UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

ANÁLISIS PREDICTIVO DEL MERCADO DE LAS CRIPTOMONEDAS MEDIANTE MACHINE LEARNING.

EDUARDO SAAVEDRA GALILEA

2022

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

ANÁLISIS PREDICTIVO DEL MERCADO DE LAS CRIPTOMONEDAS MEDIANTE MACHINE LEARNING.

Autor
EDUARDO SAAVEDRA GALILEA

Tutor
ÁLVARO GUTIERRÉZ MARTIN

2022

Resumen

Con el avance actual de la tecnología, cada vez disponemos de mayor potencia de computación y herramientas para tratar grandes cantidades de datos y poder extraer información que añada valor a dichos datos. Estas mejoras en el campo de la informática han permitido la evolución del Aprendizaje de Máquina, también conocido como Machine Learning (ML), una rama de la Inteligencia Artificial (IA), que permite crear algoritmos de predicción mucho más precisos. Con estas técnicas es interesante analizar series temporales de datos para poder predecir datos futuros.

El mercado de las criptomonedas ofrece un escenario propicio para aplicar técnicas de ML, con el objetivo de predecir los comportamientos futuros de este. El presente Trabajo de Fin de Grado compara el desempeño de un sistema de redes neuronales recurrentes Long Short Term Memory (LSTM) con un algoritmo Gated Recurrent Unit (GRU). Además, se presenta un marco teórico para poder entender en profundidad el problema y su relevancia en el contexto actual.

Palabras clave: Redes neuronales, Machine Learning, criptomonedas, LSTM, series temporales, GRU.

Abstract

With the advance of technology, there has been a rise in computing power and tools to process large amounts of data and extract information that adds value to it. These improvements in informatics have allowed the evolution of Machine Learning (ML), a branch of Artificial Intelligence (AI), making it possible to create much more accurate prediction algorithms. It is interesting to analyze time-series data to predict future data prices with these techniques.

The cryptocurrency market offers an excellent scenario to use Machine Learning techniques to predict future time series results. This BSc thesis compares the performance of an Long Short Term Memory (LSTM) recurrent neural network system with a Gated Recurrent Unit (GRU) algorithm. In addition, a theoretical framework is presented to understand in depth the problem and its relevance in the current context.

Keywords: Neural networks, machine learning, cryptocurrencies, LSTM, GRU, temporal series.

Índice general

Resumen III

Abstract V

Índice General VII

Índice de Figuras IX

Índice de Tablas XI

Lista de Acrónimos XIII

1. Introducción 1
- 1.1. Motivación . . . 1
- 1.2. Objetivos . . . 1
- 1.3. Metodología . . . 2
- 1.4. Organización del TFG . . . 2

2. Estado del Arte 3
- 2.1. Una nueva moneda y un nuevo mercado: historia de la criptomoneda 3
- 2.2. Blockchain, la base de las criptomonedas . . . 4
- 2.3. Mercado de la criptomoneda . . . 5
- 2.4. Machine Learning . . . 5
  - 2.4.1. Aprendizaje Supervisado . . . 6
  - 2.4.2. Aprendizaje No Supervisado . . . 6
  - 2.4.3. Aprendizaje por Refuerzo . . . 6
- 2.5. Algoritmos para la Regresión . . . 7
- 2.6. Redes Neuronales . . . 7
- 2.7. Redes Neuronales Recurrentes . . . 9
- 2.8. Long Short Term Memory . . . 10
- 2.9. Gated Recurrent Units . . . 11
- 2.10. Árboles de Decisión . . . 12
- 2.11. Random Forest . . . 12
- 2.12. Evaluación de resultados . . . 12
- 2.13. Trabajos Relacionados . . . 14

3. Diseño del Sistema **15**
3.1. Tecnologías utilizadas 15
3.1.1. Python 16
3.1.2. Google Colab 17
3.2. Hardware 17
3.3. Recopilación de Datos 17
3.4. Preprocesado de los Datos 17
3.4.1. Correlación de los Datos 18
3.4.2. Transformación de los Datos 20
3.4.3. Algoritmo de Backpropagation 21
3.4.4. División del Dataset 22
3.5. Arquitectura de las Redes Neuronales 24
3.5.1. Long Short Term Memory 24
3.5.2. Gated Recurrent Unit 25
3.6. Hiperparámetros 26
3.6.1. Hiperparámetros de LSTM y GRU 26

4. Resultados **28**
4.1. LSTM 29
4.1.1. Resultados de los modelos 30
4.1.2. Evaluación de los Modelos LSTM 32
4.2. GRU 38
4.2.1. Resultados de los modelos 38
4.2.2. Evaluación de los modelos GRU 40
4.3. Discusión de los resultados 43

5. Conclusiones y Líneas Futuras **45**
5.1. Conclusiones 45
5.2. Líneas futuras 46

A. Aspectos económicos, sociales y ambientales **47**
A.1. Impacto social 47
A.2. Impacto económico 47
A.3. Impacto medioambiental 47
A.4. Responsabilidad ética y profesional 47

B. Presupuesto económico **48**

Índice de figuras

2.1. Capitalización total del mercado de la criptomoneda. 4
2.2. Dominio de capitalización total de mercado, % 5
2.3. Representación blockchain. 5
2.4. Tipos de aprendizaje. 7
2.5. Neurona 8
2.6. Perceptrón 8
2.7. Funciones de activación. 9
2.8. Red neuronal. 10
2.9. Red neuronal recurrente. 10
2.10. Neurona LSTM. 11
2.11. Neurona GRU. 11
2.12. Árbol de decisión 12

3.1. Diseño de los sistemas. 15
3.2. Características de Python. 16
3.3. Gráficos de velas. 18
3.4. Datos obtenidos mediante API Binance. 18
3.5. Todos los datos del BTC 19
3.6. Matriz de correlación. 20
3.7. Correlación de close. 20
3.8. Normalización de valores 21
3.9. Algoritmo de backpropagation para redes neuronales. 22
3.10. Tensor de entrenamiento X. 23
3.11. Tensor de resultados Y. 23
3.12. Arquitectura LSTM. 24
3.13. Arquitectura GRU. 25

4.1. Disminución learning rate. 29
4.2. RMSE y MAE modelos LSTM. 31
4.3. Máximo error modelos LSTM. 31
4.4. R^2 modelos LSTM. 32
4.5. Predicciones LSTM mejores modelos 33
4.6. Predicciones LSTM peores modelos 34
4.7. Curvas de aprendizaje LSTM 36
4.8. Predicción próximo día 37
4.9. Métricas GRU 39

4.10. Predicciones GRU mejores modelos 41
4.11. Predicciones GRU peores modelos 42
4.12. Curvas de aprendizaje GRU 44

Índice de tablas

3.1. Características del Hardware. 17

4.1. Modelos generados. 28
4.2. Pruebas del optimizador. 30
4.3. Resultados de los modelos LSTM. 30
4.4. Hiperparámetros óptimos LSTM. 35
4.5. Resultados de los modelos GRU. 38
4.6. Mejores modelos LSTM y GRU. 43

B.1. Costes de personal. 48
B.2. Costes de recursos materiales. 48

Lista de Acrónimos

BTC: Bitcoin.

ETH: Ethereum.

GRU: Gated Recurrent Unit.

LSTM: Long Short Term Memory.

LTC: Litecoin.

ML: Machine Learning.

MAE: Mean Absolute Error.

ME: Máximo Error.

RNN: Recurrent Neural Network.

RMSE: Root Mean Square Error.

TFG: Trabajo de Fin de Grado.

TP: Tiempo de Predicción.

USD: U.S dollar.

Capítulo 1

Introducción

El objetivo principal del presente Trabajo de Fin de Grado (TFG) es el diseño e implementación de un sistema de predicción de las fluctuaciones en los precios del mercado de las criptomonedas. Este sistema está alimentado por un flujo de datos discreto en tiempo real. Para su implementación se usan dos tipos de algoritmos de Machine Learning: Gated Recurrent Units (GRU) y Long Short Term Memory Network (LSTM). El objetivo es realizar una predicción numérica en un conjunto de datos continuo. Se comparan los resultados obtenidos con cada sistema para analizar su eficacia y extraer conclusiones.

Hay que tener en cuenta que existen multitud de factores que influyen en el precio de cada criptomoneda. Para simplificar el problema se usan únicamente los valores históricos del precio para realizar los pronósticos. En consecuencia, las predicciones no tienen la máxima precisión, sino que son una aproximación de la realidad.

1.1. Motivación

El problema de predicción de una serie temporal ha sido ampliamente estudiado tanto por matemáticos como por economistas. Con el nacimiento de la Inteligencia Artificial y el Aprendizaje de Máquina, más comúnmente conocido por su nombre en inglés, Machine Learning (ML) se han actualizado estas técnicas, automatizando las predicciones de los mercados mediante diversos algoritmos. La volatilidad de las criptomonedas, así como su rápido crecimiento las convierte en un tema muy interesante para su estudio.

Además de la intrínseca motivación por la investigación de las redes neuronales y el estudio del software específico para el desarrollo de proyectos de Machine Learning, el hecho de poder realizar predicciones en un mercado con una volatilidad tan alta ofrece la capacidad de generar rentabilidad a corto plazo.

1.2. Objetivos

El objetivo principal de este TFG es desarrollar un modelo que sea capaz de predecir una serie temporal, en este caso, el precio de la criptomoneda. Se utilizan varios algoritmos de Machine Learning para ver cuál se ajusta mejor al problema y genera predicciones más precisas. Además, se entrega un marco teórico

lo suficientemente desarrollado como para poder entender el diseño y funcionamiento de los sistemas. Se definirán conceptos fundamentales del machine learning, así como las tecnologías necesarias para lograr el objetivo de este TFG.

1.3. Metodología

Para realizar el Trabajo de Fin de Grado se seguirán los siguientes pasos :

1. Realización de un marco teórico sobre el funcionamiento del mercado de la criptomoneda y de las redes neuronales.
2. Extracción, preprocesado y análisis de los datos obtenidos mediante una API.
3. Desarrollo software del modelo neuronal y selección de algoritmos a utilizar.
4. Pruebas y análisis del sistema.
5. Generación de conclusiones a partir de la interpretación de los resultados obtenidos.

1.4. Organización del TFG

En el Capítulo 2 se detalla el estado del arte tanto de las criptomonedas como del Machine Learning, profundizando en los algoritmos que se utilizan. Se finaliza el capítulo haciendo una revisión de trabajos relacionados con el tema.

En el Capítulo 3 se definen las tecnologías usadas en el proyecto y se explica el proceso de obtención y procesado de los datos. Así mismo, se muestra el diseño de la arquitectura de todos los sistemas y se definen los hiperparámetros utilizados en el momento de realizar las predicciones.

El Capítulo 4 está destinado a presentar los resultados logrados con cada algoritmo, se cierra el capítulo discutiendo dichos resultados.

En el último Capítulo se exponen las conclusiones obtenidas a lo largo del TFG. Para finalizar se analizan las líneas futuras del presente TFG.

Capítulo 2

Estado del Arte

2.1. Una nueva moneda y un nuevo mercado: historia de la criptomoneda

Desde el comienzo de la humanidad siempre ha existido el trueque, el intercambio de bienes y/o servicios ha sido clave para el desarrollo de la sociedad tal y como la conocemos. Posterior al trueque aparecieron las monedas de plata y oro para regular los intercambios, con lo que se introduce un elemento de referencia con el que comerciar más fácilmente. [1]

Actualmente tenemos un sistema fiduciario donde el valor del dinero descansa en la confianza de la sociedad sobre el instituto emisor, en la mayor parte de los casos un banco central. En el año 2009 con el fin de acabar con el monopolio de los estados sobre los mercados y sobre el dinero mundial, Satoshi Nakamoto crea el Bitcoin[2], una moneda electrónica descentralizada sin costes de transacción y con una alta velocidad de transferencia. Este es el nacimiento de las criptomonedas, un tipo de divisa digital que emplea un encriptamiento fuerte y tiene un control descentralizado. Para que un sistema pueda ser considerado una criptodivisa tiene que cumplir los siguientes requisitos [3].

- El sistema no necesita una autoridad central. Así, su estado es mantenido a través de un consenso distribuido.
- El sistema mantiene control sobre todas las unidades y los propietarios de estas.
- El sistema determina si se pueden crear nuevas unidades. En caso afirmativo, el sistema debe definir las circunstancias de su origen y cómo determinar el propietario de las nuevas unidades.
- Sólo se puede asegurar la propiedad de una unidad a un usuario de manera criptográfica.
- El sistema permite las transacciones de unidades, en las cuales se cambia el propietario de estas. Una transacción solo puede ser efectuada si se puede probar el actual propietario de estas unidades.

Con la creación de esta nueva forma de pago se genera un nuevo mercado: el mercado de las criptomonedas. Actualmente existen más de 4000 criptomonedas,

aunque el 90 % del mercado está dominado por sólo 20 criptodivisas [4]. La capitalización de mercado en el año 2021 alcanzó los 2.5 billones de dolares (Ver Figura 2.1), de los cuales los mayores porcentajes corresponden a Bitcoin (BTC) y Ethereum (ETH), 44 % y 18 % respectivamente, como se puede apreciar en la Figura 2.2.

Figura 2.1: Capitalización total del mercado de la criptomoneda.[5]

2.2. Blockchain, la base de las criptomonedas

Las criptomonedas están basadas en un sistema peer-to-peer. Para poder comprobar que las transacciones solo se realizan una vez se usa el sistema Blockchain.[7] Blockchain es un sistema distribuido, donde todos los nodos están conectados sin depender de ninguno central. Se basa en un conjunto de bloques unidos en orden (Figura 2.3), donde cada bloque almacena una cadena de las transacciones realizadas, información del propio bloque y dos hash: uno del bloque anterior y otro del propio bloque para relacionarlos.

Para añadir un nuevo bloque se usa una proof-of-work, para lo que se resuelven algoritmos de alta complejidad que requieren un alto uso de CPU, una vez añadido el bloque es imposible modificar la cadena, de esta manera se asegura que cada moneda se pueda gastar solamente una vez [2].

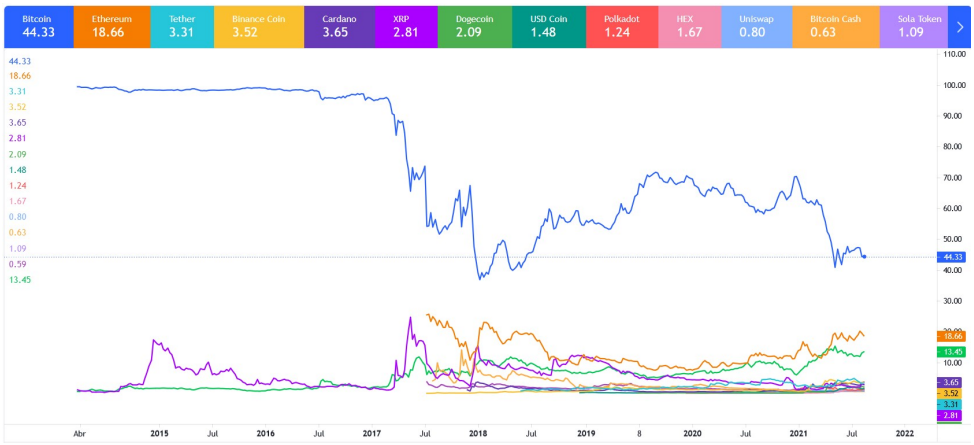


Figura 2.2: Dominio de capitalización total de mercado, % [6]

.

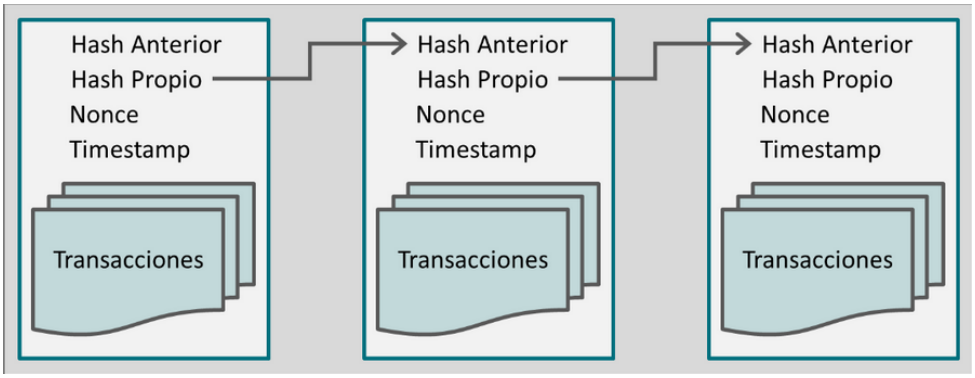


Figura 2.3: Representación blockchain.[8]

2.3. Mercado de la criptomoneda

El mercado de la criptomoneda funciona únicamente mediante la oferta y la demanda. Es extremadamente volátil debido a múltiples factores, como la falta de regulación del mercado o la gran cantidad de especulación [9]. Es por esto que hay que ser muy cuidadoso al invertir en este mercado. Los precios de la criptodivisa son un proceso estocástico no estacionario y no lineal. Se ha tratado de predecir con modelos de series lineales clásicos como ARIMA(modelo autorregresivo integrado de promedio móvil), ARMA (modelo autorregresivo de promedio móvil) o α-Sutte indicator (Ver apartado 2.13 sobre trabajos relacionados). Estos modelos al estar diseñados para series temporales lineales no funcionan de manera óptima con el mercado de la criptomoneda, por eso emplearemos algoritmos de Machine Learning.

