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

Introducción **4**

Capítulo 1: Marco teórico **6**

1.1. Blockchain 6

1.1.1. Hash: SHA252 7

1.1.2. Descentralización P2P 7

1.1.3. Minería 8

1.2. Banco Central 10

1.3. Coins y stablecoin 11

1.4. Machine Learning 12

1.5. Deep Learning y redes neuronales 12

1.6. Inteligencia Artificial 13

1.6.1. Entorno 14

1.6.2. Agente 14

1.6.3. Recompensas 15

Capítulo 2: Metodología **16**

2.1. Descarga de datos 16

2.1.1. Datos históricos de Bitcoin 16

2.2. Construir entorno 16

2.2.1. Definir el entorno 16

2.2.2. Definir las acciones 17

2.3. Cerebro 17

2.3.1. Capa totalmente conectada 18

2.3.2. Función de activación: ReLU 18

2.3.3. Capa de Dropout 18

2.3.4. Función de perdidas: error cuadrático medio 19

2.3.5. Optimizador: Adam 19

2.4. Algoritmo de Deep Reinforcement Learning 19
2.4.1. Ecuación de Bellman.....19
2.4.2. Experiencia de repetición.....20
2.4.3. Ratio de aprendizaje.....20
2.4.4. Mecanismo de detención temprana.....21
2.5. Entrenamiento de la IA 21
2.6. Testeo de la IA 21
2.7. La IA en el blockchain.....22
Capítulo 3: Resultados 2**4**
3.1. Desempeño de la IA en el entrenamiento 24
3.2. Testeo de los agentes de IA 25
Capítulo 4: Discusión **30**
4.1. Uso del blockchain.....30
4.2. Debilidades del entorno 30
4.3. Limitaciones técnicas.....30
4.4. Trabajo futuro 31
4.5. Conclusión 31

ÍNDICE DE FIGURAS

Figura 1: Ejemplo de un bloque de una cadena 6
Figura 2: Comparación sistema centralizado y descentralizado 8
Figura 3: Arquitectura de red neuronal 13
Figura 4: Interacción de la IA 14
Figura 5: Gráfica de la función ReLU.....18
Figura 6: Ecuación de Bellman 20
Figura 7: Evolución de la capitalización de mercado de Bitcoin.....24

Figura 8: Capitalización de mercado de Bitcoin durante el entrenamiento 25

Figura 9: Capitalización de mercado de Bitcoin durante el testeo.....26

Figura 10: Precio durante el testeo del agente 1.....26

Figura 11: Porcentaje de éxito del agente 1 27

Figura 12: Precio durante el testeo del agente 2.....28

Figura 13: Porcentaje de éxito del agente 2 28

ÍNDICE DE CÓDIGO

Código 1: Entorno.....35

Código 2: Brain.....38

Código 3: DQN.....39

Código 4: Train.....40

Código 5: Test.....43

Código 6: Blockchain 45

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. 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. 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 algorítmico 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 empleando monedas digitales alternativas, 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 la metodología utilizada 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: ¿Cómo aprenden las máquinas y qué tipos de problemas resuelven?
- Deep Learning y redes neuronales: ¿Para qué sirven? ¿En qué consiste el aprendizaje profundo?
- Inteligencia Artificial: ¿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.

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, 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.

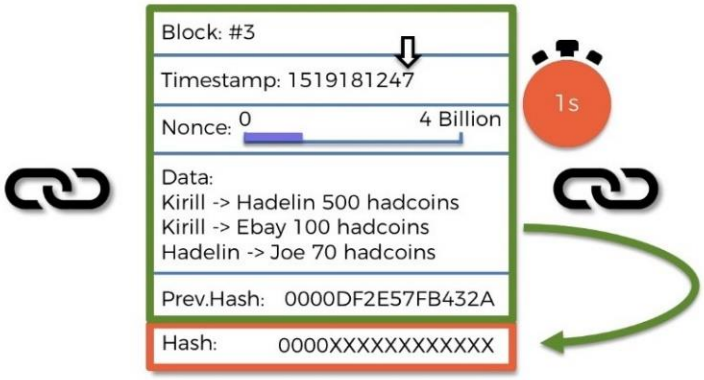

Figura 1: Ejemplo de un bloque de una cadena

1.1.1. Hash: SHA256

El hash 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 funciona para 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.

