Trabajo Final de Máster

Indicador para operadores discrecionales en mercados financieros: Predicción basada en series temporales y redes neuronales

MÁSTER EN INGENIERÍA Y CIENCIA DE DATOS
CURSO: 2020-2021

SIST DE COMUNICACIÓN Y CONTROL - ETS INGENIERÍA INFORMÁTICA

Estudiante: Emilio Segarra Moliner
Tutor: Rafael Pastor Vargas

Septiembre 2021

Índice

Índice de figuras

1. Introducción

Cada día se producen sesiones bursátiles en gran cantidad de mercados, como pueden ser acciones, futuros o divisas. En cada una de ellas, se mueven millones de euros(o dólares) de una forma que intenta ser comprendida con diversos métodos para poder sacar provecho, bien sea a través de la inversión, de la especulación, o de cualquier otro enfoque que pueda derivar en un beneficio económico según el interés subyacente en cada caso.

Hace ya décadas que se empezaron a utilizar ordenadores con el avance de las telecomunicaciones para realizar transacciones financieras, por lo que estar físicamente en una ubicación determinada dejó de ser un requisito para poder realizar cualquier tipo de transacción.

Por contra, el rápido avance de las nuevas tecnologías ha producido un efecto inesperado y contrario al que motivó su movimiento inicial. Hoy en día, en algunos mercados más del 50 % de las transacciones que se realizan son llevadas a cabo por algoritmos que, trabajando a velocidades inalcanzables por el ser humano, se han apoderado por completo de este mundo haciendo necesario que cualquier persona que quiera participar tenga que adaptarse a las reglas impuestas por el Trading de Alta Frecuencia (HFT).

Es por ello, que cada día es más importante que aquellas personas que realizan una operativa discrecional, bien de forma individual o en grupos reducidos, dispongan de algunos recursos que les permitan de esta forma librar su batalla particular dentro de este complejo mundo que sigue avanzando día a día e incluyendo de forma rápida todos los avances tecnológicos.

Por suerte en unos casos, y por desgracia en otros, los algoritmos todavía no son capaces de abstraer la parte artística de este mercado, por lo que hay una gran cantidad de personas que consiguen ir adaptándose a las nuevas situaciones que van surgiendo y sacar provecho de esos escenarios con un plan de trabajo adecuado. En contraposición, puesto que los citados algoritmos son creados por personas, en colaboración con estudios a nivel psicológico del comportamiento de la psique humana, éstos pueden aunque no controlar generar movimientos que pueden derivar en pánico, euforia o cualquier otro sentimiento desatando un aluvión de operaciones incontroladas y fuera de todo plan de operativa que acaba, en gran cantidad de ocasiones, en un resultado calculado al milímetro que reporta beneficios a aquellas personas que lo articulan.

Sucesos excepcionales (buenos o malos), resultados económicos de empresas, normativas y un largo etc. son elementos que condicionan y que se perciben como positivos o negativos aunque luego siempre aparezca una justificación u otra.

Cualquier herramienta que permita a esas personas que operan de forma discrecional desde con sus modestos equipamientos, no ya contra avanzados sistemas informáticos con una capacidad computacional infinitamente superior, sino alinearse y detectar las intenciones de éstos son bienvenidos ya que permiten exprimir esa parte artística que es, al fin y al cabo, la única ventaja frente a los potentes algoritmos que funcionan hoy el día operando millones de cálculos y probabilidades por segundo.

2. Objetivos

Plantear de forma clara los objetivos que se persiguen en la realización del trabajo final de máster es una tarea nada fácil puesto que es un alcance. Los contenidos del máster son muy amplios y transversales por lo que definir y acotar algo en concreto tiene su punto de dificultad.

Por una parte se encuentra todo el contenido que se ha trabajado a lo largo del máster, dónde en el desarrollo del proyecto se encuentra contenido de las asignaturas:

- Modelado Estadístico de Datos
- Programación en Entornos de Datos
- Aprendizaje Automático I y II
- Infraestructuras Computacionales para procesamiento de datos masivos.
- Visualización de datos
- Gestión/Almacenamiento de información no estructurada.
- Deep Learning

En concreto, realizando este proyecto se aplican las competencias:

- Competencias básicas
    - CB6 - Poseer y comprender conocimientos que aporten una base u oportunidad de ser originales en el desarrollo y/o aplicación de ideas, a menudo en un contexto de investigación
    - CB7 - Que los estudiantes sepan aplicar los conocimientos adquiridos y su capacidad de resolución de problemas en entornos nuevos o poco conocidos dentro de contextos más amplios (o multidisciplinares) relacionados con su área de estudio
    - CB8 - Que los estudiantes sean capaces de integrar conocimientos y enfrentarse a la complejidad de formular juicios a partir de una información que, siendo incompleta o limitada, incluya reflexiones sobre las responsabilidades sociales y éticas vinculadas a la aplicación de sus conocimientos y juicios
    - CB9 - Que los estudiantes sepan comunicar sus conclusiones y los conocimientos y razones últimas que las sustentan a públicos especializados y no especializados de un modo claro y sin ambigüedades
    - CB10 - Que los estudiantes posean las habilidades de aprendizaje que les permitan continuar estudiando de un modo que habrá de ser en gran medida autodirigido o autónomo.
- Competencias Generales
    - CG1. Identificar los métodos apropiados para la solución de problemas asociados a la ciencia de datos y la analítica de información.

- CG2. Ser capaz de aplicar diferentes técnicas de aprendizaje máquina, seleccionando el algoritmo óptimo que genere modelos precisos y permita el desarrollo de soluciones predictivas en diferentes ámbitos de uso.
- CG3. Desarrollar sistemas de gestión/almacenamiento/procesamiento de grandes volúmenes de datos de una manera eficiente y segura, teniendo en cuenta las normativas/legislaciones existentes.
- CG4. Aplicar las técnicas de visualización sobre grandes volúmenes de datos para comunicar/presentar los resultados más relevantes del análisis de dicha información a diferentes roles de la organización.
- CG5. Utilizar las habilidades de científico de datos y/o ingeniero de datos en entornos de trabajo multidisciplinares y ser capaz de distinguir/organizar las diferentes actividades de los roles en dicho entorno.

- Competencias Específicas
  - CE1 - Conocer los fundamentos de la inferencia estadística y el análisis probabilístico y desarrollar diferentes tipos de modelos probabilísticos.
  - CE2 - Desarrollar aplicaciones/servicios/scripts orientados a la analítica de datos y analizar el uso de diferentes librerías para el desarrollo e implementación de métodos numéricos, algoritmos y modelos asociados a los datos.
  - CE3 - Diseñar entornos visuales y cuadros de mando en diferentes entornos computaciones, usando las técnicas de visualización de datos más eficientes.
  - CE4 - Conocer las diferentes librerías e implementaciones de las técnicas de representación de información, especialmente relevantes en la visualización de datos masivos.
  - CE5 - Desarrollar modelos de aprendizaje máquina (Machine Learning) basados en las diferentes categorías de clasificación: supervisada, no supervisada y semi-supervisada.
  - CE6 - Diseñar mecanismos de evaluación de modelos de aprendizaje y comprender las métricas usadas para dicha evaluación.
  - CE8 - Diseñar y utilizar modelos de gestión de datos masivos (Big Data) basados en herramientas disponibles sobre clústeres y en la nube.
  - CE10 - Describir las diferentes alternativas de almacenamiento de información estructurada y no estructurada, y los mecanismos disponibles para cada tipo de dato disponible.
  - CE11 - Conocer y comprender los fundamentos de las redes neuronales y sus variaciones, asi como las técnicas de implementación y desarrollo de este tipo de algoritmos de aprendizaje supervisado.

- Competencias Transversales
  - CT1 - Ser capaz de abordar y desarrollar proyectos innovadores en entornos científicos, tecnológicos y multidisciplinares.

- CT2 - Ser capaz de tomar decisiones y formular juicios basados en criterios objetivos (datos experimentales, científicos o de simulación disponibles).

El desarrollo se ha realizado de manera que todo el detalle de competencias sea fácilmente identificable en cada uno de sus apartados, en cualquiera de las categorías de competencias indicadas.

Por otra parte, la definición del proyecto intenta dar también un punto de vista con un enfoque más práctico. Existe en la actualidad una ingente cantidad de trabajos y publicaciones en ámbitos académicos y científicos pero que trabajan en gran medida el punto de vista matemático subyacente pero que, por otra parte, olvidan dar un punto de utilidad práctica para las personas que realizan una operativa discrecional. Se busca aportar ese valor añadido, de manera que el trabajo realiza un estudio de varias series temporales univariantes, centrándose no ya en la obtención de un único valor de cierre como es habitual, sino en la obtención de los posibles valores máximo y mínimo dentro un marco temporal concreto.

Para recoger toda la información posible, uno de las representaciones más utilizadas en cualquier mercado es la de las llamadas velas japonesas. A continuación se puede observar en la figura 1, el esquema de 2 posibles velas japonesas, cuyo cuerpo representa los valores contenidos entre la apertura y el cierre, el color indica si el movimiento ha sido alcista (verde) o bajista (rojo), cuyo significado en valores números simplemente traduce qué valor es mayor de los dos, y por último los tramos de las llamadas "mechas", que acaban en el valor máximo y mínimo respectivamente. Cada una de estas representación, recoge por una parte la totalidad de valores del activo durante el periodo de tiempo definido en el marco temporal (5 minutos en el caso de este trabajo), así como la estructura en la que queda organizado que recoge también una idea de la psicología subyacente en el mercado.

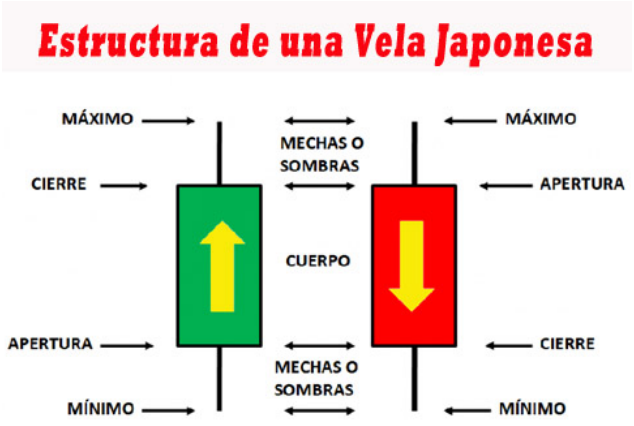


Figura 1: Representación de vela japonesa

Mediante este cálculo, se consigue dotar al proyecto final de un nuevo enfoque ya que no se centra en el valor único de cierre al final de la franja temporal en un periodo determinado, sino que permite abrir una nueva vía de operativa generando como resultado la citada estructura de vela, cuyos niveles de precio asociados(**O**pen, **H**igh, **L**ow, **C**lose), separan el problema en la realización de varias predicciones univariantes, una para cada una de sus componentes. Recalcar también la particularidad del estudio, que como pretende realizar la predicción de la vela en curso (1 step), en ese momento el valor del precio de apertura ya es conocido, por lo que no es necesario para su estudio.

Los resultados obtenidos, generan pues otro campo de estudio para la realización de predicciones no en base a un valor, sino a un posible comportamiento ya la operativa discrecional tiene en cuenta otros factores aparte del precio. Oferta, demanda, volatilidad son algunos conceptos que acompañan a los movimientos del mercado y detectar cualquier tipo de divergencia entre la predicción generada en base a unos datos históricos y el mercado real también aporta un valor añadido a la operativa.

3. Funcionamiento del Mercado de Futuros

El mercado de futuros es un mercado de subastas en el que los participantes compran y venden contratos estandarizados para su entrega en una fecha futura específica. Son productos derivados, no activos en sí, por lo que tienen una serie de particularidades que se traducen en ventajas o desventajas según la operativa.

Su origen se data entre los siglos XVII y XVIII y su objetivo no era (originalmente) otro que el de proteger a los productores de materias primas debido a la gran incertidumbre que se producía en elementos como cosechas. Así pues, se pactaba de antemano un precio y una cantidad a una fecha determinada que llegado el vencimiento se tenía que ejecutar. De esta manera el agente comprador presuponía que el precio podía subir en exceso y se garantizaba la adquisición dentro de unos límites asumibles. Por contra, si no se daban circunstancias que hacían encarece el precio en esa fecha, se producía una pérdida que se consideraba asumible ya que se estaba intentando reducir un riesgo.

Por otra parte, la parte vendedora también asumía riesgos ya que se podían producir fluctuaciones en otros elementos que podían aumentar sus costes de producción y por lo tanto ver su beneficio inicial mermado.

Todo este funcionamiento fue evolucionando creando organismos que regulaban el cumplimiento por ambas partes y garantizando de esta manera el correcto funcionamiento.

Con el funcionamiento más actual, encontramos su origen en 1848, que es cuando nació el Chicago Board of Trade (CBOT), que fue la primera bolsa de futuros de los Estados Unidos y actualmente la más importante del mundo. Se seleccionó esta sede debido a su situación geográfica, que facilitaba su funcionamiento original basado en la entrega de mercancías. Inicialmente se comercializaron productos como el maíz, trigo, semilla de soya, se siguió ampliando con otras materias primas como café, azúcar y ya con el paso del tiempo se fue diversificando e incluyendo otros productos que ya no tenían nada que ver, como activos financieros, minerales, divisas, etc.

Algunas de las Bolsas de Futuros más importantes son (CME Group, fusionadas en 2008):

- Chicago Mercantil Exchange (CME)
- Chicago Board od Trade (CBOT)
- Ney York Mercantil Exchange (NYMEX)
- Commodity Exchange Inc. (COMEX)
- Kansas City Board of Trade (KCBOT)

Existen muchas otras en todo el mundo como EUREX en Europa, pero no tienen tanta representación como las anteriores.

Así pues, cuando se desea operar con un tipo determinado de futuros, en primer lugar hay que localizar el mercado por el que está regulado, para tener la certeza que el broker con el que se van a contratar los servicios lo tiene en su catálogo como producto negociado.

La operativa del mercado de futuros es muy similar a la de cualquier otro activo, pero la diferencia principal radica en las garantías subyacentes y el volumen centralizado que permiten saber con exactitud qué cantidad de operaciones se están dando en todo el mundo. Por el contrario, otros productos derivados como contratos por diferencias, que también se

realizan sobre activos como en los futuros tales como índices, materias primas, etc, no se encuentran centralizados, y suelen provenir de creadores de mercado que no garantizan su funcionamiento equitativo en diferentes plataformas operativas.

Así pues, lo único que se precisa es tener claro sobre que activo se va a operar y con que plataforma se accede al mercado a través del broker, siendo en teoría independiente y menos afectado por ventajas tecnológicas. Se puede operar tanto a favor como en contra del mercado, y se puede acceder al igual que en cualquier otra bolsa mediante operaciones pasivas (definimos una operación a un precio determinado) o con operaciones activas(lanzamos una operación al precio que esté el mercado, se ejecutará segúna la oferta/demanda del momento).

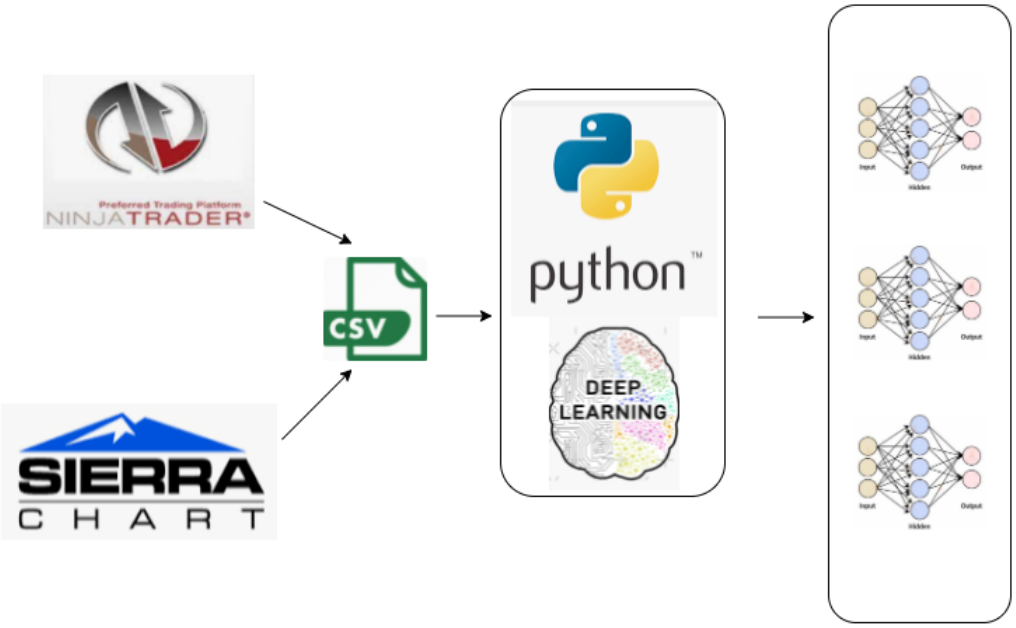
4. Big picture. Ciclo del proyecto

La realización de la totalidad del proyecto implica varias secuencias internas independientes. Cada una de ellas tiene una duración diferente y el proceso de retroalimentación también difiere de uno a otro, así pues encontramos la siguiente estructuración:

- **Creación de los modelos:** Este proceso se marca como revisable cada 6 meses, incorporando 1 (o varios) vencimiento(s) y aportando como información los últimos comportamientos del mercado. Se generará dentro de cada activo, un modelo para calcular el máximo en el periodo, otro para el mínimo y otro para el cierre. Como la predicción sólo va a realizarse a un paso, no es necesario el cálculo del valor de apertura puesto que éste se encuentra disponible en el momento de la predicción.

  - **Creación/Actualización del dataset:** Mediante el uso de las aplicaciones indicadas, o a partir de los datos almacenados en la base de datos externa, se obtienen los datos necesarios para generar los modelos.
  - **Exploración:** Se realizan las funciones típicas de revisión de corrección de datos.
  - **Generación de modelos:** Se vuelven a generar los modelos (máximo, mínimo y cierre) con cada uno de los algoritmos planteados.
  - **Comparación de modelos:** Se comparan y valoran los resultados obtenidos en los distintos estudios realizados, dejando constancia de los puntos fuertes de cada uno de ellos.

  Esta primera parte tendría pues la siguiente estructura, detallada ya con sus componentes.



- **Preparación del entorno de producción:** Se realizan los pasos necesarios para una vez finalizada la fase anterior, se pueda preparar la infraestructura que en la fase

posterior permita explotar los modelos en los distintos activos y sean aprovechables ya para la persona que quiera hacer uso de ello.

- **Volcado de históricos a sistema de gestión de bases de datos.** Se realiza el volcado del conjunto de datos utilizado a un sistema de bases de datos para su explotación en iteraciones posteriores del ciclo de vida.
- **Volcado de nuevos datos a medida que se generan.** Se desarrollan también las herramientas necesarias para que los datos del mercado estén volcados en la citada base de datos. De esta forma se podrán consultar también por parte de la herramienta final. En este caso, se trata de desarrollar un indicador que exporte los valores de cada vela una vez se cierre para que dichos valores sean almacenados para su explotación por parte de la interfaz de usuario final. Es necesario pues, disponer de un software en funcionamiento en tiempo real que se encargue de alimentar la base de datos con los nuevos valores.

- **Interfaz principal:** La idea es realizar una aplicación que permita de una forma fácil e intuitiva la explotación de todos los pasos realizados con anterioridad. Así pues, de una forma rápida se podrán consultar los diferentes predicciones realizadas por los algoritmos. Esta solución conectará de forma independiente con el sistema de gestión de bases de datos anterior para obtener los valores de las que serán la variables explicativas del sistema y generar la nueva predicción.

Esta última parte de la estructura, más detallada en sus respectivos apartados, se podría representar en la siguiente figura (2):

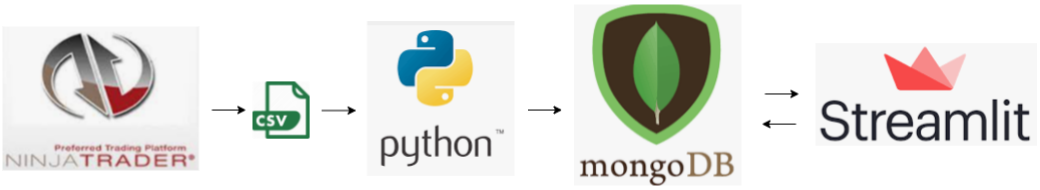


Figura 2: Flujo de la aplicación

5. Fase inicial de estudio

Debido a la transversalidad del proyecto, es necesario realizar una mínima introducción a algunos de los elementos que se encuentran en su ciclo de vida. Un apartado muy importante del punto indicado, es la plataforma de trading sobre la que se trabaja. Básicamente, es la herramienta que se utiliza para acceder a los datos.Dada la gran cantidad de alternativas que existen, se ha optado por la selección de 2. Éstas son algunas de las más extendidas a nivel internacional en lo que se refiere a trading discrecional. Por otra parte, también incluyen gran cantidad de funcionalidades que las hacen aptas para su funcionamiento.

Otro de los motivos a tener en cuenta, aparte de su funcionalidad, es que faciliten en gran medida la obtención de los datos del mercado. Disponer de esta información del mercado de futuros a través de una API u otro recurso tiene un coste económico muy elevado debido al gran beneficio en el que pueden repercutir, por lo que en este caso, disponer de las herramientas que en modalidad de demostración ofrecen una funcionalidad total durante un periodo limitado a coste 0 es un factor determinante para su elección.

5.1. Plataformas software: Sierrachart y Ninjatrader 8

La primera de ellas, Sierrachart, cuya página web puede verse en la figura 3 (`http://www.sierrachart.com`), es una plataforma que apuesta más por su funcionalidad y veteranía dejando de lado aspectos más comerciales. Tiene una gran comunidad detrás en muchos idiomas y una gran flexibilidad, de manera que fácilmente se pueden añadir funcionalidades adicionales, tales como indicadores o estrategias. Como base de programación para tal fin utiliza C++, lo que garantiza que la velocidad de ejecución está optimizada.

