Predicción del precio del bitcoin mediante redes neuronales recurrentes

Daniel Ramos Hoogwout
Tutor: Dr. Victor Emilio Troster

Treball de fi de Màster Universitari en Analisi de Dades Massives en Economia i Empresa (MADM)

Universitat de les Illes Balears
07122 Palma de Mallorca
dramoshoogwout@gmail.com

Resumen

En los últimos años se ha incrementado exponencialmente la utilización de criptomonedas, que pretenden sustituir al dinero fiduciario emitido por los bancos centrales. Este trabajo busca predecir el precio del Bitcoin, la criptomoneda por excelencia, mediante redes neuronales recurrentes (RNR). Se emplean redes neuronales Long Short Term Memory (LSTM), un tipo de RNR que, gracias a su celda de memoria, permite almacenar información del pasado para hacer predicciones más precisas. En las predicciones se utiliza el método de *rolling window*, que emplea un número definido de observaciones pasadas para predecir la siguiente observación de manera recurrente con cuatro tamaños distintos, y se elige el mejor modelo según el MAE (del inglés, *mean absolute error*) y RMSE (del inglés, *root mean squared error*). Finalmente, se contrasta si el modelo elegido predice mejor que el resto mediante el contraste de Diebold-Mariano. Los resultados obtenidos confirman que el uso de RNR puede ser útil para predecir el precio del Bitcoin.

Palabras clave: Predicción, redes neuronales recurrentes, LSTM, Bitcoin, criptomoneda.

Abstract

In recent years, the use of cryptocurrencies has increased exponentially, which are virtual currencies that aim to replace fiduciary money issued by central banks. In this study, we predict the price of Bitcoin, the cryptocurrency by excellence, through recurrent neural networks (RNN). We use Long Short-Term Memory (LSTM) neural networks, a type of RNN that stores past information–due to its memory cell–to make more accurate predictions. We employ the rolling-window method in the forecasts by using a fixed number of past observations to predict the next observation recursively, with four different window sizes; we select the best model according to the MAE (mean absolute error) and RMSE (root mean squared error). Finally, we apply the Diebold-Mariano test to verify whether the selected model outperforms all other competing models. Our results show that RNN are useful to predict the price of Bitcoin.

Keywords: Forecast, Recurrent Neural Networks, LSTM, Bitcoin, cryptocurrencies.

1. Introducción

En la última década la popularidad de las criptomonedas ha crecido de manera exponencial gracias su innovador sistema de transacciones que permite a sus usuarios hacer movimientos monetarios especulativos o de divisa alternativa de manera instantánea, con un coste bajo y sin la intervención del sistema bancario, así como tampoco de la intervención del gobierno. Este sistema está basado en la tecnología *blockchain* [1], una tecnología en la cual los propios usuarios validan las transacciones del resto de usuarios de la red.

Debido a la creciente demanda de este tipo de activos como instrumento financiero alternativo se hace especialmente interesante predecir el precio de estas criptomonedas, tal y como ya se hace con otro tipo de divisas en el mercado de divisas FOREX (abreviatura del término en inglés *foreign exchange*)

[2, 3, 4] u otros activos financieros, como las acciones de las empresas [5, 6, 7, 8], mediante redes neuronales.

En este estudio se utilizará como criptomoneda de referencia el Bitcoin. Esta moneda virtual fue el primer testimonio de lo que hoy se conoce como criptomoneda que, gracias a su arquitectura, consiguió resolver varios problemas para hacer viable su uso.

La característica más importante es que es descentralizada, es decir, se gestiona y mantiene mediante un grupo de voluntarios, no hay un órgano central que lo controle ni intermediarios. Su suministro es limitado ya que es controlado por un algoritmo subyacente que reduce la creación de bitcoins hasta que se llegue a la cifra máxima de 21 millones, a diferencia de los bancos centrales que pueden emitir dinero sin límites. Las transacciones son anónimas, sus usuarios no tienen que identificarse para operar en la red y estas transacciones no son reversibles, ya que una vez realizadas no pueden modificarse. La última característica es la divisibilidad, la unidad más pequeña es una cien millonésima parte de la moneda, lo que permite hacer microtransacciones que el dinero electrónico tradicional no puede realizar [9].

Las redes neuronales recurrentes representan uno de los algoritmos más avanzados que existen en el mundo del *deep learning* supervisado. Como ya su propio nombre indica, una red neuronal intenta imitar un cerebro humano para resolver problemas y, en este caso, la red neuronal recurrente intenta imitar el lóbulo frontal, la parte del cerebro humano que se encarga de la memoria a corto plazo. Haciendo uso de esta memoria, el algoritmo es capaz de reaccionar a situaciones futuras con información del pasado y, por este motivo, hace que este tipo de algoritmos sean tan potentes para predecir series temporales.

Las redes neuronales recurrentes, en concreto, la variante de *Long Short-Term Memory* (LSTM) son redes neuronales muy adecuadas para datos de series temporales, ya que gracias a su estructura incluyen una celda de memoria que permite mantener información por periodos prolongados en el tiempo, venciendo así al problema del gradiente descendente, un problema inherente de las redes neuronales recurrentes clásicas [10].

Este tipo de redes se ha aplicado con resultados notables en aplicaciones tales como reconocimiento de voz [11], lenguaje natural [12] y síntesis de voz [13]. Por lo tanto, se va a utilizar este tipo de redes neuronales en este trabajo para intentar predecir el precio del Bitcoin, haciendo uso de datos de series temporales con su precio pasado poder hacer una predicción lo más precisa posible.

