TRABAJO DE FIN DE GRADO

Doble Grado en Economía y Estadística

Título: El trading de alta frecuencia (HFT) y sus consecuencias en los mercados. Creación de un sistema automático mediante redes LSTM.

Autor: Javier Cano Monclús

Director: Salvador Torra Porras

Departamento: Econometría, Estadística y Economía Española

Convocatoria: Septiembre 2021

UNIVERSITAT DE BARCELONA

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Resumen y palabras clave

Este trabajo trata de unificar conceptos del trading algorítmico de alta frecuencia tanto desde un punto de vista teórico como desde el punto de vista de la programación algorítmica. Por este mismo motivo, y debido a que el TFG es de doble grado, el trabajo se encuentra dividido en dos partes.

En primer lugar, se realiza una presentación del Trading de Alta Frecuencia (HFT por sus siglas en inglés), cuáles son sus características y que estrategias se utilizan en este tipo de trading. Seguidamente, a través de la literatura actualmente existente sobre este ámbito, se muestra como afecta el HFT en los mercados de valores tanto en la parte de liquidez, como de volatilidad y como de eficiencia para poder entender si este tipo de trading es algo negativo para otros tipos de agentes o si, a diferencia de lo que la gente cree, puede beneficiar a los usuarios del mercado. Finalmente se hace una breve mención a los efectos que puede tener este tipo de trading en los Flash Crashes (o si es el culpable de los mismos) y se explica cómo esta regulado este tipo de trading en la actualidad en los diferentes mercados, haciendo énfasis en el mercado europeo.

La segunda parte del trabajo consiste en la creación de un algoritmo de trading automático con la particularidad de que imita algunos de los aspectos del Trading de Alta Frecuencia, pero realizados en servidores con una mayor latencia y una menor potencia computacional que los utilizados en el HFT. Para ello, se han mezclado herramientas del Análisis Técnico junto a dos modelos diferentes de Redes Neuronales. La primera, una red de tipo LSTM que intentará realizar las predicciones. La segunda, una red neuronal FFNN que juntará las predicciones de la red anteriormente mencionada con las características extraídas del Análisis Técnico para poder, de este modo, obtener un modelo que nos indique en que valores se debe invertir minuto a minuto y en cuales no para obtener unos beneficios mayores que la media del conjunto de los valores del NASDAQ-100.

Palabras clave: Trading de alta frecuéncia, Análisis técnico, LSTM, FFNN.

Abstract and key words

This work is tries on unify concepts of the high frequency algorithmic trading from a theoretical point of view as from an algorithmic programming point of view. Because of that, and due to this is a double degree TFG, this work is divided into two differentiated sections.

First of all, it has been done a presentation of the High Frequency Trading (HFT), which are its particularities, and which are the strategies that are usually used in that type of trading. Next, going through the currently existing academic literature on this area, it is showed how the HFT affects the stock markets in the liquidity, the volatility and efficiency in order to be able to understand if this type of trading is negative for other types of traders or if, contrary to what people usually believes, it can benefit the stock market users. Finally, a brief mention is made of the effects that this type of trading can have on Flash Crashes (or if the HFT is the guilty of causing them) and it is explained how the HFT is now a days regulated in the different stock markets, emphasizing in the European Stock Market.

The second part of the work consists of creating an automatic trading algorithm with the particularity that it mimics some of the aspects of the High Frequency Trading but performed on higher latency and no co-located servers and with less computational power than those used by the HFT. For that purpose, Technical Analysis (TA) tools has been blended with two different model of Neural Networks. The first, a Long-Short Term Memory Neural Network (LSTM) that will make the predictions. The second, a Feed Forward Neural Network (FFNN) that will combine the predictions of the previously aforementioned network with the key parameters extracted from the TA in order to be able, in this way, to obtain a model that indicates in which values is better to invest minute by minute and in which not to do it to obtain higher benefits than the average of all NASDAQ-100 stocks.

Key words: High Frequency Trading, Technical Analysis, LSTM, FFNN.

Clasificación AMS

- **62M20:** Inference from stochastic processes and prediction.
- **91B84:** Economic time series analysis.
- **68T07:** Artificial neural networks and deep learning.
- **91-10:** Mathematical modeling or simulation for problems pertaining to game theory, economics, and finance.

Índice general

Introducción

Cuando el público general piensa en el mercado de valores suele definir a este como un lugar físico lleno de personas dando órdenes a viva voz mientras hacen extraños gestos con sus manos para vender y comprar las acciones de sus clientes, los cuáles han utilizado un teléfono para decirle a sus brókeres que operaciones quieren realizar. Pero, realmente, el mercado de valores actual dista mucho de esta idea.

Hoy en día, el mercado de valores es un lugar lleno de grandes servidores que reciben órdenes de compra y venta en cuestión de segundos mediante el uso de internet. Estas órdenes son realizadas en su mayor parte por brókeres online (páginas web desde las cuales una persona puede decidir que operación realizar en cada instante a cambio de una comisión), los cuales permiten que el tiempo entre la realización de la orden y su propia ejecución sea de tan solo segundos.

Pero no todas las operaciones que se realizan hoy en día son realizadas en estas páginas web. Desde principios de el siglo XXI se ha gestado un nuevo fenómeno llamado Trading de Alta Frecuencia (HFT, por sus siglas en inglés). Un tipo de negociación de instrumentos financieros que utiliza algoritmos matemáticos para la ejecución de órdenes a una velocidad miles de veces mayor que la que los humanos pueden realizar.

Es por este motivo que la primera parte de este trabajo, que coincide con la parte de economía del grado, consiste en analizar las principales características del Trading de Alta Frecuencia, así como cuales sus efectos en el mercado de valores a partir de una recopilación de la literatura actualmente existente sobre este ámbito.

La segunda parte, que coincide con la parte de estadística del grado, está basada en la realización de un algoritmo de trading a partir de redes neuronales que intente simular el funcionamiento del HFT. A continuación, se utilizaran los datos del NASDAQ-100 para simular su operatividad y observar si el algoritmo creado permite generar unos mejores resultados que el propio índice bursátil minuto a minuto.

Metodología

Este trabajo se encuentra dividido en 2 partes muy diferenciadas. La primera hace referencia a los aspectos más teóricos del HFT, mientras que la segunda intenta crear un algoritmo de trading automático utilizando algunas de las características del trading de alta frecuencia.

La primera parte, tal y como se ha mencionado, parte de una base más teórica donde se define lo que es el Trading de Alta Frecuencia y su funcionamiento. Además, se hace énfasis en analizar cuales son sus diferentes estrategias y como afectan estas a los diferentes mercados. Para poder realizar este análisis se utilizan y recopilan un conjunto de artículos académicos donde simulan o tienen información de primera mano de los mercados. Este hecho se realiza de esta manera debido a que el trading de alta frecuencia no puede ser identificado a no ser que tengas información interna de los propios mercados que se encuentra restringida para el público en general. Por este motivo, la mayor parte del trabajo realizado se basa en recopilar y explicar cuales son los resultados a los cuales diferentes expertos han llegado y, de esta manera, poder explicar si el HFT es perjudicial o beneficioso para los mercados.

La segunda parte, en cambio, se trata de la creación de un algoritmo de trading. Para poder realizar este algoritmo se utilizan varios tipos de software que a continuación se enumeran y seguidamente se presentarán. La principal herramienta utilizada es el lenguaje de programación Python[1], el cual se ejecuta en la IDE Spyder[2].

Esta segunda parte del trabajo utilizará diversos indicadores de análisis técnicos ampliamente conocidos tales como el MACD, pero adaptados al trading intradía con una frecuencia de minutos realizando modificaciones en sus expresiones para poderles sacar el máximo partido. Además, también se utilizara una red LSTM que, unida a los indicadores técnicos anteriores mediante una FeedForward Neural Network se espera obtener que valores son indicados para tener en la cartera en cada momento del tiempo.

[1] https://www.python.org/
[2] https://www.anaconda.com/

Debido a que Python es un lenguaje de programación no visto en el grado, a continuación se va a proceder a explicar en que consiste este, como utilizarlo e instalarlo, cuales son los entornos en los cuales se ha trabajado y cuál es el motivo por el cual ha sido elegido.

Python es un lenguaje de programación interpretado que destaca por su simplicidad y limpieza en el código. Es de código abierto y, por lo tanto, todo el mundo puede aportar sus conocimientos para mejorar el lenguaje con la creación de bibliotecas que facilitan las tareas a los usuarios del software.

Python fue creado a finales de la década de 1980, pero es en esta última década donde ha tomado una mayor relevancia gracias al impulso que ha recibido por parte del campo del Data Science. Y es que la mayor parte de científicos de datos utilizan Python y, debido a su característica de código abierto, permite que una gran cantidad de personas colaboren a mejorar el lenguaje para facilitar su campo de desarrollo. De hecho, grandes empresas como Google o Facebook han creado proyectos de código abierto como TensorFlow, Keras y Pytorch para facilitar el uso de herramientas de Data Science entre los programadores.

Pero Python, solo es un lenguaje de programación y, por lo tanto, ejecuta el código que previamente le facilitamos. Para crear este código se recomienda usar una IDE (entorno de desarrollo integrado), un software que se utiliza como editor de código, compilador, depurador e interfaz gráfica. En este caso se ha usado la IDE Spyder, debido a que, al igual que Python, es de código abierto y está ideada para la programación científica.

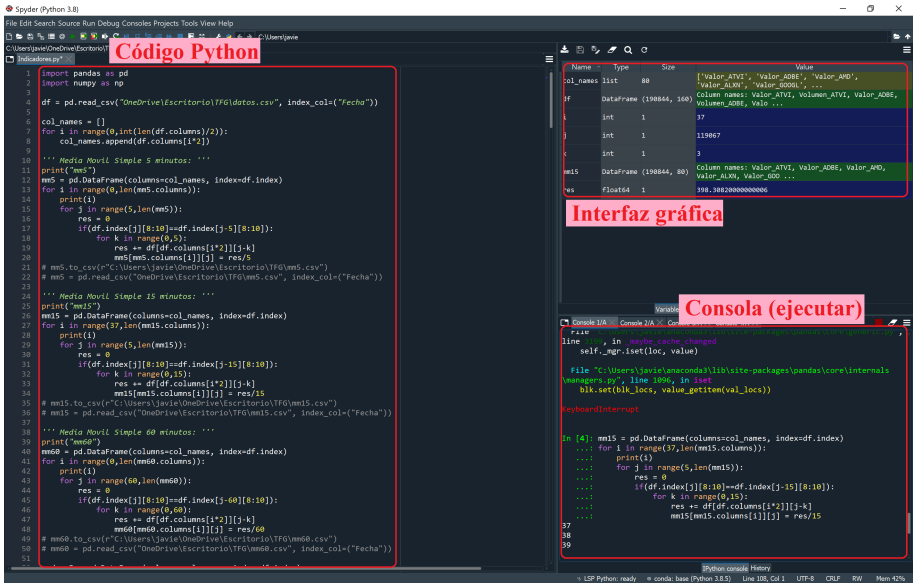

Figura 1: Visualización del IDE Spyder
Fuente: Elaboración propia

Para instalar y poder programar en Python en la IDE Spyder es recomendable instalar

Anaconda, una distribución de código abierto que contiene una serie de aplicaciones y librerías previamente instaladas y clasificadas para poder desarrollar código en Python en el campo de la ciencia de datos.

En esta distribución podemos encontrar una terminal para ejecutar código en Python y para instalar diferentes bibliotecas que sean de utilidad y que no se encuentren previamente instaladas, aplicaciones para la ciencia de datos como RStudio o Jupyter Notebook e incluso el propio IDE Spyder, en la cual se ha desarrollado este proyecto.

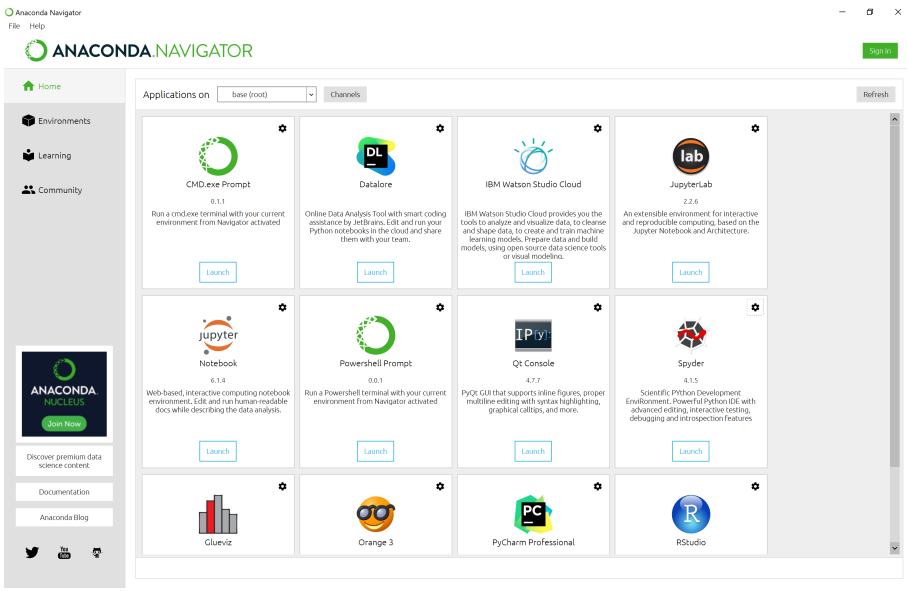

Figura 2: Visualización de la distrubución Anaconda
Fuente: Elaboración propia

Capítulo 1

Análisis del HFT

En esta primera parte, el objetivo es definir las características que conforman el Trading de Alta Frecuencia, entender los diferentes tipos de metodologías que se utilizan y finalmente observar de que manera se ve afectado el mercado por estas metodologías. Pero para ello, es imprescindible analizar si el HFT tiene suficientemente peso en el conjunto del mercado como para poder modificarlo. Con ese fin se obtiene la Figura 1.1.

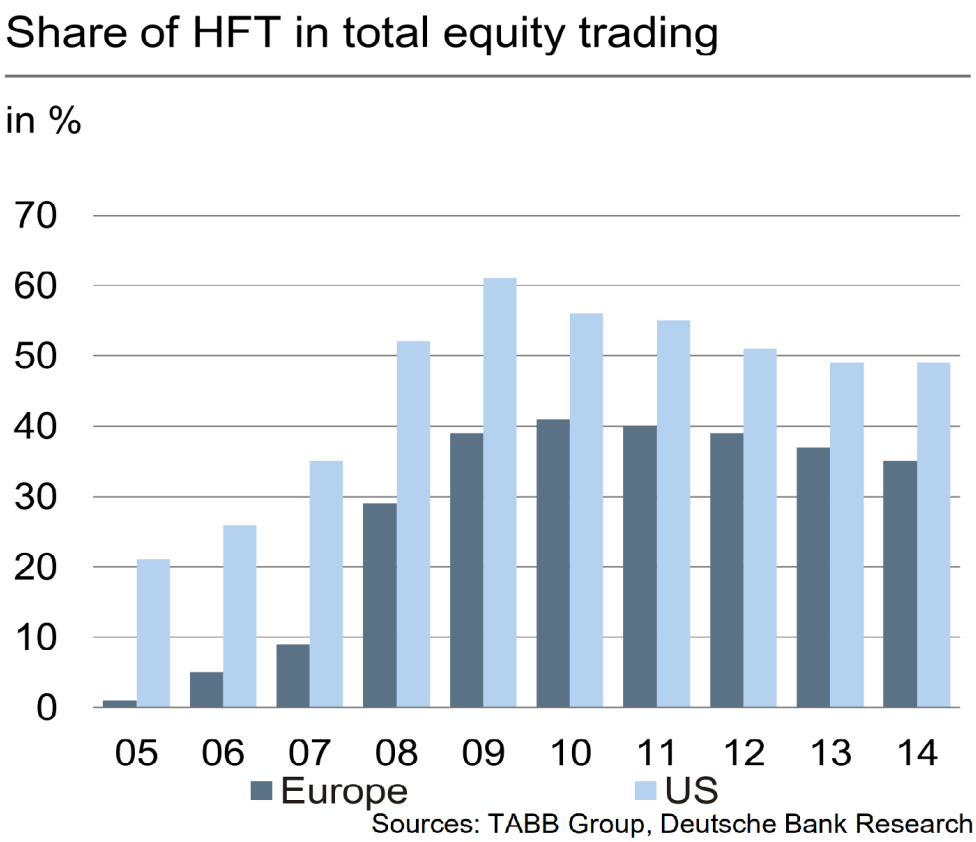

Figura 1.1: Cuota del HFT en los mercados Europeos y Estadounidenses
Fuente: DeutcheBank; Research Briefing: High-frequency trading

Esta figura muestra la cuota que tiene el Trading de Alta Frecuencia en los Mercados de Europa y Estados Unidos entre los años 2005 y 2014. Se puede observar que el HFT se encuentra más presente en el mercado americano, donde en el año 2009 éste llegó a ocupar el 60 % de las operaciones que se realizaban en sus diferentes mercados de acciones después de varios años al alza, para después descender ligeramente. Algo parecido ocurrió en el mercado europeo, donde el año 2010 fue el año donde mayor cuota tuvo el HFT, con un 40 % del total de las operaciones que se realizaron en sus diferentes mercados.

Como se puede apreciar, desde principios de la década de 2010 se aprecia una reducción de la participación del HFT en el mercado. Este efecto es debido a varias razones [1]. En primer lugar, un descenso en los beneficios debido al coste creciente de la infraestructura y la competitividad a causa de la entrada de nuevos competidores en el uso de estas nuevas metodologías (Ver Figura 1.2). En segundo lugar, el crecimiento de plataformas de trading alternativas, las cuales han permitido que una mayor población realice operaciones en el mercado de valores. En último lugar, una regulación que era inexistente y que día a día se está volviendo más restrictiva para proteger a otros tipos de inversores.

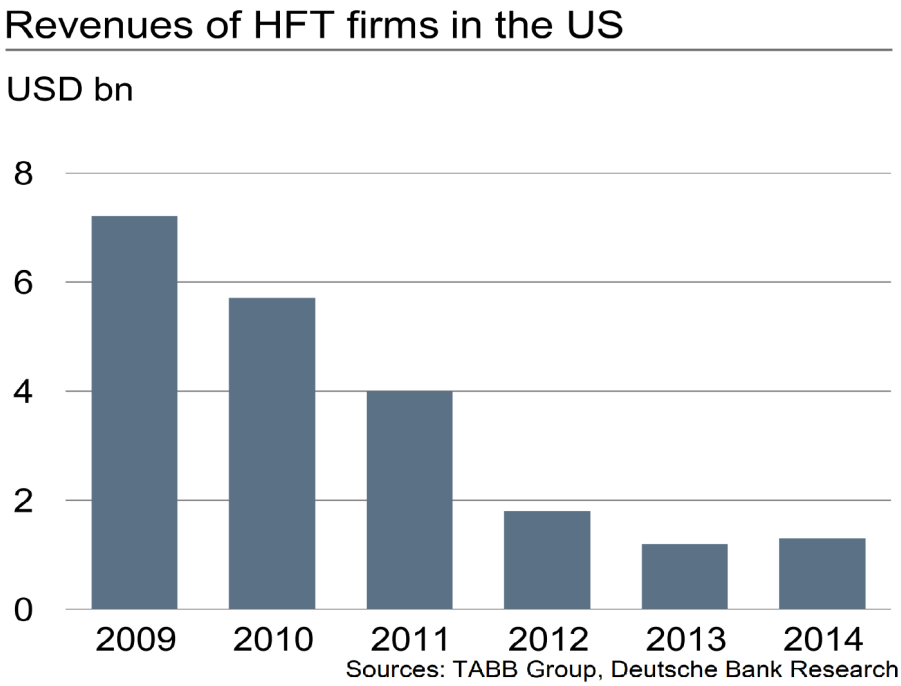

Figura 1.2: Beneficios de las firmas de HFT en los Estados Unidos
Fuente: DeutcheBank; Research Briefing: High-frequency trading

Tal y como se ha podido apreciar, el HFT tiene suficientemente peso en el mercado de valores como para ser un elemento a tener en cuenta a la hora de saber cómo se va a comportar el propio mercado y, aunque en los últimos años se ha estancado en torno al 40 % del total del peso en el mercado, parece que esta nueva herramienta de Trading ha venido para quedarse.

1.1. Características del HFT

El Trading de Alta frecuencia (HFT) es una subcategoría del Trading Algorítmico (AT) y, por lo tanto, es un tipo de trading que utiliza algoritmos informáticos para determinar automáticamente los parámetros que permiten decidir cuáles son las órdenes que se deben ejecutar y en qué momento se deben ejecutar estas órdenes, siempre sin la ayuda de la intervención humana. Las características que diferencian al Trading de Alta Frecuencia del Trading Algorítmico son las siguientes:[2]

1.1.1. Velocidad

Ésta es la característica más importante del trading de alta frecuencia y la que más ha evolucionado en el tiempo ya que, tal y como indica Andy Haldane, el director ejecutivo de análisis y estadística del Banco de Inglaterra, en el principio del siglo XX el tiempo medio de ejecución de las órdenes en el Trading de Alta Frecuencia era de la escala de los segundos, pero actualmente ya nos encontramos en la escala de los microsegundos y, por lo tanto, el tiempo de ejecución ha descendido una millonésima parte en poco más de una década.[3] Para llevar a cabo esta reducción de tiempo, han sido necesarios grandes avances en la tecnología, tanto en los ordenadores que realizan los cálculos y las operaciones como en la velocidad de transmisión de la propia orden. Hecho que ha sido posible con la siguiente característica.

1.1.2. Co-location

La co-location, es el hecho de alojar los servidores y equipos informáticos lo más cerca posible de los centros de procesamiento de los mercados para poder recibir la información y ejecutar las órdenes lo más rápido posible, ahorrándose de esta manera el tiempo de transporte de las diferentes señales.

En la imagen siguiente (Figura 1.3) se puede observar el caso del Mercado de Valores de Johannesburgo, el cuál se encuentra en Sandton. Desde allí el retardo que se tienen al realizar las operativas es de 2.7 ms, pero si operáramos desde Europa, el tiempo sería de 167 ms, unas 60 veces más. Un tiempo crucial que en el mundo del Trading de Alta Frecuencia puede suponer que tu orden llegue unas décimas de segundo más tarde de lo esperado y, por lo tanto, puedes llegar a perder dinero porque la operativa que estás intentando llevar a cabo ya no es efectiva.

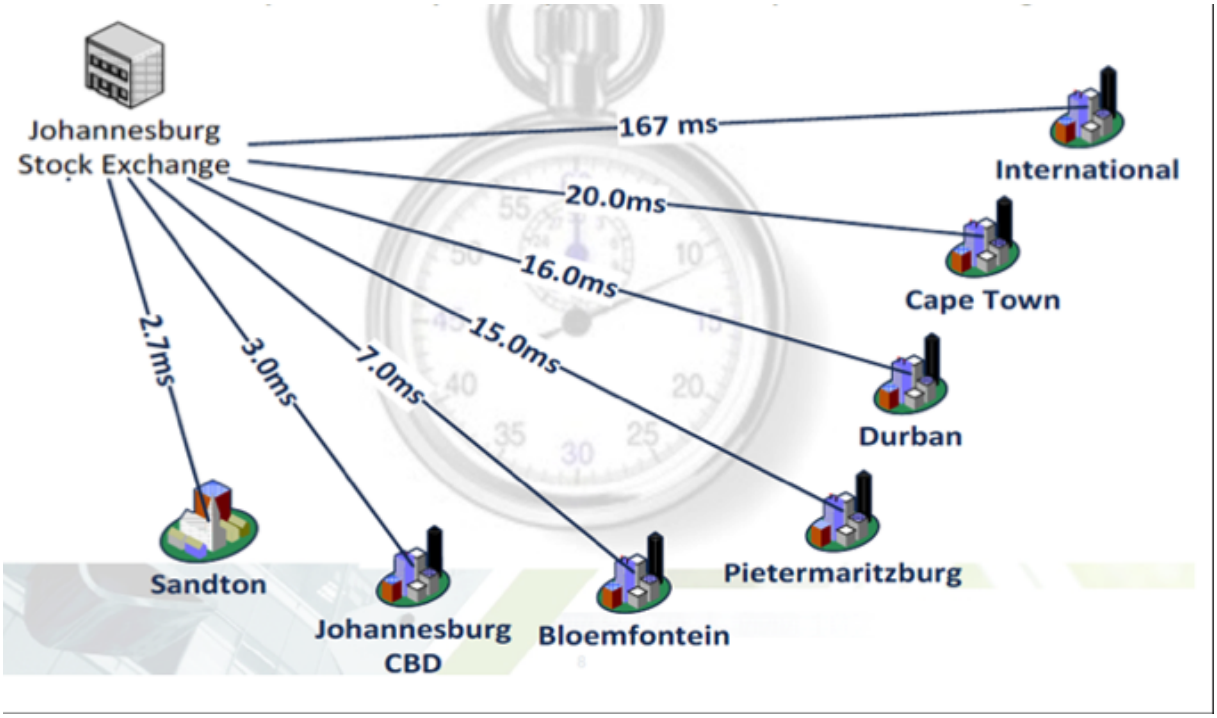

Figura 1.3: Como afecta la Co-Location al tiempo medio esperado de ejecución
Fuente: SA Financial Markets Journal; Colocation: reducing latency in financial market transactions and creating an 'HFT and Algo trading friendly' market environment

1.1.3. Baja latencia

Una baja latencia implica un tiempo muy breve para establecer y liquidar posiciones. Esto permite que, si una orden enviada necesita ser modificada porque ya no es eficiente, ésta puede ser eliminada con gran rapidez. También indica que en general el tiempo que pasa entre que se crea la orden y hasta que se elimina o entre dos órdenes es prácticamente nulo.

La baja latencia, por lo tanto, implica que las acciones no están grandes periodos de tiempo en manos de los Traders de Alta Frecuencia y, por lo tanto, la estrategia para obtener beneficios no es tanto el hecho de elegir las empresas correctas para recibir una rentabilidad a largo tiempo, sino realizar un gran número de operaciones que nos ofrezcan una pequeña rentabilidad individualmente, pero que en el conjunto ofrezcan unos beneficios elevados.

1.1.4. Trading intradía

El trading intradía es una estrategia de inversión que implica que las todas las posiciones que se abren en un día son cerradas antes del cierre del mercado. Este hecho permite que no les afecten las oscilaciones que pueda sufrir una acción durante el periodo donde el mercado está cerrado.

El HFT, debido a que realiza sus operaciones pensando en vender en periodos muy breves de tiempo, no presta atención a las tendencias que pueda tener a medio o largo plazo una acción. Debido a este motivo, los algoritmos de HFT evitan mantener posiciones durante la noche, ya que una acción puede ser beneficiosa a muy corto plazo pero de un día para otro se podría perder dinero si la tendencia de la acción va en la dirección opuesta a la esperada por el algoritmo.

1.2. Estrategias en el HFT

Tal y como se ha podido ver en el apartado anterior, el Trading de Alta Frecuencia esta definido por una serie de características, las cuales se pueden resumir como: muy alta velocidad de ejecución de las órdenes gracias a la ubicación y potencia de los servidores, hecho que permite generar beneficios bajos repetidamente en un solo día que, de forma acumulada al final de la sesión, permite obtener en suma unos grandes beneficios al final de la sesión.

Aún y así, todavía no se han definido los mecanismos a partir de los cuales el HFT obtiene los beneficios previamente mencionados.

Este hecho es debido a que el HFT engloba un conjunto de estrategias muy diferentes para poder obtener los beneficios ya que, al igual que en el trading más tradicional existen diversas estrategias para operar que resultan todas ellas válidas, en el Trading de Alta Frecuencia también existen diferentes estrategias que permiten obtener beneficios.

A continuación, se presentan las estrategias más usadas y se da una explicación del funcionamiento de cada una de ellas.[2]

1.2.1. Creación de mercado

La creación de mercado, o market-making en inglés, es una estrategia usada desde hace años por diferentes tipos de agentes y que hoy en día también realizan los HFT. Por lo tanto, no es una estrategia nueva.

Consiste en que un agente, en este caso el Trader de Alta Frecuencia, realiza órdenes de compra y venta de forma constante en un rango de precios determinado. Estas órdenes se encuentran cerca del valor esperado de la acción en ese momento, pero nunca llegan a este. Este hecho produce que las órdenes de compra siempre se encuentren un poco por debajo del precio que piden los vendedores y que las órdenes de venta siempre se encuentren un poco por encima del precio que piden los compradores. De esta manera, las órdenes que crea el HFT se encuentran siempre entre las primeras posiciones en el libro de órdenes y, en el caso de que algún inversor decida lanzar una orden de mercado, siempre se ejecutará la orden del HFT.[4]

La creación de mercado es compatible con el Trading de Alta Frecuencia debido a que el HFT puede calcular en todo momento un valor esperado de la acción y, gracias a su baja latencia, puede dar un par de órdenes de compra y venta que sean lo suficientemente

ajustados al valor esperado como para encontrarse en todo momento entre los primeros valores en el libro de órdenes y, a la vez, conseguir tener suficiente margen entre el valor ofrecido y el valor esperado para que, cuando la orden se ejecuta, poder obtener beneficios.

En la figura 1.4 se puede observar un ejemplo del conjunto de órdenes de compra y venta existentes en cada segundo de una acción de Intel. En primer lugar, se pueden ver una línea roja y una verda (bid y ask) que indican a que precios se está vendiendo y comprando la acción en todo momento. En segundo lugar, se puede observar un mapa de calor alrededor de estas líneas que indican la disposición del libro de órdenes en cualquier instante del tiempo. Cuanto mas cálido es el color, más órdenes hay en ese precio. Y cuanto más frio, menos órdenes existen.