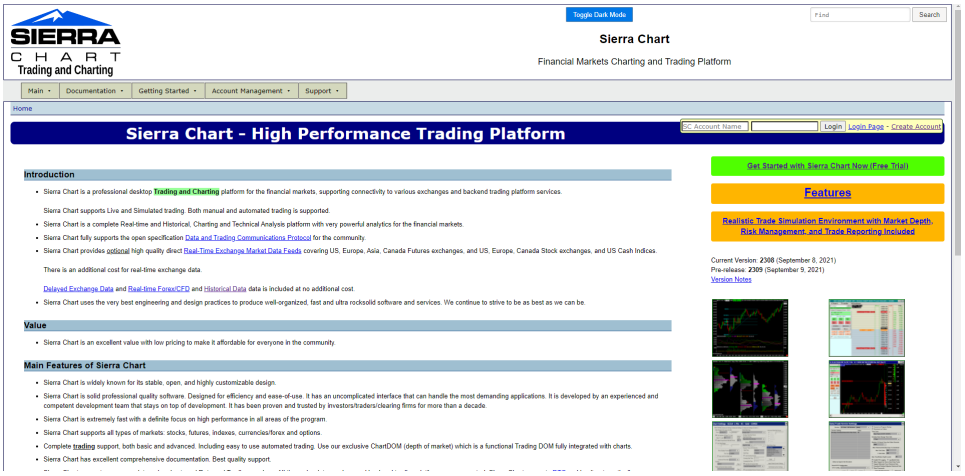


Figura 3: Página web de SierraChart

Además de la funcionalidad básica indicada, todo tipo de herramientas son susceptibles de ser aprovechadas debido a, tal y como se ha indicado, su extensa comunidad a nivel internacional. Diferentes tecnologías o indicadores son rápidamente adaptados para que cualquier tipo de operativa esté representada. Así pues, no solo realizan la representación de las velas japonesas y las funciones de operativa, se pueden incluir otro tipo de cálculos y representaciones como el que se puede observar en 4, que muestra la oferta y la demanda en este preciso

instante (derecha) y cómo ha ido evolucionando con el paso del tiempo en los distintos niveles de precio.

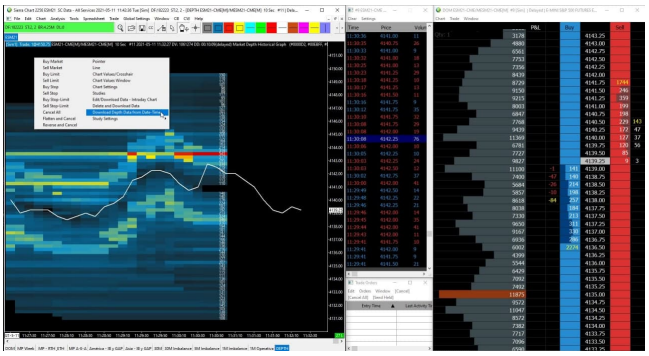


Figura 4: Funcionalidades adicionales de SierraChart

Por otra parte también es una herramienta de amplio uso Ninjatrader 8 (`www.ninjatrader.com`, figura 5). Además de plataforma en este caso realiza la función de broker de acceso a los mercados y gestión de cuentas, por lo que es mucho más sencillo su funcionamiento.

Figura 5: Página web de Ninjatrader

Esta plataforma aporta como la anterior una gran flexibilidad lo que la hace más potente. Además, como lenguaje de expansión tiene una versión reducida de C# llamada Ninjascript, que permite con una pequeña curva de aprendizaje desarrollar nuevas características. Visualmente está más elaborada que la anterior pero como todo tiene sus pros y sus contras según las necesidades de cada persona.

5.2. Generación del conjunto de datos

Para la generación del conjunto de datos se ha realizado un estudio de las opciones proporcionadas para tal fin por parte de cada una de las herramientas anteriormente citadas. Ambas proporcionan en una versión demo toda su funcionalidad por lo que son totalmente aptas para dicho cometido. Así pues, se ha procedido a la generación de un histórico de unos 2 años de antigüedad para poder garantizar que se dispone de los suficientes datos.

Por una parte se ha realizado con Sierrachart. La aplicación dispone de una gran versatilidad en este aspecto ya que ha permitido generar una representación de los activos de forma continua de los últimos 2 años, adaptando los datos de los vencimientos de forma correcta y adecuada y todo ello de una forma rápida.

Por contra, la herramienta Ninjatrader 8 no ofrece dicha ventaja y se ha tenido que acceder a un servicio de bajo coste para obtener todos esos datos históricos. En este caso se descargaron los mismos datos, pero fue necesario en primera instancia crear un indicador que exportara los datos ya que se trata de un formato propietario y el acceso directo implicaría un incumplimiento de la legislación, y, en segunda instancia, crear el código en Python necesario para poder realizar la unión de todos esos datos exportados en un único fichero.

Además de todo el trabajo adicional requerido, se ha podido observar gran cantidad de ausencia de datos, hecho que puede ser motivado por la mayor cantidad de manipulación exigida por parte de la segunda herramienta. Por contra, además de la facilidad, tal y como se observará en en el apartado de análisis exploratorio de los datos, no hay valores vacíos ni en variables explicativas a nivel individual, ni en huecos creados en la serie temporal. Este último punto es el que ha permitido seleccionar de forma unívoca la solución a utilizar ya que la citada ausencia de datos deriva en dificultades adicionales que a su vez empeoran los resultados puesto los datos inexistentes han de ser concretados de alguna forma.

La siguiente figura (6) muestra la comparativa de datos generada a partir de los cálculos extraídos:

Plataforma	Registros Obtenidos	% Correctos	% Serie temporal correcta
Ninjatrader 8	124.588	100 %	81 %
Sierrachart	153.813	100 %	100 %

Figura 6: Tabla comparativa de porcentajes de plataformas

Con todo lo anterior, y, además del menor coste exigido para obtener los datos, no ya económico sino de exigencia de recursos, se selecciona la herramienta Sierrachart para exportar la información y generar los conjuntos de datos.

5.3. Selección de activos y marco temporal

La selección de activos en este trabajo no es una de las decisiones críticas puesto que tal y como se ha indicado, cualquier producto regulado por el mercado de futuros garantiza una transparencia y corrección a la hora de acceder al mercado que lo hacen mucho más atractivo que otros como los contratos por diferencias, dónde es cada broker comercial el que se encarga de replicar el precio del mercado real en un entorno propio. Se accede a una operativa regulada y centralizada, por lo que en todo momento se puede determinar si es el momento oportuno para operar en estrategias basadas en volumen.

Se trata de un procedimiento totalmente iterativo y, el proceso de generación seguido con el primer activo será repetible en otros. Así pues, inicialmente y dentro de las opciones ha optado inicialmente por trabajar con el futuro sobre el índice del Nasdaq (ticker NQ), con vencimientos trimestrales, con un volumen de operativa considerable y una volatilidad relativa, que lo hacen atractivo para operativas discrecionales basadas en volumen que suele ser el alto porcentaje de ellas.

La selección del marco temporal ya exige de una visión un poco más afinada, puesto que según la que se tome estarán disponibles más o menos registros para la creación de los modelos. Cabe mencionar que cada franja temporal se refiere con la terminología OHLC (Open High Low Close) y representa mediante esos 4 valores el conjunto de la actividad desarrollada en ese espacio de tiempo con la estructura de vela japonesa.

Tomando como valor de marco temporal 5 minutos, se dispone de un conjunto de datos de más de 150.000 registros para estos 2 años, lo que lo convierte en un conjunto de datos adecuado y que se adapta además a un timing correcto para operativa intradía sin sobresaltos. De esta forma, cada una de las predicciones tendrá una validez máxima de 5 minutos, con lo que se puede ir valorando de forma visual el funcionamiento a lo largo del día sin necesidad de aumentos de estrés innecesarios. Se toma pues como activo y marco temporal de referencia **NQ 5m**.

6. Exploración y limpieza de datos

Para la realización de este apartado, se han seguido las pautas estándar de estudio y transformación de datos realizadas en este tipo de trabajos. Inicialmente se realiza una inspección inicial de los datos importados, para proceder con la sucesiva simplificación para hacerlo adecuado al trabajo deseado. Así pues, el punto de partida sería la siguiente tabla (figura 7).

```
1 df.head(5)
```

	Date	Time	Open	High	Low	Last	Volume	# of Trades	OHLC Avg	HLC Avg	HL Avg	Bid Volume	Ask Volume	Volume.1
0	2019/6/19	20:30:00.0	7701.75	7701.75	7699.50	7701.25	371	214	7701.06	7700.83	7700.63	141	230	371
1	2019/6/19	20:35:00.0	7701.25	7701.75	7700.00	7700.25	91	75	7700.81	7700.67	7700.88	45	46	91
2	2019/6/19	20:40:00.0	7700.25	7700.75	7698.00	7698.00	149	122	7699.25	7698.92	7699.38	93	56	149
3	2019/6/19	20:45:00.0	7698.25	7698.75	7696.25	7698.50	168	150	7697.94	7697.83	7697.50	102	66	168
4	2019/6/19	20:50:00.0	7698.75	7699.50	7697.25	7698.50	139	135	7698.50	7698.42	7698.38	92	47	139

Figura 7: Conjunto de datos inicial

Del detalle anterior, directamente se puede observar que hay mucha información sobrante que no va a aportar información a nivel lógico al estudio. Como elementos a tener en cuenta, simplemente sería necesario quedarse con las siguientes variables: *Date, Time, Open, High, Low, Close, Volume*. Se deduce que se refieren a fecha y hora del intervalo, el valor de apertura, el máximo que se alcanza y el mínimo que se alcanzan, el cierre y el volumen, que se refiere al total de contratos que se han operado en toda esa franja. Un valor alto de volumen indica que hay mucha actividad en ese nivel de precio, y que, por lo tanto, es un precio atractivo para operar. Normalmente los valores altos indican que hay profesionales trabajando en esos niveles.

Una vez eliminadas las columnas restantes, la estructura sería (figura 8):

	Date	Time	Open	High	Low	Close	Volume
0	2019/6/19	20:30:00.0	7701.75	7701.75	7699.50	7701.25	371
1	2019/6/19	20:35:00.0	7701.25	7701.75	7700.00	7700.25	91
2	2019/6/19	20:40:00.0	7700.25	7700.75	7698.00	7698.00	149
3	2019/6/19	20:45:00.0	7698.25	7698.75	7696.25	7698.50	168
4	2019/6/19	20:50:00.0	7698.75	7699.50	7697.25	7698.50	139

Figura 8: Conjunto de datos modificado

Para finalizar con su estructuración como una Serie Temporal, se unifican los distintos campos relativos a fecha y hora en un único y se podría ya proceder al estudio detallado, quedan la estructura reflejada en la figura 9):

Profundizando ya en la estructura disponible, se observa que se carece de variables categóricas como ya se podía deducir de la estructura anterior. No se observan valores nulos y queda, por lo tanto, un esquema bastante simplificado respecto al punto inicial.

Fecha	Open	High	Low	Close	Volume
2019-06-19 20:30:00	7701.75	7701.75	7699.50	7701.25	371
2019-06-19 20:35:00	7701.25	7701.75	7700.00	7700.25	91
2019-06-19 20:40:00	7700.25	7700.75	7698.00	7698.00	149
2019-06-19 20:45:00	7698.25	7698.75	7696.25	7698.50	168
2019-06-19 20:50:00	7698.75	7699.50	7697.25	7698.50	139

Figura 9: Conjunto de datos modificado con índice unificado

La información relativa al dataset sería:

```
DatetimeIndex: 153813 entries, 2019-06-19 20:30:00 to 2021-08-27 17:55:00
Data columns (total 5 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Open    153813 non-null  float64
 1   High    153813 non-null  float64
 2   Low     153813 non-null  float64
 3   Close   153813 non-null  float64
 4   Volume  153813 non-null  int64
dtypes: float64(4), int64(1)
```

Por lo que el resumen de valores nulos quedaría:

```
Open      0
High      0
Low       0
Close     0
Volume    0
dtype: int64
```

Cabe recordar que el enfoque del trabajo no está orientado a la estimación de un valor único, sino al conjunto de valores OHLC pertenecientes a un intervalo de tiempo (de 5 minutos en este caso), por lo que aunque muy parecidos al ir ligados a lo largo del tiempo, hay que considerarlos como elementos independientes.

Si se procede a representar los valores de las variables independientes y dependientes tenemos que el gráfico es prácticamente el mismo en los casos de las componentes, puesto que los valores van muy ligados en su comportamiento a lo largo del tiempo. La representación de los valores máximo y mínimo sería pues (figura 10):

Se confirma, que el comportamiento de las series es prácticamente el mismo y que difiere muy poco. Este suceso se repite con mayor claridad a medida que se utiliza un marco temporal más bajo, y en activos con muy poca volatilidad. Para observar diferencias sería necesario subir considerablemente la división temporal e incluso cambiar a otro activo como el Petróleo (Tick CL) mostraría más diferencias.

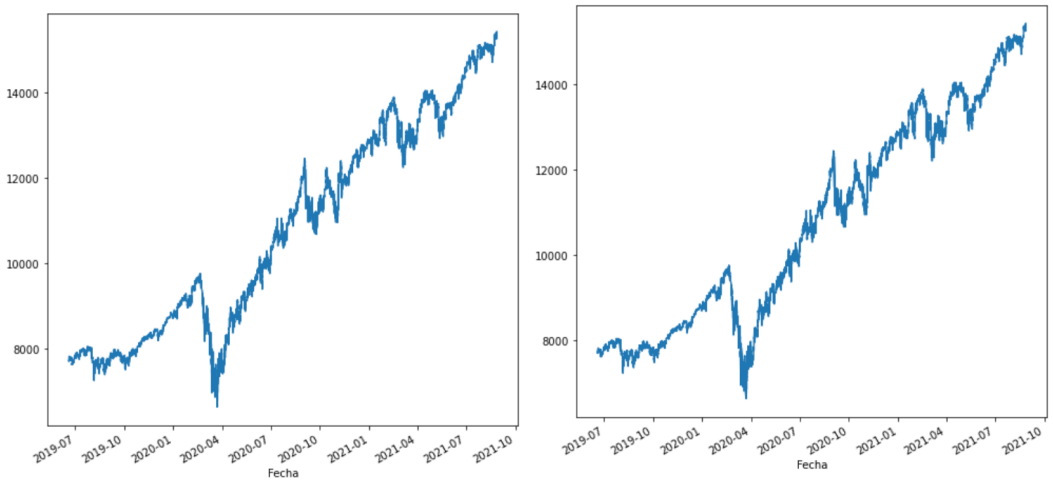


Figura 10: Representación de series temporales máximo y mínimo

Este comportamiento recientemente comentado, se puede observar si observamos la correlación entre variables. A continuación se muestra la tabla en cuestión (Fig 11):

	Open	High	Low	Close	Volume
Open	1.000000	0.999995	0.999994	0.999992	0.008280
High	0.999995	1.000000	0.999989	0.999995	0.009727
Low	0.999994	0.999989	1.000000	0.999995	0.006630
Close	0.999992	0.999995	0.999995	1.000000	0.008171
Volume	0.008280	0.009727	0.006630	0.008171	1.000000

Figura 11: Representación de matriz de correlación de variables

Observando la matriz anterior se puede confirmar la argumentación citada, con unas variables muy correlacionadas por una parte, y otra, totalmente independiente por la otra. Se recalca que es un comportamiento esperado y que sólo modificando esos parámetros (marco temporal más amplio y activo más volátil) se conseguiría rebajar ligeramente esos valores.

Con todas las características y comportamientos estudiados, se puede confirmar que el caso a desarrollar se compone de varios casos univariantes e independientes, uno por cada característica del dataset *High, Low, Close* cuya combinación proporcionará el resultado de la estimación OHLC para un periodo de tiempo definido y determinado. Volver a remarcar que el valor *Open* ya estará disponible en el momento de la solicitud de la estimación.

7. Estudio de las series temporales.

Tal y como se recoge en la definición de Serie Temporal, disponemos de un conjunto de datos que tienen la particularidad de que su ocurrencia está ordenada en el tiempo y a intervalos iguales. Es por ello, que se requiere de un tratamiento especial respecto al manejo estándar que se realiza con otros algoritmos de machine o deep learning.

El análisis de series temporales en el mundo financiero tiene ya una larga trayectoria a lo largo del tiempo, tanto por su posible beneficio económico, como por la naturaleza que subyace detrás del movimiento de los mercados. Resultados de elecciones, crisis políticas y un larga lista de situaciones pueden derivar en un cambio brusco que afecta a la economía mundial en cuestión de minutos.

Por otra parte también es importante remarcar que hay que realizar una serie de puntos para garantizar que es necesario el tratamiento de la información como una secuencia de sucesos en el tiempo. En este caso, tal y como se ha indicado es obvio que sí que requiere dicho estudio, ya que no se trata de obtener un valor constante o calculable a partir de una función definida y/o cíclica.

Para poder trabajar con la serie temporal, tenemos que descomponerla en una serie de elementos básicos como son:

- **Tendencia.** Movimiento al alza o a la baja del dato a lo largo de un periodo de tiempo considerable.
- **Estacionalidad.** Si hay algún hecho determinante por el periodo del año, como por ejemplo venta de productos típicos de una época del año.
- **Ruido o irregularidades.** Valores atípicos muy altos o bajos que se dan de forma aleatoria.
- **Repetición Cíclica**. Si hay algún tipo de repetición a lo largo de algún tipo de periodo.

Por otra parte, también hay que garantizar que cumple una serie de requisitos (Stationary):

1. Debería tener una media constante.
2. Varianza o desviación típica constante.
3. Auto-covarianza no debería depender del tiempo

Para comprobar este hecho, que la serie temporal es estacionaria, se hace uso del **Augmented Dickey Fuller Test**, ADF, que mediante el uso de hipótesis determina el cumplimiento o no de las condiciones. La hipótesis nula refiere a que la series es No-Stationality y la alternativa que la serie cumple la estacionalidad.

De la representación gráfica del apartado anterior, en el que se ha realizado la exploración de datos, se observa que hay una alta correlación entre los valores asociados al precio en sí por cada instante de tiempo. Así pues, el estudio de las distintas variables (Open, High, Low, Close) será muy similar.

Ejecutamos el código necesario para realizar la comprobación, obteniendo el siguiente resultado:

```
result = adfuller(df["High"].values)

pvalue = result[1]
if pvalue < 0.05:
  print('stationary')
else:
  print('not stationary')
```

El código anterior, una vez ejecutado, genera la siguiente salida:

```
(-0.04672653624059898, 0.954492978513328, 69, 153743,
{'1%': -3.4303925346792083,
'5%': -2.8615587997342393,
'10%': -2.5667800064280697}, 1089247.1014451724)
not stationary
```

De la salida anterior, hay que detallar que en primer lugar tenemos el valor del estadístico, el segundo valor(0,954492....) representa la probabilidad de que la hipótesis nula no sea rechazada(p-value). El tercer elemento (69) representa el número de puntos retrasados en el tiempo para determinar el t-statistic. El cuarto valor el número de observaciones de la muestra y el quinto valor los valores T correspondientes al test de Adfuller.

Realmente simplemente con el visualizado del gráfico ya se observa la tendencia así como la varianza no constante que permitiría concluir de forma directa dicha afirmación. En lo referente a la auto-correlación de la variable a lo largo del tiempo, encontramos la representación de la figura (12), del que se recalca la autocorrelación parcial de nivel 1.