Este estudio utiliza cuatro tamaños de *rolling window* [14] distintos para generar modelos LSTM y escoge el mejor de ellos utilizando las medidas MAE, RMSE y el contraste de Diebold-Mariano [15]; se confirma que el mejor modelo es el de 90 días. Estos resultados concluyen que las redes LSTM pueden ser útiles a la hora de predecir el precio de apertura del Bitcoin, concordando con otros estudios similares, que demuestran que el precio de las criptomonedas es predecible [16], y que las redes LSTM representan algoritmos con excelentes resultados a la hora de predecir el precio de las criptomonedas [17]. Varios estudios [18, 19] han utilizado una metodología similar a la hora de generar estos modelos con resultados similares.

El resto de este estudio está organizado como se explica a continuación. La Sección 2 describe los conceptos de red neuronal recurrente (RNR) que se van a utilizar en el trabajo. La Sección 3 explica la metodología utilizada a la hora de seleccionar y ajusta la red neuronal, así como las técnicas de series temporales utilizadas para valorar las predicciones fuera de la muestra. La Sección 4 presenta el análisis empírico y el desempeño de los distintos modelos de predicción del precio del Bitcoin. Finalmente, la Sección 5 concluye este estudio.

2. Redes neuronales recurrentes

2.1. Redes neuronales recurrentes

Una red neuronal recurrente (RNR) es un tipo de red neuronal que dentro de su estructura puede retroalimentarse, es decir, utiliza su propio *feedback* para influenciar la entrada actual y la salida. Tal y como se puede observar en la Figura *1*, la red neuronal es capaz de modificar la información de entrada utilizando información que tenía previamente disponible en la capa oculta.

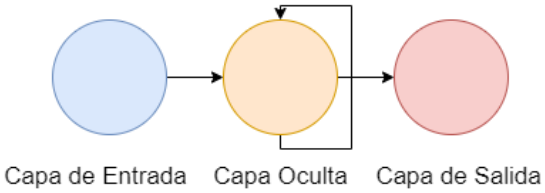

Figura 1: Diagrama de la red neuronal recurrente

La salida de una red neuronal recurrente depende de los elementos previos de la propia secuencia. Esto genera una característica distintiva, la red neuronal recurrente comparte el mismo parámetro de peso dentro de cada capa de la red [20]. Es decir, cada vez que la red actualiza sus pesos debe propagarse hacia atrás para actualizar todos y cada uno de los pesos de las capas que constituyen la red, generando así un posible desvanecimiento del gradiente.

Este problema que presentan las redes neuronales recurrentes fue descubierto por Josep Hochreiter [21]. El problema radica que la red neuronal utiliza

un algoritmo de gradiente descendente para encontrar el mínimo global de la función de costes, que es la configuración óptima de la red.

En la *2* se puede observar la estructura de varios módulos conectados entre sí de una RNR, donde el módulo central permite ver el funcionamiento interno de ellos.

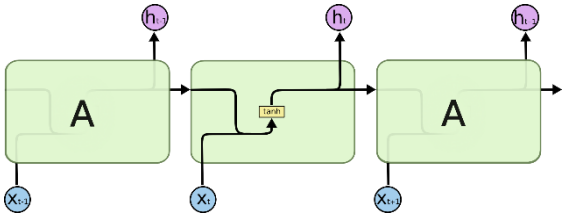

Figura 2: Estructura de un módulo RNR clásico
Fuente: Colah's Blog [22]

En el caso de las redes neuronales recurrentes, la información pasada se utiliza como *input* para determinar la predicción presente; por lo tanto, cuando se actualizan los pesos para minimizar el error, se tendrá que actualizar todas las neuronas que han participado en el proceso tal y como se puede observar en la Figura *3*; en este caso, todas las capas ocultas (en amarillo) tendrán que actualizar sus pesos.

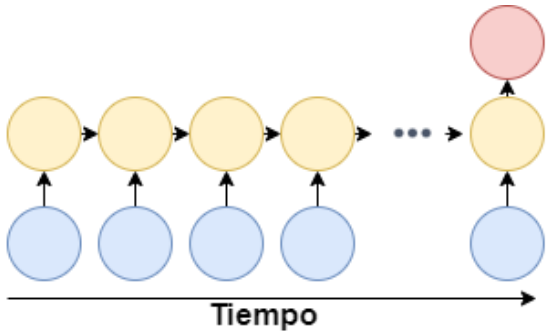

Figura 3: Diagrama de una RNR *Many-to-one*

Al principio del proceso se asignan pesos aleatorios con valores cercanos a cero, y entonces se comienza a entrenar la red; al multiplicar todos los momentos pasados por un valor cercano a cero el gradiente se reduce tras cada multiplicación, por lo que al final se hace cada vez más difícil actualizar los pesos de la red y aumenta el tiempo en el que se obtiene el resultado final. De ahí que a este problema se le conoce como desvanecimiento del gradiente.

2.2. LSTM

El LSTM (*Long short-term memory*, en inglés) es una variación de la red neuronal recurrente clásica que soluciona el problema del desvanecimiento del gradiente, comentado en la subsección anterior. Fue una de las soluciones propuestas por Josep Hochreiter y Jürgen Schmidhuber [10]. Tal y como se puede apreciar en la Figura 4, esta evolución añade celdas de entrada, salida y olvido. Gracias a esta nueva arquitectura, se separan las celdas de memoria (C_t) y los valores de salida de la red (h_t), evitando así el desvanecimiento del gradiente. Cabe recordar que, en la Figura 4, C_t, h_t y X_t son vectores, es decir, son capas completas de neuronas.

