Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Informática

Trabajo Fin de Grado

Estudio de la capacidad predictiva de redes neuronales profundas sobre la cotización de criptomonedas

Autor: Jorge Cabrejas Marina

Tutor: David Fernández Barerro

Cotutor: Mario Cobos Maestre

2022

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

Trabajo Fin de Grado

Estudio de la capacidad predictiva de redes neuronales profundas sobre la cotización de criptomonedas

Autor: Jorge Cabrejas Marina

Tutor: David Fernández Barerro

Cotutor: Mario Cobos Maestre

Tribunal:

Presidente: Julia María Clemente Párraga

Vocal 1º: María del Mar Lendínez Chica

Vocal 2º: David Fernández Barrero

Fecha de depósito: 15 de septiembre de 2022

A nuestros alumnos pasados, presentes y futuros...

"Empieza haciendo lo necesario, luego haz lo posible y de pronto empezarás a hacer lo imposible."
Francisco de Asís

Agradecimientos

Last but not least, I want to thank me.

Snoop Doggy Dog

Este trabajo supone el fin de 4 años maravillosos, los cuales no podrían haber sido posibles si no hubiese sido por la ayuda de muchisima gente. No puedo irme sin antes agradecer a David y Mario, por todo lo que me han hecho aprender a lo largo de estos meses, a mis padres, por el apoyo que he tenido en los momentos de mayor dificultad a lo largo de estos años, a María Dolores, que invirtió su propio tiempo en ayudarme sin tener ninguna obligación, a todos mis amigos de clase, que hicieron que realmente el proceso mereciese la pena, a Berti, por ser mi compañero en las etapas más duras de este proceso, a mis amigos, Simon, Miguel y Nerea, por haber pasado horas de su tiempo leyendo este trabajo de fin de grado sin ni siquiera saber que es un bucle for, y a todos mis amigos, que me apoyaron en el proceso y me ayudaron a superar todas las dificultades que encontré en el camino.

Gracias a todos vosotros, este trabajo también es vuestro.

Resumen

Con la reciente aparición de las criptomonedas como activo de inversión, método de pago e intercambio de valor entre individuos, y su reciente y creciente adopción, indudablemente se han vuelto un instrumento financiero de peso de creciente importancia que día tras día tiene mayor relevancia dentro la vida de las personas. Numerosas plataformas donde comprar, vender e intercambiar las distintas criptomonedas surgen día a día, así como nuevas maneras que facilitan el acceso a gente con pocos conocimientos sobre ellas. A su vez, cada día más empresas invierten en la creación de sistemas que usan las mismas tecnologías que estas criptomonedas, creando un nuevo abanico de posibilidades en cuanto a innovación, permitiendo la formación de un mundo descentralizado donde la aparición de terceros es innecesaria para su manejo.

Esto abre la puerta a la pregunta de si realmente es posible predecir cual va a ser el precio de las mismas en el futuro, hecho que es de especial relevancia en el mundo de la informática y las matemáticas aplicadas, surgiendo la duda de si las diferentes técnicas y modelos usados en otros problemas similares, como la bolsa de valores, son aplicables en este ámbito.

Con esta premisa, este proyecto busca analizar distintos grupos de criptomonedas para posteriormente aplicar diferentes métodos pertenecientes al Machine Learning y el Deep Learning a las mismas, a efectos de determinar si alguna de estas posee unas características concretas que faciliten el proceso predictivo que se aplicará sobre las mismas.

Palabras clave: Métodos Autorregresivos, Machine Learning, Deep Learning, Criptomonedas, Redes Neuronales.

Abstract

With the recent growth of cryptocurrencies as an investment, way of payment and value exchange method and theirs increasingly adoption, they have undoubtedly become an important financial tool that each and every day has more and more importance in the life of people. Several different platforms where to buy, sell and exchange numerous cryptocurrencies are born every day, as many ways for people to improve their knowledge about them. Also, everyday new companies invest in the creation of systems based on the same technologies as these cryptocurrencies, creating a new range of possibilities in matters of innovation, making it possible for a decentralized world where third-parties are unnecessary for its management to be born.

This opens the door to the question: is it really possible to predict what their price is going to be in the future? This is notably relevant in the world of Computer Science and Applied Mathematics, making us think if the different techniques and models used in other similar problems, as the stock market, are applicable in this field.

With this premise, this project looks forward to analyze different groups of cryptocurrencies to afterwards apply different methods from Machine Learning and Deep Learning to them, with the purpose of determining if any of them possess some specific characteristics which ease the predictive labour that will be applied to them.

Keywords: Autoregresive Methods, Machine Learning, Deep Learning, Cryptocurrencies, Neural Networks.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Desocupado lector, sin juramento me podrás creer que quisiera que este libro [...] fuera el más hermoso, el más gallardo y más discreto que pudiera imaginarse

Miguel de Cervantes, Don Quijote de la Mancha

1.1 Motivación

Desde la creación del protocolo Bitcoin en 2008 hasta el día de hoy, el interés sobre la creación de métodos de pago alternativos a los usuales sin la necesidad de participación de terceros, ha crecido enormemente. Si bien el objetivo inicial de estas propuestas no ha sido cumplido por el momento, la creciente adopción y uso de las mismas las ha convertido en activos que han sido, y son, objeto de especulación a diario.

Durante los últimos años, el uso y adopción de las criptomonedas ha crecido de manera muy notable, abriéndose paso en la vida de un gran número de personas. Numerosas empresas conocidas como *exchanges* han surgido para facilitar la compra, venta e intercambio de estas criptomonedas a usuarios no familiarizados con entornos informáticos, como pudieran ser antes de la creación de estas plataformas. Hoy en día, podemos ver publicidad de los *exchanges* y las criptomonedas que venden en televisión, estadios y equipos de fútbol. También, desgraciadamente, hemos visto como distintas estafas han surgido aprovechando el desconocimiento del usuario medio para lucrarse de manera ilegítima. Esto remarca la importancia de la educación y el conocimiento de este tipo de activos, que si son usados de la manera correcta, pueden convertirse en mecanismos muy útiles, ya que aportan la posibilidad de disponer de un método de intercambio de valor entre personas sin la necesidad de un tercero que gestione las transacciones, papel que ocupan los bancos en el uso del dinero fiat.

De esta manera, dada la naturaleza especulativa de las mismas, con objeto de disminuir el impacto de la volatilidad inherente a su comportamiento bursátil [3] se han realizado distintos trabajos para intentar inferir estadísticamente el precio de dichas criptomonedas, y especialmente Bitcoin, usando diferentes técnicas pertenecientes a la Inteligencia Artificial y al *Machine Learning* (ML). Estos métodos difieren de otros más clásicos ampliamente utilizados en el mundo de la bolsa para el mismo objetivo, como puede ser el modelo ARIMA, que son fácilmente superados por métodos pertenecientes al ámbito del ML [4].

1.2 Objetivos a conseguir

Tal y como ha ocurrido con una gran cantidad de otros activos desde la creación de la bolsa de valores, la especulación abre la puerta a la pregunta: ¿es posible estimar su precio de cotización?. Si la respuesta a esta pregunta fuese positiva, tendriamos en nuestras manos una muy poderosa herramienta a la hora de monitorizar los mercados financieros.

En este proyecto se busca estudiar distintas criptomonedas, para elegir un subconjunto y aplicar un análisis más exhaustivo sobre dicho conjunto. Tras esto, se crearán unas métricas de desempeño para evaluar el rendimiento de distintas técnicas predictivas, y la ejecución de dichas técnicas sobre el conjunto dado, creando una arquitectura común para la realización de la labor comparativa. Posteriormente se procederá a el análisis de los resultados obtenidos para la identificación de cuales criptomonedas son más facilmente predecibles, indentificando distintos factores que faciliten o compliquen la labor predictiva sobre los conjuntos de datos elegidos.

Con estos objetivos en mente, se han descargado los datos históricos de 1837 criptomonedas desde el API de Binance y realizado un análisis exploratorio sobre un conjunto elegido. Sobre este, se han aplicado diversos métodos pertenecientes al ML y al *Deep Learning* (DL), en concreto, una red MLP (del inglés *Multilayer Perceptron*), y una red profunda recurrente haciendo uso una capa LSTM (del inglés *Long-Short Term Memory*).

1.3 Organización

Para la consecución de dichos objetivos, en el Capítulo 2 se realizará un estudio del Estado del Arte en el modelado predictivo de instrumentos financieros, así como se profundizará en algunos de los métodos usados más relevantes, y en conceptos esenciales para la comprensión de las criptomonedas. En el Capítulo 3 se explorarán los métodos aplicados y los pasos seguidos para la consecución de los objetivos que se plantean en este estudio. En el Capítulo 4 se expondrán los resultados experimentales obtenidos y se explorarán sus posibles consecuencias. Por último, en el Capítulo 5 se expondrán las conclusiones e hipótesis que pudieran derivarse del análisis de los resultados obtenidos.

Capítulo 2

Marco teórico

Y así, del mucho leer y del poco dormir, se le secó el cerebro de manera que vino a perder el juicio

Miguel de Cervantes Saavedra

En este capítulo se dará un análisis sobre el Estado del Arte en el modelado predictivo de instrumentos financieros, y en concreto, criptomonedas. También se aportará una descripción de las herramientas que se han utilizado en este proyecto, así como algunos de los conceptos más importantes para entender el ámbito en el que se desenvuelve este estudio.

2.1 Estado del Arte

El estudio del modelado predictivo sobre distintos instrumentos financieros ha sido ampliamente tratado en la literatura, lo cual no ocurre en el caso concreto de las criptomonedas y productos de alta volatilidad [5]. Las técnicas más utilizadas en este campo se basan en la predicción según los datos históricos de la serie temporal mediante el uso de modelos lineales, autorregresivos (ARIMA, GARCH, etc.), y más recientemente, con los avances en el campo del ML y el DL, se han aplicado nuevos enfoques que han llegado a superar a métodos clásicos en el modelado predictivo de series temporales. Entre estos nuevos enfoques cabe destacar el uso de Redes Neuronales Profundas basadas en distintos modelos, tales como RNN (del inglés *Recurrent Neural Networks*) o capas como LSTM.

Dentro de la literatura sobre el modelado predictivo de activos financieros, encontramos numerosos estudios aplicando métodos pertenecientes al campo de la econometría. Kjaerland *et al.* [6] buscan encontrar los principales elementos que guían el comportamiento del precio del Bitcoin, aplicando modelos GARCH (del inglés *Generalized Autoregressive Conditional Heteroskedasticity*) y ARDL (del inglés *Autoregressive Distributed Lag*). En este estudio los autores concluyen que es necesario excluir el *hashrate* (velocidad a la que la red de Bitcoin trabaja) de la red dentro del modelado predictivo. También concluyen que el sentimiento de mercado actual, los datos históricos sobre el precio de Bitcoin y el volumen de búsquedas del término 'bitcoin' en Google, tienen una relación muy estrecha con las dinámicas del precio de Bitcoin. Troster *et al.* [7] utilizan modelos GARCH y GAS (del inglés *Generalized Autoregressive Score*) para el análisis de retornos y riesgos. Los resultados muestran que el modelo GAS con una distribución con colas pesadas produce la mejor predicción. Bakar y Rosbi [8] aplican un modelo ARIMA (del inglés

Autorregresive Integrated Moving Average) a la predicción del precio del Bitcoin. Los resultados muestran que ARIMA produce un modelo predictivo fiable a corto plazo, pero que el error de este necesita consideraciones especiales en entornos de alta volatilidad, ya que tiende a crecer.

En estudios posteriores, se comienza a comparar el rendimiento de estos métodos con los avances en el campo del ML y el DL. Uras *et al.* [9] realizan una comparativa entre dos modelos regresivos lineales, simple y múltiple; y dos Redes Neuronales, un modelo MLP (del inglés *Multi-Layer Perceptron*) y un modelo LSTM. En los modelos lineales, para el simple se usó únicamente el precio de cierre de Bitcoin, mientras que en el múltiple se incluye también el volumen de transacciones. Los resultados reflejan una clara mejora por parte de las redes neuronales frente a los modelos lineales, obteniendo valores de error significativamente menores. Munim, Shakil y Alon [10] estudian la capacidad predictiva de un modelo ARIMA frente Redes Neuronales Autorregresivas (NNAR) a la hora de predecir el precio del Bitcoin. Se utilizan con este propósito distintos conjuntos de test y entrenamiento. Este estudio es realizado de dos maneras, directa y reestimando el modelo predictivo a cada paso. Los resultados exponen que los modelos NNAR superan al modelo ARIMA en periodos de baja volatilidad sin reestimación, que el modelo ARIMA produce resultados similares en periodos de tanto alta como baja volatilidad, con o sin reestimación, y que los modelos NNAR producen mejores resultados cuando se aplica reestimación. Mangla *et al.* [11] predice el precio de Bitcoin mediante un modelo ARIMA, un modelo SVM (del inglés *Support Vector Machine*), un modelo basado en Regresión Logística y Redes Neuronales Recurrentes (RNN). Se puede observar que el modelo ARIMA es el que tiene un mayor acierto, sobre el 53 % en predicciones a corto plazo, pero su funcionamiento empeora en periodos de tiempo más largos. En cambio, las RNN predijeron de manera consistente desde un corto plazo hasta los 6 días. El peor caso es el de la Regresión Logística, con un acierto del 47 %, por detrás de las SVM, que solo es capaz de clasificar de manera efectiva en el caso de que exista un hiperplano separable.

En cuanto a la aplicación de redes neuronales, y en concreto el uso del modelo LSTM, cabe destacar a Salman *et al.* [12], que aplica este mismo método a la predicción climática, obteniendo un valor de precisión de alrededor del 85 %. Roondiwala *et al.* [13], que aplican este mismo a la bolsa de valores, que obtienen una predicción razonable en términos de RMSE (del inglés *Root Mean Squared Error*). Shah *et al.* [14] realizan un estudio comparativo entre Redes Neuronales Profundas y Redes Neuronales Recurrentes, en concreto LSTM. Ambas redes tienen un rendimiento razonable en una predicción diaria, mientras que el modelo LSTM realiza una mejor predicción a nivel de semana, teniendo en cuenta que se utilizaron datos reales para predecir a nivel semanal en vez de las propias predicciones del modelo. Se concluye que LSTM es superior en el modelado predictivo de series temporales en la mayoría de los casos. Siami-Namini *et al.* [15] presentan una comparativa entre los modelos ARIMA y LSTM en la predicción de series temporales. Los resultados reflejan una mejora considerable por parte del modelo LSTM a las predicciones realizadas por ARIMA.

2.2 Criptomonedas

Las criptomonedas son un activo digital almacenado en una red conocida como *blockchain* o cadena de bloques. La aparición de estas se remonta al año 2008, cuando un hombre llamado Satoshi Nakamoto (el cual se presupone es un nombre falso) publicó un estudio en el que hablaba de un sistema de dinero electrónico *peer-to-peer* [16]. Su objetivo era la creación un sistema de pago electrónico, que no tuviese que ser gestionado por una entidad financiera intermedia mientras que al mismo tiempo pretendía prevenir el problema del doble gasto. Actualmente las mismas no cuentan con la consideración de ser un método de pago oficial al no estar refrendadas por un banco central (hecho el cual era uno de los objetivos de las mismas). Pese a ello, han demostrado ser una reserva de valor efectiva. Sin embargo, existe debate al

respecto de si son una forma fiable de almacenar este valor, frente a otras alternativas, tales como el oro, que ha cumplido esta función a lo largo de los años. Estas son almacenadas en una cartera digital, que consiste en un registro dentro de la red, a través de la cual se pueden trazar las transacciones realizadas.

En cuanto al modelado predictivo de estas, debido a lo novedoso del campo en el que nos encontramos, se ha enfocado especialmente sobre la principal de estas, Bitcoin. Estudios como el de Chen, Li y Sun [17] aplican distintos métodos pertenecientes al ML con este objetivo. Otros como Akyildirim, Goncu y Sensoy [18] realizaron un estudio sobre las 12 criptomonedas con más liquidez de su momento, aplicando distintos métodos, como Redes Neuronales Artificiales (ANN, del inglés *Artificial Neural Networks*) o Máquinas de Vectores de Soporte (SVM, del inglés *Support Vector Machines*), obteniendo resultados entorno al 55-65 % de acierto. A continuación, se procede a indagar en la tecnología sobre la que se apoyan a la mayoría de esta clase de activos.

Blockchain, o cadena de bloques en español, es la tecnología subyacente a la mayoría de criptomonedas actualmente. Cabe destacar el caso de IOTA como criptomoneda que no utiliza esta tecnología, enfocada al mundo del IoT y que basa su protocolo conocido como *The Tangle* en Grafos Acíclicos Dirigidos. La *blockchain* es una base de datos pública, distribuida y descentralizada basada en un protocolo P2P, que guarda sus registros en una estructura de datos conocida como bloque. Estos bloques se unen respectivamente siguiendo una estructura de árbol, en la que cada nodo es etiquetado con el *hash* del bloque anterior, conocido como Árboles de Merkle. En la Figura 2.1 se muestra la estructura que sigue un árbol de Merkle.

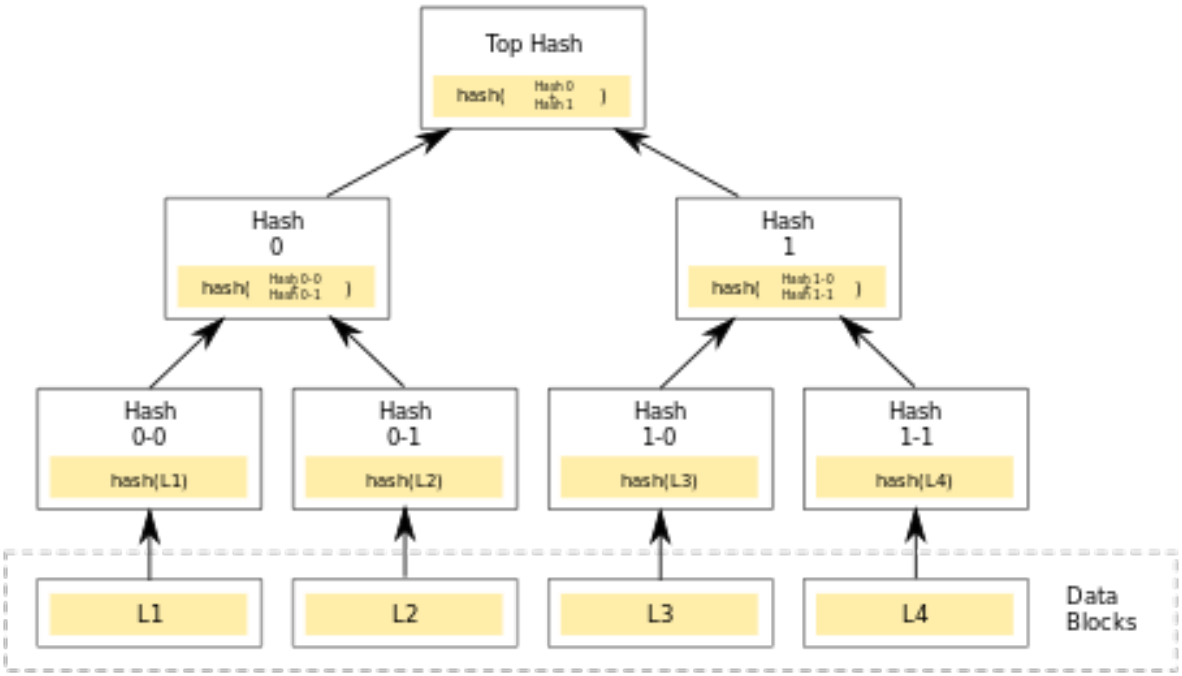


Figura 2.1: Árbol de Merkle. Obtenido de
https://es.wikipedia.org/wiki/%C3%81rbol_de_Merkle

Esta tecnología, si bien no es especialmente eficiente para el almacenamiento de información, posee una cualidad enormemente útil que es capaz de compensar dicha ineficiencia. Esto es, otorgar a un usuario la capacidad de confiar en cualquier otro usuario de la red sin conocer absolutamente nada sobre él. Esto se debe al proceso que se utiliza para añadir nuevos registros a la red, conocido como *Proof of Work* o

Prueba de Trabajo. Esto es especialmente importante porque ha sido el primer sistema que ha permitido resolver el problema del doble gasto, sin que una entidad intermediaria intervenga en las transacciones de la red. Este problema, consiste en la posibilidad de un fallo en el sistema que permita que una moneda digital pueda ser gastada más de una vez.