2.4. Machine Learning

El Machine Learning (ML) es una aplicación de la Inteligencia artificial. En el ML se entrena un sistema para procesar y hacer uso de una gran cantidad de datos, esto es especialmente útil actualmente, donde el Big Data está floreciendo gracias al uso de

nuevas tecnologías, que nos proporcionan una mayor capacidad de almacenamiento y procesamiento. Existen múltiples usos de esta tecnología, desde el reconocimiento de imágenes y sonidos hasta coches autónomos, pasando por la realización automática de diagnósticos médicos.

Hay 3 paradigmas básicos en el ML (Ver Figura 2.4): Aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo. Todos ellos tienen el mismo objetivo, generar una función de la forma $f : X \rightarrow Y$. Es decir, que al introducir unos datos (X) a la entrada se devuelva una predicción de datos (Y) a la salida. Independientemente del paradigma usado y del problema que se quiere solucionar siempre se siguen una serie de pasos principales:

1. Definición del problema.
2. Recopilación de datos.
3. Procesado de los datos.
4. Elección del modelo.
5. Entrenamiento del modelo.
6. Evaluación del modelo.

2.4.1. Aprendizaje Supervisado

En este paradigma se divide el conjunto de datos en dos subconjuntos, el set de entrenamiento y el set de testeo. Primero se entrena el modelo utilizando unos datos etiquetados. Que estén etiquetados significa que para cada característica de entrada X, conocemos el valor de su atributo objetivo Y. Una vez que termina el proceso de entrenamiento se usan los datos de testeo para predecir el atributo objetivo y contrastar con los valores reales, y así hallar la precisión de nuestro algoritmo. El aprendizaje supervisado se usa en problemas de predicción de valores continuos, llamados problemas de regresión y en problemas de predicción de valores discretos, llamados problemas de clasificación. En este TFG se usará aprendizaje supervisado dado que se plantea un problema de regresión. El objetivo es estimar la variable x en un tiempo futuro: $\hat{x}[t+s] = f(x[t], x[t-1], ..., x[t-n], s > 0$, donde s es el horizonte temporal a predecir.

2.4.2. Aprendizaje No Supervisado

Cuando no tenemos una característica objetivo y solo conocemos los datos de entrada usamos el aprendizaje no supervisado. En este tipo de aprendizaje describimos la estructura de los datos para tratar de obtener un ordenamiento de estos. Se usa en problemas de segmentación y clustering.

2.4.3. Aprendizaje por Refuerzo

En el aprendizaje por refuerzo se usa un sistema de retroalimentación, donde la máquina aprende por ensayo y error. Se intenta alcanzar un comportamiento óptimo

para la resolución del problema, aprendiendo de iteraciones pasadas y maximizando la recompensa. [10]

Figura 2.4: Tipos de aprendizaje.

2.5. Algoritmos para la Regresión

Para predecir el valor de la criptomoneda, típicamente se usa la regresión. Existen múltiples algoritmos para la realización de esta tarea y el objetivo de este TFG es encontrar el más fiable y rápido. A continuación, se presenta una lista de los más importantes:

- Regresión Lineal
- Árbol de Decisión
- Algoritmos Bayesianos
- Support Vector Machine (SVM)
- Random Forest
- Redes Neuronales
  - Red Neuronal Recurrente (RNN)
    - Long Short Term Memory (LSTM)
    - Gated Recurret Units (GRU)
  - Red Neuronal Convolucional (CNN)
  - Red Neuronal Recurrente Convolucional (CTRNN)

Entraremos en detalle en los algoritmos que se usan en el presente TFG: Redes neuronales recurrentes, en específico LSTM y GRU.

2.6. Redes Neuronales

Las Redes Neuronales Artificiales (ANN) están inspiradas por las redes neuronales biologicas. Una neurona biológica típica consta de 3 partes principales: Axón, dendrita y cuerpo celular. Las dendritas permiten establecer contacto con otras neuronas mediante sinapsis. Esta genera una descarga química que provoca una

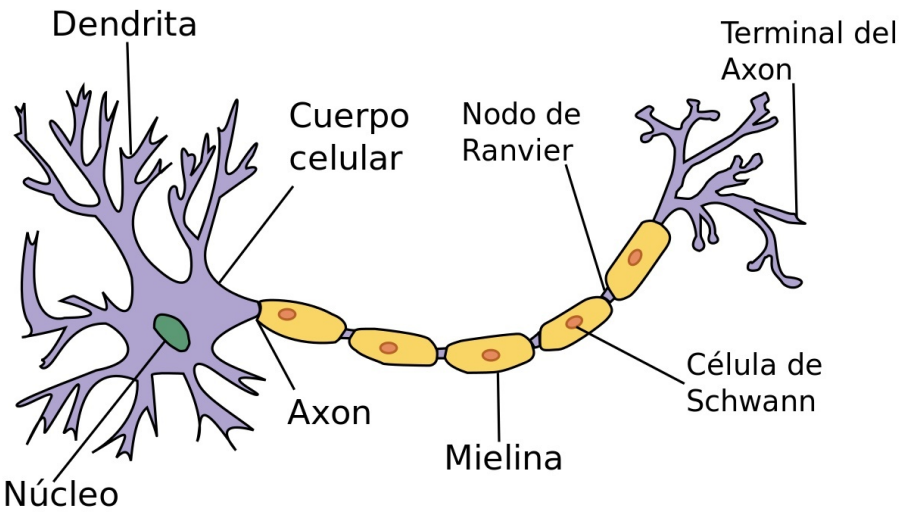


Figura 2.5: Neurona. [11]

corriente eléctrica. Cuando un impulso llega al extremo del axón la neurona segrega neurotransmisores que inhiben o excitan a la neurona.

El equivalente de las neuronas biológicas en las ANN es el perceptrón, a la entrada tenemos un valor x_n que es multiplicado por su peso W_n, estos valores se suman y se pasan por la función de activación que devuelve un resultado binario. Se puede representar matemáticamente su comportamiento mediante la Ecuación 2.1:

$$y = \sum_{i=1}^{n} f(W_n x_n + b) \tag{2.1}$$

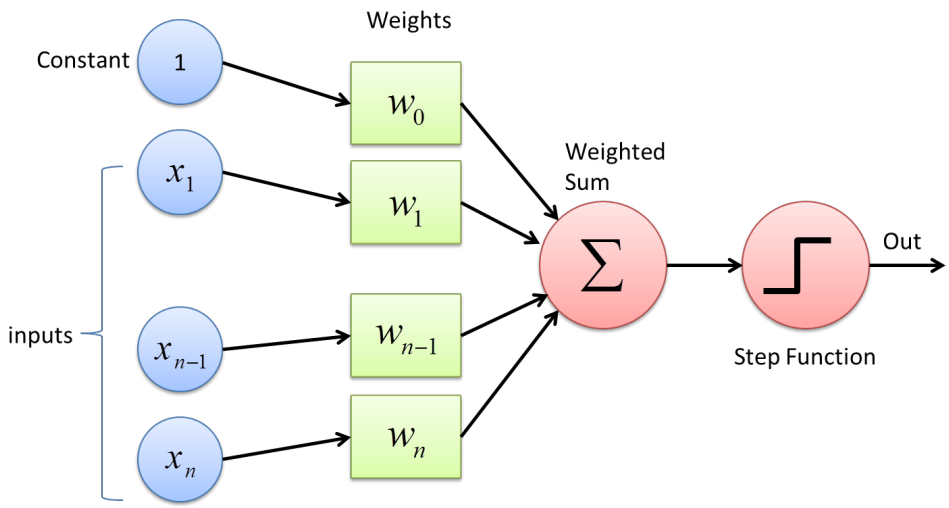


Figura 2.6: Perceptrón. [12]

Existen diversas funciones de activación dependiendo del resultado deseado, las más utilizadas son la función escalón, la función sigmoide y la función lineal.

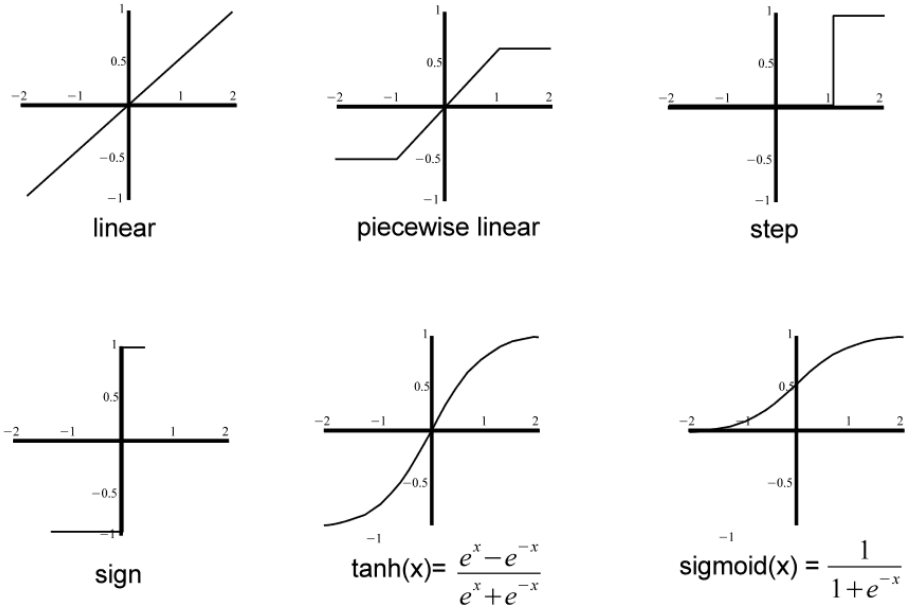


Figura 2.7: Funciones de activación. [13]

Una combinación de neuronas artificiales crea una red neuronal. Dependiendo de la configuración de la red esta puede ser multicapa, modular o de realimentación. La que se usará en este TFG será una red multicapa con realimentación, que consta de 3 tipos de capas interconectadas, la capa de entrada, la capa oculta y la capa de salida (ver Figura 2.8). La información se propaga a través de las neuronas de entrada hasta la capa de salida. La primera capa es pasiva, tiene tantos nodos como variables existan y solamente transmite la información hasta las capas ocultas, estas tienen funciones de activación no lineales que permiten que la red aprenda a resolver problemas más complejos, pero su proceso de aprendizaje es más complicado.[13] La red aprende mediante la actualización de un peso w_{ij} entre una neurona j obteniendo datos x_i de una neurona i y produciendo una salida o_j, este peso representa la frecuencia de interacción entre las dos neuronas para producir el resultado de la red. La actualización del peso Δw_{ij} viene dada en la Ecuación 2.2:

$$\Delta w_{ij} = F(x_i, o_j, w_{ij}, \eta) \tag{2.2}$$

donde η es el ratio de aprendizaje y F es una función multiplicativa.

2.7. Redes Neuronales Recurrentes

En 1982 John Hopfield creo la primera neurona recurrente, la red de Hopfield. [15] Las redes neuronales recurrentes permiten estudiar secuencias de datos temporales de un tamaño incierto y predecir el próximo valor. Una neurona recurrente puede

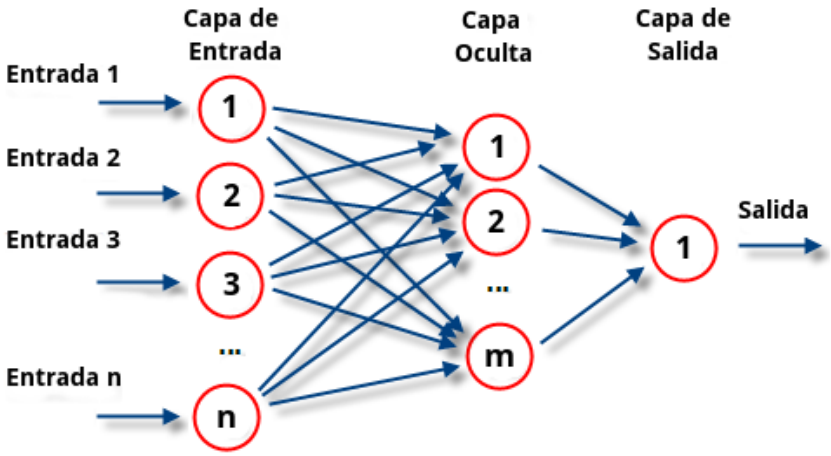


Figura 2.8: Red neuronal. [14]

recordar valores anteriores, es decir, posee memoria. En cada instante de tiempo la neurona recibe dos entradas, X que es la entrada de la capa e Y que es la salida del instante anterior. En la Figura 2.9 se representa una red neuronal recurrente, que es una combinación de neuronas recurrentes.

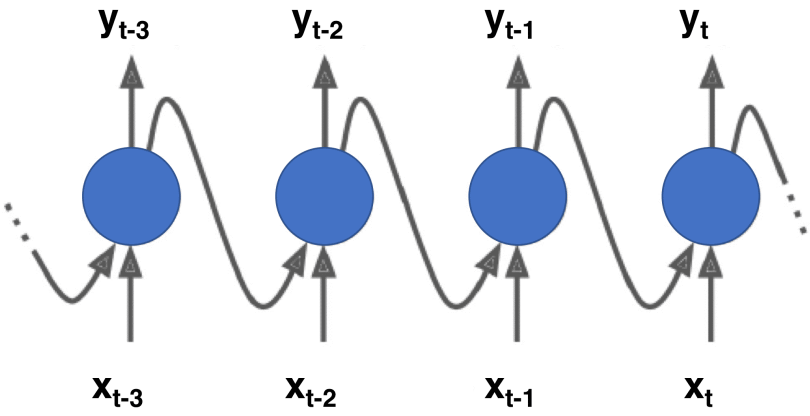


Figura 2.9: Red neuronal recurrente.[16]

2.8. Long Short Term Memory

En las redes neuronales recurrentes existe una limitación importante, el problema de desvanecimiento del gradiente. Esto provoca que las redes no tengan memoria a largo plazo. Los gradientes son valores usados para actualizar los pesos de las neuronas de la red, el problema ocurre cuando el gradiente se desvanece mientras se propaga, impidiendo el cambio de los pesos de la red al ser demasiado pequeño para aportar. Para solventar este problema se crean las LSTM, redes que amplían su memoria para poder recordar las entradas por un período más largo. En función de la importancia

de la entrada se almacenará o no la información. En una neurona LSTM existen 3 puertas (ver Figura 2.10), la puerta de entrada, la puerta de olvido y la puerta de salida [17].

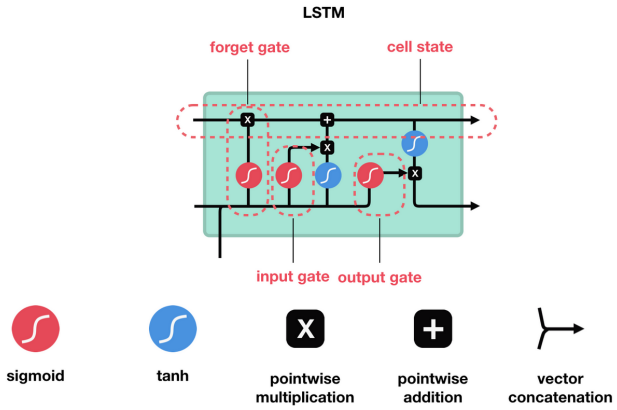


Figura 2.10: Neurona LSTM.[17]

En la puerta de olvido entra la suma del estado anterior y la información de la entrada actual, el resultado de la suma se multiplica por una sigmoide y el valor de salida está entre 0 y 1. Cerca de 0 se olvida y cerca de 1 se mantiene. La puerta de entrada decide qué valores serán actualizados en el estado de la célula. La puerta de salida entrega el próximo estado oculto.

2.9. Gated Recurrent Units

Una neurona GRU (ver Figura **??**es muy similar a una neurona LSTM. En una neurona GRU existen dos puertas, la de actualizar y la de reset [17]. La puerta de actualizar decide la información que se conserva y la información que se olvida. La puerta de reseteo discrimina cuanta información pasa al siguiente estado oculto.

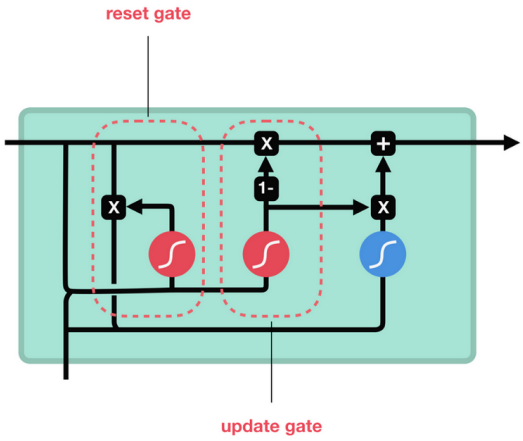


Figura 2.11: Neurona GRU[17].

2.10. Árboles de Decisión

Un árbol de decisión es un tipo de algoritmo parecido a un diagrama de flujo (ver Figura 2.12), posee un nodo raíz, nodos de decisión y nodos hoja. Cada nodo de decisión representa un atributo de los datos y ofrece varias ramas, llamadas reglas de decisión, a través de estas ramas se conectan los nodos hoja, los cuáles corresponden a una etiqueta de la predicción [18].