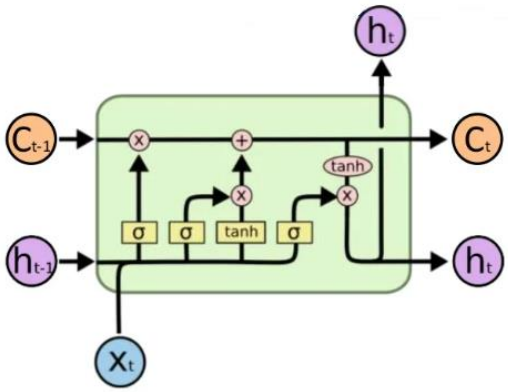

Figura 4: Estructura de un módulo LSTM
Fuente: Adaptado de [22]

Haciendo uso esta modificación de red neuronal recurrente, empleando técnicas de predicción de series temporales tales como *rolling window* y ajustando los hiperparámetros del modelo para tener resultados consistentes se ha demostrado que la red LSTM es una herramienta útil a la hora de hacer predicciones sobre el precio de apertura del Bitcoin.

3. Metodología

En esta sección, se expone la metodología utilizada para obtener el modelo de redes neuronales recurrentes que se empleará para predecir el precio del bitcoin. Generalmente, una predicción se basa en la descomposición de una serie temporal (y_t), la cual es constituida por tres componentes:

$$y_t = T_t + S_t + I_t, \qquad (1)$$

donde la tendencia T_t son comportamientos suaves de la serie a largo plazo, la estacionalidad S_t son movimientos de oscilación dentro del año y el componente irregular I_t son variaciones aleatorias alrededor de las componentes anteriores. Es interesante aislar estas distintas componentes para entender su comportamiento y así poder hacer predicciones. Por tanto, el objetivo de la red es capturar este tipo de oscilaciones, aprender los patrones que siguen y así poder hacer predicciones.

3.1. Construcción de la red neuronal

Para la construcción de la red neuronal se va a utilizar una estructura compuesta por 4 capas de entrada

interconectadas, que contienen 50 neuronas en cada capa dándole así una elevada dimensionalidad para que el modelo pueda capturar correlaciones en el precio del bitcoin a un nivel mucho más complejo que si el modelo fuese sencillo ya que no podría capturar correlaciones más avanzadas.

Tras cada capa de entrada se le añade una capa de *dropout*, que es una capa de olvido que se utiliza para desactivar aleatoriamente neuronas de la capa anterior y solo un cierto porcentaje de ellas pase la información hacia delante para prevenir el sobreajuste.

Este hiperparámetro se mantendrá constante para todos los modelos y se ha situado en un 20 %, por tanto, solo un 80 % de la información pasa de una capa de entrada a otra.

3.2. Complejidad de entrenamiento

A priori las redes neuronales pueden parecer modelos sencillos que imitan a un cerebro humano y pueden aplicarse fácilmente a todo tipo de problemas; nada más lejos de la realidad, son modelos muy complejos que requieren de ajustes muy minuciosos para poder obtener resultados aceptables. Las redes neuronales recurrentes son especialmente complejas a la hora de ser entrenadas. Aparte del problema del desvanecimiento del gradiente [21] comentado anteriormente, también existe un grado de aleatoriedad muy elevado al iniciar el modelo que puede afectar el ritmo de aprendizaje o incluso que el modelo aprenda de manera ilógica. Asimismo, la estructura o arquitectura escogida e incluso los hiperparámetros escogidos pueden tener una importancia significativa.

Todos estos factores juegan un papel fundamental en la dificultad de entrenar una red neuronal, y entender los efectos que tienen estos factores está aún por determinar en investigaciones en curso [23]. Por lo tanto, entrenar este tipo de redes requiere de un nivel de pericia elevado y numerosos intentos de prueba y error hasta alcanzar objetivos aceptables. Por esta razón, en este estudio se va a intentar limitar el número de variables escogidas arbitrariamente para poder obtener resultados objetivos y comparables.

3.3. Modelo inicial e hiperparametrización

Se comienza con un modelo sencillo para establecer una base sobre la cual introducir cambios y mejorar el poder predictivo del modelo. El modelo inicial solo toma como *input* los precios de apertura pasados, por lo que inicialmente podría describirse de la siguiente manera:

$$y_t = \alpha + \theta_1 y_{t-1} + \cdots + \theta_p y_{t-p} + \epsilon_t, \qquad (2)$$

donde $\epsilon_t \overset{i.i.d.}{\sim} N(0, \sigma_\epsilon^2)$ es un ruido blanco. La Ecuación (2) tendría la forma de un AR(p), un modelo autorregresivo de orden p, donde p sería el número de periodos atrás en el tiempo que la red neuronal es capaz de observar.

Cabe destacar que los pesos iniciales en las redes neuronales se asignan de manera completamente aleatoria; por tanto, es muy importante fijar esta aleatoriedad con una semilla para poder replicar los resultados y compararlos con otros modelos.

En el proceso de entrenar una red neuronal hay varios parámetros que pueden alterarse para obtener mejores resultados. En este caso los más importantes son el *batch size*, los *epochs* y el periodo en el cual la red neuronal puede observar valores pasados.

El *batch size* es el número de muestras para trabajar antes de actualizar los parámetros internos del modelo. Es decir, el conjunto de datos se va a dividir en muestras de 32 unidades tras las cuales los parámetros del modelo se actualizarán [24].

Un buen tamaño por defecto es 32, ya que usar tamaños pequeños permite mejorar la estabilidad y el rendimiento de generalización [25], aparte de que el proceso converge rápidamente para el coste computacional que supone.