Proof of Work es una prueba criptográfica en la que una de las partes (el *probador*) prueba a otros (los *verificadores*) que una cierta cantidad de poder computacional (*hashrate*) ha sido utilizado, siendo los verificadores capaces a su vez de comprobarlo empleando una cantidad mínima de recursos. Generalmente, en el caso de las criptomonedas, y en concreto de Bitcoin, esto se hace de la siguiente manera: en primer lugar, cuando una transacción se produce, con el objetivo de validarla e incluirla en la base de datos, se busca obtener un *hash* de dicha transacción que cumpla unos requisitos, en este caso, que la cadena de texto empiece por una cantidad arbitraria de ceros (p.e. 00000000000000anye1n628nd82 %72bda). Para esto, se añade al final de los datos de la transacción un número conocido como *nonce*. Una vez una persona lo ha conseguido, el resto de los usuarios de la red pueden comprobar por su cuenta que esto es correcto, de tal manera que es añadido a la red. Para que los conocidos como *mineros* (usuarios que invierten recursos en el proceso de validación de transacciones) quieran realizar este trabajo, al minero que consiga encontrar primero el *hash* que cumple los requisitos, se le otorga una recompensa. De esta manera, para poder realizar alguna acción fraudulenta sobre la red, por ejemplo, introduciendo una transacción falsa para obtener fondos de manera ilícita, se debe contar con al menos el 51 % del poder computacional de la red.

Esto ha causado que exista una competición entre los mineros, haciendo que los mismos tengan interés en que este proceso sea legítimo, ya que, de otra manera, no podrían obtener la recompensa asociada a este trabajo. No obstante, existen otros métodos para alcanzar este objetivo, donde cabe destacar *Proof of Stake*, ya que la que es la segunda criptomoneda en capitalización de mercado actualmente, y a su vez, la red donde más avances e innovación se producen día a día, Ethereum, planea migrar su sistema de *Proof of Work* a *Proof of Stake* en los próximos meses. El propio *Proof of Stake* se basa en la necesidad de tener como mínimo un cierto número de criptomonedas acumuladas para poder validar la transacción, evitando la competición y el gasto energético que lleva asociado. Para evitar que la gente con un mayor poder adquisitivo adquiera el control sobre la red, existen varias medidas, por ejemplo, cada vez que un usuario valida una transacción, se reduce la probabilidad de que pueda validar la siguiente. También se permite la posibilidad de que varios usuarios unan sus fondos para alcanzar el mínimo necesario para poder validar transacciones [1].

Podemos distinguir así, distintas ventajas y desventajas en este enfoque. Entre las ventajas cabe destacar la gran seguridad que ofrece, ya que toda la información de la red está replicada en cada uno sus nodos, haciéndola prácticamente imposible de modificar. Por otro lado, la gran velocidad y disponibilidad que ofrece, gracias a la carencia de intermediarios, permite que sea el propio sistema el que valida las transacciones y lo hace disponible los 365 días del año. También cabe destacar la transparencia que ofrece este sistema, al ser una base de datos pública que cualquiera puede consultar (protegiendo la privacidad de los usuarios identificándolos mediante un seudónimo), y su inmutabilidad, ya que al hacer uso de árboles de Merkle es imposible cambiar un bloque, ya que esto, alteraría todos los *hashes* de los bloques anteriores en la red.

Entre sus desventajas encontramos los altos costes que conlleva, dado el uso de Proof of Work y la alta competencia que se genera con este. Los gastos energéticos producidos por una red de este estilo son enormes debido a la alta capacidad de computación necesaria para obtener beneficio (ya que únicamente el primer minero que encuentre el valor del hash que cumpla la condición recibe la recompensa). Otra de

[1] Cabe destacar que actualmente el precio de 32 eth ronda los 60.000$, número mínimo que será necesario para validar transacciones tras la transición a *Proof of Stake*.

sus desventajas es al mismo tiempo una de sus ventajas, que es la inmutabilidad, ya que, si un usuario es estafado, es prácticamente imposible que pueda recuperar sus fondos. Por último, otra desventaja remarcable es el uso seudónimo de las carteras, que en el caso de que se pierda el acceso a una de ellas, es muy probable que este se pierda de manera perpetua, y, por consiguiente, a los fondos que esta contenga.

2.3 Hipótesis del mercado eficiente

Esta hipótesis, propuesta por Eugene Fama [19] en 1998, expone la bien conocida frase en el mundo de los mercados financieros, 'el precio lo descuenta todo', es decir, cualquier noticia o evento futuro que afecte al precio no podrá ser utilizada para obtener un rendimiento económico de la misma, ya que dicho impacto en el precio será tan rápido que quedará inmediatamente descontado en el precio, no existiendo por ende ningún activo infravalorado o sobrevalorado. Existen varios factores que condicionan o motivan que un mercado tenga un mayor grado de eficiencia:

- Número de participantes: cuanto mayor sea el número de inversores que participen en un mercado, más eficiente será. Esto va directamente ligado con su volumen de transacciones, que tiene una proporcionalidad directa con la eficiencia del mercado. Un ejemplo de un mercado muy eficiente que cumple con ambas premisas es el mercado de divisas, donde millones de dólares son intercambiados a diario.
- Información disponible: cuanto mayor es la información que está en manos del inversor, más eficiente será el mercado. Necesariamente, no todos los inversores cuentan con la misma cantidad de información dentro de un mercado, lo que se conoce como *asimetría de la información*.
- Costes de transacción e información: a mayor sean los costes de operar en un mercado, menor será su eficiencia, es decir, si los costes operativos son tan grandes que exceden a los beneficios de negociar en este, el mercado será poco eficiente. Esto se debe a que los inversores serán más reticentes a operar en dicho mercado, reduciendo así el número de participantes, y junto a ello, su eficiencia.

Al mismo tiempo, se considera que existen distintos tipos de eficiencia que pueden estar presentes en un mercado:

- Eficiencia débil: los precios de los activos reflejan toda la información histórica, descartando por lo tanto la capacidad predictiva de los precios históricos y el volumen de los mismos. De tal manera, en un mercado débilmente eficiente sería imposible aplicar métodos autorregresivos, ya que no podríamos obtener información relevante de precios pasados en la serie temporal. Esto también descarta el uso del análisis técnico, que se basa en precios históricos y volumen. En conclusión, solo podría actuarse en dicho mercado mediante información pública y privada.
- Eficiencia semi-fuerte: en este caso, los precios reflejan tanto la información histórica como pública del activo en cuestión. Así, un inversor solo podría obtener rentabilidades del mercado mediante información privada o privilegiada sobre la masa de inversores. De esta manera, sería el análisis fundamental el que quedaría descartado, dado que basa su funcionamiento el aprovechamiento de la información pública del mercado.
- Eficiencia fuerte: el precio refleja toda la información existente sobre el mercado, tanto histórica como pública y privada. Cualquier inversor que obtuviese información privilegiada sería incapaz de sacarle partido, dado que el mercado reflejaría esta información rápidamente.

Teniendo en cuenta lo que esta hipótesis establece, se ha considerado la predicción del precio de distintos activos como una tarea imposible. Debido al gran número de variables potencialmente influyentes en el precio de un activo, así como el factor emocional y psicológico que guía a los inversores en su operativa, explica porque la predicción del comportamiento de un mercado es una tarea francamente complicada.

A pesar de ello, los inversores no han cesado en sus intentos por predecir valores futuros para el precio de distintos activos, o al menos, las tendencias presentes en los mismos. A lo largo de los años, distintos inversores como Warren Buffet han sido capaces de vencer a los mercados de manera consistente, poniendo en entredicho esta hipótesis. Es cierto que es posible encontrar oportunidades para conseguir ganancias, cuando el sentimiento humano o ciertos eventos (noticias, medidas de los gobiernos, etc.), mueven los precios más allá de su punto de equilibrio. Estas ineficiencias abren la puerta a un gran número de enfoques distintos para generar beneficios, identificando las tendencias de dichos mercados incluso si no es posible determinar el precio exacto.

Inicialmente, métodos estadísticos como ARIMA o regresiones lineales, han sido empleados con este propósito. ARIMA en concreto, ha sido capaz de modelar satisfactoriamente algunos de estos comportamientos, tal y como ha podido verse en 2.1. Pero con el auge del ML y el DL, los investigadores han estado focalizándose en nuevos enfoques y técnicas, como Redes Neuronales o Algoritmos Genéticos, para predecir este tipo de series temporales. En estos, se han aplicado a muchas variables más allá del simplemente el precio y el volumen de transacciones existentes en el mercado, como correlaciones con distintos activos, y en el caso concreto de las criptomonedas, datos propios de la cadena de bloques. A continuación, se procede a profundizar en algunos de los métodos más comúnmente utilizados con este objetivo.

2.4 Técnicas comúnmente utilizadas

A lo largo del tiempo el interés por modelar predictivamente distintos eventos e instrumentos financieros, ha sido una constante. Debido al gran impacto en la economía de la especulación con activos, desde inmobiliarios, a la bolsa de valores, tan pronto como en los años 70, se comenzaron a crear distintos modelos matemáticos con este propósito. Debido al objetivo específico de este proyecto se hará hincapié en los relativos a la bolsa de valores, dada su similitud con las criptomonedas. Dentro de este ámbito, cabe destacar algunos como ARIMA, desarrollado a finales de los años 60 y sistematizado en 1976 por Box y Jenkins, considerado como un método clásico del campo de la estadística financiera y la econometría. Más adelante, con el creciente interés en el ML, se han explorado nuevos enfoques al problema llegando a superar a planteamientos más clásicos. Entre ellos destaca el campo de las Redes Neuronales, y en concreto las Redes Neuronales Profundas dentro del campo del DL.

2.4.1 ARIMA

El modelo ARIMA consiste en una generalización del modelo ARMA (del inglés *Autorregressive Moving Average*), que combina los procesos Autorregresivos (AR) y de Media Móvil (MA) para generar un modelo sobre la serie temporal a analizar. Este se suele definir en función de tres parámetros como ARIMA(p,d,q) que definen las claves del modelo:

- **AR(p):** Autorregresión. Un modelo regresivo que utiliza la dependencia entre una observación y un número p de observaciones pasadas.

- **I(d):** Integrado. Parte del modelo que convierte la serie temporal en estacionaria mediante la medición de las diferencias de las observaciones en diferentes instantes de tiempo, donde d es el número de veces que se realiza esta diferenciación.

- **MA(q):** Media Móvil. Un enfoque que tiene en cuenta la dependencia entre observaciones y los términos de error residual cuando un modelo de media móvil es aplicado a las observaciones pasadas, donde q es el tamaño de la ventana de la media móvil.

Así, una manera de expresar un modelo AR de orden p, p.e., *AR(p)*, sería:

$$x_t = c + \sum_{i=1}^{p} \phi_i x_{t-i} + \epsilon_t \tag{2.1}$$

Donde x_t es la variable estacionaria, c es una constante y los términos ϕ_i son los valores de autocorrelación para los retardos 1,2,...,p, y ϵ_t son los residuos, ruido blanco gaussiano con media 0 y varianza σ_ϵ^2. Un modelo MA de orden q puede escribirse en la siguiente forma:

$$x_t = \mu + \sum_{i=0}^{q} \theta_i \epsilon_{t-i} \tag{2.2}$$

Donde μ es la esperanza de x_t (usualmente se asume igual a 0), y los términos θ_i son los pesos aplicados a cada término en la serie temporal, con $\theta_0 = 1$. Se asume que ϵ_t es una serie de ruido blanco gaussiano con media 0 y varianza σ_ϵ^2. De esta manera, podemos combinar estos modelos para formar un nuevo modelo ARIMA de orden (p, q):

$$x_t = c + \sum_{i=1}^{p} \phi_i x_{t-i} + \epsilon_t + \sum_{i=0}^{q} \theta_i \epsilon_{t-i} \tag{2.3}$$

Con $\phi_i \neq 0, \theta_i \neq 0$, y $\sigma_\epsilon^2 > 0$. Los parámetros p y q son conocidos como los órdenes de AR y MA respectivamente. Este modelo, a veces también llamado modelo Box-Jenkins, es capaz de tratar series temporales no estacionarias debido a su paso de 'integración'. Este componente consiste en la diferenciación de la serie temporal, para convertirla de una serie no estacionaria a una estacionaria. Se entiende el concepto de diferenciación como la obtención de una serie mediante la diferencia entre cada valor de la serie original y su valor predecesor en la misma serie. El parámetro d es obtenido calculando las diferencias necesarias para convertir la serie en estacionaria. Entendemos el concepto de serie temporal estacionaria como aquella tal que sus propiedades son independientes del instante de tiempo en el que realicemos la observación de la serie. En su forma general este modelo se denota como $ARIMA(p, d, q)$. Por lo tanto, si la serie es estacionaria inicialmente, se crearía un modelo $ARIMA(p, 0, q)$, también denominado ARMA.

También existen ampliaciones de este modelo que permiten incluir la estacionalidad presente en una serie temporal dentro del modelo, conocidas como SARIMA, con la adición de nuevos parámetros (P,D,Q) que modelan dicha estacionalidad.

2.4.2 Redes Neuronales

2.4.2.1 Perceptrón

En 1957, mientras trabajaba en los laboratorios Cornell en Bufallo, el psicólogo Frank Rosenblatt [20] propone un nuevo modelo probabilístico basado en la estructura del cerebro humano, que sería conocido como la primera red neuronal de la historia: el Perceptrón.

En concreto, el Perceptrón no es una red neuronal en sí, sino una unidad básica para la construcción de la misma, una neurona. Partiremos de un ejemplo simple con una sola entrada, para posteriormente extenderlo al caso de múltiples entradas, y, por último, uniremos distintas capas para llegar al caso de una red completa.

Una neurona de una sola entrada puede verse en la Figura 2.2.

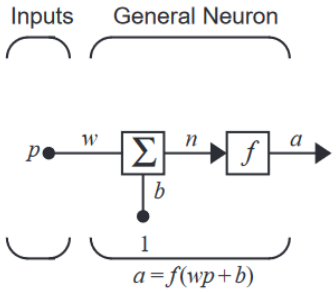


Figura 2.2: Neurona de una sola entrada. Obtenida de [1]

En este la entrada p es multiplicada por un peso w, y posteriormente sumada con un bias b. Este valor es posteriormente pasado por una función matemática, conocida como función de activación. Una vez realizado este proceso obtenemos la salida de la neurona a, que puede ser expresada como:

$$a = f(wp + b) \tag{2.4}$$

Normalmente, una neurona tiene más de una entrada. Una neurona con R entradas se muestra en la siguiente Figura 2.3. Las entradas individuales $p_1, p_2, p_3, ..., p_R$ son multiplicadas por sus respectivos pesos $w_1, w_2, w_3, ..., w_R$, los cuales son ordenados en una matriz de pesos $\mathbf{W}$.

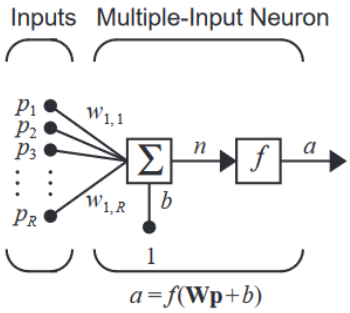


Figura 2.3: Neurona con R entradas. Obtenida de [1]

La neurona de múltiples entradas posee al mismo tiempo un bias que se incluye en el cálculo de la salida del sumatorio previo a su paso por la función de activación:

$$n = w_{1,1}p_1 + w_{1,2}p_2 + ... + w_{1,R}p_R + b \tag{2.5}$$

En el caso de una única neurona, la matriz de pesos **W** tendrá una única fila. La salida de la red puede a su vez en este caso ser expresada en forma matricial como:

$$a = f(\mathbf{W}\mathbf{p} + b) \tag{2.6}$$

Dada la simplicidad de este modelo, es raro el caso en el que sea suficiente para resolver el problema que se afronta. Para ampliar el alcance de estas, se unen varias neuronas que realizan la función anteriormente descrita de manera conjunta, lo que se conoce como *capa*. Una capa de S neuronas es mostrada en la Figura 2.4. Puede verse como cada una de las R entradas se conecta a cada una de las neuronas haciendo que ahora la matriz **W** tenga S filas.

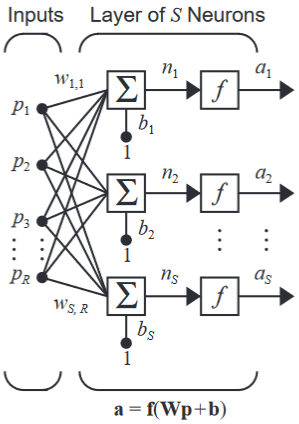


Figura 2.4: Capa de S neuronas. Obtenida de [1]

Esta es la estructura básica de un perceptrón de una capa. Pero esta presenta un problema, y es que esta es únicamente capaz de resolver problemas de clasificación linealmente separables. Para resolver este problema, varias capas como las vistas anteriormente son concatenadas, cada una con su propia matriz de pesos y bias.

