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, una moneda virtual que pretende sustituir al dinero fiduciario emitido por los bancos centrales. Este tipo de activos ha crecido tanto en número como en capitalización al nivel de poder compararse incluso con índices bursátiles, por ello, el objetivo de este estudio es intentar predecir el precio del Bitcoin, la criptomoneda por excelencia. Este trabajo analiza el uso de redes neuronales recurrentes (RNR) para predecir el precio de apertura del Bitcoin, en concreto, 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. Para hacer la predicción se utilizará el método de Rolling Window, un método que emplea un número definido de observaciones pasadas para predecir la siguiente observación de manera recurrente con cuatro tamaños distintos y la elección del mejor modelo se establece por medio de la puntuación obtenida según las medidas de MAE (del inglés, mean absolute error) y RMSE (del inglés, root mean squared error) y contrastando finalmente si el modelo escogido es superior al resto mediante el contraste de Diebold-Mariano. Este análisis confirma que el uso de redes neuronales recurrentes puede ser útil a la hora de hacer predicciones sobre el precio de las criptomonedas y abre vía para futuras investigaciones mediante este método

**Palabras clave**: predicción, redes neuronales recurrentes, LSTM, Bitcoin, Cripto.

Abstract

In recent years, the use of cryptocurrencies has increased exponentially, a virtual currency that aims to replace fiduciary money issued by the central banks. This type of asset has grown both in number and in capitalization to the level of being able to be compared even with stock market indexes, for this reason the objective of this study is to try to predict the price of Bitcoin, the cryptocurrency by excellence, through a type of recurrent neural network, specifically, Long Short-Term Memory (LSTM), one of the most advanced deep learning algorithms to date, that thanks to its memory cell, allows to store past information to make more precise predictions.

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

1. Introducción

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, 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 incluye 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.

Este tipo de redes se han aplicado con resultados notables en aplicaciones como reconocimiento de voz, lenguaje natural, síntesis de voz… Por ello 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 entre otros indicadores para poder hacer una predicción lo más precisa posible.

* 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. 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

Por tanto, 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 [1]. En la Figura 2 se puede observar la estructura de varios módulos donde el módulo central permite ver el funcionamient

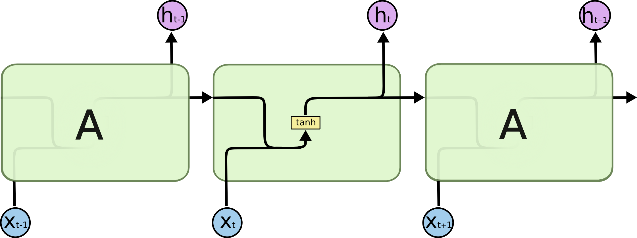


Figura 2: Estructura de un módulo RNR clásico

Fuente: Colah’s Blog [2]

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 [3].

El problema radica que en 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 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.



Figura 3: Diagrama de 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.

* 1. 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 [4]. Esta evolución añade celdas de entrada, salida y olvido.



Figura 4: Estructura de un módulo LSTM

Fuente: Adaptado de [2]

Gracias a esta nueva arquitectura, se separa las celdas de memoria () y los valores de salida de la red () evitando así el desvanecimiento del gradiente. Cabe recordar que en la figura anterior , y son vectores, es decir, son capas completas de neuronas.

1. 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 (la cual es constituida por tres componentes:

|  |  |
| --- | --- |
| , | (1) |

donde la tendencia *Tt* son comportamientos suaves de la serie a largo plazo, la estacionalidad *St* son movimientos de oscilación dentro del año y el componente irregular *It* 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.

* 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.

* 1. 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, porque 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 [3] 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 [5]. 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.

* 1. 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:

|  |  |
| --- | --- |
| , | (2) |

donde 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 [6].

Un buen tamaño por defecto es 32, ya que usar tamaños pequeños permite mejorar la estabilidad y el rendimiento de generalización [7], 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 [6]. 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* [8] *–*ventana móvil, en inglés–.

* 1. 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 (, el nuevo modelo se representaría de la siguiente manera:

|  |  |
| --- | --- |
| + | (3) |
|  | (4) |

donde y 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 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.

* 1. 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 , definido como , donde es el valor predicho de 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:

|  |  |
| --- | --- |
|  | (5) |

El MAE es el promedio del valor absoluto de los errores de predicción en (), 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:

|  |  |
| --- | --- |
|  | (6) |

* 1. 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.

* 1. 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 [9], que emplea los errores de predicción para comparar los modelos entre ellos y determinar si estos tienen precisiones significativamente distintas. Sean y los errores de predicción de los modelos 1 y 2 en , respectivamente. Sea el diferencial de los errores cuadráticos de predicción de los modelos 1 y 2. El contraste de Diebold-Mariano [9] 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:

|  |
| --- |
| *,* |
|  |

El estadístico de prueba del contraste de Diebold-Mariano [9] bajo la hipótesis nula se define como:

|  |
| --- |
|  |

donde es un estimador consistente de las autocovarianzas de. Bajo la hipótesis nula , DM tiene una distribución normal estándar.

1. 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.

* 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.

* 1. Matriz de entrada y estructura

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.

Gráfico, Histograma

Descripción generada automáticamente

Figura 8: Ritmo de aprendizaje de la RNR

La Figura 6 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.

Gráfico, Gráfico de líneas, Histograma

Descripción generada automáticamente

Figura 9: Predicción de la red neuronal (**–)** del precio de 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.

Gráfico, Histograma

Descripción generada automáticamenteFigura 10: Predicción de la red neuronal (–**)** del precio de 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 , 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. | | | | |

En la Tabla 1 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. | | | | |

En la Tabla 2 se utiliza una 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) y los *p*-valores del contraste de Diebold-Mariano[9], cuya hipótesis 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 de que el modelo con el período de 90 días tiene un de error de predicción menor. Los asteriscos <<\*\*\*>> indican un rechazo de al 1 % de significación. | | |

La Tabla 3 muestra los resultados del contraste de Diebold-Mariano [9], definido en la Subsección 2.7, 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.

1. Conclusiones

Las criptomonedas son activos que no están regulados ni controlados por ninguna institución, tampoco requieren de intermediaros y esto hace que el tiempo de transacción sea mucho más reducido respecto al sistema bancario. 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 índices bursátiles.

En este análisis 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 uno 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 así el proceso converge rápidamente para el coste computacional que supone, también se ha escogido un máximo de 100 *epochs* con una paciencia de 10% para evitar sobre ajustar el modelo.

Por último, se han seleccionado cuatro tamaños de *Rolling window*, en concreto: 30, 60, 90 y 120 días y se han puntuado los modelos según las medidas de MAE y RMSE donde se ha podido observar que el modelo que mejor puntuación obtenía en la fase de validación era el de 90 días. En última instancia se ha utilizado el contraste de Diebold-Mariano confirmando que este modelo predecía mejor que el resto.

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. Sería interesante la adición de índices bursátiles u otros indicadores que puedan ayudar a detectar cambios en el precio mejorando así la capacidad predictiva de la red.

1. Bibliografía

# 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.

# Modelos descartados

Modelo inicial

Modelo con un índice bursátil

Modelo con un periodo de 30 días

Modelo con un periodo de 60 días

Modelo con un periodo de 120 días

Referencias

[1] 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).

[2] «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).

[3] 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.

[4] 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.

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

[6] 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).

[7] 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).

[8] 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).

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