Los *epochs* definen el número de veces que el algoritmo de aprendizaje va a digerir el conjunto de datos de entreno [24]. Este valor va a ser seleccionado automáticamente por el propio algoritmo de aprendizaje ya que se ha especificado un máximo de 100 *epochs* con una paciencia de un 10 %; es decir, si tras 10 *epochs* no hay una mejora en el modelo, el algoritmo detiene el entrenamiento. Esto es importante, dado que al aprender relativamente rápido el modelo puede sobreajustarse con facilidad al aumentar el número de *epochs*.

Finalmente, el periodo en el cual tiene visión la red neuronal se ha establecido en 30, 60, 90 y 120 días. Esto significa que, para hacer la predicción de un día, la red neuronal tendrá acceso a la información de los p días anteriores definidos por este periodo, a esta metodología de predicción se la conoce como *rolling window* [14] –ventana móvil, en inglés–.

3.4. Evolución del modelo

Una vez se obtiene un modelo de partida es interesante observar si la inclusión de otras variables aumentan el poder predictivo del modelo. En este caso, si se incluye otra variable (x_t), el nuevo modelo se representaría de la siguiente manera:

$$y_t = \beta_{10} + \beta_{11} y_{t-1} + \cdots + \beta_{1p} y_{t-p} + \gamma_{11} x_{t-1} + \cdots + \gamma_{1p} x_{t-p} + u_{1t}, \qquad (3)$$

$$x_t = \beta_{20} + \beta_{21} y_{t-1} + \cdots + \beta_{2p} y_{t-p} + \gamma_{21} x_{t-1} + \cdots + \gamma_{2p} x_{t-p} + u_{2t}, \qquad (4)$$

donde $u_{1t} \overset{i.i.d.}{\sim} N(0, \sigma_{u_1}^2)$ y $u_{2t} \overset{i.i.d.}{\sim} N(0, \sigma_{u_2}^2)$ son procesos ruido blanco. Las Ecuaciones (3)-(4) muestran un vector autorregresivo de orden p, VAR(p), un algoritmo de predicción que es utilizado cuando dos o más series temporales se influencian entre ellas. Esta relación entre las series temporales y_t y x_t es bidireccional, donde los valores pasados de una serie influencian la otra y viceversa.

Es interesante analizar si la red puede observar el comportamiento de otra serie temporal tales como el precio de cierre, otra criptomoneda o un índice bursátil, y en base a este comportamiento predecir mejor el precio de apertura del bitcoin.

3.5. Modelo final

El modelo final se escoge comparando los resultados de los distintos modelos creados a partir de la combinación de los hiperparámetros. Para evaluar los modelos se utilizarán dos medidas, en concreto, la raíz del error cuadrático medio (RMSE, del inglés *Root Mean Squared Error*) y el error absoluto medio (MAE, del inglés *Mean Absolute Error*) de predicción en $t+1$, definido como $\varepsilon_{t+1} = y_{t+1} - \hat{y}_{t+1|t}$, donde $\hat{y}_{t+1|t}$ es el valor predicho de y_{t+1} usando la información hasta el periodo *t*. Sea *H* el número de periodos fuera de la muestra y *T* el número de observaciones de entrenamiento, se define el RMSE como:

$$RMSE = \sqrt{\frac{\sum_{h=1}^{H}\left(y_{T+h} - \hat{y}_{t+h|T+h-1}\right)^2}{H}}. \qquad (5)$$

El MAE es el promedio del valor absoluto de los errores de predicción en $t+1$ (ε_{t+1}), y mide la magnitud media del error penalizando sobre todo los errores de predicción más elevados. Esto es deseable porque separarse mucho del precio real implica la posibilidad de tener pérdidas elevadas. El MAE se define como:

$$MAE = \frac{\sum_{h=1}^{H}\left|y_{T+h} - \hat{y}_{t+h|T+h-1}\right|}{H}. \qquad (6)$$

3.6. Predicción con *rolling window*

Las predicciones se suelen hacer sobre un periodo de tiempo. En cambio, en este estudio se toma una aproximación diferente, se hace una predicción mediante *rolling window*, es decir, se predice el precio del bitcoin para cada día utilizando los precios reales de los últimos *T* días definidos por el hiperparámetro periodo, el nombre que se le ha dado al hiperparámatro que indica a cuantas observaciones pasadas tiene acceso el algoritmo LSTM. Luego, cada día adelante se predice sobre los valores reales de la siguiente ventana, en vez de predecir sobre predicciones anteriores.

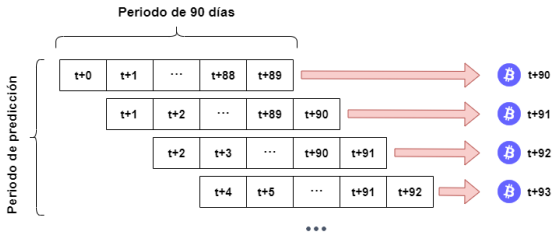

Figura 5: Predicción con *rolling window*

Tal y como se puede observar en la figura anterior, se hace una predicción recurrente donde el modelo puede observar los valores pasados de los últimos 90 días y en base a estos hace una predicción y a continuación repite el proceso siempre utilizando el precio real.