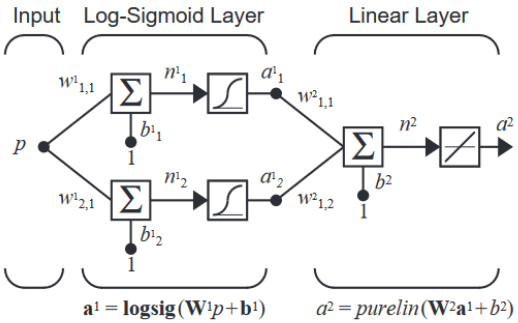


Figura 2.5: Perceptron de 2 capas. Obtenida de [1]

Este modelo puede ser utilizado para resolver una multitud de problemas, pero para ello, debe aprender características de los mismos, por lo que necesita pasar por un proceso de *entrenamiento*. Al pertenecer al campo del aprendizaje supervisado, este contará con una serie de entradas y una serie de objetivos o *targets*. Inicialmente los valores de los pesos y el bias son elegidos arbitrariamente. Este proceso de entrenamiento puede comprenderse como la búsqueda de los valores de **W** y **b** que resuelve el problema. Para ello, se utiliza la siguiente regla de aprendizaje:

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \mathbf{e}(k)\mathbf{p}^T(k) \tag{2.7a}$$

$$\mathbf{b}(k+1) = \mathbf{b}(k) + \mathbf{e}(k) \tag{2.7b}$$

Donde $e(k)$ denota el error en el paso k del entrenamiento, siendo este $e(k) = t(k) - a(k)$.

Como se puede ver en la Figura 2.5, cada capa cuenta con una función de transferencia o activación propia que aplica sobre todas las neuronas de esta. En la Figura 2.6 se muestran algunas de las principales funciones de activación utilizadas hoy en día.

Figura 2.6: Funciones de activación. Obtenido de
https://www.v7labs.com/blog/neural-networks-activation-functions

Tras comprender que es y como funciona un Perceptrón, pasamos a hablar sobre las Redes FeedForward. Estas redes también conocidas como MLP o Perceptrón multicapa, se trata de redes en la que la salida de cada una de las capas es usada como entrada de la siguiente. Estas en concreto tienden a ser totalmente conexas o *fully connected*, es decir, todas las entradas de una capa se conectan a todas las neuronas de esta. En este caso, el número de entradas en la capa de entrada y de salida, viene definido por las características del problema que se afronte. Hornik, Stinchcombe y White [21] demostraron en 1989 que estas podían ser consideradas como una clase de aproximadores universales. En la Figura 2.7 se muestra una red de este tipo en la que cada una de las neuronas individuales es un Perceptrón.

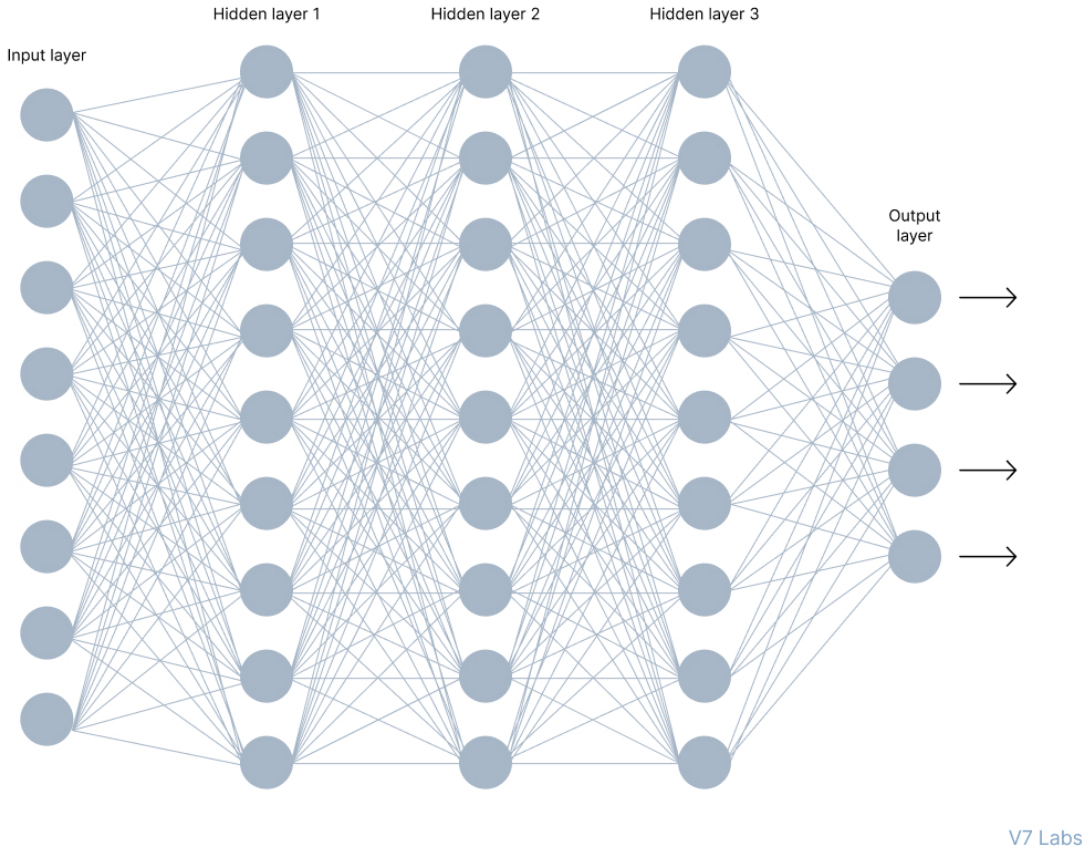


Figura 2.7: Red FeedForward. Obtenido de
https://www.v7labs.com/blog/neural-networks-activation-functions

2.4.2.2 Redes y Capas Habituales para Tratamiento de Series Temporales

Los avances en la capacidad de cálculo han abierto la puerta al desarrollo de técnicas mucho más complejas, dando lugar a arquitecturas mucho más potentes. Esto ha permitido el reciente desarrollo del ML y en concreto, el DL. Este último, es un subcampo del ML que basa sus algoritmos en la estructura del cerebro, aplicando capas sucesivas de ANNs. Este ha experimentado grandes avances recientemente, superando a los métodos más punteros en problemas como reconocimiento de texto, imágenes o voz [22]. A continuación, se procede a explorar algunos principales tipos de redes neuronales y capas que son utilizados hoy en día, con especial hincapié en los empleados en el tratamiento de series temporales y en este estudio.

1. **Redes Neuronales Recurrentes**: una Red Neuronal Recurrente (RNN), es un caso especial de red neuronal, que une a la entrada el valor de la salida en el paso anterior, lo que se conoce como retroalimentación, con el objetivo de predecir el siguiente paso en una secuencia de observaciones en función de los pasos previos observados en la secuencia. La idea subyacente a este tipo de redes es la capacidad de aprender de los datos en pasos anteriores para predecir tendencias futuras. Con este objetivo, los pasos previos deben ser recordados a la hora de predecir el siguiente paso. En este caso, las capas ocultas actúan como un almacén interno de información para almacenar la información contenida en los pasos previos. Son conocidas como *recurrentes* ya que realizan la misma acción para cada elemento de la secuencia, con la característica de utilizar la información

capturada previamente para predecir datos no vistos. En la Figura 2.8 se muestra la estructura básica de una RNN donde se puede apreciar a su vez el proceso denominado de desenrollado (unfold). En 2006, Schafer y Zimmerman [23] demostraron que las RNN eran también una clase de aproximadores universales basándose en el trabajo previo de Hornik, Stinchcombe y White. En el caso concreto de las redes neuronales recurrentes, se puede encontrar el problema de que solo son capaces de recordar unos pocos pasos anteriores y, por lo tanto, no son útiles a la hora de recordar secuencias más grandes de datos. Así, cabe destacar el caso de Long-Short Term Memory (LSTM), una arquitectura de RNN que pretende resolver este problema ampliando el horizonte sobre el que se es capaz de almacenar información.

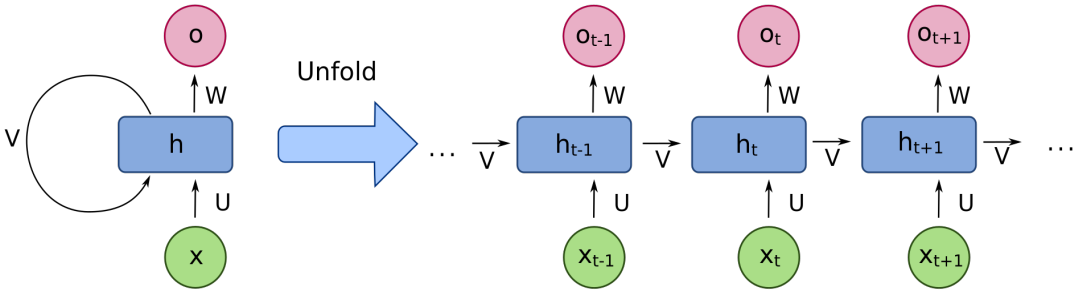


Figura 2.8: Red Neuronal Recurrente. Obtenida de
https://commons.wikimedia.org/wiki/File:Recurrent_neural_network_unfold.svg

(a) **Capa LSTM (*Long-Short Term Memory*)**: estas redes consisten en una variación de las RNN con el objetivo de recordar datos en horizontes temporales más lejanos que las RNN comunes. Estas fueron propuestas por Hochreiter *et al.* [24] en 1997. Con este objetivo utilizan unos mecanismos denominados puertas, que se encargan de aprender qué datos de una secuencia son importantes de aprender u olvidar. A continuación, describiremos las operaciones que lleva a cabo una unidad LSTM con este objetivo. La estructura concreta de una unidad LSTM se describe en la Figura 2.9.

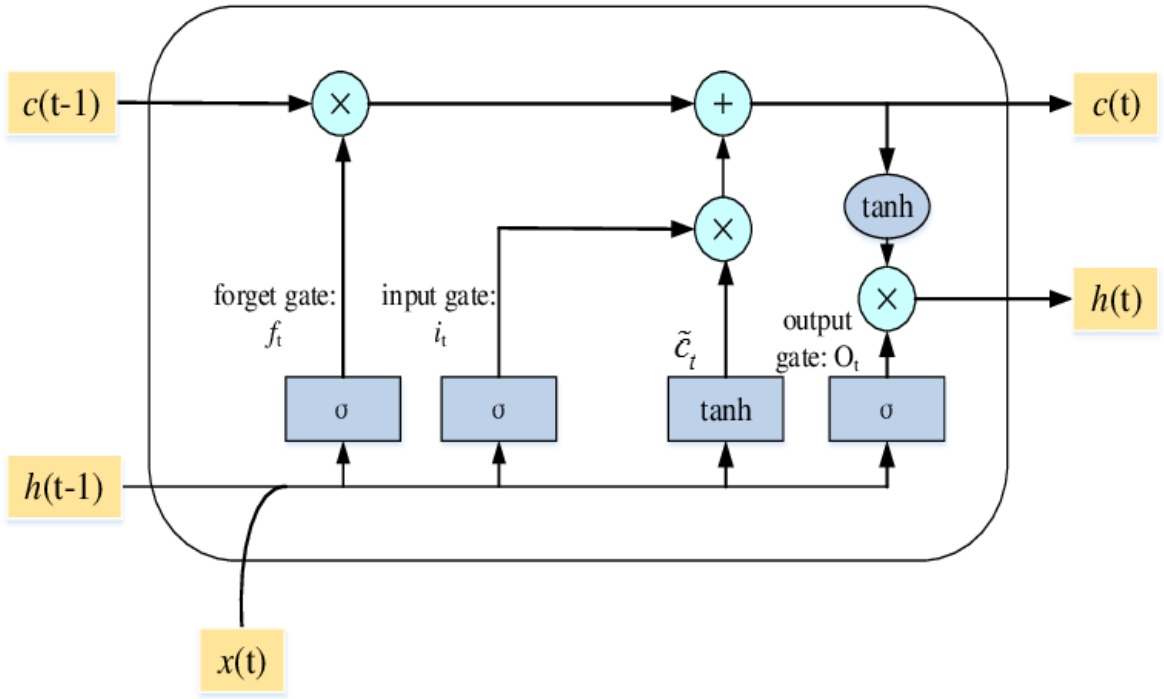


Figura 2.9: Unidad LSTM. f_t hace referencia a la puerta de olvido, i_t a la de entrada y O_t a la de salida. Obtenida de https://www.researchgate.net/figure/The-structure-of-LSTM-unit-51_fig4_349393514

i. ***Forget Gate*** **(Puerta de Olvido)** f_t: esta puerta se encarga de decidir qué datos son importantes o no para recordar. Para ello utiliza una función sigmoide, la cual devuelve un valor en el rango (0, 1), siendo un valor cercano a 0 la señal de que el dato es irrelevante, y uno cercano a 1 lo opuesto. Así, este toma el valor de salida del instante de tiempo anterior (h_{t-1}) y la entrada actual (x_t), y devuelve un valor en el rango para cada número dentro del estado de la celda (C_{t-1}).

ii. ***Input Gate*** **(Puerta de Entrada)** i_t: estas puerta utiliza las operaciones sigmoide y tangente hiperbólica para cumplir su cometido. Este es decidir qué información vamos a dejar que se guarde en el estado de la celda (*cell state*). Con la primera, realiza el mismo trabajo que en el caso anterior, decidiendo la relevancia de los datos y si actualizaremos estos o no; mientras que, con la segunda, devuelve un valor en el rango (-1, 1), siendo estos posibles candidatos a actualizar el valor de la celda, combinándose posteriormente.

iii. ***Output Gate*** **(Puerta de Salida)** o_t: esta puerta ofrece dos salidas distintas, en primer lugar la salida de la celda LSTM y en segundo lugar el estado de la celda tras el computo de esos datos. Para ello, primero calcula con la función sigmoide qué datos se desean devolver, y posteriormente estos son pasados a la función tanh para convertirlos al rango (-1, 1), multiplicándolos por la salida de la sigmoide.

De esta manera, la definición formal de este proceso podría escribirse como:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.8a}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.8b}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.8c}$$

i. σ: función sigmoide

ii. w_k: pesos para la puerta k

iii. h_{t-1}: salida de la unidad LSTM al instante de tiempo inmediatamente anterior

iv. x_t: entrada actual

v. b_k: bias para la puerta k

Las ecuaciones para el estado de la celda, estados candidatos y salida:

$$\tilde{c}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2.9a}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{2.9b}$$

$$h_t = o_t \odot tanh(c_t) \tag{2.9c}$$

i. $\odot$: producto de Hadamard (multiplicación de matrices elemento a elemento)

ii. c_t: estado de la celda en el instante de tiempo t

iii. $\tilde{c}_t$: candidatos para el estado de la celda en el instante de tiempo t

2. **Redes Convolucionales**: están enfocadas a la extracción de características, mediante la aplicación de una serie de filtros a la entrada de esta capa. Estas basan su comportamiento en la operación matemática de convolución de tensores, en la que un tensor denominado filtro o kernel se aplica al tensor que nosotros deseemos, devolviendo otro nueva que contiene la información de las características que se busca extraer. El valor de cada elemento del tensor de salida será el siguiente:

$$C[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k] \tag{2.10}$$

Donde:

(a) f: tensor de entrada.

(b) h: kernel o filtro.

(c) m: número de fila.

(d) n: número de columna.

(e) j: número de filas del kernel.

(f) k: número de columnas del kernel.

En la Figura 2.10 se muestra el proceso de convolución de matrices 2D, donde a cada paso se aplica la operación de convolución a un fragmento del tamaño del filtro (*kernel*), obteniendo así el *feature map* correspondiente (la salida de la operación de convolución) a la aplicación de este filtro, y en la Figura 2.11 se muestra el resultado de aplicar distintos filtros a una imagen, con el objetivo de reconocer los bordes de los objetos presentes en la figura. A su vez, en estas redes, cada una de las capas convolucionales que implementan cuentan con ciertos hiperparámetros que dictan su funcionamiento:

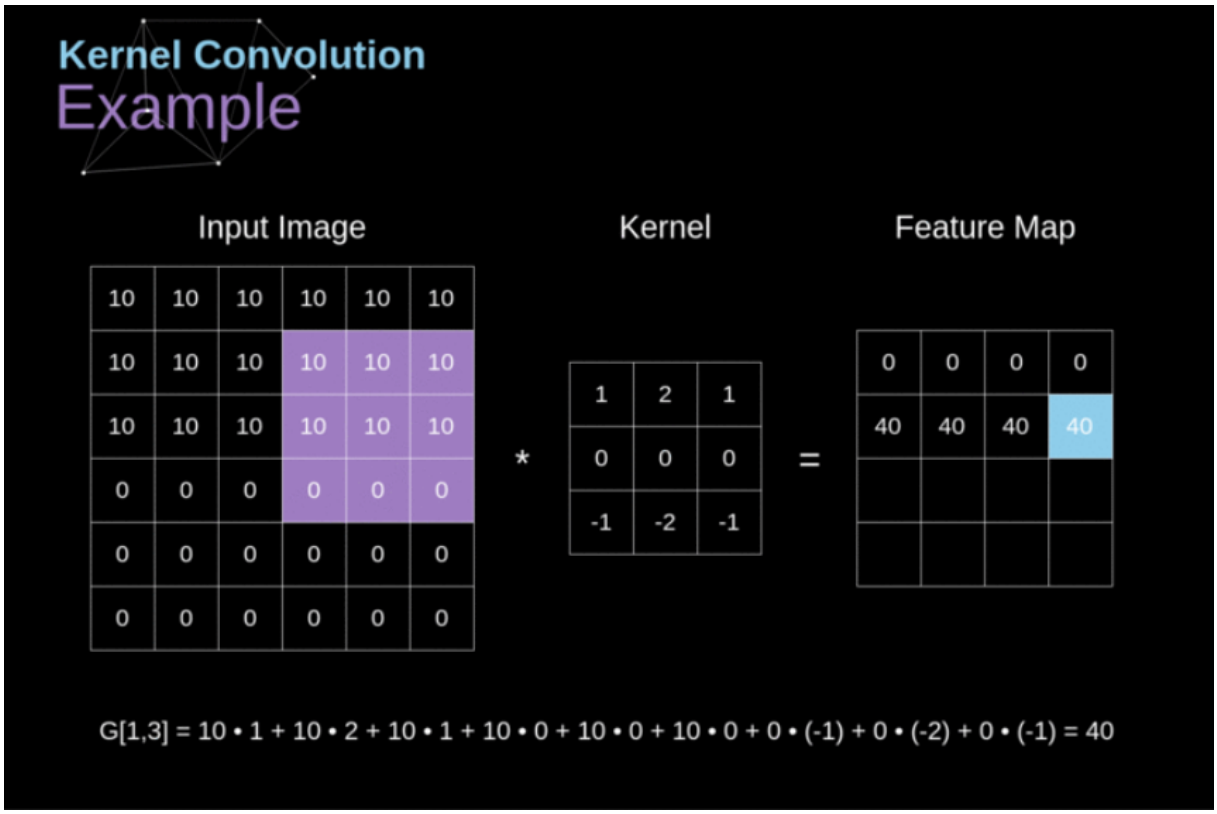
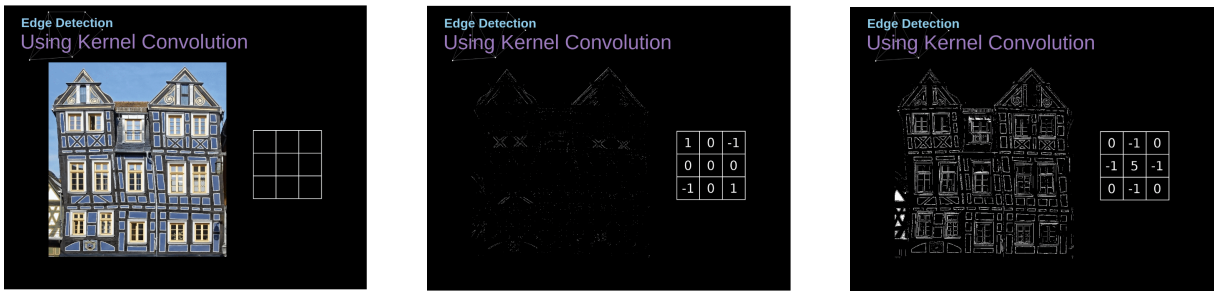


Figura 2.10: Proceso de convolución de matrices. Obtenida de [2]



(a) Imágen original (b) Aplicación del primer filtro (c) Aplicación del segundo filtro

Figura 2.11: Aplicación de filtros en una imagen para el reconocimiento de bordes. Obtenida de [2]

(a) **Tamaño de filtros**: hace referencia al número de filtros que se van a aplicar, es decir, el número de patrones o características que se busca aprender. Este valor dictamina la dimensión de la salida.