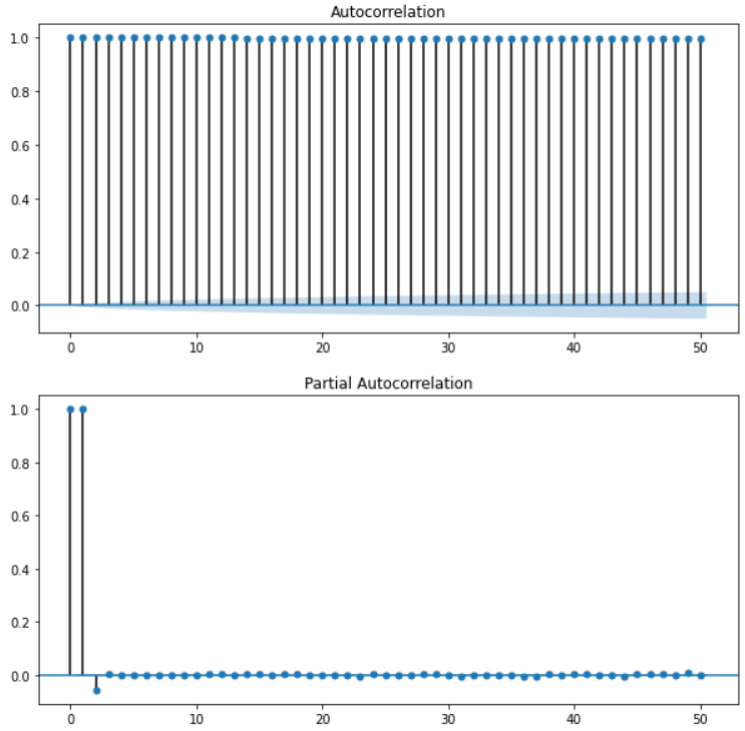


Figura 12: Autocorrelación de variable (igual en las 3) respecto al tiempo

Si se realiza la descomposición de la serie temporal en sus componentes básicas, se obtiene la representación de la figura 13:

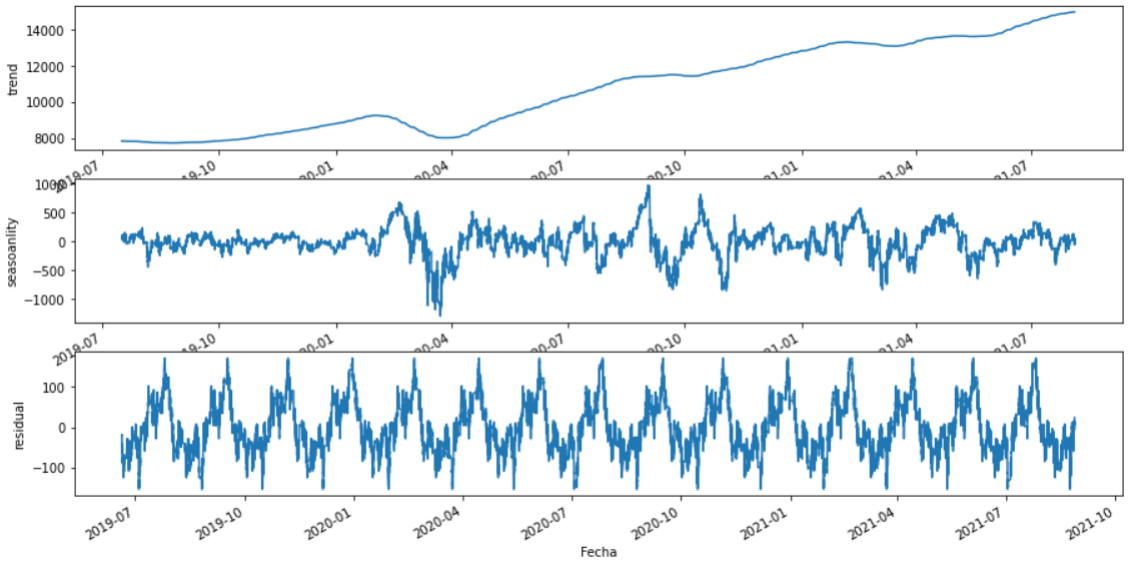


Figura 13: Descomposición de la serie High en sus componentes básicas(Trend, Seasonality, Residual

8. Generación de modelos

8.1. Creación y valoración de las distintas opciones

La realización de pronósticos por medio de redes neuronales tiene un enfoque sustancialmente diferente respecto a las técnicas clásicas que se centran más en el estudio de la serie temporal y su naturaleza. El citado enfoque clásico, requiere de una serie de transformaciones y comprobaciones realizadas de forma manual hasta que se permita su utilización. Dichas comprobaciones van desde la existencia de tendencia, al cumplimiento de las comprobaciones mediante contrastes de hipótesis. En cambio, la idea básica de las redes neuronales es la de la transformación de la serie temporal (como ente secuencial) en un caso de aprendizaje supervisado, de manera que cada elemento de la serie aparece tanto como posible valor de salida, como de variable explicativa.

Así pues, se busca generar un escenario dónde cada valor de la serie temporal forma parte de un conjunto de pasos requiere una consideración especial ya que se puede considerar un problema en el que los valores están inicialmente en una dimensión y pasan a un espacio bidimensional. Además de ello, también hay que hacer hincapié en que según la estructura de red neuronal utilizada, puede requerirse alguna dimensión adicional si cabe. Se busca dar un esquema donde se pueda determinar $y = f(x)$, donde x es un valor requerido para realizar la predicción (y). Como ejemplo sencillo para facilitar la compresión, de transformación de una serie temporal sencilla de una única variable, se podría considerar el siguiente:

```
Serie temporal original: [1, 2, 3, 4, 5, 6]
```

Conjunto de datos generado para aprendizaje supervisado:

```
X,           y
[1, 2, 3], [4]
[2, 3, 4], [5]
[3, 4, 5], [6]
```

Para completar todos los apartados necesarios pues, se procederá a realizar la citada transformación y estudio para cada uno de los campo *High, Low, Close* que junto con el valor conocido *Open* formarán el conjunto de la predicción. Para la realización de la citada conversión, se define como parámetro *steps* el nivel 3, de manera que la representación anterior reproduce de forma fidedigna la transformación realizada. La consideración de dicho valor viene justificada por el comportamiento del activo, ya que 3 valores en el tiempo equivalen en el caso que se está estudiando a 15 minutos, tiempo que se considera más que suficiente para plantear que empiezan a dejar de influir los valores en el momento de realizar el pronóstico. En cualquier caso, queda como pendiente la realización del estudio de cómo cambia el comportamiento también según las exigencias de la operativa, si se trata de una operativa más pausada aunque intradiaria o si por el contrario se busca al más tipo *swing* o a largo plazo. Se mantiene el valor en todos los modelos pese a que por su naturaleza, según el valor de este ajuste se aprovechan más o menos las características propias de cada método, por lo que cambiar dicho valor puede derivar en un comportamiento y resultados totalmente distintos.

El código en común que ejecutarán todos los modelos, será el siguiente, ya que se encargará de descargar los datos y realizar las transformaciones iniciales básicas, tales como el preprocesado, limpieza de datos, etc.

```
n_steps = 3
activo="NQ"
timeframe="5m"

ubicacion="https://github.com/emiliosegarra/tfm/raw/main/"
ruta=ubicacion+activo+timeframe+".txt"

df = pd.read_csv(ruta)
borrar=[' # of Trades', ' OHLC Avg', ' HLC Avg', ' HL Avg', ' Bid Volume',' Ask Volume',
    ' Volume.1']
df=df.drop(borrar,axis=1)
columnas=['Date','Time','Open','High','Low','Close','Volume']
df.columns = columnas
df['Fecha'] = pd.to_datetime(df['Date']+df['Time'])
df=df.drop(['Date','Time'],axis=1)
df=df.set_index("Fecha")

# Realizamos la transformación de la serie temporal en muestras
def split_sequence(sequence, n_steps):
    X, y = list(), list()
    for i in range(len(sequence)):
        end_ix = i + n_steps

        if end_ix > len(sequence)-1:
            break
        # Parte la secuencia en componentes
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
        X.append(seq_x)
        y.append(seq_y)
    return array(X), array(y)
```

En el Anexo C está el código fuente completo de la generación de los modelos, con sus cálculos y representaciones si se desea comprobar el funcionamiento.

8.2. Multilayer Perceptron (MLP)

El modelo Perceptron Multicapa o MLP es la evolución del modelo básico de perceptrón. Su característica principal es su sencillez dado que su origen viene derivado de la necesidad de resolver problemas que no son linealmente separables. Dada su sencillez, suele utilizarse constantemente como marco de referencia para otros modelos más elaborados.

El planteamiento realizado es el de comparar esquemas sencillos de modelos e intentar ya en un futuro, dotarlos de la complejidad adicional necesaria para ir obteniendo mejores resultados. El patrón genérico utilizado para la creación de los modelos de cada una de las variables explicativas es:

```
model = Sequential()
model.add(Dense(100, activation='relu', input_dim=n_steps))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
```

Como se ha indicado se trata de un modelo muy sencillo para iniciar el proceso de cálculo y, a su vez, valorar la viabilidad de incrementar recursos en la obtención de mejores resultados dada la dificultad de realizar cualquier tipo de pronóstico por la gran cantidad de factores que influyen dada su naturaleza.

El esquema de la red neuronal inicial generada sería pues:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 100)               400
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 101
=================================================================
Total params: 501
Trainable params: 501
Non-trainable params: 0
```

El código específico generado para la creación de los modelos se encuentra en el Anexo C tal y como se ha indicado,

Tras realizar el proceso de entrenamiento y predicción, se obtienen los siguientes resultados. Se ha utilizado una *callback* para limitar la ejecución del proceso, cuya salida es:

```
====================
Calculando MLP-High
====================
Epoch 1/100
1750/1750 [==============================] - 3s 1ms/step - loss: 1213794.2500 - mse: 1213794.2500
- mae: 165.6570 - val_loss: 156.0508 - val_mse: 156.0508 - val_mae: 7.9677
Epoch 2/100
1750/1750 [==============================] - 2s 1ms/step - loss: 121.0493 - mse: 121.0493 -
mae: 7.0252 - val_loss: 155.5802 - val_mse: 155.5802 - val_mae: 7.9472
...
====================
Calculando MLP-Low
====================
Epoch 1/100
1750/1750 [==============================] - 3s 1ms/step - loss: 997531.2500 - mse: 997531.2500
- mae: 150.3201 - val_loss: 209.9902 - val_mse: 209.9902 - val_mae: 9.3119
Epoch 2/100
1750/1750 [==============================] - 2s 1ms/step - loss: 170.3155 - mse: 170.3155 -
mae: 8.3257 - val_loss: 207.3852 - val_mse: 207.3852 - val_mae: 9.2801
...
====================
Calculando MLP-Close
====================
Epoch 1/100
1750/1750 [==============================] - 2s 1ms/step - loss: 119290.4609 - mse: 119290.4609
- mae: 48.4980 - val_loss: 207.1844 - val_mse: 207.1844 - val_mae: 9.9372
Epoch 2/100
1750/1750 [==============================] - 2s 1ms/step - loss: 148.7266 - mse: 148.7266 -
mae: 7.9805 - val_loss: 218.7506 - val_mse: 218.7506 - val_mae: 10.4757
...
```

Tras la ejecución del proceso para las 3 series temporales *High, Low,Close* se obtienen los siguientes resultados respecto a la métrica **mse** que se ha tomado como referencia para comparar todos los modelos obtenidos:

	High	Low	Close
MSE MLP	62.364526383969924	86.46579384821796	111.56853684133364

Tal y como se recoge en apartados anteriores, este resultado basado en una propuesta sencilla y rápida sirve de referencia respecto a los otros modelos en teoría más enfocados al trabajo con series temporales. Si se analizan los gráficos de evolución del error de entrenamiento y validación se observa claramente que no hay una mejora por más iteraciones que se realicen.

Aunque hay diferencias en el gráfico, el comportamiento de las 3 series temporales ha de ser muy similar por la naturaleza del mercado, por lo que en este caso se cumple el "malçomportamiento en las 3 tal y como se puede observar en los gráficos:

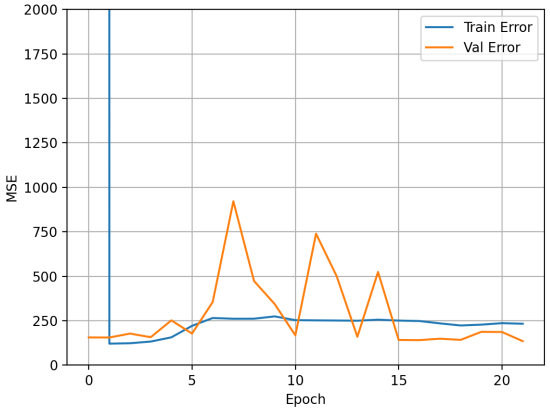


Figura 14: Comparación error de entrenamiento y validación serie temporal High MLP

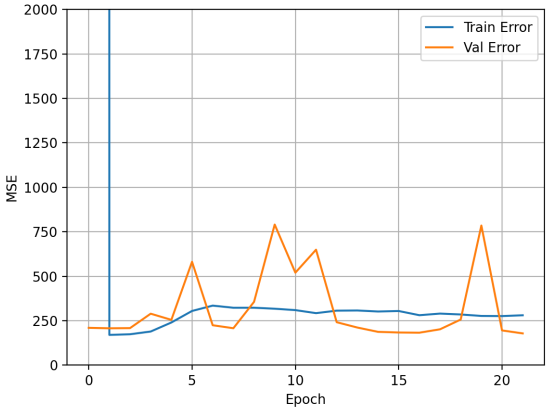


Figura 15: Comparación error de entrenamiento y validación serie temporal Low MLP

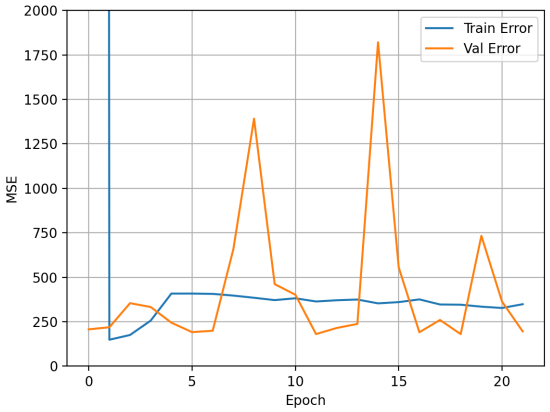


Figura 16: Comparación error de entrenamiento y validación serie temporal Close MLP

Por otra parte, se representa también un segmento de 100 predicciones realizadas sobre las distintas series temporales. Cabe recalcar en esta situación, que los valores obtenidos contienen la incertidumbre intrínseca de los mercados, por lo que intentar aproximar valores a un punto más que a unos niveles que contienen varios puntos harían inviable cualquier estudio. En estas representaciones, el eje de las Y contiene el valor del precio asociado al activo para ese instante del tiempo.

Se realiza también la representación para las 3 series temporales en el orden realizado desde el principio *High,Low,Close*.

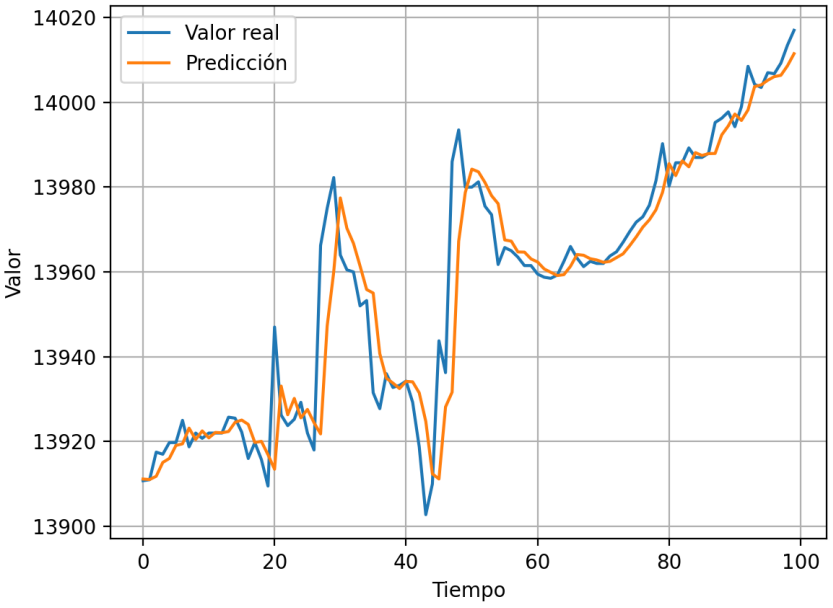


Figura 17: Comparación del valor del precio en serie temporal High respecto a su predicción MLP

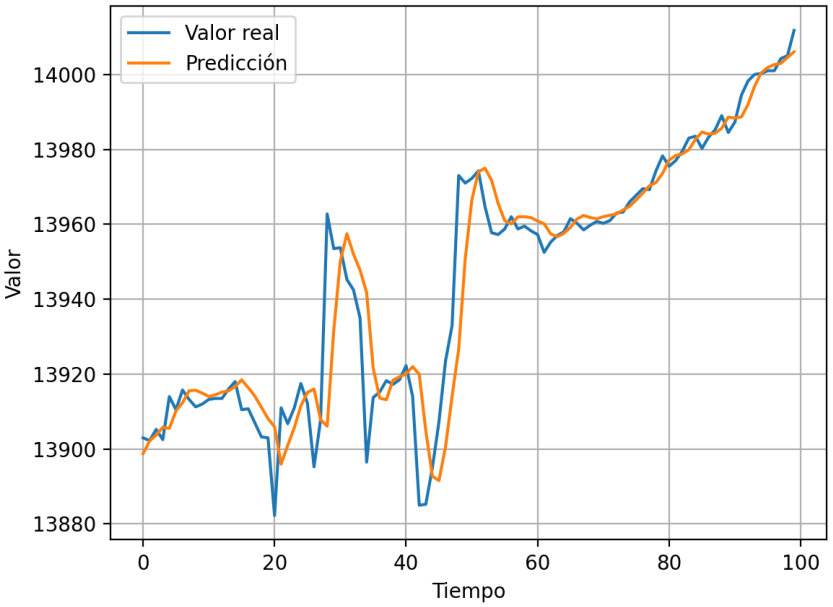


Figura 18: Comparación del valor del precio en serie temporal Low respecto a su predicción MLP

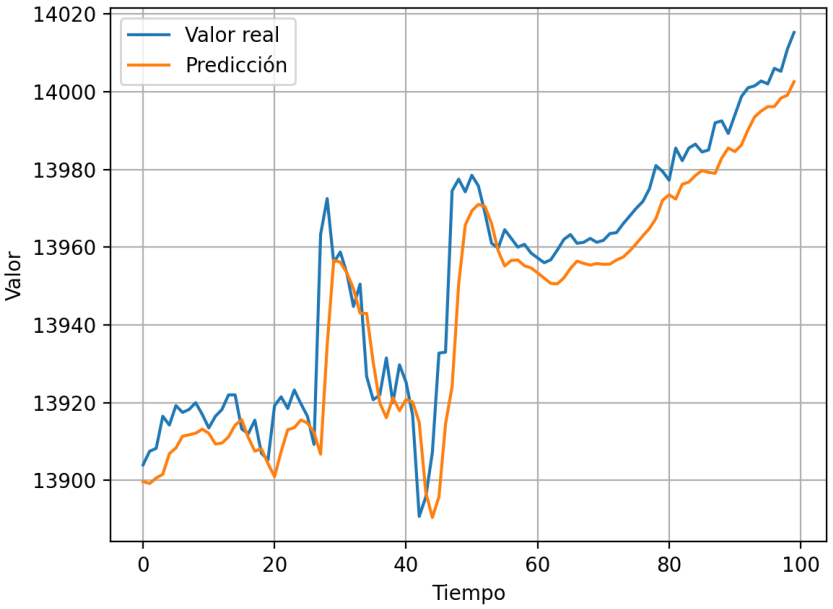
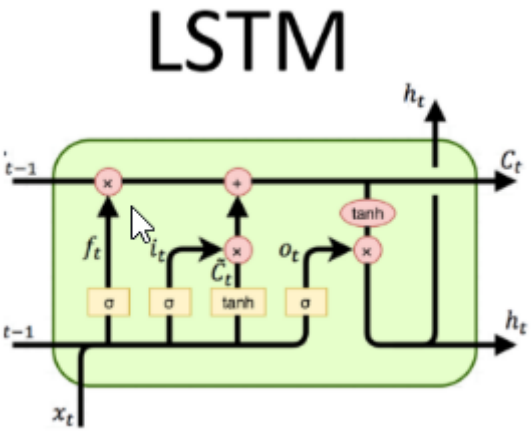


Figura 19: Comparación del valor del precio en serie temporal Close respecto a su predicción MLP

8.3. LSTM

Las redes LSTM (Long-Short Term Memory) son un tipo de redes neuronales recurrentes cuya principal característica es que dotan de más capacidad de memoria para aprender de experiencias sucedidas anteriormente. En el caso planteado, esta particularidad la hace más apropiada para realizar los pronósticos en series temporales aunque no por ello queda exenta de realizar los ajustes necesarios para la obtención de los resultado deseados.

La diferencia respecto al resto de redes recurrentes radica principalmente en el término *Long*, que se traduce en una estructura más compleja y almacenando o descartando según qué valores intermedios van circulando por su estructura. Internamente funciona como las otras, asignando pesos que se van aprendiendo a través del algoritmo.



En cada una de las neuronas LSTM se pueden observar 3 mecanismos distintos que limitan el acceso a las celdas de información. Estos mecanismos (*input gate, forget gate y output gate*) determinan si se permite o rechaza una nueva entrada, se elimina la información o se acepta y sigue adelante.

Existen varios tipos predefinidos de redes LSTM, y se han tomado como referencia para no añadir complejidad sin criterio alguno, dado que la existencia de éstas viene determinada por justificaciones teóricas. Los tipos seleccionados son:

- Vanilla
- Stacked
- Bidirecional
- LSTM-CNN

Se procede pues a la realización de los cálculos mediante los esquemas citados para continuar con la comparación. Añadir la particularidad de estas redes, que requieren de una modificación adicional en la capa de entrada, ya que se necesita un esquema en 3 dimensiones, por lo que se realizan los pasos necesarios en cuestión.

8.3.1. Vanilla

Esta red es un modelo LSTM que está formado por únicamente una capa oculta LSTM, y una capa de salida para realizar la predicción. El código utilizado para su creación, se encuentra en el Anexo C para su pertinente comprobación.

La estructura obtenida con su creación y tras la llamada al descriptor *summary* es la siguiente:

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 50)                10400
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 51
=================================================================
Total params: 10,451
Trainable params: 10,451
Non-trainable params: 0
```

Del mismo modo que en el apartado anterior, se muestra la salida de los primeros epoch:

```
====================
Calculando Vanilla-High
====================
Epoch 1/50
1750/1750 [==============================] - 5s 3ms/step - loss: 1628284.3750 - mse: 1628284.3750
- mae: 263.2969 - val_loss: 225.0794 - val_mse: 225.0794 - val_mae: 9.8374
Epoch 2/50
1750/1750 [==============================] - 4s 3ms/step - loss: 171.1031 - mse: 171.1031 -
mae: 8.4559 - val_loss: 222.2571 - val_mse: 222.2571 - val_mae: 9.7293
...
====================
Calculando Vanilla-Low
====================
Epoch 1/50
1750/1750 [==============================] - 6s 3ms/step - loss: 6732634.5000 - mse: 6732634.5000
- mae: 762.4439 - val_loss: 212.0472 - val_mse: 212.0472 - val_mae: 9.3414
Epoch 2/50
1750/1750 [==============================] - 5s 3ms/step - loss: 173.4130 - mse: 173.4130 -
mae: 8.3653 - val_loss: 212.7755 - val_mse: 212.7755 - val_mae: 9.4265
...
====================
Calculando Vanilla-Close
====================
Epoch 1/50
1750/1750 [==============================] - 6s 3ms/step - loss: 3543373.7500 - mse: 3543373.7500
- mae: 476.2422 - val_loss: 263.1051 - val_mse: 263.1051 - val_mae: 10.3962
Epoch 2/50
1750/1750 [==============================] - 5s 3ms/step - loss: 207.6964 - mse: 207.6964 -
mae: 9.1878 - val_loss: 267.3408 - val_mse: 267.3408 - val_mae: 10.6545
...
```