3.7. Comparación de la precisión predictiva

Se debe contrastar si el modelo final escogido tiene una precisión significativamente superior al resto de modelos, es decir, si el modelo con el periodo de 90 días predice mejor que el resto de modelos. Por consiguiente, se utiliza el contraste de Diebold-Mariano [15], que emplea los errores de predicción para comparar los modelos entre ellos y determinar si estos tienen precisiones significativamente distintas. Sean $\varepsilon_{1,T+h|T}$ y $\varepsilon_{2,T+h|T}$ los errores de predicción de los modelos 1 y 2 en $T+h$, respectivamente. Sea $\Delta^{T+h|T} = \varepsilon_{1,T+h|T}^2 - \varepsilon_{2,T+h|T}^2$ el diferencial de los errores cuadráticos de predicción de los modelos 1 y 2. El contraste de Diebold-Mariano [15] contrasta si ambos modelos tienen errores cuadráticos de predicción bajo la hipótesis nula frente a la hipótesis alternativa de que el modelo de predicción 1 tiene un error cuadrático de predicción menor:

$$H_0: E\left(\Delta^{T+h|T}\right) = 0,$$

$$H_A: E\left(\Delta^{T+h|T}\right) < 0.$$

El estadístico de prueba del contraste de Diebold-Mariano [15] bajo la hipótesis nula se define como:

$$DM = \frac{\sum_{h=1}^{H}\Delta^{T+h|T}}{\sqrt{\frac{2\pi\hat{f}_{\Delta}(0)}{T}}},$$

donde $2\pi\hat{f}_{\Delta}(0)$ es un estimador consistente de las autocovarianzas de $\Delta^{T+h|T}$. Bajo la hipótesis nula

H_0: $E\left(\Delta^{T+h|T}\right) = 0$, DM tiene una distribución normal estándar.

4. Análisis empírico

En esta sección se aplican las redes neuronales recurrentes para predecir el precio de apertura del bitcoin durante un periodo de tres meses. El modelo evoluciona desde un modelo inicial, donde solamente se tiene en cuenta el precio de apertura del bitcoin para utilizarlo como modelo de partida, y se ajusta y añade información hasta llegar al modelo final.

4.1. Datos

Para la muestra de entrenamiento se ha escogido el periodo que abarca desde el 17 de septiembre de 2015 hasta el 30 de noviembre de 2019. Se ha optado por este periodo porque comienza desde el dato más antiguo disponible en el portal *Yahoo Finance* (https://finance.yahoo.com/) hasta tres meses antes de que el efecto de la pandemia del COVID-19 fuese global, considerando marzo de 2020 como el inicio de la pandemia de COVID-19. Por tanto, la muestra de prueba comienza desde el 1 de diciembre de 2019 hasta el 29 de febrero de 2020.

Además, se ha escogido como muestra de validación el periodo que comprende desde el 1 de marzo de 2021 hasta el 27 de noviembre de 2021, se utiliza para dar una estimación imparcial de la habilidad predictiva del modelo.

El portal *Yahoo Finance* ofrece varios datos referentes a la cotización de activos de los cuales se han escogido el precio de apertura y de cierre diarios del bitcoin para entrenar la red neuronal. En un principio también se había incluido un índice bursátil, en concreto, el IBEX 35 ya que en los últimos meses se ha comentado mucho la correlación creciente entre el bitcoin y los índices bursátiles, desafortunadamente la inclusión del índice empeoró mucho las predicciones y se ha optado por eliminarlo de este análisis.

4.2. Matriz de entrada y estructura

Como se muestra en la Figura *6*, la entrada de información a la red neuronal está compuesta por una matriz tridimensional que contiene el precio de apertura y de cierre del bitcoin con 90 *time steps*¸ es decir, en cada momento del tiempo la red neuronal observará 90 días previos al día actual. Previamente, se había hecho uso de 60 días como período base, pero el incremento a 90 días supuso una mejora considerable en el modelo.

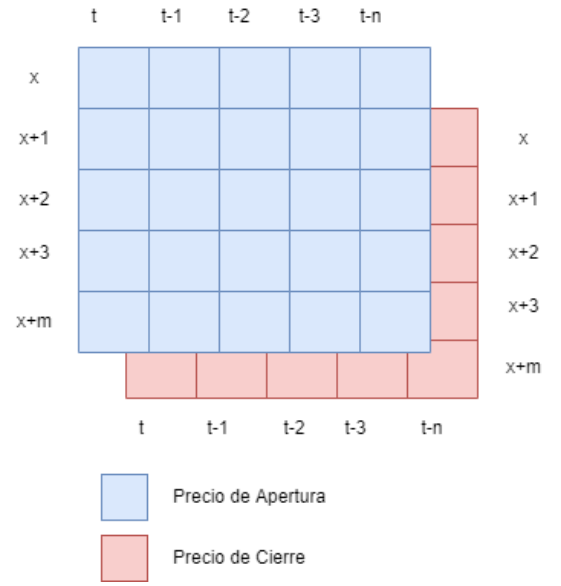

Figura 6: Matriz de entrada

Antes de introducir estos datos en la red neuronal estos se han normalizado a valores entre 0 y 1 para simplificar la entrada a la red neuronal, evitando así que una variable domine sobre otra en el caso de que sus magnitudes fuesen diferentes.

La red neuronal que procesa la matriz tridimensional está compuesta tal y como observar en la figura mostrada a continuación.

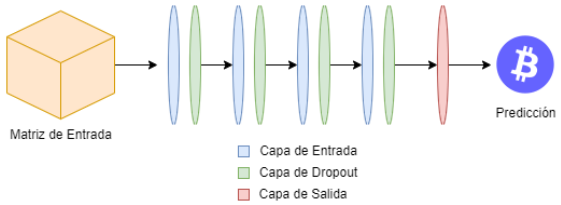

Figura 7: Estructura de la red neuronal

La matriz de entrada se introduce en la primera capa, que procesa la información, actualiza las neuronas y reenvía el 80 % de las neuronas actualizadas a la siguiente capa de entrada; este proceso ocurre sucesivamente hasta llegar a la capa de salida, donde se genera la predicción final.