(b) **Tamaño del *Kernel***: este valor determina el tamaño de la ventana de convolución, es decir, el tamaño de los filtros que vayan a aplicarse a la matriz de entrada.

(c) **Padding**: este parámetro se usa para determinar sí el tensor de entrada será rellenado, de tal manera que la salida conserve el tamaño de dicho tensor. De no realizarse, la dimensionalidad de la salida se vería reducida. Para ello se añade un borde a la matriz de entrada, añadiéndolo alrededor del tensor de entrada. Este valor determina la manera en el que el rellenado será realizado, ya que existen distintas técnicas para ello, aunque usualmente se utiliza el valor adyacente o el valor 0.

En el caso concreto de este estudio, se hace uso de la capa Conv1D, se trata de una aproximación de una capa convolucional usual, con la variación de que, en nuestro caso concreto, no aplicaremos esto a un tensor de datos, sino a un vector unidimensional. Una representación visual de este proceso se muestra en la Figura 2.12.

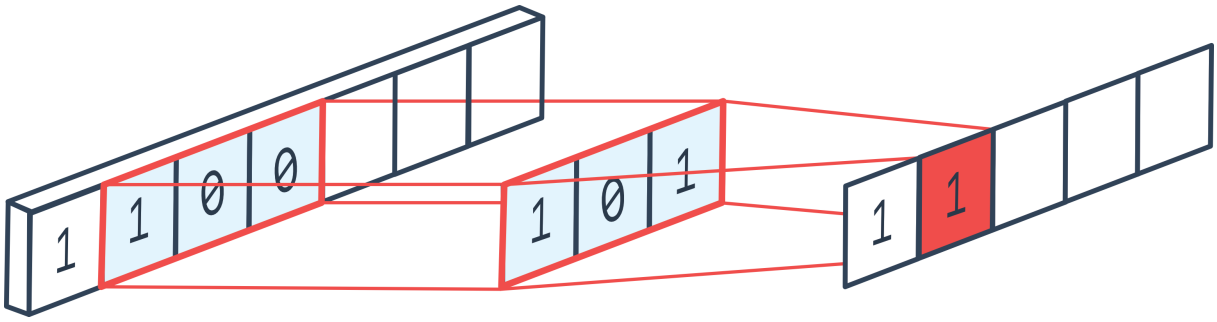


Figura 2.12: Convolución de vectores unidimensionales. Obtenido de https://peltarion.com/knowledge-center/modeling-view/build-an-ai-model/blocks/1d-convolution

Capítulo 3

Metodología

A fuerza de construir bien, se llega a buen arquitecto.

Aristóteles

En este capítulo se discuten las técnicas y herramientas empleadas, así como el proceso que se ha seguido para la consecución del estudio.

3.1 Elección de las criptomonedas a estudio

Para la elección de que criptomonedas emplear en este estudio, se tuvieron en cuenta tres criterios distintos. En primer lugar, la cantidad de datos disponibles para la criptomoneda, la volatilidad de la misma, y por último, la capitalización de mercado de la misma. En cuanto a la capitalización de mercado, se decidió no elegir ninguna criptomoneda por debajo del puesto 50 en aquel momento. Las razones de esta decisión se deben a la relación directa entre la volatilidad y la capitalización de mercado de un activo, ya que, a menor capitalización de mercado, menor será la cantidad de dinero necesaria (en forma de compra o de venta) para generar un cambio brusco en el precio del activo en cuestión.

En cuanto a la volatilidad y cantidad de datos entra en consideración el factor de la temporalidad sobre la que se realiza el estudio. A la hora de elegirla es importante destacar, que a mayor es la agregación temporal de los datos, menor es el ruido presente en ellos. Entendemos el término ruido como el concepto estadístico de una variabilidad inexplicable dentro de una muestra de datos dada. Así, teniendo en cuenta estos factores, y el hecho de que aproximadamente se ha de tener el mismo número de datos que de parámetros en una red con objeto de evitar el sobreaprendizaje (*overfitting*), se eligió una temporalidad de 1 día. Así, considerando que una red capaz de afrontar el problema tendría aproximadamente 1000 parámetros, se decidió establecer que el 80 % de las instancias totales habría de ser superior a 1000 (esta consideración se debe al uso de un 80 % de los datos para el conjunto de entrenamiento y el 20 % restante en el conjunto de validación). Para establecer una medida de volatilidad, se calculó para cada una de las criptomonedas y las temporalidades la desviación estándar y la media aritmética. Posteriormente se ordenaron cada una de ellas según número de instancias y volatilidad, y posteriormente se calculó una media de la posición en ambas ordenaciones. En la Tabla 3.1 se muestran las 18 criptomonedas que superaron el umbral de instancias junto con su puesto y los valores de media y desviación estándar.

Nombre	Número de instancias	Desv. Estandar	Media	Puesto Medio
BTCUSDT	1542	0.24340044	16128.3594	33
ADAUSDT	1299	0.22873614	0.41971898	52
BNBUSDT	1461	0.22498146	86.0463916	55
ETHUSDT	1542	0.21398929	771.065331	78
ICXUSDT	1242	0.21076829	0.68462383	86
IOTAUSDT	1255	0.19246327	0.5709749	136
VETUSDT	1200	0.18953613	0.03227307	145
XLMUSDT	1255	0.18510829	0.1782639	161
QTUMUSDT	1328	0.18468074	5.38563931	162
TRXUSDT	1244	0.17767248	0.03618347	181
LTCUSDT	1424	0.17260567	102.986397	193
XRPUSDT	1282	0.16807936	0.45323166	199
NEOUSDT	1447	0.15783215	30.3180422	225
ETCUSDT	1243	0.14896388	15.7833432	243
NULSUSDT	1202	0.14321809	0.56433544	251
EOSUSDT	1258	0.13522143	4.25447417	267
ONTUSDT	1247	0.11613568	1.10907442	290

Tabla 3.1: Estudio de variabilidad y número de instancias

De esta manera, de las 17 criptomonedas que superaron el umbral se eligieron las 9 que aparecen en la Tabla 3.2, donde se muestra su capitalización de mercado a día 21 de diciembre de 2021, aproximadamente cuando concluyó la descarga de los datos.

Nombre	Capitalización	Puesto
Bitcoin (BTC)	$925,255,868,119.27	1
Ethereum (ETH)	$477,751,386,114.62	2
Binance Coin (BNB)	$88,189,598,447.11	3
Ripple (XRP)	$44,773,731,359.42	6
Cardano (ADA)	$43,771,331,723.98	7
Litecoin (LTC)	$10,727,182,011.11	18
Tron (TRX)	$8,037,056,793.10	25
VeChain (VET)	$5,362,766,995.69	32
Eos (EOS)	$3,233,575,784.97	49

Tabla 3.2: Criptomonedas elegidas según su capitalización a 21 de Diciembre de 2021. Obtenido de https://coinmarketcap.com

3.2 Adquisición de los datos

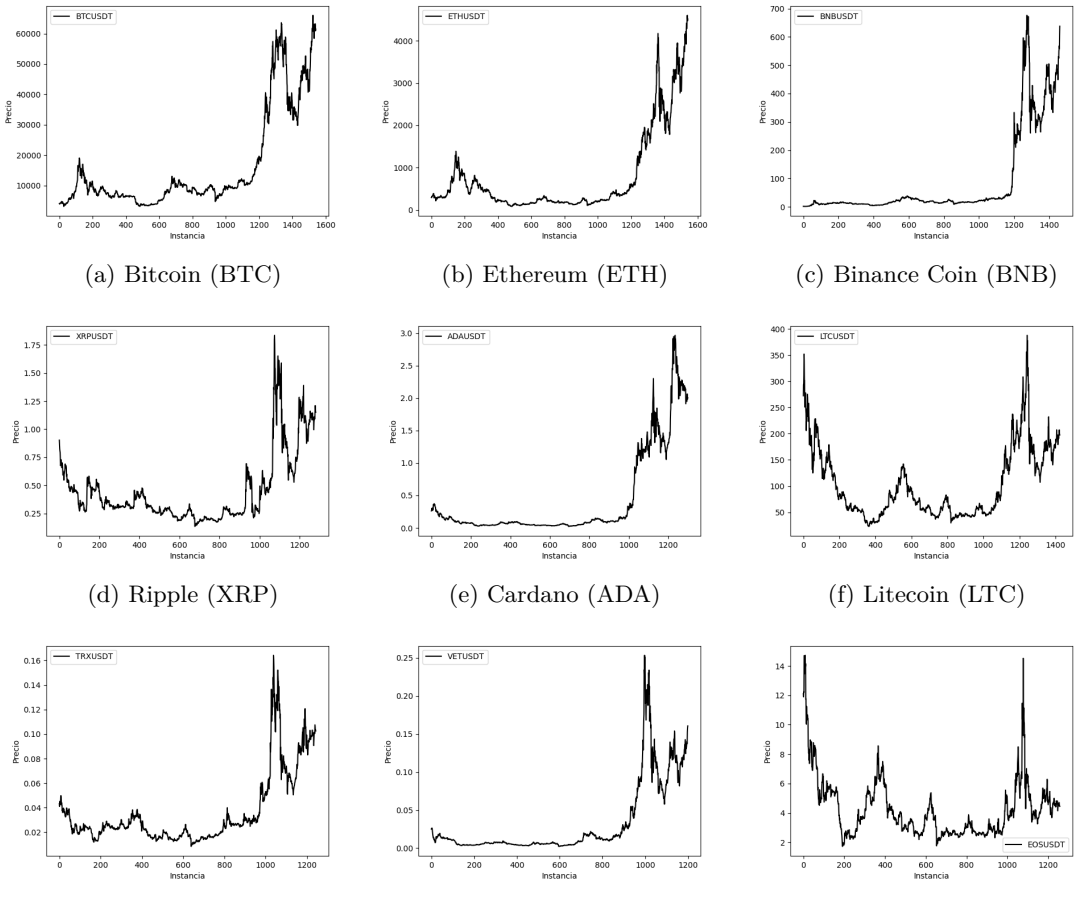
Los datos utilizados fueron obtenidos a través de la API Rest v3 de Binance que ofrece datos históricos sobre los activos listados en su plataforma. Para ello se hizo uso de la librería python-binance[1]. Se descargaron en total datos históricos de 1837 criptomonedas diferentes a temporalidades desde 1 semana hasta 1 minuto. Para poder establecer una comparabilidad entre estas criptomonedas, se decidió elegir el precio de estas criptomonedas respecto a USDT, otra criptomoneda con la peculiaridad de que se busca que su precio se igual a 1 dólar en todo momento. Esta (USDT) es actualmente la tercera criptomoneda

[1] https://python-binance.readthedocs.io/en/latest/

por capitalización de mercado. Se descargaron datos de Apertura, Máximo, Mínimo, Cierre y Volumen de transacciones para cada instancia. Los datos fueron obtenidos desde su listado en la plataforma Binance hasta el 6 de noviembre de 2021, día en el que concluyó su descarga, a vistas de maximizar el número de instancias disponibles para cada una de estas criptomonedas.

3.3 Análisis exploratorio

A continuación, se realizó un análisis exploratorio de los datos previo al comienzo del entrenamiento de las redes neuronales. En la Figura 3.1 podemos ver una visualización de todas las series temporales empleadas. Podemos ver a simple vista como la gran mayoría siguen una fuerte tendencia alcista.



(a) Bitcoin (BTC) (b) Ethereum (ETH) (c) Binance Coin (BNB)

(d) Ripple (XRP) (e) Cardano (ADA) (f) Litecoin (LTC)

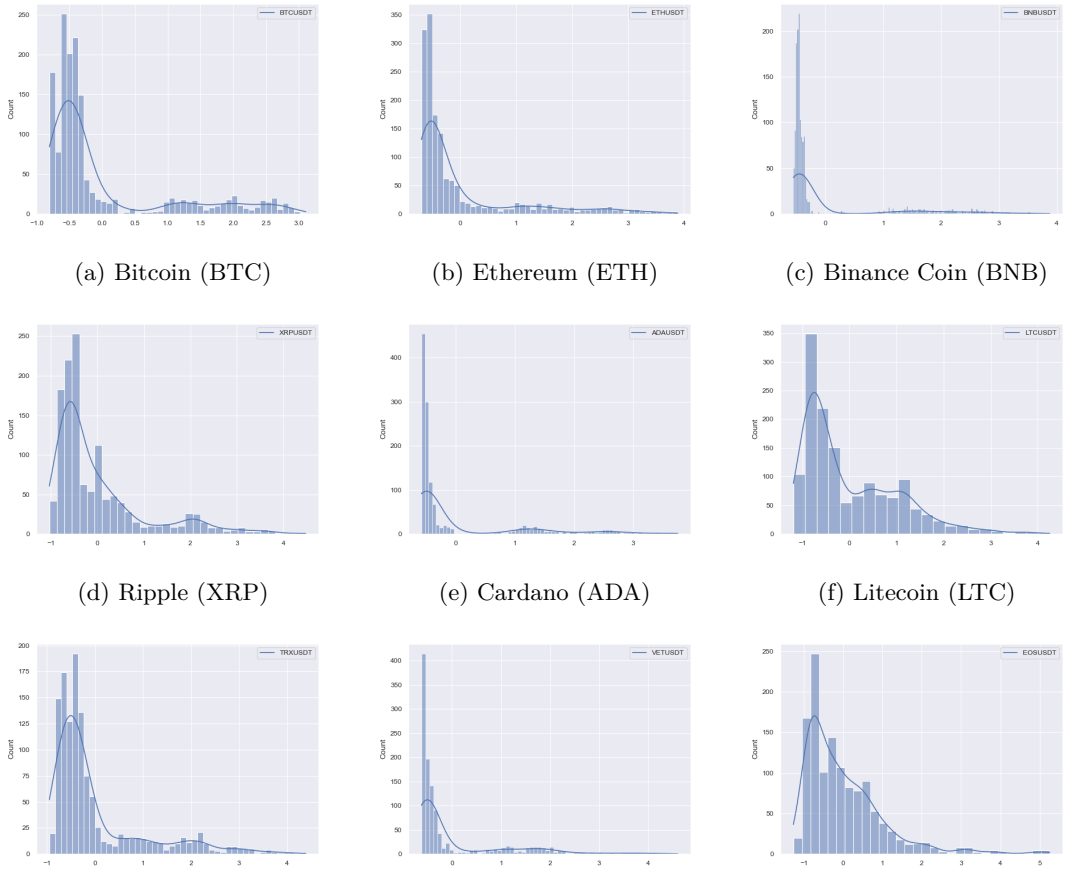
(g) Tron (TRX) (h) VeChain (VET) (i) Eos (EOS)

Figura 3.1: Visualización de cada una de las series temporales de las criptomonedas a estudiar

3.3.1 Análisis de la distribución de los datos

En primer lugar, se realizó un análisis sobre la distribución que los datos de la serie temporal de cada una de las criptomonedas seguían. Para ello se hizo uso de las bibliotecas de python `matplotlib` y `seaborn`. En concreto, se usaron las técnicas de un histograma de frecuencias y Kernel Density Estimation. En la Figura 3.2 se muestran los resultados del análisis para cada una de las criptomonedas. Los resultados son especialmente importantes a la hora de elegir el preprocesado que recibirán los datos antes de entrar a las redes neuronales. Como puede observarse, los datos siguen una distribución unimodal con una cola mucho

más pesada, presentando una asimetría muy marcada, donde la mayoría de los datos se concentra en la cola inferior. Esto será de especial importancia a la hora de la elección del proceso de normalización que se aplicará a los datos, de cara a conservar la distribución original de los mismos a la hora de alimentarlos al modelo, ya que el uso de, por ejemplo, una normalización basada en la media aritmética y la varianza resultaría en un conjunto de datos sesgados, debido a que estos tienden a sobreestimar valores altos (*upwardly biased*).



(a) Bitcoin (BTC) (b) Ethereum (ETH) (c) Binance Coin (BNB)

(d) Ripple (XRP) (e) Cardano (ADA) (f) Litecoin (LTC)

(g) Tron (TRX) (h) VeChain (VET) (i) Eos (EOS)

Figura 3.2: Histogramas de distribución y Kernel Density Estimation de las criptomonedas a estudiar

3.3.2 Análisis de autocorrelación

Posteriormente, se realizó un estudio de autocorrelación de los datos. Se entiende como autocorrelación la correlación existente entre la serie temporal y esta misma desplazada n veces hacia atrás. Este estudio aporta una cantidad de información no desdeñable, ya que nos indica si existe información relevante que aporte capacidad predictiva en los precios históricos de las criptomonedas en sí. En la Figura 3.3 se muestran los diagramas de autocorrelación de las 9 criptomonedas a estudio.