Cuando un bloque es añadido esto es comunicado al resto de computadores y validado por ellos. Posteriormente se añaden más bloques a medida que pasa el tiempo 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.

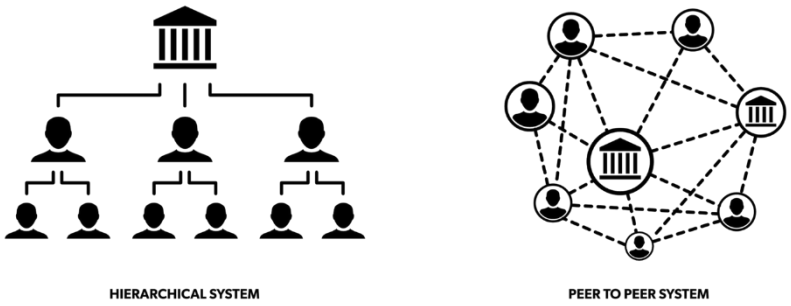

Figura 2: Comparación sistema centralizado y descentralizado

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.

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, por ejemplo. 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 cryptomonedas 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.

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.
- Las estrategias no colateralizadas que no asocian su valor a ningún activo externo, son que obedecen a un algoritmo para evitar las fluctuaciones del precio.

1.4. Inteligencia Artificial y Machine Learning

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 ML 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.

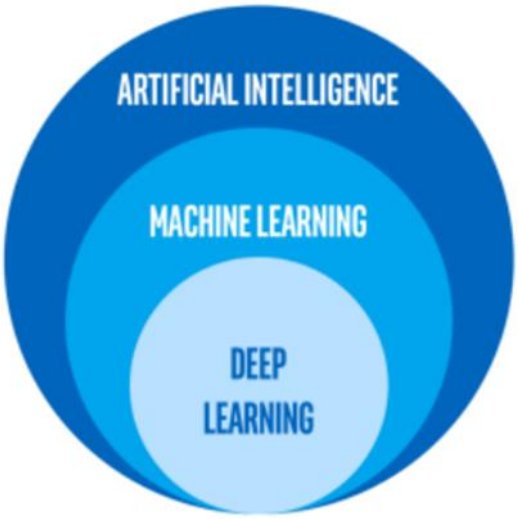

Figura 3: Ilustración de los campos de la IA

Dentro de los algoritmos de ML 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 que ha de predecir se dedicará a encontrar patrones. Por ejemplo, se podrían proporcionar fotos de prendas 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 rasgos concretos, 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 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.

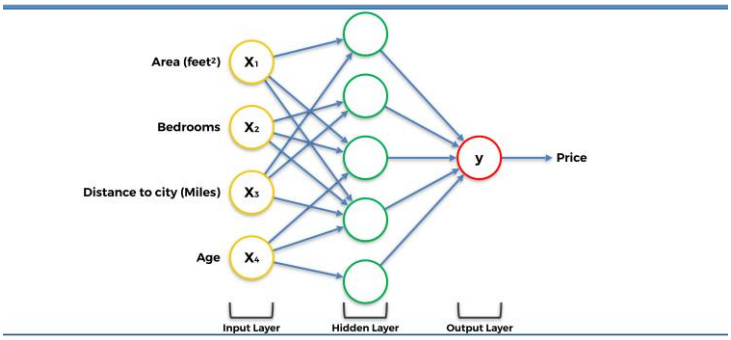

Figura 3: Arquitectura de red neuronal

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.

1.6. Reinforcement 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).

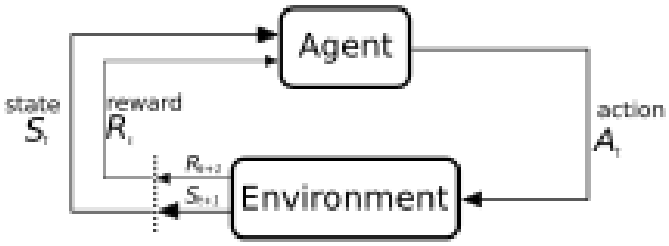

Figura 4: Interacción de la IA

1.6.1. 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.

