

## **Maestría Oficial en Big Data y Data Science**

Análisis y predicción de mercado de criptodivisas  
con algoritmos de deep learning

14MBID Trabajo Fin de Máster

Alumno: de la Morena Coco, Carlos

Tutor: Fuentes Hurtado, Félix José

DNI: 70421561T

Fecha de entrega: 03/10/2022

# Índice

Resumen .....	6
1. Introducción .....	7
2. Objetivos.....	10
2.1. Evaluación de modelos.....	10
2.2. Conocimiento de factores de mercado.....	10
3. Estado del arte y Marco teórico .....	11
3.1. El mercado de las criptodivisas.....	11
3.1.1. Definición de criptomoneda .....	11
3.1.2. Características del mercado financiero .....	12
3.1.3. Características intrínsecas del mercado criptográfico .....	16
3.2. Proceso KDD.....	19
3.3. Ciencia de datos .....	21
3.4. Inteligencia artificial .....	21
3.5. Aprendizaje automático .....	22
3.6. Aprendizaje profundo.....	23
3.6.1. Perceptrón multicapa.....	24
3.6.2. Redes neuronales convolucionales .....	25
3.6.3. Redes neuronales recurrentes.....	27
3.7. Métricas.....	28
3.7.1. Error medio absoluto .....	28
3.7.2. Matriz de confusión.....	29
3.8. Antecedentes.....	30
4. Desarrollo del proyecto.....	33
4.1. Herramientas utilizadas .....	33
4.1.1. Anaconda .....	33
4.1.2. Python .....	34
4.1.3. Jupyter.....	34
4.1.4. Pandas .....	35
4.1.5. Tensorflow y Keras .....	36
4.1.6. Matplotlib .....	37
4.2. Serie temporal univariable .....	38
4.2.1. Análisis de ventana deslizante.....	38
4.2.2. Obtención de los datos .....	39

4.2.3.	Formateo de los datos .....	40
4.2.4.	Preparación para el entrenamiento .....	41
4.2.4.1.	Datasets de entrenamiento y testeo .....	41
4.2.4.2.	Hiperparámetros .....	42
4.2.4.3.	Callbacks .....	44
4.2.5.	Creación y comparación de modelos .....	45
4.2.5.1.	Perceptrón multicapa .....	45
4.2.5.2.	Red neuronal convolucional .....	45
4.2.5.3.	Red neuronal recurrente .....	46
4.2.5.4.	Algoritmo N-BEATS .....	46
4.2.5.5.	Comparación de los modelos .....	48
4.3.	Serie temporal multivariable.....	49
4.3.1.	Obtención de los datos .....	49
4.3.2.	Formateo de los datos .....	50
4.3.2.1.	Valores perdidos .....	50
4.3.2.2.	Preprocesamiento de características .....	52
4.3.3.	Construcción del modelo .....	53
5.	Resultados.....	57
5.1.	Predicción de mercado .....	57
5.2.	Análisis multivariable .....	59
5.2.1.	Tratamiento del sobreentrenamiento .....	59
5.2.2.	Precisión del modelo .....	60
5.2.3.	Importancia de las variables .....	62
6.	Conclusiones .....	64
7.	Opiniones y desarrollos futuros.....	66
8.	Apéndices.....	67
9.	Bibliografía.....	68

# Índice de ilustraciones

Ilustración 1: Diagrama de desarrollo del proyecto. Elaboración propia. ....	8
Ilustración 2. Gráfica del coeficiente de relación entre dos partículas con distintos valores. Fuente: <a href="https://rpubs.com/camilamila/correlaciones">https://rpubs.com/camilamila/correlaciones</a> .....	13
Ilustración 3. Fórmula de emisión de unidades de bitcoin. Fuente: <a href="https://yourcryptolibrary.com/es/bitcoin/bitcoin-halving/">https://yourcryptolibrary.com/es/bitcoin/bitcoin-halving/</a> .....	18
Ilustración 4. Gráfico histórico del precio de Bitcoin en cada ciclo. Fuente: <a href="https://yourcryptolibrary.com/es/bitcoin/bitcoin-halving/">https://yourcryptolibrary.com/es/bitcoin/bitcoin-halving/</a> .....	18
Ilustración 5. Proceso KDD. Fuente: Manual de la asignatura Minería de datos, VIU, 2022 .....	20
Ilustración 6. Esquema de ciencia de datos e inteligencia artificial. Fuente: <a href="https://lotuslabs.medium.com/clarifying-ai-machine-learning-deep-learning-data-science-with-venn-diagrams-c94198faa063">https://lotuslabs.medium.com/clarifying-ai-machine-learning-deep-learning-data-science-with-venn-diagrams-c94198faa063</a> .....	21
Ilustración 7. Esquema de una capa simple de una red neuronal. Fuente: <a href="https://www.xpertup.com/blog/deep-learning/activation-function/">https://www.xpertup.com/blog/deep-learning/activation-function/</a> .....	23
Ilustración 8. Distintas funciones de activación con sus gráficos. Fuente: <a href="https://www.datasciencepreparation.com/blog/articles/what-is-an-activation-function-what-are-commonly-used-activation-functions/">https://www.datasciencepreparation.com/blog/articles/what-is-an-activation-function-what-are-commonly-used-activation-functions/</a> .....	24
Ilustración 9. Esquema simple de un perceptrón multicapa. Fuente: <a href="https://www.mdpi.com/2073-4425/10/7/553#">https://www.mdpi.com/2073-4425/10/7/553#</a> .....	25
Ilustración 10. Esquema del kernel de una red neuronal convolucional. Fuente: <a href="https://www.mdpi.com/2073-4425/10/7/553#">https://www.mdpi.com/2073-4425/10/7/553#</a> .....	26
Ilustración 11. Esquema de una red neuronal convolucional. Fuente: <a href="https://www.mdpi.com/2073-4425/10/7/553#">https://www.mdpi.com/2073-4425/10/7/553#</a> .....	27
Ilustración 12. Esquema de una red neuronal recurrente. Fuente: <a href="https://www.mdpi.com/2073-4425/10/7/553#">https://www.mdpi.com/2073-4425/10/7/553#</a> .....	27
Ilustración 13. Esquema de una capa LSTM. Fuente: <a href="https://www.slideshare.net/MehrnazFaraz/lstm-163514628">https://www.slideshare.net/MehrnazFaraz/lstm-163514628</a> .....	28
Ilustración 14: Fórmula del error medio absoluto. Fuente: <a href="https://medium.com/@polanitzer/the-minimum-mean-absolute-error-mae-challenge-928dc081f031">https://medium.com/@polanitzer/the-minimum-mean-absolute-error-mae-challenge-928dc081f031</a> .....	29
Ilustración 15. Esquema de matriz de confusión. Fuente: <a href="https://www.researchgate.net/figure/Confusion-matrix-showing-what-the-true-positive-TP-false-positive-FP-false-negative_fig7_336927771">https://www.researchgate.net/figure/Confusion-matrix-showing-what-the-true-positive-TP-false-positive-FP-false-negative_fig7_336927771</a> .....	30
Ilustración 16. Logotipo de Anaconda. Fuente: <a href="https://www.anaconda.com/">https://www.anaconda.com/</a> .....	33
Ilustración 17. Logotipo de Python. Fuente: <a href="https://www.python.org/">https://www.python.org/</a> .....	34
Ilustración 18. Logotipo de Jupyter. Fuente: <a href="https://www.jupyter.org/">https://www.jupyter.org/</a> .....	35
Ilustración 19. Logotipo de Pandas. Fuente: <a href="https://www.pandas.pydata.org/">https://www.pandas.pydata.org/</a> .....	36
Ilustración 20. Logotipo de Tensorflow. Fuente: <a href="https://www.tensorflow.org/">https://www.tensorflow.org/</a> .....	36
Ilustración 21. Logotipo de Keras. Fuente: <a href="https://keras.io/">https://keras.io/</a> .....	37
Ilustración 22. Logotipo de Matplotlib. Fuente: <a href="https://www.matplotlib.org/">https://www.matplotlib.org/</a> .....	38
Ilustración 23. Esquema de una ventana deslizante. Fuente: <a href="https://www.kaggle.com/code/cworsnup/backtesting-cross-validation-for-timeseries/notebook">https://www.kaggle.com/code/cworsnup/backtesting-cross-validation-for-timeseries/notebook</a> .....	39
Ilustración 24. Formateo de dataset de partida para análisis univariable. Elaboración propia .....	40
Ilustración 25. Creación de ventanas. Elaboración propia .....	40

Ilustración 26. DataFrame de precios en diario preparada para análisis univariable. Elaboración propia .....	41
Ilustración 27. Función de separación de datos en conjuntos de entrenamiento y test. Elaboración propia .....	42
Ilustración 28. Creación de perceptrón multicapa con TensorFlow. Elaboración propia .....	45
Ilustración 29. Creación de una red convolucional con TensorFlow. Elaboración propia .....	46
Ilustración 30. Creación de una red recurrente con TensorFlow. Elaboración propia ..	46
Ilustración 31. Esquema del algoritmo N-BEATS. Fuente: <a href="https://arxiv.org/pdf/1905.10437.pdf">https://arxiv.org/pdf/1905.10437.pdf</a> .....	47
Ilustración 32. Implementación del algoritmo N-BEATS. Elaboración propia .....	48
Ilustración 33. Comparación de error de modelos. Elaboración propia .....	49
Ilustración 34. Lectura y concatenación de datasets. Elaboración propia .....	50
Ilustración 35. Extracción de último minuto y obtención de datos complementarios. Elaboración propia .....	50
Ilustración 36. Creación de DataFrame con todas las fechas necesarias. Elaboración propia .....	51
Ilustración 37. Valores perdidos del conjunto de datos. Elaboración propia .....	51
Ilustración 38. DataFrame preparado para entrenamiento. Elaboración propia .....	53
Ilustración 39. Creación de ventanas multivariantes. Elaboración propia .....	53
Ilustración 40. Esquema de la red neuronal utilizada. Elaboración propia .....	54
Ilustración 41. Ejemplo de capas de agrupación de media y de máximo. Fuente: <a href="https://www.researchgate.net/figure/Example-of-max-pooling-and-average-pooling-operations-In-this-example-a-4x4-image-is_fig4_332092821">https://www.researchgate.net/figure/Example-of-max-pooling-and-average-pooling-operations-In-this-example-a-4x4-image-is_fig4_332092821</a> .....	55
Ilustración 42. Predicciones modelo N-BEATS sobre conjunto de datos de validación. Elaboración propia .....	57
Ilustración 43. Realización de la estructura de ventana para realizar predicciones en el futuro. Elaboración propia .....	58
Ilustración 44. Predicción de precios de bitcoin en el futuro con N-BEATS. Elaboración propia .....	58
Ilustración 45. Ejemplos de entrenamiento ineficiente, entrenamiento correcto y sobreentrenamiento. Fuente: <a href="https://wiki.loginom.ru/articles/overtraining.html">https://wiki.loginom.ru/articles/overtraining.html</a> .....	59
Ilustración 46. Matriz de confusión del modelo convolucional sobre los datos minutales. Elaboración propia .....	61
Ilustración 47. Matriz de confusión del modelo convolucional sobre los datos diarios. Elaboración propia .....	62
Ilustración 48. Importancia de valores. Elaboración propia. ....	63

# Resumen

En este momento, la predicción del curso de los activos del mercado financiero se lleva a cabo con distintos métodos. Estos incluyen tanto el análisis fundamental, el cual permite divagar sobre nuevos cambios en el mercado mediante el estudio del campo de noticias; como el técnico, en el que los medios estadísticos y matemáticos sirven como herramientas principales. Muchos operadores profesionales utilizan sus habilidades para predecir el curso de una determinada divisa o acción con la búsqueda de los patrones más similares (modelos de situaciones repetitivas en el mercado), utilizando varios indicadores. Además, un buen indicador también puede llegar a ser el precio de otros activos de mercado. Esto ocurre con un alto nivel de dependencia (correlación) de los cursos. Algunos analistas de mercado utilizan tecnologías modernas para ayudar en la toma de decisiones de inversión. Estas tecnologías incluyen redes neuronales artificiales, las cuales, diseñadas y entrenadas adecuadamente, pueden convertirse en una excelente herramienta para la predicción de precios y la elaboración de estrategias.

En el presente trabajo, dado un conjunto de datos de mercado de distintos activos criptográficos, obtenido mediante técnicas de *web scraping*, usando la librería *requests* y considerando factores externos, se realizan modelos predictivos para predecir el precio de uno de estos activos criptográficos en un corto o medio plazo, así como la probabilidad de movimientos al alza o a la baja.

Dentro de los factores externos se han tenido en cuenta la emisión monetaria de la divisa en cuestión y los eventos que influyen a la inflación de la misma, promedios móviles de distintos periodos, la variación diaria y la variación máxima.

Además, para realizar el análisis mencionado, se han utilizado diferentes técnicas y herramientas estudiadas en el Máster de *Big Data* y *Data Science* de la VIU. Algunas de las herramientas utilizadas han sido: *Python*, *Anaconda*, *Tensorflow*, *Matplotlib*, *Jupyter Notebook*.

Los resultados obtenidos por los distintos modelos (secuencial, recurrente, convolucional y N-BEATS) mostraron que esta última ofrece un mejor desempeño en el análisis diario de ventana deslizante, mientras que para el análisis binario se elaboró un modelo mixto el cual mostró una precisión del 65.58% en diario y de un 52.21% con un *dataset* minutal.

# 1. Introducción

La criptomoneda es un sistema de pago digital en el que los bancos no participan en la verificación de transacciones. Es un sistema de igual a igual que permite a cualquier usuario, en cualquier lugar, enviar y recibir pagos. Los pagos de criptomonedas existen exclusivamente de forma digital en una base de datos en línea que describe transacciones específicas y en la cual es imposible modificar los registros ya añadidos. No implican transacciones con dinero físico, el cual cuenta con circulación y oportunidades de intercambio en el mundo real. Al transferir fondos en una criptomoneda, las transacciones se registran en un registro público. Estas monedas se almacenan en billeteras digitales.

El término “criptomoneda” entró en uso debido al hecho de que se utiliza el cifrado (criptografía) para verificar las transacciones: se utiliza una codificación avanzada para almacenar y transferir datos sobre la criptomoneda entre las billeteras y en los registros públicos. El objetivo del cifrado es proporcionar confiabilidad y seguridad.

La primera criptomoneda fue Bitcoin, creada en 2009 y la más conocida hasta la fecha. El comercio de criptomonedas es interesante desde el punto de vista de las ganancias, ya que como resultado de las acciones especulativas, se observan periódicamente movimientos muy bruscos en los precios de las mismas.

La predicción del mercado de valores es un intento de determinar el valor futuro de las acciones de una empresa u otro instrumento financiero que se negocia en una bolsa de valores. Predecir con éxito el precio futuro de distintos activos puede generar ganancias significativas. La hipótesis del mercado eficiente asume que los precios de las acciones reflejan toda la información disponible actualmente, y cualquier cambio en los precios que no se base en la información revelada recientemente es inherentemente impredecible. Sin embargo, otros autores no están de acuerdo, y aquellos con este punto de vista poseen innumerables métodos y tecnologías que supuestamente les permiten obtener información sobre precios futuros.

Dentro de este grupo se dan distintos análisis, dependiendo de la temporalidad y las métricas usadas en los mismos. Por un lado, existen los análisis fundamentales, realizados generalmente a largo plazo y con la intención de predecir tendencias en los precios de los distintos activos. Estos análisis se suelen basar en variables macroeconómicas y del propio nicho de mercado del activo a analizar.

Por otro lado, existen los análisis técnicos. Este tipo de análisis se basa en interpretar las variaciones en el precio de los activos con el fin de encontrar distintos patrones los cuales puedan ayudar a predecir estas mismas variaciones de precio.

Los análisis de mercado con redes neuronales artificiales en parte entran en este segundo grupo, ya que se basan en el uso de algoritmos de aprendizaje automático con el fin de detectar estos mismos patrones en las variaciones del precio con el fin de reconocerlos en análisis futuros para poder dar una proyección en el precio a ese periodo.

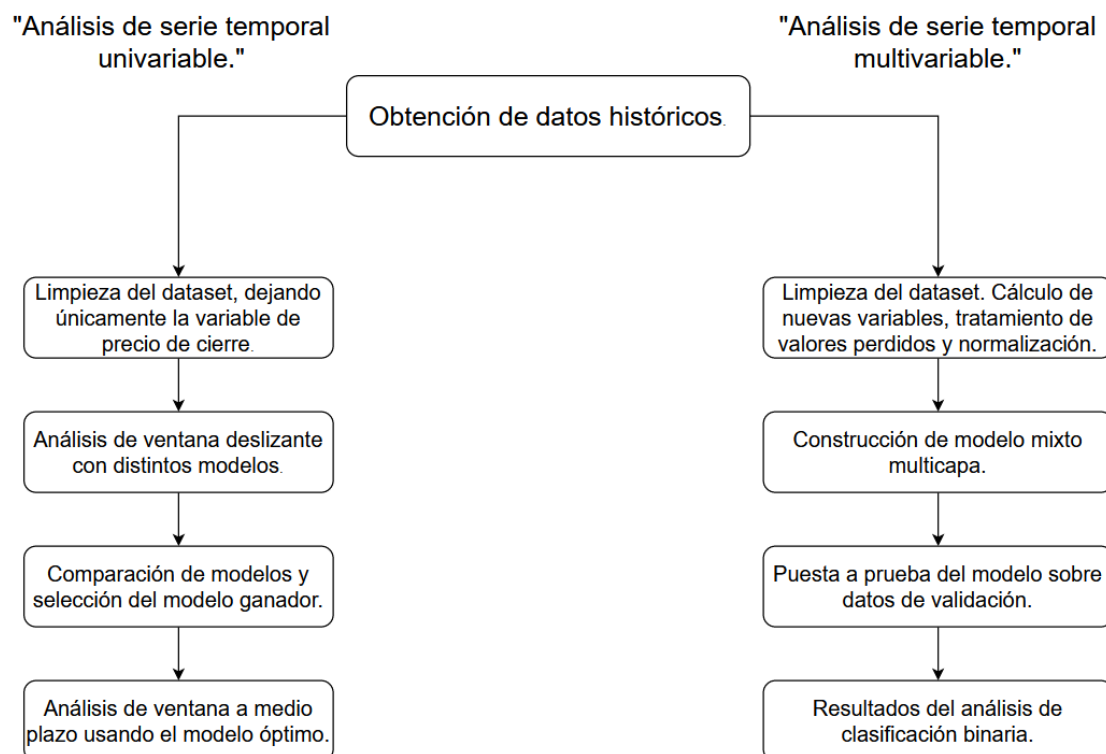
La novedad más importante es que estas redes neuronales incluyen en sus análisis variables fundamentales, incluyendo las de carácter macroeconómico, lo cual permite llevar el análisis técnico a otro nivel gracias a esta nueva perspectiva.

A pesar de los pocos años de existencia de estos algoritmos, ya es posible encontrar un gran número de trabajos dedicados al análisis del mercado financiero con estos, gracias a las posibilidades que ofrecen.

Sin embargo, hay un número bastante menor de estos análisis enfocados al análisis del mercado de las criptomonedas. Existen distintos trabajos de elaboración de estrategias para este mercado gracias a algoritmos de aprendizaje automático, pero en ellos por regla general se usan redes neuronales simples, además de no usar todas las variables que influyen en este mercado. Además, todos estos análisis se basan en estrategias de compra-venta, mientras que el mundo de las criptomonedas, gracias al exponencial desarrollo de los últimos años, ofrece otras muchas posibilidades a la hora de obtener beneficios del mismo.

En el presente Trabajo de Fin de Máster (TFM), fue realizada una investigación, comparando distintos modelos, en la cual se obtuvo un modelo ganador según distintas métricas.

El desarrollo del proyecto se ha llevado a cabo según el diagrama mostrado en la siguiente ilustración:



*Ilustración 1: Diagrama de desarrollo del proyecto. Elaboración propia.*



En primer lugar, se llevó a cabo la extracción de los datos históricos de distintas métricas de Bitcoin (precios de apertura y cierre, precio mayor, precio menor, volumen, capitalización de mercado). El proceso de análisis de los mismos se dividió en dos partes:

1. Por un lado, se procesaron los datos de tal forma que el *dataset* final únicamente tuviese una variable (el precio de cierre). Esta parte del análisis ha sido denominada “análisis de serie temporal univariable”. Como su propio nombre indica, en este apartado se realizó una comparación de modelos sobre un conjunto de datos de una única variable (el precio de cierre). Los modelos usados fueron diseñados para realizar un análisis de ventana deslizante, y fueron comparados con la métrica *mae* (*mean absolute error*, error medio absoluto). El problema a resolver fue un problema de regresión, donde la variable objetivo era el precio del día siguiente.
2. Por otro lado, se procesaron los datos de tal forma que se añadieron distintas variables al conjunto de datos original, tales como el tamaño de la recompensa por bloque, el número de días hasta el siguiente *halving*, la volatilidad o el oscilador estocástico. Esta parte del análisis fue denominada “análisis de serie temporal multivariable”, ya que en este caso se analizaba un *dataset* con distintas variables. En este caso, en el conjunto de datos utilizado la información histórica de Bitcoin tenía entradas cada 5 minutos. En este apartado, se creó un modelo mixto para este tipo de análisis, en el cual se resolvía un problema de clasificación binaria, donde la variable objetivo se encargaba de definir si el precio de Bitcoin tendría un movimiento al alza o a la baja en el siguiente periodo.

## 2. Objetivos

Los principales objetivos hacia los que se ha orientado el presente Trabajo de Fin de Máster son los siguientes:

### 2.1. Evaluación de modelos

Gracias a las herramientas proporcionadas por las librerías de aprendizaje profundo del lenguaje de programación *Python*, existe una infinidad de posibilidades a la hora de crear modelos de redes neuronales. Esto resulta ser un arma de doble filo, puesto que a pesar de contar con una gran flexibilidad para adaptarse a distintos problemas, resulta complicado dar con el modelo adecuado para cada caso concreto.

Por este motivo en el presente Trabajo de Fin de Máster se hizo énfasis en la diversidad de modelos, con lo que fueron evaluados distintos tipos de estos con distintos elementos con el objetivo de dar con el modelo que más se ajuste a cada uno de los distintos problemas presentados.

### 2.2. Conocimiento de factores de mercado

La volatilidad en el precio de las criptomonedas mencionada en el punto anterior es justamente uno de los motivos por lo que la mayoría de los usuarios poseedores de este tipo de activos lo hacen con fines especulativos, pues a mayor variación de precio en menor periodo de tiempo, mayor la rentabilidad de las operaciones de compra-venta a corto plazo.

Sin embargo, y debido a que se trata de un mercado relativamente nuevo, se desconocen con detalle los factores que afectan al precio del mismo. Así pues, con el fin de reconocer estos patrones, es necesario realizar un análisis estadístico.

En el transcurso del mencionado análisis en este Trabajo de Fin de Máster, se comparan distintos modelos de redes neuronales con el fin de construir un modelo con una precisión de más del 50%. Una vez entrenado y puesto a prueba tal modelo, es posible observar las variables más influyentes gracias a los pesos entrenados resultantes.

## 3. Estado del arte y Marco teórico

### 3.1. El mercado de las criptodivisas

#### 3.1.1. Definición de criptomoneda

Las criptomonedas son un sistema de pago descentralizado, es decir, no dependiente de ninguna entidad centralizada, tales como bancos o gobiernos. Esto se consigue gracias a un sistema de registro de transacciones basado en algoritmos *peer to peer* y encriptado gracias a herramientas de criptografía.

En el presente Trabajo de Fin de Máster la criptomoneda analizada es *Bitcoin*, por ser esta la primera criptomoneda y la de mayor capitalización de mercado a día de hoy.

Cabe destacar que justamente la descentralización es la característica más importante de las criptomonedas, y no el hecho de que sean digitales, a pesar de las creencias populares.

Cuando se realiza una transferencia en un sistema digital centralizado, es necesaria la interferencia de una entidad en la que todos los participantes de la transacción depositen su confianza. Esta entidad es la encargada de verificar que el usuario que realiza la transacción tiene fondos suficientes, para posteriormente sustraer estos fondos y enviarlos a la cuenta de destino.

Aparte de los problemas de confianza, esto supone un problema de privacidad, dado que en cada transacción se registran los datos personales de cada uno de los participantes.

Otro de los problemas de este sistema es el de la previsión de la inflación, ya que en el sistema fiduciario la emisión de moneda que desemboca en la devaluación de la misma es controlada por los organismos burocráticos de cada país, los cuales han demostrado su ineficiencia en este ámbito.

Las criptomonedas fueron creadas con el objetivo de solucionar estos tres problemas.

El mecanismo de registro de transacciones de forma descentralizada protegido con algoritmos de minería, además de ser inquebrantable, pone solución al primer problema mencionado, ya que la seguridad de las transacciones está sustentada por la red de minado de *Bitcoin*, la cual es a día de hoy la red con mayor poder computacional de la historia.

Cada dirección de estas monedas consiste en una combinación alfanumérica de decenas de caracteres sin ningún tipo de dato personal, por lo que el problema de privacidad queda resuelto. Además, la emisión de estas monedas queda predefinida en el momento de su creación, por lo que es posible predecir la emisión de esta en cada instante de tiempo, así como la inflación que esto supondría en el futuro.

### 3.1.2. Características del mercado financiero

El conjunto de todos los valores obtenidos a intervalos de tiempo iguales durante un período determinado se denomina serie temporal (Anatolyev, 2013). El análisis de estas series es el método más simple para restaurar la dependencia, basado en la serie temporal seleccionada. A partir de los objetivos principales del estudio de esta serie, se puede distinguir la descripción de las características que caracterizan la serie temporal, la elaboración de un modelo matemático que explique el comportamiento de la serie, y la predicción de valores futuros basados en observaciones pasadas (Afanasiev, 2010). Debido a la escasez de datos de referencia (por ejemplo, la corta duración de las observaciones), no siempre es posible alcanzar los objetivos establecidos

El análisis de series temporales puede llevarse a cabo por varios métodos. Los más comunes son:

- Análisis espectral: permite encontrar componentes periódicos en una serie temporal.
- Autorregresión: diseñada para ser la investigación en una serie temporal de un componente aleatorio (Bolshakov, 2014).
- Suavizado y filtrado: necesario para la conversión de series temporales, permiten deshacerse de las oscilaciones estacionales y de alta frecuencia. Esta técnica de detección de tendencia general es una de las más simples, consistente en una ampliación del intervalo de una serie dinámica.
- Método del cálculo de promedios móviles necesario para detectar tendencias. Con períodos pequeños, este método no funciona, y con grandes, se producen pérdidas significativas de datos en los extremos del intervalo.

Sobre la base de los datos obtenidos, mediante el análisis de la serie temporal, es posible construir un modelo predictivo para predicciones posteriores. Para hacer esto, se deben usar varios métodos de predicción. Actualmente, existe una gran cantidad de estos, pero generalmente se usan unos pocos en la práctica. Los más populares son los siguientes:

- Método normativo. Su esencia radica en los estudios de factibilidad de las predicciones utilizando distintas normas y regulaciones (Devyatov, 2008).
- Método de modelación. El pronóstico con modelos incluye su desarrollo, análisis experimental, comparación de los resultados de los cálculos predictivos preliminares con los datos reales del estado del proceso u objeto, refinamiento y ajuste del modelo (Chuchueva, 2012).
- Método de evaluación experta. Implica que el pronóstico representará la opinión de uno o un grupo de expertos basados en la experiencia científica, práctica y profesional (Itsjoki, 2006).
- Método de extrapolación. Este método consiste en el análisis de tendencias de desarrollo pasadas y presentes, y trasladarlas a una perspectiva futura. Este método es más común que los anteriores. Es solo un análisis matemático y contiene una gran cantidad de submétodos.

Además de los métodos descritos anteriormente para analizar series temporales, existe la variable de correlación. Bajo la dependencia de correlación se entiende la relación estadística de varias variables aleatorias. Además, los cambios en los valores de una (o varias) variables acompañan a los cambios sistemáticos en otra cantidad. En situaciones en las que el cambio de una magnitud no conduce a cambios naturales en la otra pero conduce a un cambio en la característica estadística de la misma variable aleatoria, la relación no puede considerarse correlativa (Brillinger, 1981).

La cuantificación de la relación entre dos variables aleatorias se puede realizar utilizando el coeficiente de correlación.

Este coeficiente se calcula como la suma de la multiplicación de la diferencia de una característica concreta de cada elemento y su media por la diferencia de la característica con la cual se quiere medir la correlación de cada elemento y su media, todo ello dividido por la raíz cuadrada de la multiplicación de las diferencias mencionadas anteriormente al cuadrado.

Este coeficiente de correlación puede tomar valores entre -1 y 1.

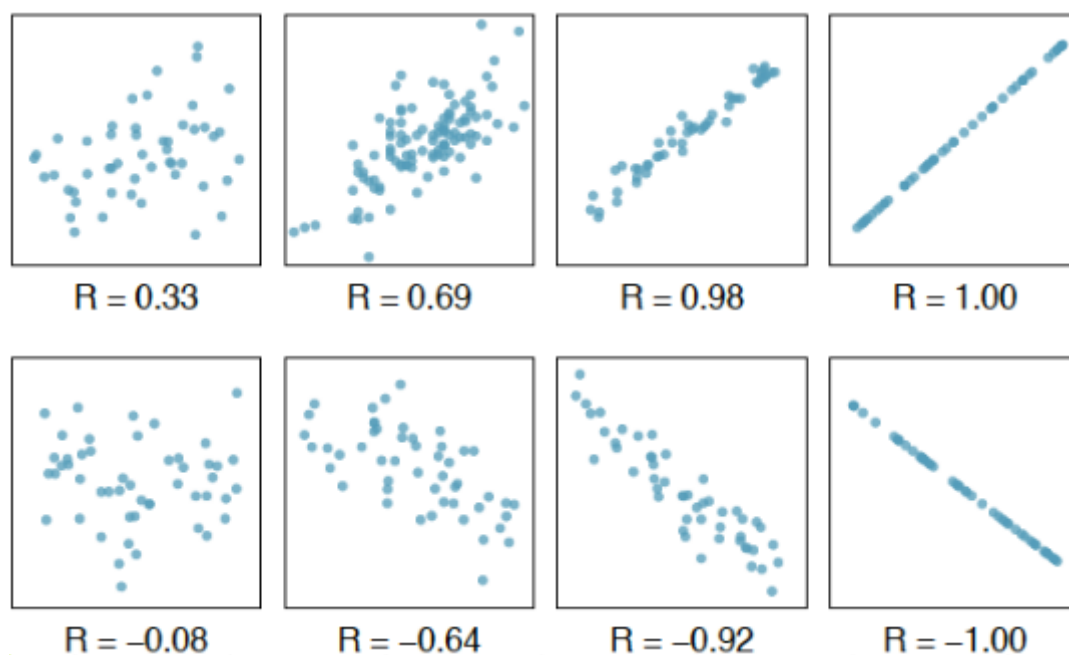


Ilustración 2. Gráfica del coeficiente de relación entre dos partículas con distintos valores. Fuente: <https://rpubs.com/camilamila/correlaciones>

Algunas de las principales propiedades de este coeficiente son las siguientes:

- En el caso de una relación positiva (directa), el coeficiente de correlación recibe un signo positivo y su valor se encuentra dentro del rango de 0 a +1. En otras palabras, si los valores de un rasgo aumentan y los valores de otro aumentan con él, entonces  $0 < r < 1$ .
- En el caso de una relación negativa (inversa), el coeficiente de correlación recibe un signo negativo y su valor se encuentra dentro de -1 a 0. En otras

palabras, si los valores de un rasgo aumentan y los valores de otro disminuyen, entonces  $-1 < r < 0$ .

- Cuanto más cerca esté el valor del coeficiente de correlación a uno, más fuerte será la relación entre los rasgos. La relación entre los rasgos se vuelve funcional con un coeficiente de correlación de  $\pm 1$ . La conectividad funcional ocurre cuando cualquier exponente del rasgo X caracterizará uno o más exponentes específicos del rasgo Y.

Para un análisis exitoso, es necesario comprender el flujo de información diferente y poder filtrar las noticias económicas más significativas y las previsiones del mercado. Dependiendo de la velocidad y el volumen de las operaciones realizadas, se distinguen dos métodos: análisis fundamental y técnico. El análisis fundamental se basa en la evaluación de las situaciones en términos de política económica, política y financiera y crediticia. El técnico, a su vez, se basa en modelos, así como en métodos de análisis gráfico e investigación, que se basa en principios matemáticos.

El análisis fundamental analiza los factores que afectan el valor cambiario de un valor. Este análisis se basa en los estados financieros de la empresa emisora, así como en el cálculo de los indicadores financieros que caracterizan su actividad. Teniendo en cuenta los objetivos, el pronóstico del mercado se divide en 3 niveles: nivel macro, meso y micro.

El nivel macro representa el estado económico del estado en su conjunto. También se pueden estudiar mercados internacionales para seleccionar países. En este caso, se analizan los efectos de los factores económicos y políticos en el mercado.

Los principales indicadores para este nivel son:

- Volumen del PIB. El crecimiento de este indicador indica un aumento en las ventas en las empresas.
- Desempleo e inflación. Estas cifras comienzan a aumentar en tiempos de crisis.
- Las tasas de interés para las empresas. Cuanto mayor sea este indicador, menos desarrollado será el mercado.
- Tipo de cambio. Este indicador es seguido por el mercado de valores.

Estos indicadores pueden considerarse un reflejo de la economía. Cuanto más cerca estén del valor ideal, mejor será la situación económica del país y, en este caso, el mercado se vuelve más atractivo para los inversores de diferentes países del mundo.

En la mesoescala, es posible identificar las industrias más exitosas de la economía del país con la ayuda de índices bursátiles, como, por ejemplo, los índices *Dow Jones*, o *Standard and Poor's* los cuales muestran la dinámica del desarrollo de varios sectores de la economía.

El nivel micro es un análisis de una empresa o ente financiero en particular: el estudio de la estructura corporativa de la empresa, el entorno competitivo, así como los factores productivos y financieros y económicos (Weatherall, 2014). De los más importantes se pueden destacar:

- Flujo de capital. Se contabilizan todos los pagos e ingresos.
- Liquidez. Cuanto más efectivo exista para pagar posibles deudas, mejor será la situación.
- Rentabilidad. Cuanto mayor sea el rendimiento de la empresa, según las métricas de ROA y ROE, mejor.
- Deuda. Si el negocio de la compañía se basa en préstamos permanentes, entonces es muy probable que en algún momento no haya disponible suficiente dinero para pagarlos.

Además de estos análisis, se puede atribuir a un enfoque fundamental al análisis de la actividad social. Las emociones, junto con la información, juegan un papel importante para una persona en la toma de decisiones. En cuanto a los mercados de valores y criptomonedas, se puede decir que las decisiones financieras están impulsadas en gran medida por los estados de ánimo y las emociones. Es por eso que las noticias, el ruido informativo y el estado de ánimo de los inversores pueden considerarse estimulantes de las fluctuaciones de mercado a corto plazo. Como regla general, estas fluctuaciones son insignificantes (desde unos pocos centavos por día hacia arriba o hacia abajo), pero esto es suficiente para el comercio especulativo diario en la bolsa de valores. Sin embargo, para hacer esto, es necesario seguir constantemente las noticias y responder rápidamente a las señales de noticias que aparecen. Además, es necesario evaluar correctamente el impacto en el curso tendrá una noticia en particular y cómo reaccionará la mayoría de los jugadores del mercado. A veces, un poco de pánico puede afectar mucho el curso de una acción e incluso colapsarlo en un porcentaje.

La base del análisis técnico es la idea de las acciones de todas las fuerzas que afectan al mercado y se manifiestan en los niveles de precio del activo financiero y los indicadores del volumen de comercio, así como en los estados recurrentes del mercado. El objetivo del análisis técnico es determinar el momento del cambio en la tendencia del precio de un valor y responder a la pregunta de cuándo debe comprarse o venderse.

El análisis técnico de los mercados de valores y criptomonedas consiste en la construcción y el uso de tales indicadores, niveles de soporte y resistencia; como una línea de tendencia; osciladores técnicos; formas geométricas; indicadores técnicos de tendencia, etc.

Si la dirección del movimiento del precio es hacia arriba o hacia abajo, entonces este movimiento puede llamarse tendencia. En caso de que el movimiento del precio en la dirección lateral, la tendencia lateral. Cada tendencia tiene varios ciclos, como la nucleación, el desarrollo y la finalización de la tendencia. El comerciante se dedica a la búsqueda de signos de la aparición de la tendencia para entrar en el mercado en su dirección. A la primera señal de finalización de la tendencia, debe salir de la posición. La influencia de los factores fundamentales resulta en la tendencia a largo plazo. Se puede formar bajo la influencia de varias noticias importantes. La aparición de situaciones perdedoras se asocia, por regla general, con la entrada en una posición en contra de la dirección de la tendencia.

El nivel de soporte y resistencia se denomina línea, que durante un cierto tiempo tocó el precio varias veces y se defendió en la dirección opuesta. Si el precio todavía rompe la



línea de soporte o resistencia, entonces esta línea cambia y se convierte en el nivel de resistencia o soporte respectivamente.

Además de los niveles y la tendencia, el análisis técnico utiliza figuras como: rectángulo, triángulo, bandera, banderín, cabeza y hombros, doble/triple superior/inferior y otros. Todas las formas se pueden dividir en dos grupos principales: formas de continuación de la tendencia y formas de inversión. Las cifras de continuación de la tendencia indican al comerciante la probabilidad de continuar en una dirección determinada, y las cifras de reversión muestran la posibilidad de su reversión. La mayoría de las veces, este método se usa junto con otros medios de análisis.

Los osciladores identifican zonas de sobrecompra y sobreventa basadas en mediciones del volumen del mercado. En tales zonas, la probabilidad de una reversión de la tendencia es alta, especialmente cuando se mueve dentro del canal.

Para que el activo adquirido produzca el máximo rendimiento posible con riesgos mínimos, es necesario prestar atención a los siguientes parámetros de liquidez y volatilidad (Graham, 2008).

La liquidez ofrece la oportunidad de abrir de forma rápida y eficiente una posición de cualquier volumen sin un cambio significativo en el precio del activo. En otras palabras, determina la facilidad con que un comerciante puede comprar o vender cualquier valor. La liquidez se mide por los siguientes parámetros:

Profundidad del mercado, es decir, el número de órdenes que están detrás de los mejores precios de oferta y demanda (BID y ASK).

El ancho del mercado caracteriza la diferencia de precio de oferta y demanda.

La elasticidad indica el tiempo necesario para que el mercado se recupere después de la ejecución de una orden grande.

Otro indicador estadístico importante es la volatilidad, que muestra el rango de variabilidad del precio de alto a bajo en el proceso de negociación. Este indicador es muy importante en la gestión de riesgos. Cuanto mayor sea la volatilidad del mercado, mayores serán los riesgos en el comercio.

### 3.1.3. Características intrínsecas del mercado criptográfico

A pesar de que las características explicadas en el punto anterior son aplicables a la inmensa mayoría de activos financieros, las criptomonedas poseen ciertas características las cuales pueden explicar movimientos bruscos de mercado.

La criptomoneda es un tipo de moneda digital, cuya creación se basa en métodos de criptografía. Este término apareció después de un artículo titulado "*Crypto currency*" (moneda criptográfica en español), publicado en la revista *Forbes*. La principal característica de la criptomoneda consiste en la ausencia de administradores internos o



externos. Las autoridades fiscales, bancarias, públicas y otras autoridades privadas o judiciales no pueden influir en las transacciones de los participantes en este sistema de pago (Nakamoto, 2008). El envío de criptomonedas es irreversible. En otras palabras, nadie podrá impugnar, bloquear, cancelar o forzar una transacción. Este hecho fue una razón importante para la aparición de una nueva forma de atracción de capital para las organizaciones.

El mercado de criptomonedas apareció en 2010. En este momento, incluye muchas monedas digitales diferentes. La más popular es Bitcoin. A lo largo de la historia del mercado, siempre ha ocupado una posición de liderazgo en el nivel de capitalización de mercado total. Otras criptomonedas (llamadas también *altcoins*) no han tenido tiempo de ganar fama de esta escala, sin embargo, atraen mucha atención y son populares entre los inversores y comerciantes.

En cuanto a las diferencias con el dinero electrónico clásico, las monedas digitales no tienen análogos en el mundo físico, ya que son un conjunto de datos cifrados que se almacenan en la red *blockchain*. Una cadena de bloques es una secuencia continua de bloques en una cadena, construida de acuerdo con ciertas reglas y que contiene información. Por lo general, las copias de todas las cadenas de bloques se almacenan de forma independiente en una gran cantidad de computadoras diferentes. Por primera vez, este término apareció como el nombre de una base de datos replicada completamente distribuida implementada en el sistema de criptomoneda *bitcoin*, por lo que la tecnología *blockchain* generalmente se refiere a transacciones de varias monedas digitales, pero puede distribuirse entre cualquier bloque de información interconectado.

El rápido crecimiento del interés en las criptomonedas surgió después de la difusión de la tecnología *blockchain*. En su trabajo esta tecnología comenzó a ser utilizada por las grandes corporaciones mundiales, como, por ejemplo, *Amazon*. Esta tecnología permitió transferir y cifrar datos de manera confiable dentro del sistema.

Este mercado está prácticamente dominado por la primera y por tanto principal criptomoneda, el *bitcoin*. Es por ello que cualquier movimiento brusco de esta criptomoneda mueve todo el mercado criptográfico tras ella. Por esta razón, en el presente Trabajo de Fin de Máster esta criptomoneda tuvo el papel protagonista, ya que analizando esta es posible extrapolar dicho análisis a otra criptomoneda.

La emisión de esta moneda se rige por unas estrictas reglas. Cada diez minutos, al minarse un bloque, un afortunado minero recibe una recompensa en *bitcoins*, por lo que la cantidad circulante de esta aumenta cada diez minutos en dicha cantidad. Para mantener un carácter deflacionario, esta cantidad se reduce a la mitad cada cierto tiempo, lo cual permite describir la cantidad de monedas circulantes en un sumatorio para el cual existe un límite cuando el tiempo tiende a infinito.

El número de monedas circulantes se describe con la siguiente fórmula:

$$\sum_{i=0}^{32} 210,000 \left( \frac{50}{2^i} \right)$$

total # of halvings to ever occur

# of new bitcoins issued per block

# of blocks between halvings

cumulative # of halvings so far

@anilsaidso

Ilustración 3. Fórmula de emisión de unidades de bitcoin. Fuente: <https://yourcryptolibrary.com/es/bitcoin/bitcoin-halving/>

Produciéndose esta reducción de la oferta cada 210.000 bloques y siendo la recompensa inicial 50 bitcoins, este sumatorio cuando “i” tiende al infinito da un resultado máximo de 21 millones, los cuales son el máximo teórico de unidades de bitcoin por haber en toda la historia.

El *halving* supone una disminución de la oferta, lo cual históricamente ha producido movimientos bruscos en el precio, como se puede apreciar en el siguiente gráfico:

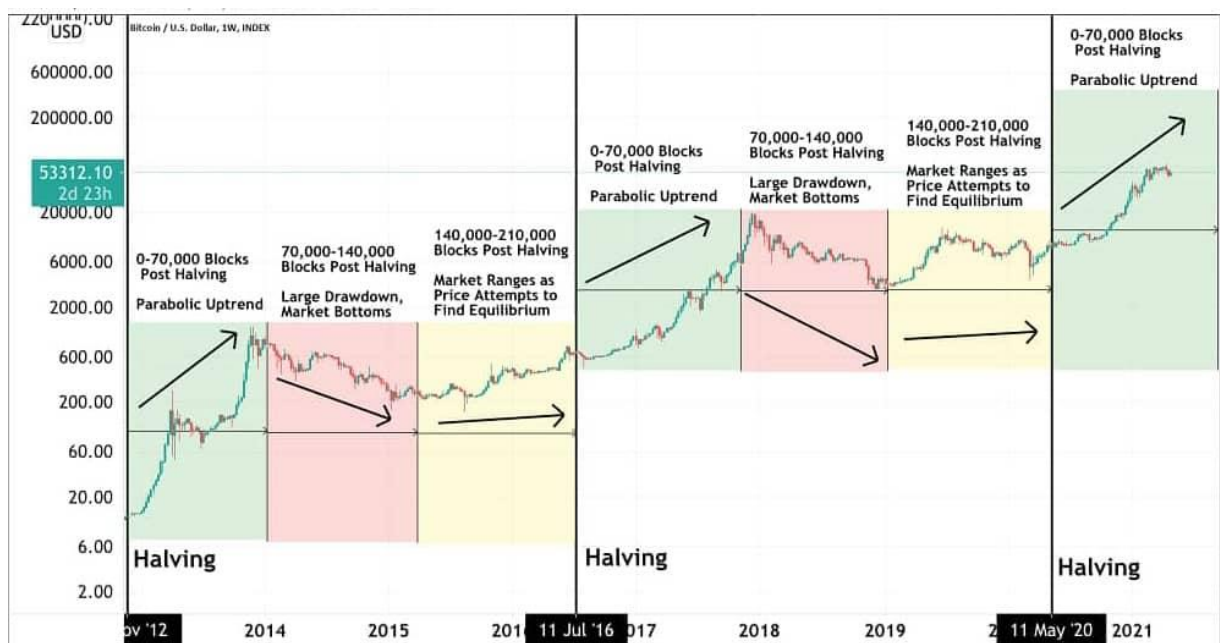


Ilustración 4. Gráfico histórico del precio de Bitcoin en cada ciclo. Fuente: <https://yourcryptolibrary.com/es/bitcoin/bitcoin-halving/>

Se puede observar que, históricamente, tras cada *halving* se ha producido una etapa de crecimiento acelerado en el precio de *bitcoin*, seguido de un periodo de consolidación. Esto son los llamados ciclos de *bitcoin*, los cuales son un factor determinante en el precio.

En los siguientes apartados se explicarán las técnicas utilizadas para realizar este trabajo.

### 3.2. Proceso KDD

El proceso KDD (*Knowledge Discovery in Databases* por sus siglas en inglés) es el proceso de búsqueda, transformación y refinamiento de datos de una base de datos en crudo con el objetivo de identificar las relaciones entre los datos y distintos patrones que no son apreciables a simple vista.

En otras palabras, consiste en la extracción de conocimiento de los datos (Maimon & Rokach, 2005). Este proceso está reconocido en el área de investigación debido a su importancia a la hora de entender las bases de datos y a la necesidad de obtener información y conocimiento de unos datos cada vez más numerosos.

Los patrones encontrados en los datos disponibles deben ser correctos para cualquier dato de dominio nuevo con cierto grado de confianza. Además, los patrones deberían haber sido previamente desconocidos y potencialmente útiles, es decir, permitir algún beneficio al resolver un problema específico. Por último, los patrones deben ser comprensibles e interpretables, si no de inmediato, después de un pequeño procesamiento posterior.

KDD se formó y evolucionó como una dirección interdisciplinaria en la unión de tales disciplinas como:

- El aprendizaje automático
- Reconocimiento de patrones
- Bases de datos
- Estadística matemática
- Inteligencia Artificial
- Visualización
- Computación de alto rendimiento

El proceso KDD es interactivo e iterativo, y que contiene varios pasos en cada uno de los cuales el usuario puede tomar decisiones específicas.

Dichos pasos se pueden apreciar en la siguiente ilustración:



Ilustración 5. Proceso KDD. Fuente: Manual de la asignatura Minería de datos, VIU, 2022

Dichos consisten en:

1. Selección: comprender el área temática y obtener un conocimiento a priori sobre ella, formular las metas y objetivos del proceso KDD, así como crear un conjunto de datos objetivo en el que se buscarán las plantillas.
2. Preprocesamiento: incluye la limpieza de datos, la selección de datos para la construcción de modelos, la selección de métodos de procesamiento de saltos y duplicados, el procesamiento de series temporales, etc.
3. Transformación: incluye la reducción de la dimensionalidad de los datos, así como la determinación de la forma de su representación, la más óptima en términos del problema resuelto.
4. Minería de datos: los métodos y modelos analíticos son aplicados a los datos seleccionados y preparados para resolver problemas de clasificación y regresión, búsqueda de reglas asociativas y agrupación, y predicción para detectar patrones.
5. Interpretación — los patrones detectados se presentan como reglas decisivas y sus árboles, estructuras de clústeres, regresiones, etc. Es posible utilizar directamente el conocimiento obtenido de este proceso, combinado con el conocimiento obtenido de otros sistemas y áreas temáticas, aplicado para documentar y generar informes. También en esta etapa, se comprueba la presencia de posibles conflictos con los conocimientos adquiridos anteriormente y su resolución.

KDD no especifica qué métodos y algoritmos de procesamiento deben usarse para resolver una tarea en particular, sino que especifica la secuencia de acciones que debe realizar para obtener conocimiento de los datos originales. Este enfoque es universal e independiente del área temática.

Para apoyar los proyectos de KDD, a mediados de la década de 1990 fue desarrollado un estándar abierto de modelado de procesos que describe los enfoques comunes utilizados en la búsqueda de conocimiento en bases de datos, denominado CRISP-DM.

Los fundadores del concepto de KDD son Pyatetsky-Shapiro y Usama Fayyad.

### 3.3. Ciencia de datos

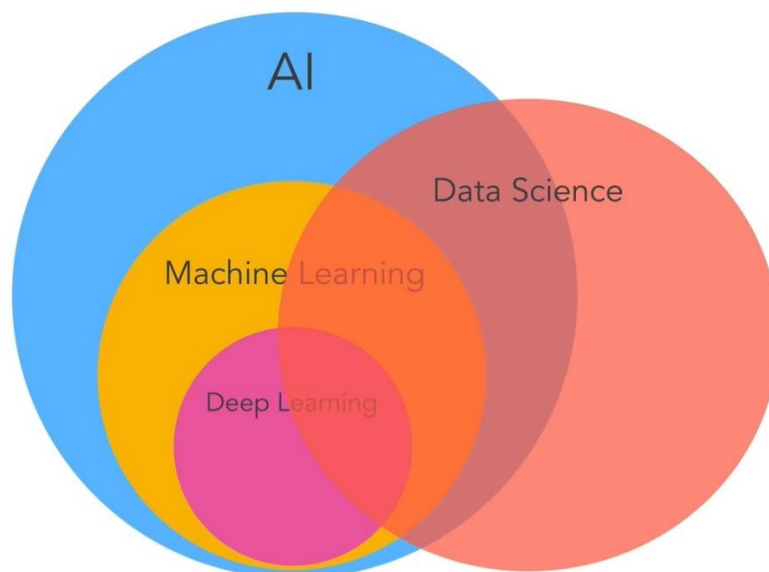


Ilustración 6. Esquema de ciencia de datos e inteligencia artificial. Fuente:  
<https://lotuslabs.medium.com/clarifying-ai-machine-learning-deep-learning-data-science-with-venn-diagrams-c94198faa063>

La ciencia de datos es una disciplina en la cual se aplican técnicas para procesar grandes cantidades de información (Shan, Chen, Wang, & Song, 2015). Construye y prueba modelos matemáticos de comportamiento de datos. Esto ayuda a encontrar patrones en ellos o predecir valores futuros. Por ejemplo, según los datos de demanda de productos en el pasado, un científico de datos ayudará a la compañía a predecir las ventas del próximo año. Los modelos se construyen con algoritmos de aprendizaje automático y las bases de datos se ejecutan a través de SQL.

Esta disciplina, a grandes rasgos, consiste en analizar datos para obtener conocimiento, normalmente para un objetivo de negocio.

Dentro de la misma, generalmente se usan algoritmos de inteligencia artificial para obtener este conocimiento. Es por ello que en la ilustración 9 se muestra la estrecha correlación entre estos campos.

### 3.4. Inteligencia artificial

La inteligencia artificial es un término usado para representar la inteligencia humana exhibida por ordenadores (Haykin, 2008).

La IA no es un formato ni una función, es un proceso y una habilidad para pensar y analizar datos. A pesar de que al mencionar la "inteligencia artificial", muchos imaginan

robots humanoides inteligentes que capturan el mundo, en realidad la IA no está diseñada para reemplazar a los humanos. Su objetivo es ampliar los límites de las capacidades y capacidades humanas. Por lo tanto, esta tecnología es un recurso empresarial valioso.

La IA ofrece la oportunidad de reproducir y mejorar la forma en que una persona percibe y reacciona al mundo que lo rodea. Lo que hace que la IA sea una poderosa piedra angular en la base de la teoría de la innovación. El uso de diferentes formas de aprendizaje automático que reconocen patrones en los datos que permiten la predicción y contribuye a aumentar el valor del negocio a través del aprovechamiento de todo el potencial de los datos y de la elaboración de predicciones fiables y la automatización de tareas complejas.

La tecnología se basa y ayuda a mejorar la eficiencia y la productividad mediante la automatización de los procesos y tareas que solían realizar los seres humanos. La IA también es capaz de interpretar cantidades de datos que no pueden ser interpretados por una persona. Esta habilidad puede traer beneficios sustanciales para las empresas. Por ejemplo, *Netflix* utiliza el aprendizaje automático para la personalización, lo que ayudó a aumentar la audiencia en un 25% para 2017.

La mayoría de las empresas han hecho del estudio de datos su prioridad e invierten mucho en él. Según un estudio reciente realizado por Gartner entre más de 3, 000 CEOs, los encuestados citaron el análisis de datos y la inteligencia empresarial como las principales tecnologías para tener éxito. Según los encuestados, estas tecnologías tienen la mayor importancia estratégica, por lo que representan la mayor cantidad de inversión.

### 3.5. Aprendizaje automático

El aprendizaje automático es un subconjunto de la inteligencia artificial, tal y como se puede observar en la ilustración 6. Esta disciplina consiste en una aproximación hacia la inteligencia artificial a través de modelos estadísticos que pueden reconocer patrones en un conjunto de datos (Géron, 2019).

La universidad de Standford describe al aprendizaje automático como la ciencia consistente en hacer funcionar a los ordenadores en una forma concreta sin necesidad de haber sido programados específicamente para cierta tarea.

Con la ayuda del aprendizaje automático, la IA puede analizar datos, recordar información, crear predicciones, reproducir modelos terminados y elegir la opción más adecuada de las propuestas.

Estos sistemas son especialmente útiles cuando es necesario realizar grandes cantidades de cálculos: por ejemplo, puntuación bancaria (cálculo de calificación crediticia), análisis de marketing y estudios estadísticos, planificación empresarial, estudios demográficos, inversiones, búsqueda de noticias falsas y sitios fraudulentos.



### 3.6. Aprendizaje profundo

El aprendizaje profundo es una disciplina que engloba técnicas de implementación de aprendizaje automático consistentes en la creación de modelos matemáticos basados en las redes neuronales biológicas (Géron, 2019).

El cerebro humano está formado por neuronas conectadas entre sí por sinapsis y que transmiten pulsos electroquímicos. Una red neuronal consiste en neuronas artificiales, elementos computacionales creados a partir del modelo de una neurona biológica.

Para la resolución de problemas de predicción del precio de distintos activos financieros en series temporales han sido ampliamente usadas en la última década las redes neuronales artificiales. Su uso en el campo de las finanzas se debe principalmente a la capacidad de trabajar con datos ruidosos y contradictorios.

Estas neuronas artificiales son básicamente celdas en las cuales se realizan operaciones matemáticas concretas.

En primer lugar, una serie de señales llega a la entrada de una neurona artificial, que, al mismo tiempo, es la salida de otra neurona  $x = (x_1, x_2, \dots, x_n)$ . Posteriormente, cada señal de entrada se multiplica por el peso correspondiente a su respectiva sinapsis -  $\omega_1, \omega_2, \dots, \omega_n$ , igual a la fuerza sináptica, y todos los productos resultantes se suman, para determinar el nivel de activación de la neurona, como se representa en el siguiente esquema:

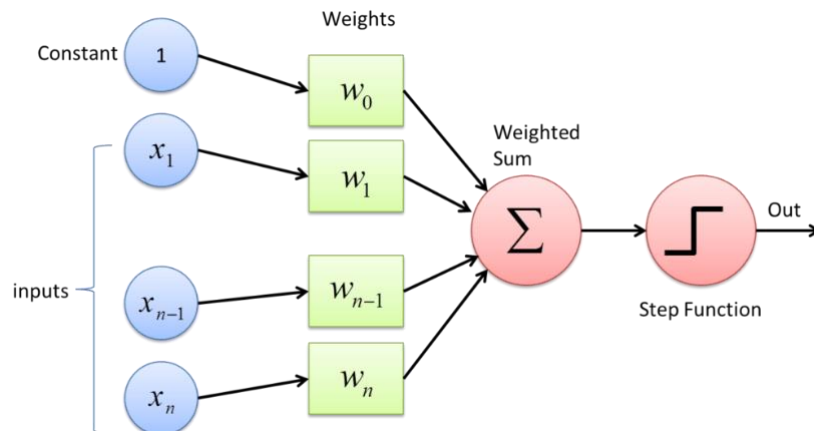


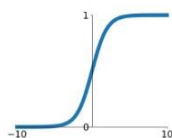
Ilustración 7. Esquema de una capa simple de una red neuronal. Fuente: <https://www.xpertup.com/blog/deep-learning/activation-function/>

Posteriormente, y con el objetivo de prevenir linealidad, se ejecuta una función (denominada función de activación) sobre el resultado de la suma, lo cual resulta en la salida de la neurona.

Existen distintas funciones de activación para distintos casos. Las más populares son las mostradas en la siguiente ilustración:

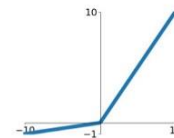
### Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



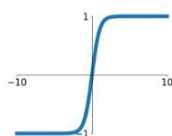
### Leaky ReLU

$$\max(0.1x, x)$$



### tanh

$$\tanh(x)$$

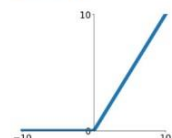


### Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

### ReLU

$$\max(0, x)$$



### ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

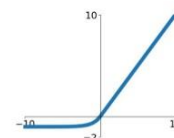


Ilustración 8. Distintas funciones de activación con sus gráficos. Fuente:

<https://www.datasciencepreparation.com/blog/articles/what-is-an-activation-function-what-are-commonly-used-activation-functions/>

Estas salidas pasan de una capa a otra hasta llegar a la capa de salida, donde se calcula el error con respecto a la muestra con una llamada función de pérdidas. Más adelante, y con el objetivo de determinar el peso de cada parámetro sináptico sobre el resultado, se toma el gradiente del resultado de esta función de pérdidas sobre cada uno de los parámetros sinápticos y estos se actualizan según cada resultado correspondiente, multiplicado por otro parámetro llamado la “tasa de aprendizaje”.

Este proceso de actualización de los pesos se denomina “proceso de entrenamiento”, y la técnica descrita es la técnica de retropropagación del error. Este proceso presupone el entrenamiento en numerosas iteraciones.

Al tratarse de un problema de regresión, en el presente Trabajo de Fin de Máster las funciones usadas fueron *ReLU* y *Leaky ReLU*, mientras que las funciones de pérdidas usadas fueron *mse* y *mase*.

## 3.6.1. Perceptrón multicapa

El perceptrón multicapa (o MLP por sus siglas en inglés, *Multi Layer Perceptron*) es un tipo de red donde todas las conexiones están estrictamente dirigidas desde las neuronas de entrada a salida, tal y como se muestra en la siguiente ilustración:



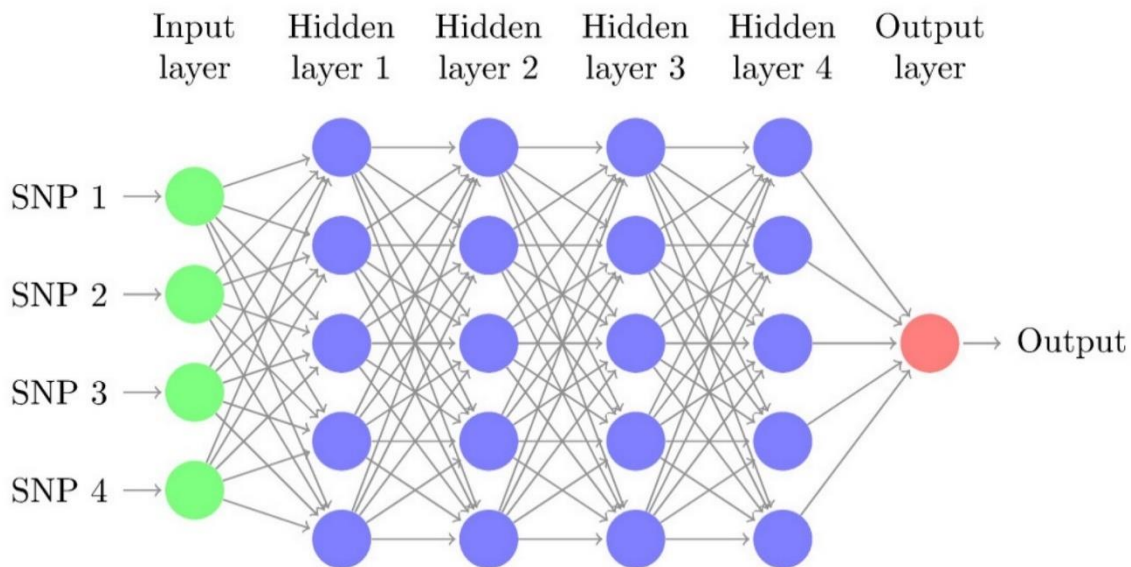


Ilustración 9. Esquema simple de un perceptrón multicapa. Fuente: <https://www.mdpi.com/2073-4425/10/7/553#>

El perceptrón multicapa es el modelo de red neuronal más simple, y el primero cuya efectividad fue comprobada en el transcurso de este Trabajo de Fin de Máster.

Este tipo de arquitectura se denomina multicapa debido a la presencia de una u varias capas ocultas adicionales además de las capas de entrada y salida. La señal en la red MLP se extiende desde la capa de entrada en la dirección directa a la oculta, y luego a la salida. Cada una de las neuronas de la capa anterior están conectadas a cada neurona de la capa oculta.

El MLP se refiere a dos tipos diferentes de perceptrones multicapa: Rosenblatt y Rumelhart (Wasserman, 1992). El primero tiene más de una capa que contiene elementos A, y el segundo también está sujeto a entrenamiento y comunicación S-A. Además, el aprendizaje en el perceptrón multicapa según Rumelhart se realiza mediante el método de propagación inversa del error. En el proceso de aprendizaje del peso de las neuronas de todas las capas de la red neuronal, este método se ajusta teniendo en cuenta las señales que provienen de la capa anterior.

### 3.6.2. Redes neuronales convolucionales

En una red con esta arquitectura, los datos no se procesan en su totalidad, sino en fragmentos, lo cual ayuda a mantener la información espacial. Cada uno de estos fragmentos de datos se denomina *kernel* (Lee, Grosse, Ranganath, & Ng, 2009).

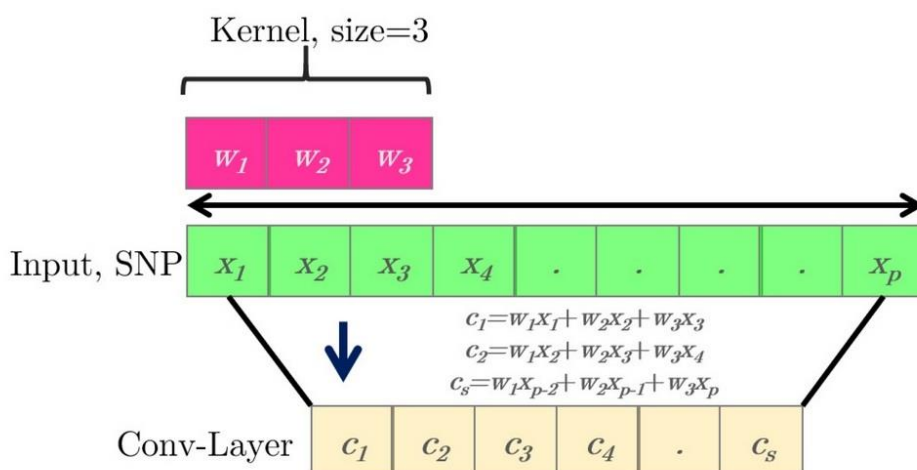


Ilustración 10. Esquema del kernel de una red neuronal convolucional. Fuente: <https://www.mdpi.com/2073-4425/10/7/553#>

Al mismo tiempo, no se dividen en partes, sino una especie de ejecución secuencial, después de lo cual los datos se transfieren más a lo largo de las capas. Además de las capas convolucionales, también se aplica una capa sub-muestreo (*pooling*). A las últimas capas ocultas se agregan varias capas más de enlace completo para procesar los datos de convolución. Cada una de estas capas realiza funciones específicas:

Una capa convolucional es un conjunto de mapas (matrices) que contienen núcleos sinápticos (Goodfellow, Bengio, & Courville, 2016). Tanto el número de tales mapas como su tamaño es determinado por los requisitos establecidos para la tarea. Si se toman muchas cartas, la precisión del reconocimiento mejorará, pero la complejidad del cálculo aumentará. El núcleo sináptico es una ventana o filtro que se desliza por toda el área de los datos estructurados. El núcleo, en este caso, es un sistema de sinapsis compartidas, o pesos, que es la característica principal de las redes neuronales convolucionales. En un MLP convencional, hay muchas sinapsis entre las neuronas, lo que ralentiza enormemente el proceso de detección en datos estructurados. En CNN, los pesos totales permiten reducir el número de enlaces y detectar en toda la región de datos el mismo rasgo.

La capa de muestreo (*pooling*) es utilizada para reducir la dimensionalidad de los mapas en la capa anterior. Si ya se han identificado signos en la capa anterior, entonces en el procesamiento posterior ya no se necesitarán datos detallados, lo que significa que se pueden compactar a menos detalles. Esta capa, al igual que la capa convolucional, tiene mapas que, en el proceso de escaneo del núcleo, no se cruzan entre sí.

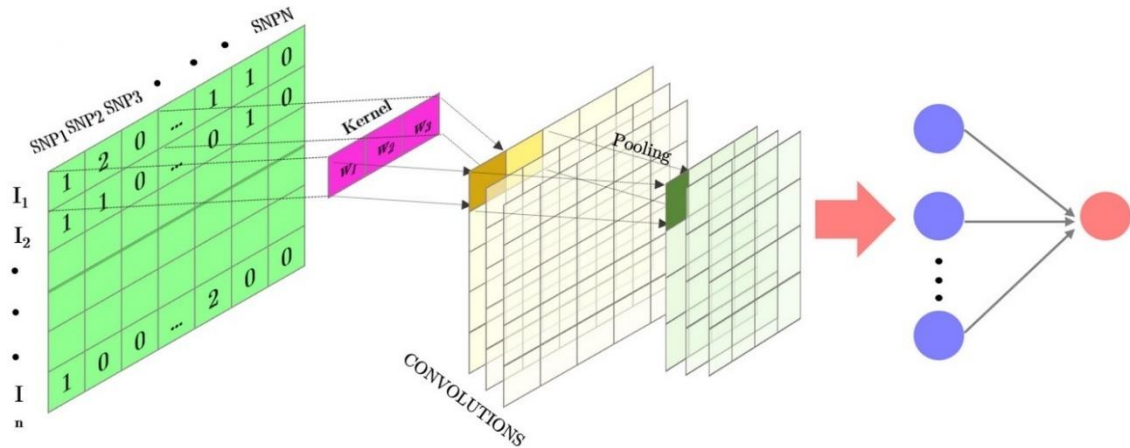


Ilustración 11. Esquema de una red neuronal convolucional. Fuente: <https://www.mdpi.com/2073-4425/10/7/553#>

En el presente Trabajo de Fin de Máster fue probado un modelo convolucional, cuyo *kernel* representaba la ventana de los días anteriores.

### 3.6.3. Redes neuronales recurrentes

Las redes neuronales recurrentes o redes de propagación inversa (RNN por sus siglas en inglés) son un tipo de redes neuronales diseñadas para reconocer en una secuencia de datos patrones, ya sean palabras habladas, texto o secuencias numéricas que provienen, por ejemplo, del mercado de valores (Hamme, 2001). La principal diferencia entre esta arquitectura y el resto, es la presencia de la llamada memoria. En redes similares de neuronas de capa oculta o neuronas de salida, la señal se transmite parcialmente a las entradas de las neuronas de la capa de entrada.

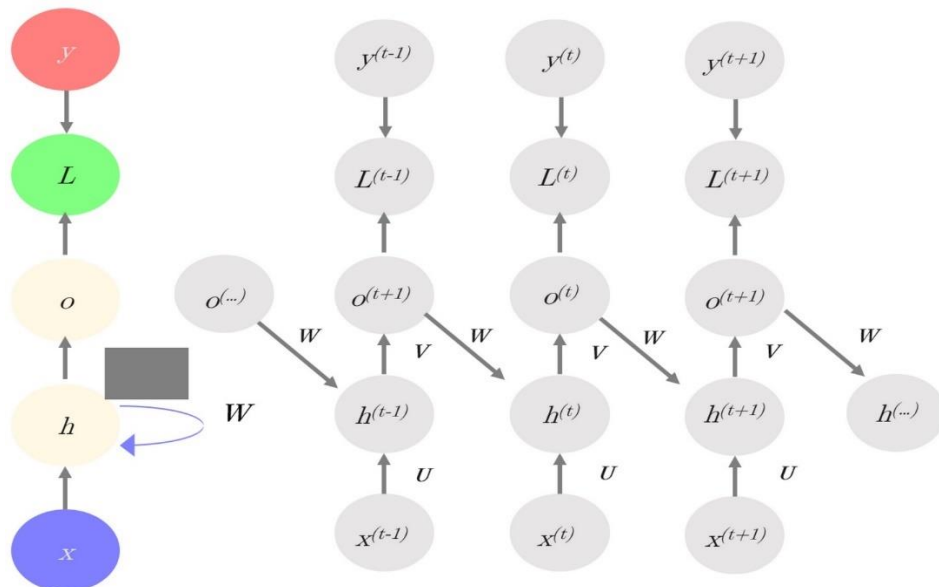


Ilustración 12. Esquema de una red neuronal recurrente. Fuente: <https://www.mdpi.com/2073-4425/10/7/553#>

En otras palabras, este tipo de redes permiten la construcción de modelos que cambian con el tiempo de modo que hay una alta probabilidad de lograr una precisión predictiva

suficientemente alta, dependiendo de los ejemplos que se han presentado en la capa de entrada.

Las redes neuronales recurrentes agregan a las redes neuronales artificiales una memoria que resulta corta. En palabras simples, en cada etapa del aprendizaje, la información en la memoria se mezcla con la nueva y, después de un cierto número de iteraciones, se sobrescribe por completo. Esto se puede evitar utilizando la red de memoria a corto plazo (LSTM por sus siglas en inglés).

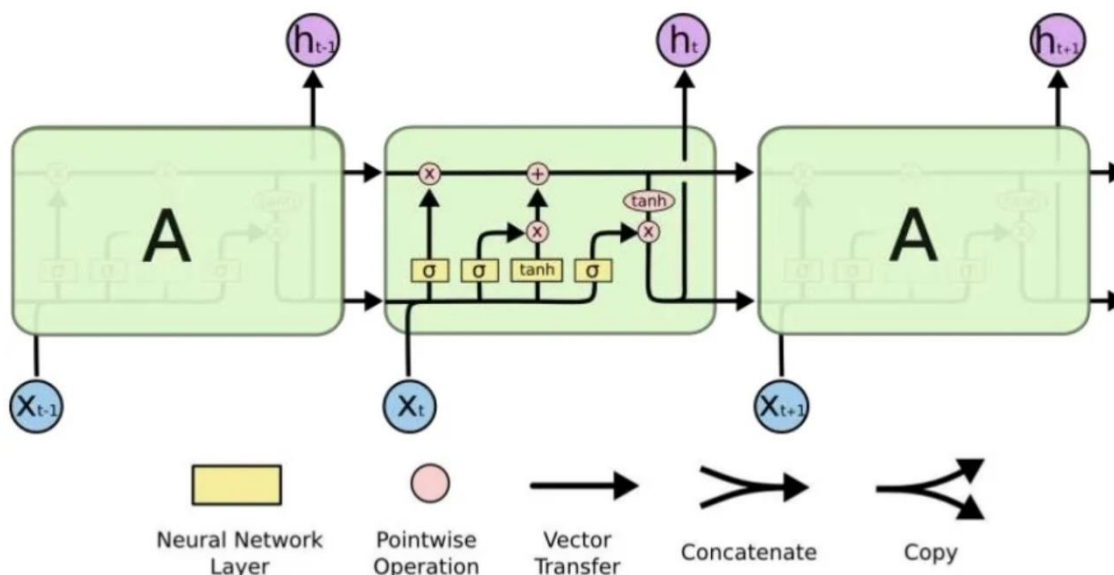


Ilustración 13. Esquema de una capa LSTM. Fuente: <https://www.slideshare.net/MehrnazFaraz/lstm-163514628>

Las redes que contienen módulos LSTM se diseñaron específicamente para evitar el problema de la dependencia a largo plazo. Memorizan los valores por períodos cortos y largos. Esto puede explicarse por el hecho de que el módulo GSM no usa una función de activación dentro de sus componentes recurrentes, por lo que los valores almacenados no se difuminan en el tiempo.

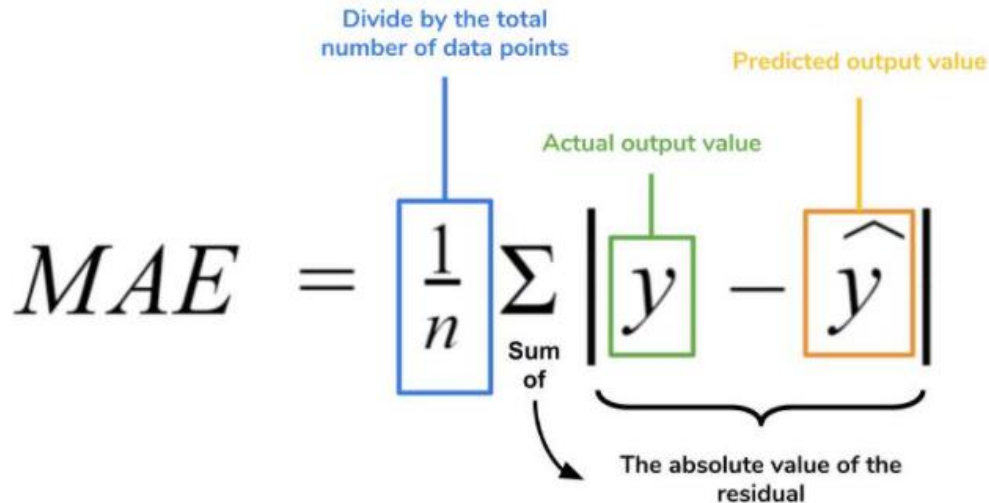
Debido a las ventajas que este tipo de módulos ofrecen para el análisis de series temporales, en el presente Trabajo de Fin de Máster fue probado un modelo recurrente, en el cual fueron usados distintos módulos LSTM.

## 3.7. Métricas

### 3.7.1. Error medio absoluto

La métrica usada para medir la precisión de los modelos creados para el análisis de la serie temporal univariable es el error medio absoluto.

Los errores son la diferencia entre los datos históricos reales y los datos ajustados al pronóstico previstos por el modelo.



The diagram illustrates the Mean Absolute Error (MAE) formula with the following components and annotations:

- MAE**: The metric being calculated.
- =**: The equals sign.
- $\frac{1}{n}$** : A blue box containing the fraction 1 over n. An annotation "Divide by the total number of data points" points to this box.
- $\Sigma$** : The summation symbol.
- Sum of**: An annotation pointing to the summation symbol.
- $|y - \hat{y}|$** : The absolute value of the difference between the actual and predicted values.
  - $y$** : A green box containing the variable y. An annotation "Actual output value" points to this box.
  - $\hat{y}$** : An orange box containing the variable y with a hat. An annotation "Predicted output value" points to this box.
  - The absolute value of the residual**: An annotation pointing to the entire absolute value expression.

Ilustración 14: Fórmula del error medio absoluto. Fuente: <https://medium.com/@polanitzer/the-minimum-mean-absolute-error-mae-challenge-928dc081f031>

En el contexto del aprendizaje automático, el error absoluto se refiere a la magnitud de la diferencia entre la predicción de una observación y el valor real de esa observación. Esta función toma el promedio de errores absolutos para un grupo de predicciones y observaciones como una medida de la magnitud de los errores para todo el grupo. Esta métrica también se puede denominar función de pérdida L1.

Cabe destacar que el modelo idóneo es el modelo en el que esta métrica es la menor.

### 3.7.2. Matriz de confusión

El método escogido para la determinación de la precisión del modelo usado para el análisis multivariable fue el de las matrices de confusión (Kulkarni, 2020). Este método permite medir el rendimiento de modelos de aprendizaje automático dedicados a la clasificación.

Este tipo de matrices para el análisis binario se dividen en cuatro casillas, como se representa en el siguiente esquema:

		Actual Value	
		Present	Absent
Predicted Value	Present	TP	FP
	Absent	FN	TN

Ilustración 15. Esquema de matriz de confusión. Fuente: [https://www.researchgate.net/figure/Confusion-matrix-showing-what-the-true-positive-TP-false-positive-FP-false-negative\\_fig7\\_336927771](https://www.researchgate.net/figure/Confusion-matrix-showing-what-the-true-positive-TP-false-positive-FP-false-negative_fig7_336927771)

- TP (*True Positive*, verdadero positivo): los valores predichos como positivos que han sido pronosticados correctamente.
- FP (*False Positive*, falso positivo): los valores predichos como positivos que han sido pronosticados de forma errónea, es decir, los valores negativos que erróneamente han sido predichos como positivos.
- FN (*False Negative*, falso negativo): los valores predichos como negativos que han sido pronosticados de forma errónea.
- TN (*True Negative*, verdadero negativo): los valores predichos como negativos que han sido pronosticados correctamente.

La precisión en este tipo de métodos se mide dividiendo la suma de los valores predichos correctamente (la diagonal marcada de verde en el esquema) entre la suma del conjunto de predicciones que forman la matriz. Durante el análisis llevado a cabo, la precisión fue calculada antes de llevar a cabo el graficado de la propia matriz de confusión, para poder añadir el mismo al título de esta, con el objetivo de obtener la mayor cantidad de información del mismo gráfico.

### 3.8. Antecedentes

En este momento, existen muchos trabajos dedicados o que abordan el tema de la predicción de la tasa de datos financieros. Se analizan tanto los diversos indicadores fundamentales como los técnicos, utilizando diversas arquitecturas de redes neuronales



artificiales. Lograr la precisión necesaria de los resultados para el comercio rentable es bastante difícil debido a la volatilidad y el ruido del mercado.

En el proceso de familiarización con los tipos de arquitecturas de redes neuronales y su eficacia en la predicción de series temporales, se revisó un gran número de trabajos, entre los que se pueden destacar aquellos en los que la red neuronal mostró una alta precisión o contribuyó al desarrollo general de la tecnología de predicción de cursos de instrumentos financieros.

El primer trabajo (Wlodarczak, 2017) describe el impacto de la actividad social en el precio de las acciones de Apple. La gran red social Twitter actúa como objeto de estudio. En primer lugar, los textos de los tweets se recopilaron y procesaron previamente para analizar más a fondo su polaridad de los sentimientos de los usuarios hacia la compañía Apple. La clasificación en sí se realizó a través de una red neuronal convolucional, para cuya capacitación se suministraron más de 250 mil entradas. del 1 de noviembre de 2015 al 31 de diciembre de 2015 en los que se menciona a Apple o sus productos. Se crearon varios conjuntos de datos utilizando diferentes métodos de preprocesamiento y análisis. La frecuencia de los tweets se presentó como una serie de tiempo para una comparación adicional con una serie de datos sobre el curso de las acciones de Apple. Los métodos utilizados en este trabajo no mostraron una alta correlación debido al ruido y la distribución asimétrica en los conjuntos de datos, sin embargo, este estudio contribuyó a la presentación de un análisis cuantitativo de los datos del espacio de medios sociales para predecir el precio de las acciones.

El segundo artículo (Sen & Tamal, 2017) propone un enfoque de predicción de precios de acciones que combina métodos estadísticos y de aprendizaje automático con algunos conceptos descritos en varias publicaciones de análisis técnico. El modelo predictivo construido utilizó datos de 5 minutos desde 2013 hasta 2014 sobre los precios de las acciones de *Tata Steel* y *Hero Moto* pertenecientes a la bolsa nacional de valores en la India.

La verificación de la precisión del pronóstico se realizó utilizando el indicador de desviación estándar. El mejor resultado fue un enfoque de regresión multivariable basado en un árbol de decisión aleatorio. Este algoritmo consiste en un conjunto de árboles de decisión, lo cual resuelve el problema del sobreentrenamiento y mejora la precisión en comparación con un solo árbol. La esencia de esta predicción es agregar las respuestas de muchos árboles que se entrenan en diferentes subconjuntos (independientemente uno del otro), lo que resuelve el problema de construir copias de árboles para un solo conjunto de datos.

El tercer trabajo (Yan, 2017) en el ejemplo del índice SCI (*Shanghai Composite Index*) muestra la ventaja de la red neuronal recurrente (RNN) sobre el perceptrón multicapa (MLP) al predecir el curso. El artículo propone un modelo de predicción de series temporales que cubre sus características complejas, como la no linealidad, la inestabilidad de las Series de tiempo financieras. Este trabajo utiliza un modelo de red neuronal recurrente de memoria a largo plazo (LSTM). Su principal diferencia con el RNN es la capacidad de recordar información durante largos períodos, lo cual es excelente en el análisis de series de tiempo financieras. Además de los datos de precio y volumen, se suministraron datos de algunos indicadores a la entrada. A continuación,

se diseñaron dos redes (LSTM y MLP), que recibieron datos diarios durante un período de 3 años y medio. Los resultados empíricos mostraron que el LSTM muestra un resultado más preciso que el MLP en este caso.

El último trabajo considerado en este análisis (Liu, 2018) es una predicción de precio de las acciones por medio de la red recurrente (LSTM) en conjunto con otras arquitecturas. La entrada recibió noticias financieras de las 12 grandes compañías de capital que forman parte del índice S&P 500. Específicamente, la red LSTM se utiliza para codificar texto de noticias y capturar información contextual. Las redes se entrenaron con estos datos y, sobre la base de los resultados obtenidos, se reveló que algunas relaciones, en particular CNN-LSTM, mostraron un resultado de precisión predictiva muy alto.



## 4. Desarrollo del proyecto

El desarrollo de este proyecto ha sido llevado a cabo en dos partes fundamentales. Por un lado, se realizó un análisis univariable de los datos, con el objetivo de predecir precios en gráficos diarios en un corto y/o medio plazo.

En segundo lugar, se analizó una serie de datos en diario a la cual se le añadió distinta información que puede influir bastante en el precio, con el objetivo de predecir subidas o bajadas de precio en gráficos minutales en un corto plazo.

### 4.1. Herramientas utilizadas

Dentro de la ciencia de datos podemos encontrar una gran variedad de herramientas que se ajustan a cada una de las necesidades del proceso KDD. Para este proyecto en concreto fueron seleccionadas las herramientas de Anaconda para la programación en *Python* gracias a los cuadernos que nos ofrece *Jupyter*.

Las herramientas aquí mencionadas fueron usadas para todo el proceso KDD, desde la obtención de los datos con métodos de minería gracias a la biblioteca *requests* hasta la obtención de conocimientos después de un análisis con las técnicas de aprendizaje profundo mencionadas en el quinto apartado del presente Trabajo de Fin de Máster.

#### 4.1.1. Anaconda

Anaconda es un administrador de paquetes de código abierto y un sistema de administración de entornos que se ejecuta en *Windows*, *macOS* y *Linux*.

Esta herramienta es fácil de instalar, ejecutar y actualizar paquetes y dependencias. Con ella, es posible crear, guardar, cargar y cambiar fácilmente entre distintos entornos en el equipo local. Funciona como un gestor de entorno y un gestor de paquetes.

Se concibió para programas de *python*, pero puede crear paquetes y distribuciones de software en cualquier otro lenguaje de programación.



Ilustración 16. Logotipo de Anaconda. Fuente: <https://www.anaconda.com/>

Para la realización de este Trabajo de Fin de Máster se utiliza anaconda para crear el entorno de desarrollo del mismo junto con el resto de herramientas mencionadas en los puntos posteriores.

### 4.1.2. Python

*Python* es un lenguaje de programación orientado a objetos utilizado para resolver un gran volumen de los problemas más diversos. Este lenguaje es útil en la creación de aplicaciones informáticas y móviles, se usa en el trabajo con una gran cantidad de información, en el desarrollo de sitios web y otros proyectos diversos, además de ser la herramienta por excelencia en el aprendizaje automático. Este lenguaje de programación es utilizado por grandes corporaciones conocidas como *Spotify*, *Amazon*, *YouTube* e incluso *Disney* en distintas áreas, desde el desarrollo de distintas aplicaciones de su ecosistema a los algoritmos de recomendación basados en machine learning. Por lo tanto, *Python* ha encontrado su lugar en varias áreas: con su ayuda, es posible resolver muchos problemas de diferente complejidad.



Ilustración 17. Logotipo de Python. Fuente: <https://www.python.org/>

Python es ampliamente utilizado debido a ser un lenguaje de programación de código abierto, y debido a la gran facilidad que ofrece a la hora de usar librerías escritas en otros lenguajes de programación.

Uno de los pocos puntos negativos de este lenguaje es que, al ser interpretado, su ejecución es bastante lenta comparada con otros lenguajes más rápidos como *C* o *Java*. Este problema se soluciona usando librerías escritas en otros lenguajes para realizar las operaciones más pesadas.

Las librerías usadas para la creación de algoritmos de aprendizaje profundo en este Trabajo de Fin de Máster y expuestas más adelante están escritas en *C++*, lo que agiliza enormemente el proceso de entrenamiento y predicciones.

### 4.1.3. Jupyter

*Jupyter* es un entorno de desarrollo donde es posible ver inmediatamente el resultado de la ejecución del código y sus fragmentos individuales. La diferencia con el entorno

de desarrollo tradicional es que el código se puede dividir en fragmentos y ejecutarlos en un orden aleatorio.

En este entorno de desarrollo, se posibilita, por ejemplo, escribir una función y verificar inmediatamente su funcionamiento, sin ejecutar el programa en su totalidad. Y también es posible cambiar el orden de ejecución del código. Se puede cargar un archivo en la memoria por separado, verificar su contenido por separado, procesar el contenido por separado.

Y en los cuadernos virtuales *Jupyter* hay una salida de resultado justo después del fragmento de código. Esto posibilita, por ejemplo, ver un gráfico construido en el medio del código, obtener números preliminares o cualquier otra visualización.



Ilustración 18. Logotipo de Jupyter. Fuente: <https://www.jupyter.org/>

Las principales aplicaciones de los cuadernos virtuales de *Jupyter* son el aprendizaje automático, las redes neuronales, la visualización de datos y la estadística.

Incluso este entorno a menudo se utiliza para el desarrollo por etapas, cuando es necesario verificar el trabajo de diferentes fragmentos de código en pasos. El hecho es que el código en estos cuadernos se almacena en celdas independientes y se puede ejecutar en cualquier orden o uno por uno. Esto permite experimentar rápidamente con algoritmos y encontrar la solución óptima.

*Jupyter* permite usar esta tecnología en la nube con *Google Colab* o en local. Para este trabajo, *Jupyter* fue usado en local.

#### 4.1.4. Pandas

*Pandas* es una biblioteca de *Python* para el procesamiento y análisis de datos estructurados, su nombre proviene del inglés, "*panel data*" ("datos de panel"). Los datos del panel se refieren a la información obtenida de la investigación y estructurada en forma de tablas. Para trabajar con tales matrices de datos fue creado *pandas*.

*Pandas* es una biblioteca de código abierto, es decir, su código fuente está disponible públicamente en *GitHub*. Los usuarios pueden agregar su código allí: añadir explicaciones, complementar los métodos de trabajo y actualizar las secciones. El trabajo requiere un compilador (un programa que traduce texto del lenguaje de

programación al código de máquina) C/C++ y un entorno de desarrollo de *Python*. El proceso detallado de instalación del compilador C para diferentes sistemas operativos se puede encontrar en la documentación de *pandas*.



Ilustración 19. Logotipo de Pandas. Fuente: <https://www.pandas.pydata.org/>

Pandas es usado habitualmente para el análisis de datos ya que permite la apertura de cualquier archivo, como excel, csv, etc y su reorganización en un tipo de datos personalizado, *DataFrame*. Este tipo de datos permite estructurar los datos y realizar una gran cantidad de modificaciones a gran velocidad, además de extraer información que nos pueda resultar interesante.

#### 4.1.5. Tensorflow y Keras

*TensorFlow* es un *framework* abierto de aprendizaje automático creado por el equipo de *Google Brain* que combina muchos modelos y algoritmos de aprendizaje profundo. *TensorFlow* proporciona una API en *python*, y los cálculos en sí ocurren en C ++, lo cual permite realizar distintas operaciones con un gran rendimiento.

Este *framework* puede entrenar redes neuronales profundas para, reconocimiento de imágenes, incrustación de palabras, redes neuronales recurrentes, modelos de traducción automática, procesamiento de lenguaje natural, etc. todos los modelos de aprendizaje automático se pueden lanzar a producción con la herramienta *TensorFlow Serving*.



Ilustración 20. Logotipo de Tensorflow. Fuente: <https://www.tensorflow.org/>

Por el momento, la versión moderna es *TensorFlow 2*. Pero algunos científicos de datos siguen usando la versión 1 a día de hoy, ya que entre estas versiones existen diferencias sustanciales y no se ha podido pasar todo el código escrito en la versión más antigua (*Tensorflow 1*) a la versión más moderna (*Tensorflow 2*).

La principal diferencia entre las versiones es el cambio de computación diferida a instantánea. Además, la segunda versión se combina con otro *frameworks* de aprendizaje profundo — *Keras*.

La esencia de los cálculos diferidos es que todas las operaciones de tensor ocurren solo en el momento de la compilación. Esto es un poco diferente al estilo de programación de *Python*, ya que el resultado se puede ver de inmediato, por lo que *TensorFlow 2*, siguiendo a su rival *Pytorch*, se ha movido al modo de computación instantánea. Aunque los desarrolladores aún dejaron la posibilidad de cambiar a la computación retrasada.

El marco de aprendizaje profundo *Keras* hizo una gran contribución. Hasta hace algún tiempo, *TensorFlow* era una biblioteca de computación de tensor, mientras que *Keras* sirve para crear arquitecturas de redes neuronales. A partir de la versión 2, *Keras* se convirtió en parte de *TensorFlow*, lo que le dio una gran ventaja, ya que no es necesario configurarlos individualmente.



Ilustración 21. Logotipo de Keras. Fuente: <https://keras.io/>

*Keras* proporciona todas las herramientas necesarias para modelar arquitecturas de redes neuronales profundas de alto nivel, como distintos tipos de capas, funciones de pérdida y activación, métricas, optimizadores y *callbacks*.

En este Trabajo de Fin de Máster se usa la versión más moderna de *Tensorflow* con *Keras*, la 2.9.1.

#### 4.1.6. Matplotlib

La biblioteca *matplotlib* es una biblioteca de gráficos bidimensionales para el lenguaje de programación *Python* con la que se pueden crear dibujos de alta calidad de varios formatos. *Matplotlib* es un módulo-paquete para este lenguaje de programación.



Ilustración 22. Logotipo de Matplotlib. Fuente: <https://www.matplotlib.org/>

*Matplotlib* es una alternativa al módulo de visualización del programa para cálculos técnicos de *MatLab*. *Matplotlib* posee una interfaz orientada a objetos, es decir, el usuario interactúa directamente con cada objeto. Esto permite utilizar el código para especificar cualquier elemento del gráfico, incluidas las etiquetas y las marcas de los ejes.

*Matplotlib* se utiliza para representar todo tipo de gráficos. Es una biblioteca indispensable para cualquier analista de datos. Además de eso, *Matplotlib* subyace a otras bibliotecas, como *Seaborn*, que presenta una interfaz de alto nivel sobre *Matplotlib*. En algunos casos, se usa *Seaborn*, por ejemplo, cuando se requiere realizar gráficos simples de manera rápida, pero cuando se necesitan más detalles y trabajo, la opción a elegir suele ser *Matplotlib*.

En el presente Trabajo de Fin de Máster se usa esta biblioteca para realizar las visualizaciones, según los consejos y técnicas dados por el profesor Raúl Reyero Díez en la asignatura Visualización de Datos.

## 4.2. Serie temporal univariable

### 4.2.1. Análisis de ventana deslizante

El análisis llevado a cabo en este primer punto se trata de un análisis de ventana deslizante. La base de este algoritmo consiste en una estructura de datos similar a una cola (*queue*). Al igual que esta, sigue la metodología *First-In-First-Out*, que básicamente asegura que los datos introducidos en esta estructura en primer lugar sean los primeros en salir.

Esta estructura de datos permite tener un orden en los mismos. La principal característica de la ventana deslizante con respecto a las colas tradicionales es que el tamaño de la misma es constante.

En este caso, esta ventana se usa para recorrer los datos de precios de forma ordenada, tal y como se muestra en la siguiente ilustración.

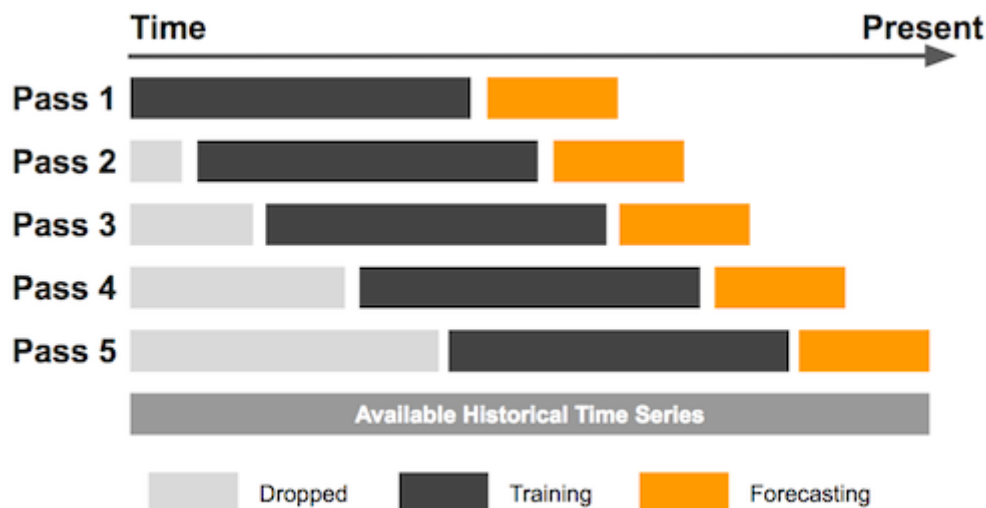


Ilustración 23. Esquema de una ventana deslizante. Fuente: <https://www.kaggle.com/code/cworsnup/backtesting-cross-validation-for-timeseries/notebook>

De esta forma, el problema a analizar es un problema de análisis supervisado, donde las características a analizar son los precios anteriores al precio etiqueta. Esto permite reconocer patrones en una serie temporal.

Además, este algoritmo abre nuevas posibilidades ante procesos de entrenamiento de modelos de *machine learning* con redes neuronales, ya que introduce dos nuevos parámetros, adaptables para cualquier *dataset*: el tamaño de la ventana y el tamaño del llamado “horizonte”.

Este último hace referencia a la cantidad de días a predecir a partir de la ventana.

Para este apartado del presente Trabajo de Fin de Máster, se realizaron pruebas con distintos valores para estos parámetros, y empíricamente tras numerosas pruebas y experimentos se demostró que para la serie temporal que nos concierne, los valores óptimos son una ventana de siete días y un valor objetivo (horizonte) de un día.

#### 4.2.2. Obtención de los datos

El conjunto de datos utilizado para el análisis de datos en diario es el obtenido directamente de la *API* de la casa de cambios “coinbase”, siendo la *url* utilizada la siguiente: <https://web-api.coinmarketcap.com/v1/cryptocurrency/ohlcv/historical>.

Los datos usados fueron los de bitcoin, debido a que al ser la criptomoneda más antigua es la que más datos históricos tiene. Igualmente, el experimento es replicable con cualquier otra criptomoneda, simplemente cambiando el símbolo de la misma en la parte del código en la que se obtiene el *dataset*.

Gracias a las posibilidades que nos ofrece de por sí, para realizar estas peticiones se usó una biblioteca ya desarrollada para este fin, la cual permite directamente obtener los datos históricos en formato *DataFrame*, lo cual facilita enormemente el análisis.



Esto permite tener la red neuronal lo más actualizada posible, ya que cada vez que se ejecuta el cuaderno en el cual se realiza el entrenamiento de la misma, se obtienen los datos más recientes posibles, lo cual posibilita que los modelos entrenados tengan en cuenta las últimas modificaciones.

Dicha biblioteca se puede encontrar en el siguiente enlace: <https://github.com/guptarohit/cryptoCMD/tree/44ac099efbeef53eebf7e082f2fefeedb11f1dce>

### 4.2.3. Formateo de los datos

Como se ha expuesto en el punto anterior, tras obtener los datos de la API de *coinbase*, los datos son procesados y devueltos en formato *DataFrame*, lo cual permite directamente formatearlos usando la biblioteca *pandas*.

Sin embargo, para este primer análisis la única variable que nos interesa es la variable del precio, por lo que la extraemos del *dataset*, tal y como se muestra en la siguiente ilustración.

```
def only_prices_df(df):
    result = df.copy()
    result.set_index('Date', inplace=True)
    return pd.DataFrame(result['Open']).rename(columns={'Open': PRICE}).astype('float32')
```

*Ilustración 24. Formateo de dataset de partida para análisis univariable. Elaboración propia*

Como se puede observar, se realizan además dos modificaciones adicionales. En primer lugar, se sitúa como índice de los datos del *dataset* el campo que indica la fecha, para que esta sea la característica por la cual se ordenen los datos. Finalmente, se reformatean los datos de la tabla al formato *float32*, lo cual permite agilizar los cálculos con tensores en *TensorFlow*.

Sin embargo, es necesario realizar una modificación extra para poder extraer las etiquetas y las características del *dataset* fácilmente, y esta es la creación de ventanas.

```
def create_windows(df, window_size):
    result = df.copy()
    for i in range(1, window_size + 1):
        result[f"Precio - {i} día(s)"] = result[PRICE].shift(periods=i)
    return result.dropna()
```

*Ilustración 25. Creación de ventanas. Elaboración propia*

Estas ventanas consisten en la serie de datos de los días anteriores, tal y como se explicó en el primer punto. Esta función nos permite modificar el tamaño de la ventana en cuestión simplemente con la modificación de un parámetro de la misma.

Para este análisis en cuestión, el tamaño de ventana elegido fue siete, ya que este fue el valor que dio mejores resultados en los numerosos experimentos.



Después de las modificaciones mencionadas, la tabla de datos quedó de la siguiente manera:

	Precio	Precio - 1 dia(s)	Precio - 2 dia(s)	Precio - 3 dia(s)	Precio - 4 dia(s)	Precio - 5 dia(s)	Precio - 6 dia(s)	Precio - 7 dia(s)
Date								
2013-05-05	112.900002	98.099998	106.250000	116.379997	139.000000	144.000000	134.444000	135.300003
2013-05-06	115.980003	112.900002	98.099998	106.250000	116.379997	139.000000	144.000000	134.444000
2013-05-07	112.250000	115.980003	112.900002	98.099998	106.250000	116.379997	139.000000	144.000000
2013-05-08	109.599998	112.250000	115.980003	112.900002	98.099998	106.250000	116.379997	139.000000
2013-05-09	113.199997	109.599998	112.250000	115.980003	112.900002	98.099998	106.250000	116.379997
...	...	...	...	...	...	...	...	...
2022-08-17	23881.316406	24126.136719	24318.316406	24429.056641	24402.187500	23957.203125	23948.345703	23162.898438
2022-08-18	23341.039062	23881.316406	24126.136719	24318.316406	24429.056641	24402.187500	23957.203125	23948.345703
2022-08-19	23213.312500	23341.039062	23881.316406	24126.136719	24318.316406	24429.056641	24402.187500	23957.203125
2022-08-20	20872.841797	23213.312500	23341.039062	23881.316406	24126.136719	24318.316406	24429.056641	24402.187500
2022-08-21	21160.392578	20872.841797	23213.312500	23341.039062	23881.316406	24126.136719	24318.316406	24429.056641

3396 rows × 8 columns

*Ilustración 26. DataFrame de precios en diario preparada para análisis univariable. Elaboración propia*

## 4.2.4. Preparación para el entrenamiento

### 4.2.4.1. Datasets de entrenamiento y testeo

Uno de los pasos más importantes a la hora de entrenar un modelo es el de seleccionar los datos de entrenamiento y de testeo. Los datos de entrenamiento sirven para que la red neuronal encuentre los pesos óptimos para un *dataset* concreto, mientras que los datos de testeo sirven para probar la red resultante en la práctica. Esta prueba se realiza con datos del conjunto del *dataset* de partida ya que se necesitan datos con etiqueta, y no se pueden usar los datos que se han usado durante el entrenamiento ya que, al haber entrenado el modelo sobre ellos, los pesos estarían ya optimizados para estos datos, por lo que se obtendrían resultados positivos pero completamente irreales.

En el caso del presente Trabajo de Fin de Máster la función usada para realizar la separación de datos en el conjunto de entrenamiento y el conjunto de testeo fue la siguiente:

```
def get_train_test_datasets(X, y, split=0.8, batch_size=128):
    split_size = int(len(X) * split)
    X_train, y_train = X[:split_size], y[:split_size]
    X_test, y_test = X[split_size:], y[split_size:]

    train_datos = tf.data.Dataset.from_tensor_slices(X_train)
    train_etiquetas = tf.data.Dataset.from_tensor_slices(y_train)
    test_datos = tf.data.Dataset.from_tensor_slices(X_test)
    test_etiquetas = tf.data.Dataset.from_tensor_slices(y_test)

    train_dataset = tf.data.Dataset.zip((train_datos, train_etiquetas))
    test_dataset = tf.data.Dataset.zip((test_datos, test_etiquetas))
    AUTOTUNE = tf.data.experimental.AUTOTUNE
    train_dataset = train_dataset.batch(batch_size).prefetch(AUTOTUNE)
    test_dataset = test_dataset.batch(batch_size).prefetch(AUTOTUNE)
    return train_dataset, test_dataset
```

*Ilustración 27. Función de separación de datos en conjuntos de entrenamiento y test. Elaboración propia*

Además de separar los datos en conjuntos de entrenamiento y testeo, permitiendo indicar el tamaño de cada partición, esta función pasa estos conjuntos a un formato de datos nativo para la biblioteca *Tensorflow*, lo cual agiliza el proceso de entrenamiento. Este formato, además, ofrece la posibilidad de guardar las etiquetas y las características correspondientes en una misma variable, lo cual permite simplificar el código.

#### 4.2.4.2. Hiperparámetros

Durante la creación de una red neuronal, uno de los pasos principales es la selección correcta de hiperparámetros, es decir, parámetros que no cambian durante el aprendizaje de la red neuronal (Chapelle, Vapnik, & Bousquet, 2002). Estos parámetros deben ajustarse para que el modelo resultante pueda resolver de manera óptima la tarea en cuestión. La optimización se refiere a la búsqueda de una tupla de hiperparámetros que pueda producir un modelo óptimo con datos independientes dados.

La dificultad de esta optimización radica en el hecho de que el desarrollador debe encontrar un compromiso entre la velocidad de aprendizaje y la calidad de la predicción. Además, la presencia de algunos hiperparámetros depende en gran medida del valor de otros. Por ejemplo, el tamaño de cada capa oculta en una red neuronal puede depender del número total de todas las capas de la red.

La elección correcta de estos parámetros es importante e influye directamente en la convergencia de la red neuronal, lo que indica si la arquitectura de la red neuronal es correcta y si, de acuerdo con la tarea en cuestión, los hiperparámetros se han emparejado.

Al diseñar una red neuronal, la primera elección a realizar la de su arquitectura, dependiendo de la cual se seleccionan las capas necesarias, su número, así como sus

hiperparámetros. Los tipos de capas utilizados en el presente Trabajo de Fin de Máster son los explicados en el marco teórico.

En este paso, algunos de los hiperparámetros a elegir son tanto el número de neuronas en cada capa como el número de capas, los cuales se corresponderán con la complejidad de la red neuronal. Cabe destacar que no necesariamente una cantidad de neuronas y de capas se corresponde con una mayor precisión. Normalmente, se suelen usar múltiplos de dos para la cantidad de neuronas, decreciendo este número en cada capa (a excepción de las capas convolucionales, donde este número se incrementa).

Una etapa adicional en el desarrollo de la red neuronal es la selección de la función de activación neuronal. El tipo de esta función determina en gran medida el método de aprendizaje y la funcionalidad de la red neuronal. Se sabe que si hay suficientes capas en las redes neuronales y la función de activación no es lineal, entonces son capaces de aproximar la función de cualquier complejidad.

La función de activación usada en las capas ocultas de los modelos presentados en este apartado ha sido *ReLU* (ver ilustración 11). Esta función suele usarse para estas capas ya que, con respecto a otras funciones (como sigmoide o tangencial), esta función permite la ejecución de operaciones menos intensiva en recursos, el recorte de detalles innecesarios y el aprendizaje rápido. Al tratarse de un problema de regresión, todos los modelos creados en el presente apartado tienen una última capa formada por una única neurona y con una función de activación lineal.

Otro importante hiperparámetro a elegir para el entrenamiento de una red neuronal es la función de pérdidas. Esta función es necesaria para calcular el error entre las respuestas recibidas y los datos reales. El error es la cantidad que refleja la discrepancia entre las respuestas recibidas y esperadas. Se forma después de la ejecución de cada época. Con la configuración correcta de la red neuronal, esta tasa debería disminuir. El estándar para problemas de regresión es la función de pérdidas *mse* (*mean squared error* por sus siglas en inglés, o error cuadrático medio en castellano). Esta función representa la diferencia cuadrática entre los valores predichos y reales. El resultado es siempre positivo, independientemente del signo, y el valor ideal es 0. La elevación al cuadrado conlleva que las desviaciones más grandes de la evidencia conducen a errores mayores que las desviaciones más pequeñas, lo que significa que el modelo está penalizado por errores más grandes.

Otro de los hiperparámetros son los optimizadores. En función del error calculado, las funciones de optimización se utilizan para ajustar los pesos utilizando el descenso de gradiente y la retropropagación.

Se conoce como algoritmo de retropropagación a una familia de métodos que se utiliza para entrenar redes neuronales de manera efectiva, basándose en el enfoque de descenso de gradiente, que a su vez utiliza la regla de la cadena. Este algoritmo requiere que, durante el diseño de la red, se conozcan los derivados de las funciones de activación.

Los optimizadores desempeñan un papel muy importante en el aprendizaje de redes neuronales, ya que no solo mejoran el resultado final, sino que también pueden acelerar

el proceso de aprendizaje. Por lo tanto, la selección correcta del optimizador es una de las etapas más importantes en la creación de redes neuronales.

El optimizador seleccionado para el entrenamiento de modelos es el optimizador “Adam”.

*Adam* es un optimizador que indica la reducción de la evaluación adaptativa del momento y calcula la velocidad de aprendizaje de cada parámetro. Difiere significativamente de otros algoritmos de aprendizaje adaptativo de redes neuronales, ya que converge muy rápidamente a una velocidad de aprendizaje del modelo bastante rápida y eficiente, y también elimina todos los posibles problemas con los que otros optimizadores se pueden encontrar, como la convergencia lenta o la tasa de aprendizaje que desaparece.

El parámetro principal del optimizador es la tasa de aprendizaje (*learning rate*, *lr*). Este parámetro de métodos de entrenamiento de gradiente de redes neuronales le permite controlar el valor de corrección de peso después de cada iteración. La tasa varía de 0 a 1. Además, el número de épocas (o pasos) necesarios para encontrar el óptimo aumenta y, al mismo tiempo, aumenta la precisión de ajuste, lo que puede reducir las pérdidas (errores) durante el entrenamiento de la red. Para el entrenamiento de los modelos creados en este apartado, fue decidido usar la tasa de aprendizaje por defecto del optimizador *Adam*.

#### 4.2.4.3. Callbacks

Un *callback* es un conjunto de funciones que se aplicara en determinadas etapas del procedimiento de entrenamiento. Estas funciones son usadas para obtener información de los estados internos y las estadísticas del modelo durante el entrenamiento.

Estas funciones nos permiten automatizar algunas tareas tras cada iteración del proceso de entrenamiento. Esto incluye detener el entrenamiento cuando se alcanza una determinada puntuación de precisión o pérdida, guardar un modelo como un punto de control después de cada época exitosa, ajustar las tasas de aprendizaje con el tiempo, etc.

Generalmente, durante el presente Trabajo de Fin de Máster, los *callbacks* usados han sido los siguientes:

- **ModelCheckpoint:** esta función se utiliza para guardar un modelo determinado en unos momentos concretos del entrenamiento. En este apartado, dicho *callback* fue usado para guardar los pesos de un modelo cada vez que, durante el entrenamiento, se mejoran los pesos con respecto al mejor modelo anterior del mismo entrenamiento.
- **ReduceLROnPlateau:** esta función se utiliza para reducir la tasa de aprendizaje en el momento en el que una métrica específica deja de mejorar y se estanca.

## 4.2.5. Creación y comparación de modelos

En el siguiente apartado fueron creados y comparados cuatro modelos. Dichos modelos fueron comparados por el resultado de la función de pérdidas utilizada (*mse*) sobre el conjunto de datos de validación.

### 4.2.5.1. Perceptrón multicapa

Este primer modelo fue el modelo más simple de los desarrollados. Esta red consiste en un perceptrón de únicamente cuatro capas. La primera de ellas, la de entrada, posee tantas neuronas como tamaño nuestra ventana de precios. La segunda y la tercera se tratan de dos capas ocultas de 128 y 16 neuronas respectivamente, activadas ambas con la función ReLU. Finalmente, la última capa es de únicamente una neurona con una función de activación lineal.

Sobre el código, esta función se ve de la siguiente forma:

```
model_1 = tf.keras.Sequential([
    layers.Dense(128, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="linear")
], name="FCC_simple")
```

*Ilustración 28. Creación de perceptrón multicapa con TensorFlow. Elaboración propia*

### 4.2.5.2. Red neuronal convolucional

El segundo modelo a desarrollar fue el de una red neuronal convolucional. Este tipo de modelos están compuestos por dos partes, un *base model* donde se encuentran las capas convolucionales y un *top model* donde se encuentran las capas de neuronas de perceptrón multicapa.

Al ser la ventana un conjunto de datos en una única dimensión, las capas convolucionales usadas fueron las *Conv1D*, creadas con este propósito. Estas capas, sin embargo, requieren de una expansión dimensional de los datos, y es por ello que la primera capa de este modelo es una capa con esta función.

Tras las dos capas convolucionales de 16 y 32 filtros respectivamente y activadas con la función de activación *ReLU*, fue usada una capa *Flatten* para reducir la dimensionalidad del espacio latente y poder pasar la información a la siguiente capa.

El *top model* de este modelo consiste en dos capas conectadas, ambas activadas linealmente, teniendo la última capa una única neurona.

Esto está representado en el código de la siguiente forma:



```
model_2 = tf.keras.Sequential([
    layers.Lambda(lambda x: tf.expand_dims(x, axis=1)),
    layers.Conv1D(filters=16, kernel_size=4, padding="same", activation="relu"),
    layers.Conv1D(filters=32, kernel_size=4, padding='same', activation='relu'),
    layers.Flatten(),
    layers.Dense(16),
    layers.Dense(1)
], name="CNN_simple")
```

*Ilustración 29. Creación de una red convolucional con TensorFlow. Elaboración propia*

### 4.2.5.3. Red neuronal recurrente

La base de este tercer modelo son las capas recurrentes. Este tipo de capas son usadas con gran frecuencia para el análisis de series temporales, gracias a la versatilidad que ofrece su procesamiento de memoria.

En este caso, para la creación de este modelo fueron creadas dos capas de redes neuronales recurrentes. Este tipo de capas, al igual que las capas convolucionales unidimensionales, requieren una expansión dimensional de los datos. Con el objetivo de que los conjuntos de datos de validación y entrenamiento sean uniformes y versátiles para cada modelo, este problema (al igual que con el modelo anterior) se soluciona añadiendo una capa extra al principio que cumpla esta función.

Tras dicha capa, el modelo continúa con dos capas recurrentes de 128 y 16 neuronas respectivamente activadas con la función *ReLU*, y finaliza con una capa interconectada de una única neurona activada con una función lineal.

La creación de este modelo se ejecuta de la siguiente manera:

```
model_3 = tf.keras.Sequential([
    layers.Lambda(lambda x: tf.expand_dims(x, axis=1)),
    layers.LSTM(128, activation="relu", return_sequences=True),
    layers.LSTM(16, activation="relu"),
    layers.Dense(1, activation="linear")
], name="RNN_simple")
```

*Ilustración 30. Creación de una red recurrente con TensorFlow. Elaboración propia*

### 4.2.5.4. Algoritmo N-BEATS

El algoritmo *N-BEATS* se centra en resolver problemas de pronóstico puntual de series temporales univariadas usando aprendizaje profundo (Oreshkin, Carpov, Chapados, & Bengio, 2019). La arquitectura propuesta se basa en retrospectiva y envía enlaces residuales y una pila muy profunda de capas totalmente conectadas. Esta arquitectura tiene una serie de propiedades, siendo aplicable a una amplia gama de dominios de destino a través de distintas modificaciones.

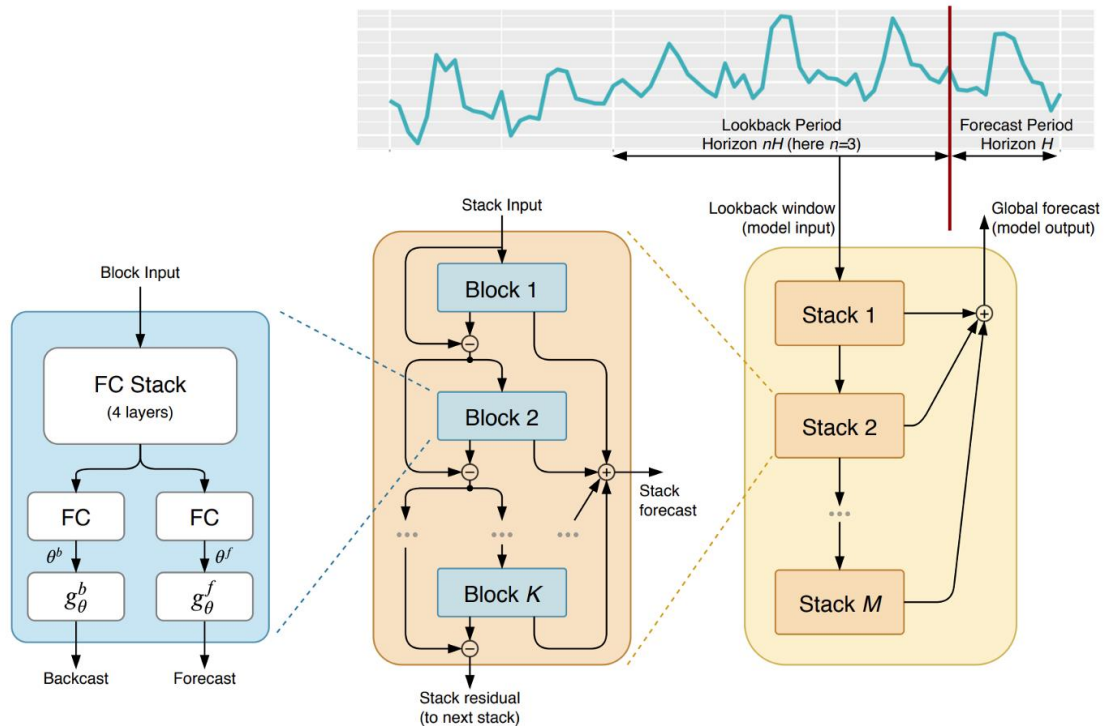


Ilustración 31. Esquema del algoritmo N-BEATS. Fuente: <https://arxiv.org/pdf/1905.10437.pdf>

Este algoritmo se basa en distintas capas que están formadas por distintos bloques conectados en serie entre sí, estando todos conectados a la salida tal y como se muestra en la ilustración. A la entrada de cada bloque se le conecta la sustracción de la salida del último bloque y la del penúltimo.

A su vez, cada uno de estos bloques está dividido en distintas partes. La primera parte y la que recibe los datos de entrada consiste en un perceptrón multicapa de cuatro capas. La salida de la última de estas cuatro capas se procesa por dos capas paralelas, produciendo estas dos las salidas del bloque. Una de estas salidas (*backcast*) es la que irá a la entrada de otro bloque y la otra (*forecast*) es la que irá al acumulado de la predicción, siendo a su vez la salida del modelo.

Este algoritmo fue representado para este trabajo con el siguiente código:



```

nbeats_input = layers.Input(shape=(input_size), name="input")
residuals = nbeats_input

for i in range(stacks_len):
    x = residuals
    for j in range(n_layers):
        x = layers.Dense(n_neurons, activation="relu", name=f"{j}_dense_{i}th_stack")(x)
        theta = layers.Dense(theta_size, activation="linear")(x)

    backcast, block_forecast = theta[:, :input_size], theta[:, horizon:]
    residuals = layers.subtract([residuals, backcast])
    if not i:
        forecast = block_forecast
    else:
        forecast = layers.add([forecast, block_forecast])

model_4 = tf.keras.Model(inputs=nbeats_input,
                          outputs=forecast,
                          name="n-beats")

```

*Ilustración 32. Implementación del algoritmo N-BEATS. Elaboración propia*

La implementación de este algoritmo con la *API* secuencial de *Tensorflow* no es posible, por lo que en este fragmento de código fue usada la *api* funcional de esta biblioteca.

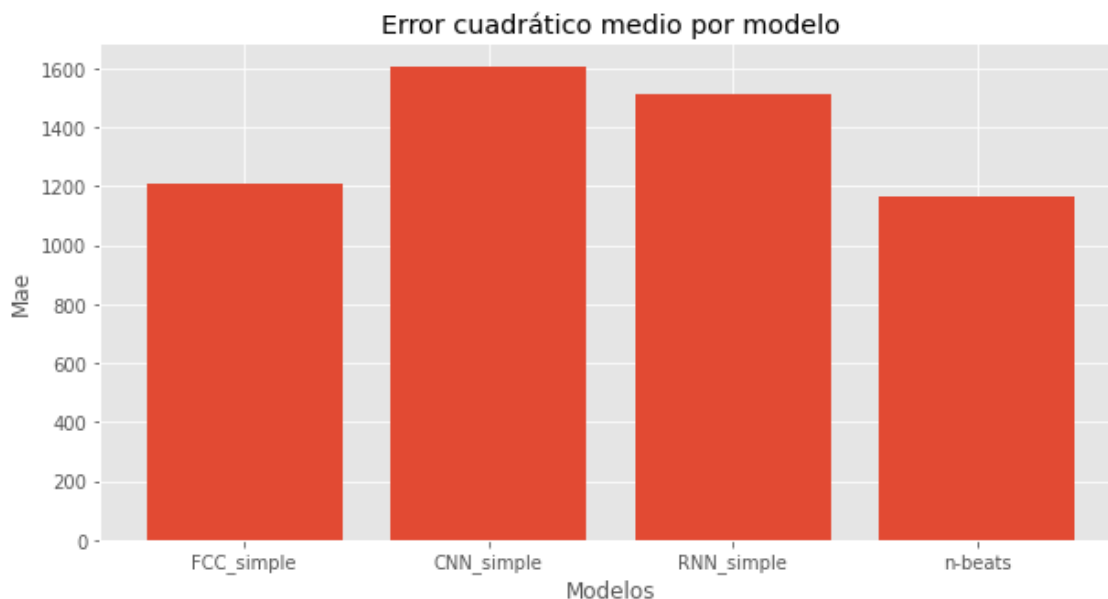
#### 4.2.5.5. Comparación de los modelos

Para llevar a cabo la comparación de los modelos descritos anteriormente, se comparó el error cuadrático medio obtenido por cada uno en el mismo conjunto de datos de validación.

Gracias al *callback ModelCheckpoint*, se aseguró que las versiones guardadas de cada uno de los modelos fuese la que tuviese un error cuadrático menor.

Cabe recordar que, para la siguiente gráfica, el resultado óptimo es el resultado con el valor más bajo.

Los resultados fueron los siguientes:



*Ilustración 33. Comparación de error de modelos. Elaboración propia*

Como se puede ver en la gráfica, el modelo con el error cuadrático menor y, por tanto, el más indicado para la labor de predicción de precios de criptomonedas con datos univariados es el modelo implementado usando el algoritmo *N-BEATS*.

### 4.3. Serie temporal multivariable

#### 4.3.1. Obtención de los datos

El conjunto de datos usado en este análisis, al igual que el conjunto usado para el análisis en diario, ha sido obtenido directamente de la *API* de la casa de cambios “*coinbase*”. Sin embargo, y al tratarse de un conjunto más detallado, fue usada una biblioteca distinta a la anterior, la cual nos permite acceder a la *API* pro de *coinbase* de forma completamente gratuita.

El código fuente de dicha librería se puede encontrar en el siguiente enlace: [https://github.com/David-Woroniuk/Historic\\_Crypto](https://github.com/David-Woroniuk/Historic_Crypto).

Al tratarse de un conjunto de datos bastante más voluminoso que el usado para el conjunto de diario (alcanzando casi el millón de entradas), el proceso de carga de estos datos se realiza en dos etapas, para lo cual se creó un cuaderno aparte.

En la primera etapa, se obtienen los *datasets* históricos desde la primera fecha de la que se tiene constancia (en el caso de *bitcoin*, el 20 de julio de 2015). Estos datos se almacenan en diferentes archivos, los cuales abarcan seis meses, y se guardan en una misma carpeta. La razón por la que los datos se guardan en archivos separados en lugar de en un único archivo es que si el proceso de obtención de datos es muy largo y pesado, este se interrumpe, dejándonos sin datos. Dejar estos en archivos de 6 meses son el compromiso perfecto entre tiempo de carga y tamaño de archivo.

La segunda etapa tiene lugar en el mismo script en el que se desarrolla la red neuronal. En esta etapa, se leen todos los archivos escritos en la

primera etapa, se identifica el último minuto y se hace una petición a la *API* de *coinbase* para obtener la información desde el último minuto de nuestros archivos hasta el momento actual, para después concatenar estos *datasets*.

Esto nos permite que la ejecución en el script dedicado al desarrollo de la red neuronal lleve el menor tiempo posible, y también poseer la mayor cantidad de información para el entrenamiento de la misma.

## 4.3.2. Formateo de los datos

### 4.3.2.1. Valores perdidos

Como se ha expuesto en el punto anterior, durante la ejecución del script dedicado a la red neuronal se disponen de distintos *datasets* guardados en una carpeta, así como de otro con los datos complementarios que serán proporcionados por la *API* de *coinbase*. Estos datos consisten en los precios de entrada, salida, máximos y mínimos y el volumen de bitcoin cada cinco minutos.

Sin embargo, la *API* puede presentar algún error y no devolver todos los minutos necesarios.

En este apartado se explican los procesos realizados para la determinación de minutos perdidos y su posterior procesamiento.

En primer lugar y tal como se explicó en el apartado anterior, se realizó la lectura de todos los *datasets* obtenidos anteriormente, para posteriormente concatenarlos usando la función correspondiente de *pandas*.

```
df_list = [get_dataset(f'Bitcoin_{i}.csv') for i in range(1, len(os.listdir(FOLDER_NAME)) + 1)]

minutal = pd.concat(df_list, axis=0)
minutal.sort_index(inplace=True)
```

*Ilustración 34. Lectura y concatenación de datasets. Elaboración propia*

Después, se procedió a la extracción del último minuto de la última entrada para, finalmente, obtener los datos correspondientes desde este último minuto al momento de ejecución, tal y como se muestra en la siguiente ilustración:

```
last_minute = minutal.tail(1).index.item()
last_minute = last_minute.strftime('%Y-%m-%d-%H-%M')
last_minute

'2022-08-25-00-00'

last_data = HistoricalData('BTC-USD', 300, start_date=last_minute).retrieve_data()
```

*Ilustración 35. Extracción de último minuto y obtención de datos complementarios. Elaboración propia*

Una vez obtenidos los datos complementarios, se procedió a la concatenación de estos con el *dataset* original, con la función “pandas.concat” como se ha mostrado anteriormente.

Con todos los datos en forma de *DataFrame*, se procedió a la creación de otro, cuyos índices estarían definidos por las fechas de nuestro *dataset* original. Estos índices consisten en fechas, desde el primer hasta el último minuto de los obtenidos en nuestro conjunto de datos original, con un salto de 5 minutos.

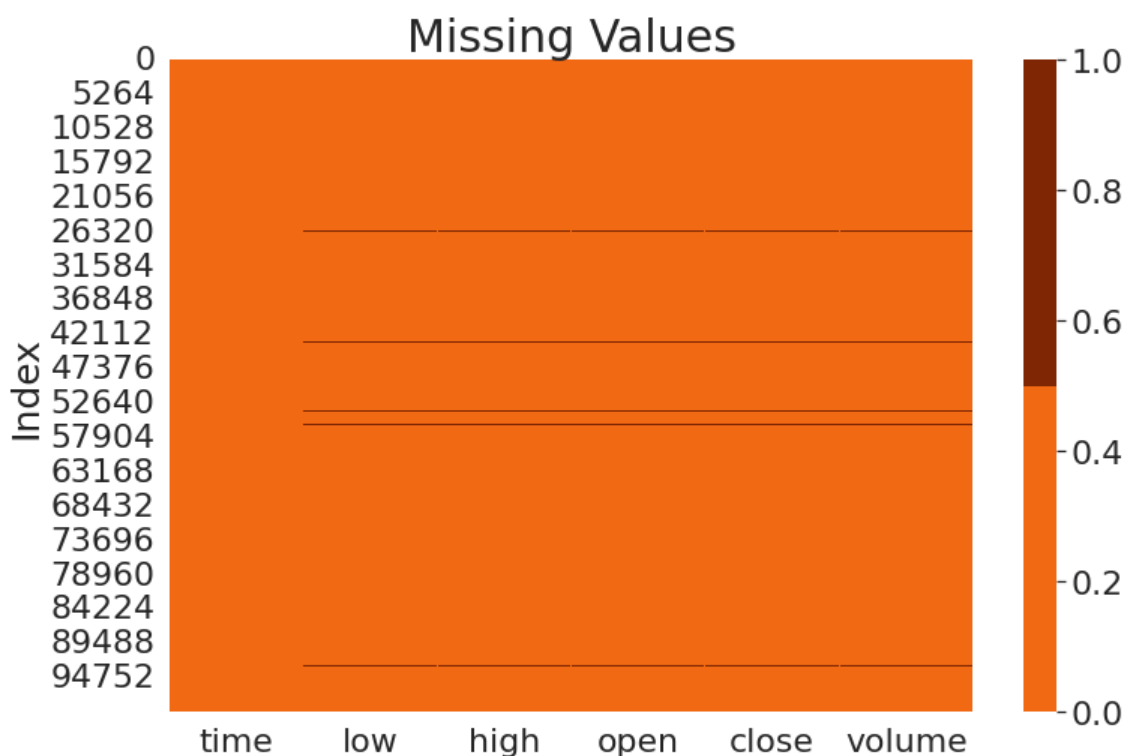
```
ts_index = pd.date_range(start=minutal.head(1).index.item(),
                        end=minutal.tail(1).index.item(),
                        freq='5T')
```

```
all_minutes = pd.DataFrame(index=ts_index).rename_axis("time")
```

*Ilustración 36. Creación de DataFrame con todas las fechas necesarias. Elaboración propia*

Una vez creado este *DataFrame*, se procedió a juntarlo con el original usando un *inner join*. Esto permite que los valores obtenidos queden intactos, pero que los valores que no existan en el conjunto original figuren como nulos.

Después de realizar esta unión de conjuntos de datos, fueron imputados los valores perdidos de los mismos.



*Ilustración 37. Valores perdidos del conjunto de datos. Elaboración propia*

Como se puede observar, en el conjunto de datos utilizado existen valores perdidos, aunque estos constituyen una parte ínfima del conjunto total.

Dadas las circunstancias, la mejor forma de afrontar los valores perdidos en una serie temporal como la actual es con la función “ffill”. Esta función simplemente complementa estos valores con los valores de las entradas vecinas, lo cual no influye en las características principales de este conjunto de datos, cuyo cálculo se explica en el siguiente punto.

### 4.3.2.2. Preprocesamiento de características

Como se ha explicado en los primeros puntos de este apartado, existe una gran cantidad de características que se pueden extraer de los precios en los periodos de tiempo obtenidos del conjunto de datos original.

En periodos de tiempo tan cortos, lo más importante es predecir no el precio en un instante de tiempo determinado, sino si este subirá o bajará en dicho instante de tiempo. Esto permite ganar dinero en distintos *exchanges* descentralizados, los cuales permiten apostar criptomonedas al hecho de si la cotización de *bitcoin* subirá o bajará en los próximos cinco minutos.

Es por ello que la variable elegida para realizar las predicciones fue la del incremento (diferencia entre el precio de subida y el precio de bajada).

Las variables con las cuales fue realizado el análisis fueron las siguientes:

- Incremento: Como se explicó anteriormente, se trata de la diferencia del precio de apertura y de cierre en un día concreto. Para este análisis, esta variable se guarda como booleana, utilizando “True” para el caso en el que este incremento haya sido positivo y “False” para el caso contrario.
- Error (Gap): Esta variable se usa para medir el error de cálculo entre una entrada en los datos y la anterior. En estos casos, se supone que el precio de entrada de un día debe coincidir con el anterior, pero con estos datos no siempre es así, por eso se tomó la decisión de incluir esta variable.
- Volumen: Esta variable indica el volumen en dólares de movimientos de mercado en un periodo de tiempo.
- Autocorrelación: esta variable indica el coeficiente de autocorrelación, explicado en el primer punto de este apartado.
- Mayor diferencia: indica la diferencia entre el punto más alto y el más bajo en cierto periodo de tiempo, lo cual da información sobre la volatilidad en el mismo.
- Promedio móvil (EMAD): los promedios móviles dan información sobre la tendencia de una gráfica.
- Oscilador estocástico (*Stoch*): esta variable ayuda a identificar los periodos de sobreventa o sobrecompra.
- Volatilidad: información de la volatilidad en ciertos periodos, los cuales se pueden indicar como hiperparámetros.
- Días hasta *halving* (DTH): indica la cantidad de días que faltan hasta que se produzca un *halving*, lo cual nos da información sobre el momento de ciclo de *bitcoin* en el que se encuentra una instancia de tiempo.

- Tamaño de bloque (BS): esta variable indica el número de monedas que entran en circulación cada 10 minutos en cierto instante de tiempo, lo cual da información sobre el número de ciclo en el que se encuentra una instancia de tiempo.

Estas variables fueron normalizadas para que sus valores se encuentren en el intervalo de cero a uno, de forma que la red neuronal les de la misma importancia a todos.

Tras todas estas modificaciones, el *DataFrame* sobre el cual se realizó el proceso de entrenamiento quedó de la siguiente manera:

time	volume	Autocorrelacion	Gap	Increment_up	Highest_difference	EMAD	Stoch	Volatility	DTH	BS
2015-07-21 00:05:00	0.008943	0.644582	0.000516	0.0	0.003903	0.599819	0.902478	0.000180	0.251963	1.0
2015-07-21 00:10:00	0.003218	0.642158	0.001033	0.0	0.002789	0.599806	0.722205	0.000136	0.251963	1.0
2015-07-21 00:15:00	0.014791	0.599730	0.009819	0.0	0.006137	0.599791	0.442133	0.000124	0.251963	1.0
2015-07-21 00:20:00	0.004212	0.595214	0.005686	1.0	0.000223	0.599782	0.252451	0.000110	0.251963	1.0
2015-07-21 00:25:00	0.012049	0.601137	0.001033	1.0	0.000335	0.599774	0.172117	0.000088	0.251963	1.0
...	...	...	...	...	...	...	...	...	...	...
2022-08-26 11:50:00	0.023296	0.496248	0.001166	1.0	0.004919	0.581197	0.751198	0.006518	0.394718	0.0
2022-08-26 11:55:00	0.026247	0.506239	0.000279	1.0	0.004881	0.583008	0.847739	0.007168	0.394718	0.0
2022-08-26 12:00:00	0.029676	0.497621	0.001063	0.0	0.008555	0.583664	0.699017	0.005595	0.394718	0.0
2022-08-26 12:05:00	0.022048	0.509364	0.002545	1.0	0.004605	0.584970	0.551442	0.005098	0.394718	0.0
2022-08-26 12:10:00	0.003204	0.507631	0.000286	0.0	0.002778	0.585849	0.349315	0.004716	0.394718	0.0

Ilustración 38. *DataFrame* preparado para entrenamiento. Elaboración propia

### 4.3.3. Construcción del modelo

Una vez preparados los datos, se llevó a cabo la construcción del modelo y su posterior entrenamiento.

Para el entrenamiento del mismo, se decidió crear ventanas, tal y como se mostró en la explicación del modelo univariable. En este caso, las ventanas poseen características no solo del precio, sino todas las expuestas en el punto anterior. El proceso de creación de estas ventanas a partir del *DataFrame* mostrado en la ilustración 34 fue el siguiente:

```
def get_label_values_from_dataframe(df, window_size=WINDOW):
    res = df.copy()
    y = df.iloc[window_size:]["Increment_up"].to_numpy()
    x = np.array([res.iloc[i: window_size + i].to_numpy() for i in range(len(res) - window_size)])
    return x, y
```

Ilustración 39. Creación de ventanas multivariantes. Elaboración propia



En esta ocasión, estas ventanas consisten en datos bidimensionales, cuyas propiedades espaciales pueden ser aprovechadas. Es por ello que se utilizó una arquitectura típica de red convolucional.

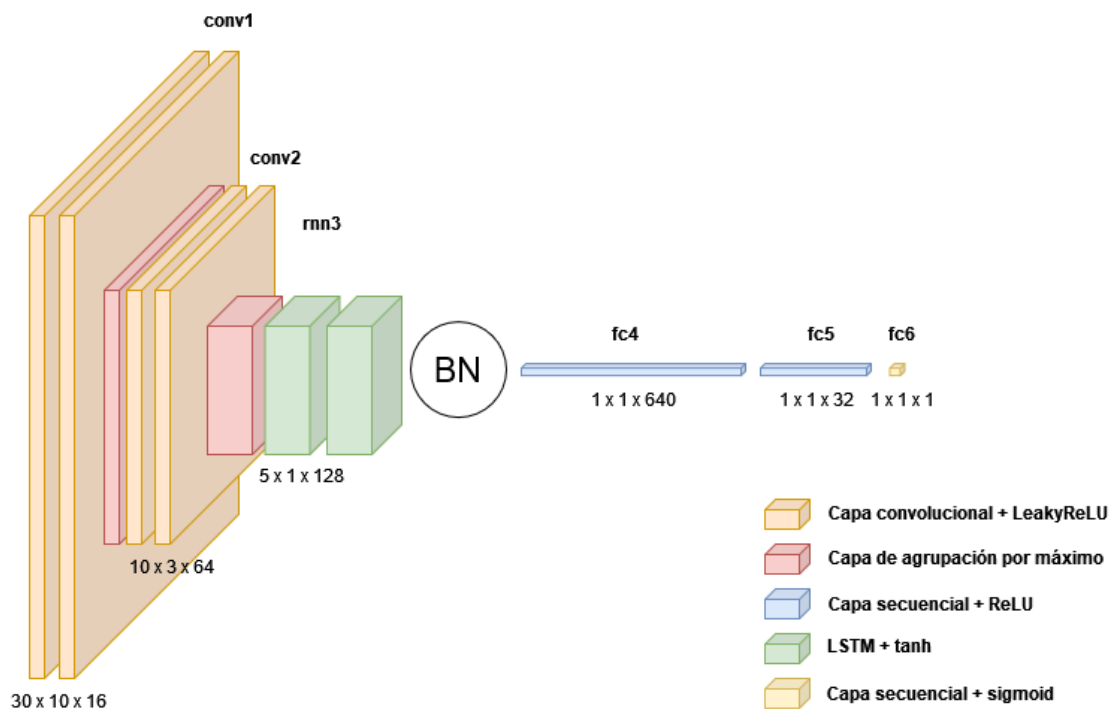


Ilustración 40. Esquema de la red neuronal utilizada. Elaboración propia

En esta ocasión, los valores de entrada son una matriz con el número de líneas correspondiente al número de elementos de la ventana seleccionada y con el número de columnas correspondiente a la cantidad de características de nuestro *DataFrame* (en este caso, diez).

Como es propio de las arquitecturas de redes neuronales convolucionales, la primera capa es una capa convolucional. Con el objetivo de reducir la complejidad mientras se mantiene la información espacial, las capas siguientes tendrán una dimensionalidad menor pero un mayor número de filtros, estirando el espacio latente en una única dirección.

Uno de los problemas de las capas de características es que son sensibles a la ubicación de las características de entrada. Una de las maneras de solucionar estos problemas es reducir la muestra del mapa de características.

La ventaja que nos ofrece este método es que, gracias al mismo, los mapas de características submuestreadas resultantes son más resistentes a los cambios de posición de la característica en la imagen, lo que se denomina “invariancia de traducción local”.



En el modelo utilizado en el presente Trabajo de Fin de Máster, este método es llevado a cabo con la utilización de capas de agrupación (también conocidas como capas de *pooling*). Estas capas proporcionan un enfoque para el muestreo descendente de capas de características al resumir la presencia de características en parches del mapa de características.

Los dos métodos de agrupación más usados son la agrupación por promedio y la agrupación por máximo, siendo este último el usado para el modelo presentado.

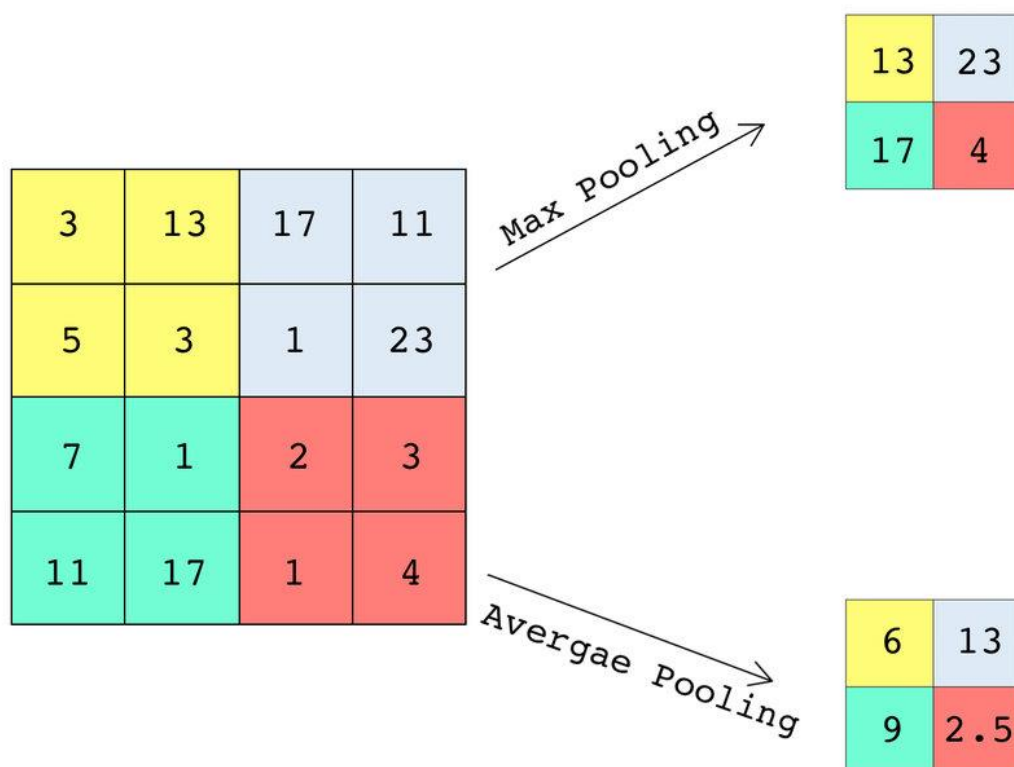


Ilustración 41. Ejemplo de capas de agrupación de media y de máximo. Fuente: [https://www.researchgate.net/figure/Example-of-max-pooling-and-average-pooling-operations-In-this-example-a-4x4-image-is\\_fig4\\_332092821](https://www.researchgate.net/figure/Example-of-max-pooling-and-average-pooling-operations-In-this-example-a-4x4-image-is_fig4_332092821)

Estas capas consisten en dividir el mapa de características en zonas concretas y extraer elementos concretos de cada una de las mismas. En el caso de la capa utilizada para este modelo, el elemento extraído de cada una de las zonas es el máximo.

Tras este bloque convolucional, en nuestro modelo se utiliza una capa recurrente con una unidad LSTM. Esta unidad proporciona la capacidad de aplicar memorización de ciertas características de los datos, gracias a su celda de memoria a largo plazo, lo cual resulta indispensable para la detección de ciclos.

Posteriormente, tras este bloque se realiza una normalización de la información con el objetivo de evitar sobreentrenamiento y se procede al aplanado de la misma, con lo que se concluye el modelo base, cuya función es la extracción de características.

Tras esta capa, se procede al aplanamiento del espacio latente en una única dirección, paso indispensable en este tipo de arquitecturas para poder pasar al modelo superior.

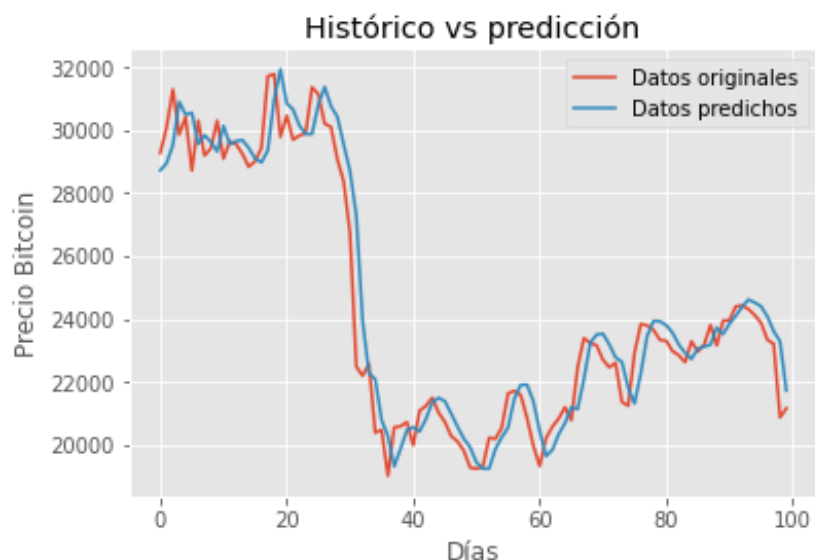
Esta última parte consiste en dos capas secuenciales totalmente conectadas, encargadas de la clasificación.

Con el fin de evitar linealidad, estas capas están activadas con la función de activación *LeakyReLU*. Finalmente, están conectadas a una última capa de una única neurona activada con la función *sigmoid*, propia de problemas de clasificación binaria.

## 5. Resultados

### 5.1. Predicción de mercado

Gracias a las comparaciones de modelos para predicción de mercado con datos univariados llevadas a cabo anteriormente, se pudo llegar a la conclusión de que el mejor modelo para este tipo de análisis es el modelo basado en el algoritmo *N-BEATS*.



*Ilustración 42. Predicciones modelo N-BEATS sobre conjunto de datos de validación. Elaboración propia*

Como se puede observar, este modelo se adapta bastante bien al conjunto de datos de prueba, por lo que es un buen candidato para realizar predicciones a medio plazo.

Una vez determinado el modelo más eficiente, este modelo fue entrenado con la totalidad de los datos del *dataset*, para poder realizar predicciones en el futuro.

Para llevar a cabo este tipo de análisis, fue usada la estructura de datos de ventana, explicada en el apartado de creación de modelos para análisis univariable.

Por sus características, el modelo N-BEATS es capaz de predecir precios futuros a partir de un número de precios pasados igual al tamaño de la ventana. Para realizar estas predicciones, el primer paso es usar una ventana con datos pasados para realizar una predicción. Después, esta propia predicción es introducida en la ventana, la cual por sus propiedades elimina el día más lejano para mantener siempre el mismo tamaño.

```
def predict_future(model, window, days):
    future_prices = np.array([], dtype=np.float32)
    for i in range(days):
        next_day = model.predict(np.expand_dims(window, axis=0))[:, 0]
        future_prices = np.append(future_prices, next_day)
        window = np.append(next_day, window[:-1])
    start_date = date.today() + timedelta(days=1)
    dates = pd.date_range(start=start_date, periods=days)
    return pd.DataFrame({"Prices": future_prices}, index=dates)
```

Ilustración 43. Realización de la estructura de ventana para realizar predicciones en el futuro. Elaboración propia

Este proceso se repite el número de días que se desee predecir el precio de *bitcoin* en el futuro. Por su parte, cada uno de estos resultados se guarda en una lista de *numpy*, con la que finalmente se creará un *DataFrame*, el cual será devuelto al usuario.

En el código, se realizaron estas predicciones para un plazo de mil días en el futuro, y los resultados fueron los siguientes:

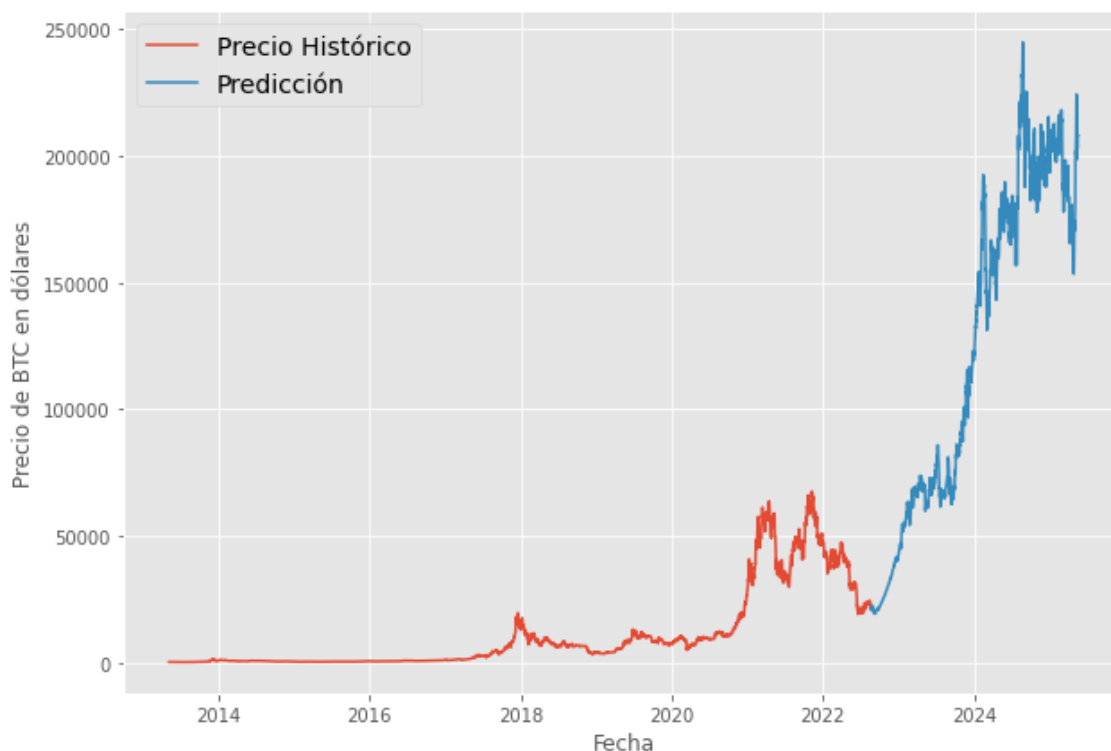


Ilustración 44. Predicción de precios de bitcoin en el futuro con N-BEATS. Elaboración propia

Como se puede observar, el modelo *N-BEATS* creado y entrenado en el presente Trabajo de Fin de Máster predice que en el próximo ciclo, el precio máximo de bitcoin alcanzará prácticamente los 250 mil dólares, para después tener una fuerte caída.

## 5.2. Análisis multivariable

### 5.2.1. Tratamiento del sobreentrenamiento

El sobreentrenamiento es un fenómeno en el que el modelo que se enseña reconoce bien los ejemplos de un conjunto de aprendizaje, pero no reconoce o reconoce mal cualquier otro ejemplo que no haya participado en el proceso de aprendizaje (es decir, que se le presente en el proceso de uso práctico).

Este fenómeno es el resultado de un ajuste excesivo de los parámetros del modelo a las dependencias contenidas en el conjunto de entrenamiento. Si se produce un reentrenamiento, el modelo no adquiere la capacidad de generalización: la capacidad de extender las dependencias y patrones encontrados en el conjunto de entrenamiento a nuevos datos. Tal modelo resulta inútil en la práctica, incluso si su error de aprendizaje es pequeño.

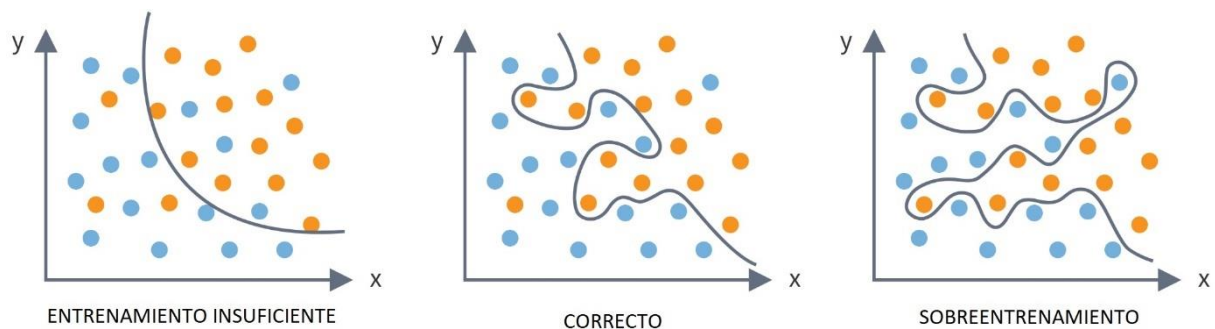


Ilustración 45. Ejemplos de entrenamiento insuficiente, entrenamiento correcto y sobreentrenamiento.

Fuente: <https://wiki.loginom.ru/articles/overtraining.html>

El problema del sobreentrenamiento es, en diversos grados, común a todo tipo de modelos aprendidos, pero es especialmente grave para las redes neuronales. Durante el entrenamiento, los pesos de la red neuronal se ajustan de manera que la red convierte las entradas a las salidas deseadas de acuerdo con las dependencias encontradas en los datos de entrenamiento.

Sin embargo, en el caso de que una red se entrene demasiado tiempo o usando demasiados parámetros (pesos), se produce un sobreentrenamiento. Esto se debe a que, a partir de cierto momento, la red comienza a ajustarse no a las dependencias generales en los datos, sino a las características de los ejemplos individuales, que pueden contener valores anormales, errores, etc.

Como consecuencia, la red comienza a probar las nuevas observaciones que se le presenten, no para determinar la conformidad de la dependencia, sino para que coincidan con ejemplos individuales del conjunto de entrenamiento. Como resultado, el modelo solo puede reconocer una nueva observación si coincide con uno de los ejemplos de entrenamiento.

Uno de los métodos más sencillos para detectar un sobreentrenamiento de una red neuronal es el seguimiento de los datos de la función de pérdida durante el proceso de entrenamiento. En un entrenamiento correcto, los resultados de la función de pérdida de los datos de validación disminuyen en sintonía con los resultados de la función de pérdida de los datos de entrenamiento.

En un caso de sobreentrenamiento, se puede observar que los resultados de la función de pérdida de los datos de validación se mantienen estables mientras que los resultados de esta función con los datos de entrenamiento disminuyeron constantemente.

Durante el desarrollo de este modelo en específico, hubo problemas de sobreentrenamiento, los cuales fueron resueltos con las siguientes técnicas:

- Dropout: esta técnica consiste en eliminar arbitrariamente una cantidad específica de neuronas de una capa de la red neuronal (Srivastava, Krizhevsky, & Salakhutdinov, 2014). Dicha cantidad se especifica como parámetro de esta técnica en forma de porcentaje. Esta eliminación consiste en eliminar todas las conexiones de una neurona en específico con las capas anteriores, de forma que esta siempre devuelve cero.
- Batchnormalization: la base de esta técnica radica en aplicar una normalización de los datos después de una capa de la red neuronal (Ioffe & Szegedy, 2015). Esta normalización consiste en restar la media y dividir entre la desviación típica de los datos del batch, después de la función de activación de una capa, para evitar en un momento dado que durante la propagación hacia adelante se traspasen datos en algunas neuronas demasiado grandes que se impongan sobre el resto.

Estos métodos y sus respectivos parámetros fueron elegidos de manera experimental, y ayudaron a resolver el problema de sobreentrenamiento, pues después de su aplicación los resultados de la función de pérdida sobre los datos de validación disminuyeron en sintonía con la aplicación de esta función sobre los datos de entrenamiento.

## 5.2.2. Precisión del modelo

El siguiente análisis fue llevado a cabo sobre los datos de validación y, durante el transcurso del mismo, fueron realizadas numerosas modificaciones con el objetivo de aumentar la precisión del modelo.

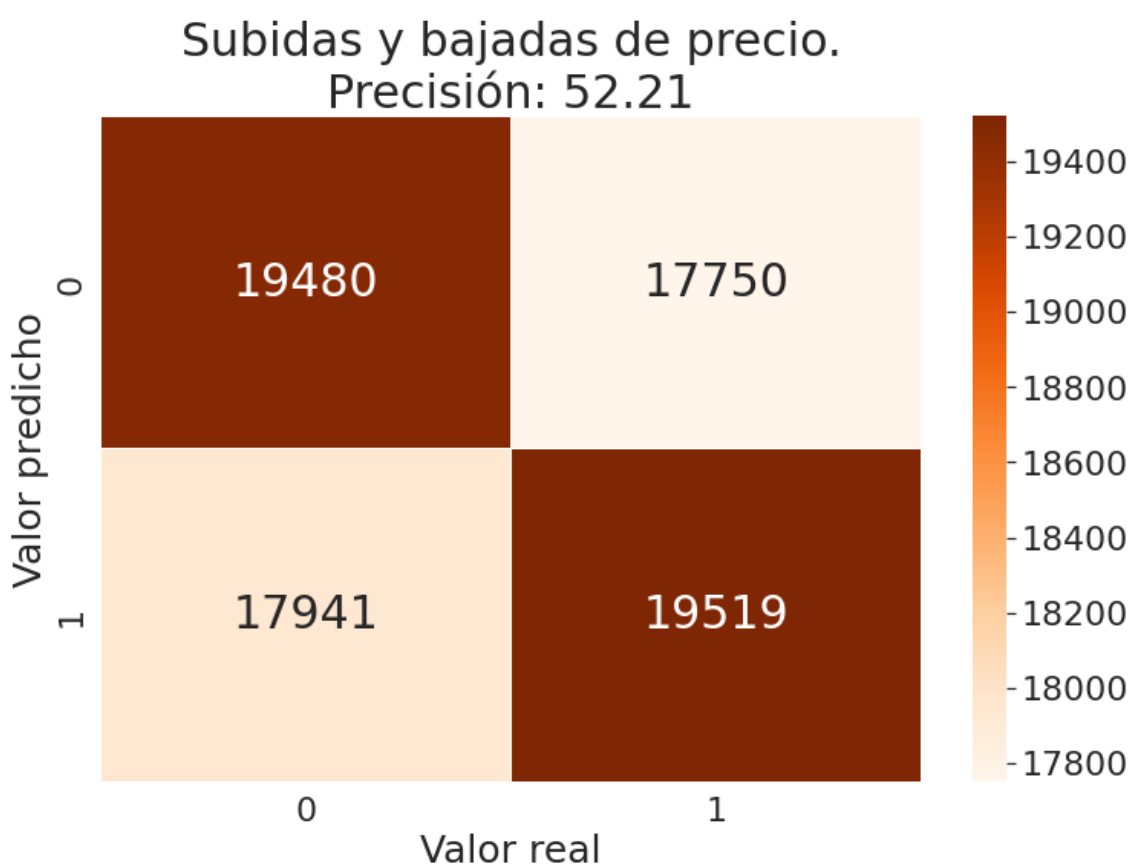
Este modelo fue diseñado para resolver un problema de clasificación binaria, donde las dos clases existentes corresponden con movimientos alcistas o bajistas en el mercado, siendo el cero la indicación de una disminución en el precio del activo analizado y el uno la indicación de un incremento en el precio del mismo.

Cabe aclarar que los análisis de mercado de activos financieros muestran que estos tienen un comportamiento impredecible, y diferentes trabajos han mostrado una exactitud ligeramente superior al 50%, por lo que esta es la exactitud objetivo de este análisis.

Para la creación de esta matriz de confusión fue utilizada la herramienta propia de la biblioteca *sklearn*, la cual genera dicha matriz automáticamente introduciendo como parámetros, por un lado, el conjunto de etiquetas de validación y, por otro, el conjunto de etiquetas predicho por el modelo cuya precisión se pretende analizar.

Por otro lado y para el graficado de la misma, se decidió hacerlo en un mapa de calor, a fin de maximizar la información proyectada gráficamente según recomendado por las técnicas de visualización de datos. Para la elaboración de este gráfico, fue usada la propia herramienta de mapa de calor de la biblioteca *seaborn*, en conjunto con las figuras de la biblioteca *matplotlib*, de cara a maximizar las posibilidades de personalización del gráfico.

La red neuronal convolucional creada fue puesta a prueba sobre el conjunto de validación, y los resultados se pueden observar en la siguiente matriz de confusión:



*Ilustración 46. Matriz de confusión del modelo convolucional sobre los datos minutales. Elaboración propia*

Como se puede observar, la precisión alcanzada supera el 50%, por lo que el modelo creado puede ser usado para la elaboración de distintas estrategias a corto plazo.

Cabe destacar que durante todo el transcurso de los experimentos, en todos y cada uno de ellos este modelo dio una precisión superior al 52%.

Este mismo análisis fue llevado a cabo para el conjunto de datos en diario, donde se obtuvieron los siguientes resultados.



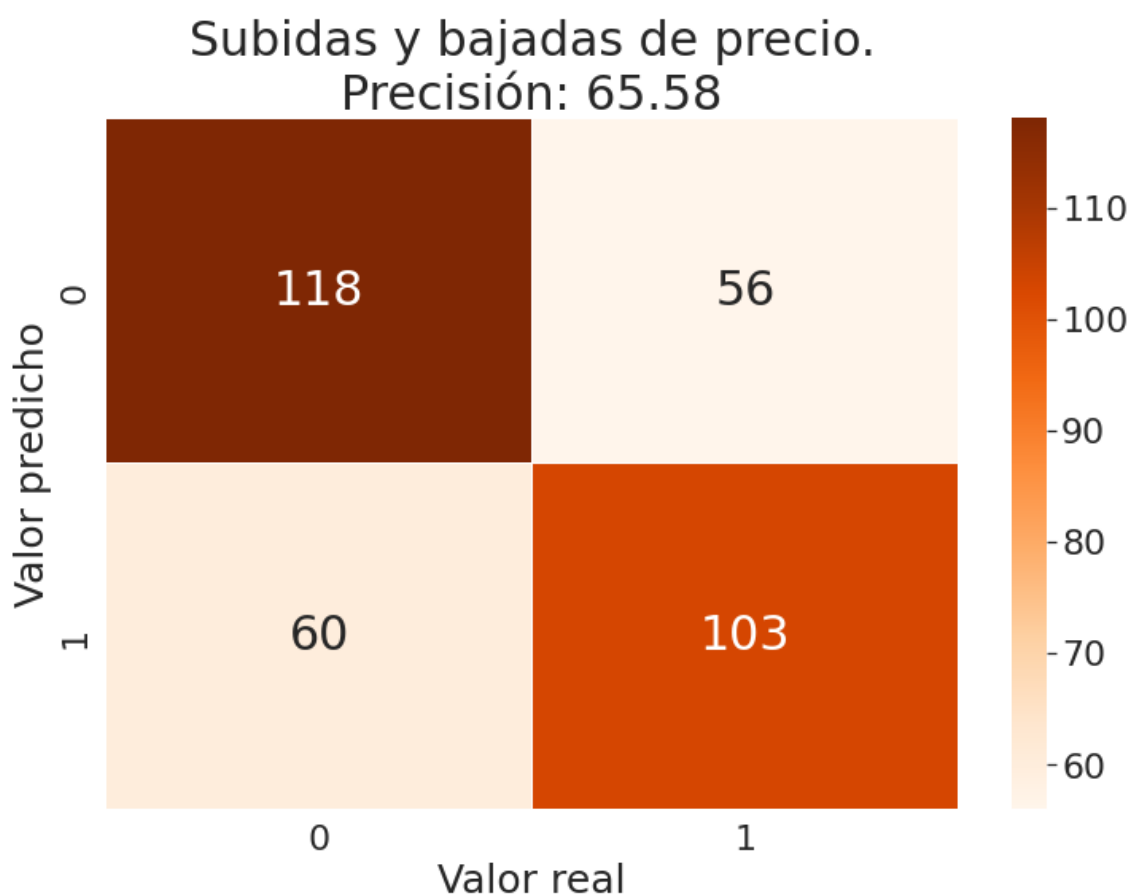


Ilustración 47. Matriz de confusión del modelo convolucional sobre los datos diarios. Elaboración propia

Estos resultados son bastante más precisos que los resultados del análisis en el *dataset* de precios cada cinco minutos, lo cual indica que en esta temporalidad las características determinantes para este tipo de comportamientos en el precio están más destacadas.

### 5.2.3. Importancia de las variables

Uno de los principales objetivos del análisis llevado a cabo durante el presente Trabajo de Fin de Máster consistió en la determinación de los valores de mercado más importantes los cuales pueden ayudar a realizar este tipo de predicciones.

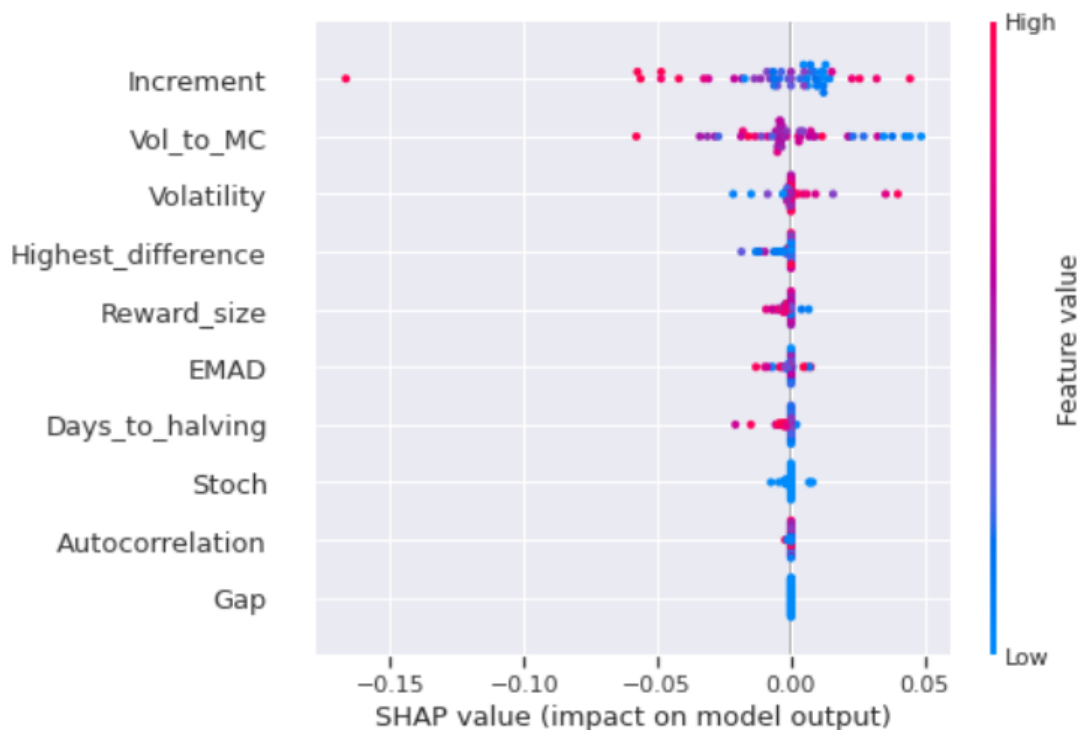
Por desgracia y a diferencia de la biblioteca de aprendizaje automático *sklearn*, *tensorflow* no cuenta con ninguna herramienta de forma nativa que permita llevar a cabo un análisis de importancia de variables.

Sin embargo, hay usuarios que han desarrollado distintas herramientas con este fin. La herramienta usada en este caso fue la llamada *shap*, la cual se puede encontrar por este enlace:

<https://github.com/slundberg/shap>

Los resultados obtenidos fueron los siguientes:

```
shap.summary_plot(shap_values = shap_values[0],
                  features = df_original.iloc[2000:2050,:])
```



*Ilustración 48. Importancia de valores. Elaboración propia.*

Como se puede observar a partir de la ilustración, el parámetro que más peso tuvo en los resultados del modelo final fue el incremento, lo cual en gran parte explica el éxito del análisis monovariable en el apartado anterior.

Asimismo, el volumen demostró también tener una gran importancia, lo cual muestra la gran susceptión de este mercado a compras y ventas agresivas.

Dentro de las variables de menor importancia, podemos encontrar la brecha entre los datos, así como el coeficiente de autocorrelación, por lo que en futuros análisis será posible obviar estos.

## 6. Conclusiones

La identificación de patrones en las variaciones de precio de los diferentes instrumentos en los mercados financieros permite construir un modelo para predecir su movimiento futuro. La detección de patrones explícitos y ocultos, utilizando redes neuronales artificiales, puede contribuir al desarrollo del proceso de toma de decisiones comerciales.

A lo largo del estudio llevado a cabo en el presente Trabajo de Fin de Máster, han sido llevados a cabo diferentes tipos de análisis de series temporales, univariantes y multivariantes, en distintas temporalidades y han sido probados distintos modelos para cada caso.

Fueron definidos los tipos óptimos de arquitectura de red neuronal artificial para la predicción de datos financieros.

La práctica sobre series temporales univariantes ha demostrado que el modelo *N-BEATS* supera a cualquier modelo simple en este tipo de análisis, aunque la diferencia no es sustancial. La precisión con la que este modelo se ha comportado en el modelo de validación muestra su gran efectividad para análisis a corto y medio plazo. De todas formas, a medida que las predicciones se adentran en el futuro, la precisión de este modelo disminuye, ya que las predicciones son realizadas sobre datos predichos y, por tanto, en este tipo de análisis el modelo arrastra el error, por lo que este tipo de modelos de ventana deslizante resultan ser poco fiables en análisis a medio y largo plazo.

Sin embargo, en la práctica también se ha demostrado que modelos más simples, como el *FCC*, también muestran buenos resultados a corto plazo.

Por otro lado, durante el análisis de la serie temporal multivariantes se ha mostrado la imposibilidad de predecir con exactitud movimientos en el precio de distintos activos. Esto no es de extrañar, ya que distintos autores han llegado a la misma conclusión tras realizar análisis similares. La mayor precisión mostrada en la práctica es inferior al 60% para modelos de especulación intradía.

Durante el desarrollo del trabajo, se dio con el tipo óptimo de arquitectura de red neuronal artificial para la predicción de datos financieros en este tipo de series temporales. Se trata de un híbrido de una red convolucional y recursiva, el cual no solo reduce la dimensión de los datos de entrada manteniendo la información espacial de los mismos, sino que adicionalmente captura el orden temporal.

En cambio, el análisis en diario de la serie temporal con las mismas características y con un modelo similar se han obtenido resultados de precisión superiores al 60%, lo cual demuestra que este tipo de series contienen más información sobre la tendencia de la misma.

Al realizar estos modelos con los datos iniciales, se pudo ver que los resultados obtenidos eran poco precisos, por lo que se procedió a la inserción de datos externos.

Las variables que resultaron tener mayor peso fueron las referentes a los incrementos diarios y el volumen, lo cual demuestra por un lado la importancia de la evolución de mercado y, por otro lado, la influencia de compraventas masivas en el mercado.

También mostraron una importancia sustancial las variables referentes al *halving*, lo cual evidencia la naturaleza cíclica del precio de *bitcoin*.

La conclusión más importante es que se han mostrado los beneficios de las redes neuronales para el análisis de mercado, y se han sentado las bases para la creación de modelos cada vez más exactos. También se ha demostrado que la precisión de estos modelos aumenta de forma paralela a la temporalidad, de modo que no es aconsejable realizar operaciones en corto con la ayuda de estos modelos.

## 7. Opiniones y desarrollos futuros

Uno de los primeros problemas a resaltar es el de rendimiento. Debido al gran número de registros, los cálculos en una máquina en local se hacen pesados y largos, llegando la ejecución del cuaderno dedicado al análisis minutal multivariable a superar la hora. Por ello, para la realización de análisis similares, es recomendable utilizar las herramientas proporcionadas por *Google Colabs* para ejecución en la nube.

Por otro lado, este análisis no tuvo en cuenta los sentimientos de mercado. Existe un indicador para este parámetro, el llamado “índice de miedo y codicia”. Lo malo de este indicador es que solo tiene registros desde el año 2018, por lo que de usarse no se podría aprovechar completamente. En el futuro, habrá registros suficientes de este parámetro como para poder incluirlo en un análisis.

Otro punto importante a desarrollar es la utilidad. Se podrían usar modelos entrenados y desarrollados en este análisis para automatizar un sistema de compraventas de criptoactivos que interactúen directamente con la API de alguna bolsa de intercambio.

Existen bolsas de intercambio de criptomonedas descentralizadas, las cuales ofrecen un servicio de apuestas sobre la variación del precio de una criptomoneda en un tiempo determinado. Se puede apostar a que, por ejemplo, en cinco minutos el precio de una criptomoneda concreta va a subir (o a bajar), y en el caso de acertar se duplicaría el dinero apostado, mientras que en el caso de fallar se perdería todo. Ya que el modelo usado en este trabajo es justamente de clasificación binaria orientado a estos movimientos en el precio, este apartado es donde este permitiría obtener una mayor rentabilidad.

Es imperativo el desarrollo de estrategias con estos modelos y de la automatización de las mismas que las permita ejecutarlas interactuando con las *APIs* de *exchanges* descentralizados.

## 8. Apéndices

En el siguiente repositorio de GitHub se podrán encontrar los diferentes ficheros utilizados para la realización del presente Trabajo de Fin de Máster.

[https://github.com/CarlosDLMC/Trabajo\\_Fin\\_Master](https://github.com/CarlosDLMC/Trabajo_Fin_Master)

Dichos ficheros incluyen:

- Un cuaderno (*jupyter notebook*) con el análisis multivariable de una serie temporal en temporalidad diaria.
- Un cuaderno con el análisis multivariable de una serie temporal en temporalidad cinco minutos.
- Un cuaderno con el análisis univariable de una serie temporal en temporalidad diaria, incluyendo la comparación de cuatro modelos de redes neuronales.
- Un cuaderno con el código utilizado para obtener los conjuntos de datos de la serie temporal en temporalidad cinco minutos.
- Un cuaderno con el código utilizado para realizar predicciones a medio plazo usando el modelo *N-BEATS*.
- Una carpeta con todos los modelos utilizados.
- Una carpeta con el conjunto de datos usado para llevar a cabo el análisis de la serie temporal en temporalidad cinco minutos.

## 9. Bibliografía

- Afanasiev, V. (2010). Análisis de series temporales y pronósticos. *Infra-M*, 320.
- Anatolyev, S. (2013). Objetos de modelado de series temporales no estructurales. *Kvantil*, 1-11.
- Bolshakov, A. (2014). Métodos de procesamiento de datos multidimensionales y series temporales. *Goriachaya Liniya - Telekom*, 522.
- Brillinger, D. (1981). *Time series: Data analysis and theory*. Toronto: Holden-Day.
- Chapelle, O., Vapnik, V., & Bousquet, O. (2002). Choosing Multiple Parameters for Support Vector Machines. *Springer*, 46.
- Chuchueva, I. (2012). Métodos de pronóstico de series temporales por muestra de similitud máxima. *Nauchnyi Zhurnal*, 156.
- Devyatov, A. (2008). Modelación y pronóstico de la economía de Rusia. *Escuela de economía de Rusia*, 341.
- Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. Paris: O'Reilly Media.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning (Adaptive Computation and Machine Learning Series). *Boston: The MIT Press*, 775.
- Graham, B. (2008). *Security Analysis*. London: McGraw Hill.
- Hamme, B. (2001). Learning with Recurrent Neural Networks. *Lecture Notes in Control and Information Sciences*, 145.
- Haykin, S. (2008). *Neural networks: full course*. Ontario: Williams.
- Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv*, 11.
- Itsjoki, O. (2006). Elección de modelo y paradojas de predicción. *Kvantil*, 43-51.
- Kulkarni, A. (2020). Foundations of data imbalance and solutions for a data democracy. *Data Democracy*, 23.
- Lee, H., Grosse, R., Ranganath, R., & Ng, A. (2009). Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. *ICML*, 8.
- Liu, H. (2018). Leveraging Financial News for Stock Trend Prediction. *arXiv*, 24.
- Maimon, O., & Rokach, L. (2005). *Data Mining and Knowledge Discovery Handbook*. Tel Aviv: Springer.
- Nakamoto, S. (31 de Octubre de 2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. Obtenido de <https://bitcoin.org/bitcoin.pdf>



- Oreshkin, B., Carpov, D., Chapados, N., & Bengio, Y. (2019). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *arXiv*, 1905.
- Sen, J., & Tamal, C. D. (2017). A Robust Predictive Model for Stock Price Forecasting. *Conference Paper*, 13.
- Shan, C., Chen, W., Wang, H., & Song, M. (2015). *The Data Science Handbook: Advice and Insights from 25 Amazing Data Scientists*. San Francisco: Data Science Bookshelf.
- Srivastava, N., Krizhevsky, A., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Net-works from Overfitting. *Journal of Machine Learning Research*, 1929-1958.
- Wasserman, P. (1992). *Neurocomputer technology: Theory and practice*. Boston: Coriolis Group.
- Weatherall, J. (2014). *The Physics of Wall Street: A Brief History of Predicting the Unpredictable*. Nueva York: Harperbusiness.
- Wlodarczak, P. (2017). Exploring the value of Big Data analysis of Twitter. *Semantic Scholar*, 166.
- Yan, H. (2017). Financial Time Series Prediction Based on Deep Learning. *Wireless Personal Communications*, 18.