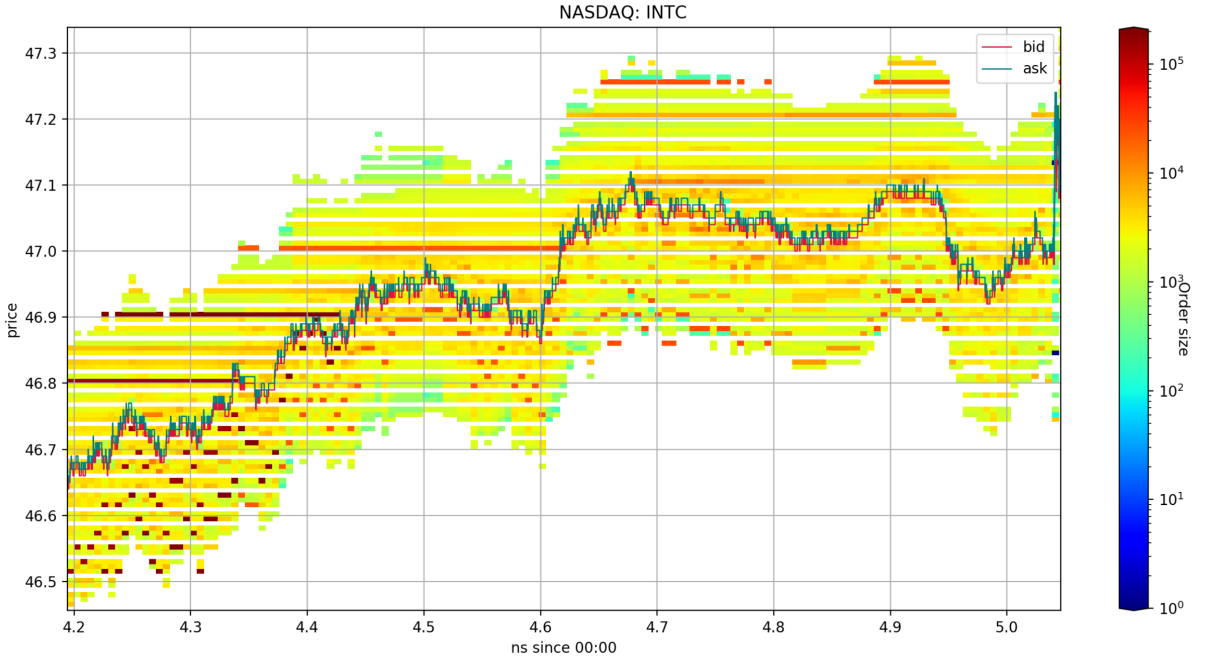

Figura 1.4: Libro de órdenes a través del tiempo
Fuente: https://github.com/martinobdl/ITCH/blob/master/README.md

En la figura 1.5, en cambio, se puede ver el libro de órdenes de compra y venta en un momento estático del tiempo. En este ejemplo se puede observar como el precio esperado se encuentra entre los valores 46.47 y 46.485. Esto es debido a que las órdenes de compra (rojo) se encuentran por debajo de 46.47 y las órdenes de venta (verde) por encima de 46.485. Como podemos apreciar, el número de acciones existentes en el libro de órdenes oscila entre las 0 y las 400 en todos los valores excepto en el 46.45, mientras que en este valor se superan las 1400 acciones para comprar. Esto puede ser debido a que en este valor se han juntado por azar varios inversores que han elegido el mismo precio, que hay un gran inversor intentando comprar acciones, o que un HFT ha elegido ese precio como valor de compra óptimo. Desgraciadamente, únicamente con este gráfico y debido a que

no es posible realizar un seguimiento del Trader que ha realizado esta orden, no hay forma de averiguarlo.

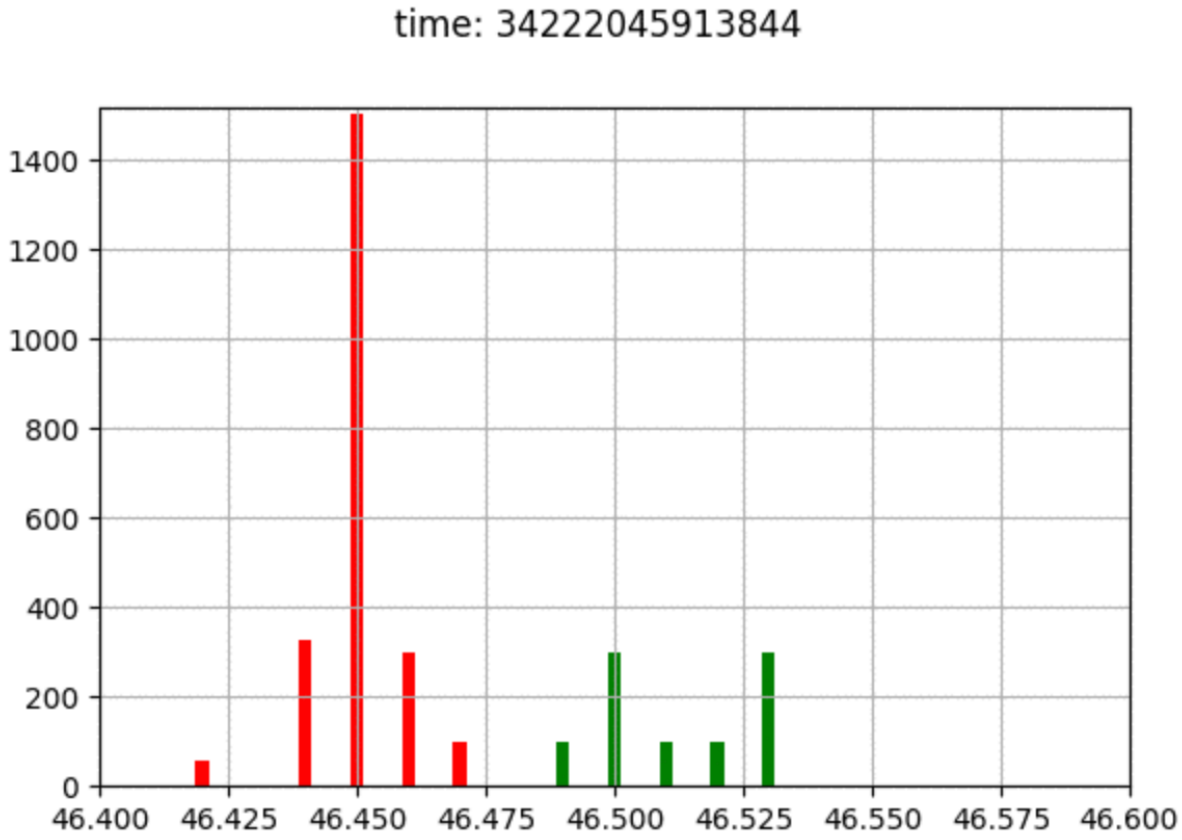

Figura 1.5: Libro de órdenes en un momento concreto
Fuente: https://github.com/martinobdl/ITCH/blob/master/README.md

La creación de mercado permite proporcionar liquidez a los mercados, aunque debido a que el HFT puede dejar de realizar esta operación en cualquier momento, en momentos de estrés puede propiciar la aparición de Flash-Crashes (caídas repentinas, muy fuertes y súbitas en la cotización de un activo concreto). Es por este motivo que actualmente existen regulaciones en los agentes que utilizan el market-making, tal y como veremos más adelante.

1.2.2. Arbitraje

Las estrategias de arbitraje son un conjunto de técnicas que permiten sacar rendimientos de posibles ineficiencias existentes en los diferentes mercados. El arbitraje debe su existencia a la información imperfecta de los mercados y al poder que tiene el HFT para poder obtener la información antes que otros tipos de traders. De esta manera, el Trader de Alta Frecuencia que utiliza estos métodos tiene un papel de estabilización y equilibrio de los diferentes mercados que se encuentran interrelacionados. [5]

Existen diversos tipos de estrategias de arbitraje, pero el HFT utiliza 3 tipos de ellos: Arbitraje estadístico, Arbitraje de índice y arbitraje de eventos. En los puntos siguientes se explica el funcionamiento de cada uno de ellos, pero para entender intuitivamente en qué consiste el arbitraje, a continuación se puede apreciar un ejemplo donde se explica.

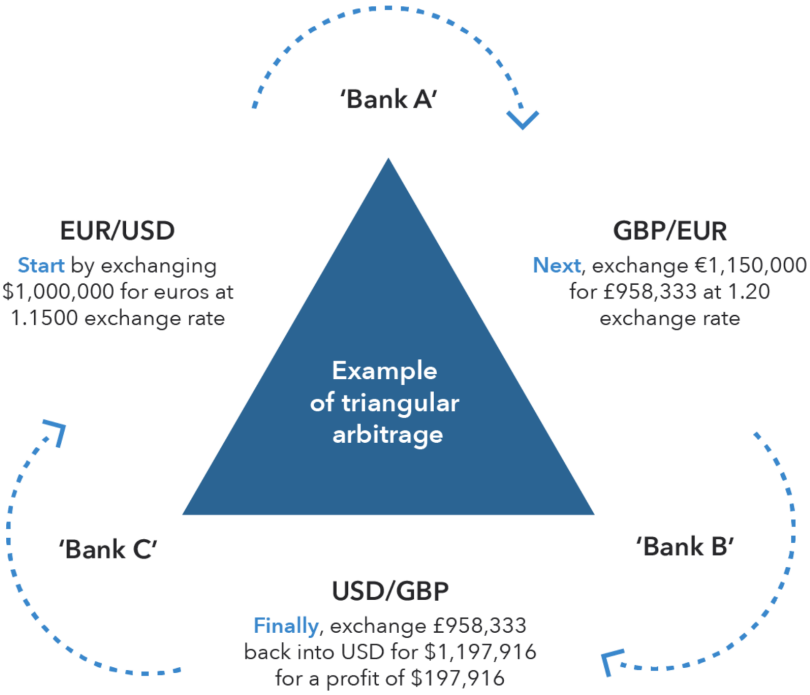

Figura 1.6: Ejemplo de funcionamiento de las estratégias de arbitraje
Fuente: IG Bank; Arbitrage trading in forex explained

En la imagen anterior (Figura 1.5), se puede observar como se obtienen beneficios a partir de una técnica de arbitraje en 3 mercados monetarios diferentes (Euro-Dólar, Libra-Euro y Dólar-Libra). El funcionamiento es el siguiente: Un agente observa que el tipo de cambio Dólar-Euro está a 1.15 euros por cada dólar y decide cambiar 1 millón de dólares por 1.15 millones de euros. Seguidamente el agente se percata que el tipo de cambio Euro-Libra está a 0.8333 libras por cada euro y decide cambiar todos sus euros por libras, obteniendo así 958333 libras. Finalmente, el agente, que previamente había observado que los mercados no estaban bien ajustados y que el tipo de cambio Libra-Dólar es de 1.25 dólares por cada libra, cambia las libras por dólares obteniendo un total de 1197916 dólares, lo que le genera un beneficio de 197916 dólares debido a que, por fluctuaciones en la cotización de las diferentes divisas, los mercados no se encontraban ajustados. Como contrapartida al obtener beneficios, el agente ha favorecido al equilibrio de los 3 mercados debido a sus intercambios de divisas y, de esta manera, el mercado vuelve al equilibrio y desaparecen estos desajustes. Este mecanismo se llama descubrimiento de precios.

A continuación, se analizan los tres diferentes tipos de arbitrajes utilizados en el HFT.

1.2.2.1. Arbitraje estadístico

El arbitraje estadístico es un tipo de arbitraje donde se utilizan los conceptos de cointegración, estacionariedad y homocedasticidad. El funcionamiento se basa en encontrar un conjunto de valores que se encuentren muy correlacionados negativamente entre ellos con el objetivo de disminuir la variabilidad. Una vez se tiene este conjunto de valores correlacionados se observa cuál es su tendencia en periodos de tiempo muy cortos (segundos) e invertir en estos conjuntos de valores a corto (si la tendencia es negativa) o a largo (si la tendencia es positiva).

Este tipo de arbitraje es una herramienta que permite recibir beneficios de manera constante debido a que su riesgo es muy bajo (esto es causado por el hecho de que notícias o variables exógenas que puedan afectar a un valor de manera negativa, harán que los otros valores se vean beneficiados de manera positiva por el mismo suceso debido a que la correlación entre ellos es negativa). Pero como contrapartida, es muy frágil ante los cambios bruscos de mercado ya que, si por algún motivo todos los valores se ven afectados de la misma manera, el trader puede tener una gran cantidad de pérdidas en un breve periodo de tiempo que no se puedan recuperar a medio-largo plazo. Es por este motivo que, aun siendo un método con bajo riesgo en promedio, es imprescindible programar bien los algoritmos para poder parar posibles eventos inesperados antes de que estos sean perjudiciales.[6]

1.2.2.2. Arbitraje de eventos

Este tipo de arbitraje se dice así porque trata de obtener beneficios a partir de desajustes en los precios causados por periodos de inestabilidad en la acción debido a variables macroeconómicas, reajustes en la empresa o en el propio valor, o por noticias que puedan afectar a la cotización. [7]

Cabe destacar que algunos especialistas defienden que el trading basado en noticias puede ser una estrategia totalmente diferente al arbitraje de eventos, pero debido a que las repercusiones en el mercado son idénticas al arbitraje de eventos, se ha decidido incluir el trading basado en noticias dentro del arbitraje de eventos. [8]

1.2.2.3. Arbitraje de índice

El arbitraje de índice es un tipo de arbitraje que trata de aprovecharse de las diferencias de precios existentes entre diversos índices. Este tipo de arbitraje es el más parecido al

que hemos podido ver en el ejemplo de la figura 1.5, ya que trate de encontrar los cambios existentes entre varios conjuntos de valores creados por los cambios en las cotizaciones de estos y ejecuta ordenes de manera casi instantánea para revertir estos procesos ya que, de por sí, estas diferencias no deberían existir. [9]

Un caso donde este método es muy utilizado es en la Negociación Básica (Basis Trading en inglés), donde se utiliza la brecha que existe entre la cotización de un índice y de su propio futuro. Si se observa que el precio de los futuros es mayor que el del índice porque el índice es más barato de lo que debería, se compran acciones del propio índice y se venderán acciones de los futuros, ya que este debería subir para equilibrarse con el precio de los futuros. En cambio, si los futuros están más baratos, se compraran futuros y se venderán acciones del propio índice. [10]

1.2.3. Estructural

Las estrategias estructurales están orientadas a explotar las vulnerabilidades del propio mercado o de otros agentes del mercado. Las estrategias estructurales son las más dañinas que existen, ya que intentan aprovecharse de la falta de recursos de los otros participantes para obtener sus propios beneficios. A continuación, se pueden observar un conjunto de estrategias que están englobadas en el trading estructural.

1.2.3.1. Stuffing

El stuffing es una estrategia que está basada en introducir en el mercado un gran número de órdenes con el fin de que los competidores (principalmente otras firmas de HFT), tengan que procesar estas órdenes, y de esta manera perder tiempo, mientras que las firmas que las generan ignoran estas órdenes. Esta estrategia intenta sacar ventaja de tiempo a los competidores en el mercado, pero puede causar retrasos en el mercado debido a la gran cantidad de órdenes generada o, incluso, Flash Crashes. [11]

1.2.3.2. Smoking

Esta estrategia está basada, al igual que el stuffing, en la creación de órdenes en el mercado. La diferencia es que el smoking crea órdenes límite con el objetivo de que parezca que el precio sube y, de esta manera, atraer a otros agentes con una latencia más alta al mercado. Una vez se han atraído estos agentes, se modifican las órdenes creadas para ejecutarlas contra los agentes que habían sido atraídos. [12]

1.2.3.3. Spoofing

Por último, el spoofing consiste, al igual que s dos estrategias anteriores, en la introducción en el mercado de una gran cantidad de órdenes en un periodo muy breve de tiempo para, a continuación, cancelarlas. El objetivo principal es manipular los precios en un periodo breve de tiempo con el fin de colocar órdenes en la dirección contraria para poder realizar sus operaciones a un precio más beneficioso para ellos. [13]

1.2.4. Direccional

Las estrategias direccionales, por su banda, son un conjunto de técnicas que implican establecer una posición en corto o en largo (según la tendencia esperada por el HFT) anticipándose a posibles subidas o bajadas en el valor.

Existen 2 tipos de estrategias direccionales llamadas orden de anticipación (más conocida como order anticipation) y encendido de impulso (más conocida como momentum ignition).

1.2.4.1. Order anticipation

Larry Harris, en el capítulo 11 de su libro Trading and Exchanges: Market Microstructure for Practitioners [14], explica que las estrategias de order anticipation son estrategias que obtienen benefios comprando acciones antes de que otros agentes lo hagan. Pero, a diferencia de las estrategias estructurales, el HFT no manipula el mercado, sino que intenta utilizar herramientas tales como el front-running, análisis de sentimientos o análisis técnico para predecir cuál va a ser la tendencia que van a seguir los diferentes agentes respecto una acción en el mercado.

Es importante remarcar que estos HFT no intentan averiguar la dirección de una acción en el mercado, sino que intentan averiguar que van a realizar otros agentes para, de esta manera, comprar a precios más favorables que estos agentes y sacar provecho de las operaciones que estos hagan. Es por este motivo, que las estrategias de order anticipation son realizadas por traders llamados parásitos.

1.2.4.2. Momentum ignition

Tal y como la define la Comisión Nacional del Mercado de Valores estadounidense (SEC por sus siglas en inglés) este tipo de estrategias utilizan las técnicas de Spoofing (vistas anteriormente) para crear una tendencia en el mercado a la vez que compran acciones antes de que se cree esta tendencia para sacar beneficio de esta de la misma manera que en las estrategias de Order Anticipation. [15]

En definitiva, el momentum ignition es una herramienta muy similar al Spoofing que es perjudicial para el mercado y, por lo tanto, se encuentra muy regulada actualmente.

1.3. Dark Pooks

A parte del mercado al cual estamos acostumbrados (un mercado donde un agente puede observar en todo momento las operaciones que se realizan en el mercado), existe un mercado alternativo llamado "Dark Pool" que consiste en un mercado privado de negociación de valores de gran tamaño. Originalmente, los Dark Pools fueron mercados diseñados para que los grandes inversores pudieran hacer sus operaciones sin que el precio del valor se viera altamente modificado por la incorporación de un gran número de órdenes en el mercado. Para realizar esto, los grandes grupos de inversión pactan un precio de compraventa de un gran conjunto de acciones fuera del mercado, realizan el intercambio y, seguidamente, notifican al mercado de la transacción realizada.

Los mercados alternativos del tipo Dark Pools son buenos para los grandes inversores porque no tienen que sufrir las pérdidas que conlleva tener que depositar un ingente número de acciones en el mercado, así como también es bueno para los pequeños inversores y el mercado porque la eficiencia de este último no se ve alterada por fluctuaciones en el precio. Aún y así, las Dark Pools pueden perjudicar a algunos participantes del mercado debido a que no pueden ver las órdenes antes de que estas hayan sido ejecutadas y, por lo tanto, el mercado pierde cierto nivel de transparencia. [16]

Es en estos casos donde los HFT hacen aparición, ya que los Traders de Alta Frecuencia tienen permitido el acceso a los mercados alternativos y, por lo tanto, pueden saber en todo momento cuales son los valores que van a ser intercambiados en los mercados. Los HFT, mediante su información privilegiada, utilizan las técnicas de "order-anticipation", en concreto las técnicas de front-running, para adelantarse a los movimientos que va a tener el mercado cuando se conozca la información que ha sido ejecutada en las Dark Pools y, de esta manera, obtener beneficios actuando en la dirección en la cual se va a mover el mercado antes de que este lo haga. [17]

1.4. Efectos del HFT

Este apartado está dedicado a los efectos que tienen los diferentes tipos de estrategias de Trading de Alta Frecuencia en los mercados. Para ello, se va a proceder a explicar cada una de las características del mercado y, seguidamente, se va a mostrar de que manera se ven afectadas por las estrategias de HFT.

Las 3 características de los mercados que se van a analizar son la volatilidad (Volatility), la liquidez (Liquidity) y la eficiencia (Price discovery). Pero para ello, es importante diferenciar dos tipos de HFT ya que, tal y como mencionan Gary Shorter y Rena S. Miller en su artículo "High Frequency Trading: Overview of Recent Developments" [18] existen 2 tipos de conductas diferenciables en el HFT.

En primer lugar, existen las estrategias pasivas, que son aquellas que implican la provisión de órdenes limitadas (ofertas que se realizan para comprar o vender una cantidad determinada de acciones a un precio específico o más barato). Son las estrategias de creación de mercado y arbitraje que hemos visto en apartados anteriores.

En segundo lugar, se pueden observar las estrategias agresivas, que son aquellas que implican la provisión de operaciones inmediatamente ejecutables, como órdenes de mercado. Son las estrategias direccionales de "order anticipation" y "momentum ignition".

1.4.1. Liquidez

La liquidez es la facilidad que tiene un agente para convertir un activo en dinero, o viceversa, de manera rápida y sin modificar su precio de mercado. Una mayor liquidez implica que se podrá comprar, o vender, un producto en un periodo de tiempo más breve y por un precio que no difiera mucho del precio original. Por el contrario, una menor liquidez implicará tener que esperar un mayor periodo de tiempo para poder realizar la operación (con las posibles pérdidas que ello podría conllevar), o que se tuviera que renunciar a una parte del dinero con el fin de acelerar el proceso de transacción del activo. Tal y como se ha podido observar en los partados anteriores, el HFT, al utilizar diferentes estrategias de trading, puede ser a la vez proveedor o consumidor de liquidez.

Para poder analizar la liquidez de los mercados se va a utilizar un término llamado spread. El spread es el nombre técnico en inglés que hace referencia a la diferencia de precio de compra y venta entre un valor específico.

Tal y como indican Zhang y Riordan (2011) [19] y Carrion (2013)[20], los spreads son

más amplios en las operaciones donde los los HFT ofrecen liquidez y más estrechos donde los HFT la consumen. Este hecho implica que cuando los spreads son mayores, los Traders de Alta Frecuencia se comportan de manera pasiva (creación de mercado), mientras que cuando los spreads son menores los Traders de Alta Velocidad se comportan de manera agresiva (estrategias direccionales).

Por su banda, Bershova y Rakhlin (2013) [21] descubren que el HFT implica, en general, una disminución de los spreads debido a que la gran mayoría de Traders de Alta Frecuencia son creadores de mercado, hecho que implica que la liquidez aumente. Este suceso lo demuestra directamente Bohemer (2012) [22], donde explica que el HFT se encuentra positivamente correlacionado con la liquidez y, por lo tanto, un augmento del HFT implica un augmento de la liquidez.

Es por este motivo que podemos resumir que las estrategias de creación de mercado son grandes creadoras de liquidez, aunque en momentos de estrés pueden dejar de serlo y ello puede crear el efecto contrario en los momentos menos deseados. El arbitraje, por su banda, también aporta liquidez a los mercados.

Por otra banda, vemos como el trading direccional únicamente crea órdenes en un lado del conjunto de las operaciones y, por lo tanto, no genera liquidez, sino que la consume.

1.4.2. Volatilidad

La volatilidad es una medida de la dispersión que existe en un activo del mercado. De hecho, se suele calcular como la variación típica de un activo. Una mayor volatilidad implica un mayor riesgo, ya que los márgenes de el activo son mayores. Esto implica que una acción más volátil puede conllevar mayores pérdidas, pero también unas ganancias más elevadas.

La liquidez puede afectar a la volatilidad, pero la volatilidad no afecta a la liquidez. Esto es debido a que una menor liquidez implica que los movimientos entre precios sean mayores y, por lo tanto, aumenta la volatilidad. Es por este motivo que siempre es preferible una alta liquidez, ya que proporcionará al mercado una volatilidad menor. Este hecho lo demuestran Zhang y Riordan [21] (ya mencionados en el apartado de la liquidez), donde a parte de observar que el HFT implica una disminución de los spreads y, por lo tanto un aumento de la liquidez, también observan que el Trading de Alta Frecuencia causa un aumento de la volatilidad.

En los trabajos de Valeria Caivano (2015) [23] y Frank Zhang (2010) [24] también se

llega a las mismas conclusiones que habían llegado anteriormente Sarah Zhang y Ryan Riordan, las cuales indican que un aumento del uso del HFT en los diferentes mercados implica tener que soportar una mayor volatilidad en los mercados.

1.4.3. Eficiencia

La eficiencia hace referencia a la capacidad que tiene un mercado de trasladar en todo momento la información relevante de un activo al propio activo. Esto significa que el precio del activo refleja en todo momento el valor esperado por un inversor hipotético que tiene toda la información y capacidad de procesamiento a su disposición.

Las estrategias de arbitraje son las principales estrategias de HFT que afecta al mercado de forma positiva.

Por otra banda, el trading estructural se aprovecha de la falta de eficiencia de los mercados para crear beneficios. Es por este motivo, que los agentes que trabajan con estos métodos prefieren que exista una eficiencia menor.

Debido a que, como se ha comentado con anterioridad, la mayor parte del HFT que existe se comporta de forma pasiva, y por lo tanto utiliza las estrategias de creación de mercado, los estudios de HFT deberían mostrar que la eficiencia aumenta con la existencia del Trading de Alta Frecuencia. Este hecho queda plasmado tanto en los trabajos de Bohemer (2012) [22] (anteriormente mencionado), como de Gerig (2015) [25], aunque este último, aún indicar que el HFT implica que el mercado sea mas eficiente, también alerta de los posibles errores a los cuales puede inducir el Trading de Alta Frecuencia si el mercado se encuentra en una situación de estrés.

1.5. HFT y Flash Crashes

Un Flash Crash es una caída instantánea y muy profunda de un conjunto de valores de un mercado que, transcurrido un breve periodo de tiempo, recupera sus valores originales. Las causas de los Flash Crashes no son muy claras, pero tradicionalmente se ha culpado al Trading de Alta Frecuencia de ser los causantes de estos hechos. Las consecuencias, por otro lado, si son muy claras, y es que en unos breves instantes de tiempo se producen grandes variaciones en el volumen de las acciones de un agente y, por lo tanto, son momentos donde se pueden generar grandes beneficios, pero a la vez, puede llevar a pérdidas descomunales y muy difíciles de recuperar para algunos agentes principalmente, si estos utilizan técnicas de trading automático.

Figura 1.7: Flash Crash de la cotizacion de Ethereum el 21 de Junio de 2017.
Fuente: https://hardzone.es/2017/06/23/la-criptomoneda-ethereum-perdio-96-valor-una-hora/

Tal y como se ha mencionado anteriormente, el Trading de Alta Frecuencia siempre ha sido muy criticado por ser el causante de los Flash Crashes. Esto es debido a que los HFT son los primeros en darse cuenta de estos errores en el mercado y, por este motivo, actúan en consecuencia y realizan las operaciones necesarias para no tener pérdidas tal y como haría un bróker normal si tuviera la capacidad de reacción que tienen los HFT. Es por este motivo, que los HFT sí pueden llegar a potenciar los Flash Crashes pero, como veremos se puede apreciar en el punto siguiente, no son los causantes de ellos.

1.5.1. Flash Crash del 2010

En este apartado se va a analizar el Flash Crash del 6 de mayo del 2010, el mayor y más estudiado Flash Crash hasta la fecha.

El 6 de mayo de 2010 los índices del SP500, el Dow Jones y el Nasdaq (los principales índices de estados unidos) empiezan a cotizar como otro día normal. Las horas van sucediendo y parece que los precios son bastante estables. A partir de la una del mediodía se empieza a observar una leve tendencia bajista en el Down Jones, pero nada fuera de lo normal. Llegan las dos del medio día y los mercados proceden con la misma tendencia, el DJIA se encuentra aproximadamente en los 10700 puntos, pero la tendencia seguía siendo a la baja. Entonces llegaron las 14:32, la hora fatídica, durante los siguientes 36 minutos, el DJIA bajó 1000 puntos (casi un 10 por ciento del valor del índice) para, luego de un periodo de 5 minutos donde se cerraron los mercados debido a la incertidumbre ocasionada, volver a recuperar gran parte de la cotización del índice. 36 minutos donde hubo caos, muchas personas vendieron por el pánico, muchas otras se arruinaron y, algunas, ganaron mucho dinero con la vuelta a la normalidad. ¿Pero, que había pasado?

En estos 36 minutos se había producido un Flash Crash y nadie sabía el motivo. En un principio salieron una serie de teorías para intentar explicarlo: Que si se había realizado por error una orden extremadamente grande en el mercado por error, que si los HFT habían utilizado sus técnicas direccionales para causar el terror y así generar beneficios, que si la estructura del mercado tenía fallas técnicas...

Evidentemente, pasaron meses y, con la nueva moda de los HFT la gente pensó que ellos habían sido los culpables. Por suerte, 5 meses más tarde del evento, la SEC (la CNMV de Estados Unidos) intervino para poner luz en el asunto. El HFT no había sido el causante, pero sí hizo de potenciador de los efectos de la crisis.

Las causas del Flash Crash fueron que ese día, el mercado americano se encontraba muy fragmentado y frágil, y unos movimientos muy bruscos podrían derrumbarlo. Algunos indicadores alertaron de ello y esto hizo aumentar la volatilidad. A este motivo se le suma que un fondo de inversiones realizó una venta inusualmente grande de los valores de E-Mini SP. Esto provocó que hicieran falta compradores y, por lo tanto, el precio se derrumbara. Cuando esto ocurrió, los HFT de percataron de este movimiento y decidieron vender a toda prisa para minimizar las pérdidas debido a que se encontraban frente un escenario que no les era nada favorable. Este hecho provocó una reacción en cadena que forzó al mercado americano a parar la cotización unos minutos para estabilizar los precios. Una vez pasados estos minutos, el mercado volvió a la calma de nuevo y los precios se fueron estabilizando hasta llegar a los precios en los que encontraban minutos antes. [26]

Este hecho, y los vistos en los artículos de los apartados anteriores, muestran como el HFT no es por si solo el causante de los Flash Crashes, pero si puede ser un potenciador de sus efectos. Es por este motivo que desde los hechos de mayo de 2010 los gobiernos han hecho grandes esfuerzos por regular estos tipos de mecanismos de trading.

En el siguiente apartado (Regulación del HFT) se explican cuales son los diferentes mecanismos de regulación creados por los diferentes países para prevenir posibles fallas de mercado como las vistas en este apartado y, de esta manera, proteger tanto los diferentes mercados de valores, así como los inversores minoritarios que se encuentran en los mismos.

1.6. Regulación del HFT

Como se ha podido observar en anteriores apartados, el HFT aporta liquidez y eficiencia en los merados, pero en momentos de estrés puede ser un arma de doble filo porque puede ayudar a la creación de Flash Crashes. Por este motivo, existen 2 tendencias en las regulaciones de los diferentes mercados.

Por una banda, encontramos los mercados emergentes y de poca liquidez. En estos mercados, las regulaciones respecto el HFT son más permisivas e intentan mejorar las infraestructuras para atraer a estos tipos de traders. El motivo por el cual realizan estas prácticas es debido a que sus mercados se encuentran con una liquidez y eficiencia limitadas debido al hecho de que no tienen una masa suficientemente grande de agentes que operen en dichos mercados. En estos mercados podemos encontrar perfiles como Hong Kong, Japón, Méjico, Rusia, Nueva Zelanda o Brasil.

Por otra banda, encontramos los mercados más establecidos. Mercados que tienen una liquidez y una eficiencia en los mercados muy elevada debido a la gran cantidad de agentes que operan en estos mercados, la estabilidad de los propios países donde operan, el gran tamaño de los mercados (tanto respecto en el elevado número de empresas que operan como en el volumen de las mismas) y el largo recorrido de los mismos. Estos mercados tienen una directriz diferente a los primeros respecto a la regulación del HFT, ya que sus objetivos no son aumentar la liquidez y la eficiencia de estos, sino prevenir fallas en el mercado que puedan producir nuevos Flash Crashes, así como proteger a otros tipos de inversores de las malas prácticas de grandes inversores como pueden ser los HFT. Estos mercados se pueden resumir en los mercados estadounidenses y europeos. Los cuales aplican políticas similares en los mercados.

1.6.1. Regulación Europea

Este trabajo se va a centrar en explicar la regulación que existe en la actualidad en el conjunto de los mercados europeos y, por lo tanto, en España. Para ello, se va a explicar la regulación más reciente llamada MiFID II.