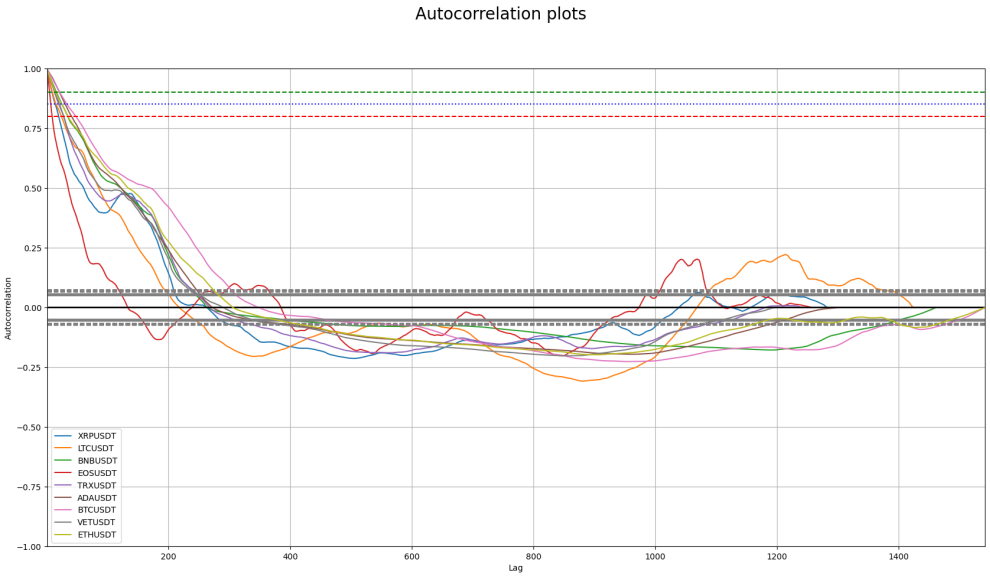


Figura 3.3: Diagrama de autocorrelación de las criptomonedas a estudio

En este mismo se presentan 3 líneas horizontales que marcan los valores de correlación 0.9, 0.85 y 0.8. Como se puede observar, BTC tiene el valor de correlación más alto para cualquier número de retardos o lags que cualquiera de las otras criptomonedas a estudio. De aquí podemos inferir que en los precios históricos de BTC encontramos una mayor cantidad de información relevante a la hora de predecir precios futuros que en cualquiera de las otras criptomonedas. A su vez, con EOS ocurre exactamente lo contrario, pudiendo suponer en este caso que en los precios históricos de la misma existe menos información relevante que en cualquier otra. Esto sugiere una mayor presencia de ruido dentro de las series temporales con menores valores de correlación.

También cabe destacar que existe un punto aproximado en el que casi todas las criptomonedas (a excepción de EOS Y LTC que lo sufren con cierta antelación) en el que el valor de la correlación cae por debajo de 0.25, cerca de los 200 retardos. Así, si bien es un valor bajo de correlación, todavía existe información valiosa que podemos utilizar hasta dicho punto, hecho que será vital a la hora de decidir ciertos parámetros de la arquitectura de la red. También se puede observar cómo existe una correlación negativa significativa (-0.25) entre los 800-1000 retardos. Teniendo en cuenta la temporalidad diaria elegida para el estudio, esto nos indica que la tendencia de los precios actual tiende a seguir una dirección contraria a la que estos seguían 3 años atrás. Así, podría concluirse que sería esperable que el proceso predictivo sin el uso de información exógena fuese más fructífero en aquellas criptomonedas cuyo valor de correlación es más alto, como es el caso de BTC, lo cual concuerda con el hecho de su mayor capitalización de mercado, y, en consecuencia, menor volatilidad, y peor en aquellas otras que es más bajo, como EOS, concordando a su vez con el hecho de que es la que menor capitalización de mercado tiene de las elegidas para este estudio.

3.4 Métricas para la evaluación de los modelos

A continuación, en la siguiente sección se presentan las métricas de error, similitud y correlación utilizadas para medir el desempeño de los modelos empleados en este estudio.

3.4.1 *Mean Squared Error* (MSE)

El error cuadrático medio o MSE representa la diferencia entre los valores actuales y los predichos de manera similar a la varianza. Esta mide la distancia entre los valores predichos y los valores reales. Puede ser formulada como:

$$MSE(y, \hat{y}) = \sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n} \tag{3.1}$$

3.4.2 *Mean Absolute Error* (MAE)

El Error Absoluto Medio o MAE representa la media del error de las observaciones respecto de los datos predichos. Es una medida muy intuitiva, ya que calcula directamente el error medio a lo largo del entrenamiento. Puede expresarse como:

$$MAE(y, \hat{y}) = \sum_{i=1}^{n} \frac{|\hat{y}_i - y_i|}{n} \tag{3.2}$$

3.4.3 *Mean Absolute Percentage Error* (MAPE)

El Error Absoluto Medio Porcentual o MAPE es una métrica muy parecida a MAE, pero en este caso en vez del error medio, calcularemos el porcentaje de error medio a lo largo del entrenamiento de nuestros modelos. Este se expresa de la siguiente forma:

$$MAPE(y, \hat{y}) = \sum_{i=1}^{n} \frac{\frac{|\hat{y}_i - y_i|}{y_i}}{n} \tag{3.3}$$

3.4.4 R^2 (R cuadrado)

Esta métrica relaciona la distancia de las predicciones al valor observado con la desviación de estos valores observados respecto a la media, obteniendo así un valor de correlación entre ambos conjuntos de datos. Un valor cercano a 1 indica una relación directa fuerte, mientras que uno cercano a -1 una relación inversa fuerte, mientras que valores cercanos a 0 indican correlación nula. Esto es así ya que el objetivo de R^2 es medir cómo de bien funciona nuestro modelo frente a una línea horizontal situada en el valor medio de la serie temporal. Este puede ser expresado en la siguiente forma:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{n}(y_i - \tilde{y})^2} \tag{3.4}$$

3.4.5 Coeficiente de correlación de Pearson

El Coeficiente de correlación de Pearson es una medida de la relación existente entre dos variables continuas, en nuestro caso, mide la correlación existente entre el conjunto de predicciones y de observaciones. Este puede expresarse de la siguiente manera:

$$r(y, \hat{y}) = \frac{\sigma_{\hat{y}y}}{\sigma_y \sigma_{\hat{y}}} \tag{3.5a}$$

$$\sigma_{xy} = \frac{\sum_{i=1}^{n}(x_i - \tilde{x})\sum_{i=1}^{n}(y_i - \tilde{y})}{n}$$

(3.5b)

$$\sigma(x) = \frac{\sqrt{\sum_{i=1}^{n}(x_i - \tilde{x})^2}}{n}$$

(3.5c)

Donde:

- $\hat{y}$: valor predicho.
- y: observación.
- $\tilde{x}$: valor medio del conjunto.
- n: total de valores predichos.
- σ_x: desviación típica de x.
- σ_{xy}: covarianza de x e y.

3.5 Normalización de los datos

La fase previa a alimentar el modelo con los datos es la normalización de estos. Esto es muy útil a la hora de entrenar los modelos debido a que si estos valores son especialmente grandes o pequeños, podemos caer en los problemas de explosión o desvanecimiento de gradiente, en los que debido a la aplicación de la regla de la cadena en la técnica de *backpropagation*, al multiplicar n valores a través de las n capas de la red, si estos son muy grandes o muy pequeños, el multiplicarlos uno tras otro provocará que estos crezcan muy rápidamente o muy lentamente, alterando el funcionamiento de los modelos. Otro motivo por el que la normalización de los datos cobra gran importancia es cuando tenemos varios atributos que dar a nuestro modelo que se encuentran en rangos muy diferentes, ya que uno que se encuentre en un rango mucho mayor que otro, afectará al desempeño del modelo en mayor medida que el resto, sin la necesidad de que este sea un predictor más importante.

Existen distintos métodos para la normalización de datos. Para ello, se ha hecho uso de la librería `scikitlearn` de python, que ofrece algunos como los siguientes:

- *StandardScaler*: este método consiste en una normalización gaussiana normal, es decir, la transformación de los datos a una distribución gaussiana normal, de desviación típica 1 y media 0. La expresión del proceso de normalización es el siguiente:

$$Media: \mu = \sum_{i=1}^{n} \frac{x_i}{n}$$

(3.6a)

$$Desviación\ típica : \sigma_x = \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{n}$$

(3.6b)

$$x_{norm} = \frac{x - \mu}{\sigma_x}$$

(3.6c)

- *MinMaxScaler*: este método permite escalar los datos de tal forma que sean todos se encuentren en un rango [a, b]. El proceso de normalizado sigue las siguientes ecuaciones:

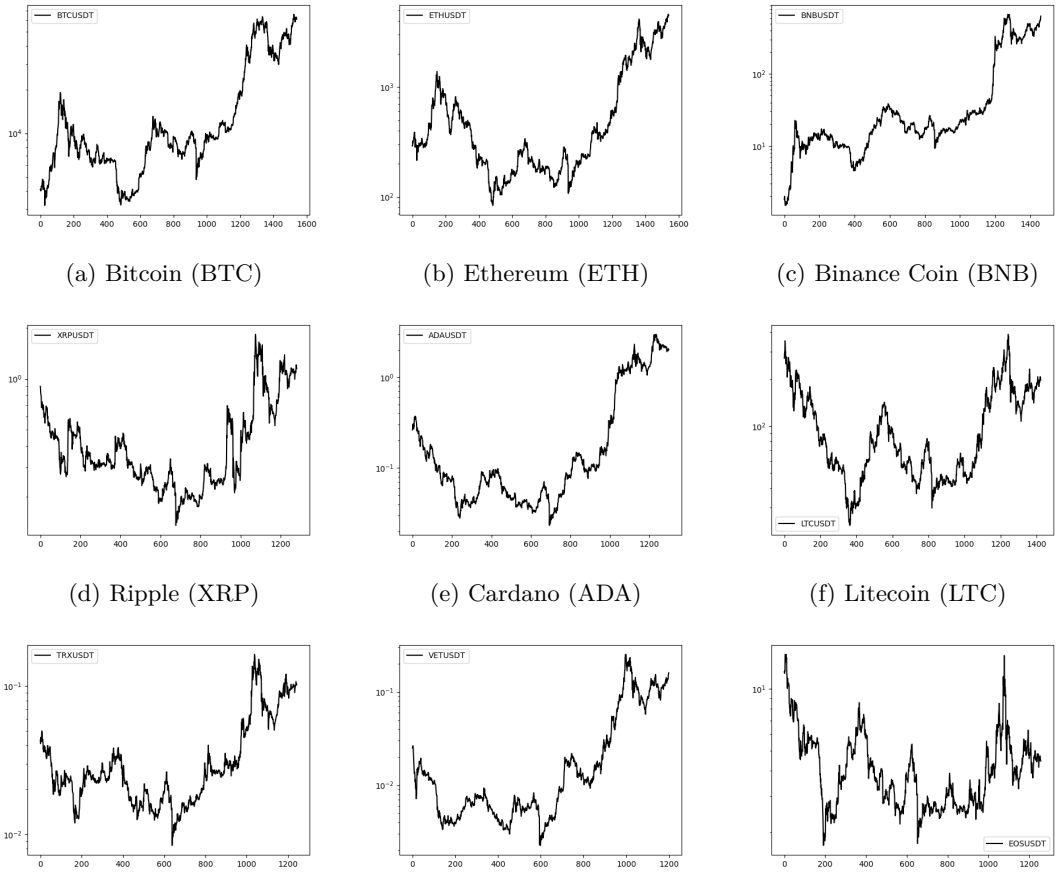
$$x_{std} = \frac{x - min(x)}{max(x) - min(x)}$$

(3.7a)

$$x_{norm} = x_{std} \cdot (a - b) + a$$

(3.7b)

De estos dos métodos de normalización, en nuestro caso aplicaremos el método MinMaxScaler, ya que al seguir los datos una distribución de colas pesadas tal y como se indicó en 3.3.1, una normalización de tipo StandardScaler cambiaría la distribución que siguen nuestros datos, no ajustándose así al problema que se pretende resolver. El método MinMaxScaler puede presentar problemas con la aparición de datos atípicos, ya que al hacer uso de los valores máximo y mínimo de nuestro conjunto de datos en el proceso de reescalado, puede causar que nuestros datos reescalados estén sesgados. En el caso concreto de las criptomonedas a estudio, no se han encontrado datos atípicos destacables. Para un análisis más en profundidad sobre esto se presenta una visualización de los datos en escala logarítmica en la Figura 3.4.



(a) Bitcoin (BTC) (b) Ethereum (ETH) (c) Binance Coin (BNB)

(d) Ripple (XRP) (e) Cardano (ADA) (f) Litecoin (LTC)

(g) Tron (TRX) (h) VeChain (VET) (i) Eos (EOS)

Figura 3.4: Visualización de cada una de las series temporales de las criptomonedas a estudiar

3.6 Diseño y optimización de la arquitectura de red

En esta sección se recoge el proceso de diseño y optimización llevado a cabo para la obtención de la arquitectura final utilizada en este estudio con el objetivo de realizar la comparativa más acertada posible entre los distintos conjuntos de datos de cada una de las criptomonedas utilizadas.

3.6.1 Perceptrón Multicapa (*Multilayer Perceptron*)

En esta subsección se recoge la estructura del modelo MLP empleado para realizar la comparativa sobre los diferentes modelos y conjuntos de datos. En primer lugar, se optó por una capa oculta de 64 neuronas y una de entrada de 32, con finalmente una salida de una sola neurona (en la última capa se deberá tener un número de neuronas igual al de características que se busque predecir, en nuestro caso el precio de cierre del día posterior). Este modelo pronto fue descartado debido al gran número de parámetros que tenía respecto al número de instancias disponibles, por lo que se redujo el tamaño de la red a efectos de paliar un posible problema de sobreaprendizaje.

Finalmente, se ha optado por un modelo simple, de una sola capa oculta, con 6 neuronas en la capa de entrada y 14 en la capa oculta (con una función de activación tangente hiperbólica), y finalmente una neurona con salida de activación lineal. En la Figura 3.5 se muestra la arquitectura de esta red.

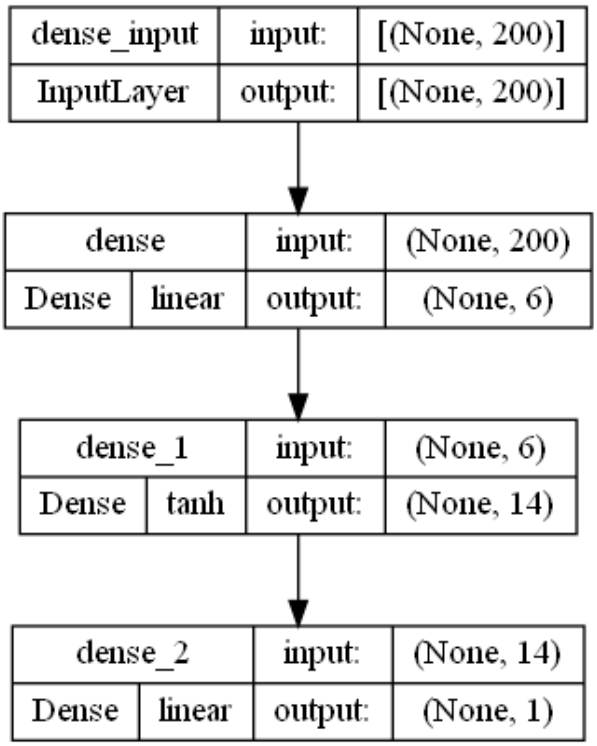


Figura 3.5: Modelo MLP

3.6.2 Arquitectura inicial

Dado que el objetivo principal de este proyecto no es la predicción, sino el establecimiento de una comparativa, se utiliza un modelo relativamente simple, el cual se muestra en el diagrama de la Figura 3.6.

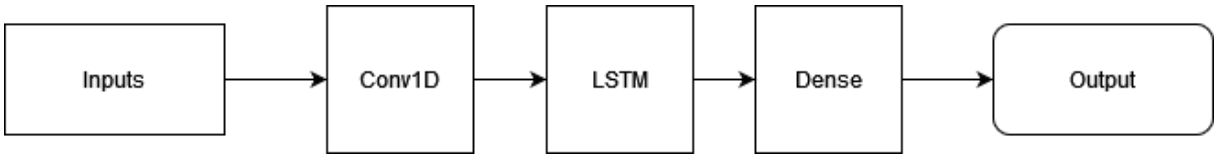


Figura 3.6: Arquitectura inicial

En primer lugar, trataremos las entradas o inputs. Para elegir como modelar las mismas nos habremos de referir a la Figura 3.3. Tal y como se mencionaba en 3.3.2, en el diagrama de autocorrelación que se acaba de mencionar, se puede apreciar que existe un valor de correlación consistente entre la serie temporal y ella misma hasta los 200 retardos, pudiendo argumentarse que existe información relevante para la predicción en el horizonte diario en los 200 días anteriores al mismo. En consecuencia, se ha optado por incluir los precios de cierre de estos últimos 200 días como entrada al modelo para la predicción del siguiente, tras haber sido normalizados mediante un MinMaxScaler tal y como se mencionó en 3.5.