1.6.2. Agente

Un agente de IA 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.

1.6.3. 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.

1.6.4. 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: Metodología

Este capítulo desarrolla los métodos utilizados para realizar el estudio. La metodología del estudio se puede dividir en siete secciones:

1. Descargar los datos.
2. Construir el entorno.
3. Construir el cerebro
4. Algoritmo de Deep Reinforcement Learning.
5. Entrenamiento de la IA.
6. Testeo de la IA.
7. La IA en el blockchain

2.1. Descarga de datos

Para realizar esta investigación se ha adquirido un único conjunto de datos. Este conjunto de datos consiste en el histórico de datos BTC/USD. Estos datos se adquirieron de blockchain.com por lo que son públicos.

2.1.1. Datos históricos de Bitcoin

Estos datos recogen la información diaria de bitcoin desde julio de 2010 hasta febrero de 2018. Los datos descargados el bitcoin se encuentra valorado en dólar americano. Las variables que encontramos utilices son la fecha, precio de mercado de bitcoin, número total de bitcoins, capitalización de mercado y volumen de transacciones.

2.2. Construir el entorno

En este caso, configuraremos nuestro propio entorno de blockchain y construiremos una IA que controlara el nº de coins que habrá en circulación para que se mantenga en un rango óptimo de precio al cambio con el dólar americano a lo largo de los días. Nuestro objetivo será crear la primera stablecoin sin ningún respaldo del cual dependa gracias al RL (Reinforcement Learning).

2.2.1. Definir el entorno

Antes de definir estados, las acciones y las recompensas, debemos explicar cómo funciona el entorno. Lo haremos en varios pasos. Primero, enumeraremos la suposición esencial del problema, en la cual nuestra IA dependerá para proporcionar una solución.

Luego especificaremos cómo simularemos todo el proceso. Y eventualmente explicaremos el funcionamiento general del blockchain y como la IA desempeña su papel.

La tarea de la IA es mantener un cambio estable de 1$ un coin. Para ello, le enseñaremos a la IA que si el cambio se mantiene entre 0.9 $ y 1.1$ la tarea ha sido resuelta satisfactoriamente. Para ello, la IA tendrá en cuenta el precio diario de bitcoin, la capitalización de mercado de bitcoin y el número de monedas en circulación. Una vez conocida toda esta información la IA ajustará el número de monedas. Siguiendo la ley de oferta y demanda incrementando el número de monedas en circulación la oferta será mayor y el precio se reducirá. Por el contrario, si la IA reduce el número de monedas en circulación el precio incrementará. Todos estos parámetros y variables serán parte de nuestro entorno de blockchain e influirán en las acciones de la IA en el blockchain.

Las acciones que tomará la IA vienen definidas en el código. En nuestro caso le propondremos a la IA 7 acciones posibles, aunque con un ordenador suficientemente potente se le podría proponer cientos de acciones.

Como se ha establecido en el marco teórico respecto al RL, estableceremos una recompensa cuando realice correctamente la tarea para que así aprenda a resolver el problema mediante prueba y error.

2.2.1. 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 entorno 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.

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

2.3. Cerebro

El cerebro consiste en la capacidad de aprendizaje del agente que viene dada por un modelo de Deep Learning. 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 de la IA utilizaremos la librería 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).

El cerebro, o más precisamente la red neuronal profunda de nuestra IA, será una red neuronal completamente conectada, compuesta de tres capas ocultas, la primera con 64 neuronas, la segunda de Dropout y la tercera con 32 neuronas. Y como recordatorio, esta red neuronal toma como entradas los estados del entorno y devuelve como salidas los valores Q para cada una de las 7 acciones. 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 IA está más segura de ser esa acción la indicada. De esta forma teniendo en cuenta todos los valores Q elegirá el mayor de todos ellos.

2.3.1. 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.