La MiFID II es la directiva europea relativa a los mercados de instrumentos financieros y se encuentra basada en las reglas previamente adoptadas en las directrices MiFID. La directiva MiFID II es la encargada de regular los mercados financieros europeos desde el 3 de enero de 2018 y tiene cuatro principales objetivos. En primer lugar, persigue que la negociación organizada se desarrolle en plataformas reguladas. Seguidamente, introduce

reglas sobre negociación algorítmica y de alta frecuencia. En tercer lugar, mejora la transparencia y la supervisión de los mercados financieros, incluido los mercados de derivados, y aborda determinadas carencias de los mercados de derivados sobre materias primas. En último lugar, refuerza la protección del inversor y las normas de conducta, así como las condiciones de competencia en la negociación de instrumentos financieros. [27]

Respecto al segundo punto, las reglas de negociación algorítmica y de alta frecuencia, la unión europea reconoce que estos tipos de negociación pueden ser positivos debido a que mejoran la liquidez y la eficiencia de los mercados. Sin embargo, estas ventajas pueden ser aprovechadas por los HFT para realizar algunas estrategias nocivas para manipular los precios de mercado (tal y como se había observado en apartados anteriores). Es por este motivo, que se crean un conjunto de reglas que se deben llevar a cabo por las empresas que hagan uso de la negociación algorítmica. [28]

Las medias, tal y como indica el artículo de la empresa Cuatrecasas, son las siguientes:

- Implementación de sistemas efectivos de control de riesgos y limitación y prevención del envío de órdenes erróneas.
- Envío al regulador, al menos anualmente, de información sobre la naturaleza, parámetros y estrategias del algoritmo.
- Las estrategias de negociación algorítmica deberán funcionar de manera continua durante el horario de negociación y colocar cotizaciones en firme a precios competitivos, asegurándose así su provisión de liquidez al mercado.
- Implantación de sistemas y controles efectivos que garanticen la evaluación y revisión apropiadas de la idoneidad de las personas que utilizan el servicio de acceso electrónico directo a un centro de negociación, que garanticn que tales usuarios no excedan de ciertos umbrales de negociación y de crédito preestablecidos.
- Imposición a los centros de negociación de una serie de procedimientos encaminados al control del riesgo aparejado con los posibles excesos derivados de este tipo de prácticas.

Como se ha podido apreciar, y tal y como indicábamos al principio de este apartado, la Unión Europea, con unos mercados eficientes y bien estructurados, acepta el uso del HFT en sus mercados debido a que este tipo de operativas genera beneficios en los mismos al aumentar la eficiencia y la liquidez. Pero, para evitar problemas de liquidez en momentos en los que el mercado se encuentra en momentos de estrés, decide implementar mecanismos regulatorios que aseguren una liquidez en el mercado en los momentos más críticos para evitar nuevas fallas del mercado como la vivida en el Flash-Crash de 2010.

Capítulo 2

Creación de un algoritmo

En la segunda parte del trabajo, el objetivo consiste en crear un algoritmo de trading automático que se aproxime lo máximo posible al HFT.

Como se ha indicado en el apartado de metodología, este algoritmo estará creado en el lenguaje de programación Python, y utilizará indicadores técnicos usados por la mayoría de analistas del mundo (pero adaptados al trading intradía), así como redes neuronales en el entorno de trabajo Keras[1] para poder obtener una rentabilidad superior a la esperada si se eligiera una cartera con un peso proporcional de todas las acciones del mercado.

El mercado utilizado para la realización del algoritmo será el NASDAQ. En concreto, un conjunto de 80 empresas dentro del índice NASDAQ-100. El periodo temporal de los datos será de 2 años, empezará el 10 de mayo de 2019 y terminará el 23 de abril de 2021. Estos se tendrán en un intervalo de 1 minuto de diferencia. Mientras que los datos de 2019 y 2020 serán utilizados para realizar los procesos de Training y Validation, los datos del 2021 serán utilizados para el proceso de Test. De este modo, se evaluará la utilidad del modelo con los datos de los primeros meses de 2021.

El algoritmo constará de dos partes diferenciadas que serán unidas para llegar a el objetivo. Primero se realizará un Análisis Técnico de los diferentes valores con diferentes indicadores para encontrar tendencias en los valores. Seguidamente se utilizará una Red Neuronal LSTM para predecir cual va a ser la diferencia de cotización esperada entre dos periodos de tiempo a partir de los 60 minutos anteriores. Finalmente, mediante una Red Neuronal Feedforward se obtendrá un valor entre 0 y 1 que indicará si en cada minuto de tiempo, un valor se debe mantener en cartera un cierto periodo de tiempo o no a partir de los resultados obtenidos en la red LSTM y los diferentes indicadores técnicos.

[1] https://keras.io/

2.1. Limitaciones

Como se puede apreciar, el objetivo de esta segunda parte del proyecto no es simular una estrategia de HFT, sino crear una estrategia de trading automático que sea lo más parecida posible al Trading de Alta Frecuencia intentando utilizar algunas de las características y estrategias que se han mencionado en la primera parte del trabajo. Este hecho es debido a que recrear un algoritmo de Trading de Alta Frecuencia no tiene sentido en este proyecto debido a que no es posible simular muchas de las condiciones que utilizan los HFT debido a que el hardware que puede tener a su disposición . Para poder entender este hecho, analizamos las posibilidades que puede tener un pequeño usuario sobre los HFT a partir de las características y estrategias analizadas en la primera parte.

2.1.1. Limitaciones en las características

En el apartado de las características, vemos como el Trading de Alta Frecuencia utiliza principalmente la velocidad y la baja latencia como principales herramientas de obtención de beneficios debido al hecho de poder encontrarse a una distancia muy cercana de los servidores donde se realiza el trading. Este hecho permite al HFT poder recibir y ejecutar las operaciones en fracciones ínfimas de segundo. Además, gracias a su gran capacidad de computación, pueden realizar un gran número de órdenes en cada una de estas fracciones de segundo.

Debido al hecho que un pequeño usuario no tiene esta alta capacidad de computación ni una latencia tan baja, ya que los pequeños agentes no se suelen encontrar cerca de los servidores de los mercados, el algoritmo que se va a utilizar en este trabajo utiliza las distancias entre cotizaciones lo más pequeñas posible con el fin de tener un gran conjunto de oportunidades en las que invertir cada día y, a la vez, acercarse lo máximo posible a la simulación del HFT pero sin poner en riesgo la capacidad de los ordenadores que puede tener un usuario en su casa para realizar estas operaciones. Por este motivo, los intervalos escogidos tienen una diferencia de un minuto.

2.1.2. Limitaciones en las estrategias

En el apartado de las estrategias, se observa que el HFT utiliza 4 tipos de trading (creación de mercado, arbitraje, estructural y direccional) y que, además, puede utilizar mercados alternativos como las dark pools. En primer lugar, un pequeño inversor no es capaz de anticiparse al mercado más que otros tipos de agentes, por lo que el arbitraje

queda descartado para este tipo de trader. En segundo lugar, el pequeño inversor no tiene suficiente peso en el mercado como para hacer que el precio de una acción oscile a su voluntad y, por lo tanto, no puede usar las estrategias estructurales. Debido a que el pequeño inversor debe defenderse de estrategias estructurales por parte de otros agentes tampoco puede usar las estrategias de creación de mercado. Por este motivo, la única estrategia que puede utilizar un inversor pequeño es el trading direccional.

Debido a los puntos anteriormente mencionados, el algoritmo que se va a utilizar va a elegir sus operaciones siguiendo estrategias de trading direccional, las cuales son las más utilizadas por la mayor parte de pequeños inversores. Dentro de estas estrategias, el objetivo del algoritmo será observar patrones y tendencias en la cotización para, de este modo, suponer que empresas van a aumentar su valor y cuáles no.

2.2. Presentación del caso

Una vez se ha elegido el lenguaje de programación, se debe proceder a programar el código, pero antes, hay que tener claros los objetivos. Es por este motivo, que en este apartado se explica cuáles son las conclusiones a las que se quiere llegar, cuales son los datos de partida y como es el algoritmo que nos permite llegar desde los datos hasta las conclusiones.

2.2.1. Objetivo del Algoritmo de Trading

El objetivo de este algoritmo de trading automático es, como ya se ha explicado con anterioridad, obtener un método que nos permita tener unos mayores beneficios que la media del mercado en el cuál se está invirtiendo si se compraran i se vendieran las acciones minuto a minuto en el mismo periodo temporal en el cuál el algoritmo creado en este trabajo se encuentra operativa (aproximadamente cada día desde las 10:30 hasta las 16:00 horas).

Para ello, se van a poder realizar órdenes de compra y venta de acciones en intervalos de un minuto de duración, pero nunca se va a poder realizar operaciones en corto (vender antes de comprar para obtener beneficios con las pérdidas de cotización de un valor) ni será posible apalancarse (utilizar dinero prestado que habrá que devolver con intereses para invertir). Los motivos de estas dos restricciones vienen derivados del hecho de que en nuestro algoritmo no se van a tener en cuenta las comisiones debido a que ya existen algunas plataformas en el mercado (como por ejemplo Robinhood [2]) que permiten operar sin comisiones pero abrir posiciones en corto o apalancarse si cuestan dinero debido a que otros agentes prestan el dinero para que el inversor pueda realizar esas operaciones.

2.2.2. Elección de mercado y datos

En este caso se van a utilizar los datos del NASDAQ (National Association of Securities Dealer Automated Quotation), la bolsa de valores electrónica más grande de los Estados Unidos.

La elección de este mercado es debido, en primer lugar, al hecho que al ser una bolsa de valores electrónica, todos los agentes tienen que realizar sus operaciones a través de ordenadores y, por lo tanto, unifica los métodos de compraventa de acciones. En segundo

[2]https://robinhood.com/us/en/

lugar, se ha elegido el NASDAQ debido a que tiene una gran variedad de empresas en su mercado y, a la vez, el número de agentes que realizan operaciones en este mercado permite que existan una gran liquidez y una gran eficiencia en las diferentes cotizaciones. Por último, cabe destacar que la recolección de datos históricos a corto plazo en algunos mercados es muy difícil de conseguir, pero en el NASDAQ es posible adquirir estos valores en un intervalo de 2 años.

Debido al gran volumen de empresas que cotizan en este mercado (aproximadamente unas 3.300) y el gran número de variables que se utiliza por cada una de ellas en estos dos años (más de 150.000 valores de cierre y unos 2 millones si tenemos en cuenta el precio de cierre, el máximo, el mínimo, el precio de apertura y el volumen) es imprescindible reducir el tamaño del conjunto de datos para que los ordenadores puedan procesar los datos sin problemas de memoria. Por este motivo, se ha decidido elegir el índice NASDAQ-100, un índice con las 100 empresas más importantes del NASDAQ para realizar el caso práctico. El hecho de elegir las empresas que forman parte del índice del NASDAQ-100 permite reducir el volumen de datos mientras se mantiene una variedad de empresas suficiente en el mercado. Además, únicamente se utilizan las variables de cierre y volumen para cada minuto de tiempo, ya que de este motivo se puede reducir a menos de la mitad el uso de recursos del ordenador sin sufrir grandes alteraciones en las cotizaciones debido a que el tiempo entre datos es tan reducido que la variación es mínima.

2.3. Creación del caso

En este apartado se puede apreciar la evolución del algoritmo desde la recolección de los datos hasta la recogida de los resultados. Para ello, se va a explicar como fue evolucionando todo el código desde que se buscó un lugar desde el cuál poder crear una base de datos hasta la eliminación de errores y la manipulación de datos faltantes. Seguidamente, se presentarán los diferentes indicadores de análisis técnico, a continuación se presentarán las redes LSTM y ,finalmente, se realizará la Feedforward Neural Network para obtener los resultados finales.

2.3.1. Descarga de datos

El primer paso para poder crear el algoritmo que permita realizar trading automático de manera similar a cómo se hace en el Trading de Alta Frecuencia es descargar los datos. Pero, para ello, se ha de tener definido cuáles van a ser los datos que se van a utilizar.

Tal y como se indica en el apartado de elección del mercado y los datos, el mercado elegido ha sido el NASDAQ pero, debido a que este mercado tiene un gran volumen de empresas, se ha decidido elegir el índice NASDAQ-100, y por lo tanto el conjunto de empresas que lo forman, para realizar las operaciones.

El primer paso fue recopilar el conjunto de empresas que forman este índice. Para ello, se lee directamente de la página de Wikipedia [29] cuáles son las empresas que forman este índice y se recopilan en un vector.

A continuación se deben recopilar los datos minuto a minuto de cada empresa del índice durante los últimos dos años. Este es el paso más complejo debido a que estos datos no suelen ser almacenados debido al gran espacio que implica almacenarlos y, debido a eso, la mayoría de veces solamente se pueden adquirir pagando grandes cantidades por ellos. Finalmente, utilizando una API llamada Alpha Vantage[3] se consigue obtener un permiso para descargar datos de los dos últimos años minuto a minuto. El inconveniente es que solo se pueden realizar 5 peticiones por minuto y un máximo de 500 peticiones por día. Una petición viene definida como la descarga de los datos de una única empresa en un único mes y, por lo tanto, debido a que el NASDAQ-100 consta de 100 empresas y un año tiene 24 meses, se debían realizar un total de 2400 peticiones.

Para optimizar cada petición se decidió crear un programa que automatizara la des-

[3]https://www.alphavantage.co/

carga de estos datos. Para ello, lo que se decidió crear es bucle que realizara peticiones de 4 momentos temporales cada minuto con un tiempo de espera de 1 minuto entre ellos para que no hubiera errores y, cada día, se realizaba esta operación 100 veces para cada una de las empresas. De este modo, cada día que pasaba se conseguía obtener la cotización de las 100 empresas y, para no perder este trabajo, se guardaba todo en un archivo CSV.

Finalmente, después de una semana, se acaban de descargar todos los datos necesarios y, entonces, es momento de verificar si los datos son correctos. Para ello, se comprueban los máximos, los mínimos, los valores de cierre y los valores de apertura de los diferentes valores y se comparan con los de otras páginas web que si ofrecen estos datos en diario. Una vez se sabe que los datos son fiables, se pasa a la eliminación de los datos no necesarios (quedándonos únicamente con los datos de cierre y volumen de cada minuto. Seguidamente, se eliminan o modifican los valores NAs. Cabe destacar, que existen algunos valores no especificados en la base de datos. Ya sea porque en esos momentos no hubo cotización, o porque ésta no tubo variaciones. Para poder trabajr sin problemas, se decide eliminar las empresas que tengan una tasa de NAs mayor al 5 por ciento. Para las demás, se sustituye el precio de cotización con el precio anterior de cotización (a no ser que sea el primero, ya que en ese caso se realiza con el posterior) y el precio del volumen pasa a ser 0.

De esta manera, se acaba obteniendo un conjunto de 80 empresas del NASDAQ-100 con su cotización, minuto a minuto, de los dos últimos años (Tabla 2.1). Esto implica un total de más de 15 millones de puntos de datos diferentes. Y, con la descarga de datos hecha, ya se puede comenzar a trabajar en el algoritmo.

	Valor_ATVI	Volumen_ATVI	Valor_ADBE	Volumen_ADBE
Fecha				
2019-05-10 09:31:00	45.858518	52234.0	273.905	36208.0
2019-05-10 09:32:00	45.873350	12900.0	274.344	999.0
2019-05-10 09:33:00	45.828856	4341.0	273.415	9981.0
2019-05-10 09:34:00	45.779418	3401.0	273.465	4384.0
2019-05-10 09:35:00	45.784362	12724.0	273.500	4609.0
2021-04-23 15:55:00	93.085000	23698.0	517.020	35538.0
2021-04-23 15:56:00	93.060000	48084.0	516.210	28831.0
2021-04-23 15:57:00	93.090000	37703.0	516.060	23402.0
2021-04-23 15:58:00	93.125000	40039.0	516.690	19681.0
2021-04-23 15:59:00	93.115000	42596.0	516.340	28282.0
	Valor_XLNX	Volumen_XLNX	Valor_ZM	Volumen_ZM
Fecha				
2019-05-10 09:31:00	113.203758	77443.0	75.8435	14220.0
2019-05-10 09:32:00	114.139246	29274.0	76.9900	22610.0
2019-05-10 09:33:00	114.080778	9978.0	76.9400	15744.0
2019-05-10 09:34:00	113.798183	10068.0	77.0000	6572.0
2019-05-10 09:35:00	114.022310	12233.0	77.2500	5897.0
2021-04-23 15:55:00	129.780000	17373.0	337.8900	56280.0
2021-04-23 15:56:00	129.680000	10594.0	336.5900	57056.0
2021-04-23 15:57:00	129.720000	15023.0	337.0500	20821.0
2021-04-23 15:58:00	129.755000	17959.0	337.1150	21557.0
2021-04-23 15:59:00	129.710000	20796.0	336.7100	32925.0

Cuadro 2.1: Tabla con los primeros y últimos valores del conjunto de datos

2.3.2. Análisis Técnico

En este apartado se muestran diferentes indicadores de análisis técnico, se explica como han sido creados, su fórmula, qué se puede obtener con ellos y, finalmente, se muestra un ejemplo de la creación de cada uno de ellos.

Los indicadores que han sido utilizados en este trabajo son los siguientes: Medias Móviles de 5, 15 y 60 minutos, Momentos de 5, 15 y 60 minutos, RSI de 15 y 60 minutos, Estocástico con K=55 y D=5 minutos, Williams de 15 y 60 minutos y MACD 12, 16 con señal de 9 minutos.

Tal y como se puede apreciar, todos los indicadores utilizan un periodo inferior o igual a una hora para realizar los cálculos. Esto es debido a que un requisito indispensable es que cada vez que se cambia de día, no se pueden utilizar los primeros minutos de ese día para realizar los cálculos debido a que el cambio de día podría causar errores en los cálculos. Por este motivo, para poder tener suficiente margen para operar cada día,

se decide prescindir de la primera hora y, de este modo, tener una base suficiente para realizar las distintas operaciones.

Es necesario mencionar que todos los gráficos que se realizan en este apartado se pueden comparar entre sí debido a que todos utilizan el mismo periodo temporal y la misma empresa. En concreto, se utiliza el día 14 de Mayo de 2019 y la empresa en cuestión es Cisco Systems. Además, el conjunto de indicadores y osciladores ha sido creado y definido en función del libro de John Murphy llamado Análisis técnico de los mercados financieros [30] .

2.3.2.1. Médias Móviles

Las medias móviles son uno de los indicadores técnicos más utilizados en todo el planeta por la facilidad de su construcción y su interpretación. Además, es la base de muchos otros indicadores, tal y como se podrá apreciar en los siguientes puntos. Además, a diferencia de los otros indicadores que se explicarán más adelante, las medias móviles son excelentes cuando se trata de seguir tendencias alcistas o bajistas. En cambio, los osciladores que se presentan seguidamente, sirven para analizar mercados en los que no existe una tendencia clara y, por lo tanto, el mercado se encuentran en una tendencia de fluctuación horizontal.

La media móvil es el promedio de los diferentes valores de cotización de los últimos días y, por lo tanto, es una forma muy simplificada de seguir tendencias. Cuanto mayor es el número de días que se elije para calcular el promedio de la media móvil mayor es la tendencia que se capturará y esta se verá menos afectada por cambios bruscos de la cotización. Por otra parte, cuanto menor sea el número de periodos elegido, más rápido se podrán detectar los cambios de tendencia. Por este motivo, se dice que la media móvil es un método que suaviza la información de los precios pero, como contrapartida, añade cierto retraso en el hecho de presentar la información.

Existen varios tipos de medias móviles: la media móvil simple, la media móvil ponderada linealmente o la media móvil suavizada exponencialmente entre otras. En nuestro caso, se ha elegido la media móvil exponencial debido a que es la más usada por los técnicos y a que la media móvil suavizada exponencialmente será utilizada en un futuro para calcular el MACD.

También es importante elegir correctamente el periodo temporal y, de hecho, no existe una metodología que sea correcta. Por este motivo, la opinión más generalizada es que es importante crear varias medias móviles con diferentes periodos temporales y analizarlas

tanto de manera separada como de manera conjunta.

Para analizar una media móvil de manera aislada, ésta debe ser dibujada en el gráfico junto a la cotización del valor y, cuando el precio de cierre sobrepasa la media móvil, se genera una señal de compra.

Para analizar dos medias móviles de manera conjunta, éstas también deben ser dibujadas en el gráfico junto al al cotización del valor pero, en este caso, se produce la señal de compra cuando la media más corta cruza por encima de la media más larga.

A continuación se puede observar la fórmula de la media móvil simple de n periodos de tiempo (Fórmula 2.1), así como un gráfico donde se pueden observar las medias móviles de 5 y 60 minutos respecto la cotización de un valor (Figura 2.1).

$$MMS = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.1}$$

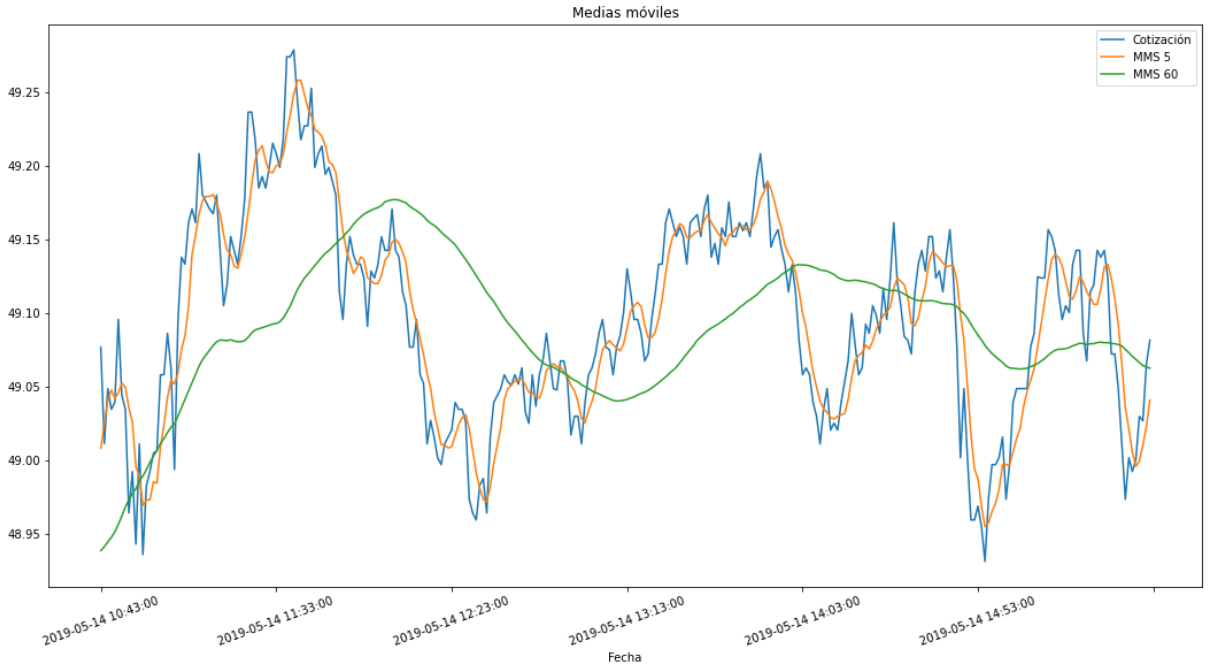

Figura 2.1: Medias móviles de 5 y 60 minutos
Fuente: Elaboración propia

Es importante destacar que, para poder dar señales en todo momento a la red neuronal donde estos datos deben ser insertados, se ha decidido calcular en todo momento cuál es la diferencia de precios que existe entre la media móvil y la cotización (en el caso de que solo se utilice una media móvil), o la distancia entre la media móvil de corto plazo respecto la de largo plazo (en el caso de que sean utilizadas dos medias móviles). Para ello, se han utilizado la fórmula siguiente, donde el numerador es la media móvil de mas corto alcance

o la cotización, y el denominador es la media móvil de mayor alcance (Fórmula 2.2). También se puede apreciar un gráfico con los resultados de las diferencias de cotización en función de cada periodo temporal (Figura 2.2). Cabe destacar que se usan logaritmos debido a que este hecho permite que las diferencias entre los valores puedan ser aditivas entre sí.

$$Dif_{MMS} = \ln(\frac{Val_{Corto}}{MM_{Largo}}) * 100 \tag{2.2}$$

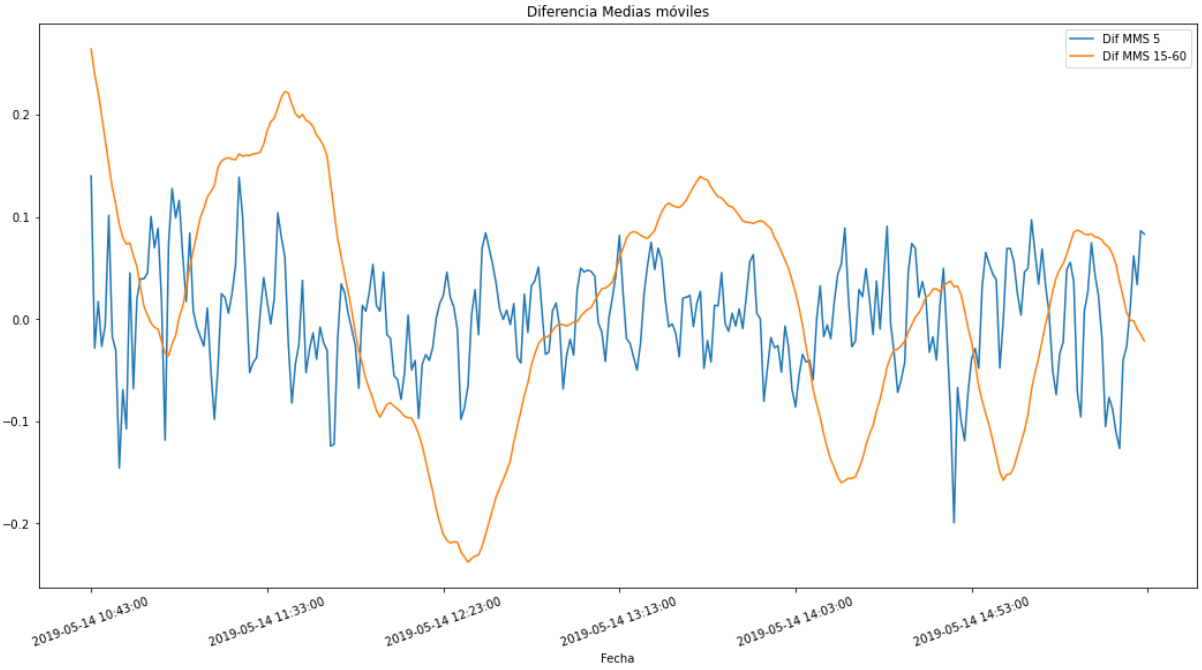

Figura 2.2: Diferencia de las medias móviles respecto su valor
Fuente: Elaboración propia

2.3.2.2. Momentos

El método de los momentos es el método más simple para crear osciladores. Éste mide la velocidad del cambio de precios respecto al nivel de precios en unos periodos atrás. Para realizar este cálculo, se resta al valor actual el valor del momento de n periodos anteriores (Fórmula 2.3). Si el último precio de cierre es mayor que el de hace n días, el precio se habrá movido al alza y éste valor se encontrará por encima de la línea del cero. Al igual que en las medias móviles, un periodo menor de los momentos implica una mayor sensibilidad a las oscilaciones más pronunciadas, mientras que un periodo mayor en los momentos implica una menor volatilidad del oscilador. A continuación se puede ver un gráfico explicativo del oscilador (Figura 2.3).

$$Mom = V - V_n \tag{2.3}$$

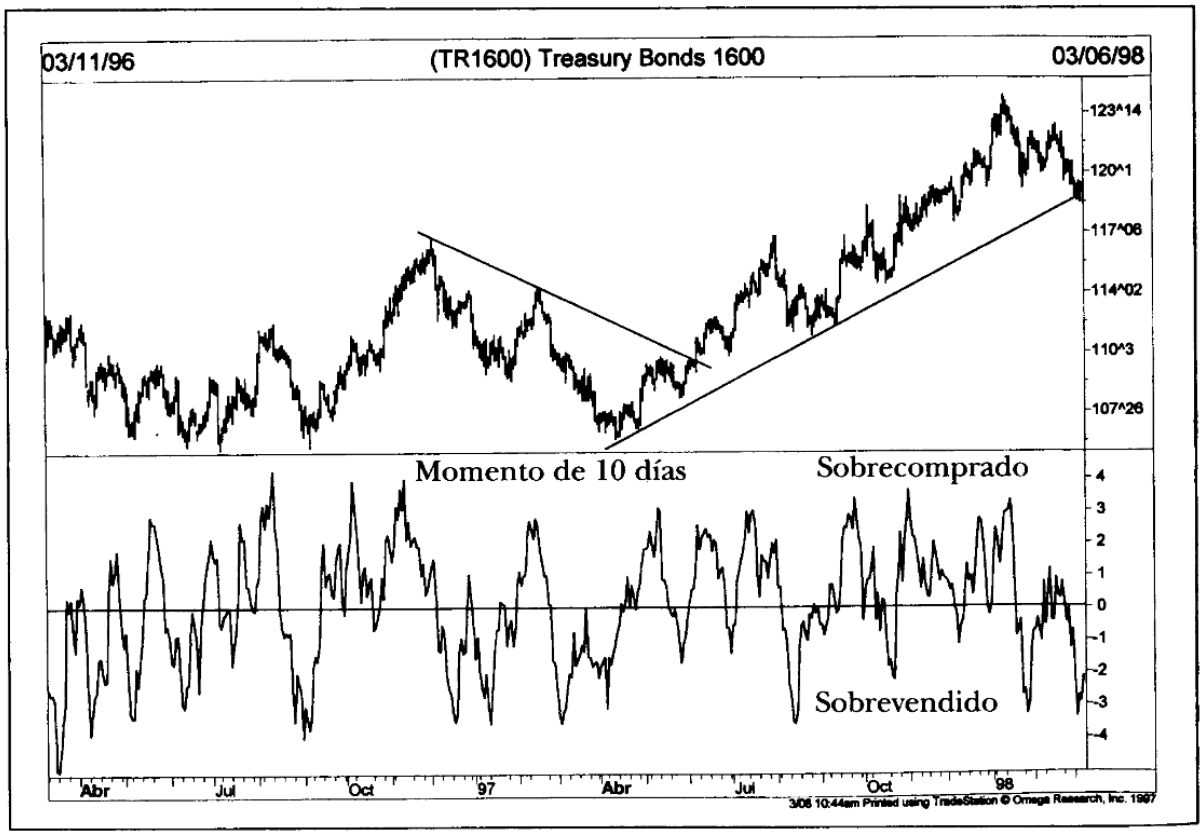

Figura 2.3: Momento de 10 días junto a la cotización de un valor
Fuente: Análisis técnico de los mercados financieros, John Murphy

Al igual que con las medias móviles, se han realizado unas modificaciones en el método de los momentos para poder dar señales en todo momento a la red neuronal. Por este motivo, se ha decidido calcular para cada momento de tiempo la tasa de cambio respecto la cotización (Fórmula 2.4). A continuación se puede observar un gráfico de cuáles son los valores que se insertan en cada momento en la red neuronal (Figura 2.4). Tal y como se puede apreciar, la fórmula consta de un cociente entre el Momento de orden n (visto en la fórmula 2.3) y el valor de la cotización en ese momento temporal. Finalmente, se multiplica por 10 su valor para obtener números más agradables a la vista y que se encuentren entre 0 y 1. Éste método se realiza de esta manera para que el 0 sea el valor central y, de este modo, poder indicar a la red neuronal de manera óptima si un valor se encuentra sobrecomprado o sobrevendido en función de si la línea se encuentra por encima o por debajo del valor en cuestión.

$$Dif_{Mom} = \frac{V - V_n}{V} * 10 \tag{2.4}$$

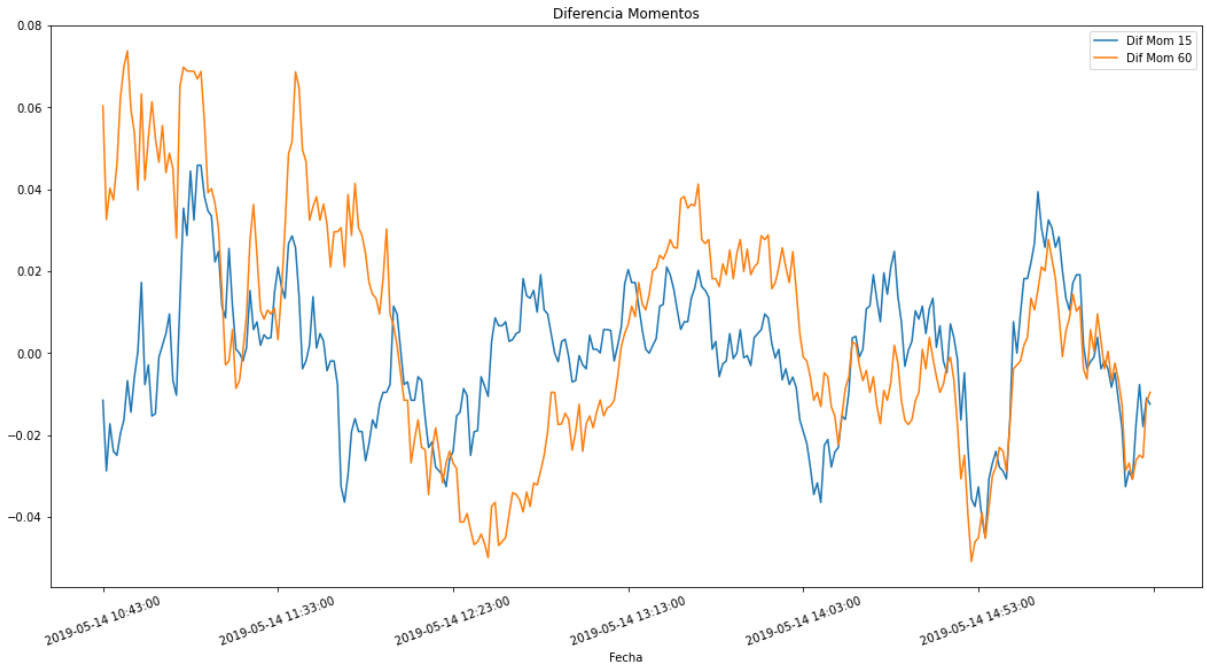

Figura 2.4: Diferencia del método de los momentos respecto su cotización
Fuente: Elaboración propia

2.3.2.3. RSI

El Índice de Fuerza Relativa (RSI por sus siglas en inglés) es un oscilador creado por Welles Wilder el año 1978 y es uno de los más utilizados entre los técnicos debido a que solventa uno de los principales problemas del método de los Momentos, y es que este método, presentado en el apartado anterior, suele causar errores debido a los cambios bruscos en los valores.

El RSI no solo permite suavizar estos errores, sino que además permite que el indicador se encuentre dentro de un intervalo definido entre 0 y 100 (a diferencia del método anteriormente visto, el cuál no se encuentra acotado por la parte superior).

A continuación se puede observar la fórmula del RSI de n periodos (Fórmula 2.5).

$$RSI = 100 - \frac{100}{1 + FR} \quad \text{and} \quad FR = \frac{\frac{1}{k}\sum_{i=1}^{k} CierAlz_k}{\frac{1}{l}\sum_{j=1}^{l} CierBaj_l} \tag{2.5}$$

Al igual que en los apartados anteriores, la fórmula del RSI ha sido modificada para ajustarse a los requisitos de la red neuronal, ya que el hecho de que el conjunto de datos se encuentre estandarizado para todos los indicadores permite a la própia red trabajar de manera más eficiente. De este modo, el RSI se encuentra entre el intervalo [-1, +1] debido

a que se ha dividido el resultado del RSI entre 500 y se le ha restado el valor de 50 al resultado realizado. A continuación se puede apreciar un gráfico con el RSI de 15 y 60 minutos (Figura 2.5).

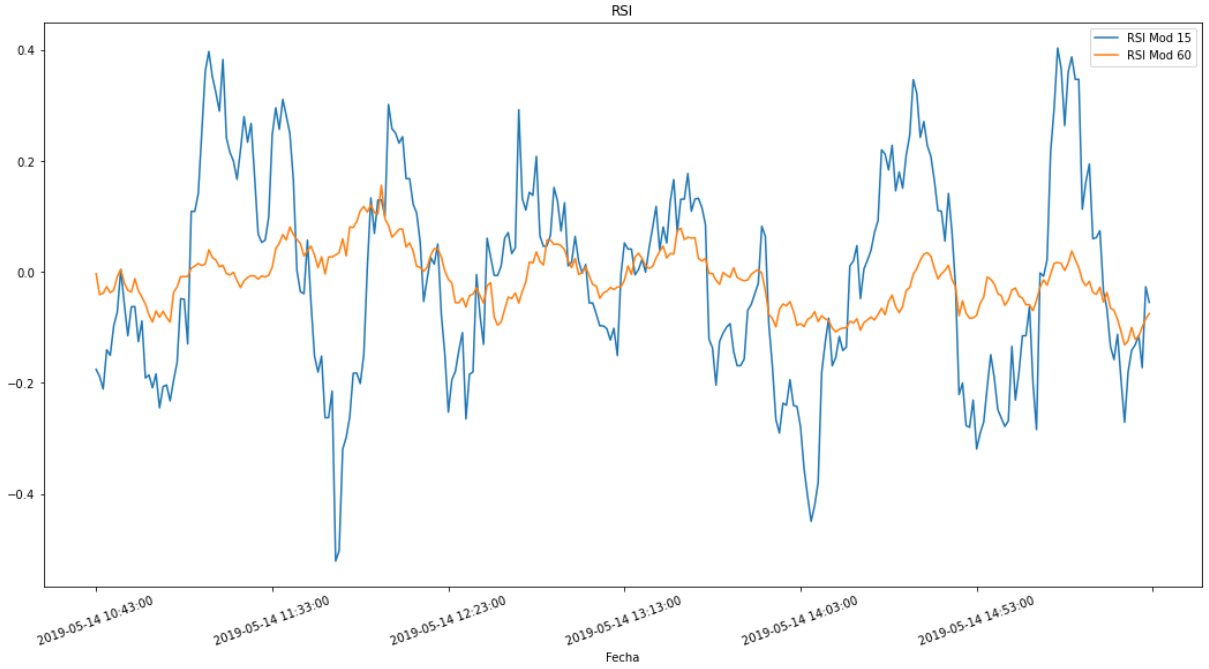

Figura 2.5: RSI modificado de 15 y 60 minutos
Fuente: Elaboración propia

Tal y como se ha mencionado, el RSI estándard se representa en una escala que va de 0 hasta 100. Los movimientos por encima de 70 se consideran sobrecomprados y los que se encuentran por debajo de 30 sobrevendidos. En nuestro caso, en cambio, podemos decir que un valor está sobrecomprado si es positivo y sobrevendido si es negativo.

2.3.2.4. Estocástico

El proceso estocástico (K %D), popularizado por George Lane, se basa en una premisa que los indicadores y osciladores vistos hasta ahora no incorporan. Esta premisa indica que a medida que los precios se incrementan, los precios de cierre tienden a acercarse más al extremo superior de la banda de precios y, por el contrario, en las tendencias a la baja, el precio de cierre tiende a acercarse al extremo inferior de la banda.

En este indicador se utilizan dos líneas (%K y %D). La línea %K es la más sensible de las dos, mientras que la línea %D es la que proporciona las señales principales. Las fórmulas de las diferentes líneas son las siguientes (Fórmula 2.6):

$$\%D = \frac{Cierre - Min_n}{Max_n - Min_n} * 100 \quad \text{and} \, \%K = MMS_m(\%D) \tag{2.6}$$

Debido a que el valor que da las señales es la %K, la media movil de m periodes de %D, éste es el valor que se va a introducir en la red neuronal. Cabe destacar que en este caso el valor central no va a ser el 0, sino el 0.5, ya que los valores mínimo y máximo van a ser los números 0 y 1 respectivamente.

Aquí se puede apreciar el gráfico del proceso Estocástico(Figura 2.6):

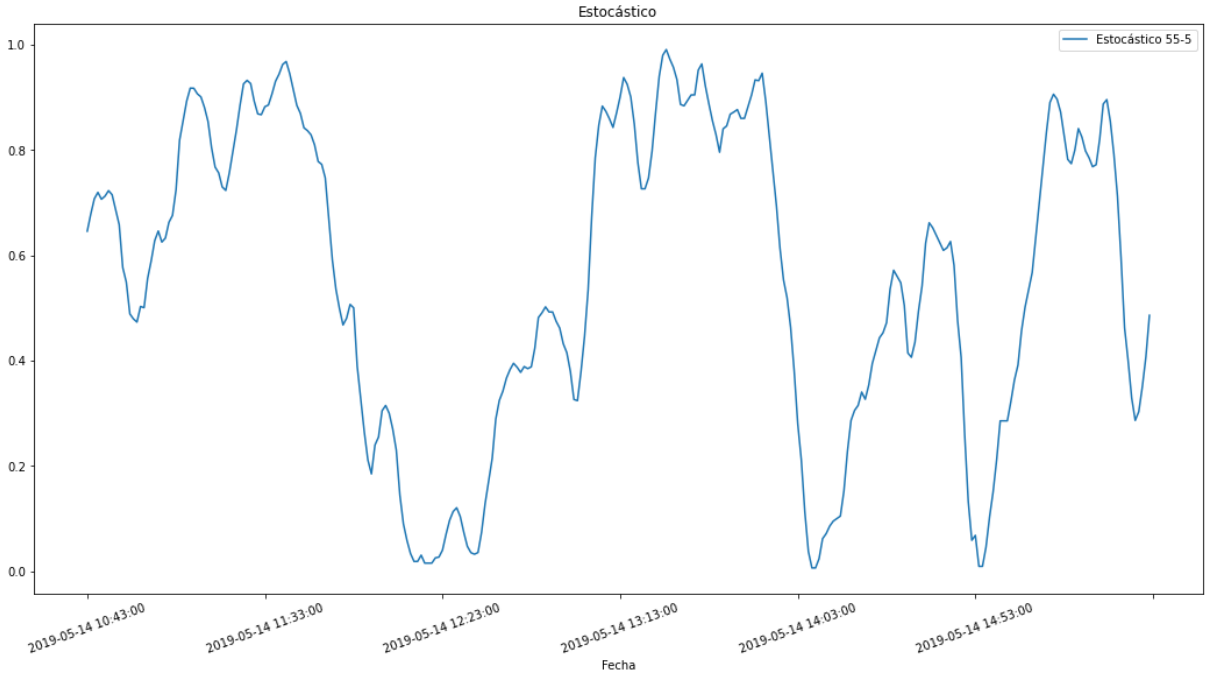

Figura 2.6: Estocástico de %K = 55 y %D = 5 minutos
Fuente: Elaboración propia

Tal y como se ha indicado en los puntos anteriores, aquí se puede decir que un valor está sobrecomprado si se encuentra por encima del 0.5 y sobrevendido si la cotización se encuentra por debajo de esta cota.

2.3.2.5. Williams

El oscilador %R de Williams está basado en un concepto similar al Estocástico debido a que mide el último cierre en relación a su banda de precios de un cierto número de días.

La fórmula del oscilador es la siguiente (Fórmula 2.7):

$$\%R = \frac{Max_n - Cierre}{Max_n - Min_n} \tag{2.7}$$

Como se puede apreciar, en este oscilador el valor actual se resta del máximo. Esto implica que el oscilador debe ser interpretado de manera contraria que la resta de osciladores. Para solventar el problema, se decide invertir el eje del gráfico multiplicando los resultados por -1. De esta manera, obtenemos los mismos límites que en el indicador Estocástico y podemos decir que un valor está sobrecomprado si se encuentra por encima de 0.5 y sobrevendido si se encuentra por debajo de esta cota.

Seguidamente se puede apreciar el oscilador %R de Williams para los periodos de 15 y 60 minutos (Figura 2.7).

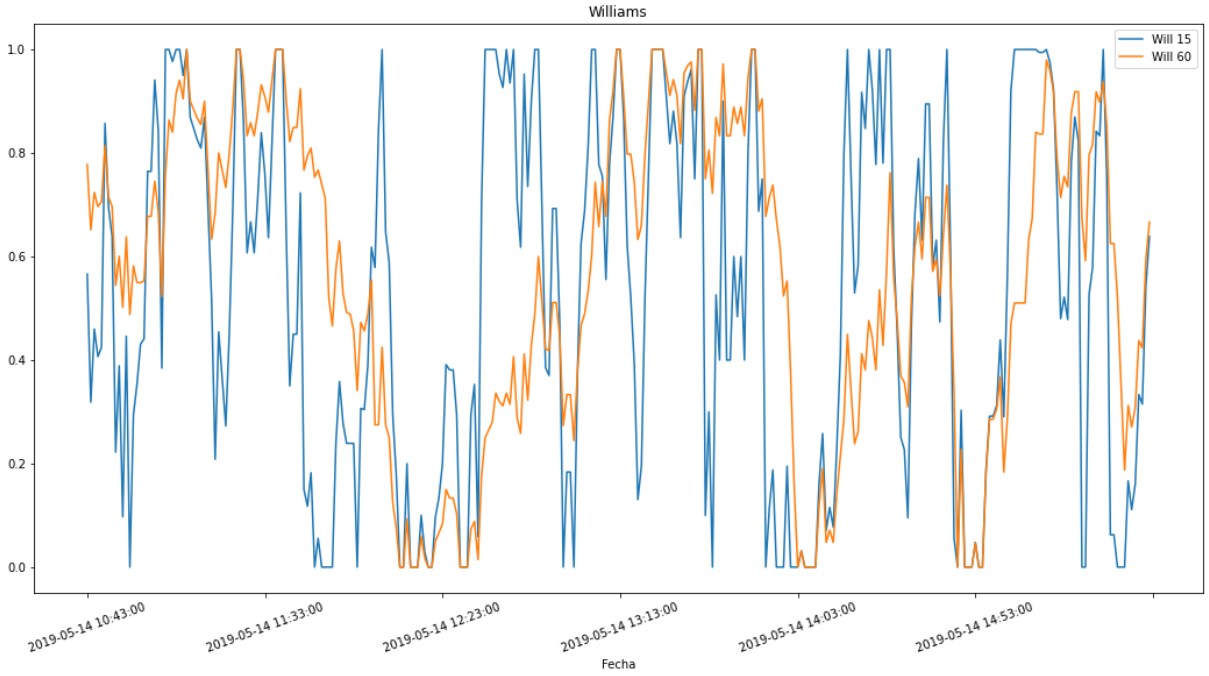

Figura 2.7: Williams de 15 y 60 minutos
Fuente: Elaboración propia

2.3.2.6. MACD

Finalmente se llega al último oscilador de todos. Éste se llama Convergencia/Divergencia de la media móvil, mucho más conocido como MACD por sus siglas en inglés.

Este indicador fue creado por Gerald Appel y es, posiblemente, el más utilizado en la actualidad debido a su gran utilidad, ya que combina los principios de los osciladores con 2 medias móviles. Este hecho convierte en el MACD en el indicador más laborioso de

crear de los vistos en este trabajo, pero su lugar en éste está más que merecido debido a la gran utilidad que ofrece a los inversores de todo el mundo.

A continuación se van a mostrar las diferentes fórmulas de este indicador. En primer lugar, se crean dos medias móviles exponenciales (EMA) (Fórmula 2.8). La primera de ellas, más corta que la segunda. A partir de estas dos líneas se crea la línea MACD como diferencia de la EMA de corto alcance menos la EMA de largo alcance. Finalmente, se crea una Media Móvil Exponencial a partir de la línea del MACD, la cuál será la línea de señal (Fórmula 2.9).

$$EMA_n = ((Valor_{cierre} - EMA_{n-1}) * \frac{2}{n+1}) + EMA_{n-1} \tag{2.8}$$

$$MACD = EMA_{corto} - EMA_{largo} \quad \text{and} \quad Signal_{MACD} = EMA_k(MACD) \tag{2.9}$$

Al igual que se ha hecho en otros apartados, este indicador debe ser modificado para que la red neuronal pueda entender más fácilmente los valores que el MACD produce. A continuación se explica el proceso realizado en el indicador, pero antes se muestra una imagen de cómo funciona el MACD estándar (Figura 2.8).

Las modificaciones que se realizan en el indicador con el fin de que estos valores puedan ser leídos por la red neuronal son los siguientes: En primer lugar, se decide que la señal no vendrá dada por una EMA, sino por una SMA. Este proceso simplifica la fórmula del MACD ya que, de otro modo, el tiempo de ejecución era demasiado elevado y, además, permite que los valores sean más suavizados en el resultado final. En segundo lugar se realiza la diferencia entre el valor del MACD menos el de la Señal en cada periodo de tiempo. De esta manera se obtiene un histograma del MACD que permite mostrar a la red neuronal en que momentos de tiempo la cotización está sobrecomprada o sobrevendida. De hecho, tal y como pasaba en la mayoría de apartados anteriores, si el valor del histograma es un número negativo, se dice que el valor está sobrevendido. De otra forma, si el valor del histograma es un número positivo, el valor estará sobrecomprado (Figura 2.9).

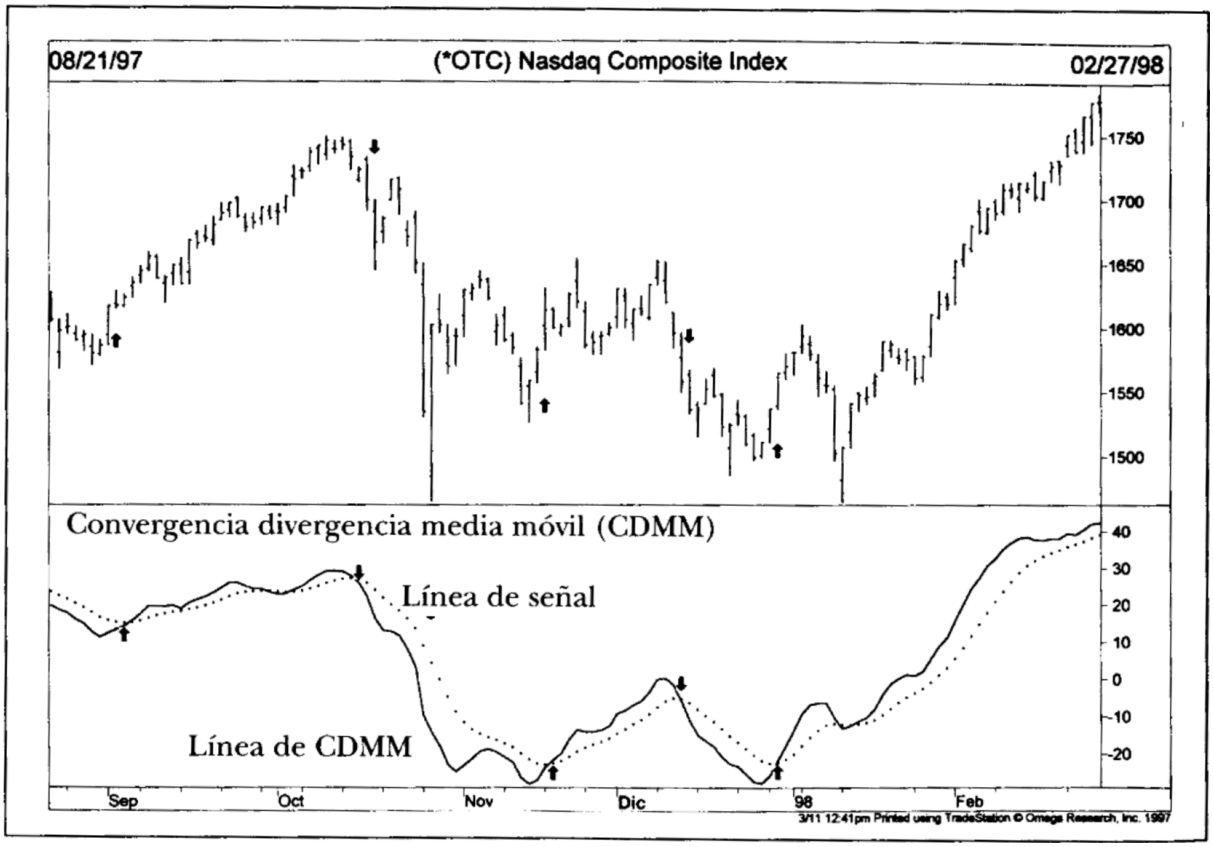

Figura 2.8: Uso e interpretación del MACD
Fuente: Análisis técnico de los mercados financieros, John Murphy

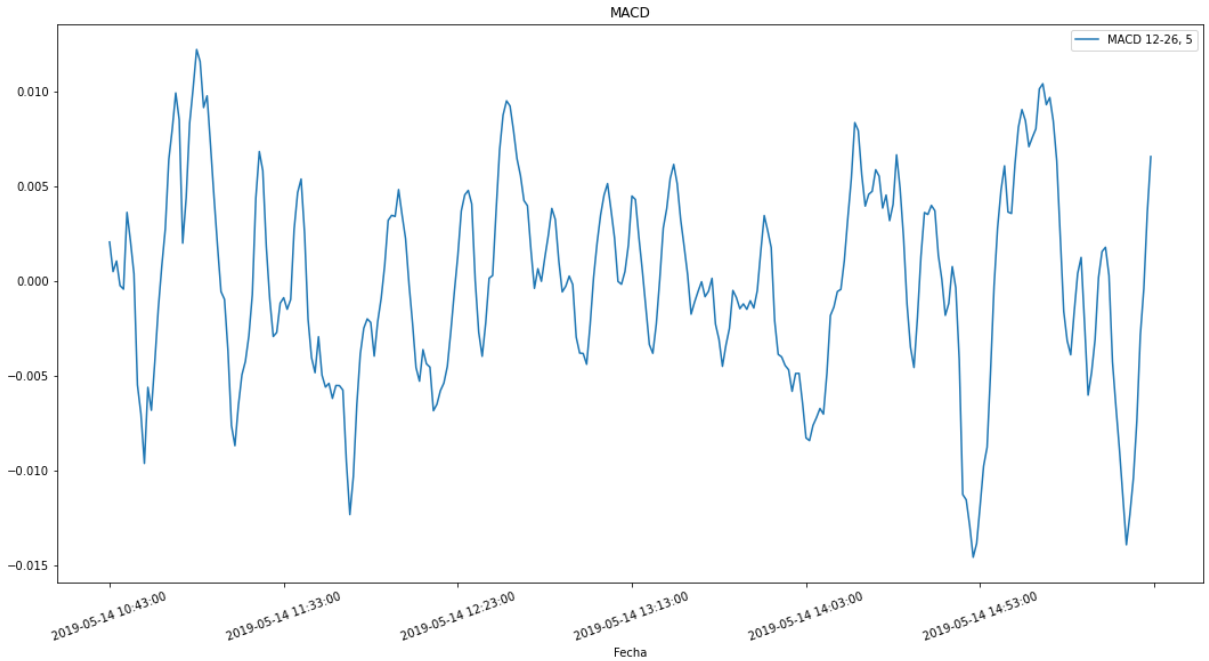

Figura 2.9: Histograma del MACD que se aporta a la red neuronal
Fuente: Elaboración propia

2.3.3. LSTM Neural Networks

En este apartado se va a proceder a explicar las Redes Neuronales de Memoria de Corto y Largo Plazo (LSTM por sus siglas en inglés). Un tipo de Redes Neuronales Convolucionales utilizadas en el campo del Deep Learning. Seguidamente, se explicará cuál ha sido el algoritmo utilizado, así como sus hiperparámetros para, finalmente, obtener una visión de los resultados de estas redes.

2.3.3.1. ANN

Las redes neuronales artificiales (ANN por sus siglas en inglés) son un subconjunto dentro del Machine Learning y, tanto su nombre como funcionamiento están basados en el cerebro humano y como las neuronas se comunican entre sí.

Estos modelos son formados por un conjunto de nodos (neuronas) que se encuentran conectadas entre sí y, a partir de esas conexiones, transmiten la información entre las neuronas desde una entrada (input) hasta generar una salida (output).

Las redes neuronales siguen un tipo de aprendizaje denominado aprendizaje automático. Éste tipo de aprendizaje implica que la propia red neuronal es capaz de aprender poco a poco de sus errores y, con el tiempo, predecir unos resultados más fieles a la realidad.

El funcionamiento es el siguiente (Figura 2.10): Se tiene una maya de neuronas (puntos donde se realizan cálculos matemáticos) organizados en capas (conjuntos de neuronas que se encuentran en el mismo momento de ejecución) conectadas entre sí. Cada neurona recibe de la capa anterior un conjunto de valores los cuales, mediante un conjunto de operaciones matemáticas, da como salida un nuevo valor que será enviado a la siguiente capa hasta llegar a la capa de salida, donde se realizaran por última vez las operaciones matemáticas adecuadas y, finalmente, se obtendrá un valor final que intentará aproximarse lo máximo posible al valor real.

Para poder aproximar el valor resultante de la red neuronal artificial al valor real se deben modificar unos valores intrínsecos que tienen cada una de las neuronas llamados pesos. Estos pesos tienen la función de potenciar o reducir la importancia de una operación matemática dentro de la neurona con el objetivo de conseguir unos valores mas cercanos a los reales. Para modificar estos valores de forma correcta se utiliza un método llamado Back Propagation que será explicado más adelante. Pero antes, se explicará que es necesario para poder utilizar una red neuronal.

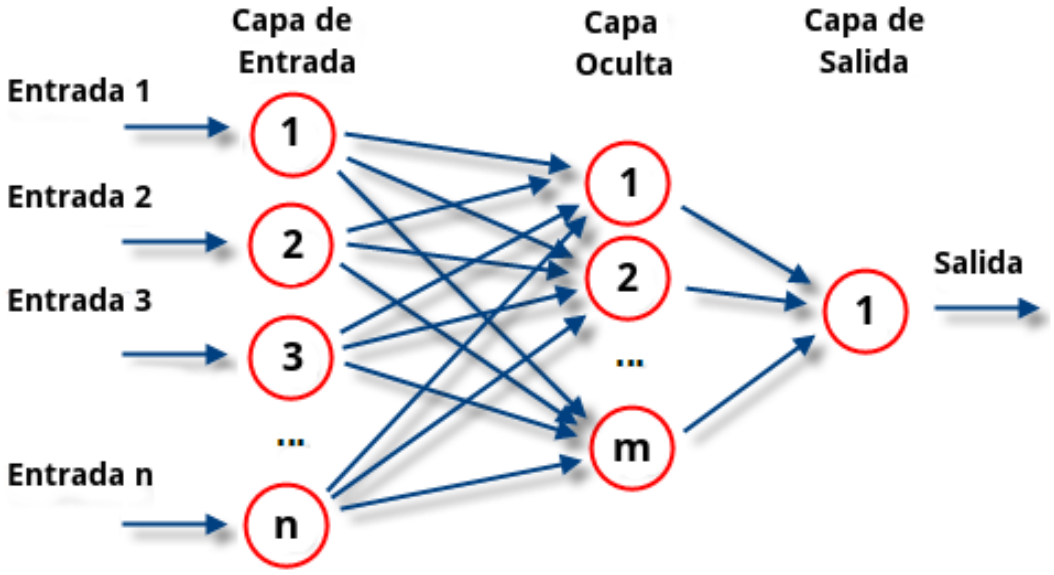

Figura 2.10: Visualización de una red neuronal artificial
Fuente: https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/

Tal y como se ha indicado anteriormente, las redes neuronales utilizan un tipo de aprendizaje llamado aprendizaje automático. Este tipo de aprendizaje implica que el propio algoritmo aprende de manera autónoma a medida que va recibiendo entrenamiento. Para entrenar una red neuronal a partir de la cuál se quieren predecir unos valores de salida a partir de unos datos de entrada, es necesario tener antes un subconjunto de estos datos de entrada y sus respectivos valores de salida. Estos datos son los que permiten entrenar a la red neuronal. En primer lugar, este subconjunto de datos de los cuales tenemos tanto los valores de entrada como de salida se dividen en 3 otros subconjuntos, los datos de Training, los datos de Test y los datos de Validation (Figura 2.11). Los datos de Training permiten, a partir del método de Back Propagation y el Descenso del Gradiente, optimizar la red neuronal. En cambio, los datos de Validation permiten observar el rendimiento de los modelos entrenados y cuál de estos es el mejor para predecir los resultados. A continuación, se elije el modelo con mejor ajuste y se prueba con los datos de Test para comprobar que el modelo funciona correctamente. Finalmente, una vez el modelo tiene el visto bueno a partir de los datos de Test, se puede utilizar la red neuronal creada para predecir los resultados de cuyos datos no tenemos información.

Por lo tanto, con lo visto hasta ahora, se sabe que una red neuronal necesita ser entrenada para mejorar su efectividad y que, este proceso de entrenamiento consiste en utilizar los datos de training repetidas veces para que, en cada ejecución, mediante el método de Back Propagation, se minimice el error existente entre los datos ofrecidos por el modelo y los datos reales. Una vez se tienen estos conceptos claros ya se puede pasar a explicar el método de Back Propagation.

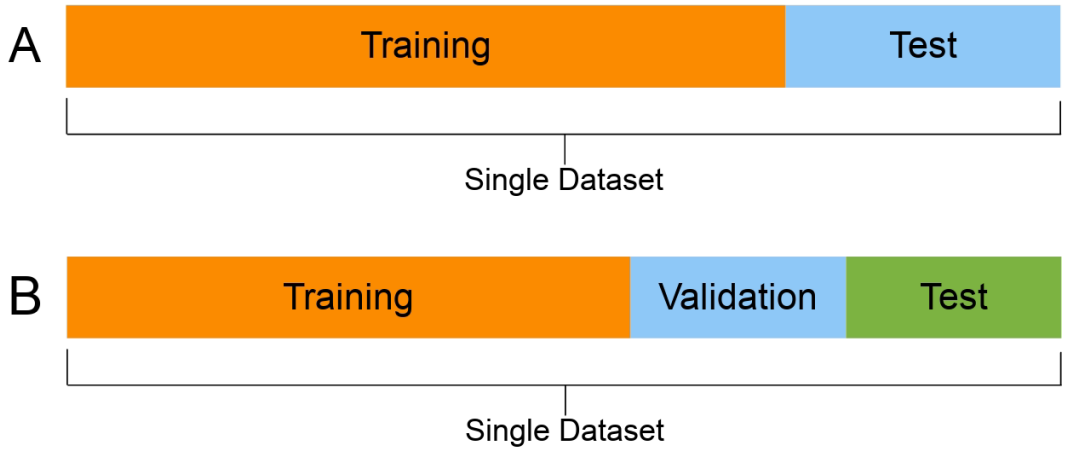

Figura 2.11: Diferentes maneras de partir un conjunto de datos. En el caso A se usa un conjunto de Training para entrenar el modelo y un conjunto de Test para comprobar la eficacia del modelo entrenado. En el caso B se una un conjunto de Training para entrenar los modelos, un conjunto de Validation para comprobar que modelos funcionan mejor y, finalmente, un conjunto de Test para comprobar la eficiencia del modelo entrenado
Fuente: https://en.wikipedia.org/wiki/Training,-validation,-and-test-sets

El método de Back Propagation es un algoritmo que se ejecutada cada vez que se realiza una iteración del conjunto de Training de la Red Neuronal. Éste consiste en calcular la diferencia que existe entre el valor de salida de la red neuronal y el real y, mediante el método del descenso del gradiente, propagar desde la última capa hasta la primera el error ofreciendo a cada neurona una modificación de sus pesos tan grande como el error que ésta provoca en el conjunto de la red. Para ello, se utiliza el método del descenso del gradiente descendiente, un método que busca el valor mínimo de una función (en este caso el error mínimo) a partir de calcular la derivada en cada punto y el valor en cuestión se desplaza en la dirección donde la función se reduce. De este modo, finalmente se llega a un punto donde la deriva de la función en un punto vale cero y, por lo tanto, se ha llegado al mínimo de la función. El método para actualizar el valor de x es el siguiente (Fórmula 2.10):

$$x_{nueva} = x_{antigua} - \alpha \frac{\delta f(x)}{\delta x} \tag{2.10}$$

Evidentemente, el ejemplo anteriormente mencionado, y que se puede ver gráficamente a continuación, es un ejemplo muy simple y que no suele darse en el mundo real. Existen varios problemas con el descenso del gradiente. Los principales son que la función tenga más de un mínimo y que, por tanto, el método del gradiente no sea capaz de encontrar el mínimo global, sino únicamente un mínimo local; o que la fórmula del método del gradiente no converja nunca debido a unos valores incorrectos de los parámetros. Afortunadamente,

estos problemas pueden ser solucionados (tal y como se verá más adelante) y, por lo tanto, podemos decir que el método del descenso del gradiente, como el que tenemos a continuación (Figura 2.12), sí permiten a las redes neuronales reducir sus errores a lo largo del tiempo y, de este modo, predecir los resultados a partir de los valores introducidos.

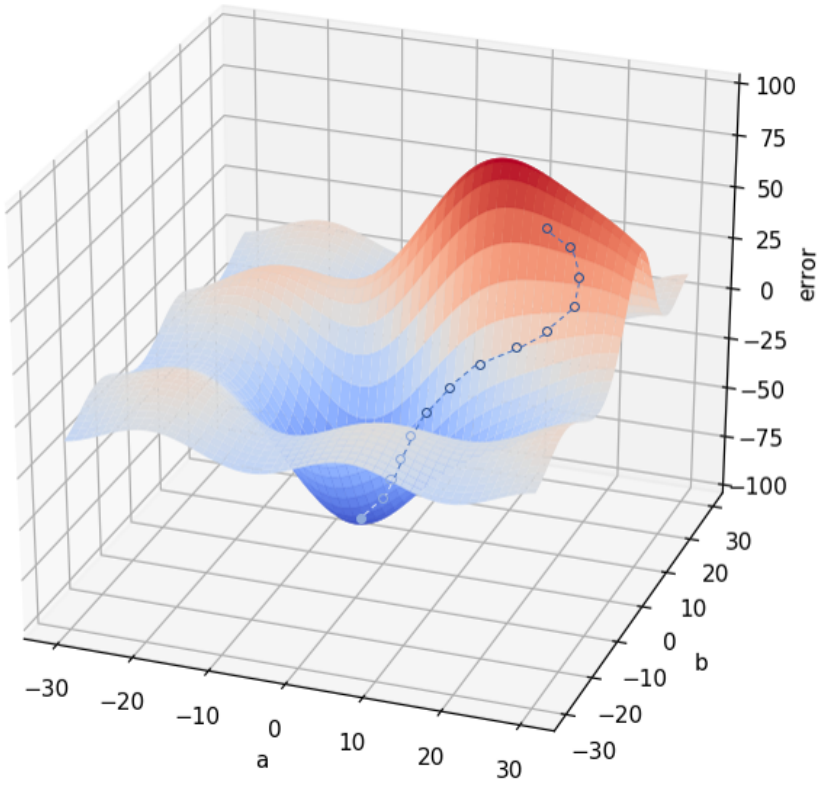

Figura 2.12: Método del descenso del gradiente
Fuente: https://interactivechaos.com/es/manual/tutorial-de-machine-learning/gradient-descent

2.3.3.2. Long Short Term Memory Networks

En el apartado anterior se explica una distribución básica de redes neuronales, pero realmente existen más tipos de ellas. En concreto, en este trabajo se utilizan unas redes llamadas redes neuronales recurrentes (RNN por sus siglas en inglés) que almacenan información del pasado durante periodos temporales a través de bucles en su estructura.

Las RNN son muy útiles para cuando los datos dependen del tiempo o son una secuencia donde el dato actual depende de los anteriores pero, debido a su estructura, pueden sufrir 2 problemas. El primer problema es llamado el problema del desvanecimiento del gradiente y surge cuando el gradiente se vuelve tan pequeño que no permite mejorar a la propia red neuronal. El segundo problema es el del gradiente explosivo, donde un pequeño error en las primeras iteraciones de la red neuronal crea un sesgo cada vez mayor debido

al hecho de que el error se propaga cada vez más.

Para evitar estos problemas, se utilizará una red de memoria de corto y largo plazo (LSTM por sus siglas en inglés), un tipo de redes recurrentes que son diseñadas para evitar las dependencias de la memoria a largo plazo.

Las redes LSTM tienen una estructura más sofisticada que las que se han visto en el punto anterior, pero a continuación se explica su funcionamiento paso a paso para poderlas entender en su perfección.

En primer lugar, es necesario destacar que las redes LSTM son unas cadenas de bloques de redes neuronales que se repiten varias veces (en nuestro caso 60 veces, una por cada momento temporal de retraso que tiene la red). Cada uno de estos bloques se encuentra conectado al anterior y al siguiente y, de esta manera, se puede transmitir la información del bloque anterior (momento temporal anterior) al bloque posterior (momento temporal posterior) (Figura 2.13).

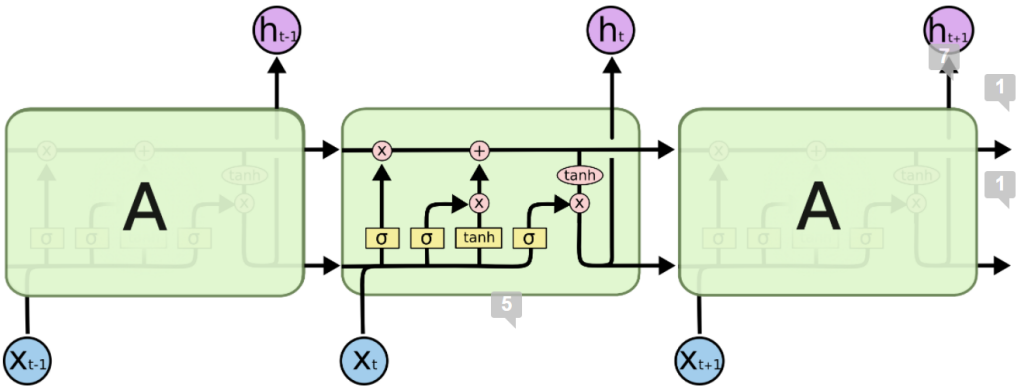

Figura 2.13: Cadena de bloques de una red neuronal
Fuente: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Una vez se ha entendido que la red LSTM esta formada por cadenas de bloques es necesario entender que se encuentra dentro de estos. Para ello, se puede observar la figura 2.13 i ver que cada bloque se encuentra formado por cuatro capas de redes neuronales (rectángulos amarillos) que se unen entre si por unos vectores de datos (líneas negras), de las cuales entran 2 valores iguales en la red neuronal y también salen 2, y en cada intersección de se encuentra un operador matemático (círculos rojos).

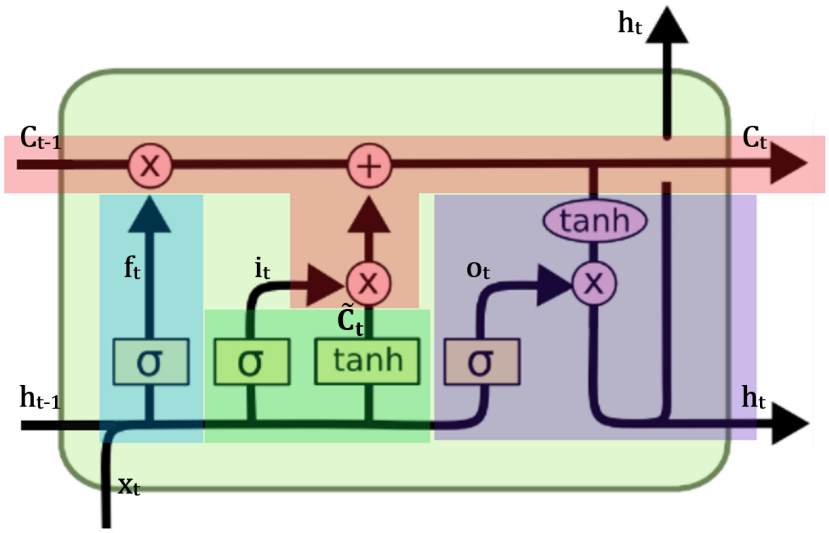

Figura 2.14: Celda de una red LSTM
Fuente: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ y modificada por cuenta propia

Pues bien, la idea fundamental de las redes LSTM se encuentra en la memoria de largo plazo (parte roja de la figura), un vector de datos que se modifica levemente a medida que van sucediendo las iteraciones y que guarda la información general del conjunto de datos. Estos datos se modifican a partir de las operaciones de suma y multiplicación que también se pueden apreciar en la parte roja de la figura, las cuales vienen definidas por las partes azul y verde de la figura. De este modo, la manera como se actualiza la memoria de largo plazo es la siguiente (Fórmula 2.11):

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.11}$$

Para modificar la memoria anteriormente mencionada, se debe borrar la información que no sea útil en la memoria a largo plazo. Para ello, se utiliza una función sigmoide (azul), la cual devuelve un valor entre 0 y 1 para que, cuanto más pequeño sea el valor, más información se olvide de la memoria ya mencionada (Fórmula 2.12).

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \tag{2.12}$$

Seguidamente se debe añadir la nueva información obtenida tanto de la memoria a corto plazo como como en los nuevos datos obtenidos. Este proceso tiene dos partes. En primer lugar, una función sigmoide que elige que valores deben ser actualizados (Fórmula 2.13) y, en segundo lugar, una función tangente hiperbólica que crea el nuevo vector de candidatos para incorporarse a la memoria de largo plazo (Fórmula 2.14).

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \tag{2.13}$$

$$\tilde{C}_t = tanh(W_C * [h_{t-1}, x_t] + b_C) \tag{2.14}$$

Finalmente, se debe crear la salida de la celda LSTM. Para ello se junta la memoria de largo plazo obtenida en los pasos anteriores, la cual es filtrada por una función tangente hiperbólica, junto a la memoria de corto plazo y los valores de entrada, los cuales son modificados por una función sigmoide (Fórmula 2.15), para así crear el output final y que, a su vez, será tambien la memoria de corto plazo del siguiente bloque de la red (Fórmula 2.16).

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \tag{2.15}$$

$$h_t = o_t * tanh(C_t) \tag{2.16}$$

2.3.3.3. Creación de las redes LSTM

En este apartado se va a mostrar paso a paso el proceso de creación de la red neuronal LSTM, así como los resultados de la misma.

La creación de la red LSTM es un proceso difícil por dos motivos principales. El más evidente es por la propia complejidad de la red a la hora de manipular y elegir sus hiperparámetros, pero el segundo, aunque menos evidente, es igual de crucial y se trata de la depuración de los datos para que la red los pueda entender. Por este motivo, antes de explicar como se creó la red, se explicará como se transformaron los datos existentes.

En este trabajo se crearon un total de 80 redes LSTM, una para cada empresa de la cuál se tenia cotización. De esta manera, cada red debía recibir los datos de la empresa que le correspondía, tanto del momento temporal en el que se encontraba como el de los sesenta periodos anteriores.

Por este motivo, se debe crear una tabla para cada valor donde cada fila indica una iteración de la red neuronal y cada columna indica cada uno de los periodos temporales que la red neuronal utiliza. De esta manera, quedan un conjunto de 80 tablas de 135099 filas y 60 columnas (Tabla 2.2). Tal y como se puede apreciar, el número de filas de esta

tabla (135099) es ligeramente inferior al número de filas del conjunto de datos original (190844), esto es debido a que la red LSTM únicamente utiliza un valor de tiempo si puede conseguir los 60 valores de tiempo anteriores sin cambiar de día, ya que esto supondría que existieran saltos temporales y crearía errores. Por este motivo, y al igual que en el análisis técnico, no se utiliza la primera hora del día para realizar los cálculos.

	0	1	2	3	4
0	49.466802	49.349326	49.443588	49.386918	49.444058
1	49.349326	49.443588	49.386918	49.444058	49.504394
2	49.443588	49.386918	49.444058	49.504394	49.443306
3	49.386918	49.444058	49.504394	49.443306	49.471501
4	49.444058	49.504394	49.443306	49.471501	49.551384

Cuadro 2.2: Subconjunto de la tabla de CSCO para la red LSTM

Para facilitar el trabajo a la red neuronal, la tabla creada se debe modificar de tal manera de que el valor inicial siempre sea el mismo ya que, de otra manera, la red neuronal podría no entender ciertas relaciones entre los cambios de día, ya que los precios entre un día y otro podrían variar en gran medida. Por este motivo, se modifica la tabla anterior y se le ofrece a la red neuronal una tabla donde para cada iteración, el periodo de tiempo t-60 es el valor 0 y los demás valores son las variaciones de precios que existen entre el periodo de tiempo t-i y t-60 (Tabla 2.3).

0	1	2	3	4
0	-0.002375	-0.000469	-0.001615	-0.000460
1	0.001910	0.000762	0.001920	0.003142
2	-0.001146	0.000010	0.001230	-0.000006
3	0.001157	0.002379	0.001142	0.001713
4	0.001220	-0.000015	0.000555	0.002171

Cuadro 2.3: Subconjunto de la tabla modificada de CSCO para la red LSTM. Cada fila es un vector que se inserta como capa de entrada en la red neuronal. Estos valores son las diferencias de cotización minuto a minuto de la última hora respecto al valor de cotización de 60 minutos antes del valor que se quiere predecir. La salida sera un único valor que intentará predecir la diferencia de cotización entre un momento temporal y el minuto siguiente.

Una vez los datos ya han sido transformados, es momento de crear la red neuronal. En este trabajo se utilizó una red neuronal LSTM Vanilla, lo que implica que no tiene modificaciones respecto la explicada en el apartado anterior y que está definida por una capa de entrada de tamaño 60 (como los periodos temporales utilizados en cada iteración). Seguidamente se encuentran 2 capas ocultas de tipo LSTM con una función de activación ReLU (una función de activación lineal y siempre positiva) y un total de 60 bloques cada una para finalmente obtener una única neurona de salida con la diferencia de cotización existente entre un periodo de tiempo y el inmediatamente posterior (Figura 2.15).

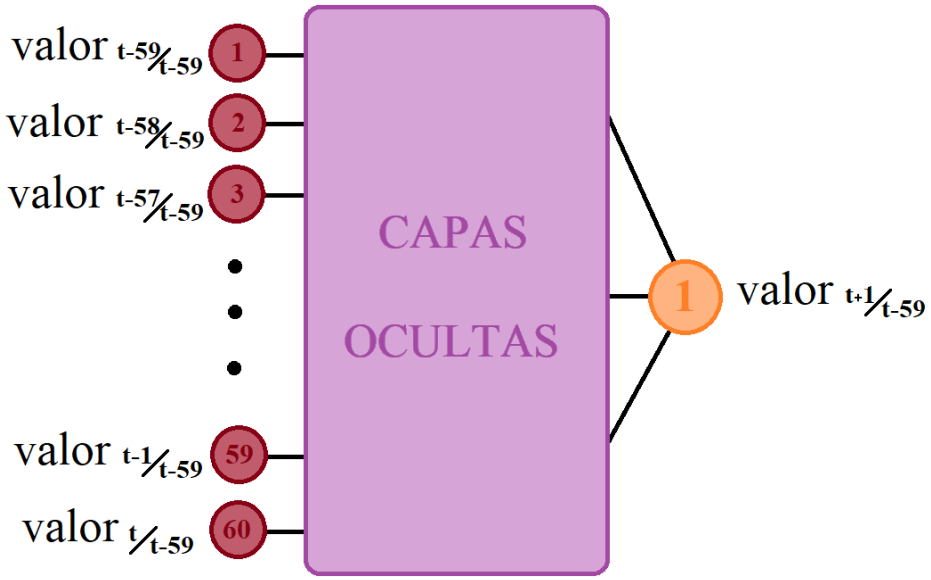

Figura 2.15: Gráfico del conjunto de neuronas de entrada y salida de la red neuronal LSTM.
Fuente: Elaboración propia

A parte de la estructura de la red neuronal, también es importante hablar de sus hiperparámetros. En primer lugar, el optimizador utilizado ha sido el optimizador Adam (Adaptive moment estimation)[4] [5], donde en lugar de utilizar el método del gradiente con un valor de descenso (ratio de entrenamiento) fijado, el optimizador lo modifica para poder realizar la fase de entrenamiento en menor tiempo y de más eficientemente. En segundo lugar, la función de pérdida utilizada (aquella que indica cuanto de alejados nos encontramos respecto el valor real) es la función de mínimos cuadrados ordinarios. El validation_split (la parte de los datos de training que permite validar el funcionamiento de la red) es del 25 % del total de los datos utilizados. Finalmente, el número de epochs (el número de veces que la red neuronal utilizará el conjunto de datos proporcionado para realizar sus conclusiones) es de 50 y el tamaño del batch (el número de muestras en los que se debe trabajar antes de realizar las predicciones para cambiar los parámetros de la red) es de 64 ya que así se puede entrenar a la red neuronal suficientemente tiempo para que los valores se aproximen a los reales, pero el tiempo de ejecución no supere 1 semana.

Una vez se ha finalizado la explicación de la creación de la red neuronal LSTM se pueden apreciar los resultados obtenidos. Para ello, se ha vuelto a elegir la empresa de referencia de todo el trabajo (Cisco Systems).

[4]https://keras.io/api/optimizers/adam/
[5]https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

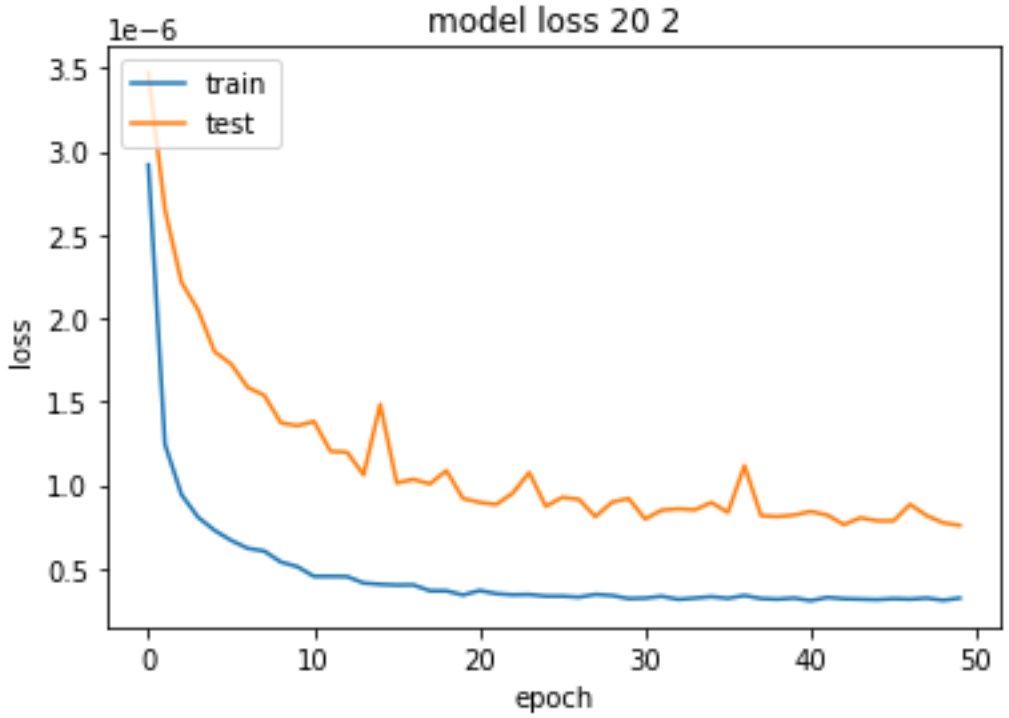

Figura 2.16: Gráfico de la pérdida por cada epoch en los conjuntos de train y de test
Fuente: Elaboración propia

En el primer gráfico (Figura 2.16) se puede apreciar las perdidas obtenidas tanto en la fase de training como en la de test. Como era esperable, el gráfico train tiene una pérdida menor que el test. Estas pérdidas, por su parte, son de aproximadamente de 1e-6, un valor pequeño pero que, si tenemos en cuenta las diferencias de cotización existentes entre un periodo temporal y el anterior, son diferencias apreciables y, por lo tanto, aunque la red LSTM permite encontrar las tendencias de las cotizaciones, sus aproximaciones quedan bastante distantes de ser ideales, tal y como se puede apreciar el la figura 2.17)

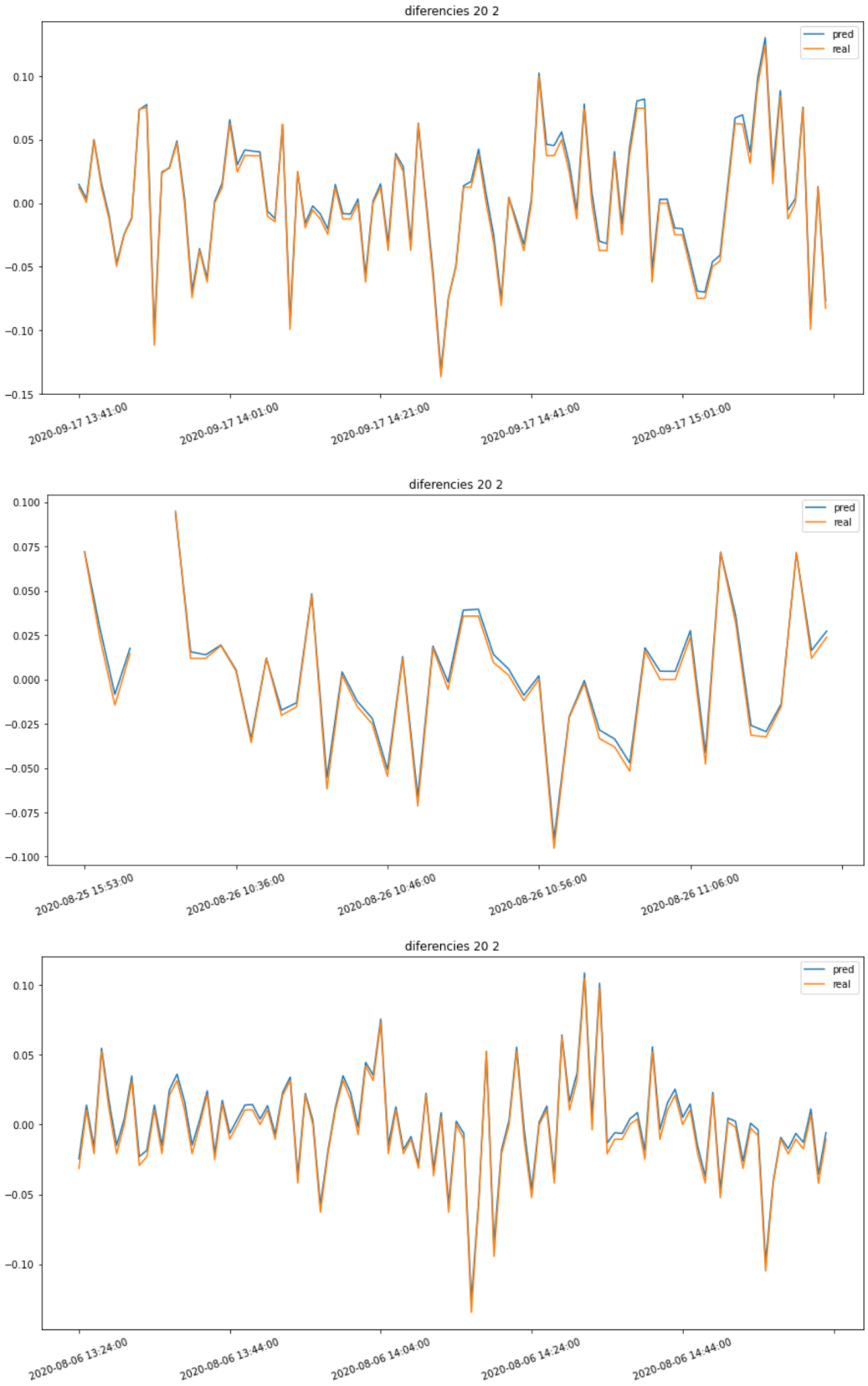

Figura 2.17: Gráfico de los valores reales y aproximados de la diferencia de cotización entre 2 periodos consecutivos en 3 periodos de tiempo diferentes
Fuente: Elaboración propia

2.3.4. Feedforward Neural Network

En este apartado se va a explicar el funcionamiento de la FFNN creada y, seguidamente, se mostraran los resultados de la misma, pero no se explicará cuál es la teoría de la red porque esto ya ha sido hecho en los apartados anteriores, donde para llegar a entender las redes LSTM se explica previamente el funcionamiento de las FFNN.

Tal y como se mostró en el apartado de creación de redes LSTM, se va a proceder a mostrar primero la transformación de los datos utilizados para que los pueda utilizar la red neuronal y, a continuación, se indicaran cuales son los hyperparámetros utilizados y los resultados obtenidos con la red.

Para poder ofrecer a la red neuronal los datos necesarios para que pueda funcionar es necesario que primero se tengan descargados y activos tanto los diferentes indicadores de análisis técnico como las redes neuronales LSTM realizados con anterioridad.

Una vez se tienen estos datos, y debido a que las redes neuronales FFNN también funcionan de forma independiente para cada una de las 80 empresas, se deben crear un total de 80 tablas con un total de 18 columnas (15 para los distintos indicadores de cada empresa, 1 para los resultados de la red LSTM, 1 para el volumen cotizado en cada momento temporal y un último para el valor real de cotización) y 58432 filas (desde el 5 de agosto de 2020 hasta 23 de abril de 2021) En este conjunto de datos encontramos el conjunto de train (datos de 2020), donde encontramos también el conjunto de validation (25 % final de los datos de train) y el conjunto de test (datos de 2021) con los cuales haremos la evaluación del modelo final. Es necesario tener en cuenta que, mientras que la red LSTM ya realiza una predicción y, por lo tanto, el momento temporal de la misma es idéntico al momento del valor que se quiere predecir (el valor real), todos los indicadores (incluida la volatilidad), deben ser utilizadas con un momento temporal de retraso, ya que de otra manera se utilizarían datos del presente para predecir el mismo presente (hecho que no tiene sentido porque el objetivo de este trabajo es predecir el futuro). Finalmente, se ha observado que la volatilidad añadía más error a la red neuronal del que se obtenía sin ella. Por este motivo, finalmente todos los cálculos se han realizado excluyendo esta variable de los cálculos(Figura 2.18).

De este modo, se obtienen un total de 80 tablas que tienen una forma parecida a la tabla 2.4, de nuevo una subtabla del conjunto de datos de la tabla de Cisco Systems, que será subministrada a la red neuronal.

Una vez los datos ya han sido transformados, se crean un conjunto de 6 redes neuronales con diferentes características para poder elegir, a partir del conjunto de validation,

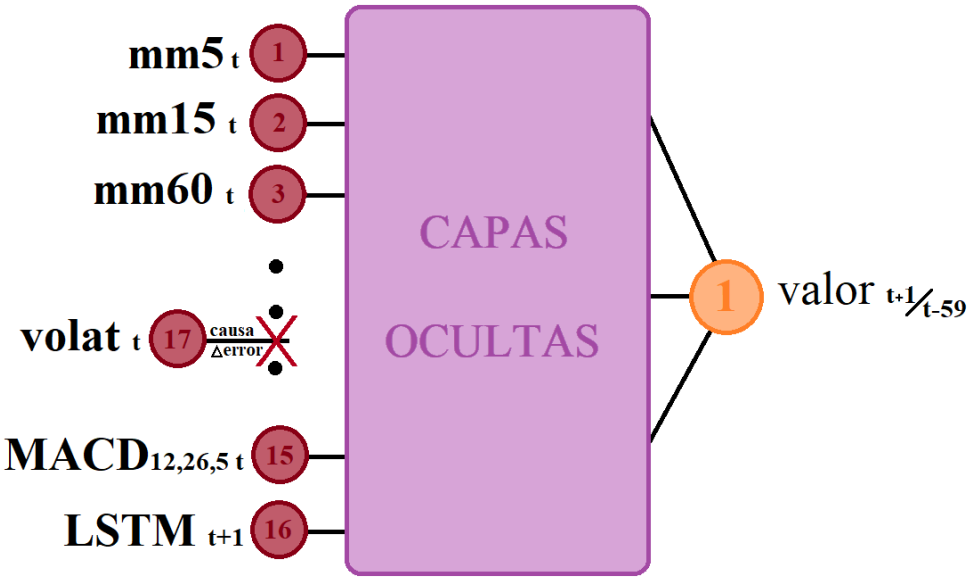

Figura 2.18: Gráfico del conjunto de neuronas de entrada y salida de la red FFNN. Fuente: Elaboración propia

0	1	2	16	17	18
1	-0.020971	0.062478	-0.200899	22327	-0.199437
2	0.025155	0.107191	-0.118030	35151	-0.105795
3	-0.022313	0.035270	0.014566	13964	0.035302
4	-0.086616	-0.049122	-0.057155	19172	-0.058817
5	-0.061862	-0.046043	0.042241	14442	0.047081

Cuadro 2.4: Subconjunto de la tabla de CSCO para la red FFNN. En este caso, la capa de entrada de la red neuronal será un vector de 16 valores. Los primeros 15 hacen referencia a los resultados obtenido en cada periodo temporal por los indicadores de análisis técnico creados, y el último hace referencia al valor de salida de la red neuronal LSTM. Con estos valores, se intentará generar un único valor como salida de la red neuronal FFNN que intentará predecir la diferencia de cotización existente entre un periodo de tiempo y el inmediatamente posterior.

aquella que menor error produzca. Estas 6 redes neuronales tienen entre 1 y 4 capas ocultas y, en el caso de que se observara que cuantas más capas mejor es el resultado, se añadirían mas redes con un número mayor de capas ocultas. Además, se puede observar que el número de neuronas por capa nunca supera la suma del número de capas de entrada más las de salida (16+1=17) de este modo, se sigue la famosa regla de oro de las redes neuronales[6] [7] [8]. A continuación, se pueden apreciar los hyperparámetros y resultados de cada una de ellas.

[6]https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/

[7]https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw

[8]https://towardsdatascience.com/17-rules-of-thumb-for-building-a-neural-network-93356f9930af

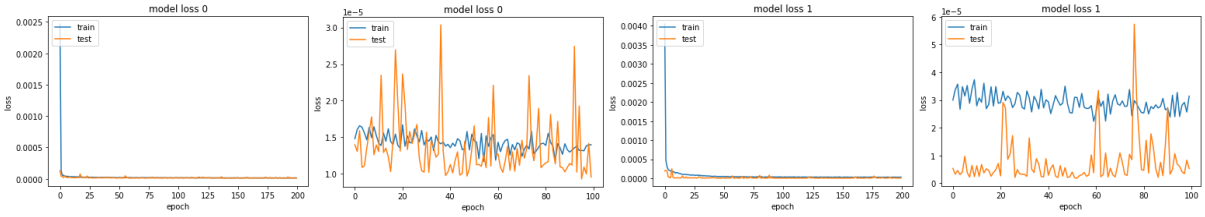

Figura 2.19: Gráfico de las funciones de error en 2 de las cotizaciones utilizadas en el trabajo con la red FFNN número 1. En el primer gráfico de cada cotización se observa el total de iteraciones, mientras que en el segundo se muestran únicamente las últimas 400 para tener un mayor detalle de las tendencias.
Fuente: Elaboración propia

La primera red neuronal (Figura 2.19) se encuentra formada por una única capa oculta de 16 neuronas. La función de pérdida, al igual que en las redes LSTM es la de mínimos cuadrados ordinarios, el optimizador utilizado vuelve ser el optimizador Adam y la función de activación es la función ReLU. Tal y como se ha mencionado anteriormente, el conjunto de validation consta del 25 por ciento de los datos. El número de iteraciones (epochs) será de 500 y el tamaño de batch será de 8. Cabe destacar que la función de pérdida, el optimizador, la función de activación, el conjunto de validation, el número de epoch y el tamaño del bach se mantendrán constantes en las siguientes redes creadas y, por lo tanto, no se hará énfasis en ellos de ahora en adelante. Se puede observar que las pérdidas en el conjunto de train se estabilizan entre los valores de 1e-6 y 1.5e-5, mientras que las pérdidas en el conjunto de validation se estabilizan entre 1e-5 y 4e-5.

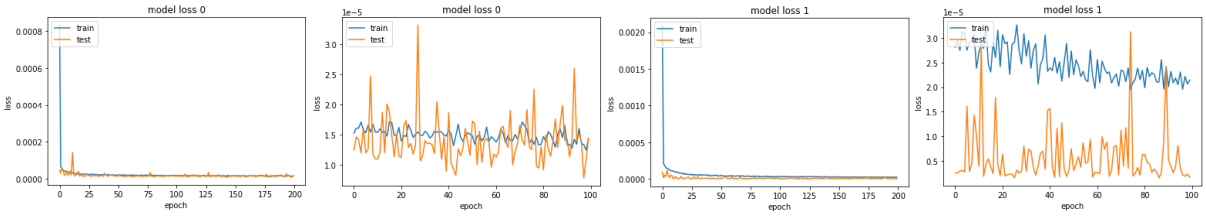

Figura 2.20: Gráfico de las funciones de error en 2 de las cotizaciones utilizadas en el trabajo con la red FFNN número 2. En el primer gráfico de cada cotización se observa el total de iteraciones, mientras que en el segundo se muestran únicamente las últimas 400 para tener un mayor detalle de las tendencias.
Fuente: Elaboración propia

La segunda red neuronal (Figura 2.20) se encuentra formada por un conjunto de 2 capas ocultas de 16 y 10 neuronas respectivamente. Se puede observar que las pérdidas en el conjunto de train se estabilizan entre los valores de 1e-6 y 2e-5, mientras que las pérdidas en el conjunto de validation se estabilizan entre 1e-5 y 3e-5. Por lo tanto, no se observa una mejora respecto la red neuronal anterior.

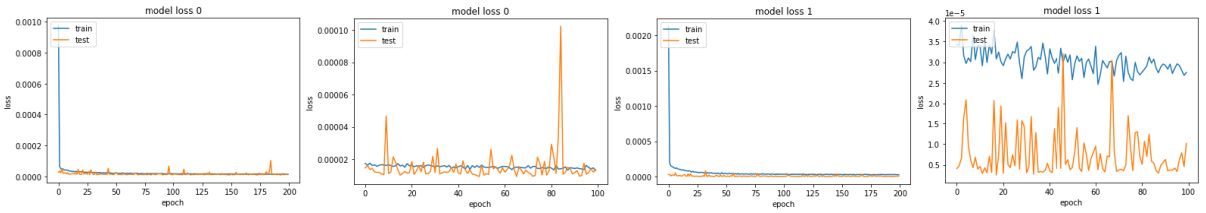

Figura 2.21: Gráfico de las funciones de error en 2 de las cotizaciones utilizadas en el trabajo con la red FFNN número 3. En el primer gráfico de cada cotización se observa el total de iteraciones, mientras que en el segundo se muestran únicamente las últimas 400 para tener un mayor detalle de las tendencias.
Fuente: Elaboración propia

La tercera red neuronal (Figura 2.21) también se encuentra formada por un conjunto de 2 capas ocultas pero, en este caso, las capas se encuentran formadas por 10 y 5 neuronas respectivamente. Se puede observar que las pérdidas en el conjunto de train se estabilizan entre los valores de 1e-6 y 2e-5, mientras que las pérdidas en el conjunto de validation se estabilizan entre 1e-5 y 4e-5. Por lo tanto, se puede observar un pequeño aumento de la función de error, pero nada que haga sospechar que el número reducido de neuronas afecte en gran medida a los resultados de la red.

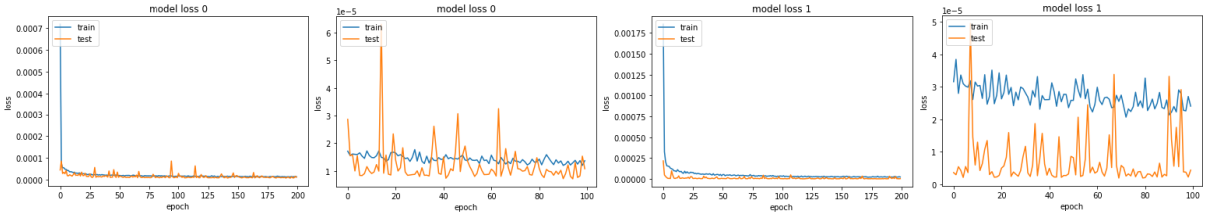

Figura 2.22: Gráfico de las funciones de error en 2 de las cotizaciones utilizadas en el trabajo con la red FFNN número 4. En el primer gráfico de cada cotización se observa el total de iteraciones, mientras que en el segundo se muestran únicamente las últimas 400 para tener un mayor detalle de las tendencias.
Fuente: Elaboración propia

La cuarta red neuronal (Figura 2.22) se encuentra formada por un conjunto de 3 capas ocultas de 16, 10 y 5 neuronas respectivamente. Se puede observar que las pérdidas en el conjunto de train se estabilizan entre los valores de 1e-6 y 2e-5, mientras que las pérdidas en el conjunto de validation se estabilizan entre 1e-5 y 3.5e-5. Por lo tanto, no se observa una mejora respecto la red neuronal anterior.

La quinta red neuronal (Figura 2.23) se encuentra formada por un conjunto de 3 capas ocultas de 16, 15 y 10 neuronas respectivamente. Se puede observar que las pérdidas en el conjunto de train se estabilizan entre los valores de 1e-6 y 2e-5, mientras que las pérdidas en el conjunto de validation se estabilizan entre 1e-5 y 3e-5. Por lo tanto, no se observa una mejora respecto la red neuronal anterior.

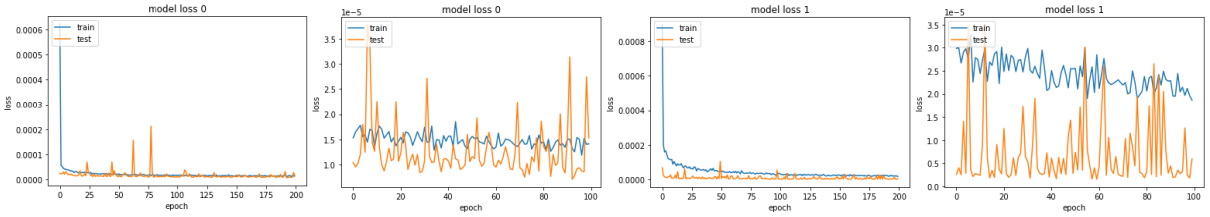

Figura 2.23: Gráfico de las funciones de error en 2 de las cotizaciones utilizadas en el trabajo con la red FFNN número 5. En el primer gráfico de cada cotización se observa el total de iteraciones, mientras que en el segundo se muestran únicamente las últimas 400 para tener un mayor detalle de las tendencias.
Fuente: Elaboración propia

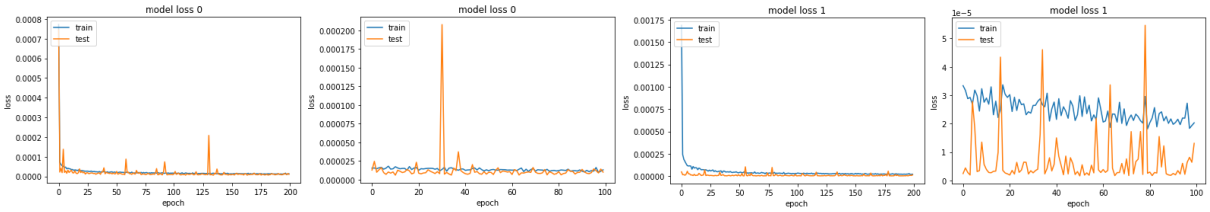

Figura 2.24: Gráfico de las funciones de error en 2 de las cotizaciones utilizadas en el trabajo con la red FFNN número 6. En el primer gráfico de cada cotización se observa el total de iteraciones, mientras que en el segundo se muestran únicamente las últimas 400 para tener un mayor detalle de las tendencias.
Fuente: Elaboración propia

Finalmente llegamos a la sexta red neuronal (Figura 2.24), la cual se encuentra formada por un conjunto de 4 capas ocultas de 16, 15, 10 y 5 neuronas respectivamente. Se puede observar que las pérdidas en el conjunto de train se estabilizan entre los valores de 1e-6 y 3e-5, mientras que las pérdidas en el conjunto de validation se estabilizan entre 1e-5 y 3.5e-5. Por lo tanto, no se observa una mejora respecto a las redes neuronales anteriores.

Una vez se han finalizado las diferentes redes neuronales se observa que no se aprecia una mejora clara ni al modificar el número de capas ni el número de neuronas. Este hecho confirma lo mencionado al principio de este apartado, donde se indicaba que una red neuronal de tipo FFNN puede realizar las tareas con una única capa oculta y con un número de neuronas comprendido entre 1 y el número de neuronas de entrada. Por este motivo, no es necesario continuar realizando más pruebas con redes neuronales ya que, por mucho más compleja que se haga la red, los datos no permiten que se realice un mayor ajuste del que se está obteniendo con redes más pequeñas.

Finalmente, se ha decidido optar por la red neuronal número 2 para la realización del modelo definitivo. Esta red neuronal constará de un total de 2 capas ocultas de 16 y 10 neuronas cada una. Los hiperparámetros se mantienen tal y como se había indicado en la red neuronal número 1, a excepción del conjunto de validation, que ahora se une al de

training para tener más datos para poder realizar las predicciones finales.

Tal y como se ha realizado en todos los apartados de esta segunda parte del trabajo, a continuación se procede a mostrar los resultados de la red neuronal con la empresa Cisco Systems con el fin de poder comparar los resultados entre los modelos creados. Estos resultados son los siguientes:

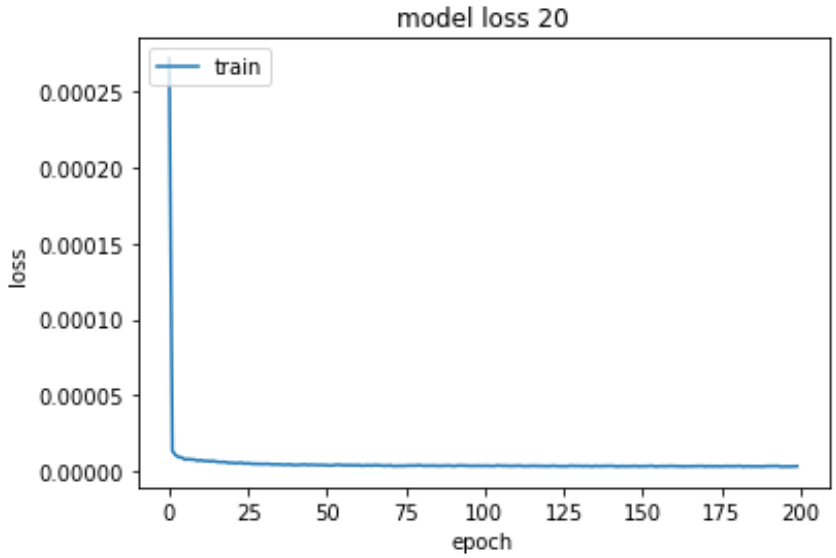

Figura 2.25: Gráfico de la función de pérdidas de Cisco Systems con el modelo de red neuronal FFNN escogido a partir de los 6 modelos propuestos anteriormente. En gráfico se observa el conjunto de 200 iteraciones, las cuales convergen en un error de alrededor de 5e-6.
Fuente: Elaboración propia

En el gráfico anterior (Figura 2.25) se puede observar como durante las 100 primeras iteraciones se observa un descenso en la función de pérdidas pero, a partir de la iteración 100 y hasta la 500 se observa como la función de pérdidas se estanca lentamente hasta llegar a un error cuadrático medio de entre 3e-5 y 4e-5.

Estos valores de la función de pérdidas son similares que los vistos en el modelo LSTM, lo que puede indicar que el modelo FFNN puede predecir los valores mejor (o igual) que el modelo LSTM.

Además, en el gráfico que se puede observar en la página siguiente (Figura 2.26), se aprecia que las predicciones parecen ser más precisas (o al menos igual de precisas que con la red LSTM). Al igual que se indicó en las redes LSTM, este modelo es capaz de predecir las tendencias del periodo temporal siguiente con un acierto ciertamente elevado. Por este motivo, se va a proceder a crear el algoritmo para invertir en una acción o en otra a partir de este modelo.

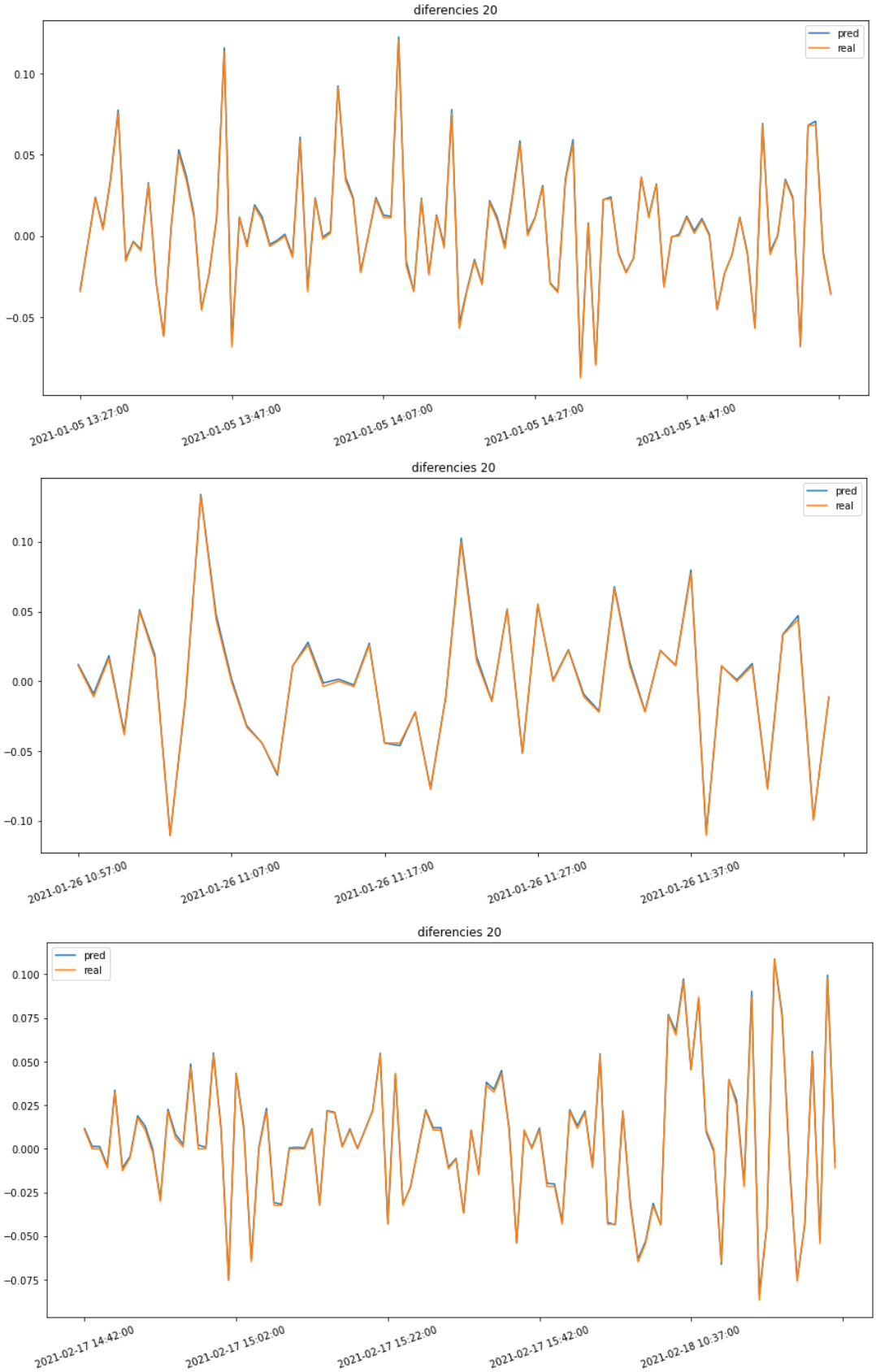

Figura 2.26: Gráfico de los valores reales y aproximados de la diferencia de cotización entre 2 periodos consecutivos en 3 periodos de tiempo diferentes.
Fuente: Elaboración propia

2.4. Resultados

Como se ha podido apreciar en los apartados anteriores, las predicciones de la red neuronal LSTM han tenido un menor error que las generadas por la red neuronal FFNN pero debido a que este error es muy similar y a que el gráfico de la red FFNN parece mostrar que las predicciones con esta red son mejores que con la red LSTM se procede a realizar el algoritmo de trading con la red neuronal FFNN. Aún así, es necesario destacar que no se observa una clara mejoría en la red FFNN respecto la red LSTM, hecho que puede indicar que los indicadores de análisis técnico no permiten predecir el futuro con gran precisión (al menos en el trading de tan baja frecuencia utilizado en este trabajo), pero sería interesante seguir investigando sobre su utilidad en futuros trabajos ya que, podría ser que su utilidad no fuera tan útil como algunos traders defienden.

Para realizar el algoritmo de trading automático es necesario, en primer lugar, explicar cual va a ser el ámbito contra el cuál se va a comparar la eficacia del algoritmo. Para ello, se indica que el objetivo es, tal y como se había mencionado con anterioridad, mejorar los resultados que se pueden obtener si se invierte de manera equivalente una cantidad de dinero en cada una de las 80 empresas seleccionadas en el periodo temporal en el cuál nuestro algoritmo se encuentra operativo. Para ello, se procede a realizar una multiplicación de las rentabilidades reales obtenidas (sean positivas como negativas) minuto a minuto por cada una de las 80 empresas utilizadas en este trabajo. Finalmente, este valor se divide entre 80 y, de esta manera, se obtiene la rentabilidad media obtenida durante el primer cuatrimestre de 2021 en los periodos de tiempo donde el algoritmo creado en este trabajo opera [9]. El beneficio obtenido con este método respecto a 1 unidad monetaria es de 1,03389, lo que implica que por cada 1000 dólares invertidos el 4 de enero de 2021 durante el periodo temporal al cuál se hace referencia en este trabajo hasta el 23 de abril de 2021 se habrían obtenido un total de 1033,89 dólares. A continuación se puede observar la fórmula con la cuál se calcula este valor (Fórmula 2.17).

$$\pi_{real} = \frac{\prod_{i=1}^{\#Periodos} \sum_{j=1}^{\#Empresas} diferencia_{ij}}{\#Empresas} \tag{2.17}$$

Un segundo método contra el cuál comparar el beneficio del algoritmo creado es el de comprar y mantener, donde en el momento en el que empieza el caso de estudio se compran acciones de las 80 diferentes empresas de manera que el peso de cada empresa

[9] El método de compra-venta minuto a minuto consiste básicamente en que, en cada minuto temporal, se compra un conjunto de acciones de cada empresa de manera que el peso que tenga cada empresa en la cartera sea de 1 entre el total de empresas de la cartera y, en el minuto posterior, se vendan todas las acciones y, con los beneficios obtenidos, se vuelva a repetir el proceso indefinidamente o hasta que el usuario quiera.

en la cartera sea de 1 entre el número de empresas. Esta cartera no se modifica durante todo el periodo de estudio y, en el último momento temporal, se venden la totalidad de las acciones y se mira cuál ha sido la rentabilidad generada por el conjunto de empresas en este periodo temporal (Fórmula 2.18). Pues bien, de esta manera, se ha obtenido un beneficio de 1.08754 respecto a una unidad monetaria, lo que implica que por cada 1000 dólares invertidos el 4 de enero de 2021 y hasta el 23 de abril de 2021 se habrían obtenido un total de 1087,54 dólares. Un beneficio superior que con el método anterior debido a que la tendencia del mercado es creciente y, al operar durante más horas, permite obtener unos beneficios mayores (aunque si la tendencia del mercado fuera decreciente las pérdidas también lo serían).

$$\pi_{mantener} = \frac{\sum_{i=1}^{\#Empresas} cot.inicial_i/cot.final_i}{\#Empresas} \tag{2.18}$$

Finalmente, se realiza el algoritmo para decidir, minuto a minuto, en que empresas se debe invertir y en cuales no. Para ello, se elige en cada periodo de tiempo el conjunto de las 10 empresas con un mayor beneficio esperado y se descartan aquellas que tienen un beneficio esperado menor o igual a 0. De las restantes, se suman sus beneficios reales en el momento temporal elegido y se divide el resultado entre el total de empresas resultantes. Finalmente, se hace un sumatorio de este proceso para cada momento temporal para obtener de esta manera el beneficio obtenido por el algoritmo creado en el periodo de estudio de este trabajo [10]. El beneficio obtenido con este método respecto a 1 unidad monetaria es de 1815829821510,745 [11]. Este valor, aunque ciertamente es irreal debido a que se deben computar muchos costes que en este trabajo no se tienen en cuenta (como las comisiones por ejemplo) tiene sentido y, de hecho, es el resultado esperado al estar multiplicando, durante un número grande de iteraciones, valores mayores a 1. A continuación se puede observar la fórmula con la cuál se calcula este valor(Fórmula 2.19).

$$\pi_{algoritmo} = \prod_{i=1}^{\#Periodos} \frac{\sum_{j=1}^{\#Empresas_{elegidas}} diferencia_{ij}}{\#Empresas_{elegidas}} \tag{2.19}$$

[10] El algoritmo creado en este trabajo funciona igual que el método de compra-venta minuto a minuto, pero con la peculiaridad que en lugar de comprar acciones de todas las empresas indiscriminadamente, se elige primeramente en qué empresas se va a invertir en ese momento temporal mediante la red FFNN.

[11] Este hecho se debe a que al reinvertir un gran número de veces valores mayores a 1 (aunque estos sean muy pequeños) se sigue la fórmula $(1+n)^k$, la cuál tiende al infinito cuando k es grande. Evidentemente, en un caso real existirían problemas que limitarían el hecho de poder seguir aumentando los beneficios (problemas de liquidez, tener que comprar las acciones a un precio más elevado que el teórico, tener que vender las acciones a un precio más bajo que el teórico o problemas con las comisiones entre otros). Aún así, el lector debe quedarse con la idea de que en cada momento temporal se están obteniendo beneficios según el modelo y, por lo tanto, se podría crear un algoritmo en el mundo real que siguiera los parámetros mostrados en este trabajo y que, a su vez, produjera beneficios, aunque estos fueran menores que los mostrados en este trabajo. Pero esto, debe ser analizado en próximos trabajos.

Tal y como se ha observado en el apartado anterior, mediante la compra-venta, minuto a minuto, de cada una de las diferentes acciones de las empresas se genera un beneficio del 3,3894 %, a partir del método de comprar y mantener se genera un beneficio del 108,754 %, mientras que gracias al algoritmo creado se genera un beneficio de más del 1000000 %. De este modo, podemos asegurar que el algoritmo creado en este trabajo permite generar beneficios al usuario que use el mismo aunque, claro está, éste deberá tener en cuenta las diferentes comisiones que se le puedan aplicar, hecho que podría provocar que los resultados fueran pérdidas si se aplica algún tipo de comisión fija al invertir o el hecho de que no existiera suficiente liquidez cuando el valor que se quiere invertir supera un volumen determinado que depende del periodo temporal o la empresa en la que se quiere invertir, entre otros.

Conclusiones

Este trabajo de doble grado y basado en el trading de alta frecuencia ha sido compuesto por dos partes diferenciadas. En primer lugar, se ha realizado un análisis teórico del trading de alta frecuencia (HFT) y, en segundo lugar, se ha creado un sistema de trading automático basado en los conceptos del mismo.

Respecto a la parte teórica se ha llegado a las conclusiones de que los parámetros que definen al HFT son su alta velocidad de operatividad y el trading intradía, los cuales se puede dar gracias a la co-location, la baja latencia y la alta capacidad de computación que tienen los ordenadores que realizan este tipo de trading. También se han podido observar las diferentes estrategias que se utilizan en este tipo de trading y cuáles son los efectos que estas causan en la liquidez, la volatilidad y la eficiencia en los mercados. Finalmente, se ha explicado cómo se encuentra la regulación de los mercados de los diferentes países para promocionar o frenar el uso de estas tecnologías según lo maduros que son los mismos. Pero si algo ha quedado claro en este trabajo es que el HFT no es nocivo para los mercados como algunos participantes de los mercados creen. De hecho, una buena regulación puede permitir que el HFT obtenga beneficios y que, además, ajuste de manera rápida y eficaz el mercado, ofreciendo liquidez para que otros agentes puedan realizar sus operaciones.

Respecto a la parte empírica, se ha realizado un algoritmo para elegir, minuto a minuto, en que conjunto de valores del NASDAQ-100 se debe invertir y en cuales no. Para ello se han utilizado indicadores de análisis técnico y 2 tipos de redes neuronales (en primer lugar, una red LSTM para realizar una primera predicción del beneficio que puede generar una empresa en cada minuto y, en segundo lugar, una red FFNN para unir los resultados de la red LSTM con los indicadores técnicos). Los resultados en esta sección han sido mejores de los esperados gracias a la alta capacidad de predicción de las redes LSTM en periodos de tiempo reducidos. Sin embargo, los indicadores técnicos no han mostrado ser una herramienta diferencial a la hora de elegir en que valores invertir y en cuales no. Por este motivo, se ha creado la duda de si los indicadores técnicos son tan útiles como la gente cree o si, por el contrario, se les da un peso demasiado elevado en la actualidad. Sea como fuere, este es un tema que queda pendiente para próximos proyectos.

Índice de figuras

Índice de cuadros

Código

El código que se muestra a continuación es propio y ha sido el utilizado para realizar la segunda parte de este proyecto.

El código se encuentra dividido en secciones, al igual que los apartados donde han sido usados.

Código de Extracción y Depuración

```
import pandas as pd
import matplotlib.pyplot as plt
from alpha_vantage.timeseries import TimeSeries as ts
import time
import numpy as np

ts = TimeSeries(key="ZXRCCY621HLMMUXV",output_format="csv")

table=pd.read_html("https://en.wikipedia.org/wiki/NASDAQ-100#Changes_in_2020")
empresas = table[3]
empresas = empresas["Ticker"]

''' Ejecutar 5 días consecutivos cambiando el parámetro seccion '''

seccion = "year2month9"

tempo=0
starttime = time.time()
while tempo<408:
    print(str(tempo)+"/408")
    if(tempo//102 == 1):
        seccion = "year2month10"
    elif(tempo//102 == 2):
        seccion = "year2month11"
    elif(tempo//102 == 3):
        seccion = "year2month12"
    if(tempo%102 == 0):
        csv_data = ts.get_intraday_extended(symbol=empresas.loc[0],
                                            interval="1min",
                                            slice = seccion)
        df = pd.DataFrame(list(csv_data[0]))
        df = df.drop(columns=[1, 2, 3])
        df = df.drop([0])
        df = df.rename(columns={0: "Fecha", 4: "Valor"+"_"+empresas.loc[0],
                                5:"Volumen"+"_"+empresas.loc[0]})
        df = df.set_index("Fecha")
        df["condicio"] = 0
        for i in range(0,len(df)):
            if((int(df.index[i][11:13])<9) or
               (int(df.index[i][11:13])==9 and int(df.index[i][14:16])<31)
               or (int(df.index[i][11:13])>=16)):
                df.at[df.index[i],"condicio"] = 1
        df = df[df["condicio"]==0]
```

```
        del df["condicio"]
        if(tempo//102 == 0):
            datos1 = df
        elif(tempo//102 == 1):
            datos2 = df
        elif(tempo//102 == 2):
            datos3 = df
        else:
            datos4 = df
    else:
        csv_data = ts.get_intraday_extended(symbol=empresas.loc[tempo%102],
                                            interval="1min",
                                            slice = seccion)
        df = pd.DataFrame(list(csv_data[0]))
        df = df.drop(columns=[1, 2, 3])
        df = df.drop([0])
        df = df.rename(columns={0: "Fecha", 4: "Valor"+"_"+empresas.loc[tempo%102],
                                5:"Volumen"+"_"+empresas.loc[tempo%102]})
        df = df.set_index("Fecha")
        df["condicio"] = 0
        for i in range(0,len(df)):
            if((int(df.index[i][11:13])<9) or
               (int(df.index[i][11:13])==9 and int(df.index[i][14:16])<31)
               or (int(df.index[i][11:13])>=16)):
                df.at[df.index[i],"condicio"] = 1

        df = df[df["condicio"]==0]
        del df["condicio"]
        if(tempo//102 == 0):
            datos1 = pd.merge(datos1,df,on='Fecha',how='outer')
        elif(tempo//102 == 1):
            datos2 = pd.merge(datos2,df,on='Fecha',how='outer')
        elif(tempo//102 == 2):
            datos3 = pd.merge(datos3,df,on='Fecha',how='outer')
        else:
            datos4 = pd.merge(datos4,df,on='Fecha',how='outer')
    time.sleep(21.0 - ((time.time() - starttime) % 21.0))
    tempo = tempo+1


datos1 = datos1.sort_index(ascending=False)
datos2 = datos2.sort_index(ascending=False)
datos3 = datos3.sort_index(ascending=False)
datos4 = datos4.sort_index(ascending=False)

endtime = time.time()

print("Ha trigat"+str((endtime-starttime)/60)+"minuts")

''' Ejecutar hasta aquí '''

''' datos = datos1 '''
''' datos = datos.append(datos2)
datos = datos.append(datos3)
datos = datos.append(datos4) '''

''' datos.to_csv(r"C:\Users\javie\OneDrive\Escritorio\TFG\datosD6.csv")
csvD6 = pd.read_csv("datosD6.csv", index_col=("Fecha")) '''

csvD1 = pd.read_csv("OneDrive\Escritorio\TFG\datosD1.csv", index_col=("Fecha"))
csvD2 = pd.read_csv("OneDrive\Escritorio\TFG\datosD2.csv", index_col=("Fecha"))
csvD3 = pd.read_csv("OneDrive\Escritorio\TFG\datosD3.csv", index_col=("Fecha"))
csvD4 = pd.read_csv("OneDrive\Escritorio\TFG\datosD4.csv", index_col=("Fecha"))
csvD5 = pd.read_csv("OneDrive\Escritorio\TFG\datosD5.csv", index_col=("Fecha"))
csvD6 = pd.read_csv("OneDrive\Escritorio\TFG\datosD6.csv", index_col=("Fecha"))

df = csvD1
df = df.append(csvD2)
df = df.append(csvD3)
for i in range(0,389):
    csvD4 = csvD4.drop(csvD4.index[0])
df = df.append(csvD4)
for i in range(0,389):
    csvD5 = csvD5.drop(csvD5.index[0])
df = df.append(csvD5)
for i in range(0,389):
    csvD6 = csvD6.drop(csvD6.index[0])
df = df.append(csvD6)
```

```
nans = df.isna().sum()/len(df)
nans = np.where(nans>=0.05)[0]

for i in range(0,len(nans)):
    df = df.drop(df.columns[nans[len(nans)-i-1]], axis=1)

for i in range(0,int(len(df.columns)/2)):
    df[df.columns[i*2+1]] = df[df.columns[i*2+1]].fillna(0)

df = df.ffill()
df = df.bfill()

''' df = df.iloc[::-1]
df.to_csv(r"C:\Users\javie\OneDrive\Escritorio\TFG\datos.csv")
df = pd.read_csv("OneDrive\Escritorio\TFG\datos.csv", index_col=("Fecha")) '''
```

Código de Indicadores

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt

df = pd.read_csv("OneDrive\Escritorio\TFG\datos.csv", index_col=("Fecha"))

col_names = []
for i in range(0,int(len(df.columns)/2)):
    col_names.append(df.columns[i*2])

''' Media Movil Simple 5 minutos: '''
print("mm5")
mm5 = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(mm5.columns)):
    print(i)
    for j in range(5,len(mm5)):
        res = 0
        if(df.index[j][8:10]==df.index[j-5][8:10]):
            for k in range(0,5):
                res += df[df.columns[i*2]][j-k]
                mm5[mm5.columns[i]][j] = res/5
# mm5.to_csv(r"C:\Users\javie\OneDrive\Escritorio\TFG\mm5.csv")
# mm5 = pd.read_csv("OneDrive\Escritorio\TFG\mm5.csv", index_col=("Fecha"))

''' Media Movil Simple 15 minutos: '''
print("mm15")
mm15 = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(mm15.columns)):
    print(i)
    for j in range(5,len(mm15)):
        res = 0
        if(df.index[j][8:10]==df.index[j-15][8:10]):
            for k in range(0,15):
                res += df[df.columns[i*2]][j-k]
                mm15[mm15.columns[i]][j] = res/15
# mm15.to_csv(r"C:\Users\javie\OneDrive\Escritorio\TFG\mm15.csv")
# mm15 = pd.read_csv("OneDrive\Escritorio\TFG\mm15.csv", index_col=("Fecha"))

''' Media Movil Simple 60 minutos: '''
print("mm60")
mm60 = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(mm60.columns)):
    print(i)
    for j in range(60,len(mm60)):
        res = 0
        if(df.index[j][8:10]==df.index[j-60][8:10]):
            for k in range(0,60):
                res += df[df.columns[i*2]][j-k]
                mm60[mm60.columns[i]][j] = res/60
# mm60.to_csv(r"C:\Users\javie\OneDrive\Escritorio\TFG\mm60.csv")
# mm60 = pd.read_csv("OneDrive\Escritorio\TFG\mm60.csv", index_col=("Fecha"))

red_mm5 = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(red_mm5.columns)):
    print(i)
    for j in range(0,len(red_mm5)):
```

```
        print(i, j)
        aux = (df[df.columns[i*2]][j])/(mm5[mm5.columns[i]][j])
        red_mm5[red_mm5.columns[i]][j] = np.log(aux)*100
# red_mm5.to_csv(r"C:\Users\javie\OneDrive\Escritorio\TFG\red_mm5.csv")
# red_mm5 = pd.read_csv("OneDrive/Escritorio/TFG/red_mm5.csv", index_col=("Fecha"))

red_mm15 = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(red_mm15.columns)):
    print(i)
    for j in range(0,len(red_mm15)):
        aux = (df[df.columns[i*2]][j])/(mm15[mm15.columns[i]][j])
        red_mm15[red_mm15.columns[i]][j] = np.log(aux)*100
# red_mm15.to_csv(r"C:\Users\javie\OneDrive\Escritorio\TFG\red_mm15.csv")
# red_mm15 = pd.read_csv("OneDrive/Escritorio/TFG/red_mm15.csv", index_col=("Fecha"))

red_mm60 = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(red_mm60.columns)):
    print(i)
    for j in range(0,len(red_mm60)):
        aux = (df[df.columns[i*2]][j])/(mm60[mm60.columns[i]][j])
        red_mm60[red_mm60.columns[i]][j] = np.log(aux)*100
# red_mm60.to_csv(r"C:\Users\javie\OneDrive\Escritorio\TFG\red_mm60.csv")
# red_mm60 = pd.read_csv("OneDrive/Escritorio/TFG/red_mm60.csv", index_col=("Fecha"))

red_mm5_15 = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(red_mm5_15.columns)):
    print(i)
    for j in range(0,len(red_mm5_15)):
        aux = (mm5[mm5.columns[i]][j])/(mm15[mm15.columns[i]][j])
        red_mm5_15[red_mm5_15.columns[i]][j] = np.log(aux)*100
# red_mm5_15.to_csv(r"C:\Users\javie\OneDrive\Escritorio\TFG\red_mm5_15.csv")
# red_mm5_15 = pd.read_csv("OneDrive/Escritorio/TFG/red_mm5_15.csv", index_col=("Fecha"))

red_mm5_60 = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(red_mm5_60.columns)):
    print(i)
    for j in range(0,len(red_mm5_60)):
        aux = (mm5[mm5.columns[i]][j])/(mm60[mm60.columns[i]][j])
        red_mm5_60[red_mm5_60.columns[i]][j] = np.log(aux)*100
# red_mm5_60.to_csv(r"C:\Users\javie\OneDrive\Escritorio\TFG\red_mm5_60.csv")
# red_mm5_60 = pd.read_csv("OneDrive/Escritorio/TFG/red_mm5_60.csv", index_col=("Fecha"))

red_mm15_60 = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(red_mm15_60.columns)):
    print(i)
    for j in range(0,len(red_mm15_60)):
        aux = (mm15[mm15.columns[i]][j])/(mm60[mm60.columns[i]][j])
        red_mm15_60[red_mm15_60.columns[i]][j] = np.log(aux)*100
# red_mm15_60.to_csv(r"C:\Users\javie\OneDrive\Escritorio\TFG\red_mm15_60.csv")
# red_mm15_60 = pd.read_csv("OneDrive/Escritorio/TFG/red_mm15_60.csv", index_col=("Fecha"))

''' Momento 5 minutos: '''
print("mom5")
red_mom5 = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(red_mom5.columns)):
    print(i)
    for j in range(5,len(red_mom5)):
        print(i, j)
        if(df.index[j][8:10]==df.index[j-5][8:10]):
            red_mom5[red_mom5.columns[i]][j] = (df[df.columns[i*2]][j]-df[df.columns[i*2]][j-5])*10/df[df.columns[i*2]][j]
# red_mom5.to_csv(r"C:\Users\javie\OneDrive\Escritorio\TFG\red_mom5.csv")
# red_mom5 = pd.read_csv("OneDrive/Escritorio/TFG/red_mom5.csv", index_col=("Fecha"))

''' Momento 15 minutos: '''
print("mom15")
red_mom15 = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(red_mom15.columns)):
    print(i)
    for j in range(15,len(red_mom15)):
        if(df.index[j][8:10]==df.index[j-15][8:10]):
            red_mom15[red_mom15.columns[i]][j] = (df[df.columns[i*2]][j]-df[df.columns[i*2]][j-15])*10/df[df.columns[i*2]][j]
# red_mom15.to_csv(r"C:\Users\javie\OneDrive\Escritorio\TFG\red_mom15.csv")
# red_mom15 = pd.read_csv("OneDrive/Escritorio/TFG/red_mom15.csv", index_col=("Fecha"))

''' Momento 60 minutos: '''
print("mom60")
red_mom60 = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(red_mom60.columns)):
    print(i)
```

```
    for j in range(60,len(red_mom60)):
        if(df.index[j][8:10]==df.index[j-60][8:10]):
            red_mom60[red_mom60.columns[i]][j] = (df[df.columns[i*2]][j]-df[df.columns[i*2]][j-60])*10/df[df.columns[i*2]][j]
# red_mom60.to_csv(r"C:\Users\javie\OneDrive\Escritorio\TFG\red_mom60.csv")
# red_mom60 = pd.read_csv("OneDrive/Escritorio/TFG/red_mom60.csv", index_col=("Fecha"))

''' RSI 15 minutos: '''
print("rsi15")
red_rsi15 = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(red_rsi15.columns)):
    print(i)
    for j in range(15,len(red_rsi15)):
        aug_count = 0
        aug_sum = 0
        dis_count = 0
        dis_sum = 0
        if(df.index[j][8:10]==df.index[j-15][8:10]):
            for k in range(0,15):
                if(df[df.columns[i*2]][j-k]>df[df.columns[i*2]][j-k-1]):
                    aug_count += 1
                    aug_sum += df[df.columns[i*2]][j-k] - df[df.columns[i*2]][j-k-1]
                elif(df[df.columns[i*2]][j-k]<df[df.columns[i*2]][j-k-1]):
                    dis_count += 1
                    dis_sum += df[df.columns[i*2]][j-k-1] - df[df.columns[i*2]][j-k]
            if(dis_count == 0 and aug_count == 0):
                red_rsi15[red_rsi15.columns[i]][j] = 0
            elif(dis_count == 0 and aug_count != 0):
                red_rsi15[red_rsi15.columns[i]][j] = 1
            elif(aug_count == 0 and dis_count != 0):
                red_rsi15[red_rsi15.columns[i]][j] = -1
            else:
                red_rsi15[red_rsi15.columns[i]][j] = (50-(100/(1+((aug_sum/aug_count)/(dis_sum/dis_count)))))/50
# red_rsi15.to_csv(r"C:\Users\javie\OneDrive\Escritorio\TFG\red_rsi15.csv")
# red_rsi15 = pd.read_csv("OneDrive/Escritorio/TFG/red_rsi15.csv", index_col=("Fecha"))

''' RSI 60 minutos: '''
print("rsi60")
red_rsi60 = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(red_rsi60.columns)):
    print(i)
    for j in range(60,len(red_rsi60)):
        aug_count = 0
        aug_sum = 0
        dis_count = 0
        dis_sum = 0
        if(df.index[j][8:10]==df.index[j-60][8:10]):
            for k in range(0,60):
                if(df[df.columns[i*2]][j-k]>df[df.columns[i*2]][j-k-1]):
                    aug_count += 1
                    aug_sum += df[df.columns[i*2]][j-k] - df[df.columns[i*2]][j-k-1]
                elif(df[df.columns[i*2]][j-k]<df[df.columns[i*2]][j-k-1]):
                    dis_count += 1
                    dis_sum += df[df.columns[i*2]][j-k-1] - df[df.columns[i*2]][j-k]
            if(dis_count == 0 and aug_count == 0):
                red_rsi60[red_rsi60.columns[i]][j] = 0
            elif(dis_count == 0 and aug_count != 0):
                red_rsi60[red_rsi60.columns[i]][j] = 1
            elif(aug_count == 0 and dis_count != 0):
                red_rsi60[red_rsi60.columns[i]][j] = -1
            else:
                red_rsi60[red_rsi60.columns[i]][j] = (50-(100/(1+((aug_sum/aug_count)/(dis_sum/dis_count)))))/50
# red_rsi60.to_csv(r"C:\Users\javie\OneDrive\Escritorio\TFG\red_rsi60.csv")
# red_rsi60 = pd.read_csv("OneDrive/Escritorio/TFG/red_rsi60.csv", index_col=("Fecha"))

''' Estocastico K 55 minutos, D 5 minutos '''
print("estocastico")
red_sto55 = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(red_sto55.columns)):
    print(i)
    for j in range(55,len(red_sto55)):
        if(df.index[j][8:10]==df.index[j-55][8:10]):
            vec = []
            for k in range(0,55):
                vec.append(df[df.columns[i*2]][j-k])
            nume = df[df.columns[i*2]][j] - min(vec)
            deno = max(vec) - min(vec)
            if(deno == 0):
                red_sto55[red_sto55.columns[i]][j] = 1
            else:
```

```
                red_sto55[red_sto55.columns[i]][j] = nume/deno
red_sto55_5 = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(red_sto55_5.columns)):
    print(i)
    for j in range(60,len(red_sto55_5)):
        if(red_sto55.index[j][8:10]==red_sto55.index[j-60][8:10]):
            res = 0
            for k in range(0,5):
                res += red_sto55[red_sto55.columns[i]][j-k]
            red_sto55_5[red_sto55_5.columns[i]][j] = res/5
# red_sto55_5.to_csv(r"C:\Users\javie\OneDrive\Escritorio\TFG\red_sto55_5.csv")
# red_sto55_5 = pd.read_csv("OneDrive/Escritorio/TFG/red_sto55_5.csv", index_col=("Fecha"))

''' Williams 15 minutos '''
print("will15")
red_will15 = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(red_will15.columns)):
    print(i)
    for j in range(15,len(red_will15)):
        if(df.index[j][8:10]==df.index[j-15][8:10]):
            vec = []
            for k in range(0,15):
                vec.append(df[df.columns[i*2]][j-k])
            nume = max(vec) - df[df.columns[i*2]][j]
            deno = max(vec) - min(vec)
            if(deno == 0):
                print(i, j)
                red_will15[red_will15.columns[i]][j] = 1
            else:
                red_will15[red_will15.columns[i]][j] = (-1*nume/deno)+1
# red_will15.to_csv(r"C:\Users\javie\OneDrive\Escritorio\TFG\red_will15.csv")
# red_will15 = pd.read_csv("OneDrive/Escritorio/TFG/red_will15.csv", index_col=("Fecha"))

''' Williams 60 minutos '''
print("will60")
red_will60 = pd.DataFrame(columns=col_names, index=df.index)
for i in range(73,len(red_will60.columns)):
    print(i)
    for j in range(15,len(red_will60)):
        if(df.index[j][8:10]==df.index[j-60][8:10]):
            vec = []
            for k in range(0,60):
                vec.append(df[df.columns[i*2]][j-k])
            nume = max(vec) - df[df.columns[i*2]][j]
            deno = max(vec) - min(vec)
            if(deno == 0):
                print(i, j)
                red_will60[red_will60.columns[i]][j] = 1
            else:
                red_will60[red_will60.columns[i]][j] = (-1*nume/deno)+1
# red_will60.to_csv(r"C:\Users\javie\OneDrive\Escritorio\TFG\red_will60.csv")
# red_will60 = pd.read_csv("OneDrive/Escritorio/TFG/red_will60.csv", index_col=("Fecha"))

''' MACD 12,26 Señal 9 '''
print("macd")
ema12 = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(ema12.columns)):
    print(i)
    aux = 0
    for k in range(0,12):
        aux += df[df.columns[i*2]][12-k-1]
    ema12[ema12.columns[i]][11] = aux/12
    for j in range(12,len(ema12)):
        if(df.index[j][8:10]!=df.index[j-13][8:10] and df.index[j][14:16]=="42"):
            aux = 0
            for k in range(0,12):
                aux += df[df.columns[i*2]][j-k-1]
            ema12[ema12.columns[i]][j] = aux/12
        if(df.index[j][8:10]==df.index[j-12][8:10]):
            alfa = 2/(12+1)
            ema12[ema12.columns[i]][j] = alfa*df[df.columns[i*2]][j]+(1-alfa)*ema12[ema12.columns[i]][j-1]
ema26= pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(ema26.columns)):
    print(i)
    aux = 0
    for k in range(0,26):
        aux += df[df.columns[i*2]][26-k-1]
    ema26[ema26.columns[i]][25] = aux/26
    for j in range(26,len(ema26)):
```

```
        if(df.index[j][8:10]!=df.index[j-27][8:10] and df.index[j][14:16]=="56"):
            aux = 0
            for k in range(0,26):
                aux += df[df.columns[i*2]][j-k-1]
            ema26[ema26.columns[i]][j] = aux/26
        if(df.index[j][8:10]==df.index[j-26][8:10]):
            alfa = 2/(26+1)
            ema26[ema26.columns[i]][j] = alfa*df[df.columns[i*2]][j]+(1-alfa)*ema26[ema26.columns[i]][j-1]
macd = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(macd.columns)):
    print(i)
    for j in range(0,len(macd)):
        macd[macd.columns[i]][j] = ema12[ema12.columns[i]][j] - ema26[ema26.columns[i]][j]
sma5 = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(sma5.columns)):
    print(i)
    for j in range(5,len(sma5)):
        res = 0
        if(df.index[j][8:10]==df.index[j-5][8:10]):
            for k in range(0,5):
                res += macd[macd.columns[i]][j-k]
                sma5[sma5.columns[i]][j] = res/5
red_macd12_26_5 = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(red_macd12_26_5.columns)):
    print(i)
    for j in range(0,len(red_macd12_26_5)):
        red_macd12_26_5[red_macd12_26_5.columns[i]][j] = (macd[macd.columns[i]][j] - sma5[sma5.columns[i]][j])
# red_macd12_26_5.to_csv(r"C:\Users\javie\OneDrive\Escritorio\TFG\red_macd12_26_5.csv")
# red_macd12_26_5 = pd.read_csv("OneDrive/Escritorio/TFG/red_macd12_26_5.csv", index_col=("Fecha"))




################################## Gráficos ##################################

import pandas as pd
df = pd.read_csv("Desktop\Javier\datos.csv", index_col=("Fecha"))
df00 = df[df.columns[0:4]][0:5]
df01 = df[df.columns[df.shape[1]-4:df.shape[1]]][0:5]
df10 = df[df.columns[0:4]][df.shape[0]-5:df.shape[0]]
df11 = df[df.columns[df.shape[1]-4:df.shape[1]]][df.shape[0]-5:df.shape[0]]
df0 = pd.concat([df00, df10])
df1 = pd.concat([df01, df11])
print(df0.to_latex(index=True))
print(df1.to_latex(index=True))

df[df.columns[40]][850:1150].plot(figsize=(18,9),rot = 20, title = "Medias móviles")
mm5[mm5.columns[20]][850:1150].plot(figsize=(18,9),rot = 20, title = "Medias móviles")
mm60[mm60.columns[20]][850:1150].plot(figsize=(18,9),rot = 20, title = "Medias móviles")
plt.legend(["Cotización", "MMS 5", "MMS 60"])


red_mm5[red_mm5.columns[20]][850:1150].plot(figsize=(18,9),rot = 20, title = "Diferencia Medias móviles")
red_mm15_60[red_mm15_60.columns[20]][850:1150].plot(figsize=(18,9),rot = 20, title = "Diferencia Medias móviles")
plt.legend(["Dif MMS 5", "Dif MMS 15-60"])

# momentos se obtiene del libro de John Murphy

red_mom15[red_mom15.columns[20]][850:1150].plot(figsize=(18,9),rot = 20, title = "Diferencia Momentos")
red_mom60[red_mom60.columns[20]][850:1150].plot(figsize=(18,9),rot = 20, title = "Diferencia Momentos")
plt.legend(["Dif Mom 15", "Dif Mom 60"])

red_rsi15[red_rsi15.columns[20]][850:1150].plot(figsize=(18,9),rot = 20, title = "RSI")
red_rsi60[red_rsi60.columns[20]][850:1150].plot(figsize=(18,9),rot = 20, title = "RSI")
plt.legend(["RSI Mod 15", "RSI Mod 60"])

red_sto55_5[red_sto55_5.columns[20]][850:1150].plot(figsize=(18,9),rot = 20, title = "Estocástico")
plt.legend(["Estocástico 55-5"])

red_will15[red_will15.columns[20]][850:1150].plot(figsize=(18,9),rot = 20, title = "Williams")
red_will60[red_will60.columns[20]][850:1150].plot(figsize=(18,9),rot = 20, title = "Williams")
plt.legend(["Will 15", "Will 60"])

red_macd12_26_5[red_macd12_26_5.columns[20]][850:1150].plot(figsize=(18,9),rot = 20, title = "MACD")
plt.legend(["MACD 12-26, 5"])
```

Código de LSTM

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense

df = pd.read_csv("OneDrive\Escritorio\TFG\datos.csv", index_col=("Fecha"))

col_names = []
for i in range(0,int(len(df.columns)/2)):
    col_names.append(df.columns[i*2])

for empr in range(22, 30): #range(k, k+1)
    lstm_x = []
    lstm_y = []
    for i in range(0,int(len(df)*0.84305)):
        if(df.index[i][8:10] == df.index[i+62][8:10]):
            add_x = df[df.columns[empr*2]][i:60+i]/df[df.columns[empr*2]][i]-1
            add_y = df[df.columns[empr*2]][60+i]/df[df.columns[empr*2]][i]-1
            lstm_x.append(add_x)
            lstm_y.append(add_y)
    lstm_x = np.array(lstm_x)
    lstm_y_ini = np.array(lstm_y)
    lstm_x_ini = lstm_x.reshape((lstm_x.shape[0], lstm_x.shape[1], 1))
    for seed in range(0, 5):
        print(empr, seed)
        tf.random.set_seed(seed)
        globals()[f"model{col_names[empr][5:]}{seed}"] = Sequential()
        globals()[f"model{col_names[empr][5:]}{seed}"].add(LSTM(60, activation='relu', return_sequences=True, input_shape=(60, 1)))
        globals()[f"model{col_names[empr][5:]}{seed}"].add(LSTM(60, activation='relu'))
        globals()[f"model{col_names[empr][5:]}{seed}"].add(Dense(1))
        globals()[f"model{col_names[empr][5:]}{seed}"].compile(optimizer='adam', loss='mean_squared_error')
        history = globals()[f"model{col_names[empr][5:]}{seed}"].fit(lstm_x_ini, lstm_y_ini, validation_split=0.25, epochs=50, batch_size=64, verbose=1)
        plt.plot(history.history['loss'])
        plt.plot(history.history['val_loss'])
        escriu = 'model loss ' + str(empr) + ' ' + str(seed)
        plt.title(escriu)
        plt.ylabel('loss')
        plt.xlabel('epoch')
        plt.legend(['train', 'test'], loc='upper left')
        plt.show()
        lstm_x = []
        lstm_y = []
        aux_x = []
        pred = []
        real = []
        fecha = []
        for i in range(int(len(df)*0.84305*0.75+100),int(len(df)*0.84305-100)):
            if(df.index[i][8:10] == df.index[i+62][8:10]):
                aux = df[df.columns[empr*2]][i:60+i]/df[df.columns[empr*2]][i]-1
                real.append(df[df.columns[empr*2]][60+i])
                aux_x.append(df[df.columns[empr*2]][i])
                fecha.append(df.index[60+i])
                lstm_aux = np.array(aux)
                lstm_aux = lstm_aux.reshape((1, 60, 1))
                lstm_pred = globals()[f"model{col_names[empr][5:]}{seed}"].predict(lstm_aux, verbose = 0)
                pred.append(float(lstm_pred))
        aux = pd.DataFrame(aux_x)
        pred = pd.DataFrame(pred)
        globals()[f"real{col_names[empr][5:]}{seed}"] = pd.DataFrame(real)
        res = []
        for i in range(1,len(aux)):
            res.append((pred[0][i]+1)*aux[0][i])
        globals()[f"res{col_names[empr][5:]}{seed}"] = pd.DataFrame(res)
        diferencies = pd.DataFrame(columns=["pred", "real"], index=fecha)
        for i in range(1, len(res)):
            if(diferencies.index[i+1][8:10] == diferencies.index[i-1][8:10]):
                diferencies["pred"][i] = (globals()[f"res{col_names[empr][5:]}{seed}"][0][i]/globals()[f"real{col_names[empr][5:]}{seed}"][0][i-1]-1)*100
                diferencies["real"][i] = (globals()[f"real{col_names[empr][5:]}{seed}"][0][i]/globals()[f"real{col_names[empr][5:]}{seed}"][0][i-1]-1)*100
        globals()[f"diferencies{col_names[empr][5:]}{seed}"] = diferencies
        escriu = 'diferencies ' + str(empr) + ' ' + str(seed)
        globals()[f"diferencies{col_names[empr][5:]}{seed}"][500:600].plot(figsize=(15,7),rot = 20, title = escriu)
        globals()[f"diferencies{col_names[empr][5:]}{seed}"][4900:4950].plot(figsize=(15,7),rot = 20, title = escriu)
        globals()[f"diferencies{col_names[empr][5:]}{seed}"][10000:10100].plot(figsize=(15,7),rot = 20, title = escriu)
        plt.show()
```

```
########################################################################

semilla = [ 1, 3, 8, 0,14, 1,14, 8, 5, 9,
            2, 2,21, 1, 1, 2, 2, 2, 5,12,
            2, 5, 1,11,11, 2, 6, 8, 7, 9,
            6, 3, 5, 2, 3, 6, 8, 8,12,13,
           25, 1, 6, 3, 8, 3, 0, 0, 1, 7,
            4, 8,10,10, 7, 2, 1,11, 2, 1,
            0, 3,10,19, 6, 1, 0, 2, 4, 0,
            4, 1, 2, 7, 6,23,12,12, 2,10]

import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense

df = pd.read_csv("OneDrive\Escritorio\TFG\datos.csv", index_col=("Fecha"))

col_names = []
for i in range(0,int(len(df.columns)/2)):
    col_names.append(df.columns[i*2])

empresa = 0

for empr in range(empresa, empresa+80):
    seed = semilla[empr]
    lstm_x = []
    lstm_y = []
    for i in range(0,int(len(df)*0.84305)):
        if(df.index[i][8:10] == df.index[i+62][8:10]):
            add_x = df[df.columns[empr*2]][i:60+i]/df[df.columns[empr*2]][i]-1
            add_y = df[df.columns[empr*2]][60+i]/df[df.columns[empr*2]][i]-1
            lstm_x.append(add_x)
            lstm_y.append(add_y)
    lstm_x = np.array(lstm_x)
    lstm_y_ini = np.array(lstm_y)
    lstm_x_ini = lstm_x.reshape((lstm_x.shape[0], lstm_x.shape[1], 1))
    print(empr, seed)
    tf.random.set_seed(seed)
    globals()[f"model{col_names[empr][5:]}"] = Sequential()
    globals()[f"model{col_names[empr][5:]}"].add(LSTM(60, activation='relu', return_sequences=True, input_shape=(60, 1)))
    globals()[f"model{col_names[empr][5:]}"].add(LSTM(60, activation='relu'))
    globals()[f"model{col_names[empr][5:]}"].add(Dense(1))
    globals()[f"model{col_names[empr][5:]}"].compile(optimizer='adam', loss='mean_squared_error')
    history = globals()[f"model{col_names[empr][5:]}"].fit(lstm_x_ini, lstm_y_ini, validation_split=0.25, epochs=50, batch_size=64, verbose=1)
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    escriu = 'model loss ' + str(empr) + ' ' + str(seed)
    plt.title(escriu)
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()
    lstm_x = []
    lstm_y = []
    aux_x = []
    pred = []
    real = []
    fecha = []
    for i in range(int(len(df)*0.84305*0.75+100),int(len(df)-100)):
        if(df.index[i][8:10] == df.index[i+62][8:10]):
            aux = df[df.columns[empr*2]][i:60+i]/df[df.columns[empr*2]][i]-1
            real.append(df[df.columns[empr*2]][60+i])
            aux_x.append(df[df.columns[empr*2]][i])
            fecha.append(df.index[60+i])
            lstm_aux = np.array(aux)
            lstm_aux = lstm_aux.reshape((1, 60, 1))
            lstm_pred = globals()[f"model{col_names[empr][5:]}"].predict(lstm_aux, verbose = 0)
            pred.append(float(lstm_pred))
    aux = pd.DataFrame(aux_x)
    pred = pd.DataFrame(pred)
    globals()[f"real{col_names[empr][5:]}"] = pd.DataFrame(real)
    res = []
    for i in range(1,len(aux)):
        res.append((pred[0][i]+1)*aux[0][i])
```

```
    globals()[f"res{col_names[empr][5:]}"] = pd.DataFrame(res)
    diferencies = pd.DataFrame(columns=["pred", "real"], index=fecha)
    for i in range(1, len(res)):
        if(diferencies.index[i+1][8:10] == diferencies.index[i-1][8:10]):
            diferencies["pred"][i] = (globals()[f"res{col_names[empr][5:]}"][0][i]/globals()[f"real{col_names[empr][5:]}"][0][i-1]-1)*100
            diferencies["real"][i] = (globals()[f"real{col_names[empr][5:]}"][0][i]/globals()[f"real{col_names[empr][5:]}"][0][i-1]-1)*100
    globals()[f"diferencies{col_names[empr][5:]}"] = diferencies
    escriu = 'diferencies ' + str(empr) + ' ' + str(seed)
    globals()[f"diferencies{col_names[empr][5:]}"][500:600].plot(figsize=(15,7),rot = 20, title = escriu)
    globals()[f"diferencies{col_names[empr][5:]}"][4900:4950].plot(figsize=(15,7),rot = 20, title = escriu)
    globals()[f"diferencies{col_names[empr][5:]}"][10000:10100].plot(figsize=(15,7),rot = 20, title = escriu)
    plt.show()
```

Código de FFNN

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense
import math

df = pd.read_csv("OneDrive\Escritorio\TFG\datos.csv", index_col=("Fecha"))

col_names = []
for i in range(0,int(len(df.columns)/2)):
    col_names.append(df.columns[i*2])

red_volum = pd.DataFrame(columns=col_names, index=df.index)
for i in range(0,len(red_volum.columns)):
    print(i)
    for j in range(0,len(red_volum)):
        red_volum[red_volum.columns[i]][j] = df[df.columns[i*2+1]][j]
# red_volum.to_csv(r"C:\Users\javie\OneDrive\Escritorio\TFG\red_volum.csv")
# red_volum = pd.read_csv("OneDrive/Escritorio/TFG/red_volum.csv", index_col=("Fecha"))

red_mm5 = pd.read_csv("OneDrive/Escritorio/TFG/red_mm5.csv", index_col=("Fecha"))
red_mm15 = pd.read_csv("OneDrive/Escritorio/TFG/red_mm15.csv", index_col=("Fecha"))
red_mm60 = pd.read_csv("OneDrive/Escritorio/TFG/red_mm60.csv", index_col=("Fecha"))
red_mm5_15 = pd.read_csv("OneDrive/Escritorio/TFG/red_mm5_15.csv", index_col=("Fecha"))
red_mm5_60 = pd.read_csv("OneDrive/Escritorio/TFG/red_mm5_60.csv", index_col=("Fecha"))
red_mm15_60 = pd.read_csv("OneDrive/Escritorio/TFG/red_mm15_60.csv", index_col=("Fecha"))
red_mom5 = pd.read_csv("OneDrive/Escritorio/TFG/red_mom5.csv", index_col=("Fecha"))
red_mom15 = pd.read_csv("OneDrive/Escritorio/TFG/red_mom15.csv", index_col=("Fecha"))
red_mom60 = pd.read_csv("OneDrive/Escritorio/TFG/red_mom60.csv", index_col=("Fecha"))
red_rsi15 = pd.read_csv("OneDrive/Escritorio/TFG/red_rsi15.csv", index_col=("Fecha"))
red_rsi60 = pd.read_csv("OneDrive/Escritorio/TFG/red_rsi60.csv", index_col=("Fecha"))
red_sto55_5 = pd.read_csv("OneDrive/Escritorio/TFG/red_sto55_5.csv", index_col=("Fecha"))
red_will15 = pd.read_csv("OneDrive/Escritorio/TFG/red_will15.csv", index_col=("Fecha"))
red_will60 = pd.read_csv("OneDrive/Escritorio/TFG/red_will60.csv", index_col=("Fecha"))
red_macd12_26_5 = pd.read_csv("OneDrive/Escritorio/TFG/red_macd12_26_5.csv", index_col=("Fecha"))
red_volum = pd.read_csv("OneDrive/Escritorio/TFG/red_volum.csv", index_col=("Fecha"))

semilla = [ 1, 3, 8, 0,14, 1,14, 8, 5, 9,
            2, 2,21, 1, 1, 2, 2, 2, 5,12,
            2, 5, 1,11,11, 2, 6, 8, 7, 9,
            6, 3, 5, 2, 3, 6, 8, 8,12,13,
           25, 1, 6, 3, 8, 3, 0, 0, 1, 7,
            4, 8,10,10, 7, 2, 1,11, 2, 1,
            0, 3,10,19, 6, 1, 0, 2, 4, 0,
            4, 1, 2, 7, 6,23,12,12, 2,10]

empresa = 0

for empr in range(empresa, empresa+80):
    seed = semilla[empr]
    lstm_x = []
    lstm_y = []
    for i in range(0,int(len(df)*0.84305)):
        if(df.index[i][8:10] == df.index[i+62][8:10]):
            add_x = df[df.columns[empr*2]][i:60+i]/df[df.columns[empr*2]][i]-1
            add_y = df[df.columns[empr*2]][60+i]/df[df.columns[empr*2]][i]-1
            lstm_x.append(add_x)
            lstm_y.append(add_y)
```

```
    lstm_x = np.array(lstm_x)
    lstm_y_ini = np.array(lstm_y)
    lstm_x_ini = lstm_x.reshape((lstm_x.shape[0], lstm_x.shape[1], 1))
    print(empr, seed)
    tf.random.set_seed(seed)
    globals()[f"model{col_names[empr][5:]}"] = Sequential()
    globals()[f"model{col_names[empr][5:]}"].add(LSTM(60, activation='relu', return_sequences=True, input_shape=(60, 1)))
    globals()[f"model{col_names[empr][5:]}"].add(LSTM(60, activation='relu'))
    globals()[f"model{col_names[empr][5:]}"].add(Dense(1))
    globals()[f"model{col_names[empr][5:]}"].compile(optimizer='adam', loss='mean_squared_error')
    history = globals()[f"model{col_names[empr][5:]}"].fit(lstm_x_ini, lstm_y_ini, validation_split=0.75, epochs=50, batch_size=64, verbose=1)
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    escriu = 'model loss ' + str(empr) + ' ' + str(seed)
    plt.title(escriu)
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()
    lstm_x = []
    lstm_y = []
    aux_x = []
    pred = []
    real = []
    fecha = []
    for i in range(0,int(len(df)-70)):
        if(df.index[i][8:10] == df.index[i+62][8:10]):
            aux = df[df.columns[empr*2]][i:60+i]/df[df.columns[empr*2]][i]-1
            real.append(df[df.columns[empr*2]][60+i])
            aux_x.append(df[df.columns[empr*2]][i])
            fecha.append(df.index[60+i])
            lstm_aux = np.array(aux)
            lstm_aux = lstm_aux.reshape((1, 60, 1))
            lstm_pred = globals()[f"model{col_names[empr][5:]}"].predict(lstm_aux, verbose = 0)
            pred.append(float(lstm_pred))
    aux = pd.DataFrame(aux_x)
    pred = pd.DataFrame(pred)
    globals()[f"real{col_names[empr][5:]}"] = pd.DataFrame(real)
    res = []
    for i in range(1,len(aux)):
        res.append((pred[0][i]+1)*aux[0][i])
    globals()[f"res{col_names[empr][5:]}"] = pd.DataFrame(res)
    diferencies = pd.DataFrame(columns=["pred", "real"], index=fecha)
    for i in range(1, len(res)):
        if(diferencies.index[i+1][8:10] == diferencies.index[i-1][8:10]):
            diferencies["pred"][i] = (globals()[f"res{col_names[empr][5:]}"][0][i]/globals()[f"real{col_names[empr][5:]}"][0][i-1]-1)*100
            diferencies["real"][i] = (globals()[f"real{col_names[empr][5:]}"][0][i]/globals()[f"real{col_names[empr][5:]}"][0][i-1]-1)*100
    globals()[f"diferencies{col_names[empr][5:]}"] = diferencies

index_df = []
for i in range(0,len(globals()[f"diferencies{col_names[0][5:]}"])):
    if(math.isnan(globals()[f"diferencies{col_names[0][5:]}"]["pred"][i]) == False):
        index_df.append(globals()[f"diferencies{col_names[0][5:]}"].index[i])

col_df = ["mm5", "mm15", "mm60", "mm5_15", "mm5_60", "mm15_60", "mom5", "mom15", "mom60",
          "rsi15", "rsi60", "sto55_5", "will15", "will60", "macd12_26_5", "dif_lstm", "volum", "real"]

for i in range(1,len(df)):
    red_mm5.iloc[i-1,:] = red_mm5.iloc[i,:]
    red_mm15.iloc[i-1,:] = red_mm15.iloc[i,:]
    red_mm60.iloc[i-1,:] = red_mm60.iloc[i,:]
    red_mm5_15.iloc[i-1,:] = red_mm5_15.iloc[i,:]
    red_mm5_60.iloc[i-1,:] = red_mm5_60.iloc[i,:]
    red_mm15_60.iloc[i-1,:] = red_mm15_60.iloc[i,:]
    red_mom5.iloc[i-1,:] = red_mom5.iloc[i,:]
    red_mom15.iloc[i-1,:] = red_mom15.iloc[i,:]
    red_mom60.iloc[i-1,:] = red_mom60.iloc[i,:]
    red_rsi15.iloc[i-1,:] = red_rsi15.iloc[i,:]
    red_rsi60.iloc[i-1,:] = red_rsi60.iloc[i,:]
    red_sto55_5.iloc[i-1,:] = red_sto55_5.iloc[i,:]
    red_will15.iloc[i-1,:] = red_will15.iloc[i,:]
    red_will60.iloc[i-1,:] = red_will60.iloc[i,:]
    red_macd12_26_5.iloc[i-1,:] = red_macd12_26_5.iloc[i,:]
    red_volum.iloc[i-1,:] = red_volum.iloc[i,:]
    print(i)

for empr in range(0,len(col_names)):
    globals()[f"df{col_names[empr][5:]}"] = pd.DataFrame(columns=[], index=index_df)
    globals()[f"df{col_names[empr][5:]}"] = pd.merge(globals()[f"df{col_names[empr][5:]}"],
```

```
                                                red_mm5[red_mm5.columns[empr]], left_index=True, right_index=True)
    globals()[f"df{col_names[empr][5:]}"] = pd.merge(globals()[f"df{col_names[empr][5:]}"],
                                                red_mm15[red_mm15.columns[empr]], left_index=True, right_index=True)
    globals()[f"df{col_names[empr][5:]}"] = pd.merge(globals()[f"df{col_names[empr][5:]}"],
                                                red_mm60[red_mm60.columns[empr]], left_index=True, right_index=True)
    globals()[f"df{col_names[empr][5:]}"] = pd.merge(globals()[f"df{col_names[empr][5:]}"],
                                                red_mm5_15[red_mm5_15.columns[empr]], left_index=True, right_index=True)
    globals()[f"df{col_names[empr][5:]}"] = pd.merge(globals()[f"df{col_names[empr][5:]}"],
                                                red_mm5_60[red_mm5_60.columns[empr]], left_index=True, right_index=True)
    globals()[f"df{col_names[empr][5:]}"] = pd.merge(globals()[f"df{col_names[empr][5:]}"],
                                                red_mm15_60[red_mm15_60.columns[empr]], left_index=True, right_index=True)
    globals()[f"df{col_names[empr][5:]}"] = pd.merge(globals()[f"df{col_names[empr][5:]}"],
                                                red_mom5[red_mom5.columns[empr]], left_index=True, right_index=True)
    globals()[f"df{col_names[empr][5:]}"] = pd.merge(globals()[f"df{col_names[empr][5:]}"],
                                                red_mom15[red_mom15.columns[empr]], left_index=True, right_index=True)
    globals()[f"df{col_names[empr][5:]}"] = pd.merge(globals()[f"df{col_names[empr][5:]}"],
                                                red_mom60[red_mom60.columns[empr]], left_index=True, right_index=True)
    globals()[f"df{col_names[empr][5:]}"] = pd.merge(globals()[f"df{col_names[empr][5:]}"],
                                                red_rsi15[red_rsi15.columns[empr]], left_index=True, right_index=True)
    globals()[f"df{col_names[empr][5:]}"] = pd.merge(globals()[f"df{col_names[empr][5:]}"],
                                                red_rsi60[red_rsi60.columns[empr]], left_index=True, right_index=True)
    globals()[f"df{col_names[empr][5:]}"] = pd.merge(globals()[f"df{col_names[empr][5:]}"],
                                                red_sto55_5[red_sto55_5.columns[empr]], left_index=True, right_index=True)
    globals()[f"df{col_names[empr][5:]}"] = pd.merge(globals()[f"df{col_names[empr][5:]}"],
                                                red_will15[red_will15.columns[empr]], left_index=True, right_index=True)
    globals()[f"df{col_names[empr][5:]}"] = pd.merge(globals()[f"df{col_names[empr][5:]}"],
                                                red_will60[red_will60.columns[empr]], left_index=True, right_index=True)
    globals()[f"df{col_names[empr][5:]}"] = pd.merge(globals()[f"df{col_names[empr][5:]}"],
                                                red_macd12_26_5[red_macd12_26_5.columns[empr]], left_index=True, right_index=True)
    globals()[f"df{col_names[empr][5:]}"] = pd.merge(globals()[f"df{col_names[empr][5:]}"],
                                                globals()[f"diferencies{col_names[empr][5:]}"][globals()[f"diferencies{col_names[empr][5:]}"].columns[0]],
                                                left_index=True, right_index=True)
    globals()[f"df{col_names[empr][5:]}"] = pd.merge(globals()[f"df{col_names[empr][5:]}"],
                                                red_volum[red_volum.columns[empr]], left_index=True, right_index=True)
    globals()[f"df{col_names[empr][5:]}"] = pd.merge(globals()[f"df{col_names[empr][5:]}"],
                                                globals()[f"diferencies{col_names[empr][5:]}"][globals()[f"diferencies{col_names[empr][5:]}"].columns[1]],
                                                left_index=True, right_index=True)
    print(empr)
    globals()[f"df{col_names[empr][5:]}"].columns = col_df


for empr in range(0,len(col_names)):
    print(col_names[empr][6:], globals()[f"df{col_names[empr][5:]}"].isna().sum().sum())
    globals()[f"df{col_names[empr][5:]}"] = globals()[f"df{col_names[empr][5:]}"].fillna(0)
    print(col_names[empr][6:], globals()[f"df{col_names[empr][5:]}"].isna().sum().sum())
    globals()[f"table{col_names[empr][5:]}"] = globals()[f"df{col_names[empr][5:]}"].to_numpy()
    globals()[f"x_train{col_names[empr][5:]}"] = globals()[f"table{col_names[empr][5:]}"][0:33445,0:17]
    globals()[f"y_train{col_names[empr][5:]}"] = globals()[f"table{col_names[empr][5:]}"][0:33445,17]
    globals()[f"x_train{col_names[empr][5:]}"] = np.asarray(globals()[f"x_train{col_names[empr][5:]}"]).astype('float32')
    globals()[f"y_train{col_names[empr][5:]}"] = np.asarray(globals()[f"y_train{col_names[empr][5:]}"]).astype('float32')
    globals()[f"x_val{col_names[empr][5:]}"] = globals()[f"table{col_names[empr][5:]}"][33445:,0:17]
    globals()[f"y_val{col_names[empr][5:]}"] = globals()[f"table{col_names[empr][5:]}"][33445:,17]
    globals()[f"x_val{col_names[empr][5:]}"] = np.asarray(globals()[f"x_val{col_names[empr][5:]}"]).astype('float32')
    globals()[f"y_val{col_names[empr][5:]}"] = np.asarray(globals()[f"y_val{col_names[empr][5:]}"]).astype('float32')

### prueba 1 ###
for empr in range(0,4):
    ffnn_x = globals()[f"x_train{col_names[empr][5:]}"][:,0:16]
    ffnn_y = globals()[f"y_train{col_names[empr][5:]}"]
    globals()[f"model_ffnn{col_names[empr][5:]}"] = Sequential()
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(16, input_dim=16, activation='relu'))
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(1))
    globals()[f"model_ffnn{col_names[empr][5:]}"].compile(loss='mean_squared_error', optimizer='adam')
    history = globals()[f"model_ffnn{col_names[empr][5:]}"].fit(ffnn_x[:,0:16],
                                                        ffnn_y,
                                                        epochs=200, batch_size=8, validation_split=0.25, verbose=1)
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    escriu = 'model loss ' + str(empr)
    plt.title(escriu)
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()
    plt.plot(history.history['loss'][100:])
    plt.plot(history.history['val_loss'][100:])
    escriu = 'model loss ' + str(empr)
    plt.title(escriu)
    plt.ylabel('loss')
```

```
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()

### prueba 2 ###
for empr in range(0,4):
    ffnn_x = globals()[f"x_train{col_names[empr][5:]}"][:,0:16]
    ffnn_y = globals()[f"y_train{col_names[empr][5:]}"]
    globals()[f"model_ffnn{col_names[empr][5:]}"] = Sequential()
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(16, input_dim=16, activation='relu'))
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(10, activation='relu'))
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(1))
    globals()[f"model_ffnn{col_names[empr][5:]}"].compile(loss='mean_squared_error', optimizer='adam')
    history = globals()[f"model_ffnn{col_names[empr][5:]}"].fit(ffnn_x[:,0:16],
                                                                ffnn_y,
                                                                epochs=200, batch_size=8, validation_split=0.25, verbose=1)
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    escriu = 'model loss ' + str(empr)
    plt.title(escriu)
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()
    plt.plot(history.history['loss'][100:])
    plt.plot(history.history['val_loss'][100:])
    escriu = 'model loss ' + str(empr)
    plt.title(escriu)
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()

### prueba 3 ###
for empr in range(0,4):
    ffnn_x = globals()[f"x_train{col_names[empr][5:]}"][:,0:16]
    ffnn_y = globals()[f"y_train{col_names[empr][5:]}"]
    globals()[f"model_ffnn{col_names[empr][5:]}"] = Sequential()
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(10, input_dim=16, activation='relu'))
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(5, activation='relu'))
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(1))
    globals()[f"model_ffnn{col_names[empr][5:]}"].compile(loss='mean_squared_error', optimizer='adam')
    history = globals()[f"model_ffnn{col_names[empr][5:]}"].fit(ffnn_x[:,0:16],
                                                                ffnn_y,
                                                                epochs=200, batch_size=8, validation_split=0.25, verbose=1)
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    escriu = 'model loss ' + str(empr)
    plt.title(escriu)
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()
    plt.plot(history.history['loss'][100:])
    plt.plot(history.history['val_loss'][100:])
    escriu = 'model loss ' + str(empr)
    plt.title(escriu)
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()

### prueba 4 ###
for empr in range(0,4):
    ffnn_x = globals()[f"x_train{col_names[empr][5:]}"][:,0:16]
    ffnn_y = globals()[f"y_train{col_names[empr][5:]}"]
    globals()[f"model_ffnn{col_names[empr][5:]}"] = Sequential()
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(16, input_dim=16, activation='relu'))
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(10, activation='relu'))
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(5, activation='relu'))
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(1))
    globals()[f"model_ffnn{col_names[empr][5:]}"].compile(loss='mean_squared_error', optimizer='adam')
    history = globals()[f"model_ffnn{col_names[empr][5:]}"].fit(ffnn_x[:,0:16],
                                                                ffnn_y,
                                                                epochs=200, batch_size=8, validation_split=0.25, verbose=1)
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    escriu = 'model loss ' + str(empr)
    plt.title(escriu)
```

```
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()
    plt.plot(history.history['loss'][100:])
    plt.plot(history.history['val_loss'][100:])
    escriu = 'model loss ' + str(empr)
    plt.title(escriu)
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()

### prueba 5 ###
for empr in range(0,4):
    ffnn_x = globals()[f"x_train{col_names[empr][5:]}"][:,0:16]
    ffnn_y = globals()[f"y_train{col_names[empr][5:]}"]
    globals()[f"model_ffnn{col_names[empr][5:]}"] = Sequential()
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(16, input_dim=16, activation='relu'))
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(15, activation='relu'))
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(10, activation='relu'))
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(1))
    globals()[f"model_ffnn{col_names[empr][5:]}"].compile(loss='mean_squared_error', optimizer='adam')
    history = globals()[f"model_ffnn{col_names[empr][5:]}"].fit(ffnn_x[:,0:16],
                                                                ffnn_y,
                                                                epochs=200, batch_size=8, validation_split=0.25, verbose=1)
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    escriu = 'model loss ' + str(empr)
    plt.title(escriu)
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()
    plt.plot(history.history['loss'][100:])
    plt.plot(history.history['val_loss'][100:])
    escriu = 'model loss ' + str(empr)
    plt.title(escriu)
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()

### prueba 6 ###
for empr in range(0,4):
    ffnn_x = globals()[f"x_train{col_names[empr][5:]}"][:,0:16]
    ffnn_y = globals()[f"y_train{col_names[empr][5:]}"]
    globals()[f"model_ffnn{col_names[empr][5:]}"] = Sequential()
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(16, input_dim=16, activation='relu'))
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(15, activation='relu'))
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(10, activation='relu'))
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(5, activation='relu'))
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(1))
    globals()[f"model_ffnn{col_names[empr][5:]}"].compile(loss='mean_squared_error', optimizer='adam')
    history = globals()[f"model_ffnn{col_names[empr][5:]}"].fit(ffnn_x[:,0:16],
                                                                ffnn_y,
                                                                epochs=200, batch_size=8, validation_split=0.25, verbose=1)
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    escriu = 'model loss ' + str(empr)
    plt.title(escriu)
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()
    plt.plot(history.history['loss'][100:])
    plt.plot(history.history['val_loss'][100:])
    escriu = 'model loss ' + str(empr)
    plt.title(escriu)
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()

### predicción ###
for empr in range(0,len(col_names)):
    ffnn_x = globals()[f"x_train{col_names[empr][5:]}"][:,0:16]
    ffnn_y = globals()[f"y_train{col_names[empr][5:]}"]
    globals()[f"model_ffnn{col_names[empr][5:]}"] = Sequential()
```

```
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(16, input_dim=16, activation='relu'))
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(10, activation='relu'))
    globals()[f"model_ffnn{col_names[empr][5:]}"].add(Dense(1))
    globals()[f"model_ffnn{col_names[empr][5:]}"].compile(loss='mean_squared_error', optimizer='adam')
    history = globals()[f"model_ffnn{col_names[empr][5:]}"].fit(ffnn_x[:,0:16],
                                                                ffnn_y,
                                                                epochs=200, batch_size=8, verbose=1)
    plt.plot(history.history['loss'])
    escriu = 'model loss ' + str(empr)
    plt.title(escriu)
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train'], loc='upper left')
    plt.show()
    plt.plot(history.history['loss'][100:])
    escriu = 'model loss ' + str(empr)
    plt.title(escriu)
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train'], loc='upper left')
    plt.show()
    pred_ffnn = []
    for i in range(0,len(globals()[f"x_val{col_names[empr][5:]}"])):
        ffnn_input = globals()[f"x_val{col_names[empr][5:]}"][i,0:16]
        ffnn_input = ffnn_input.reshape((1,16))
        ffnn_pred = globals()[f"model_ffnn{col_names[empr][5:]}"].predict(ffnn_input, verbose = 0)
        pred_ffnn.append(float(ffnn_pred))
    pred_ffnn = pd.DataFrame(pred_ffnn)
    globals()[f"diferencies_ffnn{col_names[empr][5:]}"] = pd.DataFrame(columns=["pred", "real"],
                                                                       index=globals()[f"df{col_names[empr][5:]}"].index[33445:])
    for i in range(0,len(globals()[f"diferencies_ffnn{col_names[empr][5:]}"])):
        globals()[f"diferencies_ffnn{col_names[empr][5:]}"]["pred"][i] = float(pred_ffnn.iloc[i])
        globals()[f"diferencies_ffnn{col_names[empr][5:]}"]["real"][i] = globals()[f"y_val{col_names[empr][5:]}"][i]
    escriu = 'diferencies ' + str(empr)
    globals()[f"diferencies_ffnn{col_names[empr][5:]}"][500:600].plot(figsize=(15,7),rot = 20, title = escriu)
    globals()[f"diferencies_ffnn{col_names[empr][5:]}"][4900:4950].plot(figsize=(15,7),rot = 20, title = escriu)
    globals()[f"diferencies_ffnn{col_names[empr][5:]}"][10000:10100].plot(figsize=(15,7),rot = 20, title = escriu)
    plt.show()

beneficio_real = 0
for empr in range(0,len(col_names)):
    globals()[f"diferencies_ffnn{col_names[empr][5:]}"]["real"] = (globals()[f"diferencies_ffnn{col_names[empr][5:]}"]["real"]/100)+1
    print(np.prod(globals()[f"diferencies_ffnn{col_names[empr][5:]}"]["real"]))
    beneficio_real = beneficio_real + np.prod(globals()[f"diferencies_ffnn{col_names[empr][5:]}"]["real"])
beneficio_real = beneficio_real/80
beneficio_real

beneficio_algoritmo = 1
for i in range(1,len(diferencies_ffnn_ATVI)):
    vector = [0]*80
    seguro = [0]*80
    deposito = []
    for empr in range(0,len(col_names)):
        vector[empr] = globals()[f"diferencies_ffnn{col_names[empr][5:]}"]["pred"][i]
    for j in range(0,10):
        if(vector[vector.index(max(vector))]>0):
            deposito.append(vector.index(max(vector)))
            vector[vector.index(max(vector))] = -999
    beneficio_iteracion = 0
    for j in range(0,len(deposito)):
        beneficio_iteracion = beneficio_iteracion + globals()[f"diferencies_ffnn{col_names[deposito[j]][5:]}"]["real"][i]
    if(beneficio_iteracion == 0):
        beneficio_iteracion = 1
    beneficio_algoritmo = beneficio_algoritmo * (beneficio_iteracion/max(len(deposito),1))
beneficio_algoritmo

beneficio_mantener = 0
for empr in range(0,len(col_names)):
    beneficio_mantener = beneficio_mantener + df[df.columns[empr*2]]["2021-04-23 15:18:00"]/df[df.columns[empr*2]]["2021-01-04 10:31:00"]
beneficio_mantener = beneficio_mantener/80
beneficio_mantener
```

Bibliografía

[1] Orçun Kaya. *High-frequency trading, Reaching the limits*. 2016. URL: https://www.dbresearch.com/PROD/RPS_EN-PROD/PROD0000000000454703/Research_Briefing%5C%3A_High-frequency_trading.pdf?undefined&realload=hMnAg9z_AuvGmUlUENO8/6fOu0O9I~W3OSoFIeX7YDNOZR9Hkttuxq9w-yL1bVKY2YK8tI/BXgTY_wXI0/FwaWVrw==.

[2] U.S. Securities y Exchange Commission. *Equity Market Structure Literature Review. Part II: High Frequency Trading*. 2014. URL: https://www.sec.gov/marketstructure/research/hft_lit_review_march_2014.pdf.

[3] Andy Haldane. *Patience and finance*. 2010. URL: https://www.bis.org/review/r100909e.pdf.

[4] Investorpedia. *Market Maker*. URL: https://www.investopedia.com/terms/m/marketmaker.asp.

[5] Investorpedia. *Arbitrage*. URL: https://www.investopedia.com/terms/a/arbitrage.asp.

[6] Darío Corral. *Una reflexión sobre Arbitraje Estadístico*. URL: https://www.rankia.com/blog/trading-para-dummies/2410766-reflexion-arbitraje-estadistico-julio-2014.

[7] Investorpedia. *Event-Driven Strategy*. URL: https://www.investopedia.com/terms/e/eventdriven.asp.

[8] von Beschwitz; Bastian; Donald B. Keim; y Massimo Massa. "First to "Read" the News: News Analytics and Algorithmic Trading". En: (2018). DOI: 10.17016/IFDP.2018.12330.

[9] Investorpedia. *Index arbitrage*. URL: https://www.investopedia.com/terms/i/indexarbitrage.asp.

[10] Investorpedia. *Basis trading*. URL: https://www.investopedia.com/terms/b/basis-trading.asp.

[11] Jonathan Spicer Roberta Rampton Rachelle Younglai. *"Quote stuffing" a focus in flash crash probe.* URL: https://www.reuters.com/article/us-sec-trades-idUSTRE6812ZS20100902.

[12] Market Conduct Rules. *Smoking.* URL: https://www.marketconductrules.com/risks/smoking.html#definition.

[13] Corporate Finance Institute. *Spoofing.* URL: https://corporatefinanceinstitute.com/resources/knowledge/trading-investing/spoofing/.

[14] Larry Harris. *Trading and Exchanges: Market Microstructure for Practitioners.* 2003.

[15] SEC. *Momentum Ignition.* URL: https://www.sec.gov/rules/concept/2010/34-61358fr.pdf.

[16] Price Waterhouse Coopers. *An objective look at high-frequency trading and dark pools.* URL: https://www.pwc.com/us/en/pwc-investor-resource-institute/publications/assets/pwc-high-frequency-trading-dark-pools.pdf.

[17] Chris Rose. *Dark Pools And Flash Orders: The Secret World Of Automated High-Frequency Trading.* URL: https://www.researchgate.net/publication/293022520_Dark_Pools_And_Flash_Orders_The_Secret_World_Of_Automated_High-Frequency_Trading.

[18] Gary Shorter Rena S. Miller. *High Frequency Trading: Overview of Recent Developments.* URL: https://fas.org/sgp/crs/misc/R44443.pdf.

[19] Ryan Riordan Sarah Zhang. *Technology and market quality: the casse of High Frequency Trading.* URL: https://core.ac.uk/download/pdf/301351513.pdf.

[20] Allen Carrion. *Very fast money: High-frequency trading on the NASDAQ.* URL: https://www.researchgate.net/publication/256029369_Very_fast_money_High-frequency_trading_on_the_NASDAQ.

[21] Dmitry Rakhlin Nataliya Bershova. *High-Frequency Trading and Long-Term Investors: A View from the Buy-Side.* URL: https://www.researchgate.net/publication/255855247_High-Frequency_Trading_and_Long-Term_Investors_A_View_from_the_Buy-Side.

[22] Allen Carrion. *Very fast money: High-frequency trading on the NASDAQ.* URL: https://www.researchgate.net/publication/256029369_Very_fast_money_High-frequency_trading_on_the_NASDAQ.

[23] Valeria Caivano. *The impact of high-frequency trading on volatility.* URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2573677.

[24] Frank Zhang. *High-Frequency Trading, Stock Volatility, and Price Discovery.* URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1691679.

[25] Austin Gerig. *High-Frequency Trading Synchronizes Prices in Financial Markets.* URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2173247.

[26] U.S. Commodity Futures Trading Commission, the U.S. Securities y Exchange Commission. *Findings regarding the market events of May 6, 2010.* URL: https://www.sec.gov/files/marketevents-report.pdf.

[27] CNMV. *MIFID II - MIFIR.* URL: http://cnmv.es/portal/MiFIDII_MiFIR/MapaMiFID.aspx.

[28] Miguel Sánchez Monjo y Ana Pineda Martínez. *La denominada negociación automatizada de alta frecuencia. Características y regulación.* URL: https://www.cuatrecasas.com/media_repository/docs/esp/la_denominada_negociacion_automatizada_de_alta_frecuencia_(high_frequency_trading)._revista_de_derecho_del_mercado_de_valores_n_12_2013_(enero-junio)._316.pdf.

[29] Wikipedia. *Empreas que conforman el NASDAQ-100.* URL: https://en.wikipedia.org/wiki/NASDAQ-100#Changes_in_2020.

[30] John J. Murphy. *Análisis técnico de los mercados financieros.* ISBN: 978-84-9875-428-5.