Tras la ejecución hasta el epoch número 22, al no cumplirse la mejora de los cálculos, el *callback* finaliza la ejecución con los siguientes resultados:

	High	**Low**	**Close**
MSE Vanilla	127.43686731195109	128.33931356716036	151.5648771066117

Si realizamos la representación gráfica de errores a medida que avanza el entrenamiento obtenemos:

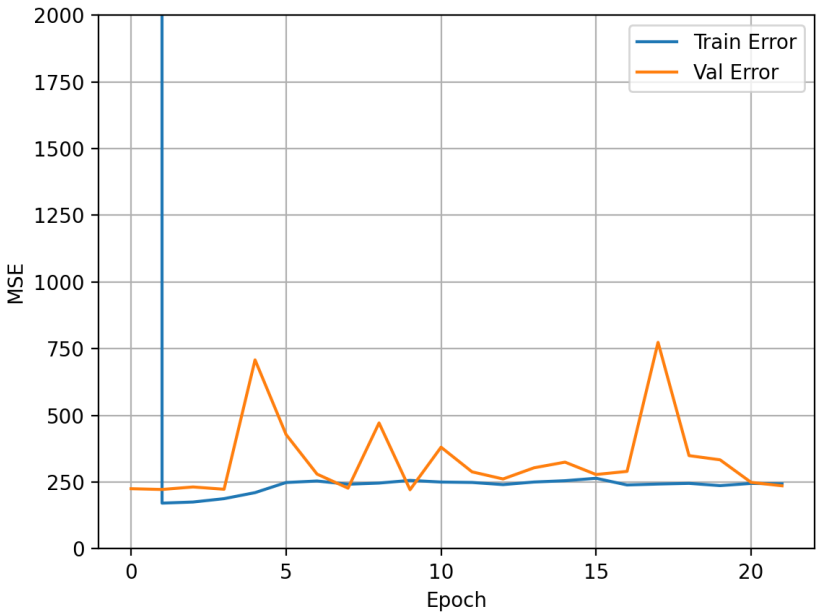


Figura 20: Comparación error de entrenamiento y validación serie temporal High LSTM Vanilla

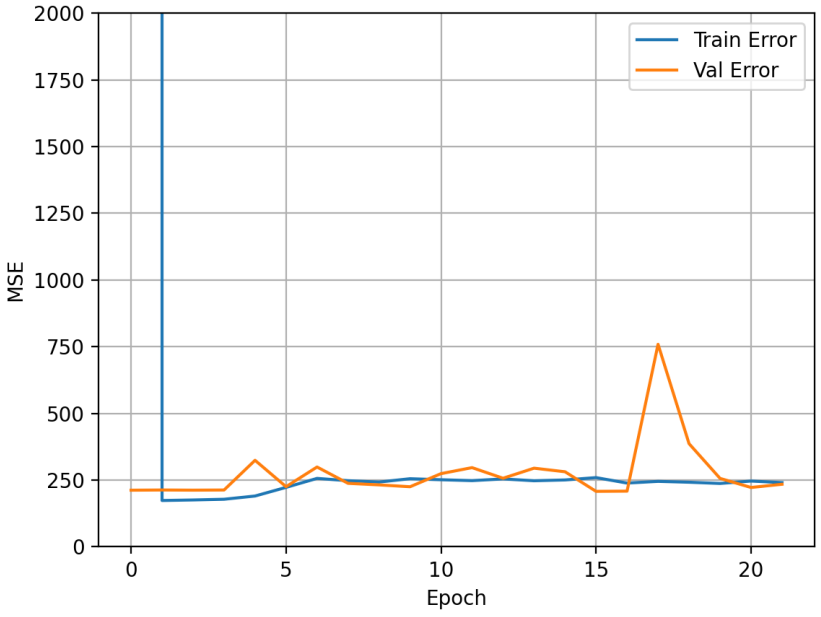


Figura 21: Comparación error de entrenamiento y validación serie temporal Low LSTM Vanilla

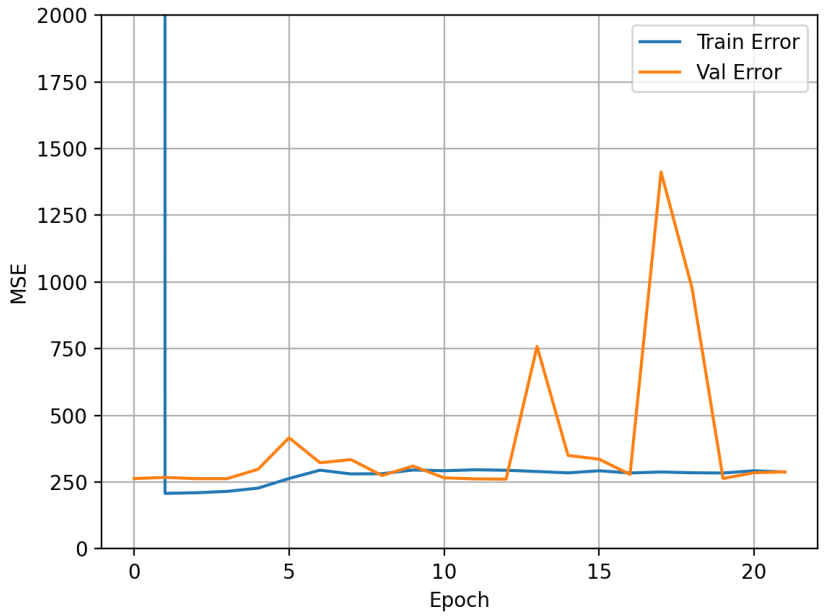


Figura 22: Comparación error de entrenamiento y validación serie temporal Close LSTM Vanilla

Para comprobar gráficamente los resultados generados en el test, se puede visualizar en los siguientes gráficos para las 3 series.

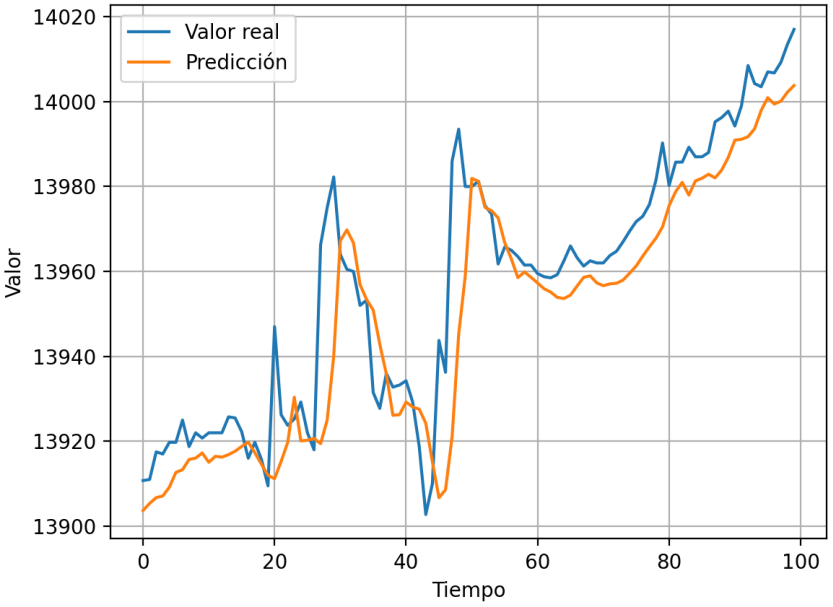


Figura 23: Comparación del valor del precio en serie temporal High respecto a su predicción LSTM Vanilla

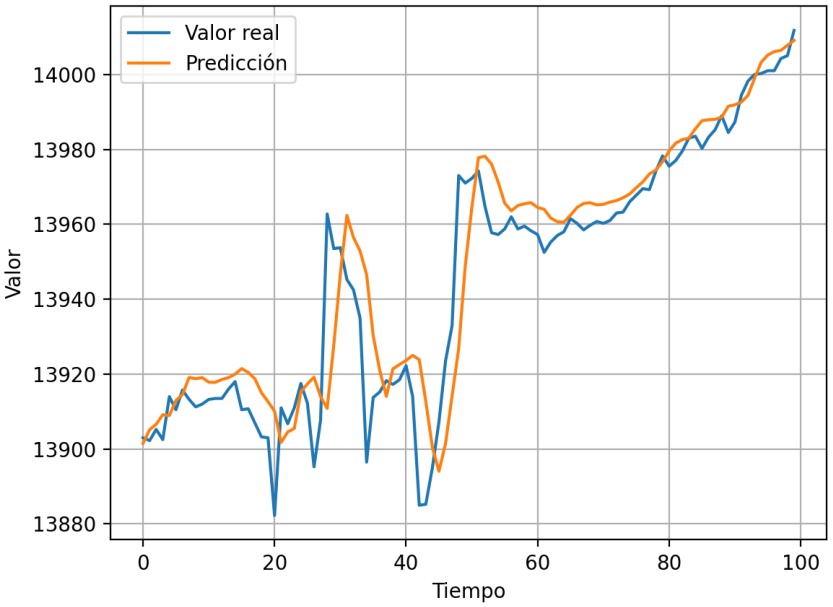


Figura 24: Comparación del valor del precio en serie temporal Low respecto a su predicción LSTM Vanilla

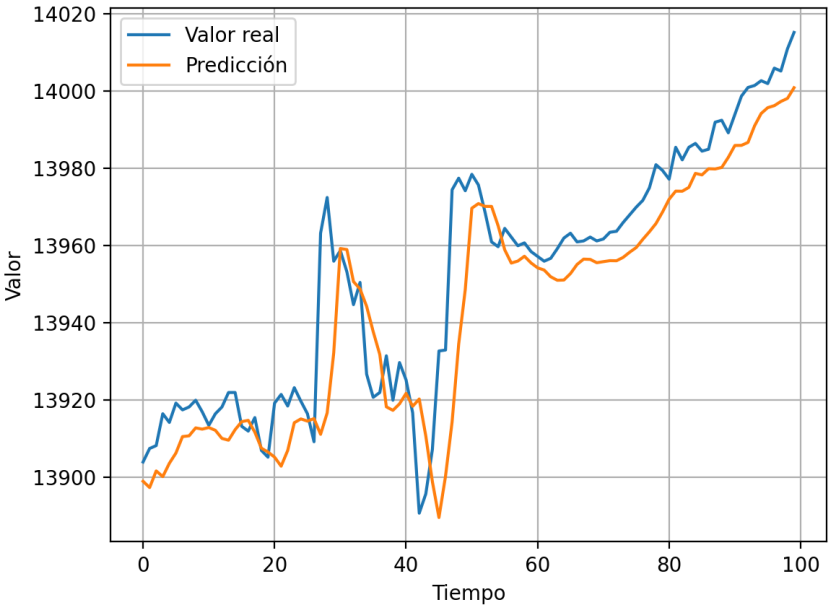


Figura 25: Comparación del valor del precio en serie temporal Close respecto a su predicción LSTM Vanilla

La comparación de las estimaciones realizadas con respecto a los valores reales, también muestra el nivel de semejanza aunque los valores de la métrica MSE se traduzcan como menos adecuados.

8.3.2. Stacked

Este es el segundo tipo de red neuronal LSTM planteada. En este caso se añaden varias capas LSTM, de ahí la definición de su nombre. La estructura que queda definida es la siguiente:

```
 Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 3, 50)             10400
_________________________________________________________________
lstm_4 (LSTM)                (None, 50)                20200
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 51
=================================================================
Total params: 30,651
Trainable params: 30,651
Non-trainable params: 0
```

El código fuente necesario para crear ese modelo se encuentra detallado en el Anexo C.

En este caso se repite la casuística del tipo anterior, y se ejecuta 22 Epoch para finalizar dado que no mejora. Obviamente este valor viene definido en el callback de parada, pero de esta manera se obliga a volcar información que permitirá mejorar los modelos en un futuro.

La salida de los primeros entrenamiento sería:

```
====================
Calculando Stacked-High
====================
Epoch 1/50
1750/1750 [==============================] - 9s 4ms/step - loss: 8446734.0000 - mse: 8446734.0000
- mae: 868.7178 - val_loss: 248.8168 - val_mse: 248.8168 - val_mae: 10.1398
Epoch 2/50
1750/1750 [==============================] - 7s 4ms/step - loss: 194.2131 - mse: 194.2131 -
mae: 8.9858 - val_loss: 252.3793 - val_mse: 252.3793 - val_mae: 10.3335
...
====================
Calculando Stacked-Low
====================
Epoch 1/50
1750/1750 [==============================] - 9s 4ms/step - loss: 4498827.0000 - mse: 4498827.0000
- mae: 504.3867 - val_loss: 270.4021 - val_mse: 270.4021 - val_mae: 10.6910
Epoch 2/50
1750/1750 [==============================] - 7s 4ms/step - loss: 217.8174 - mse: 217.8174 -
mae: 9.4738 - val_loss: 266.3803 - val_mse: 266.3803 - val_mae: 10.6822
...
====================
Calculando Stacked-Close
====================
Epoch 1/50
1750/1750 [==============================] - 8s 4ms/step - loss: 4561593.0000 - mse: 4561593.0000
- mae: 566.3125 - val_loss: 283.6313 - val_mse: 283.6313 - val_mae: 10.9488
Epoch 2/50
1750/1750 [==============================] - 7s 4ms/step - loss: 221.5024 - mse: 221.5024 -
mae: 9.6008 - val_loss: 275.7683 - val_mse: 275.7683 - val_mae: 10.6233
```

En este caso, pese a ser una estructura más compleja que permitiría mejorar los resultado de los modelos anteriores, los valores obtenidos por MSE no solo no mejoran, sino que empeoran:

	High	Low	Close
MSE Stacked	309.13927044866466	160.58988496130726	193.59288979974528

La representación de todos los cálculos, se puede visualizar con más claridad en las representaciones siguientes:

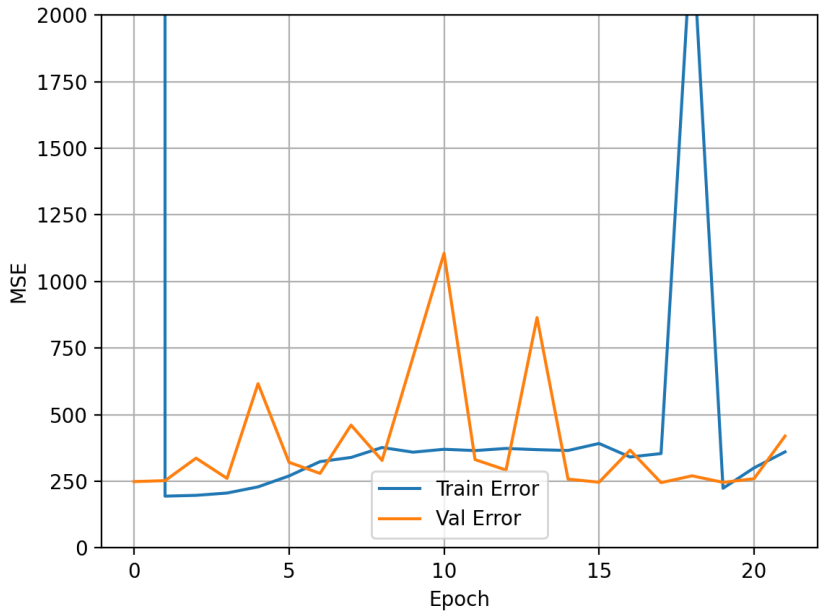


Figura 26: Comparación error de entrenamiento y validación serie temporal High LSTM Stacked

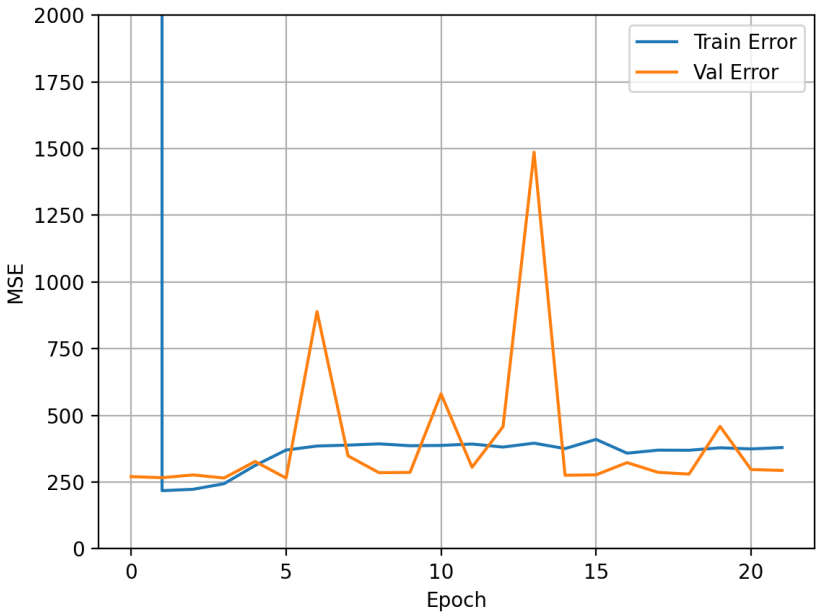


Figura 27: Comparación error de entrenamiento y validación serie temporal Low LSTM Stacked

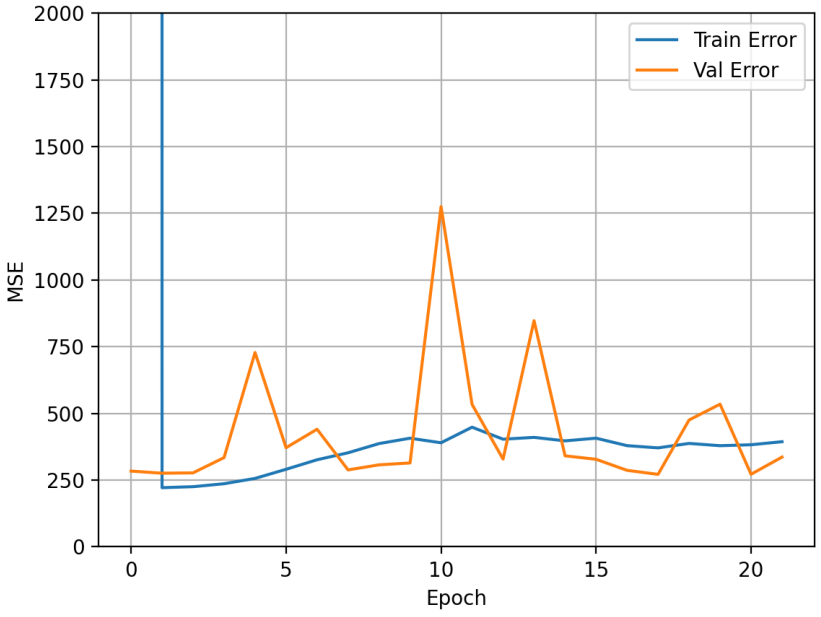


Figura 28: Comparación error de entrenamiento y validación serie temporal Close LSTM Stacked

Para comprobar gráficamente los resultados generados en el test, se puede visualizar en los siguientes gráficos para las 3 series.