Tras haber sido normalizados y agrupados en paquetes de 200 días, estos son dados al modelo. También cabe destacar el uso de MSE como función de pérdida para el ajuste de los parámetros, aunque se monitorizan todas las métricas señaladas en 3.4. A continuación, discutiremos los distintos parámetros elegidos para cada una de las capas empleadas:

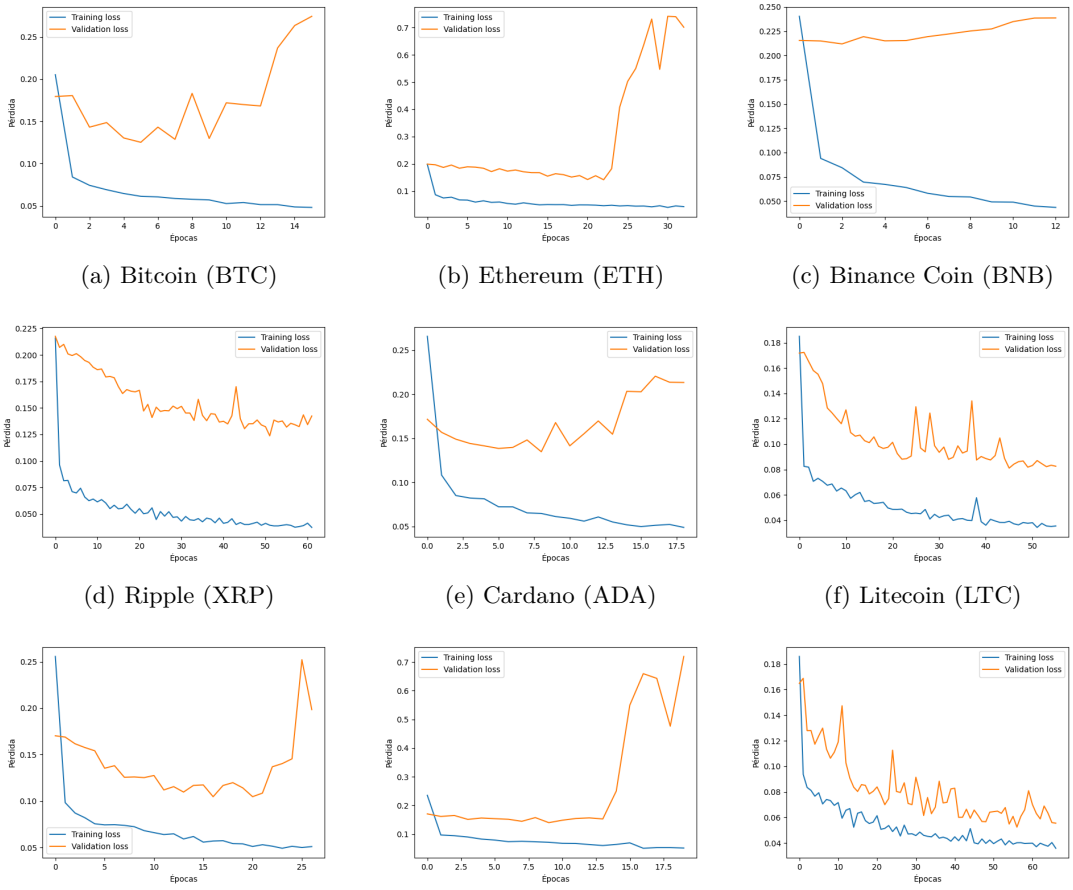
- Conv1D: En una primera instancia, los parámetros fueron elegidos de temporal para analizar su desempeño, siendo estos valores comumente utilizados en otros casos que empleaban este tipo de capas, eligiéndose número de filtros en 64, el número de kernel en 16 y el padding como 'valid',

es decir, no se completará el vector de entrada, reduciéndose la dimensionalidad de la salida de esta capa con respecto de la entrada. Como función de activación a la salida de la capa se eligió la función ReLu (Rectified Linear, definida como $f(x) = max(0, x)$).

- LSTM: en la capa LSTM habremos de elegir el número de unidades LSTM que se utilizarán. En nuestro caso, un valor de 64 unidades fue elegido inicialmente, de nuevo, por ser un valor que se ha visto utilizado en otros estudios que emplean este tipo de capas. A la salida de esta capa se decidió aplicar, al igual que en la capa anterior, la función ReLu.
- Dense: finalmente, en la última capa, se tiene una capa densa totalmente conexa, de una sola unidad y con una función de activación lineal pura, como es el estándar en los problemas de regresión.

3.6.3 Mejoras en el diseño de la arquitectura de red

A continuación se muestra en la Figura 3.7 las gráficas de entrenamiento y validación para cada uno de los modelos empleados:



(a) Bitcoin (BTC) (b) Ethereum (ETH) (c) Binance Coin (BNB)

(d) Ripple (XRP) (e) Cardano (ADA) (f) Litecoin (LTC)

(g) Tron (TRX) (h) VeChain (VET) (i) Eos (EOS)

Figura 3.7: Gráficas de entrenamiento y validación de los modelos sobre cada una de las criptomonedas a estudiar

Atendiendo a estas gráficas, podemos ver como se produce overfitting en la mayoría de los modelos a excepción de XRP, LTC y EOS, donde podríamos ver incluso underfitting, pudiendo apoyarse en los bajos valores de autocorrelación que tienen tanto LTC como EOS, sugiriendo que la predicción de ambas pueda ser de hecho, más complicada. En los casos en los que se puede apreciar overfitting, destacamos el

número de épocas de entrenamiento, siendo este bastante bajo. En el resto, podríamos argumentar una falta de épocas de entrenamiento, viendo un mejor ajuste de las gráficas en el caso de EOS. Dados estos resultados y al ser uno de los principales objetivos la construcción de una arquitectura común para todos los modelos se procedió a estudiar distintas formas para evitar el overfitting que sufren la mayoría de modelos.

Tal y como se mencionó en 3.1, el número de parámetros de la red ha de ser aproximadamente igual al número de datos que se poseen en el conjunto de entrenamiento de la red, a vistas de evitar dicho overfitting. Teniendo esto en cuenta es coherente pensar que este puede ser el motivo por el cual nuestros modelos no funcionen de manera correcta, ya que dados los hiperparámetros elegidos inicialmente, nuestra arquitectura cuenta con cerca de 35000 parámetros que optimizar, no teniendo más de 1500 instancias de datos para ninguna de las criptomonedas a estudio. Esto hace pensar que se dispone de pocos datos para la labor predictiva, y que un aumento de los mismos sería beneficioso. Si bien podríamos aplicar técnicas como regularizaciones L1 o L2, *Dropout* o añadir una capa de *Pooling* entre nuestras capas convolucional y LSTM, en primer lugar se optará por reducir de manera drástica el tamaño de nuestra red.

Así, en nuestra nueva arquitectura, los parámetros de las distintas capas son:

- Conv1D: se redujo el número de filtros a 16, con un tamaño de kernel de 2, manteniendo el resto de los valores igual. Con la corrección, esta capa aporta un total de 48 parámetros a la red.

- LSTM: esta es la capa que en mayor medida verá reducido su tamaño, ya que es a su vez la que más parámetros aporta en la arquitectura, siendo en el caso anterior alrededor de 33000. En este caso se ha optado por un valor de 8 unidades. Con el cambio, esta capa aporta un total de 800 parámetros.

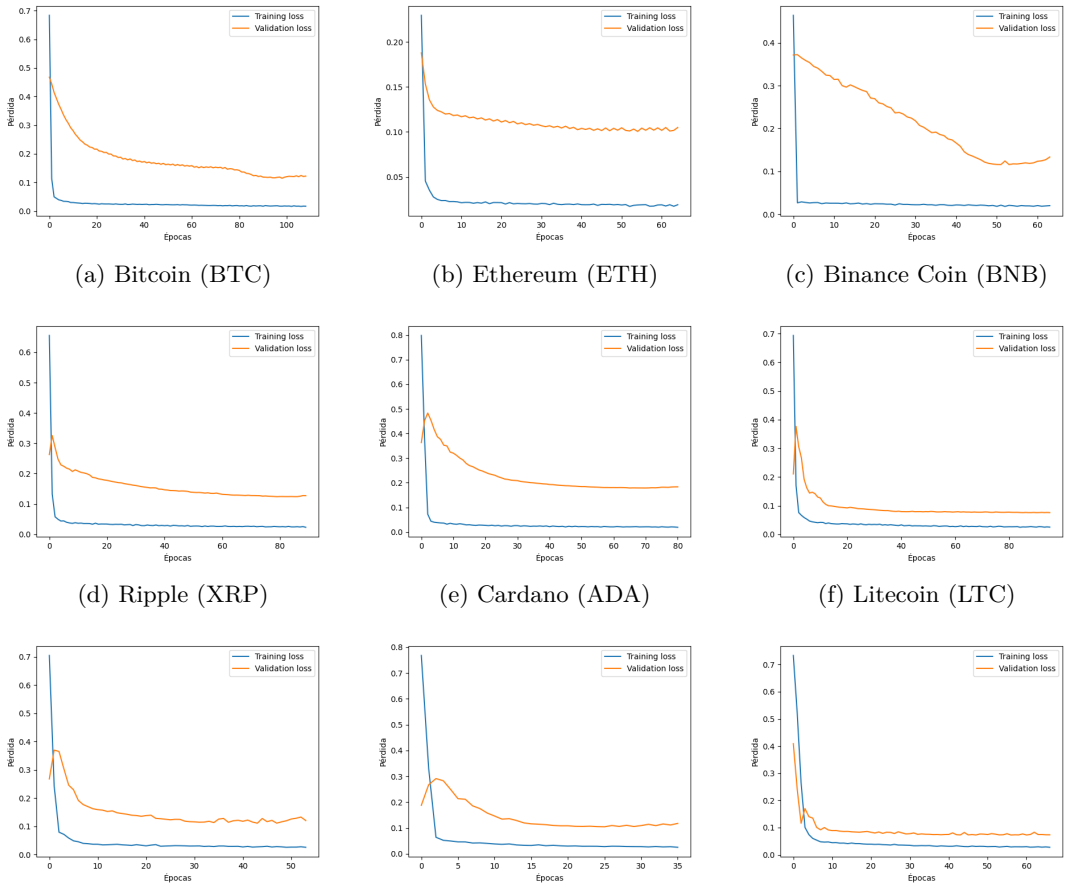
- Dense: la capa densa no sufrió cambio alguno, debido a que, al solo contar con una neurona, la cual es indispensable, el tamaño de esta capa no puede ser reducido, aportando un total de 9 parámetros.

En total tras los cambios introducidos, el tamaño total de la red se sitúa en 857 parámetros, rondando el número de instancias que se tienen para cada una de las criptomonedas (1200 - 1500 aprox.)

Haciendo uso de las gráficas de entrenamiento y validación y los valores de las métricas de error en validación se puede observar una mejora en el modelo. Las primeras pueden verse en la Figura 3.8 y las segundas en la Tabla 3.3. En estas tablas se muestra el valor mínimo encontrado en el proceso de validación. Si atendemos únicamente a las gráficas de entrenamiento podemos ver que los valores de la función de pérdida son semejantes a los del modelo anterior, pero se puede observar cómo ambas gráficas se acercan la una a la otra de una manera más consistente, destacando de nuevo el caso de EOS y LTC, donde el ajuste entre entrenamiento y validación es aún mayor. Atendiendo finalmente a la tabla de métricas, podemos ver como en todos los casos, de manera más notable en unos que en otros, los valores de error son menores en el modelo reducido en comparación al modelo original. Así, observamos tanto una mejor generalización del modelo frente a datos desconocidos como una mejora respecto al modelo original en cuanto al error cometido se refiere. A continuación se explora la adición de una capa intermedia de Pooling entre las capas convolucional y LSTM, ya que esto permitirá reducir el ruido inherente a las criptomonedas derivado de su volatilidad [3] y además reducir la dimensionalidad de los tensores pasados a la capa LSTM. El uso de este tipo de capas de Pooling es muy común en este tipo de problemas, en los que se hace uso de capas convolucionales.

	MSE (O)	MAE (O)	MAPE (O)	MSE (R)	MAE (R)	MAPE (R)
BTCUSDT	0.1347	0.3053	155.4671	0.0578	0.1940	235.1507
ADAUSDT	0.0590	0.1887	141.8232	0.1806	0.2975	105.4092
ETHUSDT	0.1711	0.3121	172.9149	0.1113	0.2333	81.4718
BNBUSDT	0.1683	0.3112	287.0167	0.0363	0.1477	220.1585
XRPUSDT	0.1086	0.2352	128.5477	0.1000	0.2198	114.7879
LTCUSDT	0.0796	0.1582	103.2566	0.0178	0.0836	63.8614
TRXUSDT	0.0272	0.1072	78.8677	0.0178	0.0873	108.3356
VETSUDT	0.0340	0.1269	135.0787	0.0176	0.0964	128.2055
EOSUSDT	0.0190	0.0841	24.6529	0.0154	0.0625	19.9584

Tabla 3.3: Valores de las métricas de error para los modelos Original (O) y Reducido (R) en validación



(a) Bitcoin (BTC) (b) Ethereum (ETH) (c) Binance Coin (BNB)

(d) Ripple (XRP) (e) Cardano (ADA) (f) Litecoin (LTC)

(g) Tron (TRX) (h) VeChain (VET) (i) Eos (EOS)

Figura 3.8: Gráficas de entrenamiento y validación de los modelos sobre cada una de las criptomonedas a estudiar en el modelo reducido

No obstante, antes de entrar en que consiste una capa de Pooling, cabe destacar que a la hora de realizar este modelo se realizó un cambio en la capa LSTM, siendo este el establecimiento de la función tangente hiperbólica como función de activación de la capa. La razón de esto es el uso de núcleos cuDNN, tal y como se indica en la documentación de keras (https://keras.io/api/layers/recurrent_layers/lstm/). Este simple cambio hizo que los modelos se entrenen alrededor de 10 veces más rápido, permitiendo realizar un mayor número de experimentos.

3.6.4 Adición de la capa de Pooling

Una capa de Pooling no es más que una transformación de los datos a partir del eje temporal de los mismos. Con este objetivo, existen dos tipos principales de Pooling, Max y Average. El primero, consiste en la elección del máximo dados n elementos a lo largo del eje temporal, mientras que el segundo se trata de la media aritmética de los n elementos a lo largo de dicho eje. Esto, permite una agregación de la información contenida en dichos elementos, permitiendo así una reducción del ruido inherente a la serie temporal a estudio, reduciendo la dimensionalidad de esta, dada la aplicación de la capa convolucional sobre esta, disminuyendo el tamaño del tensor con el que la capa LSTM tendrá que tratar en nuestro caso. Esto permite extraer las principales características que son invariantes posicionales y rotacionales, manteniendo el proceso de entrenamiento eficiente.

Con este objetivo, encontramos 3 parámetros que elegir para esta capa. Comenzamos con el parámetro padding, ya que su funcionamiento es idéntico al que realiza en la capa convolucional, ya que consiste en un rellenado del vector de entrada para mantener la dimensionalidad del vector a la salida. Esto, como se explicó anteriormente, es un proceso comúnmente realizado sobre matrices, pero dado el uso de vectores unidimensionales en el problema a tratar, nos encontramos ante una aproximación de este proceso para vectores de una única dimensión.

En segundo lugar, tenemos el parámetro pool size, o ventana de pooling, que consiste en cuantos elementos a lo largo del eje temporal utilizaremos para realizar la agregación de la información. Este parámetro variará en función del problema que tratemos, y deberá ser obtenido de manera experimental. Por último, tenemos el parámetro strides, que consiste en el paso que se realizará sobre el eje del tiempo para aplicar la ventana. Es decir, consiste en cuantos pasos hacia delante moveremos nuestra ventana para realizar la siguiente agregación. Este parámetro junto con la ventana de pooling es determinante para conocer la dimensionalidad de salida de la capa, siguiendo la siguiente expresión:

$$output\_shape = (input\_shape + 1)/strides \tag{3.8}$$

Tratando con el problema de reducción de parámetros de la red, esta capa no supone un problema, ya que no aporta ninguno, debido a que únicamente realiza transformaciones sobre los datos que recibe de la capa directamente anterior, no teniendo que ajustar ningún valor en el paso de retropropagación.

Tras ejecutar una serie de experimentos podemos observar como la adición de esta capa intermedia produce una mejoría sustancial. Experimentalmente se estableció que el valor óptimo del parámetro de la ventana se encontraba en el rango (25, 30). De esta manera, teniendo en cuenta los valores obtenidos, la obtención de este en distintas criptomonedas como valor óptimo, y ser tanto el valor medio como la mediana del rango óptimo, se eligió como valor final 27. También de forma experimental se comprobó que el funcionamiento de esta capa era mejor cuando se empleaba MaxPooling sobre AveragePooling.

Los resultados exponían una mejora en las métricas de error, que podía llegar hasta reducirlas en la mitad para MSE en caso como ETHUSDT o BTCUSDT. También se alcanzaron valores sorprendentes en métricas de correlación, destacando el caso de BNBUSDT, donde se obtuvo un valor del coeficiente de correlación de Pearson cercano a 0.9.

Como bien se ha reiterado en numerosas ocasiones a lo largo de este estudio, al no ser el objetivo de este realizar una predicción efectiva, se mantendrá la arquitectura de red de esta manera, ya que se considera suficiente como para poder establecer una comparativa consistente entre los distintos criptoactivos, pese a no realizar la predicción más exacta que podría conseguirse con ampliaciones a este modelo. En la Figura 3.9 se muestra la arquitectura final de la red.

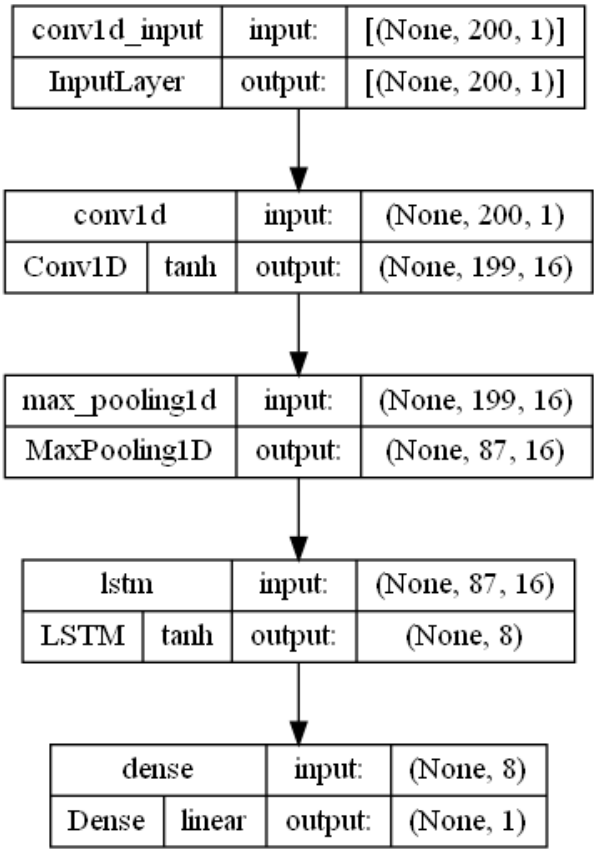


Figura 3.9: Modelo profundo final

Capítulo 4

Análisis Comparado

Rem tene, verba sequentur (Si dominas el tema, las palabras vendrán solas)

Catón el Viejo

En este capítulo se procederá a presentar los resultados experimentales obtenidos estableciendo una base comparativa con un modelo naive baseline.

4.1 Modelo baseline

Un modelo baseline es un modelo extremadamente simple, con el que se busca establecer unas métricas comparativas mínimas con las que analizar el rendimiento de otros modelos más complejos. Para la creación de este modelo se ha establecido un enfoque muy sencillo, el cual consiste en tomar como predicción en el horizonte diario el precio de la criptomoneda en el día anterior. Así, sea $\hat{Y}$ la variable a predecir y y_{t-1} el valor de la variable en el día inmediatamente anterior, este modelo puede expresarse como:

$$\hat{Y} = y_{t-1} \tag{4.1}$$

Tomando así los valores de la serie temporal desplazados un día, se ha realizado el cálculo de las distintas métricas de error para las distintas criptomonedas dado este enfoque. Los resultados de las mismas se muestran en la Tabla 4.1. Las métricas son medidas en USDT, pero dado que esta criptomoneda fue creada para tener explícitamente el valor de 1 dólar, se procederá a referirse a estas en dólares.