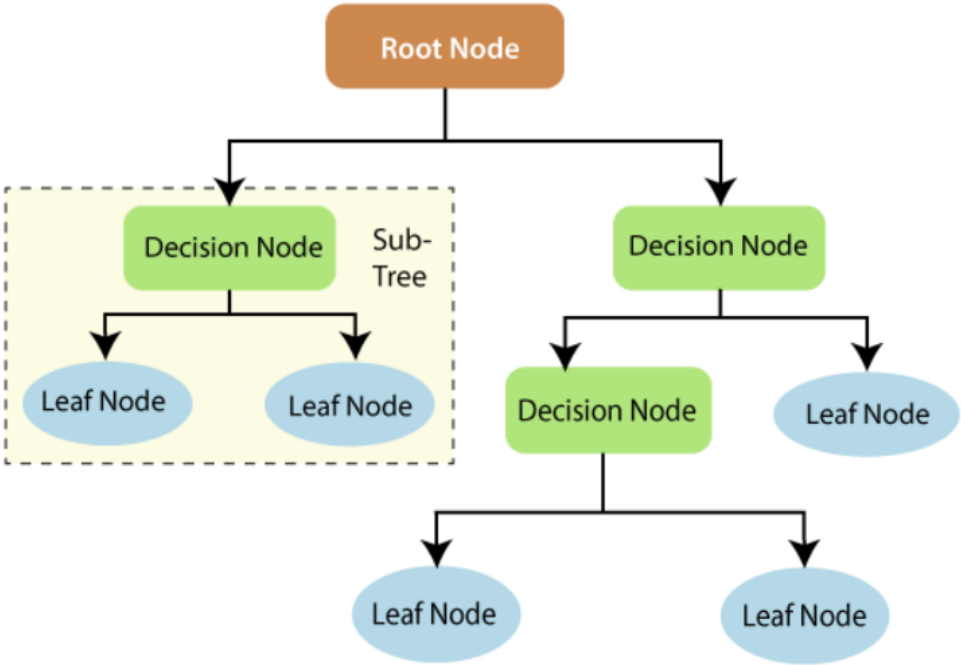


Figura 2.12: Árbol de decisión [18].

2.11. Random Forest

Random Forest está formado por árboles de decisión incorrelados que operan como un conjunto, cada árbol predice un resultado y la media de los resultados individuales se convierte en la predicción del modelo[19].

2.12. Evaluación de resultados

Para la evaluación de los resultados obtenidos, expuestos en el Capítulo 4, se usarán una serie de métricas destinadas a determinar la precisión del modelo, dichas métricas son las siguientes.

- Error Cuadrático Medio (MSE):

  Función de pérdida que mide el promedio de los errores al cuadrado. Se calcula

como la diferencia entre el estimador y lo que se estima. Véase Ecuación 2.3:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2 \tag{2.3}$$

dónde $\hat{Y}_i$ es un vector de n predicciones e Y_i es un vector de n con los valores reales.

- Raíz del Error Cuadrático Medio (RMSE):

  Es la raíz del MSE, se usa para informar del desempeño del sistema. Es útil ya que devuelve las mismas unidades objetivo. Por ejemplo, si la unidad original son dólares MSE devuelve dolares al cuadrado y RMSE devuelve dolares. Está expresando en la Ecuación 2.4.

$$RMSE = \sqrt{MSE} \tag{2.4}$$

- Error Absoluto Medio (MAE):

  Promedio de diferencias absolutas entre los valores reales y las predicciones. Al contrario que MSE y RMSE es una puntuación lineal, es decir, todos los errores se ponderan igual. Matemáticamente se ve en la Ecuación 2.5.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\|Y_i - \hat{Y}_i\| \tag{2.5}$$

- Coeficiente de determinación(R^2):

  Este coeficiente está libre de escala, devuelve valores entre $-\infty$ y 1. Cuando sea negativo significa que el modelo es peor que calcular la media. Se define en la Ecuación 2.6.

$$R^2 = 1 - \frac{MSE(modelo)}{MSE(base)} \tag{2.6}$$

  dónde MSE (modelo) es el que hemos definido en la Ecuación 2.3 y MSE (base) es el modelo más simple posible, que sería predecir siempre el promedio de todas las muestras [20].

- Máximo error (ME):

  El máximo error es el peor caso de nuestra predicción, es decir, cuando la diferencia entre el valor predicho y el valor real es máxima.

$$ME = max|valor\ predicho\ \text{-}\ valor\ real| \tag{2.7}$$

- Normalized Root Mean Square Error (NRMSE): El NRMSE es el valor normalizado del RMSE. Esta métrica expresa en porcentaje la diferencia entre los valores predichos por el modelo y los valores reales, y nos da una idea de la diferencia entre el valor máximo y el valor mínimo predicho.

$$NRMSE = \frac{RMSE}{precio\ máximo - precio\ mínimo} * 100 \tag{2.8}$$

2.13. Trabajos Relacionados

Existe una amplia bibliografía relacionada con la predicción de los movimientos del mercado de la criptomoneda. Así mismo, existen múltiples formas de hallar estas predicciones. Se usan variables como la cantidad de twits publicados sobre el BTC y el precio de éste. En [21] se prueba la correlación entre la cantidad de veces que el BTC es mencionado en Twitter y su subida de precio al día siguiente. Tradicionalmente, los mercados han sido estudiados mediante técnicas estadísticas lineales clásicas, como ARIMA(modelo autorregresivo integrado de promedio móvil), un modelo que utiliza variaciones y regresiones de datos con la finalidad de encontrar patrones para la predicción de valores. Sin embargo, en [22] se comprueba que el Machine Learning tiene un mejor rendimiento que las técnicas estadísticas lineales. En este estudio se usa RNN y LSTM, con los cuáles se alcanzó una precisión del 52 % con un RMSE del 8 %. En [23] se obtuvo mediante Redes Neuronales, Support Vector Machines y LSTM una precisión del 65 % para predicciones en el siguiente día y entre el 62 % y el 64 % para una predicción entre el 6º y el 90º día . En [24] se concluye que α-Sutte Indicator, un modelo lineal estadístico más reciente, tiene una mejor precisión y un menor valor de MSE Y RMSE que ARIMA y NNAR. Sin embargo siguen siendo más efectivos los metodos de Machine Learning. En [25] se prueba que debido a que el mercado de la criptomoneda es no lineal y contiene patrones ocultos, funcionan mejor las técnicas de aprendizaje profundo, como las redes neuronales recurrentes o las redes convolucionales.

Capítulo 3

Diseño del Sistema

Este Capítulo describe el proceso de diseño realizado para crear los sistemas de predicción del precio de la criptomoneda. En la Sección 3.1 se especifica el software utilizado, en la Sección 3.3 se muestran los datos recopilados antes y después de su procesamiento. En la Sección 3.4 se detalla cómo se ha realizado el preprocesado de los datos. Por último en la Sección 3.5 se detallan las arquitecturas de los modelos de ML.

El diseño global de los sistemas es el mismo para todos. Consiste en el preprocesado de los datos para que puedan ser introducidos en el modelo que corresponda. Posteriormente se realiza una validación para ver la efectividad de cada sistema. Cuando se comprueba que el sistema funciona correctamente se procede a la predicción de valores futuros. Durante todas las etapas de desarrollo se visualizan los resultados obtenidos para una mejor comprensión de estos. Una representación de como funcionan los sistemas se muestra en la Figura 3.1

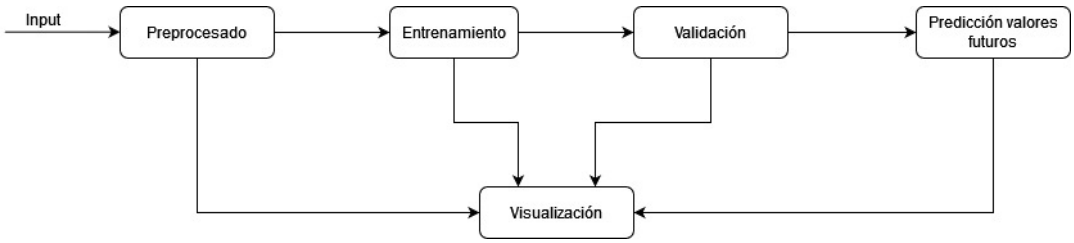


Figura 3.1: Diseño de los sistemas.

3.1. Tecnologías utilizadas

Existen dos tecnologías esenciales para el desarrollo de los sistemas y que son usadas en mayor o menor medida a lo largo de todo el Trabajo de Fin de Grado. Estas tecnologías son Python y Google Colab.

3.1.1. Python

Python es un lenguaje de programación de tipo interpretado y multiparadigma. Es de tipado dinámico y soporta orientación a objetos [26]. En la Figura 3.2 se muestran las características más importantes de este lenguaje.

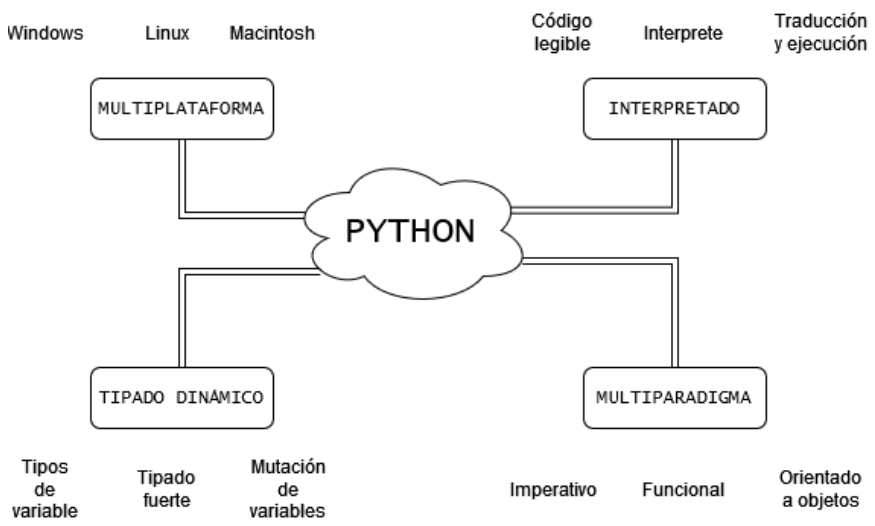


Figura 3.2: Características de Python.

Es el lenguaje dominante en el campo del Machine Learning, esto se debe a su gran cantidad de bibliotecas en código abierto, que le otorgan una gran versatilidad en todas las tareas necesarias al desarrollar un proyecto de ML: extracción y procesamiento de datos, aplicación de algoritmos y evaluación de resultados. Además, es un lenguaje que debido a su sintaxis sencilla, facilita su aprendizaje y la creación de código legible. Las bibliotecas utilizadas en el presente TFG son las siguientes:

- Numpy: Permite realizar operaciones matematicas con arrays y matrices de múltiples dimensiones.
- Pandas: Se usa para la manipulación de datos y el análisis de estos. Permite leer ficheros y convertirlos en una estructura llamada DataFrame.
- PyTorch: Libreria de Machine Learning, con ella se crean y entrenan los modelos. Permite trabajar mediante tensores y con una aceleración a través de GPU. Las redes neuronales profundas están construidas sobre un sistema de diferenciación automática.
- Matplotlib: Dibuja y grafica, es una extensión de Numpy.

Es importante señalar que con PyTorch se pueden crear redes neuronales desde cero usando tensores, pero también se pueden crear usando la clase torch.nn, que proporciona funciones y clases ya definidas para hacer redes neuronales más flexibles y fáciles de leer. En este TFG usaremos esta segunda forma de generación de redes neuronales profundas.

3.1.2. Google Colab

Google Colab es un servicio cloud gratuito, basado en los cuadernos de Jupyter y con múltiples librerías ya instaladas en el entorno, por lo que no requiere configuración. Utiliza un kernel con Python 3, los cuadernos se guardan en Google Drive y se pueden exportar a GitHub. El código se ejecuta en una máquina virtual asignada a la cuenta de Google automáticamente. Los recursos asignados a cada maquina virtual son variables dependiendo de múltiples factores. Debido a la fluctuación de los límites en dichos recursos Google no pública los detalles del entorno. Sin embargo, se puede usar una GPU para acelerar la maquina virtual.

3.2. Hardware

Para acceder a Google Colab y realizar este Trabajo de Fin de Grado se utiliza un ordenador portátil. Las características se especifican en la Tabla 3.1.

Procesador	**RAM**	**Tipo de sistema**
Intel(R)Core(TM) i5-6200U CPU @ 2.30GHz 2.40 GHz	8,00 GB	Windows 10 Home

Tabla 3.1: Características del Hardware.

3.3. Recopilación de Datos

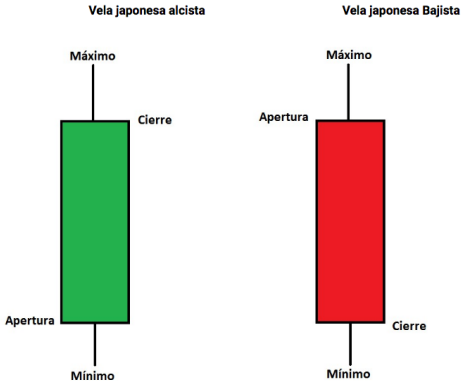
Los datos recopilados son extremadamente importantes, ya que constituyen la base de todo el sistema. Se utiliza una API (Application Programming Interface) [27] para recolectar los datos de la criptomoneda desde su creación hasta el 06/01/2022. La API utilizada es *Binance*, una API pública y gratuita que entrega información de múltiples criptomonedas. En concreto, usaremos los datos de BTC (Bitcoin). Las características que ofrece son las típicas de un gráfico de velas (ver Figura 3.3) con los siguientes datos : open price, close price, high price, low price, volume y otros valores que serán explicados posteriormente en este capítulo. Los datos se obtienen mediante una petición al servidor[1] donde se incluye toda la información necesaria: tipo de moneda, tiempo de inicio, tiempo de finalización e intervalo entre los puntos obtenidos. La respuesta a esta petición es un archivo JSON con los datos pedidos.

3.4. Preprocesado de los Datos

Una vez recolectados todos los datos hay que preprocesarlos para introducirlos en el algoritmo. Primero se formatea la respuesta en un dataframe pandas con todos los datos obtenidos. Si imprimimos la cabecera del dataframe obtenemos los datos

[1] `https://api.binance.com/api/v3/klines?symbol=BTCUSDT&interval=6h&startTime=1502942400000&limit=2000`

(a) Ejemplo de gráfico de velas.



(b) Significado de las velas

Figura 3.3: Gráficos de velas[28]

de la Figura 3.4. Se aprecia que open time está expresado en milisegundos, por lo que tenemos que pasarlo a formato estándar dd/mm/yyyy. La información mostrada en este ejemplo corresponde al mercado BTC/USD, esto es, el precio del Bitcoin al Dolar Estadounidense.

	open_time	open	high	low	close	volume	close_time	quote	trades	takers_buy_base	takers_buy_quote	ignore
0	1.502950e+12	4330.29	4485.39	4287.41	4427.30	75.292822	1.502971e+12	3.330245e+05	309.0	66.192786	2.931407e+05	8020.698096
1	1.502971e+12	4436.06	4485.39	4235.45	4289.24	301.685704	1.502993e+12	1.321749e+06	1448.0	203.178404	8.921018e+05	8243.913773
2	1.502993e+12	4289.24	4369.69	4200.74	4285.08	347.755926	1.503014e+12	1.497326e+06	1397.0	290.268777	1.249413e+06	8733.911395
3	1.503014e+12	4285.08	4340.62	4134.61	4313.56	391.159844	1.503036e+12	1.667910e+06	1670.0	306.370008	1.308099e+06	9073.698885
4	1.503036e+12	4313.56	4371.52	4247.75	4340.31	329.562357	1.503058e+12	1.417446e+06	1381.0	279.037993	1.200049e+06	9366.914508

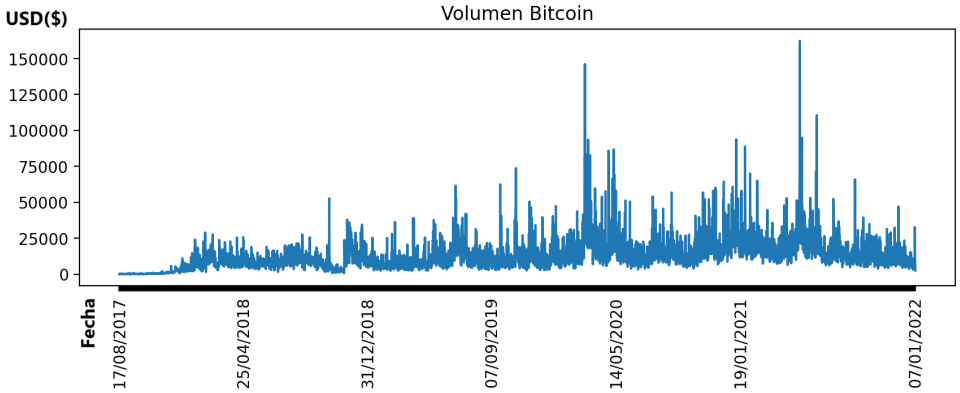
Figura 3.4: Datos obtenidos mediante API Binance.

El preprocesado consiste en depurar los datos mediante una serie de técnicas:

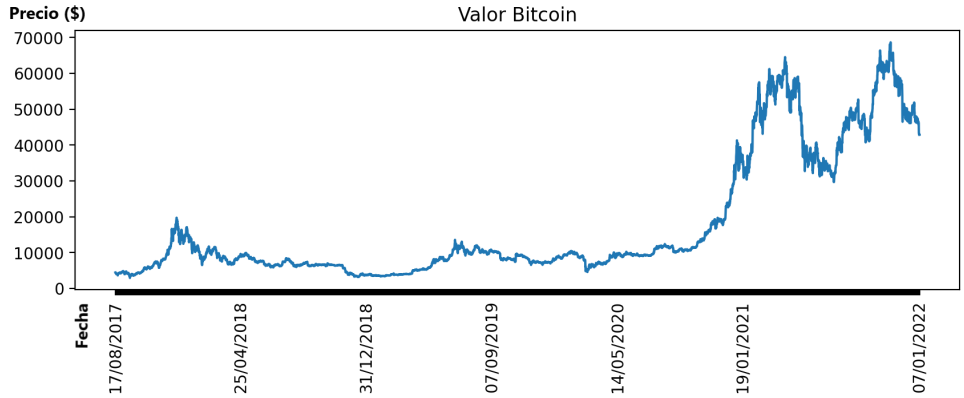
- Limpieza: Detección de datos nulos o anómalos para corregirlos. En nuestro caso no tenemos ningún error en los datos por lo que no hace falta realizar una limpieza.
- Transformación: Transformar el dataset en datos equivalentes, en nuestro caso se normalizan los datos con una media de 0 y una desviación estándar de 1.
- Reducción: Se seleccionan los atributos más útiles para la predicción, para lo cual calculamos la correlación entre ellos.

3.4.1. Correlación de los Datos

Se eligen los datos más relevantes, es decir, los que entreguen más información para el entrenamiento del algoritmo. Para determinar que datos son más importantes se realiza una correlación de todos los datos. En la Figura 3.5 se visualizan todos los datos: *open price*,*close price* , *low price*, *high price*, *volume*, *taker buy base*, *quote* (volumen del mercado en dólares). Se distingue a simple vista que *open price*, *close*



(a) Volumen BTC en USD



(b) Datos BTC

Figura 3.5: Todos los datos del BTC

price, *low price* y *high price* son extremadamente similares. Generamos una matriz de correlaciones (ver Figura 3.6), los colores claros implican una mayor correlación y los tonos oscuros no tienen correlación. En esta primera imagen se representan todas las características. Incluimos una segunda imagen donde se aprecia la fila de close en detalle (ver Figura 3.7). Se puede apreciar que *open*, *high*, *low* y *close* están altamente correlados, lo que significa que tienen información muy parecida, por lo que únicamente es necesario introducir una de las características en el algoritmo. Se ha elegido una de las características aleatoriamente, ya que aportan la misma información. Se decide emplear el valor de *close*. A su vez *volume* y *takers buy base* están correlados entre ellos pero incorrelados con el resto, por lo que se ha decidido seleccionar uno de ellos, de nuevo al aportar la misma información da igual que característica se utilice, se selecciona *volume*. El resto de características son ignoradas pues no aportan ninguna información adicional. Después de hacer las pruebas expuestas en el Capítulo 4 se observa que el rendimiento al introducir las dos características en el algoritmo es muy similar a cuando solo se introduce el precio de cierre, sin embargo el entrenamiento con dos características requiere mucha más potencia y tiempo. Por lo cual se decide entrenar los modelos con únicamente el precio de cierre.

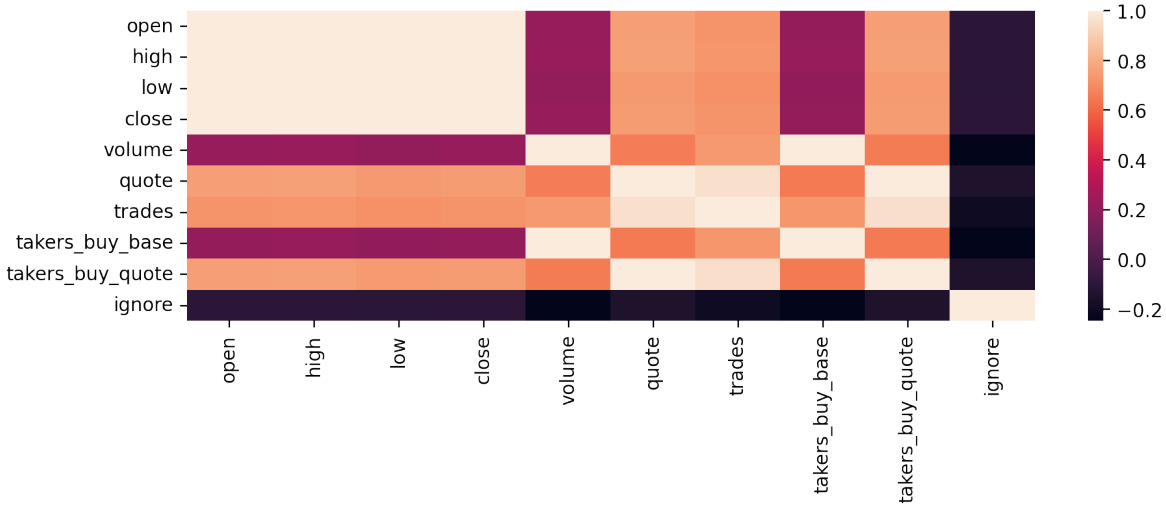


Figura 3.6: Matriz de correlación.

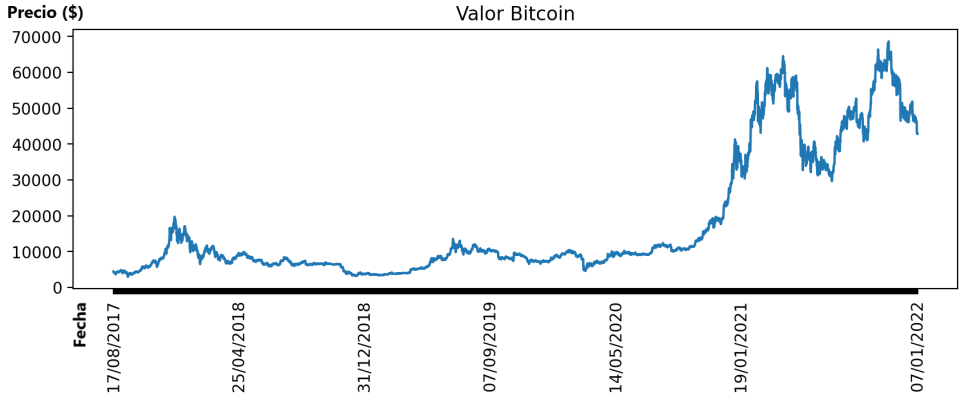
Figura 3.7: Correlación de close.

3.4.2. Transformación de los Datos

Debido a que el sistema LSTM usa en sus puertas la función sigmoide y la función hiperbólica tangente (*tanh*) es necesario normalizar los datos para optimizar el algoritmo. La razón de esta normalización es que la salida de una función sigmoide está comprendida entre 0 y 1. Lo mismo pasa con la función tanh con la diferencia que la salida oscila entre -1 y 1. Para normalizar los datos se hace un escalado con la media y la desviación típica, como se puede observar en la Ecuación 3.1:

$$z_i = \frac{x - \mu}{\sigma} \tag{3.1}$$

donde z_i es el valor normalizado con $i \in \{1, ..., N\}$ siendo N el numero de muestras, μ es la media y σ es la desviación típica. Esta ecuación devuelve el valor normalizado con una media de 0 y una desviación típica de 1. Además, la normalización de los datos permite que la red neuronal realice menos cálculos y por lo tanto sea más rápida y eficiente. Los datos conservan la misma forma, pero en vez de encontrarse entre 0 y 68 633 dolares oscilan en el intervalo [-0.8415 , 2.953], como se puede ver en la Figura 3.8



(a) BTC no normalizado

(b) BTC normalizado

Figura 3.8: Normalización de valores

3.4.3. Algoritmo de Backpropagation

El objetivo de los sistemas aquí expuestos es crear una función que relacione las variables de entrada X (precios históricos del BTC) con las variables de salida Y (futuros precios del BTC), esto se consigue minimizando mediante múltiples iteraciones la diferencia entre los valores estimados y los valores reales, esta diferencia es llamada función de coste. Para encontrar los valores mínimos de esta función hay que encontrar los pesos adecuados de las aristas que unen las neuronas de la red. El descubrimiento de estos pesos se logra gracias al algoritmo de **backpropagation** (ver Figura 3.9). De está forma se pueden predecir valores futuros.

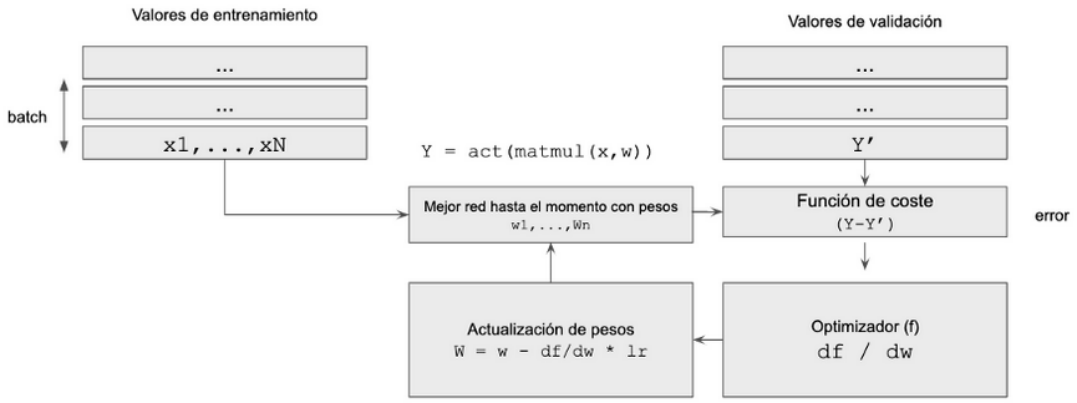


Figura 3.9: Algoritmo de backpropagation para redes neuronales.[29]

3.4.4. División del Dataset

Para generar los valores de entrenamiento y los de validación se realiza un enventanado. A continuación se detalla el proceso. El proceso de enventanado prepara el dataset (conjunto de datos) para su entrada al sistema, y genera dos arrays. Primero genera un array de dos dimensiones: X (Figura 3.10). La primera dimensión corresponde con el tamaño de la ventana, y la segunda dimensión es el tamaño del dataset menos el número de puntos del enventanado. Este array contiene los datos de entrenamiento. El segundo array Y (Figura 3.11) tiene una dimensión y contiene los datos de validación. Las predicciones se realizarán cada 1 día, 5 días, 10 días y 20 días. Esta variable se denota como TP (Tiempo de Predicción). El número de puntos del enventanado y la frecuencia de muestreo son los dos factores clave para seleccionar el TP (véase Ecuación 3.2), por ejemplo para un tiempo de predicción de 5 días con un periodo de muestreo de 6 horas se necesita un enventanado de 20 puntos (variable X) y se predice el punto 21 (variable Y), nótese que el tiempo de predicción es un múltiplo del periodo de muestreo.

$$TP = \frac{\text{Periodo de muestreo} * \text{N}^{\text{o}}\text{puntos}}{24\text{horas}} \longrightarrow 5 \text{ días} = \frac{6 * 20}{24} \tag{3.2}$$

En el preprocesado el dataset se divide en un 80 % para los datos de entrenamiento y un 20 % para los de validación. Esta separación se hace de manera no aleatoria para poder visualizar los datos fácilmente. Como resultado final obtenemos 4 arrays:

- $Xtrain$: Contiene los datos de entrenamiento.
- $Ytrain$: Contiene los resultados esperados después del entrenamiento.
- $Xtest$: Contiene los datos de validación.
- $Ytest$: Contiene los resultados de Xtest.

Sin embargo, al cargar el dataset en el algoritmo, usamos la clase DataLoader integrada en PyTorch, gracias a esta clase sí que añadimos una randomización de los datos al usar Shuffle = True. Esta división aleatoria nos asegura una máxima variabilidad en los datos. Además la clase Dataloader divide los datos en lotes de

$$\begin{matrix} p_1 & p_2 & \dots & p_{19} & p_{20} \\ p_2 & p_3 & \dots & p_{20} & p_{21} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{n-21} & p_{n-20} & \dots & p_{n-3} & p_{n-2} \end{matrix}$$

Figura 3.10: Tensor de entrenamiento X.

$$\begin{matrix} p_{21} \\ p_{22} \\ \vdots \\ p_n \end{matrix}$$

Figura 3.11: Tensor de resultados Y.

datos de tamaño 128 llamados batchs. Estos lotes se introducen en cada iteración del entrenamiento llamado epoch. Una definición más exhaustiva de batch y epoch será presentada en la Sección 3.6.

El número de datos está determinado por el tiempo de intervalo entre puntos, por ejemplo, si el día 30/11/2021 a las 15:30 horas establecemos un tiempo de intervalo igual a 6 horas obtendremos un dataset de 6 260 puntos x 10 columnas. Después de la reducción de datos quedan 2 columnas (precio de cierre y volumen) y el mismo número de puntos. Una vez terminado el preprocesamiento de los datos ya se pueden introducir en los sistemas de ML. Un alto número de datos provoca un problema de overfitting, es decir, se produce un error debido a que el modelo tiene demasiada información por lo que no aprende correctamente. Este error se puede detectar porque el error de entrenamiento es muy bajo, pero al hacer la predicción y en la etapa de validación el error es alto.

3.5. Arquitectura de las Redes Neuronales

En esta sección se muestra la arquitectura diseñada para cada sistema de predicción. Todas las redes neuronales tienen la misma estructura general, una capa de entrada donde se introducen los datos, una o varias capas ocultas que calculan la función que genera la predicción y una capa de salida que recoge todas las salidas de las capas ocultas y entrega la salida final, la predicción buscada.

3.5.1. Long Short Term Memory

Este modelo está compuesto por una capa de entrada lineal que mapea las entradas hacia un espacio de características de alta dimensión y un número de capas LSTM a definir por medio de un proceso de sintonización. En este TFG se analizarán los casos con 1, 2 y 3 capas. El número de neuronas de la capa de entrada es igual a la dimensión del tensor que se introduce en la red, o lo que es lo mismo, el número de características que entran al sistema. En el caso de las capas LSTM, se tiene que definir el número de neuronas buscando la configuración óptima. Se probará con 16, 32, 64 y 128 neuronas. En total tenemos 16 combinaciones posibles para el diseño de las capas ocultas. La capa de salida recibe la salida de las capas ocultas y devuelve un tensor como el mostrado en la Figura 3.11 . La arquitectura está representada en la Figura 3.12.

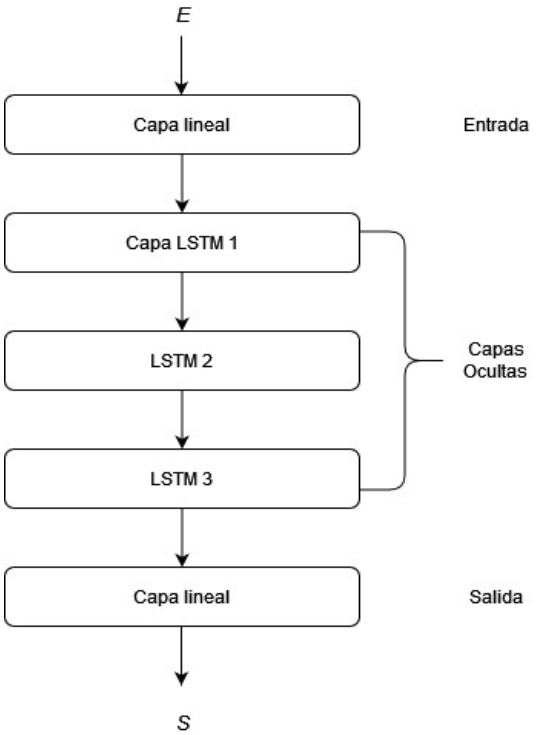


Figura 3.12: Arquitectura LSTM.

3.5.2. Gated Recurrent Unit

El diseño del sistema GRU es muy similar al de la red LSTM, tenemos una entrada lineal, un número de capas GRU por definir y una capa lineal de salida que devuelve el precio de la criptomoneda. También se prueban diferentes configuraciones para buscar la configuración óptima. La arquitectura está representada en la Figura 3.13

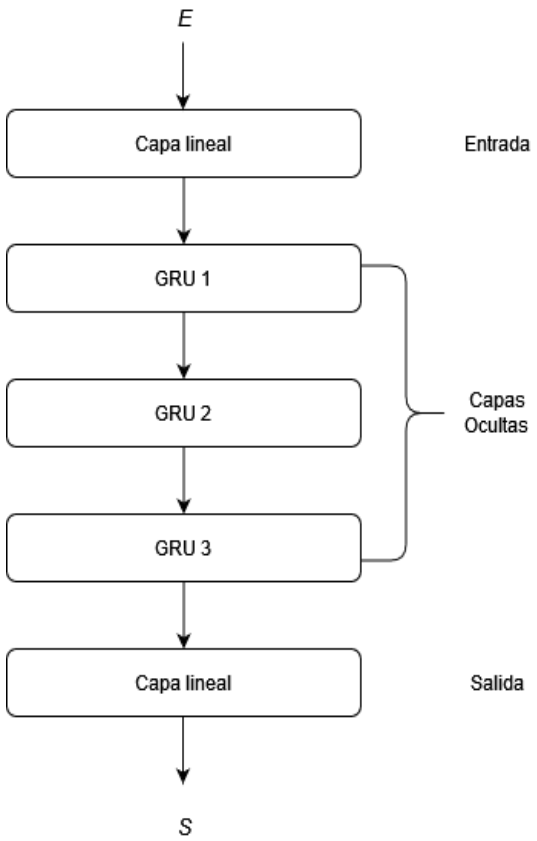


Figura 3.13: Arquitectura GRU.

3.6. Hiperparámetros

En este apartado se introducen y definen los hiperparámetros que se deben ajustar en cada sistema y para cada tiempo de predicción. En el Capítulo 4, destinado a los resultados, se muestran los hiperparámetros utilizados y como afectan a las métricas empleadas para determinar la precisión del modelo. Los hiperparámetros se obtienen de manera heurística, es decir, son una configuración externa que tenemos que determinar en base a un método de prueba y error. Son difíciles de optimizar ya que el método es inexacto y la mejora del sistema depende de la cantidad de hiperparámetros que se prueben.

3.6.1. Hiperparámetros de LSTM y GRU

Tanto el modelo LSTM como el modelo GRU tienen los mismos hiperparámetros:

- Ratio de aprendizaje: El método del descenso de gradiente nos permite automatizar la búsqueda de los coeficientes que minimizan el MSE, los coeficientes se actualizan de acuerdo a la Ecuación 3.3, dónde α es el ratio de aprendizaje. Si es demasiado grande los cambios en W serán muy bruscos, si es demasiado pequeño el descenso de gradiente tardará mucho en encontrar la solución.
$$W \longleftarrow W - \alpha \frac{\partial MSE}{\partial W} \tag{3.3}$$

- Batch size: Este parámetro define el número de muestras que pasan por el algoritmo para actualizar el modelo. En nuestro caso usamos un descenso de gradiente Mini-Batch, lo que significa que el Batch size está entre 1 y el tamaño del dataset.

- Epochs: Un epoch está comprendido por varios batches (lotes). Se puede pensar como un bucle, donde cada vez que se introduce todo el dataset en el sistema es un epoch, dentro de este bucle hay otro bucle anidado que itera las muestras con el batch size sobre todo el dataset. Es decir, el número de epochs es la cantidad de veces que todos los datos pasan por el algoritmo y el batch size es el número de muestras en las que se divide el dataset para actualizar el modelo.

- Número de neuronas de la capa oculta: El número de neuronas es el número de nodos que se encuentran dentro de la capa LSTM. Este número es difícil de obtener puesto que es complicado predecir cuál será su influencia en la red, por lo que se prueba de manera heurística. Este hiperparámetro puede provocar overfitting o underfitting.

- Número de datos: El número de datos se modifica cambiando el período de muestreo. A este valor también se le denomina granularidad. Este hiperparámetro puede generar overfitting o underfitting.

- Funciones de activación: Como se explicó en la Sección 2.6 las funciones de activación devuelven un resultado binario indicando si la neurona se activa, son una parte fundamental de las redes neuronales. En el sistema LSTM usamos

una función Rectificada linear (ReLU), esta función devuelve la entrada si es positiva y 0 si es negativa.

- Número de capas ocultas: La acumulación de capas ocultas entrega más profundidad a la red. Como estamos operando sobre una secuencia temporal de datos, añadir capas significa añadir niveles de abstracción, por lo que conseguimos modelos más complejos y eficientes.

- Optimizador: La función del optimizador es reducir la función de error. Para esto, calcula el gradiente de la función de coste y modifica el peso de cada nodo en la dirección negativa del gradiente[29]. Existen múltiples métodos iterativos que reducen la función de error. Los más destacados y los que usaremos en este TFG son: ADAM(Adaptive moment estimation), SGD(Stochastic Gradient Descent) y RMSprop(Root Mean Square Propagation).

Capítulo 4

Resultados

En este Capítulo se muestran y analizan los datos obtenidos. Además se evalúa la precisión de cada modelo, de acuerdo a las métricas previamente definidas en la Sección 2.12. Para simplificar la visualización e interpretación de los resultados se generan distintos modelos. Estos se utilizan para la sintonización de los hiperparámetros (ver Tabla 4.1). Se prueban los modelos en ambos algoritmos, ya que los dos tienen los mismos hiperparámetros. Para generar estos modelos modificamos los hiperparámetros definidos en el Capítulo 3: Ratio de aprendizaje, Batch size, Epochs, Número de neuronas de la capa oculta, Número de datos, Número de capas ocultas y optimizador. Se crean un total de 10 modelos con las combinaciones más relevantes y a cada modelo se le asigna un número de identificación del 1 al 10. El tiempo de predicción (TP) es modificado en los primeros cuatro modelos, analizando su influencia en los resultados. Eligiendo unos tiempos de predicción de 1, 5, 10 y 20 días. En los modelos 5 y 6 se modifica el Batch Size y el número de neuronas. Por otra parte, en los modelos 7 y 8 se aumenta el número de capas LSTM (aplicando dos y tres capas). Finalmente, en los modelos 8, 9 y 10 se aumenta el Batch size hasta 128. Concluyendo el capítulo se discuten y analizan los resultados.

Model	**Batch Size**	**Neuronas**	**Nº capas ocultas**	**Nº Datos**	**TP**
1	32	32	1	6 373	1 días
2	32	32	1	6 373	5 días
3	32	32	1	6 373	10 días
4	32	32	1	6 373	20 días
5	64	64	1	6 373	5 días
6	64	128	1	6 373	5 días
7	64	32	2	6 373	5 días
8	128	32	3	6 373	5 días
9	128	64	2	6 373	5 días
10	128	32	2	6 373	5 días

Tabla 4.1: Modelos generados.

4.1. LSTM

De acuerdo al diseño realizado en el Capítulo 3 la red LSTM es probada con distintos hiperparámetros, para acercarnos a los valores que entreguen mejores resultados.

Se han usado 3 optimizadores: ADAM, SGD y RMSprop. Se ha evaluado su rendimiento con los mismos hiperparámetros. Dichas condiciones corresponden al modelo 10 mostrado en la Tabla 4.1. Estas pruebas han demostrado que el mejor optimizador es ADAM, por lo que se ha usado en todos los modelos. Los resultados de dichas pruebas se encuentran en la Tabla 4.2. En cuanto al learning rate, se ha decidido usar el método de step decay, en este método el factor de aprendizaje decae cada cierto número de epochs. En este Trabajo de Fin de Grado el factor de aprendizaje decae un 10 % cada 20 epochs (ver Figura 4.1).

Las métricas que devuelven los modelos LSTM se plasman en la Tabla 4.3. El objetivo es determinar que hiperparámetros no tienen un impacto real en los resultados y que número de capas LSTM y neuronas son las óptimas para la red recurrente. Del mismo modo, se ha calculado de manera empírica cual es el mayor número de puntos para que no exista overfitting, concluyendo que el mejor intervalo entre puntos es de 6 horas, dando un total aproximado de 6 373 datos a fecha 30/12/2021 hora 19:25 (cada 6 horas se añaden más datos al sistema). A partir de un intervalo de 12 horas el error aumenta a la vez que aumenta el número de datos.

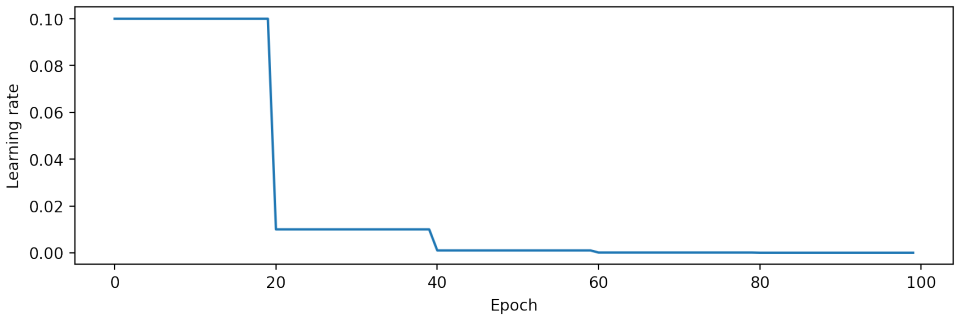


Figura 4.1: Disminución learning rate.

Optim	RMSE [$]	R^2	NRMSE[%]	MAE[$]	MSE[^{2}]	ME[$]
ADAM	1 852.908	0.961	2.819	1 312.099	3 433 271.550	7 802.806
SGD	6 219.730	0.561	9.464	4 659.830	38 685 050.339	17 631.753
RMSprop	29 373.787	-8.790	44.698	27 833.537	862 819 387.730	47 646.414

Tabla 4.2: Pruebas del optimizador.

4.1.1. Resultados de los modelos

Se ejecuta el algoritmo repetidas veces con las características mostradas en la Tabla 4.1, las métricas son apuntadas después del funcionamiento del modelo y se plasman en la Tabla 4.3. Para apreciar e interpretar mejor los resultados se crean

Model	RMSE [$]	R^2	NRMSE[%]	MAE[$]	MSE[^{2}]	ME[$]
1	2 891.502	0.90527	4.40008	1 702.9639	8 360 785.434	18 672.915
2	3 820.335	0.8349	5.813	2 577.389	14 594 964.803	41 114.888
3	4 952.0110	0.7236	7.5356	3 483.612	24 522 413.725	16 222.16059
4	78 732.319	-68.4118	119.809	32 589.254	6 198 778 193.940	257 719.795
5	1 745.780	0.9655	2.656	1 350.577	3 047 750.453	9 215.804
6	17 100.162	-2.3052	26.021	11 534.893	292 415 558.072	40 928.227
7	2 607.902	0.923	3.968	1 870.255	6 801 156.357	9 554.627
8	4 249.021	0.7957	6.4658	2 933.886	18 054 179.490	11079.675
9	5852.049	0.612	8.905	3 985.067	5 852.049	18 161.689
10	1346.690	0.9795	2.0404	921.269	1 813 574.207	12 272.89

Tabla 4.3: Resultados de los modelos LSTM.

varias gráficas con las métricas obtenidas, de esta forma se puede visualizar la información de una manera más clara. El modelo 4 es retirado de las gráficas debido a que introduce mucho error al predecir a 20 días, cambiando la escala e impidiendo visualizar correctamente el resto de la información. En la Figura 4.2 se muestra el valor del RMSE en azul y el del MAE en naranja. Es importante recordar que el RMSE indica el ajuste absoluto del modelo a los datos, es decir, la raiz de la diferencia entre el vector de predicciones y el vector con los valores reales. El MAE es el error absoluto medio, la diferencia absoluta entre los valores reales y las absolutas. Se puede observar como los mínimos RMSE y MAE se obtienen con los modelos 10 (neuronas = 64, batch size = 128, capas ocultas = 2) y 5 (neuronas = 64, batch size = 64, capas ocultas = 1). Y los máximos errores se obtienen en el modelo 4, debido al TP = 20 días, y en el modelo 6 (neuronas = 64, batch size = 128, capas ocultas = 1). Todos los modelos menos el 6 y el 4 obtienen valores similares en RMSE y MAE, en torno a los 3600 USD y 2350 USD respectivamente.

En la Figura 4.3 se grafica el máximo error cometido por el algoritmo, obviamente cuánto menor sea mejor es la predicción. El máximo error cometido por los modelos

(ver Figura 4.3) se produce en el modelo 6 y 2, en torno a los 40 000 USD. El mínimo error máximo es aproximadamente de 10 000 USD, obtenido por los modelos 5 y 7.

En la Figura 4.4 se representa R^2, el coeficiente de determinación. Esta métrica puede tomar los valores entre $-\infty$ y 1. Dónde un valor de 1 significa que el ajuste del modelo a la variable es perfecto. Cuanto menor sea este valor peor ajustado se encuentra el modelo, por lo que es menos fiable. El peor resultado corresponde al modelo 6, lo que coincide con el resto de métricas. Los mejores modelos son el 5 y el 10.

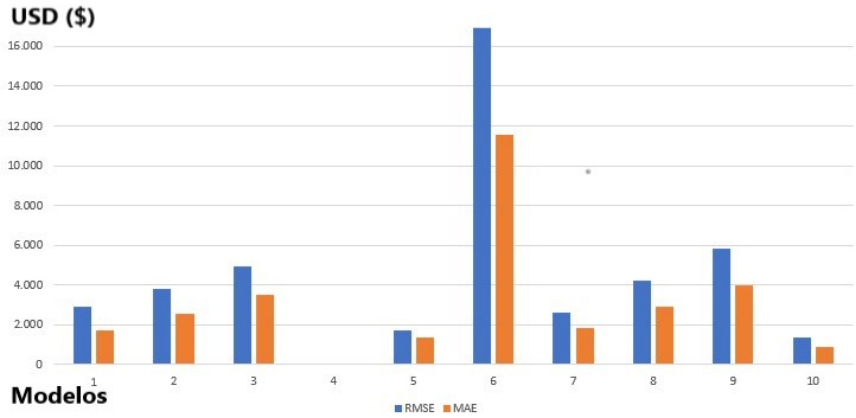


Figura 4.2: RMSE y MAE modelos LSTM.

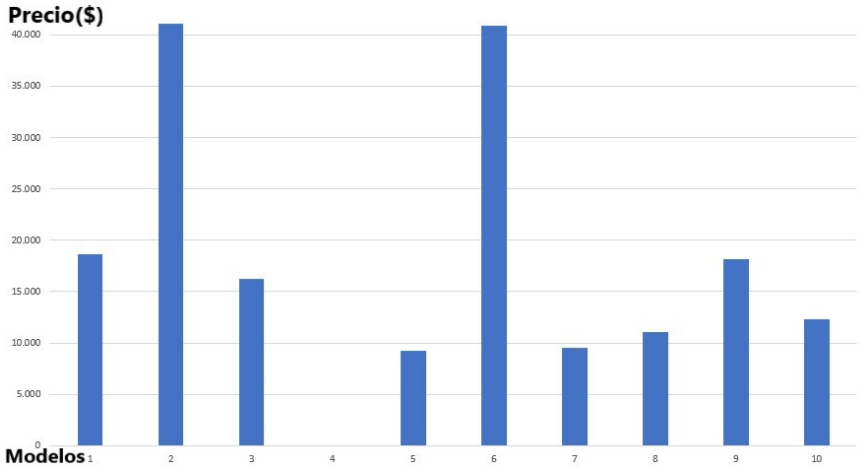


Figura 4.3: Máximo error modelos LSTM.

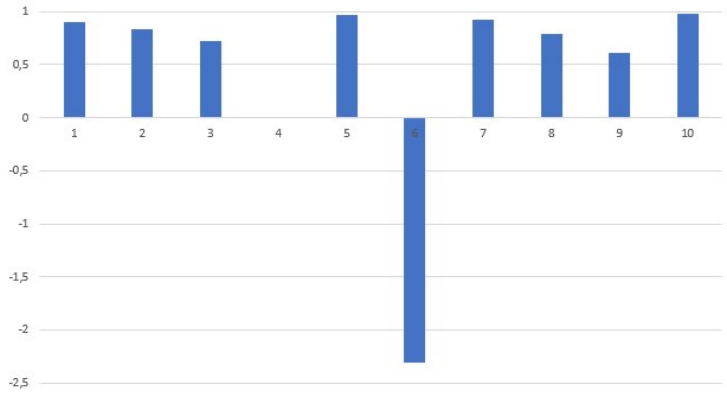
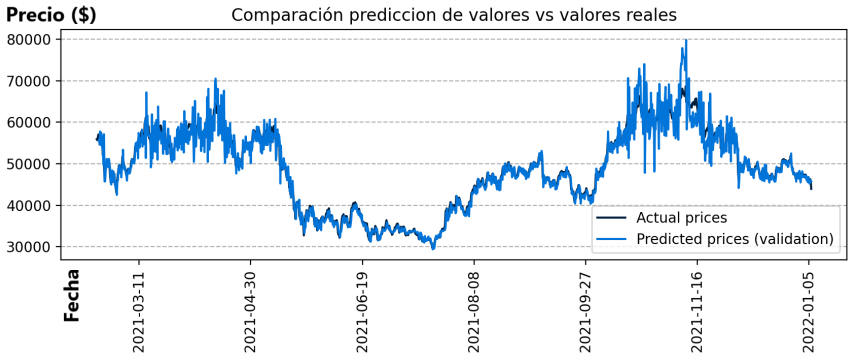


Figura 4.4: R^2 modelos LSTM.

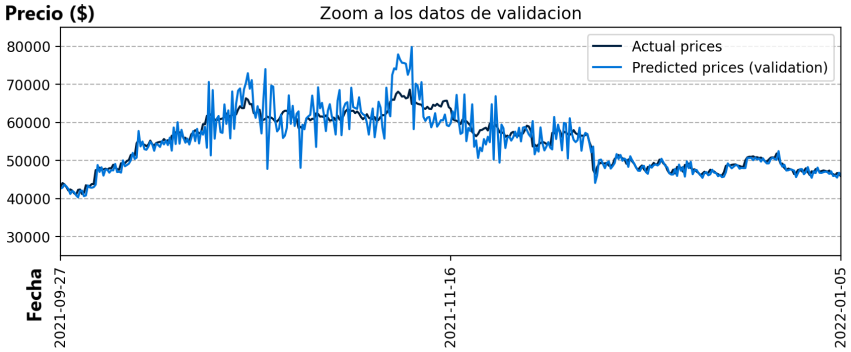
4.1.2. Evaluación de los Modelos LSTM

Es importante tener en cuenta que los modelos de aprendizaje profundo son aleatorios, es decir, cada vez que el modelo se entrena con los mismos datos puede dar predicciones distintas. Esto es debido a que las redes neuronales tienen aleatoriedad en diversos puntos del proceso de entrenamiento, como por ejemplo al inicializar los pesos del modelo. Como se puede observar en el apartado 4.1.1 los mejores resultados son obtenidos por los modelos 5 (RMSE = 1 745.780$, R^2 = 0.9655, ME = 9 215.804$) y 10 (RMSE = 1 346.690$, R^2 = 0.9795, ME = 12 272.89$). A continuación, en la Figura **??** se visualiza la predicción realizada por el modelo 5 para los datos de validación, que corresponden a un 20 % del dataset, desde el 11/03/2021 hasta el 05/01/2022. En esta gráfica la curva negra es el precio real y la curva azul claro es la predicción realizada por el algoritmo, a grandes rasgos la predicción se acerca a los valores reales pero no es completamente exacta. En la Figura **??** se realiza un zoom a la Figura **??**. En esta figura, el eje X va desde el 27/09/2021 hasta el 05/01/2022. Excepto en algunos tramos el precio predicho y el real se solapan. En la Figura **??** se muestra la predicción del modelo 10 vs los valores reales del precio del BCT y en la Figura **??** se realiza un zoom a la Figura **??**, igual que en el caso del modelo 5 las predicciones se acercan a los valores reales pero no son completamente exactas. Estas son las gráficas de los mejores modelos, el 5 y el 10.

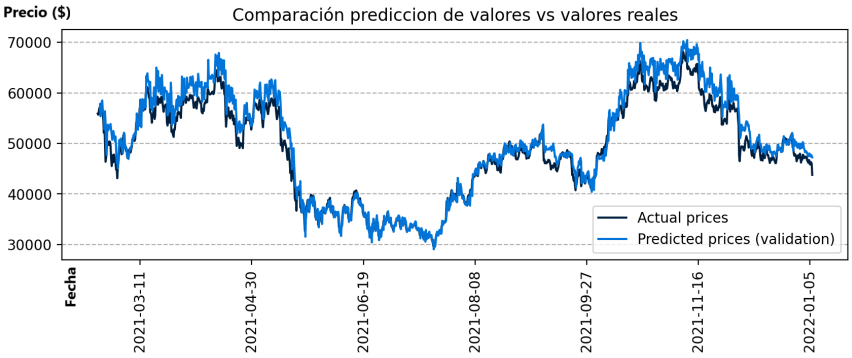
En cuanto a los modelos que realizan las peores predicciones están el modelo 4 (RMSE = 78 732.319$, R^2 = -68.4118, ME = 257 719.795$) y el modelo 6 (RMSE = 17 100.162$, R^2 = -2.3052, ME = 40 928.227$). En la Figura **??** se muestran las predicciones del modelo 6 y en la Figura **??** el zoom de la predicción realizada, dónde la línea negra representa la curva del precio actual y la línea azul claro la de la predicción realizada. En estas Figuras se observa como la predicción del modelo se separa hasta 20 000 dólares del precio real, en el modelo 6 se utiza 1 capa oculta, 128 neuronas y un Batch Size de 64. En la Figura **??** y **??** se grafican las predicciones del modelo 4 y los precios reales, se observa que las predicciones tienen un alto ruido, acercandose a los valores reales pero con una alta variación y una gran cantidad de picos.



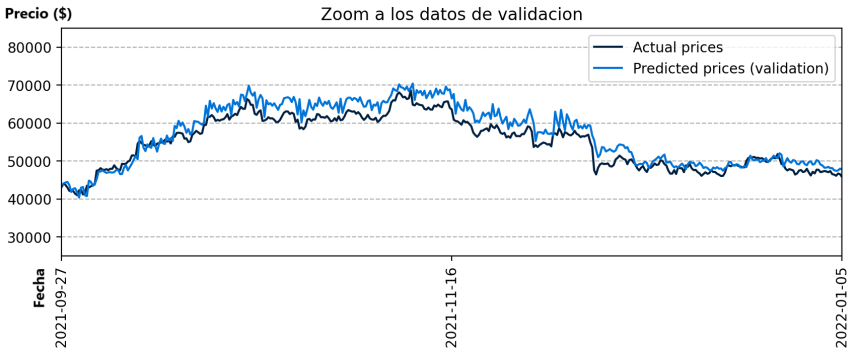
(a) Modelo 5 LSTM predicción.



(b) Modelo 5 LSTM zoom.

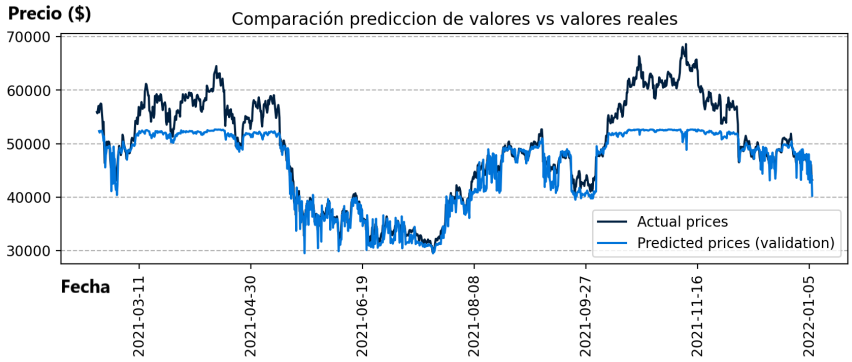


(c) Modelo 10 LSTM predicción.

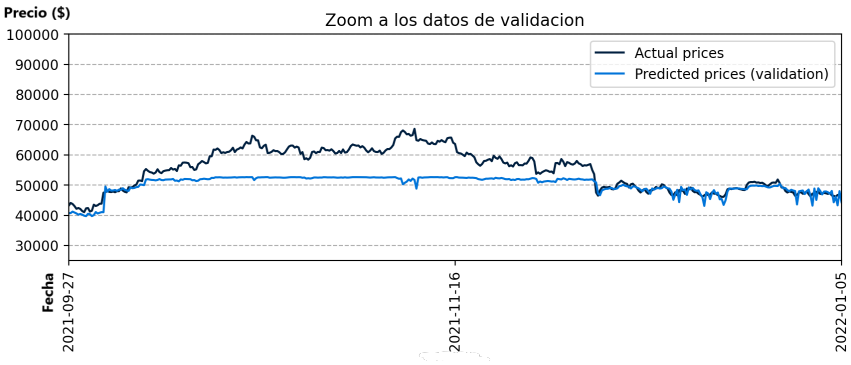


(d) Modelo 10 LSTM zoom.

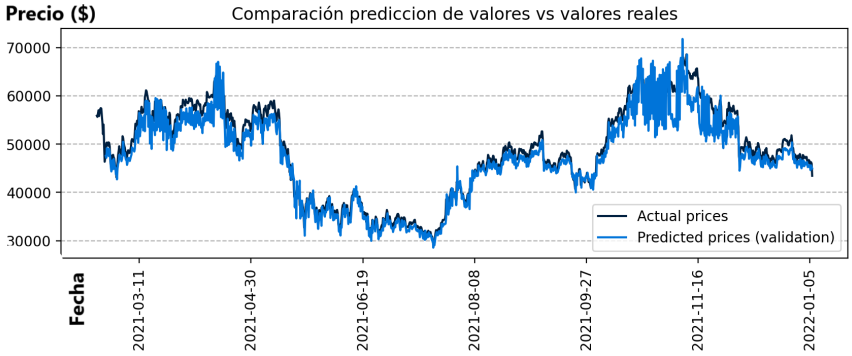
Figura 4.5: Predicciones LSTM mejores modelos



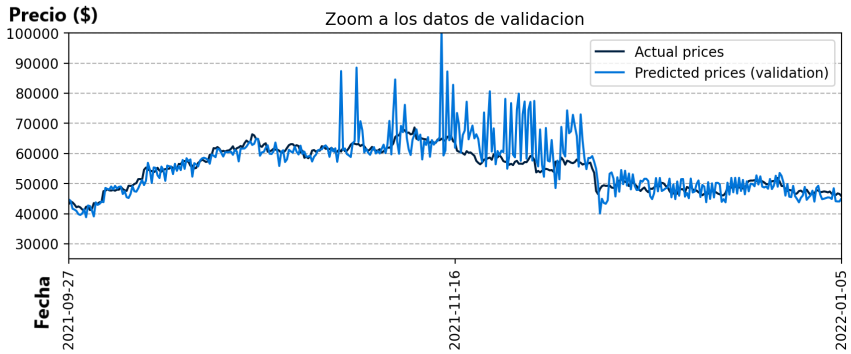
(a) Modelo 6 LSTM predicción.



(b) Modelo 6 LSTM zoom.



(c) Modelo 4 LSTM predicción.



(d) Modelo 4 LSTM zoom.

Figura 4.6: Predicciones LSTM peores modelos

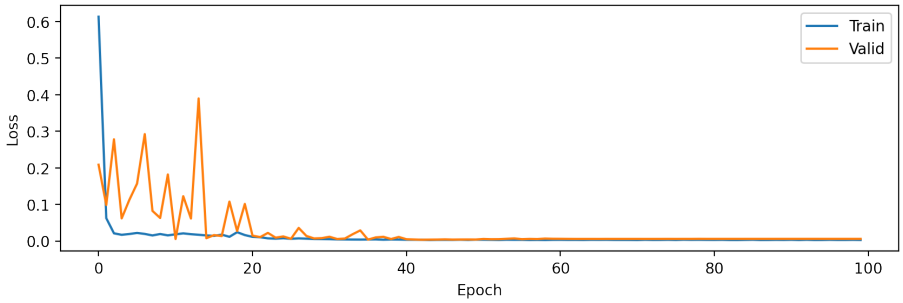
Una forma de visualizar el correcto aprendizaje del modelo es a través de la curva de aprendizaje. Estas curvas de aprendizaje se usan para diagnosticar problemas como el *underfitting* o el *overfitting*, dónde el eje Y corresponde con la pérdida (MSE) y el eje X con cada epoch. La curva azul se corresponde con el error del entrenamiento y la curva naranja con la validación. La curva del entrenamiento nos indica cómo está aprendiendo el modelo y la curva de validación refleja la forma en la que se generaliza el modelo y realiza predicciones. Pueden ocurrir 3 casos: *overfitting*, *underfitting* y *good fit*. En el primer caso el modelo se vuelve demasiado especializado con los datos del entrenamiento, por lo que tiene dificultades para predecir nuevos datos. Este comportamiento se puede distinguir en la curva de aprendizaje, si la pérdida de entrenamiento continúa disminuyendo con cada epoch o si la curva de validación disminuye, pero llegado a un punto comienza a aumentar de nuevo. *Underfitting* se refiere a un modelo que no ha aprendido correctamente en la etapa de entrenamiento, se puede identificar con la curva de aprendizaje si la pérdida en el entrenamiento es una línea plana o tiene demasiadas pérdidas. El resultado buscado es un modelo con *good fit* y se identifica por una pérdida de entrenamiento y validación que desciende hasta un punto donde se estabiliza con una diferencia insignificante entre ellas.

En la Figura 4.7, se grafican las curvas de aprendizaje de los mejores modelos (modelo 10 y modelo 5) y de los peores modelos (modelo 4 y modelo 6). La curva de validación del modelo 5 tiene un ajuste lento pero converge a un valor muy cercano a 0, igual que en el modelo 10. Por lo que concluimos que tiene un *good fit*. En el modelo 4 se aprecia un alto valor de ruido en la curva de validación y al llegar a la estabilidad se crea una separación entre el valor de entrenamiento y el de validación, igual que en el modelo 6. Esto significa que el modelo no está realizando buenas predicciones, lo que confirma los resultados expuestos previamente. Se observa también que no son necesarias 100 epochs pues las curvas alcanzan su valor mínimo entorno a la epoch 50.

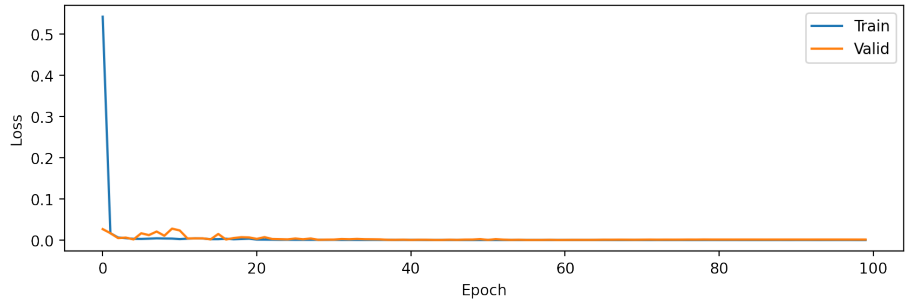
Por último, se crea una tabla (Tabla 4.4) con los hiperparámetros óptimos para el modelo y se genera una predicción para el día posterior (06/01/2022) para todos los modelos, la cual se visualiza en la Figura 4.8.

Optimizador	**Batch Size**	**Neuronas**	**Nº capas ocultas**	**Nº Datos**	**TP**
Adam	128	32	2	6373	5 días

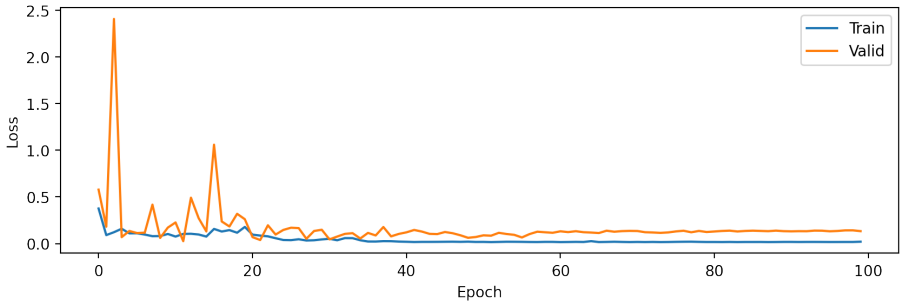
Tabla 4.4: Hiperparámetros óptimos LSTM.



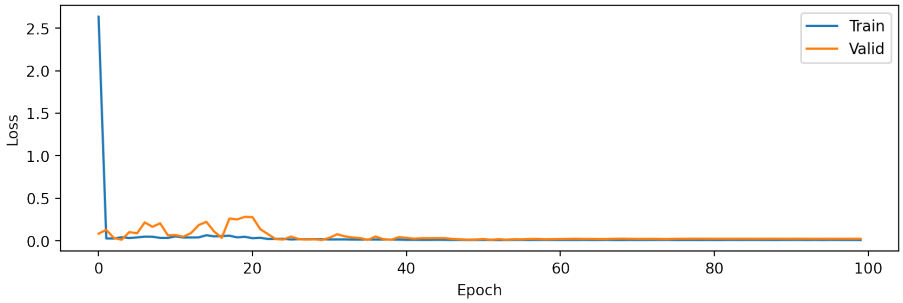
(a) Modelo 5



(b) Modelo 10

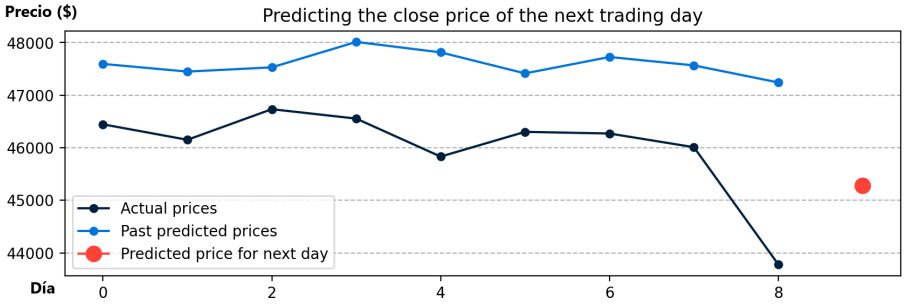


(c) Modelo 4

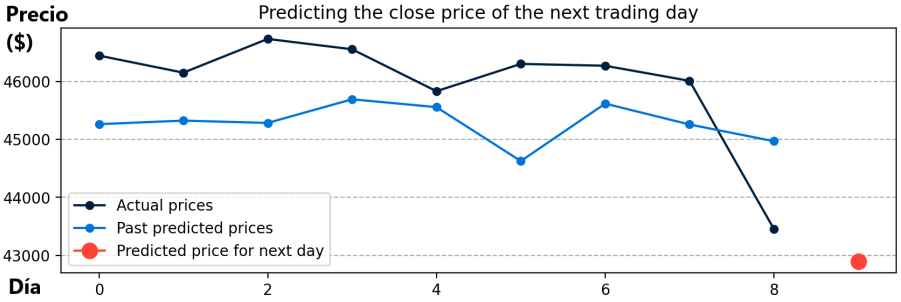


(d) Modelo 6

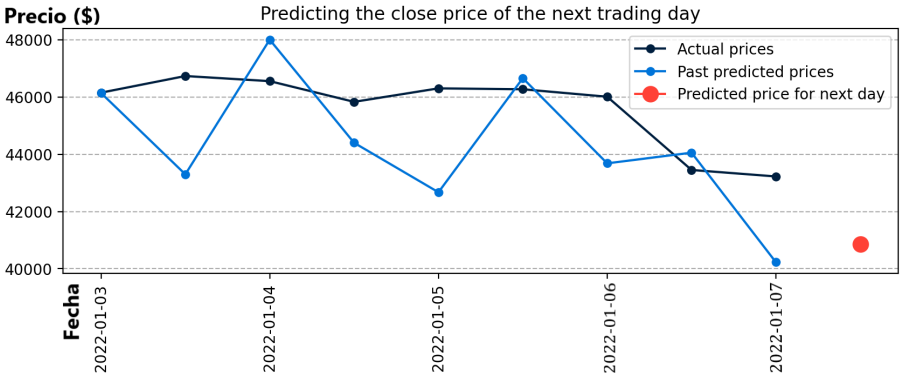
Figura 4.7: Curvas de aprendizaje LSTM



(a) Modelo 10



(b) Modelo 4



(c) Modelo 6

Figura 4.8: Predicción próximo día

4.2. GRU

En esta sección se muestran los resultados de la red neuronal GRU. Las redes neuronales GRU y LSTM son extremadamente parecidas. Se realizan las pruebas del sistema GRU sobre los modelos definidos previamente en la Tabla 4.1. Las métricas obtenidas (RMSE, R^2, NRMSE, MAE, MSE, ME) se exponen en la Tabla 4.5 y la representación de las métricas RMSE, R^2 y ME se encuentra en las Figuras **??**, **??** y **??** respectivamente.

4.2.1. Resultados de los modelos

El algoritmo se ejecuta múltiples veces con los hiperparámetros de la Tabla 4.1. Las métricas que devuelven los algoritmos, definidas en la Sección 2.12 se apuntan en la Tabla 4.5. Para visualizar mejor los resultados se generan gráficas con las métricas obtenidas de los modelos.

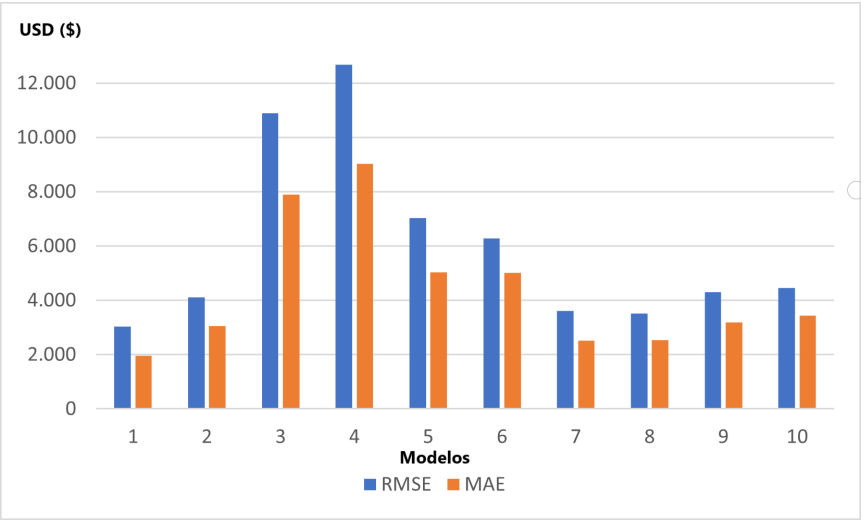
En la Figura 4.9 (a) se representa el valor del RSME (color azul) y el de MAE (color naranja) de cada modelo, los menores valores se observan en el modelo 7 (Batch Size = 64, Neuronas = 32, Nº capas ocultas = 2, TP = 5) y 8 (Batch Size = 128, Neuronas = 32, Nº capas ocultas = 3, TP = 5). En el modelo 1 se obtienen las mejores métricas al tener el TP de 1 día. Los peores resultados corresponden con los modelos 3 (Batch Size = 32, Neuronas = 32, Nº capas ocultas = 1, TP = 10) y 5 (Batch Size = 64, Neuronas = 32, Nº capas ocultas = 2, TP = 5). El modelo 4 tiene los peores resultados al tener un TP = 20.

En la Figura 4.9 (b) se grafica el máximo error de los modelos GRU. El mínimo valor lo consigue el modelo 7 y 8, lo que coincide con la Figura anterior. En el peor caso se encuentra el modelo 4, 3 y 6 (Batch Size = 64, Neuronas = 32, Nº capas ocultas = 2, TP = 5).

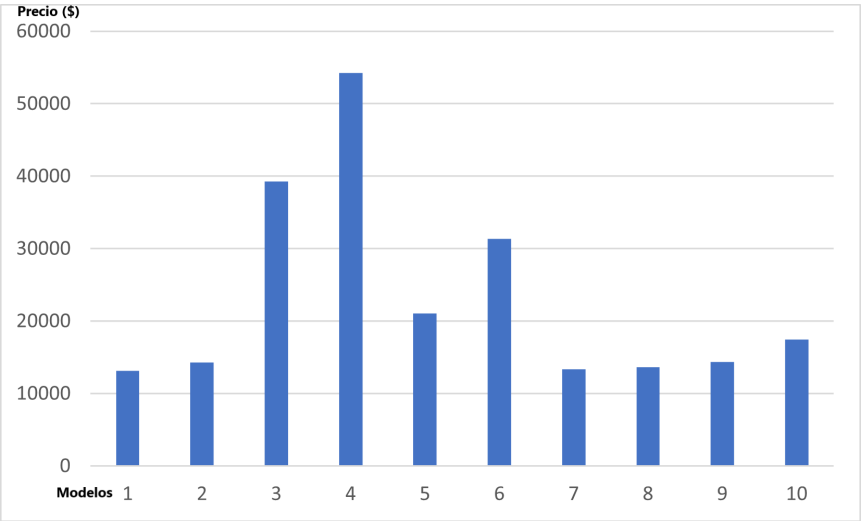
La Figura 4.9 (c) muestra la métrica R^2 dónde de nuevo los mejores valores se obtienen en los modelos 1, 7 y 8.

Model	**RMSE [$]**	**R^2**	**NRMSE[%]**	**MAE[$]**	**MSE[^{2}]**	**ME[$]**
1	3 028.056	0.895	4.607	1 948.369	9 169 128.963	13 119.912
2	4 117.811	0.8066	6.266	3 056.817	16 956 367.986	14 277.289
3	10 888.404	-0.3408	16.569	7 905.285	118 557 355.629	39 298.223
4	12 674.102	-0.816	19.286	9 029.361	160 632 872.309	54 252.156
5	7 034.799	0.435	10.705	5 025.176	49 488 397.844	21 030.614
6	6 273.617	0.551	9.546	5 013.028	39 358 272.062	18 057.276
7	3 606.646	0.851	5.488	2 505.744	13 007 900.970	13 342.012
8	3 509.646	0.859	5.340	2 533.092	12 317 616.772	13 620.421
9	4 297.698	0.789	6.539	3 183.329	18 470 216.543	14 321.015
10	4 462.358	0.772	6.790	3 443.462	19 912 643.271	17 467.315

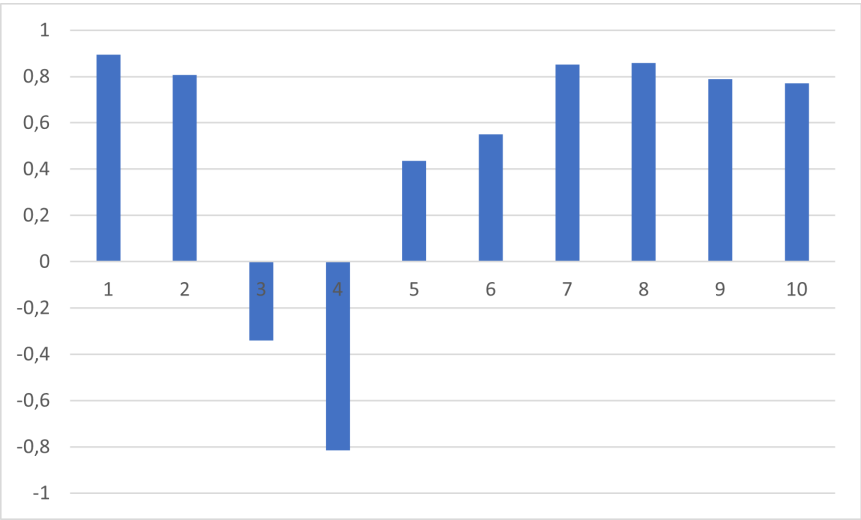
Tabla 4.5: Resultados de los modelos GRU.



(a) RMSE y MAE modelos GRU.



(b) Máximo error modelos GRU.



(c) R^2 modelos GRU.

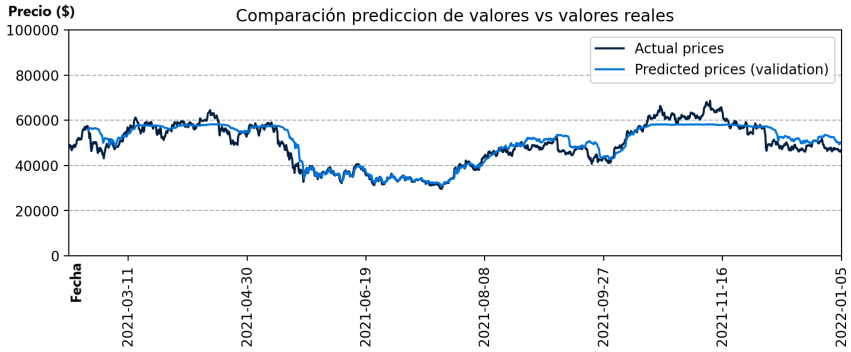
Figura 4.9: Métricas GRU

4.2.2. Evaluación de los modelos GRU

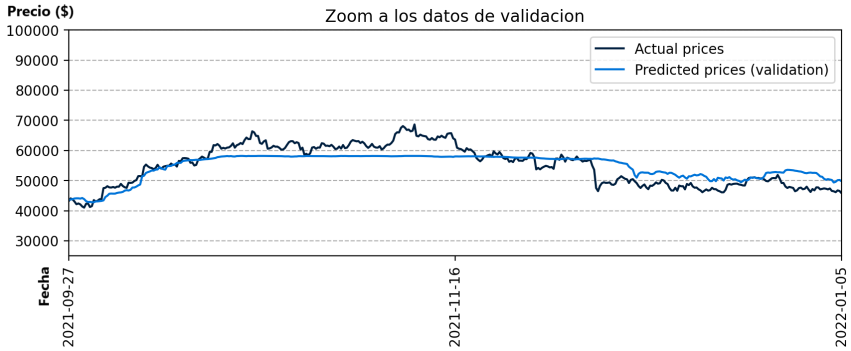
En el caso de los modelos GRU no existe ningún modelo que destaque por su rendimiento, todos tienen métricas similares, al contrario de lo que sucedía en los modelos LSTM. Los mejores modelos son el 8 (RMSE = 3 509.646$, R^2 = 0.859, ME = 13 620.421$) y el 7 (RMSE = 3 606.646$, R^2 = 0.851, ME = 13 342.012$).

Se generan gráficas del precio real y el precio predicho para los datos de validación, dichos datos van desde el 11/03/2021 hasta el 05/01/2022. El precio real está en color negro y el precio predicho se representa mediante una línea azul. En la Figura 4.10 (a) se encuentra la predicción del modelo 8 y en la Figura 4.10 (b) se ve un zoom de los datos desde el 27/09/2021 hasta el 05/01/2022. Se observa que las predicciones se acercan al precio real durante toda la serie de datos, con un error máximo de 13 620.421$. Los mismos gráficos para el modelo 7 se plasman en las Figuras 4.10 (a) (predicción) y 4.10 (b) (zoom). Los modelos que tienen mayor efectividad son los que más capas ocultas GRU poseen, aunque todos los modelos han obtenido unos errores similares.

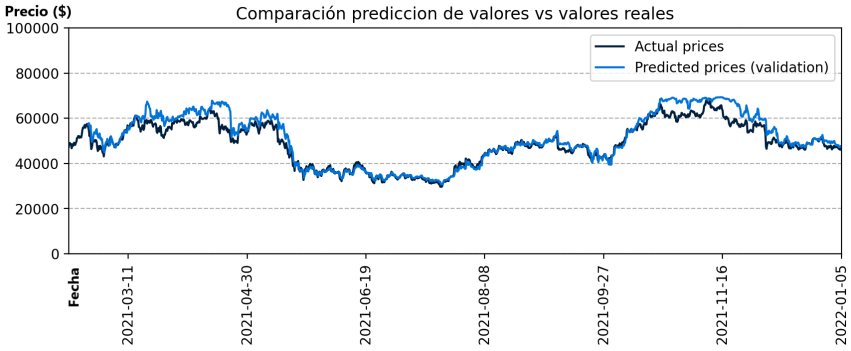
Los mejores modelos tienen una alta efectividad al realizar la predicción, como se puede ver en las Figuras 4.10 (a) y 4.10 (d). En el caso de los peores modelos nos encontramos con el Modelo 3 (RMSE = 10 888.404$, R^2 = -0.3408, ME = 39 298.223$) y el Modelo 4 (RMSE = 12 674.102$, R^2 = -0.816, ME = 54 252.156$). La predicción del Modelo 3 se encuentra en la Figura 4.11 (a) y el zoom en la Figura 4.11 (b), el resultado es notablemente peor que en el caso del Modelo 8 y 7. Esto es debido a que la predicción del Modelo 3 tiene un TP de 10 días, lo mismo pasa en el caso del Modelo 4, ya que tiene un TP de 20 días. De estos resultados podemos deducir que los algoritmos predicen fiablemente únicamente a corto plazo, entre 1 y 5 días. También se observa que cuando ocurren subidas o bajadas de precio muy bruscas el algoritmo falla al predecirlas, por este motivo podemos afirmar que el algoritmo funciona mejor cuando el precio del BTC se mantiene estable.



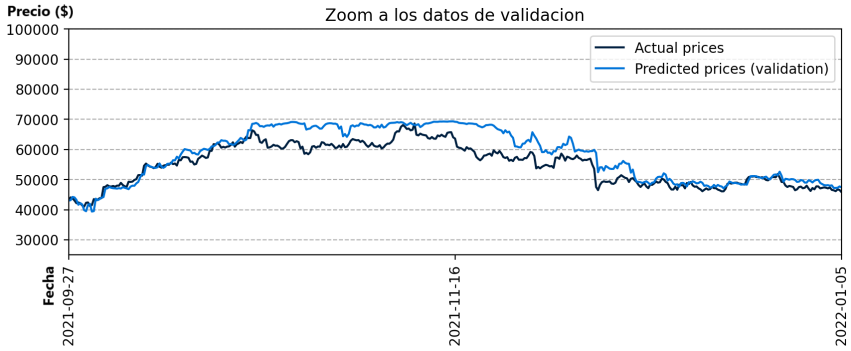
(a) Modelo 8 GRU predicción.



(b) Modelo 8 GRU zoom.

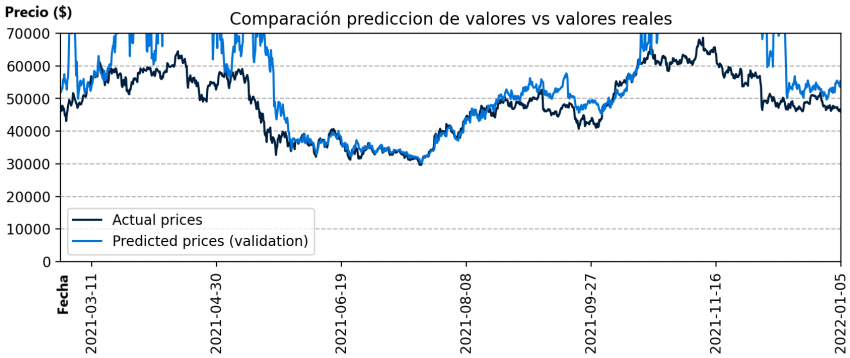


(c) Modelo 7 GRU predicción.

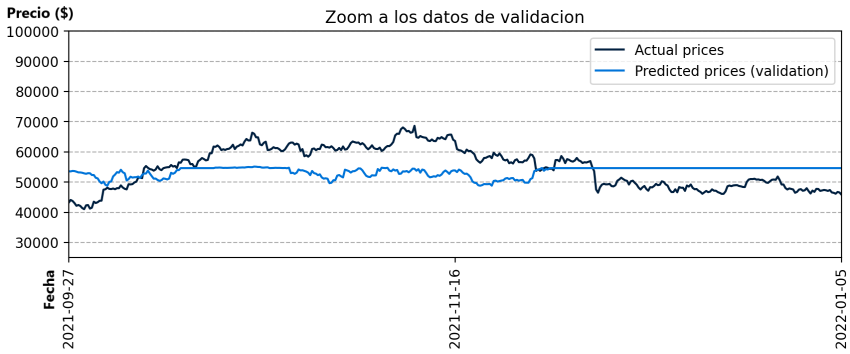


(d) Modelo 7 GRU zoom.

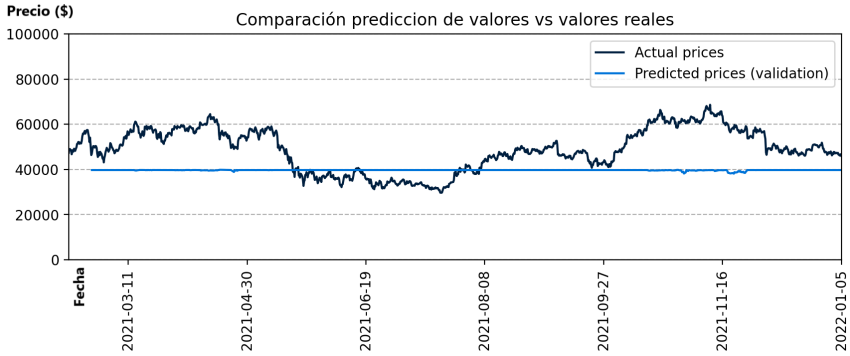
Figura 4.10: Predicciones GRU mejores modelos



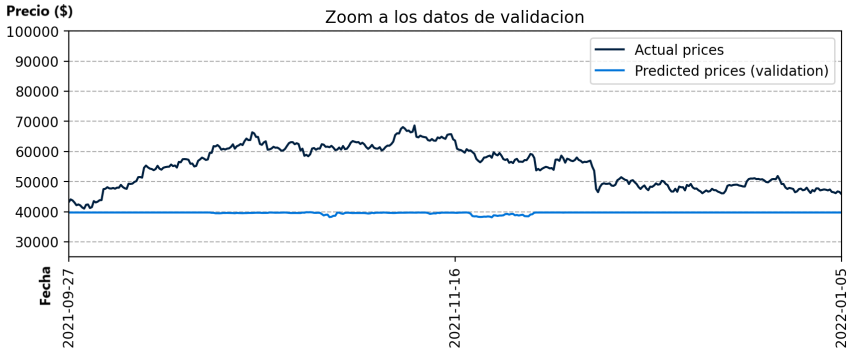
(a) Modelo 3 GRU predicción.



(b) Modelo 3 GRU zoom.



(c) Modelo 4 GRU predicción.



(d) Modelo 4 GRU zoom.

Figura 4.11: Predicciones GRU peores modelos

Igual que en el caso de LSTM se grafican las curvas de aprendizaje para analizar como funciona el algoritmo en la etapa de entrenamiento y en la etapa de validación. Estas curvas se muestran en la Figura 4.12. En ellas podemos ver como el modelo 8 (a) tiene un *good fit*, dónde el error de entrenamiento y el de validación descienden hasta casi alcanzar el 0. En el modelo 7 (b) sucede lo mismo, pero el error de validación es un poco más alto que el del entrenamiento. El modelo 4 (c) tiene un error de validación más alto, lo que significa que el modelo tiene *underfitting*. Por último en la parte (d), modelo 3 podemos ver un claro caso de *overfitting*, ya que la curva de perdidas de validación disminuye hasta cierto punto y después aumenta nuevamente.

4.3. Discusión de los resultados

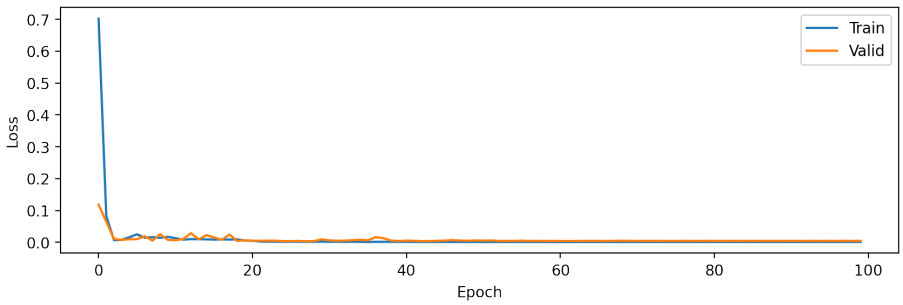
En esta sección se comparan los procedimientos y resultados obtenidos con los de otros autores. Se señalan las limitaciones y se destacan los aportes del TFG.

Analizando los resultados se puede concluir que ciertos hiperparámetros que en un principio no parecían tan importantes, tienen una incidencia muy alta en el resultado final, como por ejemplo el batch size. El mejor optimizador para nuestro caso es Adam, este resultado es avalado por la bibliografía encontrada, "hemos observado de una forma empírica cómo quizás el algoritmo Adam presenta un comportamiento adecuado en diferentes problemas"[29]. Además se comprueba que en el caso tratado por este TFG, no es necesario más de una capa LSTM/GRU para conseguir una buena predicción. Por otra parte, se concluye empíricamente que un alto número de neuronas provoca *overfitting*. En nuestra serie temporal al tener una variación de valores muy alta es difícil predecir a partir del día 10, obteniendo un error muy alto al intentar predecir a 20 días. Los datos son obtenidos desde la creación del BTC con un intervalo entre muestras de 6 horas, este valor entrega un número de datos óptimo de 6 373, con el paso del tiempo este número va a ir aumentando.

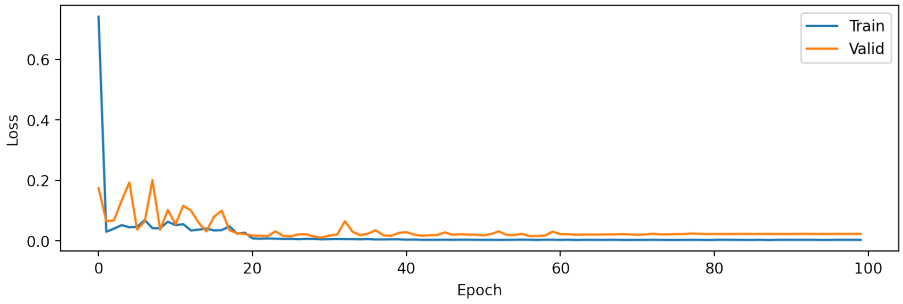
El mejor modelo GRU es el número 8 y el mejor modelo LSTM es el número 10, en la Tabla 4.6 se comparan ambos modelos, de esta tabla se desprende que los modelos LSTM tienen mejores resultados que sus pares en GRU. Se destaca que ambos modelos dan una predicción satisfactoria a corto plazo (entre 5 y 10 días), mientras que cuando se intenta predecir con un horizonte temporal de más de 10 días los modelos dan resultados poco acertados.

Model	**RMSE [$]**	**R^2**	**NRMSE[%]**	**MAE[$]**	**MSE[^{2}]**	**ME[$]**
8 GRU	3 509.646	0.859	5.340	2 533.092	1 2317 616.772	13 620.421
10 LSTM	1 346.690	0.9795	8.883	921.269	1 813 574.207	12 272.89

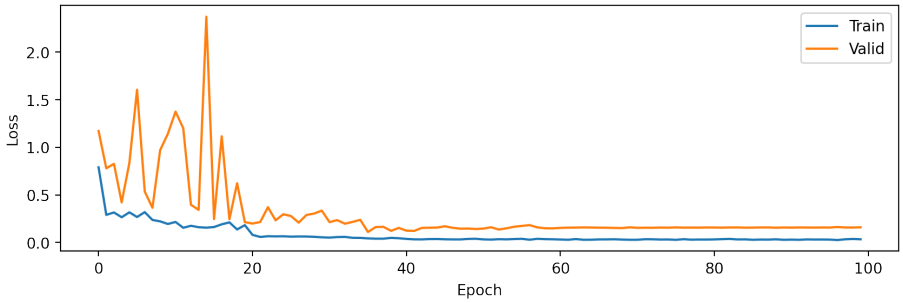
Tabla 4.6: Mejores modelos LSTM y GRU.



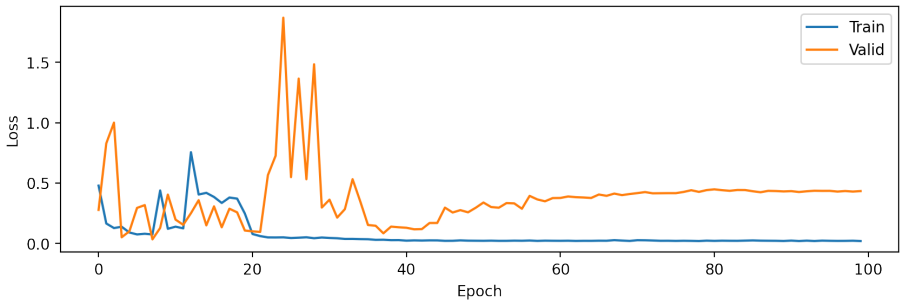
(a) Modelo 8



(b) Modelo 7



(c) Modelo 4



(d) Modelo 3

Figura 4.12: Curvas de aprendizaje GRU

Capítulo 5

Conclusiones y Líneas Futuras

En este último capítulo se presentan las conclusiones extraídas de los resultados mostrados en el Capitulo 4. Para finalizar el capítulo se exponen las principales líneas futuras que se pueden tomar en este trabajo.

5.1. Conclusiones

El problema tratado y objetivo principal de este TFG ha sido la predicción mediante técnicas de Machine Learning de una serie temporal, en este caso el precio del Bitcoin. Los algoritmos fueron desarrollados en Python con la ayuda de la librería PyTorch y ejecutados en Google Colab. Para el análisis de los algoritmos se definió una serie de modelos sobre los que se realizaron numerosas pruebas, con el objetivo de evaluar la validez de estos y sintonizar los hiperparámetros de cada Red Neuronal. Para entrenar los algoritmos se utiliza el precio histórico del Bitcoin (BTC), estos datos se obtienen de una API llamada Binance. Las pruebas efectuadas proporcionaron una alta precisión en un horizonte temporal a corto plazo (menor a 10 días), los algoritmos no son fiables al realizar una predicción a largo plazo.

Se utilizaron Redes Neuronales Recurrentes de tipo Long Short Term Memory (LSTM) y Gated Recurrent Unit (GRU), se obtienen resultados similares, sin embargo, comparativamente el sistema de tipo LSTM ofrece un error menor que el GRU. El mejor modelo GRU obtuvo un NRMSE de 4.607 % y el mejor modelo LSTM alcanzó un NRMSE de 2.0404 %. Los hiperparámetros del mejor modelo LSTM son:

- Batch Size: 128
- Neuronas: 32
- Número de capas ocultas: 2
- Número de Datos: 6373
- Optimizador: ADAM
- Tiempo de predicción: 5

En cuanto a los hiperparámetros del mejor modelo GRU son:

- Batch Size: 128
- Neuronas: 32
- Número de capas ocultas: 3
- Número de Datos: 6373
- Optimizador: ADAM
- Tiempo de predicción: 5

De estos resultados se extraen las siguientes conclusiones. No es necesario un alto número de neuronas para alcanzar las mejores predicciones, ya que se observa que los modelos con más neuronas (128) obtienen un error mayor. Esto es debido a que el sistema se vuelve demasiado específico y se genera *overfitting*. El algoritmo que mejores resultados alcanza es Adam y el mejor TP es de 5 días.

Además, se ha alcanzado el segundo objetivo propuesto en la introducción de este Trabajo de Fin de Grado, siendo este realizar un marco teórico lo suficientemente profundo como para poder comprender los conceptos básicos del Machine Learning y las tecnologías utilizadas para el desarrollo de los sistemas expuestos.

5.2. Líneas futuras

Existen numerosas líneas de trabajo que pueden mejorar los resultados obtenidos en este TFG, a continuación, se explican brevemente algunas de ellas.

- Explorar los resultados de otros algoritmos de Machine Learning, tales como Redes Neuronales Convolucionales o Support-Vector Machine. Además es interesante probar combinaciones de distintos algoritmos, como por ejemplo, un modelo con capas superpuestas de LSTM y GRU.
- Implementar un bot que analizando la tendencia del precio del BTC, sea capaz de invertir a corto plazo automáticamente. Cuando predice una subida del valor del BTC compra criptomonedas y cuando detecta una bajada a corto plazo vende para disminuir las pérdidas.
- Aplicar los algoritmos diseñados en este TFG a otras criptomonedas para buscar resultados más precisos.
- Sintonizar más exhaustivamente los algoritmos, realizando más pruebas y modificando los hiperparámetros hasta lograr mejores resultados.

Apéndice A

Aspectos económicos, sociales y ambientales

Este apéndice tiene como objetivo especificar el impacto de este TFG en los aspectos económicos, sociales y ambientales.

A.1. Impacto social

La criptomoneda es un medio de pago cada vez más utilizado, con el auge del dinero electrónico existen muchas personas interesadas en esta forma de pago. Este TFG tiene un impacto directo en quienes deseen invertir de manera segura en este mercado.

A.2. Impacto económico

Este TFG se centra exclusivamente en el mercado de la criptomoneda y en su variación de precios. Por lo cual tiene un impacto económico directo.

A.3. Impacto medioambiental

El minado de las criptomonedas es altamente contaminante, debido a la corta vida util de los ordenadores utilizados para minar y por la gran cantidad de energía que necesita. Este TFG tiene un impacto ambiental negativo por esta razón.

A.4. Responsabilidad ética y profesional

Las criptomonedas y la tecnología blockchain provocarán un impacto en la sociedad a futuro. Estas tecnologías descentralizan el mercado y el impacto ético depende de como sean usadas por las personas. Este TFG tiene un objetivo puramente investigativo y cumple los estándares éticos.

Apéndice B

Presupuesto económico

Este TFG ha sido desarrollado durante 5 meses a través de un ordenador portátil. En este anexo se calcula un presupuesto considerando los recursos humanos y el equipamiento necesario.

- **Personal**: El personal involucrado se compone únicamente de un estudiante de ingeniería de telecomunicaciones, autor del TFG. Como se puede ver en la Tabla B.1

	Coste horario (€)	**Horas**	**Total (€)**
Estudiante de ingeniería	20	300	6 000
TOTAL			**6 000**

Tabla B.1: Costes de personal.

- **Costes de recursos materiales**: El software utilizado es de uso gratuito, por lo que solamente se considera el uso del ordenador pórtatil (ver Tabla B.2).

	Tiempo de vida (años)	**Uds.**	**Coste (€)**	**Amortización (€/mes)**	**Uso (meses)**	**Total (€)**
Ordenador portátil	5	1	800	13.33	5	66.65
TOTAL						**66.65**

Tabla B.2: Costes de recursos materiales.

El coste total del Trabajo de Fin de Grado con un IVA del 21 % asciende hasta los 7 340.64 €.

Bibliografía

[1] Marcelo Moreno. Origen, evolución y problemas del dinero hasta las criptomonedas. https://burjcdigital.urjc.es/bitstream/handle/10115/17971/Trabajo%20Fin%20de%20Grado.pdf?sequence=4&isAllowed=y, 2019. [Accesed online] 09/09/2021.

[2] Satoshi Nakamoto. The bitcoin whitepaper. https://bitcoin.org/bitcoin.pdf, 2009. [Accesed online] 30/08/2021.

[3] Stafford Global. Finance 2.0: What is cryptocurrency? https://www.staffordglobal.org/articles-and-blogs/finance-articles-blogs/cryptocurrency-bitcoin-finance-banking/, 2021. [Accesed online] 17/08/2021.

[4] Statista. Number of cryptocurrencies worldwide from 2013 to august 2021. /https://www.statista.com/statistics/863917/number-crypto-coins-tokens/, 2021. [Accesed online] 18/08/2021.

[5] Coin Market Cap. Total cryptocurrency market cap. https://coinmarketcap.com/es/charts/, 2021. [Accesed online] 18/08/2021.

[6] Tradingview. Dominio de capitalización total de mercado. https://es.tradingview.com/markets/cryptocurrencies/global-charts/, 2021. [Accesed online] 18/08/2021.

[7] Jose Luis Muñoz Tapia Carlos Dolader Retamal, Joan Bel Roig. La blockchain: Fundamentos, aplicaciones y relación con otras tecnologías disruptivas. https://www.mincotur.gob.es/Publicaciones/Publicacionesperiodicas/EconomiaIndustrial/RevistaEconomiaIndustrial/405/DOLADER,%20BEL%20Y%20MU%C3%91OZ.pdf, 2017. [Accesed online] 18/08/2021.

[8] Cecilia Pastorino. Blockchain: qué es, cómo funciona y cómo se está usando en el mercado. https://www.welivesecurity.com/la-es/2018/09/04/blockchain-que-es-como-funciona-y-como-se-esta-usando-en-el-mercado/, 2018. [Accesed online] 29/09/2021.

[9] Lázaro Blanco Encinosa. Criptomonedas. breve análisis desde la perspectiva económica y financiera. *Cofin Habana*, 2021.

[10] Lorién Lascorz Lozano. Aprendizaje por refuerzo: Elementos básicos y algoritmos. https://core.ac.uk/download/pdf/289996907.pdf, 2018. [Accesed online] 19/08/2021.

[11] Significado de neurona. https://www.significados.com/neurona/, 2021. [Accesed online] 29/09/2021.

[12] Perceptron. https://deepai.org/machine-learning-glossary-and-terms/perceptron. [Accesed online] 29/09/2021.

[13] Charu C.Aggarwal. *Data Classification. Algorithms and applications.* Chapman y Hall, 1 edition, 2015.

[14] Perceptrón multicapa. https://es.wikipedia.org/wiki/Perceptr%C3%B3n_multicapa, 2021. [Accesed online] 01/10/2021.

[15] John J Hopfield. *Neural network and physical systems with emergent collective computanional abilities.* Proceedings of the national academy of sciences, 1982.

[16] Redes neuronales recurrentes. https://torres.ai/redes-neuronales-recurrentes/, 2019. [Accesed online] 01/10/2021.

[17] Michael Phi. Illustrated guide to lstm's and gru's: A step by step explanation. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21, 2018. [Accesed online] 09/09/2021.

[18] Árbol de decisión en machine learning. https://sitiobigdata.com/2019/12/14/arbol-de-decision-en-machine-learning-parte-1/. [Accesed online] 02/10/2021.

[19] Leo Breiman. Random forests. https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf, 2001. [Accesed online] 19/08/2021.

[20] Top Big Data. Métricas de regresión para el aprendizaje automático. https://topbigdata.es/metricas-de-regresion-para-el-aprendizaje-automatico/, 2021. [Accesed online] 05/09/2021.

[21] Wang P Shen D, Urquhart A. Does twitter predict bitcoin? *Econ Lett*, 2019.

[22] Caton S McNally S, Roche J. Predicting the price of bitcoin using machine learning. *26th Euromicro international conference on parallel, distributed, and network-based processing*, 2018.

[23] Mohammed Mudassir et al. Time-series forecasting of bitcoin prices using high-dimensional features: a machine learning approach. *Springer Nature*, 2020.

[24] Dian Utami et al. Forecasting historical data of bitcoin using arima and α-sutte indicator. *Journal of Physics*, 2018.

[25] S. Lahmiri y S. Bekiros. Cryptocurrency forecasting with deep learning chaotic neural networks. *Chaos,Solitons and Fractals*, 2019.

[26] Sergio Delgado Quintero. Python. https://aprendepython.es/core/introduction/python/, 2020. [Accesed online] 17/08/2021.

[27] 2pd et al. binance-spot-api-docs. https://github.com/binance/binance-spot-api-docs/blob/master/rest-api.md, 2020. [Accesed online] 29/10/2021.

[28] Gráfico de velas. https://es.wikipedia.org/wiki/Gr%C3%A1fico_de_velas, 2020. [Accesed online] 23/12/2021.

[29] Luis Velasco. Optimizadores en redes neuronales profundas: un enfoque práctico. https://velascoluis.medium.com/optimizadores-en-redes-neuronales-profundas-un-enfoque-pr%C3%A1ctico-819b39a3eb5, 2020. [Accesed online] 23/12/2021.