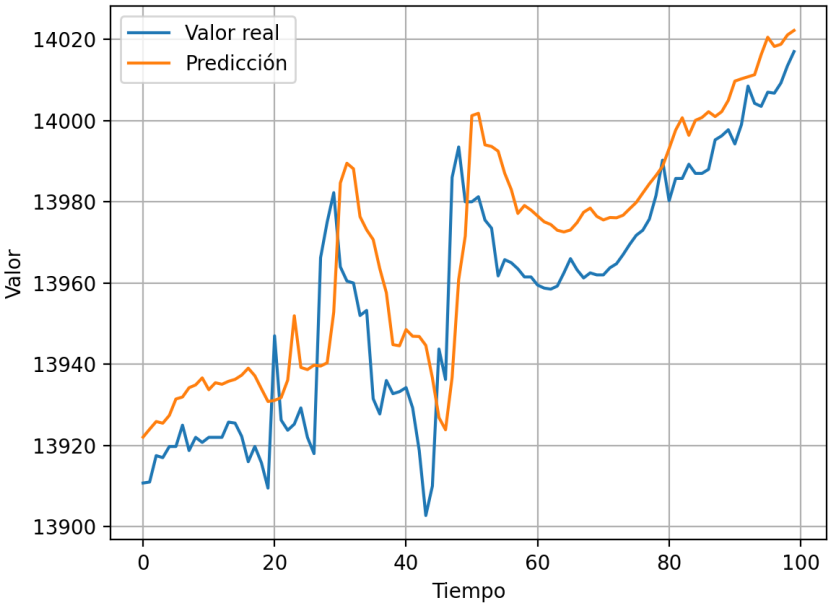


Figura 29: Comparación del valor del precio en serie temporal High respecto a su predicción LSTM Stacked

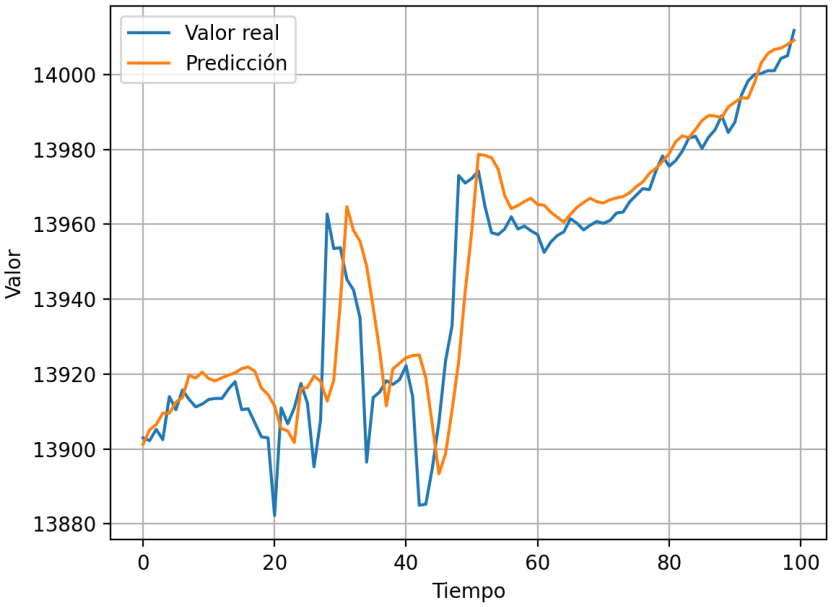


Figura 30: Comparación del valor del precio en serie temporal Low respecto a su predicción LSTM Stacked

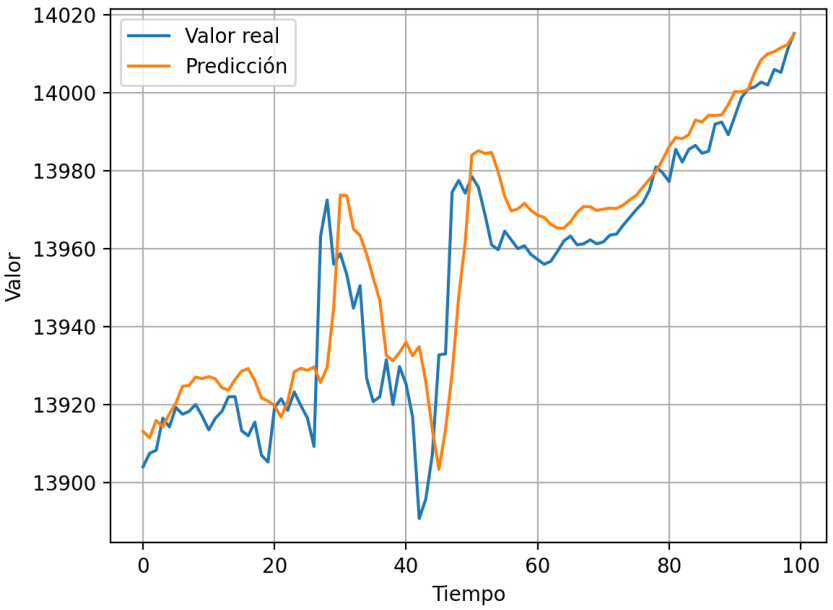


Figura 31: Comparación del valor del precio en serie temporal Close respecto a su predicción LSTM Stacked

8.3.3. Bidirectional

Es el tercer tipo de red LSTM comprobada. La partidularidad en este caso es que se dispone de una capa LSTM que es capaz de aprender en ambos sentidos y combinar los resultados, por lo que se van ajustando todos los pesos en reiteradas ocasiones. El modelo utilizado es muy sencillo, dado que solo se ha incluido una capa LSTM, de esta manera se puede determinar, en caso de mejora, que ésta es la causa de ello.

La estructura del modelo utilizado para las 3 series temporales sería:

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional (Bidirectional (None, 100)               20800
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 101
=================================================================
Total params: 20,901
Trainable params: 20,901
Non-trainable params: 0
_________________________________________________________________
```

El código necesario se encuentra como en el resto de modelos en el Anexo C.

En este caso se continúa con el mismo escenario de ejecución del entrenamiento. Debido a la estructura del conjunto de datos de entrada, no se llega a producir mejora de manera que se realiza la parada de forma inmediata en cuanto se cumple la condición del callback.

```
====================
Calculando Bidirectional-High
====================
Epoch 1/50
```

```
1750/1750 [==============================] - 8s 3ms/step - loss: 1968293.8750 - mse: 1968293.8750
- mae: 291.5004 - val_loss: 199.5029 - val_mse: 199.5029 - val_mae: 9.1159
Epoch 2/50
1750/1750 [==============================] - 6s 3ms/step - loss: 153.7843 - mse: 153.7843 -
mae: 7.9763 - val_loss: 198.2064 - val_mse: 198.2064 - val_mae: 9.0574
...
=====================
Calculando Bidirectional-Low
=====================
Epoch 1/50
1750/1750 [==============================] - 8s 3ms/step - loss: 2871045.2500 - mse: 2871045.2500
- mae: 392.4045 - val_loss: 209.8400 - val_mse: 209.8400 - val_mae: 9.3172
Epoch 2/50
1750/1750 [==============================] - 5s 3ms/step - loss: 171.9041 - mse: 171.9041 -
mae: 8.4249 - val_loss: 207.8198 - val_mse: 207.8198 - val_mae: 9.3279
...
=====================
Calculando Bidirectional-Close
=====================
Epoch 1/50
1750/1750 [==============================] - 7s 3ms/step - loss: 2460986.7500 - mse: 2460986.7500
- mae: 321.0185 - val_loss: 195.5147 - val_mse: 195.5147 - val_mae: 8.8786
Epoch 2/50
1750/1750 [==============================] - 5s 3ms/step - loss: 154.6648 - mse: 154.6648 -
mae: 7.9087 - val_loss: 194.8638 - val_mse: 194.8638 - val_mae: 8.8641
```

Los resultados en este caso para la métrica MSE serían:

	High	**Low**	**Close**
MSE Bidirectional	122.47265624495326	95.31600641004681	89.97088754282234

En este caso, pese a tratarse de una estructura más sencilla ha obtenido mejor resultado que el modelo anterior, incluso que Vanilla que era de un esquema similar. Es pues también un punto de referencia para ver si es mejorable de forma considerable respecto a MLP.

Al igual que en los apartados anteriores, se muestran los gráficos asociados a la etapa de entrenamiento para ver la evolución del error de entrenamiento y validación:

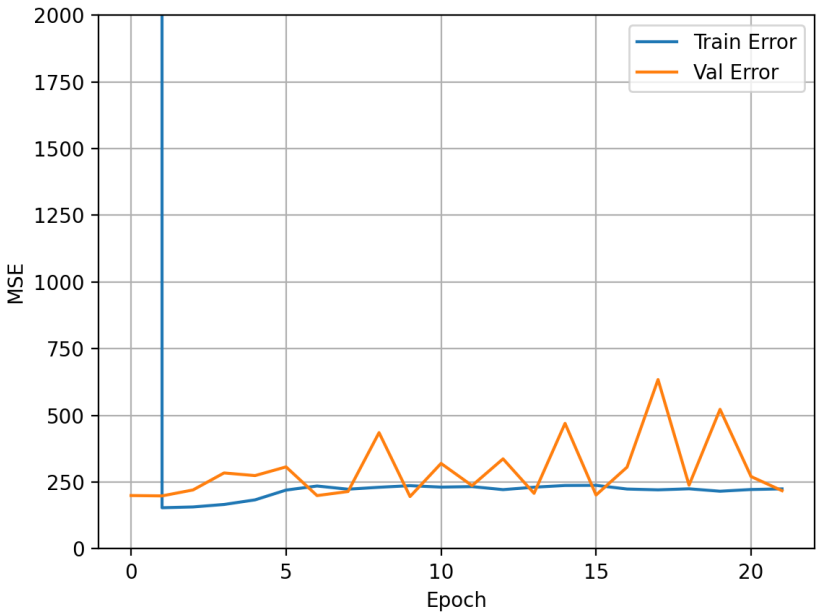


Figura 32: Comparación error de entrenamiento y validación serie temporal High LSTM Bidirectional

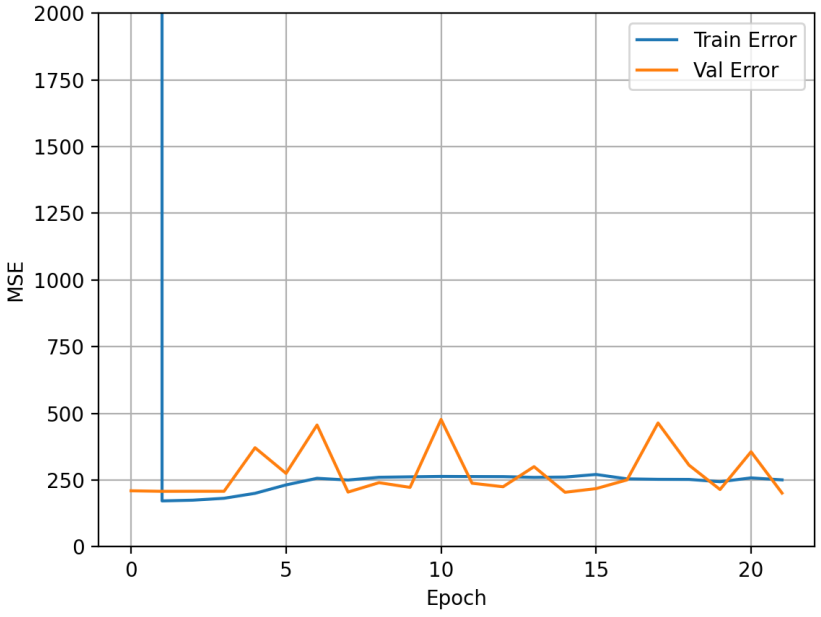


Figura 33: Comparación error de entrenamiento y validación serie temporal Low LSTM Bidirectional

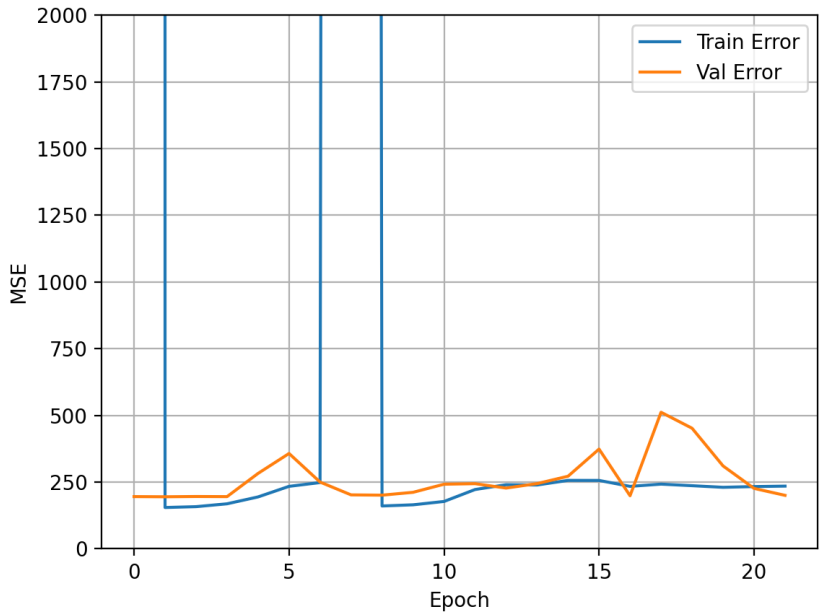


Figura 34: Comparación error de entrenamiento y validación serie temporal Close LSTM Bidirectional

Para comprobar gráficamente los resultados generados en el test, se puede visualizar en los siguientes gráficos para las 3 series.

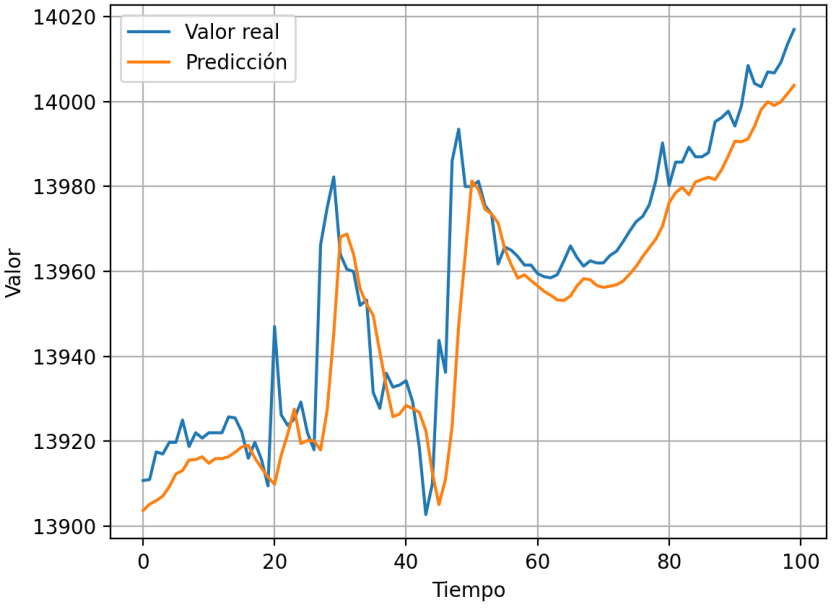


Figura 35: Comparación del valor del precio en serie temporal High respecto a su predicción LSTM Bidirectional

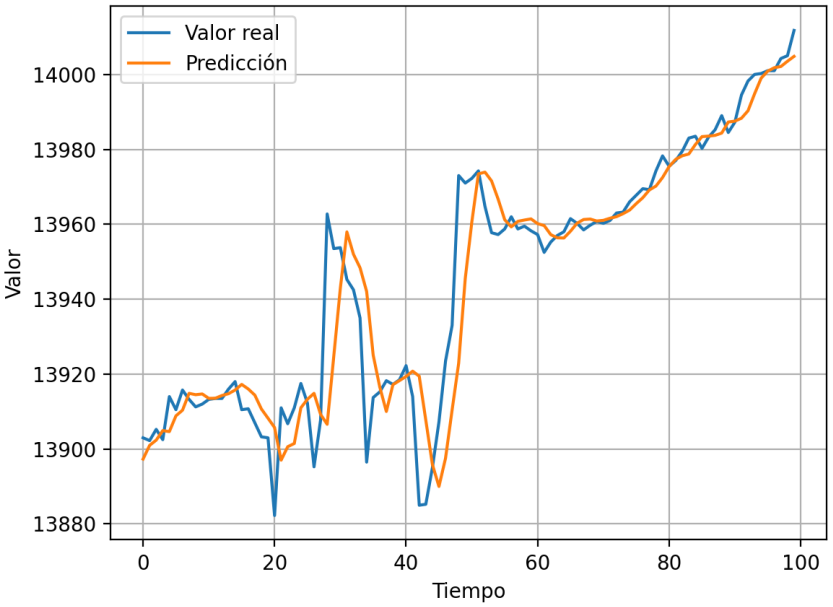


Figura 36: Comparación del valor del precio en serie temporal Low respecto a su predicción LSTM Bidirectional

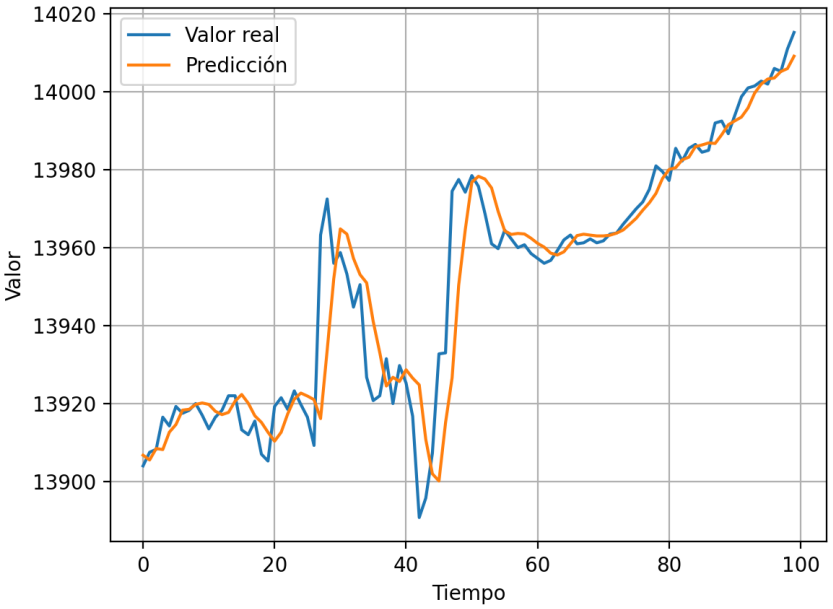


Figura 37: Comparación del valor del precio en serie temporal Close respecto a su predicción LSTM Bidirectional

8.3.4. LSTM-CNN

Este tipo de red neuronal tiene la particularidad de añadir una capa convolucional a la LSTM que aprovecha la eficiencia a la hora de extraer y aprender característas en secuencias unidimencionales. Es pues, un tipo híbrido cuyo código de creación se puede observar en el Anexo C y cuya estructura queda de la siguiente manera:

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 2, 64)             192
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 1, 64)             0
_________________________________________________________________
flatten (Flatten)            (None, 64)                0
_________________________________________________________________
dense_15 (Dense)             (None, 50)                3250
_________________________________________________________________
dense_16 (Dense)             (None, 1)                 51
=================================================================
Total params: 3,493
Trainable params: 3,493
Non-trainable params: 0
```

En este caso, la métrica MSE ha generado los siguientes resultados:

	High	**Low**	**Close**
MSE CNN	123.29302285449538	201.98126436463428	101.578734510759

Se observa de esto valores que no hay una mejora respecto al modelo básico de referencia.

Pese a la particularidad de los valores obtenidos en MSE, la representación de cómo evoluciona el error repite el formato de los anteriores:

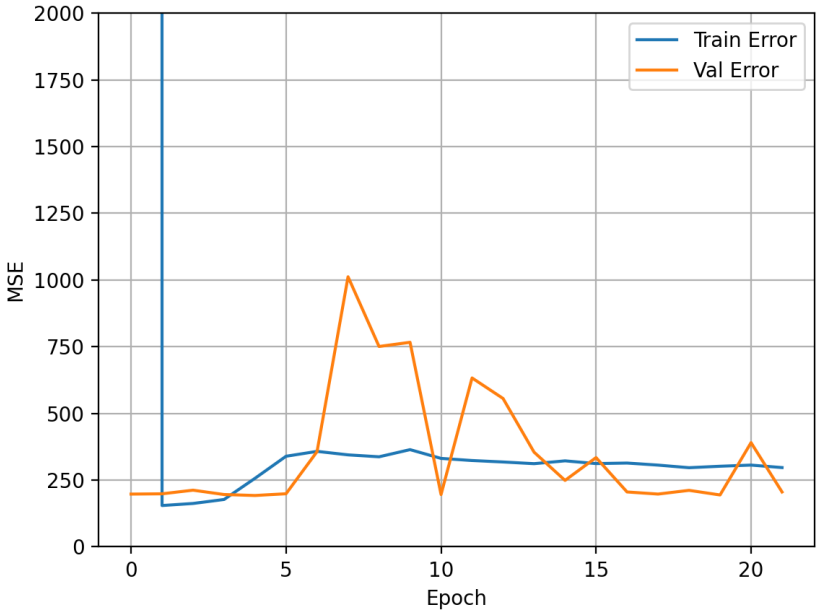


Figura 38: Comparación error de entrenamiento y validación serie temporal High LSTM CNN

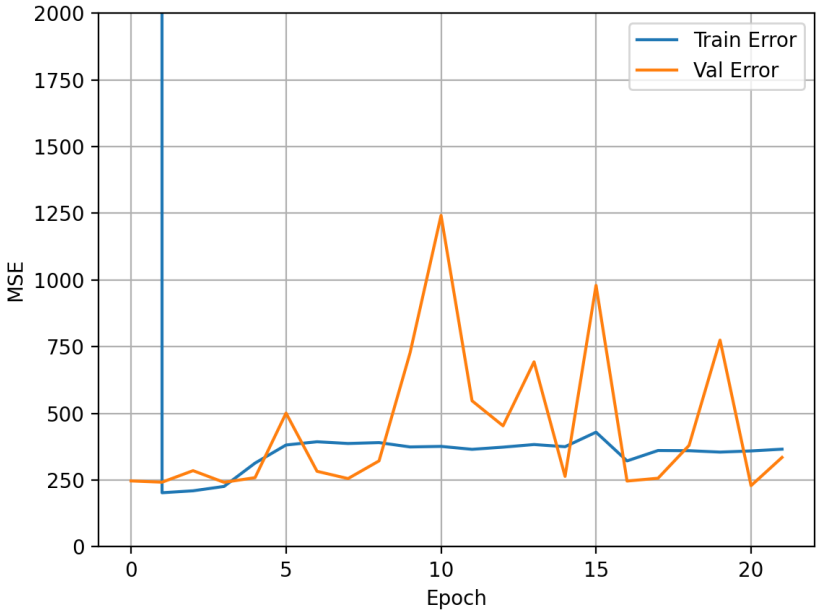


Figura 39: Comparación error de entrenamiento y validación serie temporal Low LSTM CNN

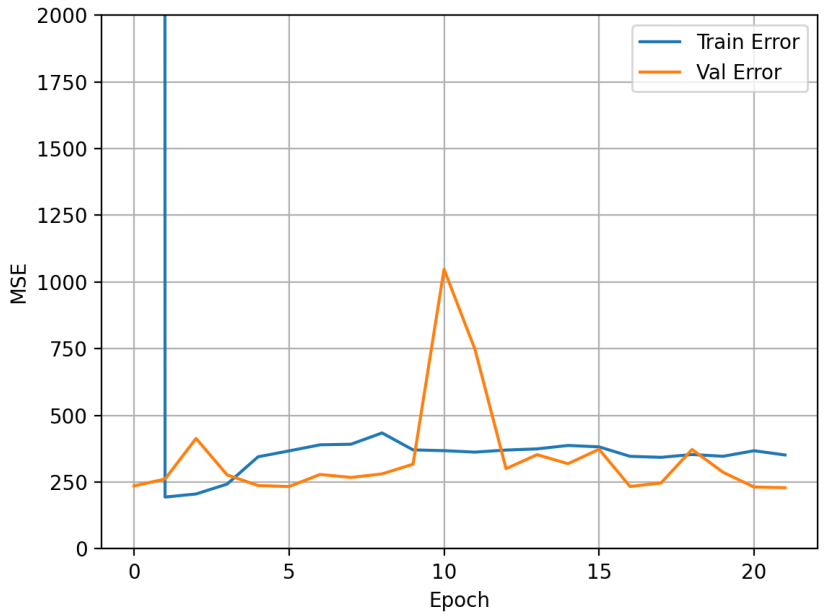


Figura 40: Comparación error de entrenamiento y validación serie temporal Close LSTM CNN

Para comprobar gráficamente los resultados generados en el test, se puede visualizar en los siguientes gráficos para las 3 series.