	RMSE (dólares)	MAE (dólares)	MAPE (dólares)
BTCUSDT	941.423	488.294942	2.87594333
ADAUSDT	0.04994316	0.0187648	4.20476859
ETHUSDT	69.9181859	31.5133074	3.7384611
BNBUSDT	12.1163505	4.01909528	4.19785073
XRPUSDT	0.04237865	0.020048	3.8103445
LTCUSDT	8.21816676	4.41601124	3.89079597
TRXUSDT	0.00330151	0.00159331	3.96907908
VETUSDT	0.00517687	0.00195464	4.90511514
EOSUSDT	0.37778058	0.18659682	4.00754197

Tabla 4.1: Métricas de error para el modelo baseline

4.2 Métricas de calidad

En esta sección discutiremos los valores de las métricas empleadas para medir el desempeño de los modelos empleados en este estudio a efectos de compararlas entre sí y a su vez, con el modelo baseline. Es necesario indicar que los valores empleados en la comparación con el baseline estarán desescalados, es decir, se invertirá la normalización que se realizó sobre los datos a efectos de emplear unidades comprensibles, en este caso dólares.

Para establecer una comparativa formal del rendimiento de los modelos, se contactó con un experto que realizó una predicción en el horizonte diario para las dos criptomonedas que se sitúan en extremos opuestos del ranking por capitalización (de entre las utilizadas en este estudio), Bitcoin (BTC) y Eos (EOS). Para BTC, obtuvo un error cuadrático de 40401 dólares^2 y un error absoluto de 201 dólares, y porcentual de 0.68 %. Para EOS obtuvo un error cuadrático de 0.00002304 dólares^2 y un error absoluto de 0.0048dólares, y porcentual de 0.37 %.

Ahora, se procederá a mostrar las métricas de desempeño de los distintos modelos. En primer lugar, mostraremos los resultados para el modelo MLP, y posteriormente el modelo profundo. En este caso, exponemos las mejores métricas obtenidas tras realizar 25 entrenamientos sobre cada una de las criptomonedas con cada uno de los modelos, a efectos de paliar un posible mal resultado derivado de la naturaleza estocástica del entrenamiento. Para el modelo profundo se exponen dos tablas, en primer lugar, una después de realizar los entrenamientos con MSE como función de pérdida, y en segundo lugar otra usando MAE.

Cabe destacar algunos de los parámetros usados para los entrenamientos de los modelos. Entre ellos destacamos el uso de RMSProp como optimizador, el tamaño del batch de datos en 64, un split de validación del 20 %, o el número de épocas, que, si bien es establecido en 2000, se ha hecho uso de un callback de la librería keras llamado EarlyStopping, el cual monitoriza la función de pérdida y para el entrenamiento automáticamente cuando esta deja de mejorar un número determinado de veces.

En las Tablas 4.2 y 4.3 se muestran los resultados del modelo MLP para entrenamiento y validación respectivamente. En las Tablas 4.4 y 4.5 se muestran los resultados para el modelo profundo con MSE, para de nuevo, entrenamiento y validación. En 4.6 y 4.7 podemos ver los datos para el entrenamiento y validación de nuestro modelo profundo usando esta vez MAE como función de pérdida. No se indican unidades para los valores de MAE y MSE debido a que estos valores se hayan normalizados.

	MAE	MSE	MAPE	Pearson	R2
BTCUSDT	0.0400	0.00259	9.09	0.980	0.930
XRPUSDT	0.0382	0.00279	5.22	0.922	0.792
EOSUSDT	0.0422	0.00354	9.26	0.960	0.887
ADAUSDT	0.0361	0.00231	5.29	0.983	0.909
ETHUSDT	0.0335	0.00209	4.31	0.965	0.829
BNBUSDT	0.0230	0.00213	6.78	0.965	0.434
LTCUSDT	0.0367	0.00239	8.07	0.971	0.928
TRXUSDT	0.0551	0.00645	9.79	0.951	0.765
VETUSDT	0.0338	0.00299	6.87	0.988	0.871

Tabla 4.2: Métricas del modelo MLP para entrenamiento

	MAE	MSE	MAPE	Pearson	R^2
BTCUSDT	0.1523	0.03462	186.08	0.584	-0.553
XRPUSDT	0.1828	0.06534	113.38	0.514	-1.465
EOSUSDT	0.0994	0.02206	28.17	0.330	-1.555
ADAUSDT	0.1392	0.03659	82.77	0.666	-0.368
ETHUSDT	0.1438	0.03215	111.23	0.625	-0.330
BNBUSDT	0.1889	0.05435	226.98	0.809	0.3498
LTCUSDT	0.1456	0.03947	94.37	0.723	0.0495
TRXUSDT	0.2245	0.07744	208.78	0.386	-1.734
VETUSDT	0.1781	0.05239	260.01	0.496	-2.805

Tabla 4.3: Métricas del modelo MLP para validación

	MAE	MSE	MAPE	Pearson	R^2
BTCUSDT	0.0243	0.00121	7.21	0.989	0.967
XRPUSDT	0.0227	0.00120	3.21	0.962	0.912
EOSUSDT	0.0331	0.00221	8.63	0.971	0.930
ADAUSDT	0.0131	0.00060	2.62	0.994	0.979
ETHUSDT	0.0112	0.00027	1.41	0.989	0.972
BNBUSDT	0.0091	0.00053	4.17	0.987	0.902
LTCUSDT	0.0295	0.00179	6.96	0.980	0.950
TRXUSDT	0.0290	0.00186	4.90	0.978	0.927
VETUSDT	0.0160	0.00090	4.34	0.993	0.976

Tabla 4.4: Métricas del modelo profundo para entrenamiento (MSE)

	MAE	MSE	MAPE	Pearson	R^2
BTCUSDT	0.1562	0.03747	125.22	0.673	-10.101
XRPUSDT	0.1244	0.03096	79.30	0.493	0.018
EOSUSDT	0.0899	0.02185	30.05	0.510	0.201
ADAUSDT	0.1258	0.03263	87.98	0.775	0.119
ETHUSDT	0.1298	0.02865	86.69	0.623	-0.615
BNBUSDT	0.1340	0.03594	143.18	0.866	0.598
LTCUSDT	0.0967	0.02317	66.03	0.831	0.581
TRXUSDT	0.1226	0.03253	93.71	0.487	0.127
VETUSDT	0.1317	0.02959	108.85	0.751	0.210

Tabla 4.5: Métricas del modelo profundo para validación (MSE)

	MAE	MSE	MAPE	Pearson	R^2
BTCUSDT	0.0187	0.00063	5.03	0.994	0.982
XRPUSDT	0.0227	0.00125	3.17	0.962	0.912
EOSUSDT	0.0369	0.00265	8.44	0.964	0.915
ADAUSDT	0.0141	0.00047	2.58	0.994	0.981
ETHUSDT	0.0147	0.00035	1.80	0.991	0.969
BNBUSDT	0.0144	0.00064	4.57	0.988	0.862
LTCUSDT	0.0285	0.00168	7.12	0.981	0.953
TRXUSDT	0.0274	0.00199	4.87	0.980	0.932
VETUSDT	0.0253	0.00156	4.73	0.993	0.962

Tabla 4.6: Métricas del modelo profundo para entrenamiento (MAE)

	MAE	MSE	MAPE	Pearson	R^2
BTCUSDT	0.1362	0.02702	99.34	0.705	-4.016
XRPUSDT	0.1252	0.02970	90.95	0.550	0.052
EOSUSDT	0.0919	0.02247	30.88	0.520	0.224
ADAUSDT	0.1095	0.02194	90.08	0.777	0.249
ETHUSDT	0.1202	0.02339	93.16	0.630	0.031
BNBUSDT	0.1390	0.03570	173.37	0.866	0.586
LTCUSDT	0.0914	0.01897	64.65	0.848	0.644
TRXUSDT	0.1214	0.03049	97.20	0.490	0.141
VETUSDT	0.1295	0.02901	114.28	0.738	0.316

Tabla 4.7: Métricas del modelo profundo para validación (MAE)

En primer lugar, compararemos el desempeño de ambos modelos tomando como referencia el modelo profundo entrenado con MSE. Puede observarse, tal y como se pudo ver en las gráficas de entrenamiento y validación de ambos modelos una gran diferencia entre ambos valores, obteniéndose al menos resultados un orden de magnitud menores en entrenamiento que en validación, fenómeno que se extiende al resto de métricas de error. Centrándonos en R^2, podemos ver de nuevo una divergencia entre entrenamiento y validación, obteniendo valores cercanos a 1 en el primero y valores negativos en el segundo de manera general. Así, puede pensarse que existe un exceso de complejidad en la red para el problema que se afronta,

tal y como se pudo ver a la hora de optimizar nuestro modelo, observándose una mejora en estos valores a la hora de reducir el número de parámetros en nuestra arquitectura. No obstante, esto no imposibilita la tarea de comparación entre las distintas criptomonedas.

A continuación, pasamos a enfocarnos únicamente en las métricas de correlación. Podemos observar, como en incluso ambos modelos, incluido MLP pese a su simplicidad, hemos obtenido valores de correlación superiores a 0.8, los cuales han sido obtenidos de manera consistente a lo largo de los entrenamientos realizados en este estudio, destacando el caso de BNB y LTC, que lo consiguen en el modelo profundo (además BNB ha obtenido un valor de correlación de 0.8 en el modelo MLP). Esto indica que, si bien el modelo no generaliza de manera totalmente idónea para el problema, sí que es capaz de captar la forma de la función a pesar de no devolver las predicciones exactas. Esto implica, que el modelo falla a la hora de predecir la amplitud de la función, pero capta de manera correcta la forma de la misma para dichas criptomonedas. Esto es especialmente importante, dado que sugiere la existencia de patrones en las tendencias ascendentes y descendentes del valor, incluso si los valores específicos son impredecibles.

Para analizar más en profundidad estos resultados, se ha calculado el valor de correlación entre la serie temporal y la salida del modelo una vez entrenado, no a lo largo del proceso de entrenamiento. Para ello se ha empleado la implementación de la librería `scipy`, la cual nos devuelve el p-valor junto con el valor de correlación. Los resultados se pueden ver en la Tabla 4.8. El p-valor nos indica la probabilidad de que la hipótesis nula de que ambos conjuntos de datos tengan distribuciones distintas, es decir, la probabilidad de haber obtenido estos valores a pesar de que ambos conjuntos tengan distribuciones distintas. Como se puede observar, se ha obtenido un valor 0, lo que indica que los valores de correlación obtenidos son consistentes. Se han obtenido valores superiores a 0.9 en todos los modelos, un valor significativamente alto.

	BTCUSDT	EOSUSDT	ADAUSDT	ETHUSDT	BNBUSDT
r	0.989	0.962	0.914	0.979	0.957
p-value	0	0	0	0	0

	XRPUSDT	LTCUSDT	TRXUSDT	VETUSDT
r	0.972	0.965	0.978	0.979
p-value	0	0	0	0

Tabla 4.8: Coeficiente de correlación de Pearson y su respectivo p-valor para cada una de las criptomonedas

En cuanto a la comparación del desempeño del modelo profundo usando MAE y MSE, se puede ver que los resultados son muy similares, pudiendo atribuirse las ligeras diferencias a casos excepcionales en alguno de los entrenamientos. De esta manera, al seguir nuestro conjunto de datos una distribución distinta de la gaussiana normal, preferiremos a MAE como función de pérdida, ya que, en distribuciones con colas pesadas, MSE tiende a ponderar de manera más fuerte errores mayores, pudiendo generar ciertos problemas de convergencia, o incluso consiguiendo dicha convergencia, pero dando como resultado un modelo sesgado.

En cuanto a la comparación entre los distintos modelos de la arquitectura, es necesario pararse a analizar los valores obtenidos para MAPE, ya que, atendiendo a los resultados de ambos modelos, pero especialmente el profundo, puede observarse como el desempeño acorde a esta métrica de los modelos

de EOS y LTC es superior al resto (son además los dos únicos modelos que rompen la barrera de 0.1 para sus valores de MAE). En el caso del profundo, tanto con el uso de MSE como con el de MAE como función de pérdida. Mientras que, en el caso general, los valores de esta métrica rondan valores entre el 90-100 %, en el caso de LTC encontramos valores en el rango 60-70 % y en el de EOS incluso mejores, en el rango 25-30 %.

También es necesario puntualizar el hecho de que de manera reiterada y en un porcentaje no desdeñable de los entrenamientos (en algunas de estas criptomonedas incluso cercanos al 50 %), se han producido malos entrenamientos en los que un gran *overfitting* podía observarse en épocas muy tempranas del entrenamiento. En la Figura 4.1 puede verse un ejemplo de esta clase de entrenamientos, en este caso concreto, perteneciente al modelo VETUSDT. En la Tabla 4.9 se expone un conteo del porcentaje de malos entrenamientos calculados a lo largo de 50 ejemplos.

	Malos Ajustes (%)
BTCUSDT	20 %
XRPUSDT	38 %
EOSUSDT	24 %
ADAUSDT	32 %
ETHUSDT	14 %
BNBUSDT	42 %
LTCUSDT	28 %
TRXUSDT	30 %
VETUSDT	46 %

Tabla 4.9: Porcentaje de malos entrenamientos para cada uno de los modelos sobre 50 pruebas

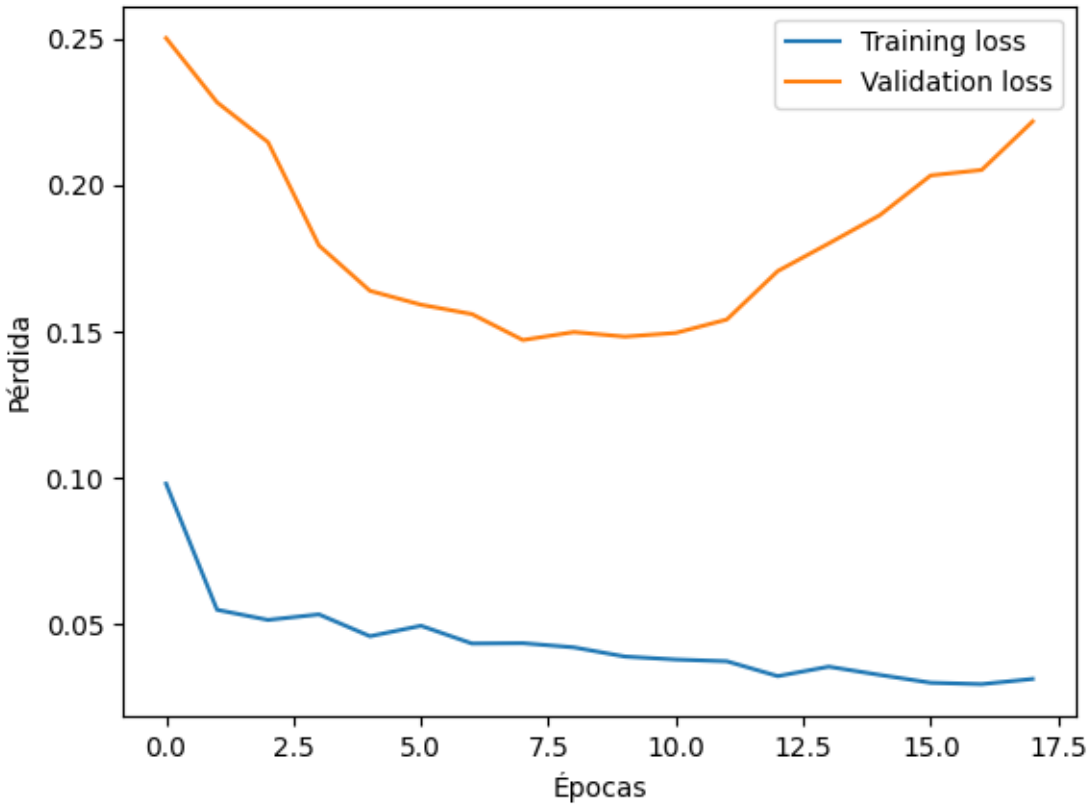


Figura 4.1: Ejemplo de entrenamiento de VETUSDT

Por último, pasaremos a realizar la comparativa del modelo baseline con ambos modelos sin escalar. En las Tablas 4.11 y 4.10 podemos encontrar los datos para el modelo MLP y profundo respectivamente tras el proceso de desescalado. En el caso de MSE, para mostrar esta comparativa se ha optado por sustituirlo por el valor de RMSE, una métrica de error cuyo computo solo difiere en la aplicación de una raíz cuadrada al valor de MSE. El objetivo es poder establecer una comparativa sin unidades cuadráticas, facilitando su comprensión. Las unidades tanto de RMSE como MAE son dólares.

	MAE	RMSE	MAPE
BTCUSDT	38873.0381	39758.4564	99.3484268
XRPUSDT	1.09153093	1.13150521	90.9592056
EOSUSDT	8.8210973	9.19784181	30.8824902
ADAUSDT	1.65586969	1.71261719	90.0822067
ETHUSDT	2615.35043	2689.22888	93.167572
BNBUSDT	385.71843	402.561113	173.372009
LTCUSDT	222.111819	230.562526	64.6515884
TRXUSDT	0.09577199	0.09991415	97.2046585
VETUSDT	0.14404841	0.14917191	114.288231

Tabla 4.10: Métricas de error desescaladas para el modelo profundo

Observando los valores obtenidos, puede verse que el modelo baseline mejora sustancialmente a los modelos utilizados en este estudio, pudiendo así concluirse que o bien estos modelos realizan una predicción pobre, o que el desempeño de un modelo naive es considerablemente bueno dado este problema. De nuevo, se considera necesario puntualizar que una mala predicción no afecta a la labor comparativa que se

	MAE	RMSE	MAPE
BTCUSDT	39380.3099	40438.9715	186.088013
XRPUSDT	1.14046943	1.20226994	113.389961
EOSUSDT	8.87002215	9.18887665	28.170332
ADAUSDT	1.69959903	1.77611692	82.7716293
ETHUSDT	2668.5331	2748.75367	111.233917
BNBUSDT	402.559645	417.468799	226.985214
LTCUSDT	232.00737	241.673027	94.3776321
TRXUSDT	0.1038043	0.10798897	208.78891
VETUSDT	0.15015654	0.15652438	260.011871

Tabla 4.11: Métricas de error desescaladas para el modelo MLP

busca en este estudio y abre la puerta a trabajos futuros que mejoren esta labor predictiva, por ejemplo, añadiendo el uso de variables exógenas que aporten más información que únicamente la contenida en los datos históricos del precio.

También es importante mencionar que tanto nuestras redes neuronales como el modelo baseline quedan lejos de los resultados obtenidos por un humano experto, dando lugar a un gran trabajo en la mejora de la labor predictiva de estos métodos en trabajos futuros.

4.3 Predicción realizada por los distintos modelos

En esta sección se discutirán los resultados de la predicción de los distintos modelos sobre la serie temporal. En primer lugar, mostraremos las gráficas con la serie temporal original y la predicción realizada por cada uno de nuestros modelos.