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

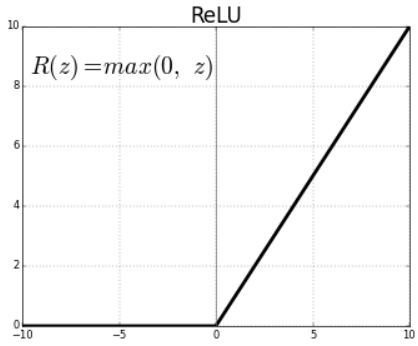

Figura 5: Gráfica de la función ReLU

2.3.3. 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

2.3.4. 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 la precisión de la predicción en la red neuronal. Esta métrica 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.

$$MSE = \frac{1}{n}(\mathrm{Y}' - \mathrm{Y})$$

2.3.5. 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).

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

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

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

2.4. Algoritmo de Deep Reinforcement Learning

Como vemos nuestro agente de IA 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 tomara decisiones para entender que efectos produce.

2.4.1. Ecuación de Bellman

A continuacion se muestra la ecuacion de Bellman que resume el aprendizaje de la inteligencia artificial (Juliani, 2016). En ella podemos observar como el valor de estado nuevo se actualiza con una ecuación muy sencilla donde se realiza una ponderación dada por el *learning rate* o ratio de aprendizaje entre el valor antiguo y el nuevo valor estimado por la inteligencia. Tambien podemos obsevar que la recompensa forma parte de esta

ecuanción y el *discount factor* o factor de descuento determina la importancia de la recompensas futuras.

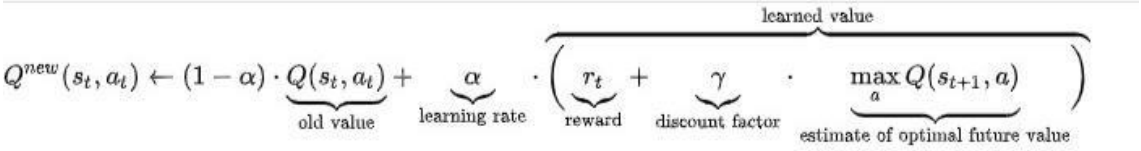

Figura 6: Ecuación de Bellman

En caso de que el resultado de sus decisiones se ajuste al objetivo que le hemos marcado de mantener el precio entre 0.9-1.1$ el agente recibirá una recompensa y repetirá esa decisión en el futuro. En caso contrario, si toma una decisión perjudicial recibirá una recompensa negativa o castigo y será menos probable que tome esa decisión en el futuro.

2.4.2. 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). 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. En nuestro caso, permitir que la IA tenga memoria le dará la capacidad de reaccionar más adecuadamente ante ciertos días en los que hay un exceso de demanda de monedas como puede ser en navidad o cuando una empresa adquiera un gran volumen de monedas para su operativa.

Para que la experiencia de rejugado sea suficiente necesitaremos que guarde en memoria al menos 3000 experiencias. La ventaja de la inteligencia artificial es que puede simular varias veces un mismo escenario y enfrentarse al mismo problema múltiples veces como si de un nivel del Mario Bross se tratase.

2.4.3. Ratio 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. Por converger entendemos que el modelo alcanza un punto donde no puede aprender más por lo que si seguimos entrenando terminará por aprenderse el problema de memoria y no generalizará bien para problemas futuros a los que aún no se ha enfrentado. En nuestro caso lo conveniente sería apostar por un ratio de aprendizaje cercano a 0. La desventaja es que tardara más en converger, pero a su vez conseguiremos un aprendizaje más preciso conseguiríamos resultados mejores.

Por otro lado, implementaremos un ratio de aprendizaje dinámico, es decir, un ratio que comenzará 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:

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

2.4.4. Mecanismo de detención temprana

Aun así, nos seguimos encontrando con la pregunta de cuando deberíamos dejar de entrenar la inteligencia artificial. Para ello, estableceremos un mecanismo llamado *early stopping* o detención temprana (Prechelt, 2015). El mecanismo es muy sencillo, la IA guardará en memoria un registro de las recompensas que va obteniendo por cada vez que se enfrenta a la simulación y se actualizará cada vez que consiga una recompensa superior a las anteriores. A la vez habrá un contador 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 Artificial dejará de entrenarse. Por ejemplo, la paciencia podrían ser 100 simulaciones por lo que en cuanto pasen 100 simulaciones sin mejorar la recompensa la IA habrá terminado su entrenamiento y estará lista para implementarla en un blockchain.