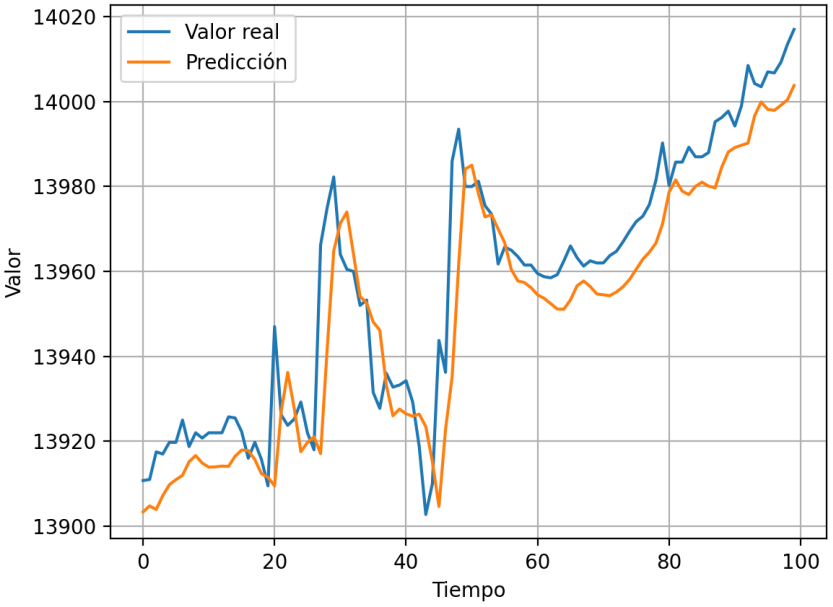


Figura 41: Comparación del valor del precio en serie temporal High respecto a su predicción LSTM CNN

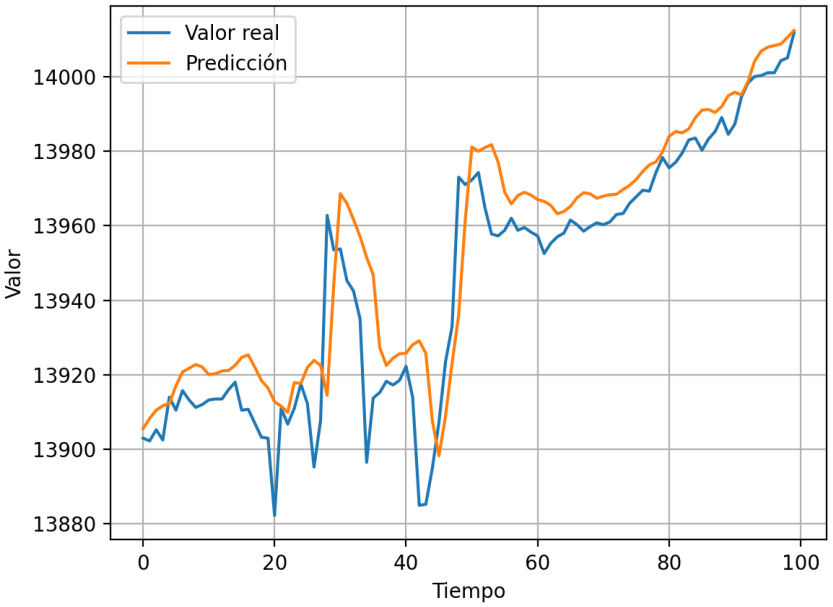


Figura 42: Comparación del valor del precio en serie temporal Low respecto a su predicción LSTM CNN

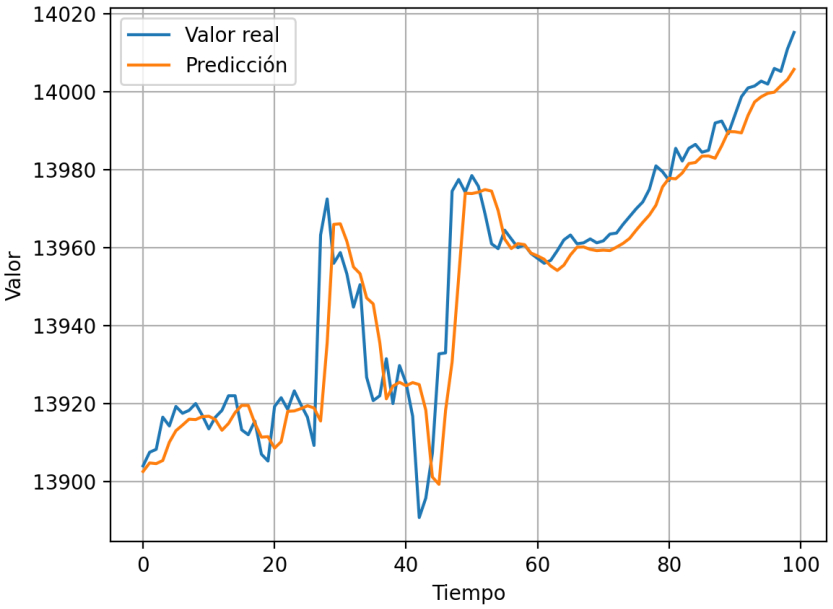


Figura 43: Comparación del valor del precio en serie temporal Close respecto a su predicción LSTM CNN

8.4. Conclusiones y mejoras

Tras revisar todas las opciones estudiadas, y compararlas, se puede confirmar que se necesita reconstruir el proceso desde las fases iniciales para poder obtener un mayor provecho al proceso realizado. Dada la siguiente tabla:

	High	**Low**	**Close**
MLP	62.364526	86.465794	111.568537
Vanilla LSTM	127.436867	128.339314	151.564877
Stacked LSTM	309.139270	160.589885	193.592890
Bidirectional LSTM	122.472656	95.316006	89.970888
CNN LSTM	123.293023	201.981264	101.578735

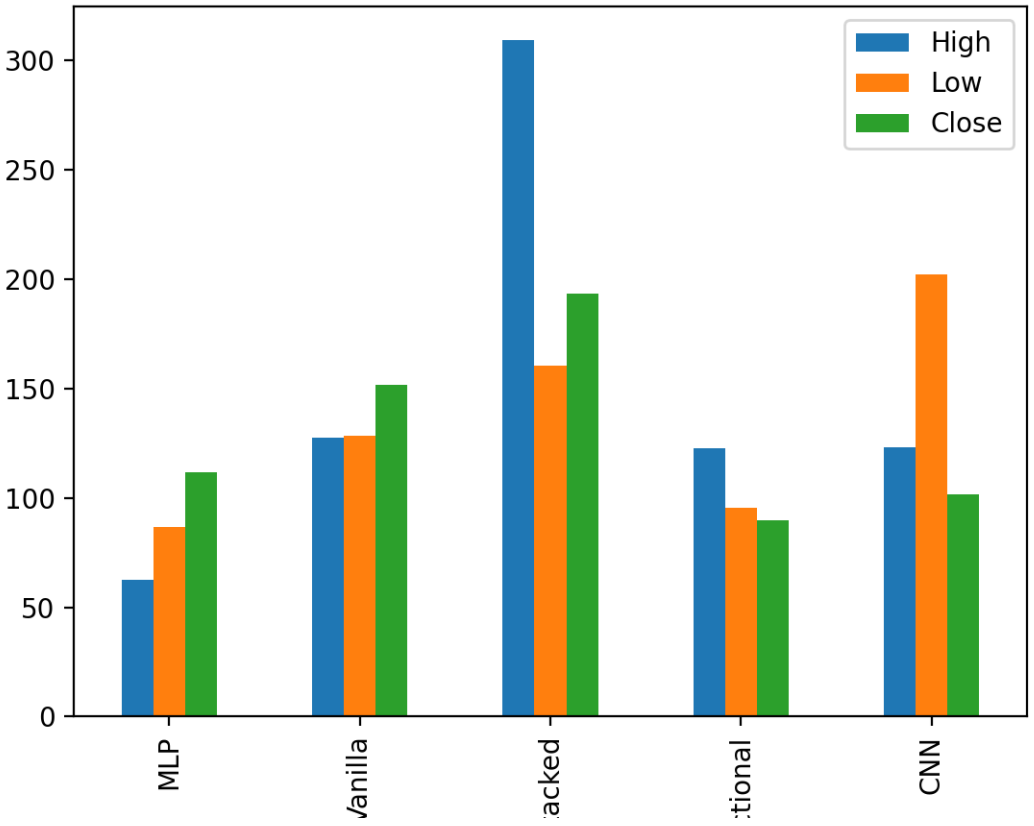


Figura 44: MSE en todas las Series Temporales por Modelos

Se observa que para esta ejecución, fijada en parámetros de aleatoriedad para poder repetir el escenario ante posibles cambios o mejoras, el modelo más sencillo que es el perceptrón multicapa, ha obtenido los mejores resultados en la métrica seleccionada en las series *High, Low*, y el modelo *Bidirectional* el mejor en valor *Close*.

Una representación de un fragmento de las 3 series temporales respecto al valor real tendría la siguiente estructura:

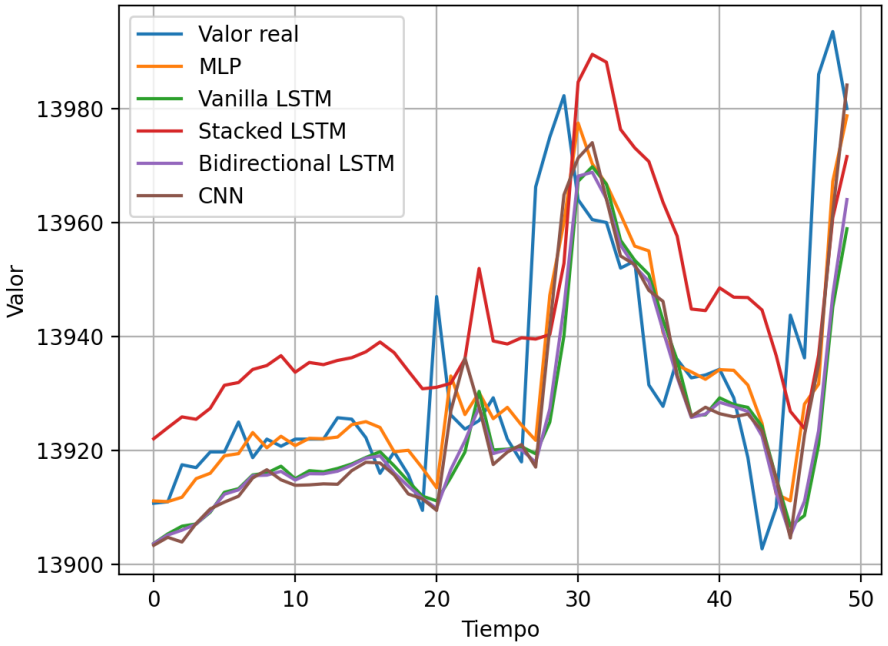


Figura 45: Comparación del precio real respecto a las predicciones en serie temporal High

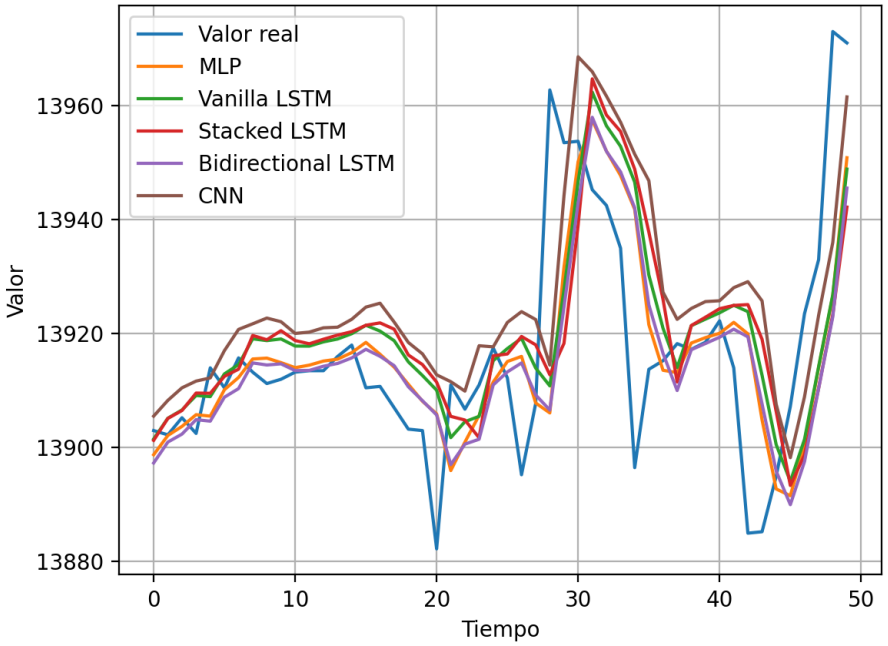


Figura 46: Comparación del precio real respecto a las predicciones en serie temporal Low

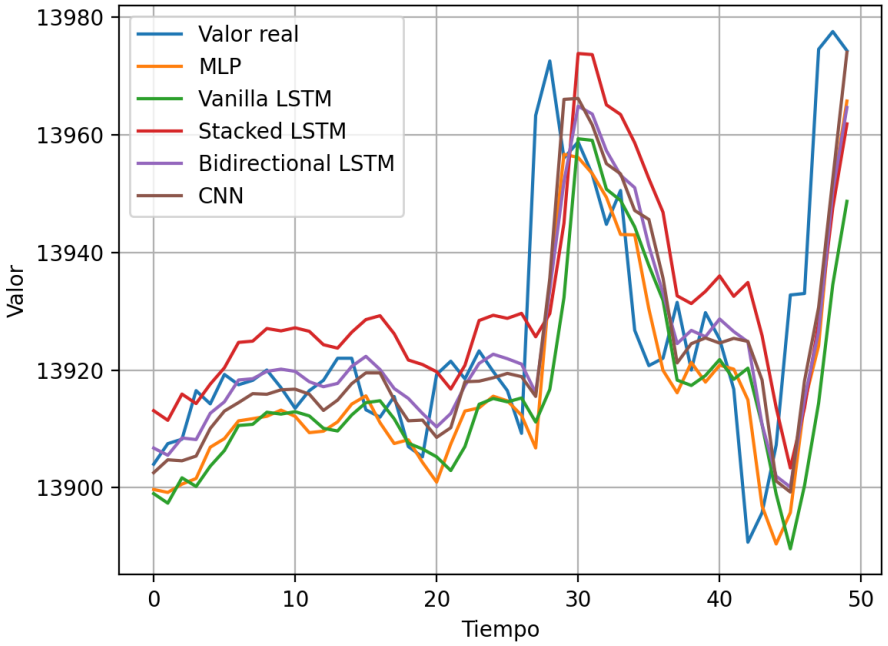


Figura 47: Comparación del precio real respecto a las predicciones en serie temporal Close

El origen de los datos se presupone con un alto contenido de ruido generado en los mercados por las distintas tipologías de estrategias y agentes que intervienen, por lo que intentar reducir todos aquellos elementos que no llegan a aportar información podrían mejorar considerablemente el proceso. Una de las posibles medidas a realizar es la de acotar la franja horaria. En el conjunto de datos se han cargado los datos obtenidos de las 24 horas de mercado, cuando realmente para el tipo de operativa discrecional no es relevante, ya que la mayoría de estrategia sólo son viables si hay un volumen detrás, si se observa que las llamadas "manos fuertes.ºperan y que por lo tanto, no va a venir algún tipo de fuerza superior a girar lo que se tenga planteado.

Obviamente existen imprevistos derivados de catástrofes o situaciones excepcionales, pero tratar ese tipo de comportamientos ya es bastante más complejo puesto que se tendría que realizar de forma específica y aislada. En principio hay una franja de 2 o 4 horas máximo al día dónde las operativas manuales tienen sentido, cuando se detecta en el mercado un volumen sustancial que respalda los movimientos y traduce la información del mercado para tomar decisiones. Según el activo con el que se esté trabajando se tomaría una franja u otra sesión si se opera en el mercado europeo, americano, asiático o una combinación de ellos. En este caso, la operativa sobre el NASDAQ se da más en sesión americana, entre las 15:00 y 17:00 aproximadamente hora peninsular, el DAX alemán sería entre las 8:45 y 10:00 principalmente o el futuro del EURUSD que se movería en ambas sesiones europea y americana.

Los siguientes gráficos muestran unas capturas de los activos citados, dón en la parte inferior se observa el comportamiento mencionado. El volumen cambia considerablemente de una franja horaria a otra, por lo que sería muy importante tenerlo en cuenta.

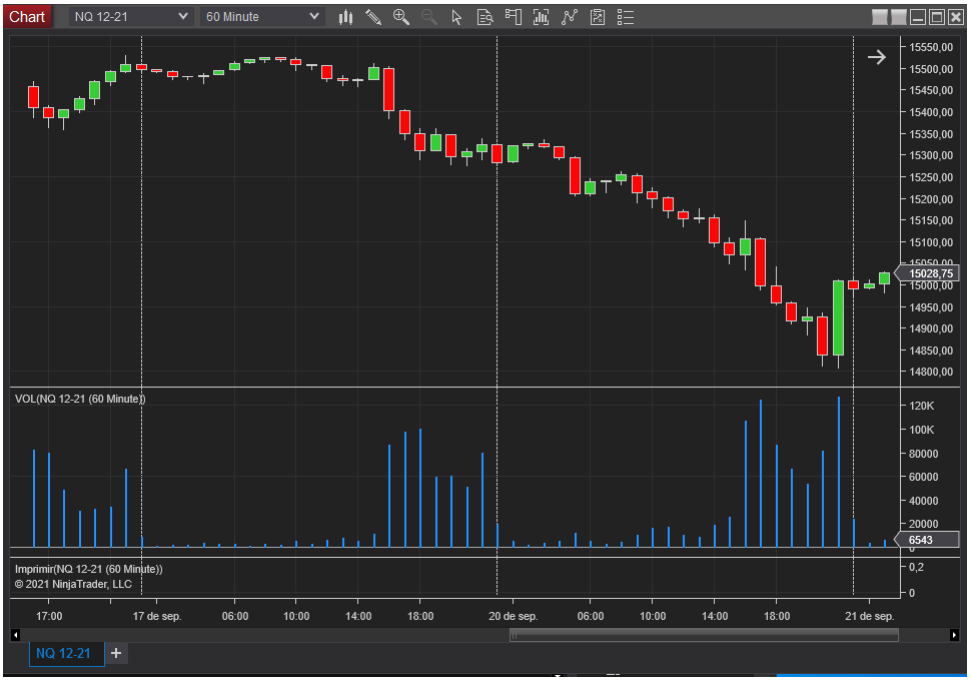


Figura 48: Representación de volumen en Nasdaq

En este primer gráfico (Figura 48 , se observa el volumen de la parte inferior, y es en la sesión americana a partir de las 15:00 aproximadamente cuando aumenta considerablemente.

Analizando el Dax alemán (Figura 49):

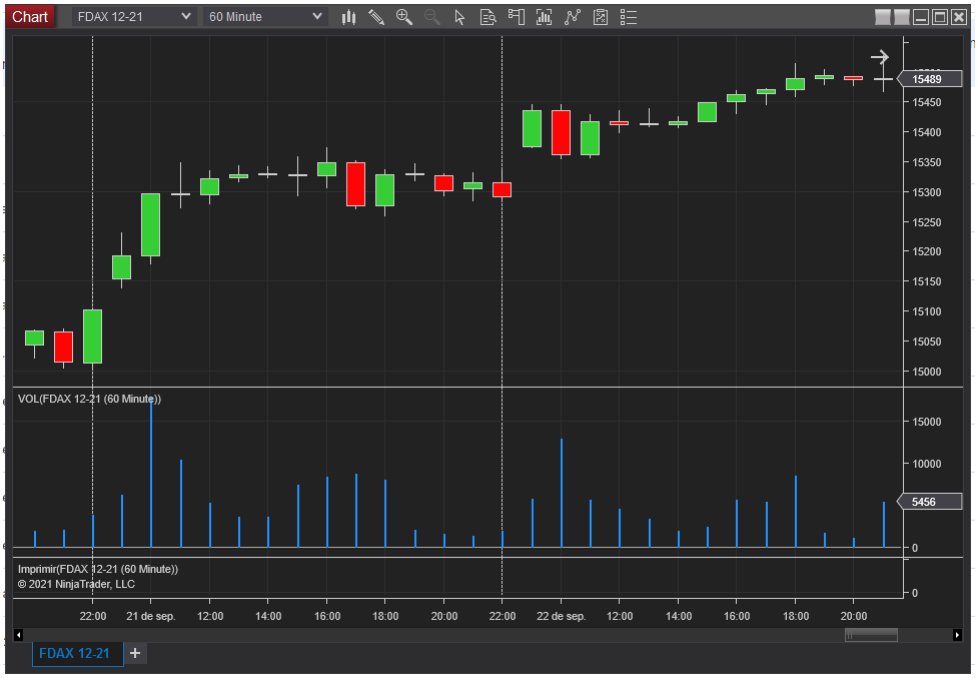


Figura 49: Representación de volumen en Dax

Es en mayor medida en la sesión europea dónde hay una mayor operativa y luego algo menor pero también significativa en la europea.

Por último, el futuro de la divisa EURUSD

Figura 50: Representación de volumen en EURUSD

En este caso se comprueba que está la operativa más diluida a lo largo de todo el día, sin picos significativos.