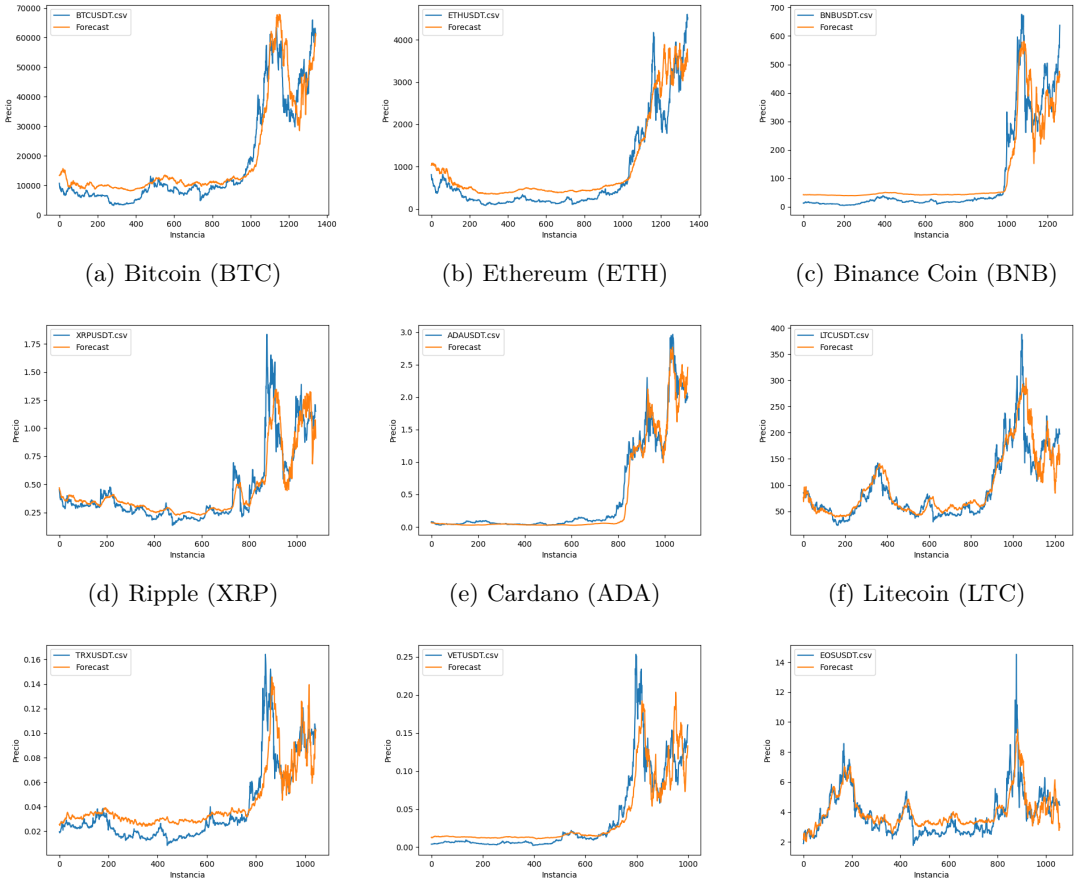
En la Figura 4.2 se muestran las gráficas comparativas entre predicciones y valores reales del modelo MLP, mientras que en la Figura 4.3 se muestran las del modelo profundo. Como primera conclusión que podemos obtener ante estos resultados, es el sorprendente buen funcionamiento que tienen ambos modelos, teniendo en cuenta que no se hace uso de variables exógenas que aporten información más allá del precio, ni siquiera el volumen de transacción diario.

De esta manera, puede verse que es posible realizar una predicción autorregresiva en función del precio de los distintos criptoactivos, pudiendo observarse que nuestros modelos han sido capaces de captar la tendencia alcista de estas criptomonedas. Puede verse que estos han sido capaces de captar la derivada de la serie temporal respecto del tiempo, ya que en los momentos en los que el precio de las distintas criptomonedas desciende, los modelos tienden a dar dicho resultado con cierto retardo, no dándonos información suficiente para ello únicamente la serie temporal, momento donde las variables exógenas cobrarían vital importancia. No obstante, está siendo capaz de extraer el comportamiento de la serie temporal a corto plazo, es decir, la derivada respecto del tiempo de esta. Esto puede además apoyarse en los valores de correlación obtenidos en la sección anterior calculados respecto a las predicciones que se muestran en estas gráficas.

Cabe destacar también el mejor funcionamiento del modelo profundo frente al modelo MLP. Pese a que el modelo MLP es también capaz de captar estas tendencias a corto plazo, podemos ver como sus predicciones oscilan mucho más que en el modelo profundo que aporta un resultado más suavizado. Puede verse que el ruido inherente a estas series temporales marca el funcionamiento del modelo MLP, dando unas predicciones mucho más volátiles.

Centrándonos concretamente en el modelo profundo, podemos ver como las gráficas de BTC ajustan de manera más exacta por ejemplo que otras como la de ETH. También puede observarse por ejemplo

como los picos alcistas de TRX han sido predichos mejor que por ejemplo los de LTC, la cual excepto en dicho pico ha tenido una predicción bastante exacta. También podemos destacar como EOS se ajusta de mejor manera que VET, pese a estar ambas en un rango de volatilidad similar. Así, de manera general, podemos ver un mejor ajuste para los modelos de EOS, TRX, y especialmente, LTC.

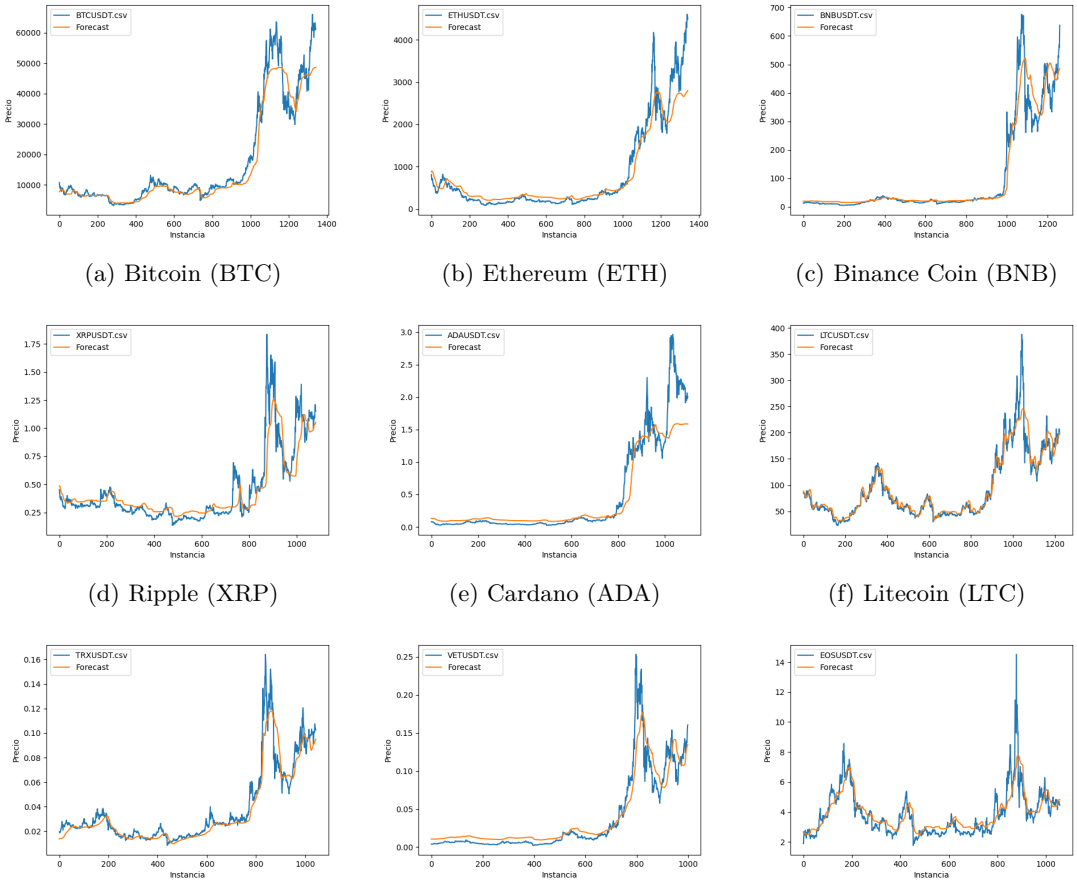


(a) Bitcoin (BTC) (b) Ethereum (ETH) (c) Binance Coin (BNB)

(d) Ripple (XRP) (e) Cardano (ADA) (f) Litecoin (LTC)

(g) Tron (TRX) (h) VeChain (VET) (i) Eos (EOS)

Figura 4.2: Gráficas con la predicción del modelo MLP sobre la serie temporal



(a) Bitcoin (BTC) (b) Ethereum (ETH) (c) Binance Coin (BNB)

(d) Ripple (XRP) (e) Cardano (ADA) (f) Litecoin (LTC)

(g) Tron (TRX) (h) VeChain (VET) (i) Eos (EOS)

Figura 4.3: Gráficas con la predicción del modelo profundo sobre la serie temporal

A modo de conclusión, se presenta a continuación la Tabla 4.12, en la que se muestran los valores del coeficiente de correlación de Pearson y de R^2 obtenidos entre la serie temporal y las predicciones realizadas por los distintos modelos, pudiendo observarse valores superiores a 0.9 para ambos coeficientes en todos los modelos. Se presentan estas métricas y no otras de error usadas en este estudio ya que, pese a ser una herramienta muy útil y en muchos casos necesaria a la hora de evaluar el desempeño predictivo de distintos modelos que trabajan con series temporales, pueden llevar a resultados engañosos sobre la utilidad de una predicción. [25]

	Pearson	R^2
BTCUSDT	0.989	0.925
XRPUSDT	0.972	0.919
EOSUSDT	0.962	0.907
ADAUSDT	0.914	0.908
ETHUSDT	0.979	0.943
BNBUSDT	0.957	0.951
LTCUSDT	0.965	0.961
TRXUSDT	0.978	0.945
VETUSDT	0.979	0.963

Tabla 4.12: Valores del coeficiente de correlación de Pearson y de R^2 entre la serie temporal y las predicciones realizadas por los distintos modelos

Capítulo 5

Conclusiones y líneas futuras

En este apartado se resumen las conclusiones obtenidas y se proponen futuras líneas de investigación que se deriven del trabajo realizado.

5.1 Conclusiones

El establecimiento de una comparativa de las características de las distintas criptomonedas aporta información muy relevante de cara a realizar una satisfactoria labor predictiva sobre las mismas.

Con el objetivo de realizar dicha comparativa se han construido dos modelos basados en redes neuronales, uno sencillo basado en MLP y otro más complejo. Tras la creación de estos, se procedió a la mejora de los mismos de cara a obtener una predicción mejor que reforzase la labor comparativa que se afronta en este proyecto. También se han realizado distintos análisis sobre las series temporales empleadas con vistas a identificar características que mejoraran dicho proceso predictivo.

Hemos podido comprobar en primer lugar, que si bien los modelos empleados no han tenido un desempeño notable a la hora de realizar las predicciones, hecho no sorprendente, ya que el análisis solo fue realizado sobre el precio de cierre y no se hizo uso alguno de variables exógenas [6], si han superado las expectativas que podría tenerse sobre ellos, destacando especialmente la capacidad que han tenido los mismos para extraer la información sobre las tendencias y derivadas de las distintas series temporales, obteniendo unos muy buenos valores de correlación entre valores reales y predicciones, similares para todas las criptomonedas

Además, como pudo observarse incluso con el modelo original, el cual demostró un exceso de complejidad notable en la mayoría de criptomonedas dado el sobreaprendizaje que se producía, pudimos ver como los modelos de LTC y EOS destacaban frente al resto. Esto invita a pensar que el conjunto de datos utilizado para este estudio se encuentra sesgado, dado que los datos se encuentran en el momento de una fuerte tendencia alcista que se produjo con la reciente burbuja de las criptomonedas. Así, estos mejores ajustes de estas dos criptomonedas pueden deberse a la mayor presencia de ruido en sus respectivas series temporales, hecho el que puede apoyarse en ser las dos con menor valor de autocorrelación de todo el conjunto a estudio.

Esto sugiere que, de hecho, estas mismas dos criptomonedas son más difícilmente predecibles que el resto, ya que es esta presencia de ruido la que provoca un menor número de mínimos locales en la función de pérdida en comparación con el resto de criptomonedas donde el sesgo es mayor, motivo por el cual se ajustan mejor a un modelo generalmente excesivamente complejo. A su vez, de entre el resto

de criptomonedas no puede concluirse que ninguna sea más fácilmente predecible que el resto con los resultados obtenidos.

5.2 Líneas futuras

Los resultados obtenidos abren la puerta a la mejora de los modelos empleados. así como a la utilización de nuevos métodos con este objetivo. Especialmente importante es el hecho de haber obtenido desempeños similares sobre casi todas las series temporales, hecho que abre la puerta al desarrollo de métodos predictivos generalistas más sofisticados. A su vez, también abre una línea de investigación a la hora de la inclusión de nuevos datos sobre los últimos meses, así como el uso de técnicas de *data augmentation* con objeto de ampliar el número de instancias disponibles dentro de la serie temporal.

Es especialmente importante destacar también el uso de variables exógenas para la mejora en la labor predictiva de los modelos, ya que existe gran cantidad de información no accesible dentro de los propios precios históricos, así como la ampliación de los modelos con técnicas más novedosas en el tratamiento de series temporales, como pueden ser capas de Atención Multi-Cabeza (del inglés *Multi-Head Attention*) [26] o arquitecturas que utilicen transformadores. Otro punto que podría mejorar significativamente el desempeño de los modelos predictivos es la inclusión de datos de precio intradía, ya que la información sobre las tendencias de precio de un mismo día son totalmente inaccesibles dentro de agregaciones de datos mayores.

Bibliografía

[1] M. T. Hagan, H. B. Demuth, and O. D. Jesús, "An introduction to the use of neural networks in control systems," *International Journal of Robust and Nonlinear Control*, vol. 12, no. 11, pp. 959–985, Sep. 2002. [Online]. Available: https://doi.org/10.1002/rnc.727

[2] P. Skalski, "Gentle dive into math behind convolutional neural networks," Apr 2019. [Online]. Available: https://towardsdatascience.com/gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9

[3] G. T. Hurtado Cuellar, "Factores que influyen en el comportamiento del bitcoin y su volatilidad," *Universidad Agustiniana*, 2019.

[4] S. McNally, J. Roche, and S. Caton, "Predicting the price of bitcoin using machine learning," in *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, 2018, pp. 339–343.

[5] S. G. Quek, G. Selvachandran, J. H. Tan, H. Y. A. Thiang, N. T. Tuan, and L. H. Son, "A new hybrid model of fuzzy time series and genetic algorithm based machine learning algorithm: A case study of forecasting prices of nine types of major cryptocurrencies," *Big Data Research*, vol. 28, p. 100315, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214579622000090

[6] F. Kjaerland, A. Khazal, E. A. Krogstad, F. B. G. NordstrÃ¸m, and A. Oust, "An analysis of bitcoinâs price dynamics," *Journal of Risk and Financial Management*, vol. 11, no. 4, 2018. [Online]. Available: https://www.mdpi.com/1911-8074/11/4/63

[7] V. Troster, A. K. Tiwari, M. Shahbaz, and D. N. Macedo, "Bitcoin returns and risk: A general garch and gas analysis," *Finance Research Letters*, vol. 30, pp. 187–193, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1544612318304859

[8] N. A. Bakar and S. Rosbi, "Autoregressive integrated moving average (ARIMA) model for forecasting cryptocurrency exchange rate in high volatility environment: A new insight of bitcoin transaction," *International Journal of Advanced Engineering Research and Science*, vol. 4, no. 11, pp. 130–137, 2017. [Online]. Available: https://doi.org/10.22161/ijaers.4.11.20

[9] N. Uras, L. Marchesi, M. Marchesi, and R. Tonelli, "Forecasting bitcoin closing price series using linear regression and neural networks models," *PeerJ Comput. Sci.*, vol. 6, no. e279, p. e279, Jul. 2020.

[10] Z. H. Munim, M. H. Shakil, and I. Alon, "Next-day bitcoin price forecast," *Journal of Risk and Financial Management*, vol. 12, no. 2, 2019. [Online]. Available: https://www.mdpi.com/1911-8074/12/2/103

[11] N. Mangla, A. Bhat, G. Avabratha, and N. Bhat, "Bitcoin price prediction using machine learning," *International Joutnal of Information And Computing Science*, vol. 6, no. 5, pp. 318–320, 2019.

[12] A. G. Salman, Y. Heryadi, E. Abdurahman, and W. Suparta, "Weather forecasting using merged long short-term memory model," *Bulletin of Electrical Engineering and Informatics*, vol. 7, no. 3, pp. 377–385, sep 2018. [Online]. Available: https://doi.org/10.11591%2Feei.v7i3.1181

[13] M. Roondiwala, H. Patel, and S. Varma, "Predicting stock prices using lstm," *International Journal of Science and Research (IJSR)*, vol. 6, no. 4, pp. 1754–1756, 2017.

[14] D. Shah, W. Campbell, and F. H. Zulkernine, "A comparative study of lstm and dnn for stock market forecasting," in *2018 IEEE international conference on big data (big data)*. IEEE, 2018, pp. 4148–4155.

[15] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "A comparison of arima and lstm in forecasting time series," in *2018 17th IEEE international conference on machine learning and applications (ICMLA)*. IEEE, 2018, pp. 1394–1401.

[16] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Dec 2008, accessed: 2015-07-01. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[17] Z. Chen, C. Li, and W. Sun, "Bitcoin price prediction using machine learning: An approach to sample dimension engineering," *Journal of Computational and Applied Mathematics*, vol. 365, p. 112395, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S037704271930398X

[18] E. Akyildirim, A. Goncu, and A. Sensoy, "Prediction of cryptocurrency returns using machine learning," *Annals of Operations Research*, vol. 297, no. 1-2, pp. 3–36, Apr. 2020. [Online]. Available: https://doi.org/10.1007/s10479-020-03575-y

[19] E. F. Fama, "Market efficiency, long-term returns, and behavioral finance," *Journal of financial economics*, vol. 49, no. 3, pp. 283–306, 1998.

[20] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[21] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0893608089900208

[22] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. [Online]. Available: https://doi.org/10.1038/nature14539

[23] A. M. Schäfer and H. G. Zimmermann, "Recurrent neural networks are universal approximators," in *Artificial Neural Networks – ICANN 2006*, S. D. Kollias, A. Stafylopatis, W. Duch, and E. Oja, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 632–640.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[25] M. J. Owens, "Time-window approaches to space-weather forecast metrics: A solar wind case study," *Space Weather*, vol. 16, no. 11, pp. 1847–1861, Nov. 2018. [Online]. Available: https://doi.org/10.1029/2018sw002059

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá