes genere un bloque será la ganadora y la otra será desechada. También la cadena que tenga una cantidad de nodos con potencia para generar hashes mayor tendrá más probabilidades de ganar.

Cuando el nuevo bloque sea minado el conflicto será resuelto porque prevalecerá la cadena las larga. Consecuentemente, los otros bloques no incluidos serán apartados y reemplazados por la nueva cadena. Esos bloques apartados son llamados *orphaned blocks* o bloques huérfanos y los mineros pierden tanto la recompensa como las tasas.

1.2. Banco Central

El banco central es la entidad que posee el monopolio de la producción y distribución del dinero oficial en una nación o bloque de países. A su vez, es la institución que dicta la política monetaria para regular la oferta de dinero en la economía.

En otras palabras, el banco central emite los billetes y monedas que luego llegan a los consumidores. Además, utiliza diversos instrumentos para controlar la cantidad de dinero que circula en el mercado.

En general, el banco central es una institución financiera que tiene la responsabilidad de supervisar y controlar el funcionamiento del sistema financiero. Y, de forma más específica, regular la cantidad de dinero que existe en circulación.

Las principales características del banco central son:

Es un ente independiente del poder político. Por esa razón, sus decisiones no dependen directamente del gobierno de turno, sino de un directorio. Este órgano, sin embargo, es designado en ocasiones por otra institución como el parlamento, por lo que siempre hay posibilidad de injerencia política.

Sigue los mandatos de sus estatutos. Por ejemplo, mantener la inflación anual entre 1% y 3% (Eurosistema). Estas metas se establecen desde el Estado, y deberían perdurar en el largo plazo, aunque cambien las autoridades en el poder.

En los últimos tiempos, han tenido un rol clave para enfrentar las crisis económicas. Por ejemplo, la Reserva Federal en Estados Unidos implementó entre el 2010 y el 2011 un plan de estímulo cuantitativo. Este consistía en comprar bonos del gobierno por 600 mil millones de dólares para inyectar liquidez al sistema (elEconomista, 2010).

1.3. Coins y stablecoins

Una moneda digital es básicamente dinero digital. Esta concepción rompe totalmente con la noción más tradicional del dinero donde el dinero era un título valor que representaba algo físico y tangible guardado en el banco. De esta forma entregando un billete al banco recibías lo que representaba en oro. Esta nueva concepción implica que no hay monedas físicas ni billetes físicos.

A día de hoy, las monedas digitales por excelencia son bitcoin y ethereum. Los inversores las catalogan como activos de elevadísimo riesgo debido a su gran volatilidad y por la falta de garantías que ofrecen. Los más grandes fondos de riesgo como BlackRock están comenzando a incluir en sus carteras estos activos o empresas comerciales como Tesla que tiene en caja 1.5 B $ (USD/BTC 38500$) en bitcoins (Francia, 2021).

El coin y el blockchain son conceptos distintos. Un blockchain es la cadena de bloques donde se realizan transacciones y el coin es la moneda en la que se realizan los intercambios. De esta forma, en un blockchain puede haber más de un coin, pero un coin solo puede pertenecer a un blockchain. Por ejemplo, en el blockchain de Ethereum encontramos coins como Binance Coin, SushiSwap, ETH, …

Un coin representa un proyecto llevado a cabo por sus fundadores y su precio variará en función de la confianza y las expectativas que haya sobre el proyecto. En general a aquellos coins distintos de bitcoin se les llama altcoin. Por ejemplo, Stellar es un proyecto sin ánimo de lucro que pretende hacer asequibles las finanzas eliminando la barrera del

cambio de divisas entre países sobre todo del tercer mundo (Stellar.org, 2021). El proyecto parece muy prometedor ya que ha alcanzado una capitalización de mercado de 9 B$ con un precio de 0,30$ por moneda.

Las criptomonedas estables o también conocidas como stablecoins nacen para tratar de eliminar o reducir la volatilidad de monedas digitales. Aquellas personas o empresas que no toleran ese riesgo porque se encuentra en contra de sus intereses acaban descartando esas monedas digitales. Ante esta problemática, surgen las stablecoin que están asociadas al valor de una moneda fiat como el dólar o euro, bienes materiales como el oro u otras monedas digitales. Al mismo tiempo hay stablecoins que no se asocian a otras criptomonedas y que están controladas por algoritmos para eliminar la volatilidad del precio. De estas formas, la necesidad principal que cubre una stablecoin es dar refugio a los inversores en momentos de volatilidad o permitir realizar transacciones con precios predecibles a corto y largo plazo. De esta forma, los fundadores del coin pretenden recibir financiación para su proyecto mediante dos vías. Por un lado, mediante la emisión inicial de monedas se venden a un precio fijo una cantidad de monedas. Por otro lado, a medida que el proyecto crece y se especula con las monedas en mercados secundarios el precio de estas sube haciendo que las monedas guardadas por los fundadores suban y puedan así autofinanciar el proyecto.

Existen stablecoins independientes, pero muchas veces funcionan como complemento para un proyecto ya consolidado. Siguiendo con el ejemplo de Stellar, en febrero de 2021 el proyecto anuncio una stablecoin llamada USDC para abarcar un mayor mercado y que así su comunidad no dependiera de terceros (Stellar.org, 2021).

En resumen, encontramos dos estrategias distintas para crear un stablecoin:

- Las estrategias colateralizadas asociadas a otro valor externo para aportar esa estabilidad. Por ejemplo, G-Coin, una plataforma de monedas que equivalen a un gramo de oro físico cada uno. La compañía asegura que el oro está almacenado de forma segura y que emplean blockchain para garantizar que el material procede de zonas libres de conflicto. Según la firma, sus monedas pueden intercambiarse por oro físico, emplearse como depósito de valor o usarse como otras criptomonedas para realizar determinados pagos digitales (G-Coin, 10).
- Las estrategias no colateralizadas que no asocian su valor a ningún activo externo, obedecen a un algoritmo para evitar las fluctuaciones del precio. Por ejemplo,

Basis contaba con el respaldo, además de los smart contracts, de un conjunto de algoritmos que replicaba la política monetaria de un banco central (Basis, 2021).

Las stablecoins se encuentran aun empezando a manifestarse y aún no ofrecen garantías suficientes como vías de inversión en muchos casos. Pero como su utilidad es tan grande con el paso del tiempo y el perfeccionamiento de sus diferentes modelos podrían llegar a reunir un mayor capital que el que mueven ahora los activos volátiles como bitcoin.

1.4. Inteligencia artificial y Machine learning

La Inteligencia artificial (IA) es entendida como la combinación de algoritmos con el propósito de crear maquinas que tomen decisiones de una forma inteligente y no sistemática. Su capacidad de aprender es tan flexible que se emplea para automatizar procesos en finanzas, educación, comercial, sanidad, logísticas, agricultura y medio ambiente. Esta tecnología se encuentra en pleno auge y se estima que el mercado de la Inteligencia artificial pueda llegar a representar 15,7 T$ de dólares en el mundo en 2030 (Rao, 2017).

El Machine learning (ML) o aprendizaje automático es una de las ramas de la inteligencia artificial que permite que las maquinas aprendan a resolver un problema sin que sean expresamente programadas para ello. El Machine learning se basa en extraer patrones a partir de un conjunto de datos dado. Un conjunto de datos está formado por observaciones que a su vez están compuestas por características. Esta tecnología se encuentra presente en muchísimas aplicaciones como el recomendador de series en Netflix o para asistentes de voz como Siri.

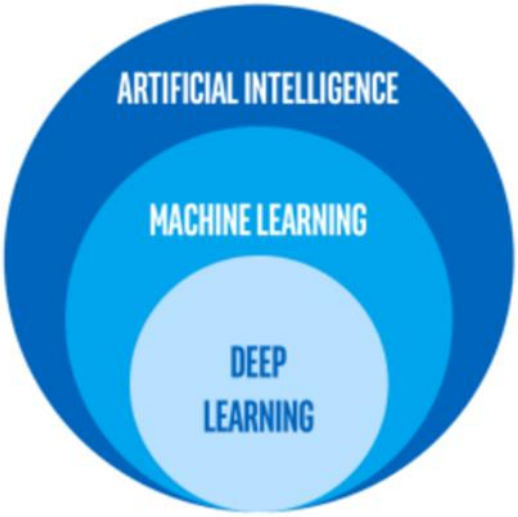

Figura 3: Ilustración de los campos de la IA

Dentro de los algoritmos de Machine learning encontramos 3 tipos de formas de aprender:

Aprendizaje supervisado, al algoritmo se le proporciona una serie de observaciones etiquetadas. Los datos serán la entrada del algoritmo y la etiqueta lo qué debe predecir. Sabiendo qué ha de predecir se dedicará a encontrar patrones. Por ejemplo, se podrían proporcionar fotos de ropa e indicar en cada foto de qué prenda se trata.

Aprendizaje no supervisado, al algoritmo no se le proporcionan las etiquetas de las observaciones. En este caso, tratará de encontrar similitudes y agrupar los datos. Por ejemplo, en el reconocimiento facial el software no busca un rostro, sino una serie de patrones que indique que se encuentra ante el mismo rostro.

Aprendizaje por refuerzo, en este caso el algoritmo aprende por prueba y error hasta que consigue resolver la tarea encomendada. Cuando el software realiza una tarea bien recibe una recompensa, pero cuando realiza alguna acción que le aleje del objetivo recibirá un castigo.

1.5. Deep learning y redes neuronales

El Deep learning (DL) o aprendizaje profundo es una rama del Machine learning que intenta abstraer de un conjunto de datos patrones para alcanzar un entendimiento de la realidad que representa dichos datos. La clave de éxito del Deep learning es que está basado en redes neuronales. Las redes neuronales son algoritmos que intentan replicar el funcionamiento del cerebro humano. El cerebro tiene neuronas interconectadas con

dendritas que reciben impulsos, en función de los impulsos recibidos producen una señal de salida a través de su axón. Las redes neuronales no resuelven los problemas de una forma sistemática, sino que evolucionan iterativamente para abordar el problema de una forma inteligente. En comparación con el aprendizaje humano estos algoritmos no hay que indicarles que deben aprender para que aprendan. Por ejemplo, si queremos que aprenda a identificar mascotas en imágenes solo hay que proporcionarle muchas imágenes de mascotas, mientras que a una persona habría que indicarle en que ubicación de la foto se encuentra la mascota. Por otro lado, podemos encontrar problemas que son muy fáciles para las máquinas y muy complejos para las personas y viceversa. En la siguiente figura vemos una arquitectura bastante simple de red neuronal donde toma como entradas los m2, dormitorios, distancia de la ciudad y edad para inferir el precio de una vivienda.

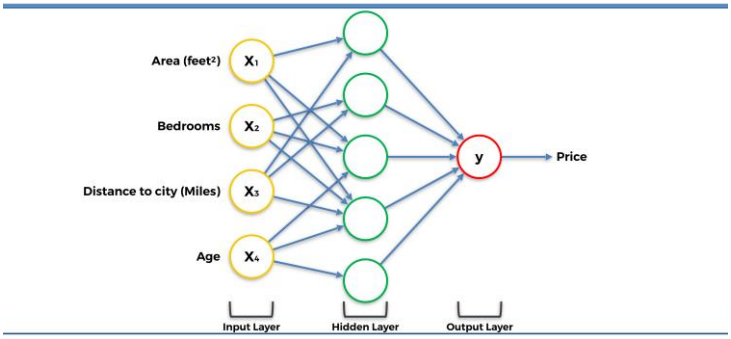

Figura 4: Arquitectura de red neuronal

Esta técnica fue inventada por Frank Rosenblatt en 1957 (Loiseau, 2019). Sin embargo, el coste de computación era demasiado alto para aquella época y los estudios académicos se estancaron. Gordon E. Moore en 1965 estableció la Ley de Moore que afirmaba que el poder de computación de los ordenadores se duplicaría cada 2 años (Moore, 1965). De esta forma, los ordenadores han conseguido alcanzar un poder de computación adecuado, aunque no óptimo para entrenar redes neuronales. Como vemos el concepto de red neuronal puede sonar muy nuevo, pero realmente tiene casi 100 años.

Las redes neuronales se han convertido en el cerebro de las inteligencias artificiales de hoy en día.

1.5.1. Capa totalmente conectada

La más sencilla es la capa completamente conectada. Esta es tan sólo una capa de red neural normal en la que todos los resultados de la capa anterior están conectados a todos los nodos de la capa siguiente.

1.5.2. Función de activación: ReLU

La función de activación lineal rectificada o ReLU devolverá como salida los valores de entrada, pero anulando los valores negativos. Esta función permite a la neurona quedarse únicamente con las correlaciones positivas.

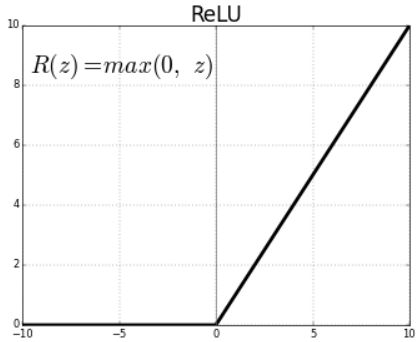

Figura 5: Gráfica de la función ReLU

En comparación con otras funciones, la ReLU permite a la red neuronal converger más rápido (Agarap, 2019).

$$f(x) = \max(0, x)$$

1.5.3. Función de activación: Sigmoide

La función sigmoide devuelve la informacion de entrada resumida en un valor entre 0 y 1. De esta forma al introducirla en la salida de nuestro modelo nos indicará para cada accion que tan conveniente es.

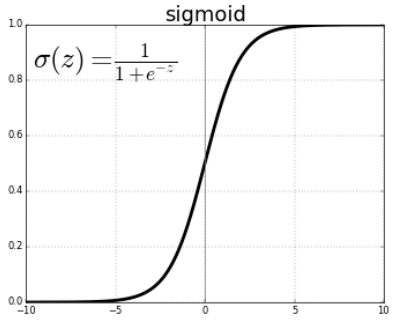

Figura 6: Gráfica de la función sigmoide

Al estar entre 0 y 1 nos permite saber que tan segura está de sus decisiones ya que si por ejemplo devuelve valores cercanos al 0,5 signifaca que no sabe que hacer, mientras que si el valor es cercano a 0 sabemos que es mala idea emplear la accion que representa esa neurona y en caso de ser cercana a 1 es que tiene muy claro que esa acción dara un resultado bueno según su política. Aun así, puede que más de una acción sea cercana a uno dando a entender que hay mas de una acción conveniente.

$$f(x) = \frac{1}{1 - e^{-x}}$$

1.5.4. Capa de Dropout

El Dropout es una técnica de regularización que evita el sobreajuste. Simplemente consiste en desactivar una cierta proporción de neuronas aleatorias durante cada paso de propagación hacia adelante y hacia atrás. De esa manera, no todas las neuronas aprenden de la misma manera, evitando así que la red neuronal sobreajuste los datos de entrenamiento (Hinton, 2014).

Activaremos el Dropout en la primera capa oculta, con una proporción de 0.1, lo que significa que el 10% de las neuronas se desactivarán aleatoriamente durante el entrenamiento a cada iteración.

1.5.5. Optimizador: Adam

Adam es la abreviatura de *Adaptative Moment Estimation*. Este optimizador para calcular el ratio de aprendizaje calcula un combinación lineal entre el gradiente y el momento anterior. Es un optimizador que se adapta muy bien a las redes neuronales en general y por ello es por lo que ha sido elegido en nuestro modelo (Kingma, 2017).

$$m = \beta 1 * m - (1 - \beta 1) * \Delta W$$

$$v = \beta 2 * v + (1 - \beta 2) * \Delta W^2$$

$$W = W - \frac{\alpha * m}{(v + \varepsilon)^{\wedge}(\frac{1}{2})}$$

1.6. Reinforcement Learning

El Reinforcement Learning es de los distintos tipos de aprendizaje probablemente el más interesante y el más complejo porque precisa de la base de los aprendizajes anteriores. La inteligencia realiza observaciones dentro de un entorno informático y su objetivo es el de maximizar las recompensas posibles a largo plazo. La inteligencia aprenderá de sus errores y optará por unas estrategias u otras dependiendo de la política de recompensas. Además, la inteligencia no guardar información sobre la arquitectura del entorno, sino que el propio entorno moldeará la personalidad de la inteligencia para que se desenvuelva de forma natural en él.

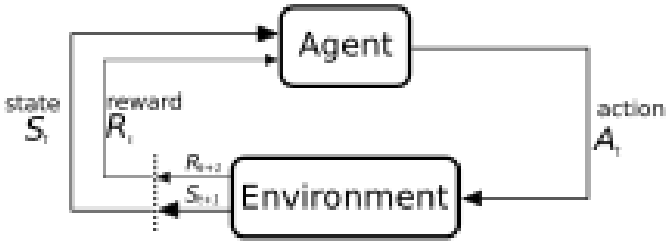

Figura 7: Interacción de la IA

En la Figura 5 podemos observar el ciclo en el que el agente lleva a cabo una acción en el entorno. Este le devuelve un *feedback* en forma de estado y recompensa causando que el agente lleve a cabo una interacción nueva.

1.6.1. Ecuación de Bellman

La ecuación de Bellman es la base lógica del aprendizaje de nuestra inteligencia artificial. La ecuación fue ideada por Richard Ernest Bellman que era un matemático que definió el concepto de programación dinámica, la cual consiste en resolver un problema a partir de subproblemas más pequeños (Bellman, 1964). Los conceptos básicos que vamos a utilizar son:

- ‘s’ para definir el estado. El agente puede encontrarse en distintos estados dentro del entorno. Los cuales cambian solos o como consecuencia de acciones.

- ‘a’ para definir acción. Sera la acción que puede llevar a cabo el agente en cada momento. El agente tendrá una lista de acciones a su disposición y cada acción lleva a cabo un cambio de estado.

- ‘R’ para definir la recompensa, el agente según los estados puede obtener recompensas positivas o negativas.

- γ para definir el factor de descuento.

$$V(s) = \max(R(s,a) + \mathrm{V(s')})$$

Aquí podemos observar el valor en cierto estado que es el valor que nos devuelve el estado en un entorno concreto V(s), vemos la “a” de la acción que hemos tomado y la s’ que representa el estado en el que nos vamos a encontrar justo después del estado en el que nos encontramos tras tomar la acción. El agente tiene un abanico de acciones por lo establece un máximo para escoger la acción que maximice el valor. De todas las acciones se toma la que es más favorable. Como resultado se generará inmediatamente una recompensa que puede ser positiva, negativa o nula.

Ese valor nuevo se suma al valor antiguo multiplicado por el factor de descuento. El punto es que de la recompensa que tengo ahora mismo más lo que se puede ganar al tomar una acción de todas las posibles se escogerán la que maximice esa suma. Escogiendo la acción que de un mayor valor más el mayor valor al cambiar de estado escogerá la que maximice la recompensa más el valor en el nuevo estado multiplicado por el factor de descuento.

El factor de descuento sirve para dirigir al agente y que el estado actual no tenga el mismo valor que el siguiente y así no pierda la dirección del camino hacia la recompensa. De esta forma, cuanto más lejos estemos menor será la recompensa y cuanto más nos acerquemos la recompensa será mayor. Por ejemplo, de manera similar en el mundo de las finanzas 5€ a día de hoy vale más que 5€ en el futuro. Sin embargo, podría pasar que ese dinero se invirtiera y generara un interés. En las finanzas el dinero incrementa con el paso del tiempo y el factor de descuento implica que cuanto más cerca estoy del objetivo mayor es el incremento del valor.

Únicamente con lo que hemos mencionado anteriormente el agente aprendería una estrategia para llegar al objetivo, pero es posible que también haya otras rutas igual de

buenas o mejores. Precisamente para evitar que no descubra varias estrategias tenemos que hacer que la exploración sea no determinista. Introduciendo un modelo estocástico en vez de escoger el 100% de las veces la acción que maximice el valor habrá un factor de exploración que hará que haya una probabilidad de escoger una acción aleatoria, lo que lo hace mucho más parecido al comportamiento de una persona frente a un robot que toma el 100% de las veces las mismas decisiones. Por ejemplo, cuando vamos por la calle no siempre vamos a un destino por la misma ruta, sino que en función del tráfico y otros factores adaptamos nuestra ruta. La propiedad de Markov implica que esta aleatoriedad no está influenciada por los estados pasados ni futuros sino únicamente por el estado actual. En este caso las decisiones del agente serán parcialmente aleatorias ya que la mayoría de las veces tomará la decisión que maximice el valor, pero a veces actuará de forma aleatoria. Añadir esta propiedad hará que la ecuación de Bellman sea un poco más sofisticada.

Ahora en vez de tener V(s') se calculará por la media ponderada por la probabilidad del valor de todos los posibles estados futuros.

$$V(s') = p1 * V(s1') + p2 * V(s2') + \cdots + pn * V(sn')$$

De esta forma al sustituir V(s') por el sumatorio anterior la ecuación de Bellman sería la siguiente:

$$V(s) = \max(R(s,a) + \gamma \Sigma P(s,a,s')V(s'))$$

Aunque parezca que lo más sensato sería que nuestra inteligencia actuara de forma determinista en la vida real no es así. Por ejemplo, en las finanzas lo lógico sería poner todo el capital en la mejor inversión sin embargo los inversores diversifican para reducir el riesgo ya que lo que parecía la mejor decisión puede terminar no siéndolo.

1.6.2. Q-Learning

Ahora en lugar de trabajar con valores en un estado sustituiremos la V por la Q valorando el estado y la acción llevada a cabo. La Q viene de *quality* ya que ese valor representará la calidad de qué tan buena es la acción correspondiente. La diferencia esencial es que el valor V representaba el valor del estado actual mientras que la Q propone el valor de que tan buena es la decisión que voy a tomar.

Q se definiría como:

$$Q(s,a) = R(s,a) + \gamma\, \Sigma P(s,a,s')\max(Q(s',a'))$$

De esta forma lo que hacemos es evaluar de forma individual cada acción a tomar. Antes, el estado futuro dependía única y exclusivamente en el estado en el que me encontraba por lo que al llegar a un estado automáticamente sabía cuál era el siguiente. Ahora, partiendo del estado en el que se encuentra pondera la calidad de las acciones para tomar la acción más inteligente. Otra diferencia es que ahora el agente no necesitará moverse al estado siguiente para conocer el valor como sucedía con la V(s), sino que evaluará la calidad Q(s,a) desde el estado en el que se encuentra.

Además, el Q-Learning es más eficiente porque en lugar de estudiar todos los posibles estados únicamente se tienen en cuenta las acciones posibles. Actuando así de forma reiterativa el agente irá adquiriendo información y a base de repetir el proceso la recompensa ponderada y la calidad de las misma convertirán la ecuación en una herramienta muy útil para que el agente vaya aprendiendo de las decisiones que vaya tomando.

Una vez cambiado al siguiente estado el agente conoce la calidad del estado en el que se encuentra frente a la calidad estimada. El agente ha sido capaz de estimar la Q porque tenía información o había pasado antes por ese estado y una vez cambiado de estado sabe el valor real de esa Q. Al haber estocasticidad o aleatoriedad en el proceso es posible que la Q predicha y la Q real no coincidan. A esta diferencia la llamamos la diferencia temporal y es ella la que permite que el agente aprenda a cada iteración (Sutton, 1988).

$$TD(a,s) = R(s,a) + \gamma\, \Sigma P(s,a,s') \max\big(Q(s',a')\big) - Q(s,a)$$

Entonces calculando lo que sabíamos antes y la nueva información podemos cuantificar cuanto ha cambiado de antes a ahora. Por lo tanto, podemos redefinir el valor Q como la calidad de una acción tomada a partir de un estado se va a actualizar con la informa previa más un α que será el ratio de aprendizaje acerca de la diferencia temporal donde Q_{t-1} será el valor predicho y Q_t el valor real.

$$Qt(s,a) = Qt - 1(s,a) + \alpha TDt(a,s)$$

A modo de concluir este apartado, calculando diferencias temporales y actualizando a cada paso los valores Q el proceso terminará convergiendo y el agente aprenderá todo lo que puede llevar a cabo dentro del entorno.

1.6.3. Algoritmo de Deep Q-Learning

Como vemos nuestro agente de inteligencia artificial comenzará a interactuar con el entorno que hemos preparado anteriormente donde en función del número de monedas que aumente o disminuya tendrá un efecto en el precio de la moneda. El agente observará cómo reacciona el entorno y tomará decisiones para entender que efectos produce.

El Deep Q-Learning lleva la ecuación de Bellman al siguiente nivel. En lugar de realizar calculos básicos dentro del entorno suministraremos la informacion de los estados a una red neuronal que nos devolverá los valores Q (Juliani, 2016).

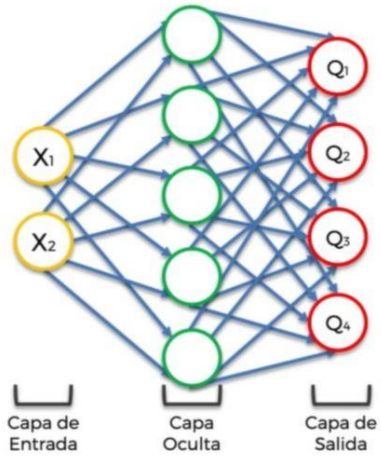

Figura 8: Red neuronal en Deep Q-Learning

En la imagen podemos ver como X1,X2,…,Xn en nuestro caso sería la capitalización de mercado, el volumen de transacciones, la cantidad de monedas en cirulación y el precio actual de la moneda. Estas entradas pasan por la capa oculta capa a capa donde aprenden las neuronas y detectan patrones. Finalmente devolverá los valores Q de las 7 acciones donde elegirá el mayor de todos ellos. La ventaja de aplicar redes neuronales es que el entorno ya no tiene porque ser matricial para aplicar la ecuacion de Bellman, sino que se puede dar el salto a entornos muchos más complejos donde las decisiones no son tan simples. Por ejemplo, en el caso de los coches autonomos de Google que perciben con sensores su alrededor para conducir de forma segura. Anteriormente, vimos que cada vez que pasaba el agente por un estado se almacenaba ese último valor haciendo que hubiera un Q anterior y otro Q posterior para más tarde realizar la diferencia temporal. Al usar redes neuronales no se almacena un Q anterior sino que este de cierta forma se encuentra

implícito en los propios pesos de la red neuronal. Podria decirse que la red se acuerda de los valores Q anteriores. Asi que la diferencia temporal se calcula por el error de la predicción que conocemos al tomar la acción.

Como la salida de la red neuronal seran tantos valores Q como número de acciones tengamos, para escoger el Q máximo añadiremos una función de softmax a la red. Esta funcion descarta los valores Q mas pequeños y se queda con la que tenga la Q más alta.

En resumen, asi es como se sumintra un problema de Q-Learning a una red neuronal. Tenemos un primer vector que describe el estado en el que se encuentra el agente. Luego la informacion pasa capa a capa y nos devuelve los valores Q. En la fase de aprendizaje se calcula error del Q predicho con los Q reales y se ajustan los pesos hacia atrás. Todo esto mientras el agente va explorando el entorno. Finalmente, se realiza la selección de accion con la función softmax.

1.6.4. Experiencia de repetición

Para optimizar su aprendizaje emplearemos la técnica Deep Q-Learning. Esta técnica desarrollada por DeepMind, departamento de inteligencia artificial de Google, dota de memoria a la inteligencia artificial mediante lo que denominaron *experience replay* o experiencia de rejugado (Tom Schaul, 2016). Esto se debe a que los estados anteriores al actual se encuentran fuertemente correlacionados con los siguientes. En principio no es necesario que se incorpore al algoritmo, pero añadirlo mejora los resultados notablemente a la vez que hace a la inteligencia más completa.

Por ejemplo, si un F1 se dispone a tomar una curva pronunciada después de una recta de máxima velocidad con la experiencia de rejugado será capaz de frenar antes de entrar en la siguiente curva y no se saldrá del circuito. El agente sin la experiencia de repetición durante el entrenamiento aprenderá que la mejor estrategia es seguir recto porque el circuito empieza con una recta. Así, al llegar la primera curva el agente tenderá a estar muy sesgado a continuar recto y tendrá muchos problemas a la hora de no salirse de la curva. Este mecanismo hace que la inteligencia artificial se inspire en una muestra aleatoria de las acciones previas que ha tomado en vez de utilizar únicamente la acción más reciente para continuar al siguiente estado. De esta forma la IA será capaz de anticiparse y reaccionar mejor ante sucesos poco normales porque está usando la memoria para recordar cómo reaccionó a situaciones a las que ya se ha enfrentado. Las experiencias se suministran en bloque a la red neuronal de modo que se otorga la recompensa en bloque

a toda la experiencia. Por ejemplo, una experiencia podría ser aprender a coger curvas a derechas y una vez aprendido a tomar la primera curva a derechas, como por lo general coger curvas a derechas suele ser muy similar, reciclará esa experiencia para la siguiente curva a derechas a la que se enfrente. El aprendizaje por experiencias en lugar del aprendizaje estado a estado es mucho más eficiente y logra mejores resultados porque mientras uno es muy cortoplacista la experiencia de rejugado permite resolver el problema al que se enfrenta con perspectiva.

1.6.5. Entorno

Los agentes deben tener un entorno donde aprender y adaptarse. Un entorno informático no es más que una representación fiel de la realidad. Cuanto mejor sea la representación más fiable será la inteligencia que entrenemos en él.

Un entorno es observable si se puede obtener una información completa, correcta y actualizada en cada instante de tiempo. Por lo tanto, cuanto más observable sea un entorno más fácil será construir un agente en él. En nuestro caso, los mercados de criptomonedas están participados por personas y no por máquinas por lo que hay un factor muy importante de comportamientos irracionales.

Un entorno es determinista si una acción tiene un único efecto sobre el entorno. Es decir, el siguiente estado estará determinado por la acción actual y el estado actual.

Un entorno es estático si este únicamente cambia con acciones del agente. En nuestro caso el mercado de criptomonedas cambiará diariamente con independencia de la inteligencia.

1.6.6. Agente

Un agente de inteligencia artificial es un software informático que toma decisiones inteligentes. El agente es capaz de interactuar con un entorno y predecir los efectos que tendrán sus acciones. Un agente actúa de la siguiente manera. Primero recibe la información del entorno y toma una decisión. Después de decidir ejecuta la acción que causará un efecto en el entorno. Finalmente, las consecuencias de esa acción se traducirán en recompensas donde el agente siempre elegirá la opción que maximice esa recompensa.

1.6.7. Acción

El agente posee un conjunto de acciones para interactuar con el entorno y modificar el estado en el que se encuentra. Una acción se entiende como un comportamiento predefinido del agente. Las acciones pueden ser continuas o discretas. Las acciones continuas implican que el conjunto de acciones posibles sea infinito mientras que las acciones discretas están acotadas. Escoger entre un tipo de acción u otro depende del problema que se quiera resolver.

1.6.8. Recompensa

La recompensa es la forma de indicar al agente de inteligencia artificial si está realizando bien una tarea. En caso de alcanzar el objetivo recibirá recompensas positivas mientras que en caso de alejarse de este recibirá un castigo. A lo largo de las simulaciones el objetivo del agente siempre será maximizar el número de recompensas conseguidas o minimizar el número de castigos según la política de recompensas que se establezca. Una política muy negativa puede ocasionar que el agente se olvide del objetivo.

Capítulo 2: Diseño del altcoin vinculado al USD

En este capítulo aplicaremos la teoría anterior para diseñar una inteligencia artificial que pueda tener encaje en un blockchain y resuelva la tarea de estabilizar el precio de una criptomoneda de forma exitosa. Debemos tener en cuenta que el mercado de criptomonedas es el más volátil y arriesgado del mundo por lo que la inteligencia a diseñar debe ser muy robusta. Además, de la implementación básica se expondrá como se han abordado preguntas y se han tomado decisiones para obtener la mejor configuración del agente. Para ello seguiremos el siguiente orden:

1. Construir el entorno.
2. Construir el cerebro del agente.
3. Entrenamiento de la IA.
4. Testeo de la IA.
5. Interacción de la inteligencia propuesta en el blockchain.

2.1. Construir el entorno

Como se ha expuesto en el marco teórico, precisamos de un entorno que recree de la forma más fiel posible la realidad a la que se enfrentará nuestro agente. Primero examinaremos la realidad a la que se enfrentaría y segundo se explicará la estrategia que se ha seguido para su recreación informática.

2.1.1. Realidad de un mercado de criptomonedas

El valor de una criptomoneda viene fijado por su precio de mercado, por lo que es el mercado el que lo fija. A su vez, el mercado sigue la ley de la oferta y la demanda que viene a decir que un incremento en la oferta reducirá es el precio del activo y un incremento en la demanda aumentará el precio del activo. La demanda será proporcional al interés que tenga una comunidad en adquirir el activo mientras que la oferta dependerá del volumen del activo que haya en circulación. De esta forma cuando oferta y demanda coinciden se genera un equilibrio denominado precio de mercado.

La oferta de monedas es intrínseca al blockchain por lo que el número de monedas en circulación se conoce de primera mano ya que lo fijamos nosotros mismos al programar el blockchain.

En cuanto a la demanda de monedas esta se conoce a través del libro de órdenes donde los usuarios de una comunidad publican ofertas de compra y venta. Cuando la oferta de

un comprador y el anuncio de un vendedor coinciden se produce un equilibrio en el mercado dando lugar a una transacción a un cambio acordado en USD. Esa transacción se denomina tick.

2.1.2. Simulación de un mercado de criptomonedas

La lógica del punto anterior es lo que debemos tratar de comprender y plasmar en el entorno informático.

El libro de órdenes se puede estudiar en tiempo real (tick a tick) o también puede resumirse su actividad en segundos, minutos, horas o días. En nuestro caso resumiremos actividad del libro de órdenes en días. Las variables que podemos obtener son casi infinitas pero la información que nos interesa en relación a la demanda es el volumen de transacciones diario y la capitalización de mercado.

El volumen de operaciones es interesante porque representa la actividad del mercado, es decir, si se están dando muchos intercambios en relación con el precio de la moneda.

La capitalización es esencial ya que representa el valor del mercado total de la moneda. Esta cifra se calcula conociendo el precio al que se intercambia la moneda por el total de monedas en circulación.

Sabiendo esto, el entorno tomará los datos históricos de Bitcoin por ser el blockchain y moneda por excelencia. Realmente la moneda no es relevante porque podría haberse usado cualquier otra criptomoneda. De este histórico de datos el entorno simulará de forma diaria la capitalización de mercado y volumen de transacciones de Bitcoin y en función del número de monedas que haya en circulación establecerá constantemente un equilibrio en forma de precio de mercado respetando la lógica de la ley de la oferta y la demanda como en los mercados reales.

Como veíamos el cálculo de la capitalización de mercado se conocía realizando el número de monedas en circulación por su valor de mercado.

$$Capitalización\ de\ mercado = Precio * Nº\ de\ monedas\ en\ circulación$$

De esta relación se puede deducir que si tenemos los datos la capitalización de mercado en un momento concreto y además conocemos el número de monedas en circulación podemos conocer el precio.

$$Precio = \frac{Capitalización\ de\ mercado}{Nº\ de\ monedas\ en\ circulación}$$

Con esta relación el entorno responderá en todo momento con el precio.

2.1.3. Validación del entorno

En este apartado estudiaremos los desplazamientos de la capitalización de mercado cuando la cantidad de monedas es estable y viceversa con el propósito de comprobar que el entorno responde de la forma esperada antes de que la inteligencia interactúe con él.

Desplazamientos de la demanda

A continuación, se muestran los gráficos de la capitalización de mercado con tendencia alcista para un número de monedas en circulación constante.

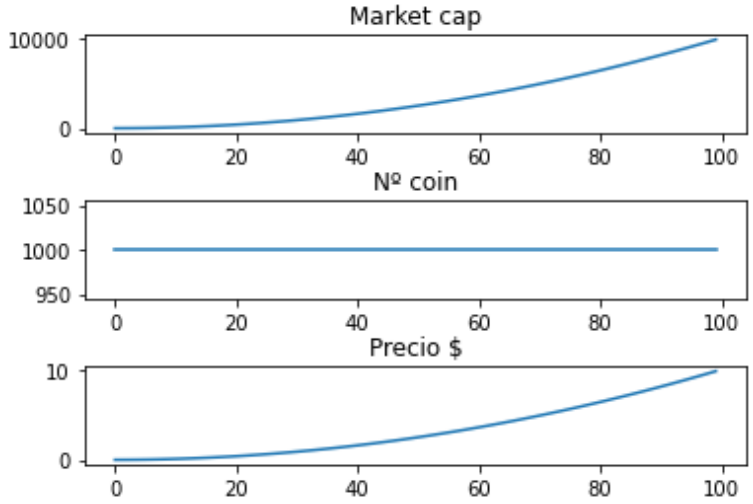

Figura 9: Validación del entorno 1

Como podemos ver ante una oferta constante de monedas y un incremento de la demanda el precio aumenta.

Desplazamientos de la oferta

A continuación, se muestran los gráficos de la capitalización de mercado con tendencia lateral para un número de monedas en incremental.

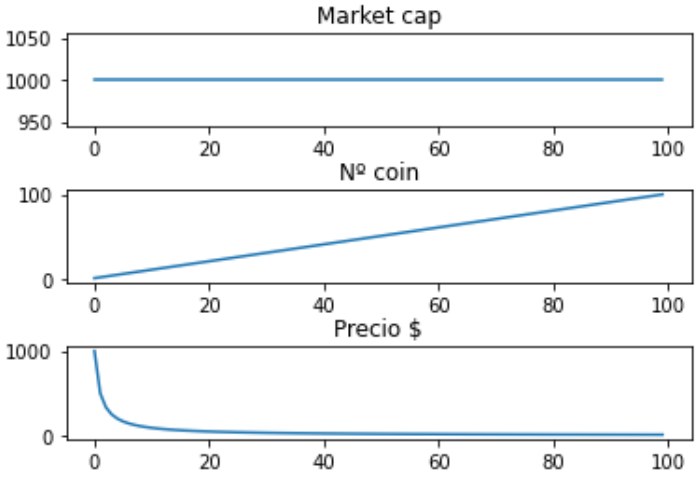

Figura 10: Validación del entorno 2

Como podemos ver ante una oferta incremental de monedas y una capitalización de mercado contante el precio disminuye.

Conclusión

Observando las gráficas hemos comprobado que el entorno respeta la lógica de un mercado y afirmamos que el precio baja cuando tiene que bajar y sube cuando tiene que subir. Por lo que este entorno es viable para cualquier agente que se quiera entrenar.

Otra cosa que vemos es que la relación del precio con el número de monedas tiende a cero en la medida en que las monedas aumentan. Esto se debe a que por mucho que aumente el número de monedas el precio decrecerá, pero nunca llegará a ser cero o negativo. Esta observación es muy importante ya que puede afectar a la forma de actuar del agente que se entrene en este entorno. Por otro lado, la relación del precio con la capitalización del mercado es lineal.

2.1.4. Definir los estados

Los estados indican el contexto del entorno en el que se encuentra el agente. Los estados de este blockchain son:

- Capitalización de mercado, representa el valor de mercado en USD que tiene todo el conjunto de monedas en circulación. Es importante el matiz de que se encuentren en circulación ya que los fundadores de una criptomoneda suelen guardar monedas para financiar el proyecto que haya detrás de dicha moneda.

- Volumen de transacciones, el volumen de transacciones corresponde con el número total en USD de monedas que han sido intercambiadas en un día.

- Nº de monedas, corresponde con la liquidez del mercado.

- Precio actual de la moneda, será el precio de cierre del día indicado en USD.

Los estados además de informarnos de cómo se encuentra el entorno también servirán al agente como fuente de conocimiento para tomar decisiones que puedan afectar al entorno.

2.1.5. Definir las acciones

Las acciones son simplemente los cambios en el número de monedas que hay en circulación en el blockchain para incrementar o disminuir el precio de la moneda. Para que nuestras acciones sean discretas, consideraremos 7 posibles variaciones en la cantidad de monedas en circulación.

La acción estará compuesta por tres partes:

Primero, el agente elegirá el signo de la variación de monedas en función del precio actual. De ser el precio en *t* superior al precio óptimo, el signo de la variación será positivo. En caso contrario el signo será negativo.

Segundo, el entorno calculará la variación de coins sin IA. Haciendo la diferencia porcentual entre el precio óptimo y el precio en el instante *t* multiplicado por el número de monedas en circulación.

$$\Delta coinsinIA = (precio\ actual - precio\ objetivo) * nº\ coins\ en\ circulación$$

Tercero, la IA corregirá el resultado del primer paso aplicando un multiplicador.

Acción	¿Qué hace?
0	ΔcoinsinIA * 0.00
1	ΔcoinsinIA * 0.50
2	ΔcoinsinIA * 0.75
3	ΔcoinsinIA * 1.00
4	ΔcoinsinIA * 1.25
5	ΔcoinsinIA * 1.50
6	ΔcoinsinIA * 1.75

Tabla 1: Catálogo de acciones de la IA

Una vez tomada la acción el agente preguntará al entorno cual ha sido el resultado de su política monetaria. De tal forma que el nuevo precio será igual a la capitalización de mercado entre el número de monedas en circulación más la variación.

$$Precio' = \frac{Capitalización\ de\ mercado}{Nº\ de\ monedas\ en\ circulación + \Delta monedas\ por\ la\ IA}$$

De este nuevo precio calcularemos las recompensas para darle una respuesta al agente. En nuestro caso estudiaremos dos tipos de políticas distintas de recompensas.

El agente 1 únicamente recibirá recompensas negativas.

$$R = -|Precio' - Precio\ objetivo|$$

El agente 2 recibirá recompensas negativas si no consigue mantener el precio entre 0.9 y 1.1 $ y 10 puntos positivos si lo consigue.

$$R = -|Precio' - Precio\ objetivo|$$

R = 10 si

$$0.9 < Precio' < 1$$

A continuación, esa recompensa se acumula al total de recompensas de los días que dure la simulación para aprender en que simulación maximiza las recompensas.

$$Total\ Recompensas = \Sigma R$$

Después de añadirse la recompensa, el entorno se actualizará al día siguiente del blockchain donde variará la capitalización de mercado y el volumen de transacción produciendo que se desajuste de nuevo el precio.

2.2. Cerebro del agente

El cerebro consiste en la capacidad de aprendizaje del agente que viene dada por un modelo de redes neuronales. Esto permite que saquemos partido de todos los avances que se han dado recientemente, pudiendo así tratar con tareas que requieran del análisis de datos.

Para construir el cerebro del agente utilizaremos Tensorflow. TensorFlow es una librería de código abierto para aprendizaje automático desarrollada por Google para satisfacer sus necesidades de construir y entrenar redes neuronales para detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos. Inicialmente iba a ser una librería de uso interno de Google, pero finalmente se hizo disponible para todo el mundo de forma gratuita (Tensorflow).

Layer (type)	Output Shape	Activation
dense_9 (Dense)	(None, 64)	ReLU
dropout_6 (Dropout)	(None, 64)	0
dense_10 (Dense)	(None, 32)	ReLU
dropout_7 (Dropout)	(None, 32)	0
dense_11 (Dense)	(None, 7)	Sigmoid

Total params: 2,567
Trainable params: 2,567
Non-trainable params: 0
Optimizer: Adam
Loss: MSE

Tabla 2: Arquitectura de la red neuronal

El cerebro, o más precisamente la red neuronal profunda, del agente será una red neuronal completamente conectada compuesta de cuatro capas ocultas: la primera con 64 neuronas y función de activación ReLU, la segunda capa de Dropout, la tercera con 32 neuronas y función de activación ReLU y la cuarta capa de Dropout. Como recordatorio, esta red neuronal toma como capa de entrada los estados del entorno y su capa de salida será de 7 neuronas con función de activación sigmoide que devuelve como salidas los valores Q para cada una de las 7 acciones. En último lugar se aplicará a la salida la función softmax

para escoger la neurona correspondiente a la acción que represente valor más alto. Este cerebro artificial se entrenará con una pérdida de "error cuadrático medio" y un optimizador Adam.

Los valores Q son valores entre 0 y 1. Cuanto más cerca este el valor Q de 1 querrá decir que la inteligencia artificial está más segura de ser esa acción la indicada o por el contrario cuanto más cerca este de -1 querrá decir que está segura de que no es la acción indicada. De esta forma teniendo en cuenta todos los valores Q elegirá el mayor de todos ellos.

2.2.1. Función de perdidas: Error cuadrático medio

Se trata de la función más común dentro de los modelos de redes neuronales. Es una forma de medir el error de la predicción en la red neuronal. Esta función de perdidas nos permite evaluar el rendimiento del modelo en la fase de entrenamiento. La particularidad error cuadrático medio o MSE es que eleva al cuadrado las diferencias entre la predicción y el valor real. De esta forma cuanto peor sea la predicción el error será elevado al cuadrado y por lo tanto la penalización será mayor. El objetivo es intentar que el error acumulado sea lo más cercano a cero.

$$Loss = \Sigma(\mathrm{Q}' - \mathrm{Q})^2$$

2.2.2. Experiencia de repetición

Hasta ahora solo hemos tenido en cuenta la transición de un día a otro, por lo que nuestro agente no está aprendiendo mucho al perderse el contexto. Esto podría mejorarse mucho si en lugar de tener en cuenta el día anterior pudiera considerar los últimos m días. A ese paquete de m días es a lo que llamamos experiencia de repetición. En nuestro caso, permitir que el agente tenga memoria le dará la capacidad de reaccionar más adecuadamente ante distintas fases del mercado en los que hay un exceso o déficit de demanda de monedas como cuando una empresa adquiera o venda un gran volumen de monedas. Para que la experiencia de rejugado sea suficiente necesitaremos que guarde en memoria al menos 3000 experiencias de 10 días.

2.2.3. Tasa de aprendizaje

La velocidad de aprendizaje la estableceremos con un parámetro conocido como *learning rate* o tasa de aprendizaje. Este parámetro debe tener valores de entre 0 y 1. Cuanto más próximo sea a 1 más rápido aprenderá la inteligencia artificial y cuanto más próximo sea

a 0 más tardara en converger. Un aprendizaje demasiado rápido puede hacer que la inteligencia nunca encuentre la mejor estrategia. Por el otro lado, un aprendizaje muy lento causaría que la inteligencia se quedará atascada en un mínimo local y conformarse con una estrategia aparentemente buena.

Para resolver el dilema implementaremos un ratio de aprendizaje dinámico, es decir, un ratio que empiece con valores cercanos a 1 para que comience con un aprendizaje superficial (Tomic, 2010). A medida que el agente aprende el ratio ira decreciendo para que se vaya optimizando y el aprendizaje sea detallado. La lógica que sigue es la de la siguiente formula:

$$lr = 0.1 * 0.9^{\frac{1}{n-250}}$$

2.3. Entrenamiento de la IA

La fase de entrenamiento del agente consiste en partir de una inteligencia que no sabe nada y hacer que interactúe con el entorno para obtener información. Para ello, prepararemos un bucle que a cada iteración el agente realizará una interacción con el entorno. El propio entorno reaccionará a las acciones del agente con recompensas en función de su desempeño.

Primero indicaremos al agente que estamos en fase de entrenamiento por lo que su tarea será aprender. Luego inicializaremos el entorno en el momento inicial y le informará de la capitalización de mercado actual, del número de monedas en circulación, el volumen de operaciones y el precio actual. Esa información pasa por la red neuronal del agente que nos devolverá la acción que crea indicada. Esta acción dará como resultado una variación de monedas en el blockchain que afectará al precio de la moneda. El entorno devolverá una recompensa positiva si se encuentra el precio dentro del intervalo marcado 0,9 – 1,1$. El bucle volverá a repetirse, pero en este caso el estado será el del día siguiente, luego el siguiente hasta acabar con los 1000 días que hemos escogido para la simulación. Acabados los 1000 días volverá a inicializarse el entorno al primer día del blockchain y vuelta a empezar. Cada conjunto de 1000 días se le llama época y simularemos 1000 épocas.

2.3.1. Batch Learning

Emplear un entorno, una red neuronal y además pretender que la inteligencia tenga memoria hace que el aprendizaje sea muy pesado computacionalmente para el ordenador. Con motivo de agilizar esta tarea emplearemos el *batch learning* o aprendizaje por lotes. Proveer a la inteligencia de lotes de días en lugar de suministrarse cada día de forma individual permite que la propagación del error se realice menos veces.

En nuestro caso, determinaremos que el tamaño de los lotes sea de 20 días. Ahora en el entrenamiento el agente realizará 20 interacciones con el entorno y las aprenderá todas de una vez, por lo que nos ahorraremos que aprenda y ajuste la red neuronal 19 veces de cada 20.

Además, el *batch learning* nos ayudará a prevenir el sobreajuste del modelo a los datos. Esto se debe a que a la hora de calcular las pérdidas en la fase de aprendizaje se calcula de forma generalizada el error y así en vez de aprender la red cada día lo hará sobre los últimos 20 días.

2.3.2. Mecanismo de detención temprana

Aun así, nos seguimos encontrando con la pregunta de cuando deberíamos dejar de entrenar la inteligencia artificial. Pasarnos de épocas causaría sobreajuste mientras que quedarnos cortos supondría que la inteligencia no ha alcanzado su máximo potencial.

Para no tomar esta decisión de forma arbitraria, estableceremos un mecanismo llamado *early stopping* o detención temprana (Prechelt, 2015). El mecanismo es muy sencillo, la inteligencia guardará en memoria un registro de las recompensas que va obteniendo por cada vez que se enfrenta a la simulación y se reseteará cada vez que consiga una recompensa superior a las anteriores. A la vez habrá un contador, llamado paciencia, que lleve la cuenta del número de simulaciones en las que no ha conseguido mejorar la última mejor recompensa. De esta forma cuando se le acabe la paciencia a la inteligencia detendrá el entrenamiento.

En nuestro caso, la paciencia será de 1000 épocas por lo que en cuanto recorra 1000 días 1000 veces sin mejorar la recompensa la inteligencia habrá terminado su entrenamiento y estará lista para probarla en un blockchain.

2.4. Testeo de la IA

Ahora, de hecho, tenemos que probar el rendimiento de nuestro agente en una situación completamente nueva. Para hacerlo, ejecutaremos una simulación de 200 días distintos a los del entrenamiento, solo en modo de inferencia, lo que significa que no habrá entrenamiento en ningún momento. Nuestra agente solo devolverá predicciones durante 200 días completos de simulación. Luego, gracias al entorno obtendremos al final las diferencias entre el precio real y el precio objetivo, también el porcentaje de días que el agente logró mantener el precio en torno a 1$. En términos de nuestro algoritmo, aquí para la implementación de prueba casi tenemos lo mismo que antes, excepto que esta vez, no tenemos que crear un cerebro ni un objeto modelo DQN, y por supuesto no debemos ejecutar el proceso de Deep Q-Learning durante las épocas de entrenamiento. Sin embargo, tenemos que crear un nuevo entorno, y en lugar de crear un cerebro, cargaremos nuestro cerebro artificial con sus pesos pre-entrenados del entrenamiento anterior que ejecutaremos en el testeo del agente.

2.5. Interacción de la inteligencia propuesta en el blockchain

En último lugar, combinaremos la tecnología blockchain con la Inteligencia artificial. Como explicamos en el marco teórico los bancos centrales añaden o quitan dinero en circulación gracias a la intermediación de los bancos. De la misma forma nuestra inteligencia artificial quitará o añadirá liquidez al blockchain mediante los mineros.

Lo mineros cobran unas tasas a los usuarios que quieren realizar una transacción y cuando generan un bloque cobran esa tasa. La inteligencia artificial modificará esas tasas que percibe el minero para añadir monedas o quitarlas. Siguiendo esta lógica si el mercado valora la moneda del blockchain por debajo de 1$ el minero percibirá menos tasas de las que le correspondían para aumentar el precio. En caso contrario, si el mercado valora la moneda por encima de 1$ la inteligencia artificial aumentará la cantidad de las tasas que percibe el minero para aumentar la liquidez de mercado y así el precio caiga en torno a 1$.

La estrategia que seguirá la inteligencia artificial será siguiente. En un blockchain normalmente las tasas se hacen públicas en cuanto el bloque es minado junto con las transacciones asociadas a esas tasas. En nuestro caso será diferente. Las tasas se harán públicas en un bloque minado diariamente por la propia inteligencia a las 23:59. Para ello, la IA guardará en memoria todas las tasas generadas y no publicadas en ese día. Como

veníamos diciendo la inteligencia artificial ha sido entrenada para estimar qué número de monedas en circulación debe variar para estabilizar su precio. De la forma que se muestra a continuación como calcula en qué proporción afecta la variación a cada transacción del bloque.

$$Ratio\ de\ recompensa = \frac{total\ de\ tasas + \Delta\ monedas}{total\ de\ tasas}$$

Obtenida ese ratio se aplicará al bloque multiplicando cada transacción por ese ratio y se publicarán las recompensas de los mineros.

$$tasa' = tasa * Ratio\ de\ recompensa$$

Lo más interesante de esta estrategia es que los mineros no se ven perjudicados ni beneficiados ya que aunque la cantidad de monedas que reciban por las tasas, sea mayor o menor, el valor también varía en sentido opuesto por lo que los dólares que perciben son los mismos.

Respecto a los usuarios esta estrategia les incentiva a valorar ellos mismo la moneda a 1$. La lógica de la IA penaliza a los usuarios que realicen transacciones a un precio distinto a un 1$. En caso de aceptar una transacción valorando la moneda por encima del USD será cuestión de horas que el precio lo actualice la inteligencia y pierda el usuario esa diferencia. Por ejemplo, si realizo una venta de un cuadro valorado a 100$ por 50 coins valorados a 2$ en cuanto entre la inteligencia el precio del coin se actualizará a 1$ y habré perdido 50$. Por esto mismo es que los usuarios objetivo de este blockchain no son los especuladores ya que la comunidad sabiendo que pagar más de 1$ por moneda es perder dinero no da lugar a la especulación. Sin embargo, en caso de especular con monedas por debajo de 1$ contribuiría a estabilizar la moneda ya que la demanda aumentaría y con ello su valor aproximándose al precio óptimo.

La implementación de la librería básica se encuentra anexada en el Código 6: Librería Blockchain y la implementación de la aplicación web que permite crear e interactuar con el blockchain de formar descentralizada se encuentra anexada en el Código 7: Aplicación Web del Blockchain.

2.6. Resumen del algoritmo de IA

En este apartado se realiza un resumen del trabajo realizado en el diseño de la inteligencia artificial. Sirve para tener una visión general no solo de lo expuesto en este capítulo sino también del código adjuntado como anexo.

Paso 1: Construcción del Entorno

- Paso 1-1: Introducción e inicialización de todos los parámetros y variables del entorno.
- Paso 1-2: Hacer un método que actualice el entorno justo después de que la IA ejecute una acción.
- Paso 1-3: Hacer un método que restablezca el entorno.
- Paso 1-4: Hacer un método que nos proporcione en cualquier momento el estado actual y la última recompensa obtenida.

Paso 2: Construcción del Cerebro

- Paso 2-1: Construir la capa de entrada compuesta de los estados de entrada.
- Paso 2-2: Construir las capas ocultas con un número elegido de estas capas y neuronas dentro de cada una, completamente conectadas a la capa de entrada y entre ellas.
- Paso 2-3: Construir la capa de salida, completamente conectada a la última capa oculta.
- Paso 2-4: Ensamblar la arquitectura completa dentro de un modelo de Tensorflow.
- Paso 2-5: Compilación del modelo con una función de pérdida de error cuadrático medio y el optimizador de Adam.

Paso 3: Implementación del algoritmo de Deep Reinforcement Learning

- Paso 3-1: Introducción e inicialización de todos los parámetros y variables del modelo de DQN.
- Paso 3-2: Hacer un método que construya la memoria en Repetición de Experiencia.
- Paso 3-3: Hacer un método que construya y devuelva dos lotes de 10 entradas y 10 objetivos.

Paso 4: Entrenamiento de la IA

- Paso 4-1: Construcción del entorno creando un objeto Environment.
- Paso 4-2: Construyendo el cerebro artificial creando un objeto de la clase de Brain.
- Paso 4-3: Construyendo el modelo DQN creando un objeto de la clase DQN.
- Paso 4-4: Elección del modo de entrenamiento.
- Paso 4-5: Comenzar el entrenamiento con un bucle durante 1000 épocas de 1000 días.
- Paso 4-6: Durante cada época, repetimos todo el proceso de Deep Q-Learning, al tiempo que exploramos el 30% de las veces.

Paso 5: Probar la IA

- Paso 5-1: Construcción de un entorno creando un objeto de la clase Environment.
- Paso 5-2: Carga del cerebro artificial con sus pesos pre-entrenados del entrenamiento anterior.
- Paso 5-3: Elección del modo de inferencia.
- Paso 5-4: Iniciación de la simulación de 200 días.
- Paso 5-5: En cada iteración (cada día), nuestra agente solo ejecuta la acción que resulta de su predicción, y no se lleva a cabo ninguna exploración o entrenamiento de Deep Q-Learning.

Capítulo 3: Resultados

Este capítulo desarrolla en la primera parte un análisis exploratorio de los datos adquiridos y posteriormente los resultados de los modelos de predicción propuestos. El objetivo es descubrir como las distintas políticas de recompensas pueden afectar al comportamiento del agente en el entorno. Esto nos ayudará a descubrir las distintas estrategias que adoptará el agente y entender por qué reacciona de esa manera.

A continuación, en la gráfica se muestra la evolución de la capitalización de mercado de bitcoin. Dentro del sector de la tecnología blockchain, el término capitalización de mercado se refiere a una métrica que mide el tamaño relativo de una criptomoneda. Se calcula multiplicando el precio de mercado actual de una criptomoneda por el número total de monedas en circulación. Los 1000 primeros días se utilizarán para el entrenamiento de los agentes y los 200 últimos para el testeo.

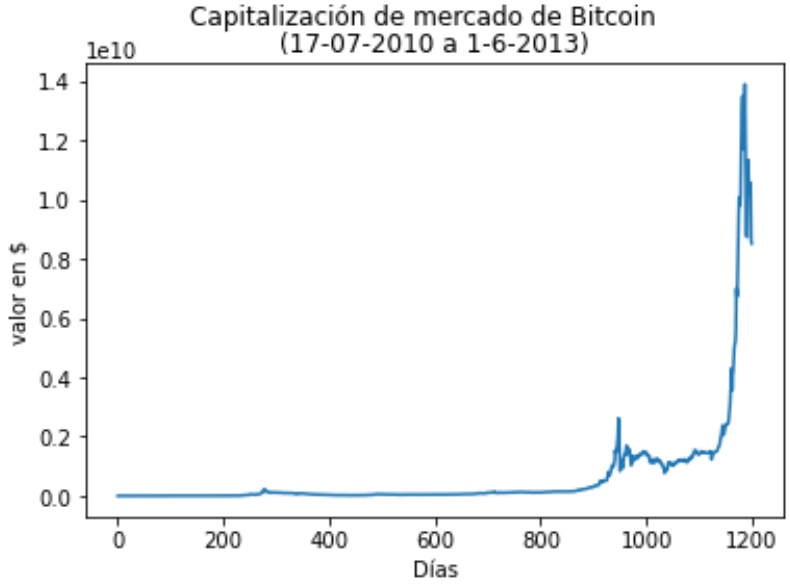

Figura 11: Evolución de la capitalización de mercado de Bitcoin

Como podemos los agentes de inteligencia artificial se enfrentarán a un mercado en auge que sufre correcciones muy fuertes y repentinas.

3.1. Desempeño de la IA en el entrenamiento

El periodo de entrenamiento será de 1000 días (17-7-2010 a 1-6-2013) y en el entorno partiremos de 3911600 monedas en circulación que es igual al número de bitcoins que había en el 17-7-2010. El numeró de épocas será 1000 donde cada época simulará 1000

días haciendo un total de 1 000 000 de días simulados. La evolución de la capitalización de mercado durante el entrenamiento se muestra a continuación.

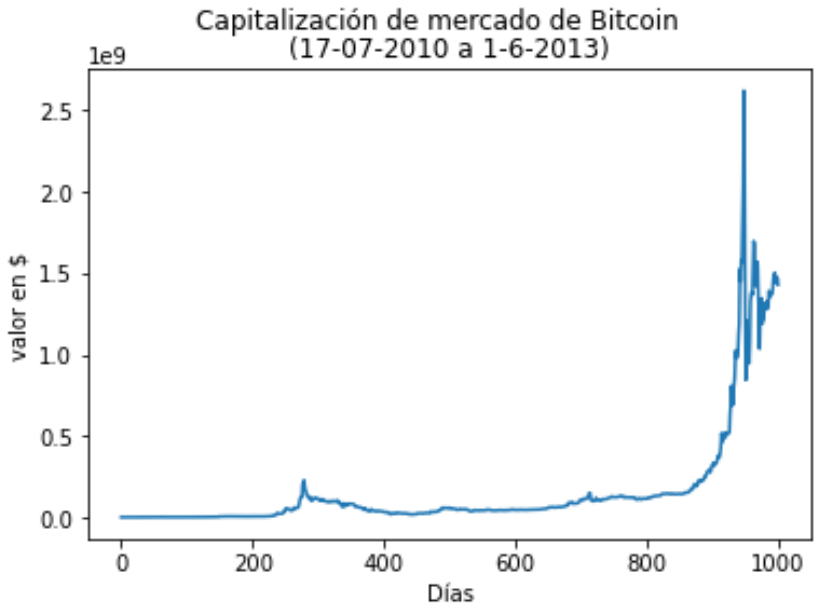

Figura 12: Capitalización de mercado de Bitcoin durante el entrenamiento

Con la configuración de los parámetros de este entrenamiento se entrenarán dos agentes. La diferencia entre ambos será la política de recompensas donde uno solamente recibirá castigos y el segundo recibirá tanto castigos como premios.

Las políticas de recompensas serán 2:

En primer lugar, el agente recibirá una recompensa negativa por cada punto que se aleje del precio óptimo. Es decir, si el precio termina siendo 1.5$ recibirá una recompensa negativa de -0.5 ya que es la diferencia con el precio óptimo.

En segundo lugar, el agente recibirá una recompensa positiva de 10 por cada vez que consiga situar el precio de la moneda entre 0.9$ y 1.1$. En cambio, por cada vez que no lo consiga recibirá una recompensa negativa por cada punto que se aleje del precio óptimo.

3.2. Testeo de los agentes de IA

El periodo de prueba será de 200 días (1-6-2013 a 18-12-2013) y en el entorno habrá 3911600 monedas en circulación, lo que supondrá que el precio de partida para los agentes de IA será 0.0000002$. El motivo por el que se parte de un precio tan bajo es que todas las criptomonedas que salen al mercado por primera vez tienen precios muy bajos.

De esta forma podemos simular y analizar también cómo funcionaría en el supuesto de una emisión inicial de monedas (ICO).

Figura 13: Capitalización de mercado de Bitcoin durante el testeo

Se ha escogido este intervalo temporal por ser el periodo con mayor volatilidad de la historia de Bitcoin. Durante los 150 primeros días no hubo casi volatilidad. Sin embargo, en los últimos 50 días podemos ver como el valor se dispara un 494% alcanzando un máximo histórico el día 186 y termina sufriendo una corrección muy fuerte del 39 % de su capitalización respecto del máximo.

En este escenario se comprobará como es capaz el agente de mantener el precio con un mercado conservador, un mercado alcista y un mercado pesimista.

Agente 1

El primer agente con el que se ha llevado a cabo el testeo ha sido el que únicamente recibía castigos o recompensas negativas.

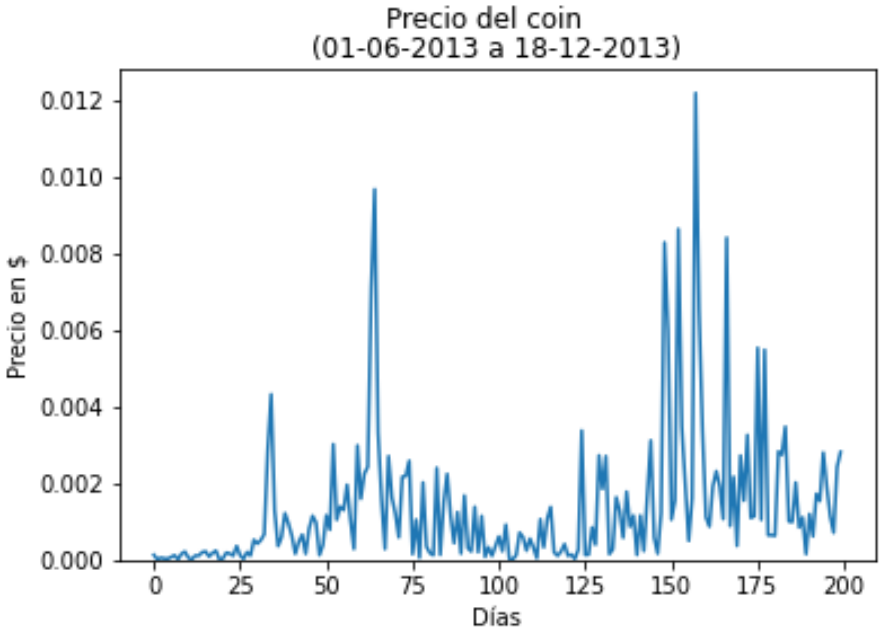

Figura 14: Precio durante el testeo del agente 1

Como podemos comprobar el agente siempre ha tratado mantener el precio cercano a 0. De hecho, apenas ha habido dos días donde el precio haya llegado a 0.01$ muy lejos del objetivo inicial de mantener el precio en torno a 1$.

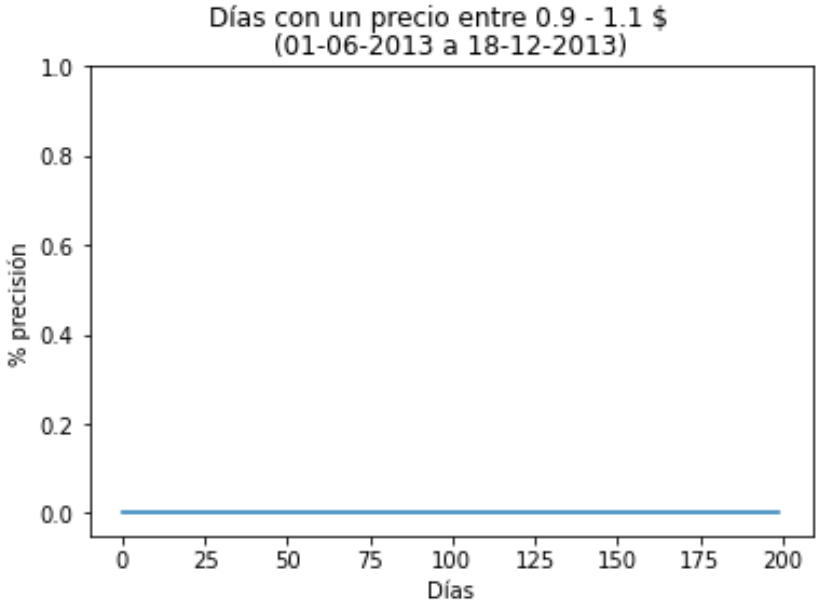

Figura 15: Porcentaje de éxito del agente 1

En esta gráfica podemos ver que no consigue mantener ni un solo día el precio entre 0.9-1.1$.

Agente 2

El segundo agente con el que se ha llevado a cabo el testeo ha sido el que además de recibir castigos o recompensas negativas también recibe una recompensa si realiza la tarea de forma satisfactoria.

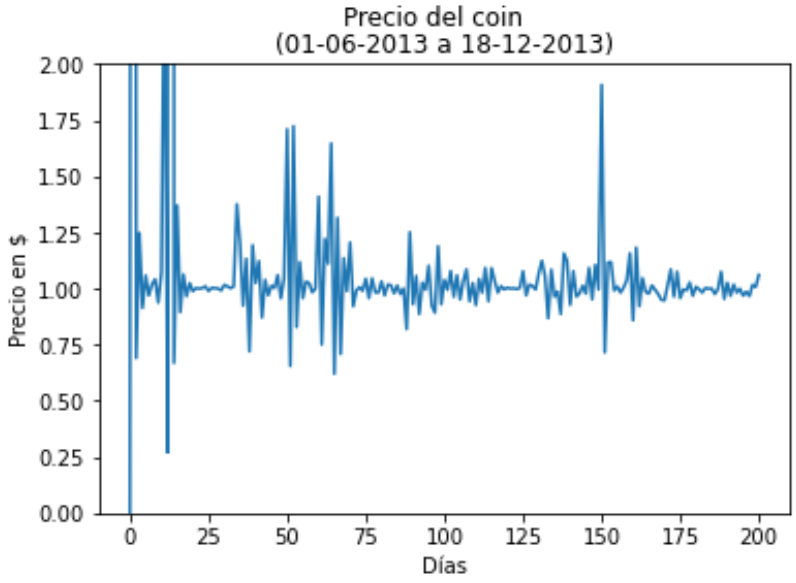

Figura 16: Precio durante el testeo del agente 2

En esta gráfica ya se puede comenzar a observar que el precio se encuentra en torno a 1$. Hay episodios de gran volatilidad sobre todo en los primeros días. Esto se debe a que inicialmente el agente parte con un número de monedas en circulación que no se corresponde con la cantidad óptima para que el precio se encuentre dentro el intervalo óptimo. Por ello, al comienzo toma decisiones más agresivas respecto al mercado y progresivamente consigue estabilizar el precio.

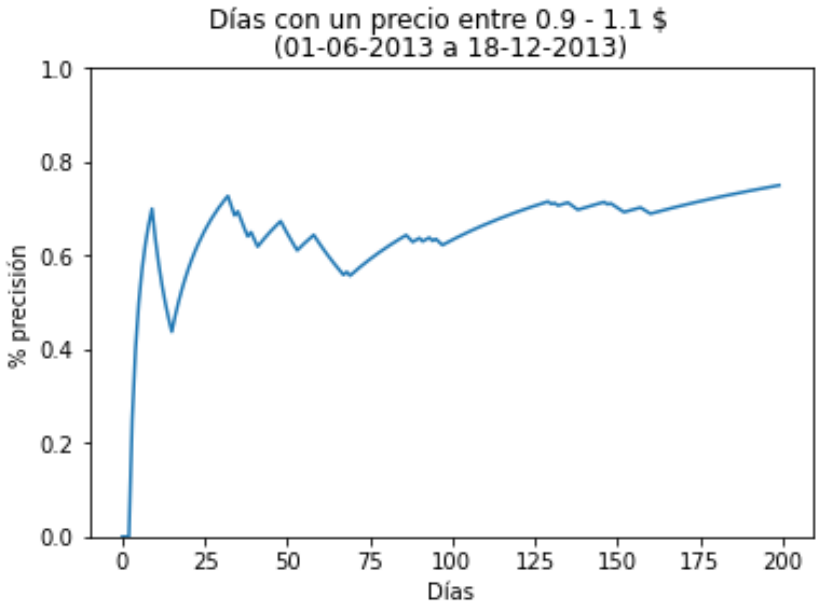

Figura 17: Porcentaje de éxito del agente 2

En esta gráfica se muestra la evolución del porcentaje de días que logra mantener el precio dentro del intervalo 0.9 – 1.1$. En línea con la gráfica del precio, el agente de IA parte de un mercado desajustado donde tiene que ir tomando políticas monetarias. Durante los primeros 75 días apenas consigue estabilizar la moneda ni el 60% de las veces. Sin embargo, de ahí en adelante el agente se ve que comienza a estabilizar mejor la moneda y mejorar progresivamente el porcentaje hasta alcanzar un 75% de días con un precio cercano a 1$. En otras palabras, este sistema de recompensas garantiza que el agente de IA mantendrá el precio estable 7,5 de cada 10 días. De hecho, el precio medio de los últimos 100 días de la simulación fue 1,01$.

3.2.1. Conclusiones del testeo de los agentes

Los resultados a evaluar de los dos agentes se muestran en la siguiente tabla.

Resultados	Precio medio últimos 100 días	Éxito estabilizando la moneda
Agente 1	0.002 $	0 %
Agente 2	1.01 $	75 %

Tabla 3: Resultados de los agentes

<u>Agente 1</u>

En primer lugar, este agente no puede ser el Banco Central de nuestro blockchain porque fracasa estrepitosamente en mantener un precio cercano a 1$.

En segundo lugar, tener una política de recompensas donde el agente solo reciba castigos provoca este comportamiento a nuestro agente porque recompensas de este problema no son simétricas. Si el precio se encuentra por debajo de 1$ el agente recibirá un castigo entre 0.999 y 0.001. Por el contrario, si el agente situase el precio por encima de 1$ recibirá un castigo entre 0.001 e infinito. De esta forma, el agente ha aprendido a moverse a la zona más segura que es por debajo de 1$.

Agente 2

El desempeño de este segundo agente de IA es muy bueno ya que ha conseguido mantener un precio estable 150 de los 200 días que ha sido puesto a prueba. De hecho, los últimos días vemos como aquellos días que se han salido del intervalo marcado son por muy poco.

Visto que los resultados han sido buenos merece la pena estudiar este agente en más profundidad en las distintas fases del mercado. La fase ICO corresponde a los primeros 74 días de la simulación, el mercado lateral corresponde entre los días 75 y 149, el mercado alcista correponde a los días 150 a 185 y el mercado bajista a los días 186 a 200.

Mercado	ICO	Lateral	Alcista	Bajista
Agente 2	58.6%	82.6%	83.3%	100%

Tabla 4: Resultados según la fase del mercado

En primer lugar, el agente ha sido capaz de partir de un escenario donde la moneda acababa de salir al mercado e ir ajustando su precio. Aunque un 58.6% pueda parecer un mal porcentaje el hecho de conseguir más de la mitad de los días estabilizar la moneda es un éxito. Por lo que, el agente está preparado para ser puesta en un blockchain desde que se haga público.

En segundo lugar, en mercados laterales podemos ver que un rendimiento del 82.6% estabilizando la moneda es bastante bueno. Sin embargo, sorprende que se le dé peor que cuando se enfrenta a un mercado alcista o bajista.

En tercer lugar, en mercados alcistas hemos podido comprobar que el precio conseguía mantenerlo estable sin ningún problema eliminando la volatilidad el 83.3% de los días con éxito.

En último lugar, el desempeño del agente para mercados bajistas es el mejor sin duda alguna puesto que ajusta el precio con éxito el 100% de las veces.

En conclusión, la introducción de una recompensa positiva en el aprendizaje del agente ha servido de motivación para que persiga nuestro objetivo con bastante éxito.

Capítulo 4: Discusión

El capítulo 4 es el último capítulo de este artículo y en él se desarrollan las ideas finales sobre la investigación realizada. El capítulo se divide en 3 secciones, la primera establece los problemas éticos y debilidades de la metodología seguida identificados durante todo el proceso de la investigación, la segunda sección trata todas las recomendaciones para investigaciones futuras, aportando una opinión subjetiva sobre el camino a seguir para contribuir a esta literatura y, por último, la tercera sección es una conclusión destacando todo el proceso y los resultados obtenidos.

4.1. Uso del blockchain

Como se ha defendido desde el inicio del trabajo la tecnología blockchain es muy útil para la democratización de la sociedad en aspectos donde aún hay un claro control de gobiernos y grandes capitales, véase el dinero controlado por bancos centrales. Sin embargo, la finalidad que le pueden dar ciertos usuarios a las criptomonedas tampoco tiene porque ser buena. Tal es así que este medio de pago predomina en las transacciones de la *dark web* donde se pueden adquirir armas y drogas entre otras cosas. No hace falta irse al lugar más recóndito de internet, también comisionistas y empresas emplean las criptomonedas para ocultar ingresos y así evadir impuestos.

El dilema principal que nos encontramos es una balanza de beneficios y perjuicios. Por un lado, habrá usuarios de blockchain que realicen actividades ilegales para su beneficio egoísta. Sin embargo, nos preguntamos si es motivo suficiente para detener todo el progreso tecnológico que aporta y aportará en un futuro cercano.

4.2. Debilidades del entorno

El mercado de criptomonedas se caracteriza por no ser un mercado determinista. Esto se debe a que las personas que son las que deciden comprar y vender una moneda se comportan de forma irracional. La irracionalidad se puede deber a la falta de información o bien por los impulsos y emociones de las personas.

La IA actúa en un entorno que simula el mercado basándose en datos históricos. Sin embargo, el componente irracional se pierde en la simulación porque la reacción del mercado se ha tratado de forma determinista. Además, se ha dado por supuesto que los mineros al recibir las recompensas pondrán en circulación automáticamente las monedas recibidas.

Aun así, en algún momento los mineros necesitarán dinero para costear su actividad y sabiendo que las tasas percibidas son su único ingreso en algún momento tendrán que ponerlo en circulación. Se podría intuir que realmente lo que pasará es que el precio tardará más tiempo en ajustarse y de una forma más suavizada.

4.3. Limitaciones técnicas

El entrenamiento del agente ha sido llevado a cabo en un portátil personal comprado en 2016. Para el entrenamiento de los dos agentes se decidió que el entrenamiento sería máximo de 1000 épocas, durando cada uno de los entrenamientos de los dos agentes aproximadamente 8 horas. Con un poder de computación tan limitado al menos hemos podido estudiar dos políticas de recompensas, dejando de lado el estudio de otros factores como las acciones, arquitecturas distintas de redes neuronales o los hiperparámetros.

4.4. Trabajo futuro

En esta sección se ofrecen mejoras necesarias para la viabilidad de este proyecto. Entre otras posibles mejoras que serían interesantes encontramos las siguientes.

En primer lugar, sería necesario estudiar la posibilidad de aumentar la frecuencia de interacción de la inteligencia artificial con el blockchain. En nuestro caso el ajuste de la liquidez de la cadena se realiza de forma diaria cuando sería mucho más útil para la comunidad que se garantizará esa estabilidad en el precio cada hora, cada minuto o incluso en cada transacción.

En segundo lugar, como testigo se dejan el código para construir un blockchain y hay dos agentes entrenados. Sería muy interesante llevar a cabo distintas simulaciones donde el número de mineros, usuarios y especuladores varié para estudiar la reacción del blockchain en distintos escenarios.

En tercer lugar, podrían estudiarse políticas sustitutivas o complementarias para controlar la oferta de monedas. En este caso solamente se controlaban las recompensas de los mineros, pero quizá introducir un interés que vaya variando al *staking*, consiste en bloquear monedas a cambio de un interés, podría contribuir a modificar el comportamiento de los *holders*, especuladores a largo plazo de criptomonedas, en favor de la estabilidad del precio de la moneda. Por ejemplo, en momentos de precios por encima de 1$ un interés negativo penalizaría a aquellos que ahorran monedas haciendo que circulen más monedas.

Una futura línea de investigación podría enfrentar en el mismo blockchain una inteligencia que estabilice la moneda frente a otra inteligencia que trate de obtener beneficio mediante la especulación y el *staking*. De esta forma, se podría estudiar como las políticas monetarias afectan el comportamiento de los especuladores y viceversa.

4.5. Conclusión

En conclusión, en este trabajo se estudia la implementación una inteligencia artificial en un blockchain para crear una stablecoin vinculada al valor del USD. Para ello, se han empleado los datos históricos de bitcoin por ser la moneda digital por excelencia. La investigación partía de la hipótesis, siguiendo la lógica de la oferta y la demanda, una modificación de la cantidad de monedas en circulación afecta en sentido opuesto el precio de la moneda.

El trabajo se presenta un modelo de Inteligencia artificial y la arquitectura de un blockchain con el que la interactuar. Con todo esto, dentro del campo de Deep Reinforment Learning se ha empleado la técnica Deep Q-Learning. Básicamente, con esta metodología le hemos dotado de un cerebro con memoria a los agentes que hemos producido. Los resultados han sido muy satisfactorios porque el segundo de los agentes ha conseguido mantener el precio estable el 75% de los 200 días de la simulación.

Esta metodología también puede seguirse para diseñar otros comportamientos deseados o replicar otros tipos de divisas o valores y conseguir resultados similares con el diseño propuesto en el trabajo. El motivo por el que se ha asociado al USD es porque es la divisa referente a nivel mundial y la tremenda utilidad que tienen las stablecoins para el uso diario de la tecnología blockchain.

Bibliografía

Agarap, A. F. (7 de febrero de 2019). Deep Learning using Rectified Linear Units (ReLU).

Barto, R. S. (31 de mayo de 1992). Reinforcement Learning I: Introduction. *MIT Press*.

Basis.io. (5 de abril de 2021). Obtenido de https://www.basis.io/

Bellman, R. (30 de julio de 1964). *The Theory of Dynamic Programming.* Obtenido de https://www.rand.org/content/dam/rand/pubs/papers/2008/P550.pdf

Çavuşoğlu, D. (19 de octubre de 2020). *Backpropagation paper from scratch.* Obtenido de Towards data science: https://towardsdatascience.com/backpropagation-paper-from-scratch-796793789248

CoinGecko. (2021 de marzo de 08). Obtenido de https://www.coingecko.com/es/monedas/bitcoin

CoinGecko. (8 de marzo de 2021). Obtenido de https://www.coingecko.com/es/monedas/bitcoin

elEconomista. (22 de julio de 2010). *La última auditoria a la Fed pone de manifiesto la corrupción del banco central*. Obtenido de El Economista: https://www.eleconomista.es/empresas-finanzas/noticias/3251040/07/11/La-ultima-auditoria-a-la-Fed-pone-de-manifiesto-la-corrupcion-del-banco-central-.html

Eurosistema. (s.f.). *Banco.* Obtenido de ECB: https://www.ecb.europa.eu/ecb/html/index.es.html

Francia, L. (8 de Febrero de 2021). *Tesla invierte 1.500 millones de dólares en bitcoin y aceptará la compra de sus coches en esta criptomoneda.* Obtenido de RTVE: https://www.rtve.es/noticias/20210208/bitcoin-alcanza-maximos-historicos-inversion-tesla-1500-millones-dolares/2074149.shtml

G-Coin. (2021 de abril de 10). Obtenido de https://www.gcoin.com/

Hinton, G. (1 de junio de 2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *University of Toronto*.

Juliani, A. (16 de agosto de 2016). *medium.com*. Obtenido de Simple Reinforcement Learning with Tensorflow: https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-0-q-learning-with-tables-and-neural-networks-d195264329d0

Kingma, D. P. (30 de enero de 2017). *ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION.* Obtenido de https://arxiv.org/pdf/1412.6980.pdf

Loiseau, J.-C. B. (11 de marzo de 2019). *Rosenblatt's perceptron, the first modern neural network*. Obtenido de towards data science: https://towardsdatascience.com/rosenblatts-perceptron-the-very-first-neural-network-37a3ec09038a

Marca. (15 de enero de 2021). David Barral se convierte en el primer fichaje de la historia en criptomonedas. págs. https://www.marca.com/futbol/mas-futbol/segunda-b-grupo-v/2021/01/15/6001d589268e3eb9448b459d.html.

Moore, G. E. (1965). The future of integrated electronics. *Electronics magazine*.

Moreno, V. (2020). La máquina financiera del expolio. Unión Editorial.

News, B. (24 de julio de 2019). Cambridge Analytica: la multa récord que deberá pagar Facebook por la forma en que manejó los datos de 87 millones de usuarios. *BBC News*, págs. https://www.bbc.com/mundo/noticias-49093124.

Poidevin, O. L. (9 de junio de 2020). George Floyd: Black Lives Matter protests go global. *BBC News*.

Ponteves, H. d. (9 de septiembre de 2020). *Blockchain A-Z: Learn how to build your Blockchain.* Obtenido de Udemy: https://www.udemy.com/course/build-your-blockchain-az/

Prechelt, L. (1 de abril de 2015). *Early stopping - But when?* Obtenido de Universit• at Karlsruhe: https://www.researchgate.net/publication/2874749_Early_Stopping_-_But_When

Rao, A. (2017). *What's the real value of AI for your business and how can you capitalise?* Obtenido de PWC: https://www.pwc.com/gx/en/issues/data-and-analytics/publications/artificial-intelligence-study.html

Sánchez, Á. (1 de enero de 2021). Una legión de foreros de Reddit hace perder miles de millones a fondos bajistas de Wall Street. *El Pais*.

Sazli, M. H. (6 de febrero de 2006). *A BRIEF REVIEW OF FEED-FORWARD NEURAL NETWORKS.* Obtenido de Ankara University: https://www.researchgate.net/publication/228394623_A_brief_review_of_feed-forward_neural_networks

Senior, A. W. (15 de enero de 2020). *AlphaFold: Improved protein structure prediction using potential from deep learning.* Obtenido de DeepMind: https://deepmind.com/blog/article/AlphaFold-Using-AI-for-scientific-discovery

Stellar.org. (9 de marzo de 2021). Obtenido de https://www.stellar.org/?locale=es

Sutton, R. (4 de febrero de 1988). *Learning to Predict by the Methods of Temporal Diferences.* Obtenido de https://link.springer.com/content/pdf/10.1007/BF00115009.pdf

Sutton, R. S. (4 de febrero de 1988). Learning to Predict by the Method of Temporal Diferences. *Kluwer Academic Publishers*.

Tom Schaul, J. Q. (25 de Febrero de 2016). PRIORITIZED EXPERIENCE REPLAY. *Googel DeepMind*.

Tomic, M. (2010). *Adaptive e-greedy Exploration in Reinforcement.* Obtenido de University of Ulm: https://doi.org/10.1007/978-3-642-16111-7_23

Tensorflow. Obtenido de https://github.com/tensorflow

Código:

Código 1: Entorno

```
# Inteligencia Artifical
# Creación del Entorno

# Importar las librerías
import numpy as np
import pandas as pd
import tensorflow as tf
data = pd.read_csv('bitcoin_dataset.csv')
data = data[['btc_market_price', 'btc_total_bitcoins',
'btc_market_cap','btc_trade_volume']]
data = data[50:1800]
data.fillna(0)



# CONSTRUIR EL ENTORNO EN UNA CLASE

class Environment(object):
    # INTRODUCIR E INICIALIZAR LOS PARÁMETROS Y VARIABLES DEL ENTORNO
    def __init__(self, optimal_price = (0.9, 1.1), data = data):
        self.data = data
        self.optimal_price = optimal_price
        self.min_price = 0
        self.min_supply = data.btc_total_bitcoins.min()
        self.max_supply = data.btc_total_bitcoins.max()
        self.min_market_cap = data.btc_market_cap.min()
        self.max_market_cap = data.btc_market_cap.max()
        self.min_trading_volume = data.btc_trade_volume.min()
        self.max_trading_volume = data.btc_trade_volume.max()
        self.initial_date = 0
        self.current_date = self.initial_date
        self.initial_trading_volume = data.iloc[0,3]
        self.current_trading_volume = self.initial_trading_volume
self.initial_market_cap = data.iloc[0,2]
        self.current_market_cap = self.initial_market_cap
        self.initial_price = data.iloc[0,0]
        self.initial_supplyai = data.iloc[0,1]
```

```
        self.coin_supplyai = self.initial_supplyai
        self.price_ai = self.current_market_cap /  self.coin_supplyai
        self.fair_price = 0
        self.reward = 0.0
        self.total_reward = 0.0
        self.game_over = 0 # Indica el fin de la simulacion, variable
booleana
        self.train = 1 # En testing sera 0

    # CREAR UN MÉTODO QUE ACTUALICE EL ENTORNO JUSTO DESPUÉS DE QUE LA
IA EJECUTE UNA ACCIÓN
    def update_env(self, direction, coin_var, current_date):
        reward_ia = 0.0
        self.coin_supplyai += coin_var * direction
        if self.coin_supplyai  < 1:
            self.game_over = 1
        else:
           self.price_ai =
(self.current_market_cap)/self.coin_supplyai

        if(self.price_ai  < self.optimal_price[0]):
            reward_ia = self.price_ai - self.optimal_price[0]
elif self.price_ai > self.optimal_price[1]:
            reward_ia = self.optimal_price[1] - self.price_ai
        else:
            reward_ia = 10
            self.fair_price += 1
        self.reward = 1e-3*reward_ia

        # OBTENCIÓN DEL SIGUIENTE ESTADO
        self.current_trading_volume = data.iloc[self.current_date+1,3]
        self.current_market_cap = data.iloc[self.current_date+1,2]
        self.current_date += 1


        # ESCALAR EL SIGUIENTE ESTADO
        scaled_trading_volume = (self.current_trading_volume -
self.min_trading_volume)/(self.max_trading_volume -
self.min_trading_volume)
        scaled_market_cap = (self.current_market_cap -
self.min_market_cap)/(self.max_market_cap - self.min_market_cap)
```

```
        scaled_supply = (self.coin_supplyai -
self.min_supply)/(self.max_supply - self.min_supply)
        next_state = np.matrix([scaled_market_cap,
scaled_trading_volume, scaled_supply]) # vector fila
        #next_state = np.matrix([self.current_market_cap,
self.current_trading_volume, self.coin_supplyai])
        reward = self.reward
        game_over = self.game_over

        # DEVOLVER EL SIGUIENTE ESTADO, RECOMPENSA Y GAME OVER
        return next_state, reward, game_over

    # CREAR UN MÉTODO QUE REINICIE EL ENTORNO
            # reiniciamos despues de cada epoch
    def reset(self):
        self.current_date = self.initial_date
        self.current_market_cap = self.initial_market_cap
        self.current_trading_volume = self.initial_trading_volume
        self.coin_supplyai = self.initial_supplyai
        self.price_ai = self.current_market_cap/ self.coin_supplyai
        self.fair_price = 0
        self.reward = 0.0
        self.game_over = 0
        self.train = 1


    # CREAR UN MÉTODO QUE NOS DE EN CUALQUIER INSTANTE EL ESTADO
ACTUAL, LA ÚLTIMA RECOMPENSA Y EL VALOR DE GAME OVER
    def observe(self, timestep):
        scaled_trading_volume = (self.current_trading_volume -
self.min_trading_volume)/(self.max_trading_volume -
self.min_trading_volume)
        scaled_market_cap = (self.current_market_cap -
self.min_market_cap)/(self.max_market_cap - self.min_market_cap)
        scaled_supply = (self.coin_supplyai -
self.min_supply)/(self.max_supply - self.min_supply)
        current_state = np.matrix([scaled_market_cap,
scaled_trading_volume, scaled_supply]) # vector fila
                # esta matriz sera la capa de entrada

        return current_state, self.reward, self.game_over
```

Código 2: Cerebro

```
import tensorflow as tf

class Brain(object):
    def __init__(self, learning_rate = 0.001, number_actions = 7):
        self.learning_rate = learning_rate
        model = tf.keras.models.Sequential()
        model.add(tf.keras.layers.Dense(units=64, activation='relu',
input_shape=(3, )))
        model.add(tf.keras.layers.Dropout(0.1))
        model.add(tf.keras.layers.Dense(units=32,
activation='sigmoid'))
        model.add(tf.keras.layers.Dropout(0.1))
        model.add(tf.keras.layers.Dense(units=number_actions,
activation='softmax'))
        self.model = model
        self.model.compile = model.compile(optimizer='adam',
loss='mse')
```

Código 3: DQN

```
# Creación de la red Q profunda

# Importar las librerías
import numpy as np

# IMPLEMENTAR EL ALGORITMO DE DEEP Q-LEARNING CON REPETICIÓN DE
EXPERIENCIA

class DQN(object):

    # INTRODUCCIÓN E INICIALIZACIÓN DE LOS PARÁMETROS Y VARIABLES DEL
DQN
    def __init__(self, max_memory = 100, discount_factor = 0.9): #
Discount factor ya esta testeado
        self.memory = list()
        self.max_memory = max_memory # maximo de transacciones
        self.discount_factor = discount_factor

    # CREACIÓN DE UN MÉTODO QUE CONSTRUYA LA MEMORIA DE LA REPETICIÓN
DE EXPERIENCIA
    def remember(self, transition, game_over):
        self.memory.append([transition, game_over])
        if len(self.memory) > self.max_memory:
            del self.memory[0]

    # CREACIÓN DE UN MÉTODO QUE CONSTRUYA DOS BLOQUES DE ENTRADAS Y
TARGETS EXTRATENDO TRANSICIONES
    def get_batch(self, model, batch_size = 10):
        len_memory = len(self.memory)
        num_inputs = self.memory[0][0][0].shape[1] # 1° transicion 2°
gameover 3° transferencia de datos
        num_outputs = model.output_shape[-1]
        inputs = np.zeros((min(batch_size, len_memory), num_inputs))
# utilizamos min para adaptar la dim de la matriz
        targets = np.zeros((min(batch_size, len_memory), num_outputs))
        for i, idx in enumerate(np.random.randint(0, len_memory,
size=min(len_memory, batch_size))):
            current_state, action, reward, next_state =
self.memory[idx][0] # enumerate
```

```
            game_over = self.memory[idx][1]
            inputs[i] = current_state
            targets[i] = model.predict(current_state)[0]
            Q_sa = np.max(model.predict(next_state)[0])
            if game_over: # gameover == True:
                targets[i, action] = reward
            else:
                targets[i, action] = reward +
self.discount_factor*Q_sa
        return inputs, targets
```

Código 4: Entrenamiento

```
# Importar las librerías y otros ficheros de python
import os
import numpy as np
import random as rn

import environment
import brain
import dqn

# Configurar las semillas para reproducibilidad
os.environ['PYTHONHASHSEED'] = '0'
np.random.seed(42)
rn.seed(12345)

# CONFIGURACIÃ“N DE LOS PARÃMETROS
epsilon = 0.3
number_actions = 7
direction_boundary = 1
number_epochs = 1000
max_memory = 3000
batch_size = 20

env = environment.Environment(optimal_price = (0.9, 1.1))
# CONSTRUCCIÓN DEL ENTORNO CREANDO UN OBJETO DE LA CLASE ENVIRONMENT

# CONSTRUCCIÓN DEL CEREBRO CREANDO UN OBJETO DE LA CLASE BRAIN
brain = brain.Brain(learning_rate = 0.001, number_actions =
number_actions)

# CONSTRUCCIÓN DEL MODELO DQN CREANDO UN OBJETO DE LA CLASE DQN
dqn = dqn.DQN(max_memory = max_memory, discount_factor = 0.9)

# ELECCIÓN DEL MODO DE ENTRENAMIENTO
train = True

# ENTRENAR LA IA
env.train = train
model = brain.model
```

```
early_stopping = True
patience = 10
best_total_reward = -np.inf
patience_count = 0
if (env.train):

    # INICIAR EL BUCLE DE TODAS LAS ÉPOCAS (1 Epoch = 10 Meses)
    for epoch in range(1, number_epochs+1):
        # INICIALIZACIÓN DE LAS VARIABLES DEL ENTORNO Y DEL BUCLE DE
ENTRENAMIENTO
        total_reward = 0.
        loss = 0.
        env.reset()
        game_over = False
        current_state, _, _ = env.observe(0)
        env.current_date = 0
        # INICIALIZACIÓN DEL BUCLE DE TIMESTEPS (Timestep = 1 día) EN
UNA ÉPOCA
        while ((not game_over)):
            # EJECUTAR LA SIGUIENTE ACCIÓN POR EXPLORACIÓN
            if np.random.rand() <= epsilon:
                action = np.random.randint(0, number_actions)

            # EJECUTAR LA SIGUIENTE ACCIÓN POR INFERENCIA
            else:

                q_values = model.predict(current_state)
                action = np.argmax(q_values[0])
            coin_price = env.current_market_cap/env.coin_supplyai
            if (coin_price <= direction_boundary):
                direction = -1
            else:
                direction = 1
            if action == 0:
                coin_var = abs((coin_price * env.coin_supplyai) -
env.coin_supplyai) * 0.1
            elif action == 1:
                coin_var = abs((coin_price * env.coin_supplyai) -
env.coin_supplyai) * 0.5
            elif action == 2:
```

```
                coin_var = abs((coin_price * env.coin_supplyai) -
env.coin_supplyai) * 0.75
            elif action == 3:
                coin_var = abs((coin_price * env.coin_supplyai) -
env.coin_supplyai) * 1
            elif action == 4:
                if env.price_ai > 1/2:
                    coin_var = abs((coin_price * env.coin_supplyai) -
env.coin_supplyai) * 1.25
                else:
                    coin_var = env.coin_supplyai - 1
            elif action == 5:
                if env.price_ai > 1/3:
                    coin_var = abs((env.price_ai * env.coin_supplyai)
- env.coin_supplyai) * 1.5
                else:
                    coin_var = env.coin_supplyai - 1
            elif action == 6:
                if env.price_ai > 0.43:
                    coin_var = abs((env.price_ai * env.coin_supplyai)
- env.coin_supplyai) * 1.75
                else:
                    coin_var = env.coin_supplyai - 1

            coin_var = int(coin_var)

            # ACTUALIZAR EL ENTORNO Y ALCANZAR EL SIGUIENTE ESTADO
            next_state, reward, game_over = env.update_env(direction,
coin_var, timestep)
            total_reward = total_reward + reward

            # ALMACENAR LA NUEVA TRANSICIÓN EN LA MEMORIA
            dqn.remember([current_state, action, reward, next_state],
game_over)

            # OBTENER LOS DOS BLOQUES SEPARADOS DE ENTRADAS Y
OBJETIVOS
            inputs, targets = dqn.get_batch(model, batch_size)

            # CALCULAR LA FUNCIÓN DE PÉRDIDAS UTILIZANDO TODO EL
BLOQUE DE ENTRADA Y OBJETIVOS
```

```
            loss += model.train_on_batch(inputs, targets)

            timestep += 1
            current_state = next_state

        # IMPRIMIR LOS RESULTADOS DEL ENTRENAMIENTO AL FINAL DEL EPOCH
        print("\n")
        print("Epoch: {:03d}/{:03d}.".format(epoch, number_epochs))
        print(" Error total en el precio: {:.0f} J.".format(loss))
        print("game over en: {}".format(timestep))


model.save("model.h5") #Guardamos modelo
```

Código 5: Test

```
# Fase de testing
# Importar las librerías y otros ficheros de python
import os
import numpy as np
import random as rn
import tensorflow as tf
import environment # no voy a usar dpq ni brain porque y tengo el
modelo guardado

# Configurar las semillas para reproducibilidad
os.environ['PYTHONHASHSEED'] = '0'
np.random.seed(42)
rn.seed(12345)


# CONFIGURACIÓN DE LOS PARÁMETROS
number_actions = 7
direction_boundary = 1
fair_price_days = []

# CONSTRUCCIÓN DEL ENTORNO CREANDO UN OBJETO DE LA CLASE ENVIRONMENT
env = environment.Environment(optimal_price = (0.90, 1.1))

# CARGA DE UN MODELO PRE ENTRENADO
model = tf.keras.models.load_model("model.h5")


# ELECCIÓN DEL MODO DE ENTRENAMIENTO
train = False

# EJECUCIÓN DE UN AÑO DE SIMULACIÓN EN MODO INFERENCIA
env.train = train
current_state, _, _ = env.observe(1800)

for timestep in range(1000, 1100):
    q_values = model.predict([current_state])
    action = np.argmax(q_values[0])

    if (env.price_ai <= direction_boundary):
```

```
            direction = -1
        else:
            direction = 1
        if action == 0:
            coin_var = abs((env.price_ai * env.coin_supplyai) -
env.coin_supplyai) * 0
        elif action == 1:
            coin_var = abs((coin_price * env.coin_supplyai) -
env.coin_supplyai) * 0.5
        elif action == 2:
            coin_var = abs((env.price_ai * env.coin_supplyai) -
env.coin_supplyai) * 0.75
        elif action == 3:
            coin_var = abs((env.price_ai * env.coin_supplyai) -
env.coin_supplyai) * 1
        elif action == 4:
            if env.price_ai > 1/2:
                coin_var = abs((env.price_ai * env.coin_supplyai) -
env.coin_supplyai) * 1.25
            else:
                coin_var = env.coin_supplyai - 1
        elif action == 5:
            if env.price_ai > 1/3:
                coin_var = abs((env.price_ai * env.coin_supplyai) -
env.coin_supplyai) * 1.5
            else:
                coin_var = env.coin_supplyai - 1
        elif action == 6:
            if env.price_ai > 0.43:
                coin_var = abs((env.price_ai * env.coin_supplyai) -
env.coin_supplyai) * 1.75
            else:
                coin_var = env.coin_supplyai - 1
        coin_var = int(coin_var)
        next_state, reward, game_over = env.update_env(direction,
coin_var, timestep)

        current_state = next_state
        if 0.9 < env.price_ai < 1.1:
            fair_price_days.append(1)
        else:
```

```
        fair_price_days.append(0)

# IMPRIMIR LOS RESULTADOS DEL ENTRENAMIENTO AL FINAL DEL EPOCH
print(np.mean(fair_price_days))
```

Código 6: Librería Blockchain

```
# Importamos las librerías
import datetime
import hashlib
import json
import requests # utilizaremos para conectar nodos
from urllib.parse import urlparse
import tensorflow as tf
import numpy as np
import environment

env = environment.Environment(optimal_price = (0.90, 1.1))
direction_boundary = 1

train = False


env.train = train
current_state, _, _ = env.observe(timestep = 1000)
# Part 1 - Diseño del Blockchain

class Blockchain:

    def __init__(self):
        self.chain = []
        self.transactions = [] # objeto con transacciones que añadimos
al block
        self.transactions_fees = []
        self.create_block(proof = 1, previous_hash = '0') # es
importante que que esta funcion por debajo de transactions
        self.nodes = set() # conjunto vacio
        self.model = tf.keras.models.load_model("model.h5")

    def create_block(self, proof, previous_hash):
        block = {'index': len(self.chain) + 1,
                 'timestamp': str(datetime.datetime.now()),
                 'proof': proof,
                 'previous_hash': previous_hash,
                 'transactions': self.transactions} # Añadimos
transacciones
```

```
        self.transactions = [] # borramos transactions
        self.chain.append(block)
        return block
    def create_reward_block(self, proof, previous_hash,node_address):
        block = {'index': len(self.chain) + 1,
                 'timestamp': str(datetime.datetime.now()),
                 'proof': proof,
                 'previous_hash': previous_hash,
                 'transactions': self.transactions_fees} # Añadimos
transacciones
        self.transactions_fees = [] # borramos transactions
        self.chain.append(block)
        return block

    def get_previous_block(self):
        return self.chain[-1]

    def proof_of_work(self, previous_proof):
        new_proof = 1
        check_proof = False
        while check_proof is False:
            hash_operation = hashlib.sha256(str(new_proof**2 -
previous_proof**2).encode()).hexdigest()
            if hash_operation[:4] == '0000':
                check_proof = True
            else:
                new_proof += 1
        return new_proof

    def hash(self, block):
        encoded_block = json.dumps(block, sort_keys = True).encode()
        return hashlib.sha256(encoded_block).hexdigest()

    def is_chain_valid(self, chain):
        previous_block = chain[0]
        block_index = 1
        while block_index < len(chain):
            block = chain[block_index]
            if block['previous_hash'] != self.hash(previous_block):
                return False
            previous_proof = previous_block['proof']
```

```
            proof = block['proof']
            hash_operation = hashlib.sha256(str(proof**2 -
previous_proof**2).encode()).hexdigest()
            if hash_operation[:4] != '0000':
                return False
            previous_block = block
            block_index += 1
        return True

    def add_transaction(self, sender, receiver, amount, fee, miner ):
# key elements of transactions, quian, a quien y cuanto
        self.transactions.append({'sender': sender,        # añadimos
la transaccion a transactions
                                  'receiver': receiver,
                                  'amount': amount,
                                  'fee': fee})
        previous_block = self.get_previous_block()
        self.transactions_fees.append({'sender': sender,
                                       'miner':miner,
                                       'fee':fee})
        return previous_block['index'] + 1


    def add_node(self, address): # address es el numero del puerto
        parsed_url = urlparse(address)
        self.nodes.add(parsed_url.netloc) # .netloc es un atributo
generado por el parse

    def replace_chain(self): # cambia cualquier otra cadena que se
duplique
        network = self.nodes
        longest_chain = None
        max_length = len(self.chain)
        for node in network: # con el bucle encontramos el node con el
longest chain
            response = requests.get(f'http://{node}/get_chain') #
requests obtenemos la chain y el leght
            # los nodos se diferencia por el port
            # f hacemos referencia al node fuera del str
            if response.status_code == 200: # comporbamos que todo OK
                length = response.json()['length']
```

```
                chain = response.json()['chain']
                if length > max_length and self.is_chain_valid(chain):
# chequeamos que la chain sea valida
                    max_length = length
                    longest_chain = chain
        if longest_chain:
            self.chain = longest_chain
            return True
        return False

    def reward_distributor(self, current_state):
        current_state, _, _ = env.observe()
        q_values = self.model.predict([current_state])
        action = np.argmax(q_values[0])


        if (env.price_ai <= direction_boundary):
            direction = -1
        else:
            direction = 1
        if action == 0:
            coin_var = abs((env.price_ai * env.coin_supplyai) -
env.coin_supplyai) * 0.1
        elif action == 1:
            coin_var = abs((env.price_ai * env.coin_supplyai) -
env.coin_supplyai) * 0.5
        elif action == 2:
            coin_var = abs((env.price_ai * env.coin_supplyai) -
env.coin_supplyai) * 0.75
        elif action == 3:
            coin_var = abs((env.price_ai * env.coin_supplyai) -
env.coin_supplyai) * 1
        elif action == 4:
            if env.price_ai > 1/2:
                coin_var = abs((env.price_ai * env.coin_supplyai) -
env.coin_supplyai) * 1.25
            else:
                coin_var = env.coin_supplyai - 1
        elif action == 5:
            if env.price_ai > 1/3:
```

```
                coin_var = abs((env.price_ai * env.coin_supplyai) -
env.coin_supplyai) * 1.5
            else:
                coin_var = env.coin_supplyai - 1
        elif action == 6:
            if env.price_ai > 0.43:
                coin_var = abs((env.price_ai * env.coin_supplyai) -
env.coin_supplyai) * 1.75
            else:
                coin_var = env.coin_supplyai - 1
        coin_var = int(coin_var)

        total_fee = 0
        for i in range(0,len(self.transactions_fees)):
            total_fee += self.transactions_fees[i]['fee']

        if direction == 1:
            fee_rate = (coin_var + total_fee)/total_fee
        else:
            fee_rate = (total_fee-coin_var)/total_fee

        for i in range(0,len(self.transactions_fees)):

self.transactions_fees[i]['fee']=self.transactions_fees[i]['fee']*fee_
rate
        return fee_rate
```

Código 7: Aplicación Web del Blockchain

```
from flask import Flask, jsonify, request
from uuid import uuid4 # crear un addres para cada nodo
import Blockchain

# Crear la Web App
app = Flask(__name__)

# Crear una dirección para el nodo en el Puerto 5000
node_address = str(uuid4()).replace('-', '') # uuid4 crea el address
aleatorio unico que se asigna al nodo
                                             # Usamos replace para
eliminar -
# Crear el Blockchain
blockchain = Blockchain()

# Minado de un nuevo bloque
@app.route('/mine_block', methods = ['GET'])
def mine_block():
    previous_block = blockchain.get_previous_block()
    previous_proof = previous_block['proof']
    proof = blockchain.proof_of_work(previous_proof)
    previous_hash = blockchain.hash(previous_block)
    blockchain.add_transaction(sender = 'jose', receiver = 'Álvaro',
amount = 1, fee = 1, miner= node_address)
    block = blockchain.create_block(proof, previous_hash)
    response = {'message': 'Congratulations, you just mined a block!',
                'index': block['index'],
                'timestamp': block['timestamp'],
                'proof': block['proof'],
                'previous_hash': block['previous_hash'],
                'transactions': block['transactions']}
    return jsonify(response), 200

# Minado del reward block
@app.route('/mine_block_reward', methods = ['GET'])
def mine_block_reward():
    previous_block = blockchain.get_previous_block()
    previous_proof = previous_block['proof']
    proof = blockchain.proof_of_work(previous_proof)
```

```
    previous_hash = blockchain.hash(previous_block)
    blockchain.reward_distributor(current_state)
    block = blockchain.create_reward_block(proof, previous_hash,
node_address)
    response = {'message': 'Congratulations, daily rewards have just
been mined in this block!',
                'index': block['index'],
                'timestamp': block['timestamp'],
                'proof': block['proof'],
                'previous_hash': block['previous_hash'],
                'transactions': block['transactions']}
    return jsonify(response), 200

# Obtención de la cadena actual
@app.route('/get_chain', methods = ['GET'])
def get_chain():
    response = {'chain': blockchain.chain,
                'length': len(blockchain.chain)}
    return jsonify(response), 200

# Comprobar si la cadena actual es válida
@app.route('/is_valid', methods = ['GET'])
def is_valid():
    is_valid = blockchain.is_chain_valid(blockchain.chain)
    if is_valid:
        response = {'message': 'All good. The Blockchain is valid.'}
    else:
        response = {'message': 'The Blockchain is not valid.'}
    return jsonify(response), 200

# Añadir una nueva transacción al Blockchain
@app.route('/add_transaction', methods = ['POST'])
def add_transaction():
    json = request.get_json() # esto cogera el json y lo subira
    transaction_keys = ['sender', 'receiver', 'amount','fee', 'miner']
    if not all(key in json for key in transaction_keys): # si todas
las key en el transaction_key no estan en json
        return 'Some elements of the transaction are missing', 400
    index = blockchain.add_transaction(json['sender'],
json['receiver'], json['amount'], json['fee'], json['miner'])
```

```
    response = {'message': f'This transaction will be added to Block
{index}'} # f' input the variable {}
    return jsonify(response), 201 # 201 todo ha ido bien Created

# Part 3 - Descentralizando el Blockchain

# Conectar nuevos nodos
@app.route('/connect_node', methods = ['POST']) # POST la usamos para
añadir info
def connect_node(): # crearemos un nuevo nodo y lo registraremos
    json = request.get_json()
    nodes = json.get('nodes') # obtenemos todos los nodos, nos
devolvera el address de cada nodo
    if nodes is None:
        return "No node", 400
    for node in nodes:
        blockchain.add_node(node)
    response = {'message': 'All the nodes are now connected. The
Hadcoin Blockchain now contains the following nodes:',
                'total_nodes': list(blockchain.nodes)}
    return jsonify(response), 201

# Sustitución de la cadena actual por la más larga
@app.route('/replace_chain', methods = ['GET'])
def replace_chain():
    is_chain_replaced = blockchain.replace_chain()
    if is_chain_replaced:
        response = {'message': 'The nodes had different chains so the
chain was replaced by the longest one.',
                    'new_chain': blockchain.chain}
    else:
        response = {'message': 'All good. The chain is the largest
one.',
                    'actual_chain': blockchain.chain}
    return jsonify(response), 200

# Ejecución de la app
app.run(host = '0.0.0.0', port = 5000)
```