Otro punto a tener en cuenta es el de qué modelo es más adecuado. Las métricas utilizadas pueden dar una pista de ello, pero se debe tener en cuenta que es tendencia natural en los desarrollos de aprendizaje automática la búsqueda de modelos combinados o ensamblados que contemplan el conocimiento de varios de ellos, resultando en valores si no más ajustados a nivel individual en comportamiento más estables. El peso de los modelos podría dejarse inicialmente como equitativo e ir ajustando hasta que se obtenga una mejora sustancial. Hay que tener en cuenta siempre que las condiciones del mercado son muy cambiantes y que interesa los modelos se comporten de forma adecuada con una alta varianza (bias-variance trade-off).

Una alternativa al planteamiento anterior, sería utilizar el modelo más óptimo para cada serie, que según los resultados de la tabla anterior sería MLP para *High,Low* y LSTM Bidirectional para *Close*. El concepto de máximo y mínimo difiere sustancialmente según si el mercado alcista, bajista o lateral, por lo que realizar una primera clasificación de tramos de mercado para posteriormente realizar los entrenamientos también ayudaría.

Otro punto a debatir es la necesidad de ajustar otros valores como el número de neuronas, el nivel de complejidad de la red profunda. Realizar todos estos cambios puede, como es su finalidad, cambiar su comportamiento y mejorar el rendimiento. La generación del modelo más adecuado requiere un análisis mucho más profundo que escapa al alcance de este proyecto, ya que el estudio de otras métricas aparte de la utilizada aportaría muchas más información, pero requiere de una dedicación mucho más extensa.

9. Diseño e implementación

9.1. Almacenamiento de datos: MongoDB

El apartado de almacenamiento es de vital importancia ya que persigue dotar de un mecanismo de comunicación y persistencia a la información a medida que va fluyendo. La principal función es, dotar a la solución final de un punto de búsqueda de los datos necesarios para realizar la predicción, además de almacenar la información generada para su posterior estudio.

En este caso, el sistema de gestión de bases de datos elegido es un elemento importante, tal y como se ha mencionado pero no es crítico en lo que a características se refiere. Es por ello, que la mayoría de alternativas disponibles en el mercado de similares características serían aptas para cubrir las funcionalidades necesarias.

Tras revisar varias opciones, se ha optado por adoptar como solución la implementación de MongoDB en la nube por las siguientes razones:

- Dada su facilidad de escalabilidad, es una opción adecuada si se disponen de mucho o pocos recursos. Como se ha indicado previamente, no se requiere de un canal de comunicación intensivo por lo que independientemente del potencial sobre el que se implementa el sistema no dará problemas.
- Esta disponible en las dos modalidades indicadas, tanto como recurso en la nube como local, por lo que da una flexibilidad adicional en los distintos laboratorios de pruebas.
- Es muy flexible en su puesta en marcha ya que no se requiere un esquema predefinido, lo que agiliza también el proceso.
- El coste es muy bajo, ya que en este caso se ofrece una funcionalidad gratuita en la nube, permitiendo incluso elegir la zona y algunos parámetros adicionales sin ningún tipo de coste.
- Dispone de varios mecanismos de acceso a los datos, por lo que es fácilmente adaptable a futuras funcionalidades que todavía no se han definido.
- Dispone de abundante documentación y herramientas analíticas para optimizar y afinar su funcionamiento.
- Además, el referido como punto débil que se refiere a su comportamiento en transacciones o consultas complejas, tampoco afecta ya que como se ha indicado se trata de una solución con poca exigencia a nivel de intensidad de recursos.

Accediendo a la propia web del producto `http://www.mongodb.com`, se puede crear un cluster de pocos recursos, en la ubicación geográfica deseada y con el proveedor que más se adapte a las necesidades de la solución (*Azure, AWS, ...*). Se parametriza el acceso limitado por usuario/clave y restringido por dirección, todo en un entorno web ágil y amigable.

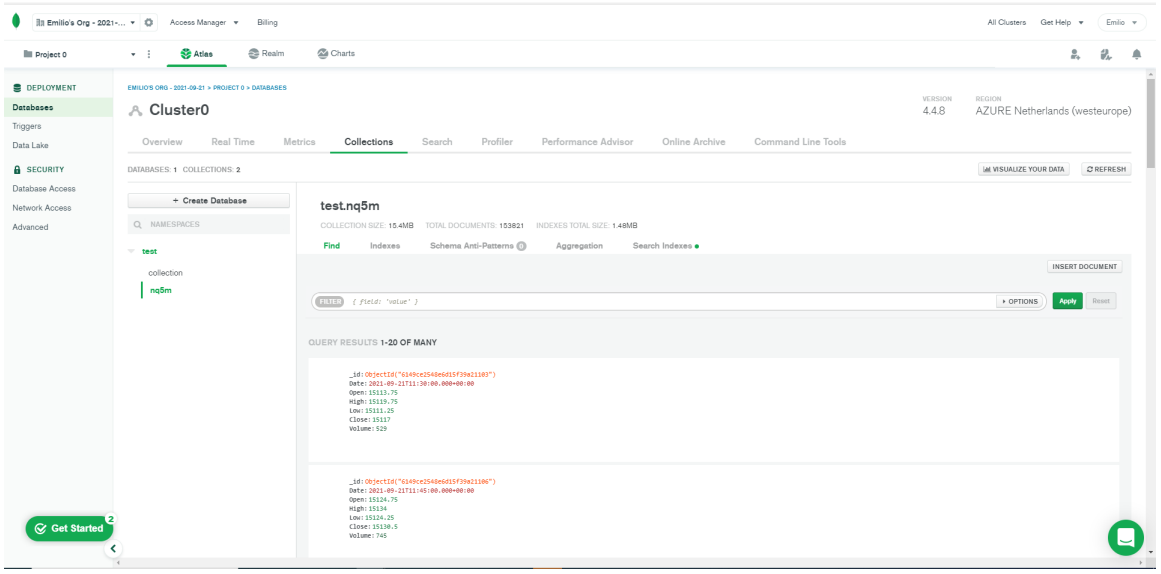


Figura 51: Representación de la zona de trabajo MongoDB

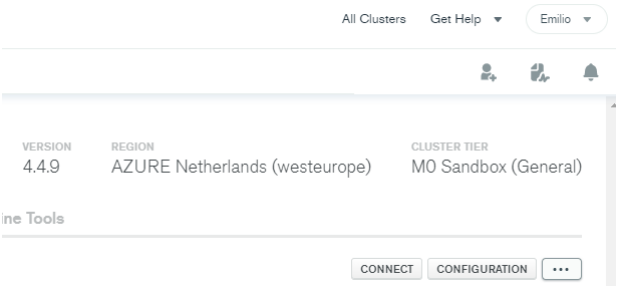


Figura 52: Zona Geográfica ubicación Cluster y Proveedor

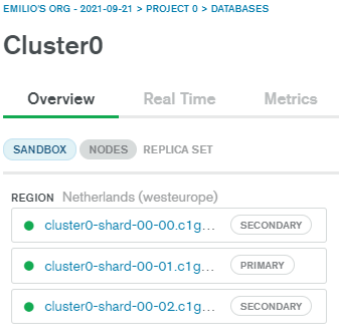


Figura 53: Estructura del cluster

Todo el proceso de creación del clúster es transparente, por los que en un breve espacio de tiempo se puede tener esta parte del proyecto en marcha.

En lo relativo a los datos almacenados, aprovechando la mencionada flexibilidad por la ausencia de esquemas predefinidos, se ha optado por crear una colección por cada activo y marco temporal (p.e NQ5m), y se encarga de almacenar los datos reales del mercado para que los modelos puedan realizar las predicciones.

Los documentos guardan una marca temporal y los valores *Open, High, Low, Close, Volume*. Las predicciones realizadas por los distintos modelos también son guardadas con idéntica estructura para su posterior estudio y revisión.

9.2. Esquema e Implementación de la solución

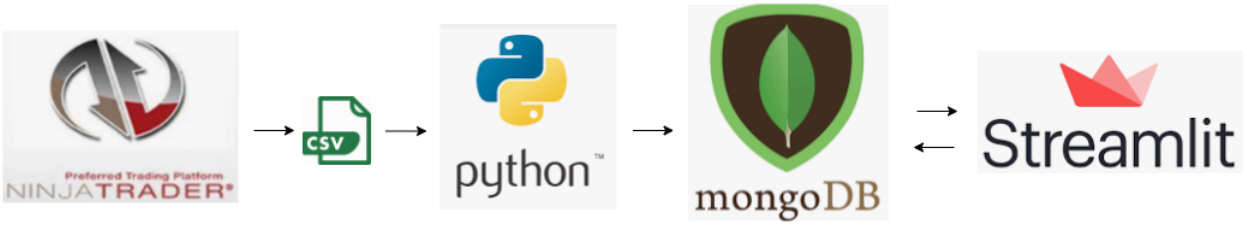
La solución elegida para la presentación de los resultados también requiere una serie de argumentaciones dada la gran variedad de opciones existentes en el mercado. Por una parte, se plantea una opción de integración dentro de las propias plataformas, que constaría de una API básica que sería llamada desde cada aplicación con sus propios recursos. Así pues, se podría realizar componente básico en cada plataforma, que sería un indicador en Ninjatrader 8, un *custom study* en Sierrachart, y así sucesivamente que se encargaría de solicitar la información y mostrarla en cada plataforma con sus recursos propios. Cada persona utilizaría cualquiera de las opciones salvo que se diera la situación de no estar disponible. En este caso, se observa que este enfoque limita mucho el alcance, además de haber un condicionamiento adicional por parte de las políticas internas de cada una de las soluciones software. En una fecha determinada puede ser factible el desarrollo y explotación pero en un futuro puede estar sujeto a un desembolso económico si cambian las condiciones contractuales.

Es por ello, que dado que desde un inicio se ha intentando generar un recurso independiente, flexible y que no esté sujeto a muchas particularidades que lo limite, se optó ya en las fases iniciales por utilizar alguno de los framework de desarrollo disponibles que dejarían la propuesta como una aplicación web. De este modo, independiente de la plataforma de trading utilizada se accede al recurso, además de dejar una vía abierta para, si se considera necesario, implementar la citada API que permita integrar todo el trabajo en algunas plataformas específicas si, tras evaluarlo, aporta un valor añadido.

Dentro del gran abanico de opciones existentes en la actualidad en el mercado, se preseleccionaron 4 alternativas, como son Flask, Django, Heroku y Streamlit. De algunas se tenían algunas nociones básicas, pero hubo una serie de factores tras realizar un pequeño análisis inicial que hicieron seleccionar *Streamlit* por encima de las restantes:

- Se trata de un framework cuyo propósito original es la explotación en aplicaciones de ciencia de datos, por lo que agiliza en gran medida muchos aspectos.
- Se dispone de una liviana curva de aprendizaje, por lo que lo hace una opción que genera productos acabados de una forma rápida y sencilla.
- Dispone también al igual que MongoDB de varios modelos de publicación de aplicaciones, local, nube propia o soluciones cloud estándar. También es fácilmente integrable en contenedores o cualquier otra alternativa de similares características.
- Permite la fácil integración de otras librerías, de manera que trabajos realizados en otros entornos como *Jupyter Notebooks* son rápidamente integrados.

En este caso, se ha creado una presentación de la información sencilla y ágil para ir consultando los cálculos obtenidos de los modelos en los distintos activos. A continuación se muestra un esquema de la estructura necesaria para tener la solución completa:



La secuencia detallada sería la siguiente:

1. **Ninjatrader.** Este es el primer elemento necesario de la solución final. Lo principal que se necesita es acceso a datos en tiempo real y en estas fechas, el precio de disponer de dicho servicio mediante API que faciliten su acceso desde el código está al alcance de empresas importantes o equipos de trabajo con un volumen considerable. A nivel de personas que operan a nivel discrecional, con unos capitales de dimensiones que no asusten al ver, hay que buscar alternativas para poder obtener dicha información. Así pues, esta aplicación es la que nos permite obtener los datos tal cual se van generando en el mercado, al igual que nos permitía obtener los históricos para la fase de generación de los modelos. La única y principal funcionalidad necesaria es, en este caso, la de facilitar los últimos datos del mercado para que los modelos generados anteriormente realicen la predicción. Para tal fin, se crea un indicador que cada vez que cierra un periodo de 5 minutos, vuelca los datos de esa vela en cuestión.

   El código asociado a dicho indicador, escrito en *Ninjascript*, versión reducida de *C#*, se encuentra en el Anexo A, siendo el método **OnBarUpdate()** el que contiene la lógica de negocio ya que el resto en este caso es necesario pero ajeno a la resolución del problema.

2. **Fichero CSV.**Se trata de un fichero separado por ';' generado desde el apartado anterior. En este fichero se vuelcan los datos relativos a periodos de tiempo anterior velas en Ninjatrader), que serán utilizados por los modelos para realizar las predicciones. Ejemplo de contenido del citado fichero sería:

   ```
   17/09/2021 22:00:00;15321,75;15339,75;15313,5;15323,5;17934
   17/09/2021 22:05:00;15323,5;15325,75;15298,5;15310;9144
   17/09/2021 22:10:00;15309,75;15313;15303,25;15305,5;2502
   17/09/2021 22:15:00;15305;15307,75;15299,5;15302;1961
   ```

   Si se revisa el código fuente del indicador, se observa que dichos valores corresponden con una estructura *Marca Temporal; Valor apertura; Máximo; Mínimo; Cierre; Volumen operado.*

3. **Código Python: Ninjatrader a MongoDB**. Este componente se encarga de subir el contenido del fichero CSV a la base de datos MongoDB. Su funcionamiento se basa en una carga inicial de n valores que se va repitiendo según el marco temporal en el que se opere. En este caso, como se utiliza 5 minutos como referencia temporal, se repite el proceso y se almacena en la nube el nuevo valor una vez se ha escrito en el fichero por parte de Ninjatrader.

   El código asociado a ese punto, volcado de historial en MongoDB, volcado de n últimos elementos y repetición cada 5 minutos de volcado a base de datos del último valor se encuentra en el Anexo B.

4. **MongoDB.** Es el punto de conexión de todos los elementos. Tal y como se ha detallado anteriormente, se ha optado por una implementación en la nube, mediante los recursos propios del proveedor, a través de *Azure* mediante un clúster creado de forma automática. El documento estándar que se repite en todas las colecciones tendrá la estructura del siguiente ejemplo:

   ```
   {"_id":{"$oid":"6149ce2548e6d15f39a21103"},
   "Date":{"$date":{"$numberLong":"1632223800000"}},
   "Open":{"$numberDouble":"15113.75"},
   "High":{"$numberDouble":"15119.75"},
   "Low":{"$numberDouble":"15111.25"},
   "Close":{"$numberDouble":"15117.0"},
   "Volume":{"$numberInt":"529"}}
   ```

   Se crean tantas colecciones como activos y marcos temporales diferentes hay en el sistema.

5. **Streamlit.** Se trata de la capa más externa de la implementación. Es, tal y como se ha detallado anteriormente, una herramienta sencilla que permite de una forma rápida, flexible y eficaz crear paneles de información con los resultados de los modelos seleccionados. En este caso, se detectan 2 zonas

principales en la pantalla. La parte izquierda nos permite seleccionar los activos y el marco temporal, mostrando los gráficos resultantes en la parte derecha. En este caso, sólo se dispone de un activo y un marco temporal, pero es fácilmente ampliable generando nuevos modelos. Las representaciones de las series temporales resultantes son elementos creados mediante la librería gráfica *Altair*, ofrecen características propias totalmente ajenas a *Streamlit* como interactividad que permite la manipulación del gráfico. Un ejemplo de captura de este apartado sería:

Dentro de este componente se ha incluido también la lógica de negocia relativa a la realización de los pronósticos por medio de los modelos. Se planteó inicialmente que dicha carga residiera también en el punto 3 (Python), pero es uno de los objetivos a posteriori, con el fin de simplificar la solución, encontrar un mecanismo que permita insertar directamente en la base de datos desde la plataforma de trading, aunque a día de hoy y con las opciones planteadas no es viable.

Así pues, este último elemento se encarga de obtener los datos del sistema de gestión de bases de datos, realizar los cálculos oportunos y realizar la visualización. No encajaría dentro de paradigmas como MVC (Modelo-Vista-Controlador) pero en este caso, dado que el esquema definitivo no está confirmado se considera lo más adecuado. De este modo, solo necesitamos acceder a la url del aplicación y seleccionar los elementos que nos interese. El código fuente en cuestión, asociado a la aplicació *Streamlit* se encuentra disponible en el Apéndice D.

10. Migración a la nube de la solución

El constante aumento de prestaciones por parte de los sistemas de información, así como el avance a todo lo relativo en el mercado de las telecomunicaciones, hacen posible una serie de modelos de explotación que eran impensables hace no mucho tiempo. Cualquier desarrollo realizado a día de hoy requiere también de un enfoque en el que el despliegue en la nube de cualquier solución sea obligadamente tenida en cuenta, por las que son sus principales ventajas:

- Eliminación de una fuerte inversión inicial en equipamiento y/o infraestructuras
- Flexibilidad a la hora de modificar los recursos existentes sin necesidad de desechar nada.
- Esquema de pago también más flexible, por lo que un sistema de pago por uso hace el producto más atractivo.
- Movilidad.

Este proyecto no ha perdido nunca de vista esta perspectiva y prueba de ello es que es independiente por completo del escenario de explotación: local, nube o híbrido. Obviamente, lo más recomendable, además de ser parte del contenido del máster, es buscar una puesta en marcha que libere de ataduras de ubicación física para, de este modo, hacerlo más aprovechable. En cualquier caso, llevar una solución de este estilo de un ámbito más doméstico a cualquier planteamiento más profesionalizado, hace impensable cualquier puesta en marcha que no sea en la nube con las ventajas y garantías que ofrece.

Teniendo en cuenta todo lo anteriormente citado, se procede pues a detallar en cada caso que sería necesario para proceder a la creación del entorno de manera que toda su explotación se lleve a cabo en la nube sin la necesidad de recursos domésticos.

Se tendría que agrupar en 2 bloques principales, uno por razones obvias detalladas en apartados anteriores y otro que comprende a la parte restante del sistema.

En lugar, como primer bloque, se encuentra el relativo a la base de datos. En este caso, y dadas las alternativas disponibles, se ha optado trabajar directamente en la nube ya que la gran cantidad de ventajas ofrecidas por MongoDB para este proyecto en particular, lo hacen como óptima y casi única solución. La alternativa pasaría por realizar una configuración doméstica que ofrece peor rendimiento, escalabilidad y funcionalidad. Así pues, la simple creación de una cuenta con los datos personales o de la empresa ya da acceso a una instancia en la nube totalmente funcional para las prestaciones necesarias por el proyecto.

Por otra parte, el resto de componentes, como son la plataforma de trading que obtiene los datos, el script que lo pasa a la base de datos para hacerlo accesible por parte del último elemento, la aplicación web con el framework *Streamlit*, se pueden alojar en una máquina virtual con sistema operativo Windows(necesario para la plataforma Ninjatrader) y desde ahí publicar la aplicación. La latencia adicional generada por subir los datos a otro host que aloja MongoDB, y volver a descargarlos, es compensada sobradamente por la total ausencia de gestión del motor de bases de datos. Al tratarse de unos requisitos de productividad muy intensiva, cualquier propuesta que cumpla los requisitos mínimos del desarrollador de Ninjatrader 8 será suficiente. Según sus especificaciones, sirve cualquier máquina que cumpla:

- Windows 8, Windows 10, Windows Server 2012.
- 1 gigahertz (GHz) or faster 32-bit or 64-bit processor.
- 2GB RAM.
- Microsoft .NET Framework 4.8 (pre-installed on most PC's and can be downloaded here: Microsoft .NET Framework)

Como se puede ver, no son unas características muy exigentes para configurar en cualquier entorno virtual tipo *Azure* o similar.

11. Conclusiones

La realización de este trabajo he supuesto un gran reto debido a la complejidad derivada de la limitación de recursos del publico objetivo que se encargará de explotarlo. Durante el desarrollo del máster se ha ido profundizando en contenidos de una forma inducida para trabajar con datos y poder ver y determinar en cada momento la importancia de la materia trabajada. En este caso, para realizar el planteamiento del trabajo se han tenido en cuenta las inquietudes personales primándolas por encima de la viabilidad o conveniencia para su desarrollo.

Así pues, todo el proceso de toma de decisiones inicial, la ubicación, el alcance y la dimensión han supuesto ya en sí un handicap a la hora de su planteamiento, puesto que había una gran cantidad de puntos por concretar, evaluar y probar antes de poder seguir adelante con la simple definición del problema. Muchas horas valorando herramientas, muchas horas realizando y desechando pruebas hasta que se encuentra la propuesta adecuada todo con la intención de llegar a una solución que, aunque es básica, cubre todo el proceso que se planteó como objetivo inicial. En teoría hay un cómputo de horas estimado para la realización de todo el trabajo, pero en este caso se ha sobrepasado con creces, aunque en ningún caso ha supuesto un sacrificio ya que toda la dedicación en este caso ha sido con la finalidad de, además de cumplir unos requisitos y objetivos planteados en la asignatura, una satisfacción por estar relacionado con inquietudes personales, dejadas de lado desde hace tiempo por temas de conciliación familiar.

Se han aplicado conocimientos adquiridos en un entorno real y sin red de seguridad ante una posible caída, en un escenario nada propicio para ello, ya que es evidente que los mercados financieros no son para nada un ente organizado y estructurado, sino todo lo contrario. Hay muchos intereses de grandes empresas detrás de cualquier movimiento, y desarrollar algún recurso que permita a personas que desde sus limitados equipamientos puedan sacar alguna ventaja de la potencia de computación ya es un hecho más que satisfactorio. No se trata ya de hacer frente a las llamadas "manos fuertes"que mueven el mercado, pero sí detectar posibles circunstancias que hagan estar en menor desventaja frente a éstas y hacer de este mundo incierto y probabilístico algo más predecible y menos estresante. Tampoco se busca crear, como lo llaman los expertos, un Santo Grial, algo que predice todos los movimientos en cualquier momento con un margen de error mínimo, sino de disponer de una herramienta adicional que en colaboración con otra estrategia consigan una esperanza matemática positiva, que conjuntamente con una gestión monetaria y emocional adecuada puede traducirse en un beneficio no ya económico, sino también emocional como es el simple hecho de intentar hacer las cosas de una forma organizada.