Se ha utilizado la función *Early Stopping*, la cual se emplea para monitorizar si el modelo deja de minimizar el gradiente descendente, especificando una paciencia de 10 iteraciones (normalmente se debe situar en un 10 % de los *epochs* totales). De ahí que si en las siguientes 10 iteraciones el modelo no reduce la pérdida el aprendizaje del modelo se detiene. Al utilizar *batches* de un tamaño de 32 unidades, el modelo aprende con mucha rapidez y de no detener el entrenamiento puede ocasionar sobreajuste.

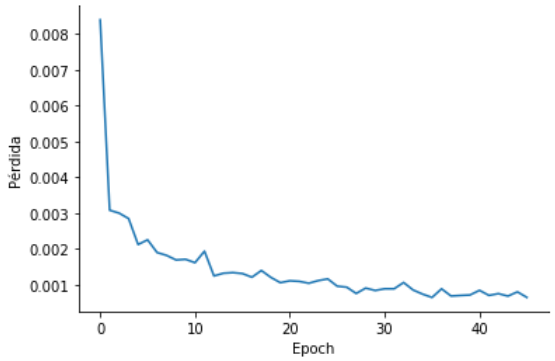

Figura 8: Ritmo de aprendizaje de la RNR

La Figura *8* muestra que el modelo aprende de manera rápida gracias a que la pérdida se reduce considerablemente en los *epochs* iniciales, posteriormente el modelo deja de aprender con tanta rapidez y el *Early Stopping* se activa a los 46 *epochs*, es decir, desde el *epoch* 36 no hay una disminución de la pérdida del gradiente descendente del modelo y, por tanto, selecciona los pesos para el modelo del *epoch* 36.

Una vez realizado el entrenamiento del modelo se predicen los siguientes 3 meses con la muestra de prueba y se obtiene la siguiente figura.

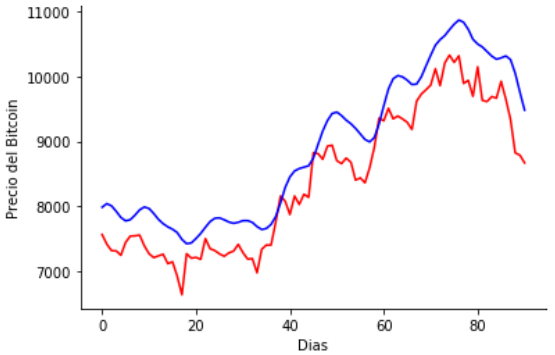

Figura 9: Predicción de la red neuronal (–) del precio del bitcoin (–): muestra de prueba

La figura anterior indica que la predicción en la muestra de prueba sigue correctamente la tendencia del precio real del bitcoin, aunque generalmente es superior al precio real.

Para poder comparar esta predicción con la de otros modelos se estima la raíz del error cuadrático medio (RMSE), así como también el error medio absoluto (MAE).

Finalmente, se aplican los modelos a una muestra de validación utilizando un período futuro, en concreto, en el que se comienza a predecir en junio de 2021 y se emplean los 90 días anteriores para hacer una predicción del precio futuro de un periodo hacia adelante del bitcoin.

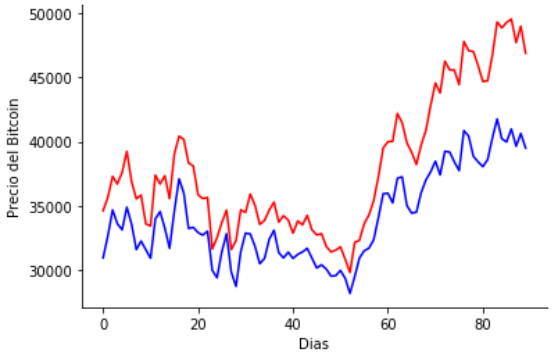

Figura 10: Predicción de la red neuronal (–) del precio del bitcoin (–): muestra de validación

La figura anterior señala que el modelo de 90 días también sigue correctamente la tendencia del precio del bitcoin, aunque en este caso el precio predicho es inferior al precio real. Se obtiene la raíz del error cuadrático medio (RMSE) y el error medio absoluto (MAE) para así poder comparar los modelos con los resultados de la muestra de prueba y determinar si ha habido sobreajuste.

Tabla 1. Desempeño de los modelos: muestra de prueba

Periodo	30	60	90	120
RMSE	497	**163**	552	201
MAE	469	**132**	506	159

Notas: La tabla muestra la raíz del error cuadrático medio (RMSE, del inglés *Root Mean Squared Error*) y el error absoluto medio (MAE, del inglés *Mean Absolute Error*) de predicción en $t+1$, definidos en las Ecuaciones (5) y (6). Se han calculado el RMSE y el MAE en la muestra de prueba para los distintos modelos dependiendo del periodo de su *rolling window*. Una vez realizada la predicción sobre un horizonte de 3 meses, se obtienen los errores de predicción y se comparan con el valor real utilizando el RMSE y MAE, cuanto menor es el error más precisa es la estimación. Se han señalado en negrita los valores más bajos del RMSE y MAE.

La Tabla 1 enseña la comparación del desempeño de los modelos en la muestra de prueba. El modelo con la mejor puntuación es el modelo con el periodo de *rolling window* de 60 días, ya que obtiene los menores valores de RMSE y MAE. Sin embargo, es la fase de prueba y al utilizar una *rolling window* las neuronas pueden sobreajustarse con facilidad a medida que va aumentando el número de *epochs*, debido a que al utilizar una *rolling window* el entrenamiento puede ver parte de la predicción final.

Tabla 2. Comparativa de los modelos: muestra de validación

Periodo	30	60	90	120
RMSE	15803	10318	**4487**	10192
MAE	14650	7997	**4005**	9177

Notas: La tabla muestra las medidas de RMSE y MAE de las Ecuaciones (5) y (6), respectivamente, en la muestra de validación para los distintos modelos dependiendo del periodo de

su *rolling window*. Una vez realizada la predicción sobre un horizonte de 3 meses, se obtienen los errores de predicción y se comparan con el valor real utilizando estas dos medidas, por lo que cuanto menor es el error más precisa es la estimación. Se han señalado en negrita los valores más bajos del RMSE y MAE.

La Tabla 2 muestra los resultados de la comparación del desempeño de los modelos en la muestra de validación, cuyos datos el modelo no ha podido emplear con anterioridad, con el fin de comprobar que el modelo no tiene sobreajuste. La Tabla 2 indica que los modelos que tenían puntuaciones muy bajas en la Tabla 1 en cambio han disparado sus errores de predicción en la muestra de validación. Además, el modelo con el periodo de 90 días, el cual tiene la puntuación más alta en el periodo de prueba, es el que se ajusta mejor en la muestra de validación. De hecho, tiene una puntuación en ambos scores de menos de la mitad del segundo mejor modelo, gracias a que este modelo no ha tenido un sobreajuste como el resto, tal y como demuestran las Tablas 1-2.

Tabla 3. Contraste de Diebold-Mariano sobre los distintos modelos

Modelos	DM	*P*-valor
90 – 30	-8.28	$<0.01^{***}$
90 – 60	-7.96	$<0.01^{***}$
90 – 120	-5.81	$<0.01^{***}$

Notas: La tabla muestra el estadístico DM, definido en la Subsección 3.7, y los *p*-valores del contraste de Diebold-Mariano [15], cuya hipótesis H_0 indica que los errores de predicción del modelo con el período de 90 días y del modelo de comparación son iguales, frente a la hipótesis H_A de que el modelo con el período de 90 días tiene un de error de predicción menor. Los asteriscos <<***>> indican un rechazo de H_0 al 1 % de significación.

La Tabla 3 muestra los resultados del contraste de Diebold-Mariano [15], definido en la Subsección 3.7, de la hipótesis nula de que los errores de predicción del modelo de 90 días son iguales al modelo de comparación –30, 60 o 120 días– frente a la hipótesis alternativa de que el modelo de 90 días tiene un error de predicción inferior. La Tabla 3 demuestra que los demás modelos en comparación con el modelo de período 90 no tienen precisiones de predicción iguales, porque se rechaza la hipótesis nula al 1 % de significación, para cada uno de los tres modelos de comparación. Por lo tanto, el modelo con el período de 90 días predice mejor que el resto al 1 % de significación.

5. Conclusiones

Las criptomonedas son activos que no están regulados ni controlados por ninguna institución, tampoco requieren intermediaros, por lo que su tiempo de transacción es mucho más reducido respecto al del sistema bancario convencional. La facilidad para operar con ellas y su proyección en el futuro hacen que el uso de este tipo de activos se haya popularizado mucho en la última década. Por estas razones, es interesante analizar si es posible predecir el precio futuro de este tipo de activos haciendo uso de sus precios pasados, tal y como también se hace con los índices bursátiles.

En este trabajo se han utilizado los precios de apertura y cierre del Bitcoin para intentar predecir su precio futuro mediante redes neuronales recurrentes (RNR), en concreto, *Long Short-Term Memory* (LSTM), uno de los algoritmos de *deep learning* más avanzados que existen que, gracias a su celda de memoria, permiten almacenar información del pasado para hacer predicciones más precisas.

Se ha construido una red con cuatro capas de entrada constituidas con 50 neuronas cada una e intercaladas con capas de *dropout* para evitar el sobreajuste. Este tipo de estructura ofrece una elevada dimensionalidad que ayuda a captar relaciones entre precios más complejas. Se ha escogido un *batch size* de 32 para que el proceso converja rápidamente dado el coste computacional que supone. Asimismo, se ha empleado un máximo de 100 *epochs* con una paciencia del 10 % a fin de evitar un sobreajuste del modelo.

Por último, se han seleccionado cuatro tamaños de *rolling window* –30, 60, 90 y 120 días– y se han puntuado los modelos según las medidas de MAE y RMSE; modelo que ha obtenido la mejor puntuación en la fase de validación es el de 90 días. Finalmente, se ha utilizado el contraste de Diebold-Mariano, confirmando que este modelo predecía mejor que el resto al 1 % de significación.

Este análisis abre las puertas a futuras investigaciones a la hora de realizar predicciones sobre el precio de las criptomonedas mediante redes LSTM, ya que se ha demostrado que son aptas para predecir el precio futuro haciendo uso del precio de apertura y cierre pasados. En trabajos futuros, sería interesante añadir al modelo índices bursátiles u otros indicadores que puedan ayudar a detectar cambios en el precio, mejorando así la capacidad predictiva de la red.

Apéndice A. Configuración de Anaconda

Para poder replicar los resultados descritos con anterioridad se tiene que configurar el entorno de desarrollo de una manera específica.

El autor de este estudio ha utilizado Anaconda para desarrollar la red neuronal, utilizando el paquete de Tensor Flow. Este paquete provoca errores en la configuración por defecto de Anaconda y es necesario crear un nuevo entorno de desarrollo en el cual se instale la versión de Numpy 1.18.5.

Anaconda instala por defecto la más reciente entre otras librerías que generan conflictos y la solución más sencilla es crear un entorno de desarrollo exclusivamente para Tensor Flow 2.3 en el cual únicamente deben instalarse la versión anteriormente comentada de Numpy, Pandas, Keras, Tensorflow y las librerías que se instalan automáticamente al seleccionar las anteriores.

Referencias

[1] D. Yaga, P. Mell, N. Roby, y K. Scarfone, «Blockchain Technology Overview», 2019, doi: 10.48550/ARXIV.1906.11078.

[2] K. K. Lai, L. Yu, y S. Wang, «A Neural Network and Web-Based Decision Support System for Forex Forecasting and Trading», en *Data Mining and Knowledge Management*, vol. 3327, Y. Shi, W. Xu, y Z. Chen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 243-253. doi: 10.1007/978-3-540-30537-8_27.

[3] J. Yao y C. L. Tan, «A case study on using neural networks to perform technical forecasting of forex», *Neurocomputing*, vol. 34, n.º 1-4, pp. 79-98, sep. 2000, doi: 10.1016/S0925-2312(00)00300-3.

[4] L. Ni, Y. Li, X. Wang, J. Zhang, J. Yu, y C. Qi, «Forecasting of Forex Time Series Data Based on Deep Learning», *Procedia Comput. Sci.*, vol. 147, pp. 647-652, 2019, doi: 10.1016/j.procs.2019.01.189.

[5] K. Kamijo y T. Tanigawa, «Stock price pattern recognition-a recurrent neural network approach», en *1990 IJCNN international joint conference on neural networks*, 1990, pp. 215-221.

[6] J.-Z. Wang, J.-J. Wang, Z.-G. Zhang, y S.-P. Guo, «Forecasting stock indices with back propagation neural network», *Expert Syst. Appl.*, vol. 38, n.º 11, pp. 14346-14355, oct. 2011, doi: 10.1016/j.eswa.2011.04.222.

[7] D. Olson y C. Mossman, «Neural network forecasts of Canadian stock returns using accounting ratios», *Int. J. Forecast.*, vol. 19, n.º 3, pp. 453-465, jul. 2003, doi: 10.1016/S0169-2070(02)00058-4.

[8] J. Qiu, B. Wang, y C. Zhou, «Forecasting stock prices with long-short term memory neural network based on attention mechanism», *PLOS ONE*, vol. 15, n.º 1, p. e0227222, ene. 2020, doi: 10.1371/journal.pone.0227222.

[9] America Digital News, «¿Cuál es la importancia del Bitcoin?» [En línea]. Disponible en: https://news.america-digital.com/que-es-bitcoin/#:~:text=Bitcoin%20resuelve%20el%20%C2%ABproblema%20del,de%20criptograf%C3%ADa%20e%20incentivos%20econ%C3%B3micos.

[10] S. Hochreiter y J. Schmidhuber, «Long Short-Term Memory», *Neural Comput.*, vol. 9, n.º 8, pp. 1735-1780, nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[11] H. Sak, A. Senior, y F. Beaufays, «Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition», 2014, doi: 10.48550/ARXIV.1402.1128.

[12] J. Cheng, L. Dong, y M. Lapata, «Long Short-Term Memory-Networks for Machine Reading», 2016, doi: 10.48550/ARXIV.1601.06733.

[13] X. Li y X. Wu, «Modeling speaker variability using long short-term memory networks for speech recognition», 2015.

[14] MathWorks, «Rolling-Window Analysis of Time-Series Models», *MathWorks*, 2022. https://www.mathworks.com/help/econ/rolling-window-estimation-of-state-space-models.html (accedido 3 de julio de 2022).

[15] F. X. Diebold y R. S. Mariano, «Comparing predictive accuracy», *J. Bus. Econ. Stat.*, vol. 20, n.º 1, pp. 134-144, 2002.

[16] W. Yiying y Z. Yeze, «Cryptocurrency price analysis with artificial intelligence», en *2019 5th International Conference on Information Management (ICIM)*, 2019, pp. 97-101.

[17] M. J. Hamayel y A. Y. Owda, «A Novel Cryptocurrency Price Prediction Model Using GRU, LSTM and bi-LSTM Machine Learning Algorithms», *AI*, vol. 2, n.º 4, pp. 477-496, 2021.

[18] K. Struga y O. Qirici, «Bitcoin Price Prediction with Neural Networks.», en *RTA-CSIT*, 2018, pp. 41-49.

[19] I. Nasirtafreshi, «Forecasting cryptocurrency prices using Recurrent Neural Network and Long Short-term Memory», *Data Knowl. Eng.*, vol. 139, p. 102009, may 2022, doi: 10.1016/j.datak.2022.102009.

[20] IBM Cloud Education, «Recurrent Neural Networks», *IBM*, 14 de septiembre de 2020. https://www.ibm.com/cloud/learn/recurrent-neural-networks (accedido 19 de mayo de 2022).

[21] S. Hochreiter, «The vanishing gradient problem during learning recurrent neural nets and problem solutions», *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 6, n.º 02, pp. 107-116, 1998.

[22] «Understanding LSTM Networks», *Colah.github.io*, 27 de agosto de 2015. https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (accedido 20 de mayo de 2022).

[23] R. Pascanu, T. Mikolov, y Y. Bengio, «On the difficulty of training recurrent neural networks», p. 9.

[24] J. Brownlee, «Difference Between a Batch and an Epoch in a Neural Network», *Machine Learning Mastery*, 20 de julio de 2018. https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/ (accedido 22 de mayo de 2022).
[25] J. Brownlee, «A Gentle Introduction to Mini-Batch Gradient Descent and How to Configure Batch Size», *Machine Learning Mastery*, julio de 2021. https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/ (accedido 22 de mayo de 2022).