2.5. Entrenamiento de la IA

El entrenamiento es la fase donde el agente de IA aprenderá a comportarse en el entorno que le hemos preparado. El entrenamiento consistirá en simular de forma iterativa los datos que tenemos en el entorno. En cada iteración el agente tomará una decisión y el entorno reaccionará con una recompensa. El aprendizaje se lleva a cabo en el cerebro del agente en dos fases:

- Fase de avance (*feed fordward*). Las conexiones entre neuronas vienen a representar unos pesos que transforman los datos de entradas para devolver una salida (Sazli, 2006).

- Fase de retroceso (*backpropagation*). Una vez tomada una acción se produce un error que es la diferencia respecto de la acción correcta. Ese error se propaga capa por capa en la red neuronal empezando por la salida hasta la entrada. A lo largo de esta fase se ajustan los pesos de las conexiones de la red neuronal (Çavuşoğlu, 2020).

2.6. Testeo de la IA

Ahora, de hecho, tenemos que probar el rendimiento de nuestra IA en una situación completamente nueva. Para hacerlo, ejecutaremos una simulación de 200 días, solo en modo de inferencia, lo que significa que no habrá entrenamiento en ningún momento. Nuestra IA 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 de IA, aquí para la implementación de prueba casi tenemos lo mismo que antes, excepto que esta vez, no tenemos que crear un 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 de la IA.

2.7. La IA en el Blockchain

En último lugar, combinaremos la tecnología blockchain con la Inteligencia Artificial. Como explicamos en el marco teórico los bancos centrales ponen o quitan dinero en circulación gracias a la intermediación de los bancos. De la misma forma nuestra IA 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 IA 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 IA 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 IA 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 IA 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 IA ha sido entrenada para estimar qué número de monedas en circulación debe variar para estabilizar su precio. Teniendo en cuenta los registros de diarios calculará dicha variación. Obtenida esa variación calculará un multiplicador para hacer la proporción con la suma total de tasas que guarda en memoria. Calculado el multiplicador se aplicará individualmente a cada tasa y se publicarán las recompensas de los mineros.

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 dólar será cuestión de horas que el precio lo actualice la IA 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 IA el precio del coin se actualizará 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.

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 de la IA en el entorno. Esto nos ayudará a descubrir las distintas estrategias que adoptará la IA 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 o token 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.

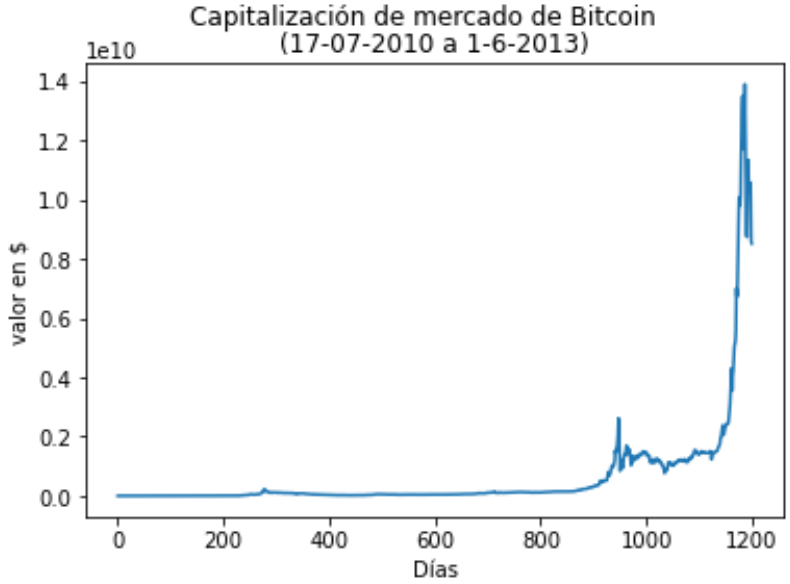

Figura 7: Evolución de la capitalización de mercado de Bitcoin

3.1. Desempeño de la IA en el entrenamiento

En la fase de entrenamiento realizaremos el seguimiento de la cantidad de monedas en circulación, el precio medio, las recompensas que consiguen y el número de días que el precio fue estable.

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.

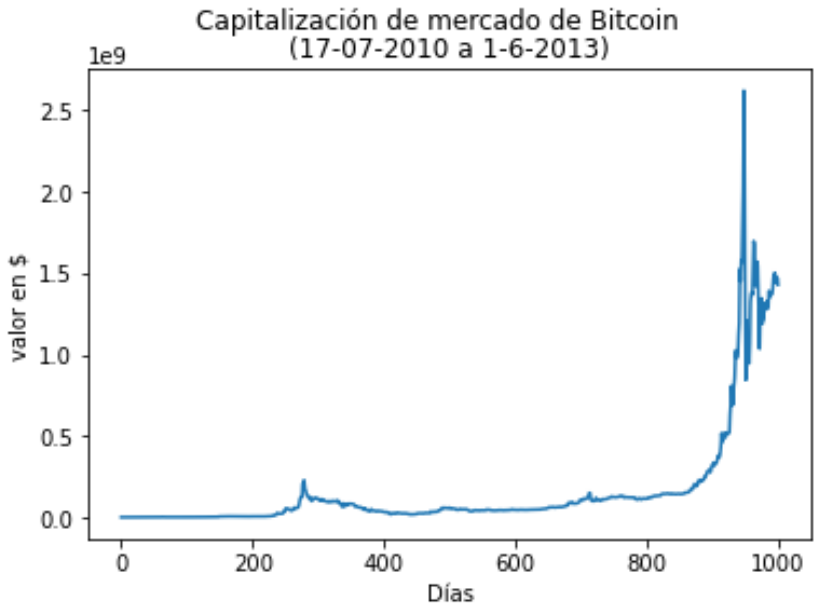

Figura 8: Capitalización de mercado de Bitcoin durante el entrenamiento

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 cryptomonedas que salen al mercado por primera vez tienen precios muy bajos.

De esta forma podemos simular y analizar también cómo funcionaría la puesta en marcha de esta IA en una nueva moneda.

Figura 9: 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.

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.

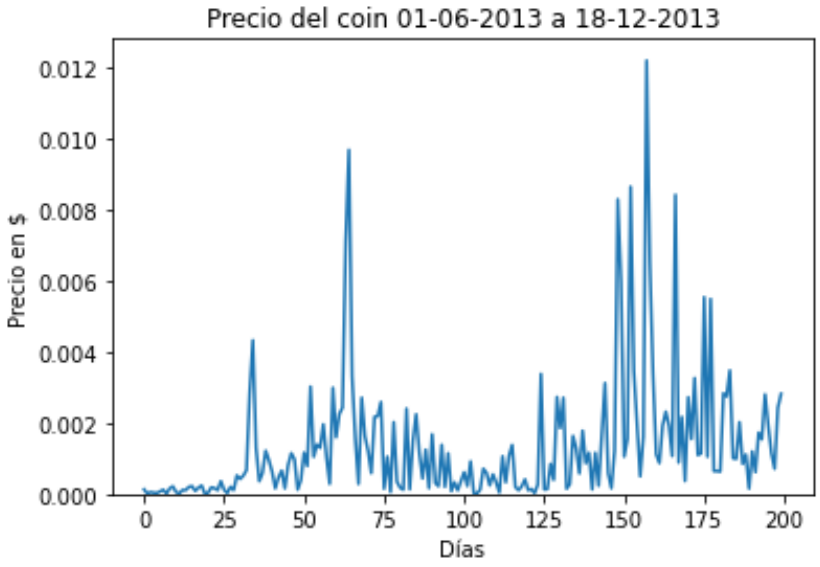

Figura 10: Precio durante el testeo del agente 1

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$.

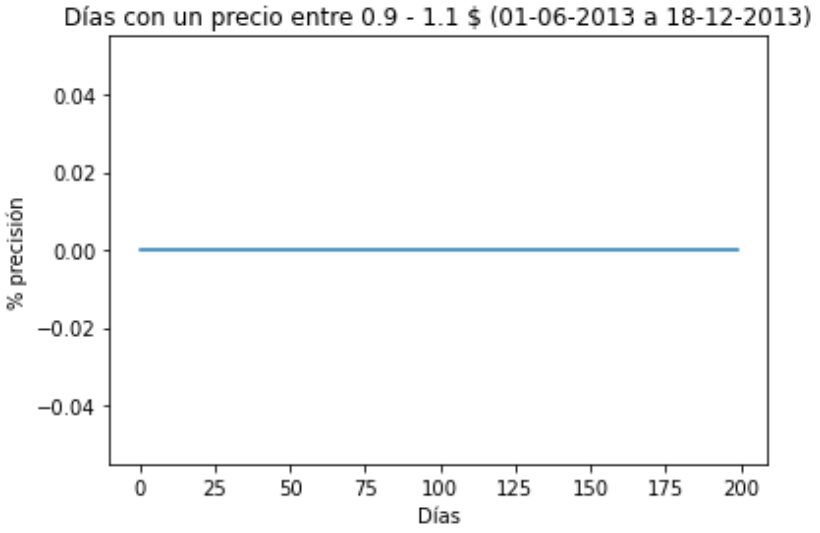

Figura 11: Porcentaje de éxito del agente 1

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

De estas graficas podemos sacar dos conclusiones principalmente.

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. Las 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, por debajo de 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.

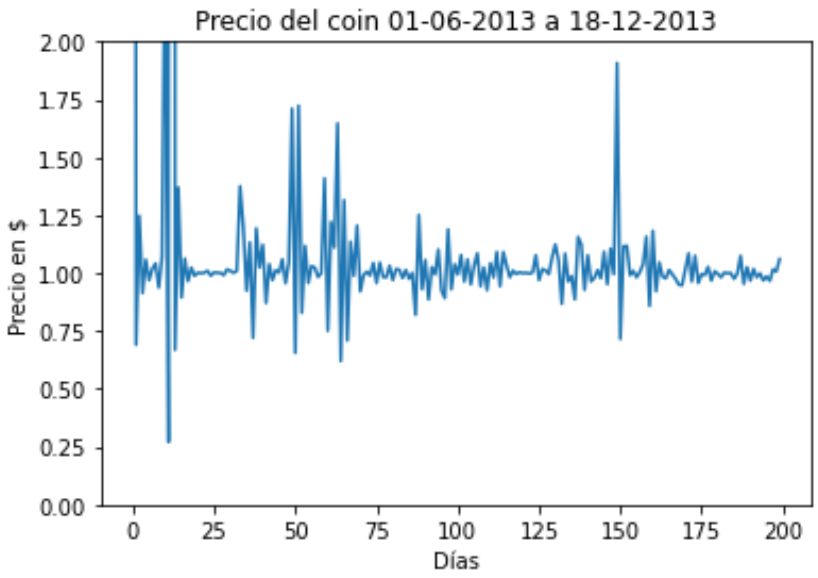

Figura 12: Precio durante el testeo del agente 2

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.

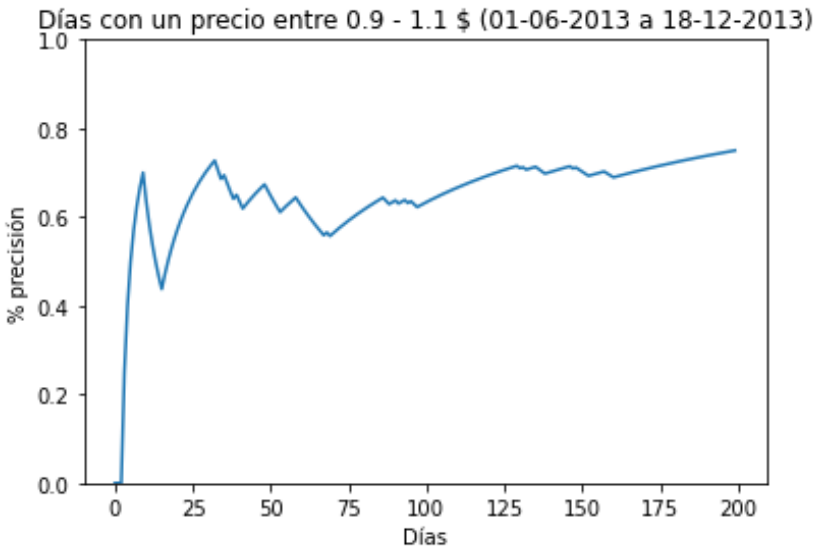

Figura 13: Porcentaje de éxito del agente 2

En esta gráfica se muestra la evolución del porcentaje de días que logra mantener el precio dentro del intervalo 0.9 – 1.1$. En línea con la gráfica del precio, el agente de IA parte de un mercado desajustado donde tiene que ir tomando políticas monetarias. Durante los primeros 75 días apenas consigue estabilizar la moneda ni el 60% de los días. Sin embargo, de ahí en adelante el agente se ve que comienza a estabilizar mejor la moneda y mejorar progresivamente el porcentaje hasta alcanzar un 75% de días con un precio cercano a 1$, 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$.

Las principales conclusiones que podemos sacar son las siguientes:

En primer lugar, el desempeño del 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.

En segundo lugar, ha sido capaz de partir de un escenario donde la moneda acababa de salir al mercado. Por lo que, está preparada para ser puesta en un blockchain desde que se haga público.

En tercer lugar, en mercados alcistas hemos podido comprobar que el precio lo mantenía estable sin ningún problema. Y cuando vino la corrección del mercado después del pico también conseguía mantener el precio dentro del intervalo óptimo.

En conclusión, la introducción de una recompensa positiva en el aprendizaje del agente de IA 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 una 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 *Deep 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 de la IA 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 interesante 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* podría contribuir a modificar el comportamiento de los *holders* 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.

4.5. Conclusión

En conclusión, en este artículo se estudia la implementación una inteligencia artificial en un blockchain para crear una criptomoneda estable vinculada al valor del dólar. 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 artículo 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 otro tipo de divisas o valores y conseguir resultados similares. El motivo por el que se ha asociado al USD es porque es la divisa referente a nivel mundial.

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*.

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

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

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.

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. 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

Figura 2. Obtenido de Udemy: https://www.udemy.com/course/build-your-blockchain-

Figura 3. Obtenido de Obtenido de Udemy: https://www.udemy.com/course/deep-learning-a-z/az/

Figura 4. Obtenido de Wikipedia: https://en.wikipedia.org/wiki/Deep_reinforcement_learning

Figura 5. Obtenido de Precious Chima: https://medium.com/@preshchima/activation-functions-relu-softmax-87145bf39288.

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), opt_nvt = 15, data
= data):
        self.data = data
        self.optimal_price = optimal_price
        self.optimal_nvt= opt_nvt
        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_nvt = self.current_market_cap /
self.current_trading_volume
        self.current_nvt = self.initial_nvt
        self.initial_supplyai = data.iloc[0,1]
        self.coin_supplyai = self.initial_supplyai
        self.price_ai = (self.current_market_cap/self.current_nvt) /
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/15)/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_nvt = self.current_market_cap /
self.current_trading_volume
        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.current_nvt = self.initial_nvt
        self.price_ai = (self.current_market_cap/self.current_nvt)/
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 = 1
max_memory = 10
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
        timestep = np.random.randint(1,800)
        current_state, _, _ = env.observe(timestep)
        stop = timestep + 300
        env.current_date = 0
        # INICIALIZACIÓN DEL BUCLE DE TIMESTEPS (Timestep = 1 día) EN
UNA ÉPOCA
        while ((not game_over) and (timestep <= stop)):
            # 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.current_nvt)/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_nvt.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_nvt.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((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)
    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: Blockchain

```
# Importing the libraries
import datetime
import hashlib
import json
from flask import Flask, jsonify, request
import requests # utilizaremos paar conectar nodos
from uuid import uuid4 # crear un addres para cada nodo
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 - Building a 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_nvt.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

    def reward_demo(self):
        fee_rate = 0.8
        for i in range(0,len(self.transactions_fees)):
            self.transactions_fees[i]['fee'] =
self.transactions_fees[i]['fee']*fee_rate


# Part 2 - Mineria en nuestro 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': 'Houston, we have a problem. 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

# Connecting new nodes
@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)
```