Como se ha indicado anteriormente, ha supuesto un gran esfuerzo, como lo suele ser un trabajo de estas dimensiones y más coexistiendo con un entorno familiar en el que la llegada de la adolescencia, con todos sus extras adicionales a la ya difícil tarea de conciliar, ha permitido salir de la zona de rutina (en ningún caso de confort), exprimir capacidades personales y volver a tener sensaciones totalmente olvidadas desde la época universitaria. Por desgracia, el mercado laboral en la mayoría de los casos, exige unos ratios de productividad y obtención de resultados que se alejan considerablemente del rigor académico que exige la demostración de adquisición de conocimientos y su puesta en práctica de forma adecuada, con unos criterios iniciales más allá del "prueba y error".

En lo relativo a los objetivos iniciales cumplidos, se considera que por una parte la aplicación de conocimientos adquiridos en gran cantidad de asignaturas han permitido desarrollar una solución muy transversal y completa. Quedaría como mejora en un futuro el trabajo ya centrado y enfocado exclusivamente a la mejora de los resultados de las métricas, tanto mediante la manipulación del conjunto de datos, como mediante el ajuste de los modelos.

12. Bibliografía

Brownlee, J. (2018) Introduction to Time Series Forecasting with Python. machinelearningmastery.com

Brownlee, J. (2019) Deep Learning for Time Series Forecasting. machinelearningmastery.com

Geron, A. (2019) Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'REILLY.

Korstanje, J. (2021) Advanced Forecasting with Python: With State-of-the-Art-Models Including LSTMs, Facebook's Prophet, and Amazon's DeepAR. Apress

Torres, J. (2020) Python Deep Learning: Introducción práctica con Keras y TensorFlow 2. Marcombo 75. **Jansen, S.** (2020) Machine Learning for Algorithmic Trading: Predictive models to extract signals from market and alternative data for systematic trading strategies with Python, 2nd Edition. Packt.

Cabana, E.;Gomila, J.G.(2020) Curso avanzado de Series Temporales con R y Python. www.udemy.com

Richards, T. (2021) Getting Started with Streamlit for Data Science: Create and deploy Streamlit web applications from scratch in Python. Packt.

Williams, T; Diaz Valdecantos, E.(trad) (2019) Master the Markets: Aprendiendo un enfoque profesional de trading e inversión bajo las ideas del "Volume Spread Analysis". DEUSTO.

Dagget, S. Ninjascript Programmer's Launch Pad. Smashwords.

A. Indicador Ninjatrader a CSV

```
#region Using declarations
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Input;
using System.Windows.Media;
using System.Xml.Serialization;
using NinjaTrader.Cbi;
using NinjaTrader.Gui;
using NinjaTrader.Gui.Chart;
using NinjaTrader.Gui.SuperDom;
using NinjaTrader.Gui.Tools;
using NinjaTrader.Data;
using NinjaTrader.NinjaScript;
using NinjaTrader.Core.FloatingPoint;
using NinjaTrader.NinjaScript.DrawingTools;
#endregion

//This namespace holds Indicators in this folder and is required. Do not change it.
using System.IO;
namespace NinjaTrader.NinjaScript.Indicators
{
  public class Imprimir : Indicator
  {
    private StreamWriter sw;
    protected override void OnStateChange()
    {
      if (State == State.SetDefaults)
      {
        Description                 = @"Generación CSV";
        Name                    = "Imprimir";
        Calculate                  = Calculate.OnBarClose;
        IsOverlay                  = false;
        DisplayInDataBox              = true;
        DrawOnPricePanel              = true;
        DrawHorizontalGridLines            = true;
        DrawVerticalGridLines            = true;
        PaintPriceMarkers              = true;
        ScaleJustification              = NinjaTrader.Gui.Chart.ScaleJustification.Right;
        //Disable this property if your indicator requires custom values that cumulate
        // with each new market data event.
        //See Help Guide for additional information.
        IsSuspendedWhileInactive          = true;
      }
      else if (State == State.Configure)
      {
      }
```

```
    }

    protected override void OnMarketData(MarketDataEventArgs marketDataUpdate)
    {

    }

    protected override void OnMarketDepth(MarketDepthEventArgs marketDepthUpdate)
    {

    }

    protected override void OnBarUpdate()
    {
      sw = File.AppendText("d:/generados/"
        +Bars.Instrument.MasterInstrument.Name
        +Bars.Instrument.MasterInstrument.GetNextExpiry(DateTime.Now).ToString("MM-yy")
        +".csv");  // Open the path for writing
      sw.WriteLine(Time[0] + ";" + Open[0] + ";" + High[0] + ";" + Low[0] + ";"
        + Close[0] + ";" + Volume[0]); // Append a new line to the file
      sw.Close(); // Close the file to allow future calls to access the file again.
      //Print(High[1]+";"+Low[1]+";"+Open[1]+";"+Close[1]);
    }
  }
}

#region NinjaScript generated code. Neither change nor remove.

namespace NinjaTrader.NinjaScript.Indicators
{
  public partial class Indicator : NinjaTrader.Gui.NinjaScript.IndicatorRenderBase
  {
    private Imprimir[] cacheImprimir;
    public Imprimir Imprimir()
    {
      return Imprimir(Input);
    }

    public Imprimir Imprimir(ISeries<double> input)
    {
      if (cacheImprimir != null)
        for (int idx = 0; idx < cacheImprimir.Length; idx++)
          if (cacheImprimir[idx] != null &&  cacheImprimir[idx].EqualsInput(input))
            return cacheImprimir[idx];
      return CacheIndicator<Imprimir>(new Imprimir(), input, ref cacheImprimir);
    }
  }
}

namespace NinjaTrader.NinjaScript.MarketAnalyzerColumns
{
  public partial class MarketAnalyzerColumn : MarketAnalyzerColumnBase
  {
    public Indicators.Imprimir Imprimir()
    {
```

```
      return indicator.Imprimir(Input);
    }

    public Indicators.Imprimir Imprimir(ISeries<double> input )
    {
      return indicator.Imprimir(input);
    }
  }
}

namespace NinjaTrader.NinjaScript.Strategies
{
  public partial class Strategy : NinjaTrader.Gui.NinjaScript.StrategyRenderBase
  {
    public Indicators.Imprimir Imprimir()
    {
      return indicator.Imprimir(Input);
    }

    public Indicators.Imprimir Imprimir(ISeries<double> input )
    {
      return indicator.Imprimir(input);
    }
  }
}

#endregion
```

B. Código Ninjatrader a MongoDB

```
##############################################################################################
# Conexión a la base de datos MongoDB para realizar el volcado de los últimos datos que #
# permitirá realizar el pronóstico de los valores                                          #
##############################################################################################
import pymongo
client = pymongo.MongoClient("mongodb+srv://U:P@cluster0.c1gnb.mongodb.net/test?retryWrites=true&w=ma
db = client.test
db.drop_collection("nq5m")
db.create_collection("nq5m")
db.nq5m.insert_many(dft.to_dict('records'))

## Carga de N valores anteriores al último
## Primero los 25 anteriores al último y ya el último en bucle
columnas=['Date','Open','High','Low','Close','Volume']
mydateparser = lambda x: pd.datetime.strptime(x, "%d/%m/%Y %H:%M:%S")
df_rt = pd.read_csv("d:\\generados\\NQ12-21.csv",sep=';',header=None,
    parse_dates=True,index_col=0,decimal=",",names=columnas, date_parser=mydateparser)
ultimosN = df_rt.tail(25)
ultimosN = ultimosN[:-1]
ultimosN.reset_index(inplace=True)
db.nq5m.insert_many(ultimosN.to_dict('records'))

############################
## Carga del último valor. Este código se repite cada 5 minutos(en este caso para
## añadir el último valor)
############################
import time

while(True):
    columnas=['Date','Open','High','Low','Close','Volume']
    mydateparser = lambda x: pd.datetime.strptime(x, "%d/%m/%Y %H:%M:%S")
    df_rt = pd.read_csv("d:\\generados\\NQ12-21.csv",sep=';',header=None
        ,parse_dates=True,index_col=0,decimal=",",names=columnas, date_parser=mydateparser)
    ultimo = df_rt.tail(1)
    ultimo.reset_index(inplace=True)
    db.nq5m.insert_many(ultimo.to_dict('records'))
    time.sleep(300)
```

C. Código Jupyter generación de modelos

Modelos

September 25, 2021

[ ]:

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

[ ]:
```
from numpy.random import seed
seed(1)
import tensorflow
tensorflow.random.set_seed(2)

from IPython.core.display import display, HTML
display(HTML("<style>.container { width:100% !important; }</style>"))


import os
directorio = "tfm_pics"   #Comprobamos la carpeta para guardar imagenes␣
 ↪generadas entrenamiento

try:
  os.stat(directorio)
except:
  os.mkdir(directorio)

import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.metrics import mean_squared_error
from tensorflow import keras
# univariate mlp example
from numpy import array
from keras.models import Sequential
from keras.layers import Dense
from sklearn.preprocessing import MinMaxScaler


n_steps = 3
activo="NQ"
timeframe="5m"
```

```
# ubicacion="./drive/MyDrive/tfm/"

ubicacion="https://github.com/emiliosegarra/tfm/raw/main/"
ruta=ubicacion+activo+timeframe+".txt"

df = pd.read_csv(ruta)
borrar=[' # of Trades', ' OHLC Avg', ' HLC Avg', ' HL Avg', ' Bid Volume',' Ask␣
 ↪Volume', ' Volume.1']
df=df.drop(borrar,axis=1)
columnas=['Date','Time','Open','High','Low','Close','Volume']
df.columns = columnas
df['Fecha'] = pd.to_datetime(df['Date']+df['Time'])
df=df.drop(['Date','Time'],axis=1)
df=df.set_index("Fecha")

# Realizamos la transformación de la serie temporal en muestras
def split_sequence(sequence, n_steps):
    X, y = list(), list()
    for i in range(len(sequence)):
        end_ix = i + n_steps

        if end_ix > len(sequence)-1:
            break
        # Parte la secuencia en componentes
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
        X.append(seq_x)
        y.append(seq_y)
    return array(X), array(y)
```

[ ]:

[ ]:
```
def plot_history(history,nombre='tfm_pics/grafico.png',titulo='X'):
    hist = pd.DataFrame(history.history)
    hist['epoch'] = history.epoch
    plt.figure()
    plt.title = titulo
    plt.xlabel('Epoch')
    plt.ylabel('MSE')
    plt.plot(hist['epoch'],hist['mse'],label='Train Error')
    plt.plot(hist['epoch'],hist['val_mse'],label='Val Error')
    plt.ylim([0,2000])
    plt.legend()
    plt.grid()
    plt.title = titulo
    plt.savefig(nombre,dpi=200)
    plt.show()
```

[ ]:

[ ]:

1 MLP

1.1 High, Low, Close

[ ]:
```
MLP_mse=[]    # Guardamos los mse asociados a cada serie temporal High Low y␣
 ↪Close
MLP={}        #Guardamos los modelos
history={}    #La evolución del entrenamiento
MLPx={}       #La X del entrenamiento de cada serie
MLPy={}       #La y del entrenamiento de cada serie
MLPxt={}      #La X del test de cada serie
MLPyt={}      #La y del test de cada serie

fields = ['High','Low','Close']
early_stop = tf.keras.callbacks.EarlyStopping(monitor='mse',patience=20)
MLPhat={}   #Guardamos las predicciones de High, Low, Close
num_epoch=100
for f in fields:
    dftmp = df[f]
    print("======================")
    print("Calculando MLP-%s"%(f))
    print("======================")

    raw_seq_all = dftmp
    #Limitamos el conjunto de datos de entrenamiento a los últimos meses
    raw_seq_all = raw_seq_all[60000:]

    raw_seq_all = [ x for x in raw_seq_all]

    raw_seq = raw_seq_all[:70000]
    raw_seq_test = raw_seq_all[70000:]
    X, y = split_sequence(raw_seq, n_steps)
    MLP[f] = Sequential()
    MLP[f].add(Dense(100, activation='relu', input_dim=n_steps))
    MLP[f].add(Dense(1))
    MLP[f].compile(optimizer='adam', loss='mse',metrics=['mse','mae'])

    history[f]=MLP[f].fit(X, y, epochs=num_epoch,validation_split=0.2,␣
 ↪callbacks=[early_stop],verbose=1)
    Xt, yt = split_sequence(raw_seq_test, n_steps)
    MLPhat[f] = MLP[f].predict(Xt, verbose=0)
```

```
    MLP_mse.append(mean_squared_error(yt,MLPhat[f]))
    MLP[f].save("MLP_%s.h5"%(f))
    MLPx[f] = X
    MLPy[f] = y
    MLPxt[f] = Xt
    MLPyt[f] =yt
```

[ ]:
```
MLP_mse
```

[ ]:
```
MLP['High'].summary()
```

[ ]:
```
hist = pd.DataFrame(history['Close'].history)
hist['epoch']= history['Close'].epoch
hist.tail()
```

[ ]:

[ ]:
```
plot_history(history['High'],titulo='Error Entrenamiento-Validación␣
↪High',nombre='tfm_pics/MLPHerror.png')
```

[ ]:
```
plot_history(history['Low'],nombre='tfm_pics/MLPLerror.png')
```

[ ]:
```
plot_history(history['Close'],titulo='Error Entrenamiento-Validación␣
↪High',nombre='tfm_pics/MLPCerror.png')
```

[ ]:

[ ]:
```
plt.figure()

valores = [i for i in range(0,100)]
plt.xlabel('Tiempo')
plt.ylabel('Valor')
plt.plot(valores,MLPyt['High'][0:100],label='Valor real')
plt.plot(valores,MLPhat['High'][0:100],label='Predicción')
plt.legend()
plt.grid()
plt.savefig('tfm_pics/CompMLP_H-R.png',dpi=200)
plt.show()
```

[ ]:
```
plt.figure()

valores = [i for i in range(0,100)]
plt.xlabel('Tiempo')
plt.ylabel('Valor')
plt.plot(valores,MLPyt['Low'][0:100],label='Valor real')
plt.plot(valores,MLPhat['Low'][0:100],label='Predicción')
plt.legend()
plt.grid()
plt.savefig('tfm_pics/CompMLP_L-R.png',dpi=200)
plt.show()
```

[ ]:
```
plt.figure()

valores = [i for i in range(0,100)]
plt.xlabel('Tiempo')
plt.ylabel('Valor')
plt.plot(valores,MLPyt['Close'][0:100],label='Valor real')
plt.plot(valores,MLPhat['Close'][0:100],label='Predicción')
plt.legend()
plt.grid()
plt.savefig('tfm_pics/CompMLP_C-R.png',dpi=200)
plt.show()
```

2 Vanilla LSTM

[ ]:
```
from keras.layers import LSTM
n_features = 1



Vanilla_mse=[]
Vanilla={}
Vanilla_history={}
Vanillax={}       #La X del entrenamiento de cada serie
Vanillay={}       #La y del entrenamiento de cada serie
Vanillaxt={}      #La X del test de cada serie
Vanillayt={}      #La y del test de cada serie
fields = ['High','Low','Close']
early_stop = tf.keras.callbacks.EarlyStopping(monitor='mse',patience=20)
Vanillahat={}   #Guardamos las predicciones de High, Low, Close
num_epoch=50
for f in fields:
    dftmp = df[f]
    print("======================")
    print("Calculando Vanilla-%s"%(f))
    print("======================")

    raw_seq_all = dftmp
    #Limitamos el conjunto de datos de entrenamiento a los últimos meses
    raw_seq_all = raw_seq_all[60000:]

    raw_seq_all = [ x for x in raw_seq_all]

    raw_seq = raw_seq_all[:70000]
    raw_seq_test = raw_seq_all[70000:]
    X, y = split_sequence(raw_seq, n_steps)
    Xt, yt = split_sequence(raw_seq_test, n_steps)
```

```
    Vanilla[f] = Sequential()
    Vanilla[f].add(LSTM(50, activation='relu', input_shape=(n_steps,␣
↪n_features)))
    Vanilla[f].add(Dense(1))
    Vanilla[f].compile(optimizer='adam', loss='mse',metrics=['mse','mae'])

    Xlstm = X.reshape((X.shape[0], X.shape[1], n_features))
    XlstmT = Xt.reshape((Xt.shape[0], Xt.shape[1], n_features))
    Vanilla_history[f]=Vanilla[f].fit(Xlstm, y,␣
↪epochs=num_epoch,validation_split=0.2, callbacks=[early_stop],verbose=1)
    Vanillax[f] = Xlstm
    Vanillaxt[f] = XlstmT
    Vanillay[f]=y
    Vanillayt[f] = yt


    Vanillahat[f] = Vanilla[f].predict(XlstmT, verbose=0)

    Vanilla_mse.append(mean_squared_error(yt,Vanillahat[f]))
    Vanilla[f].save("Vanilla_%s.h5"%(f))
```

[ ]:
```
plot_history(Vanilla_history['High'],titulo='Error Entrenamiento-Validación␣
↪High',nombre='tfm_pics/VanillaHerror.png')
```

[ ]:
```
plot_history(Vanilla_history['Low'],titulo='Error Entrenamiento-Validación␣
↪High',nombre='tfm_pics/VanillaLerror.png')
```

[ ]:
```
plot_history(Vanilla_history['Close'],titulo='Error Entrenamiento-Validación␣
↪High',nombre='tfm_pics/VanillaCerror.png')
```

[ ]:
```
import matplotlib.pyplot as plt
id = [x for x in range(50)]
plt.plot(id,Vanillahat['Close'][1150:1200])
plt.plot(id,yt[1150:1200])
plt.show()
```

[ ]:
```
Vanilla_mse
```

[ ]:
```
plt.figure()

valores = [i for i in range(0,100)]
plt.xlabel('Tiempo')
plt.ylabel('Valor')
plt.plot(valores,Vanillayt['High'][0:100],label='Valor real')
plt.plot(valores,Vanillahat['High'][0:100],label='Predicción')
plt.legend()
plt.grid()
plt.savefig('tfm_pics/CompVanilla_H-R.png',dpi=200)
```

```
plt.show()
```

[ ]:
```
plt.figure()

valores = [i for i in range(0,100)]
plt.xlabel('Tiempo')
plt.ylabel('Valor')
plt.plot(valores,Vanillayt['Low'][0:100],label='Valor real')
plt.plot(valores,Vanillahat['Low'][0:100],label='Predicción')
plt.legend()
plt.grid()
plt.savefig('tfm_pics/CompVanilla_L-R.png',dpi=200)
plt.show()
```

[ ]:
```
plt.figure()

valores = [i for i in range(0,100)]
plt.xlabel('Tiempo')
plt.ylabel('Valor')
plt.plot(valores,Vanillayt['Close'][0:100],label='Valor real')
plt.plot(valores,Vanillahat['Close'][0:100],label='Predicción')
plt.legend()
plt.grid()
plt.savefig('tfm_pics/CompVanilla_C-R.png',dpi=200)
plt.show()
```

[ ]:
```
Vanilla['High'].summary()
```

3 Stacked LSTM

[ ]:
```
from keras.layers import LSTM
n_features = 1



Stacked_mse=[]
Stacked={}
Stacked_history={}
Stackedx={}       #La X del entrenamiento de cada serie
Stackedy={}       #La y del entrenamiento de cada serie
Stackedxt={}      #La X del test de cada serie
Stackedyt={}      #La y del test de cada serie
fields = ['High','Low','Close']
early_stop = tf.keras.callbacks.EarlyStopping(monitor='mse',patience=20)
Stackedhat={}   #Guardamos las predicciones de High, Low, Close
num_epoch=50
for f in fields:
```

```
    dftmp = df[f]
    print("======================")
    print("Calculando Stacked-%s"%(f))
    print("======================")

    raw_seq_all = dftmp
    #Limitamos el conjunto de datos de entrenamiento a los últimos meses
    raw_seq_all = raw_seq_all[60000:]

    raw_seq_all = [ x for x in raw_seq_all]

    raw_seq = raw_seq_all[:70000]
    raw_seq_test = raw_seq_all[70000:]
    X, y = split_sequence(raw_seq, n_steps)
    Xt, yt = split_sequence(raw_seq_test, n_steps)
    Stacked[f] = Sequential()
    Stacked[f].add(LSTM(50, activation='relu',␣
↪return_sequences=True,input_shape=(n_steps, n_features)))
    Stacked[f].add(LSTM(50, activation='relu'))
    Stacked[f].add(Dense(1))
    Stacked[f].compile(optimizer='adam', loss='mse',metrics=['mse','mae'])

    Xlstm = X.reshape((X.shape[0], X.shape[1], n_features))
    XlstmT = Xt.reshape((Xt.shape[0], Xt.shape[1], n_features))
    Stacked_history[f]=Stacked[f].fit(Xlstm, y,␣
↪epochs=num_epoch,validation_split=0.2, callbacks=[early_stop],verbose=1)

    Stackedx[f] = Xlstm
    Stackedy[f]= y
    Stackedxt[f]= XlstmT
    Stackedyt[f] = yt



    Stackedhat[f] = Stacked[f].predict(XlstmT, verbose=0)

    Stacked_mse.append(mean_squared_error(yt,Stackedhat[f]))
    Stacked[f].save("Stacked_%s.h5"%(f))
```

[ ]:
```
plot_history(Stacked_history['High'],nombre='tfm_pics/StackedHerror.png')
```

[ ]:
```
plot_history(Stacked_history['Low'],nombre='tfm_pics/StackedLerror.png')
```

[ ]:
```
plot_history(Stacked_history['Close'],nombre='tfm_pics/StackedCerror.png')
```

[ ]:
```
Stacked_mse
```

[ ]:
```
Stacked['High'].summary()
```

[ ]:
```
plt.figure()

valores = [i for i in range(0,100)]
plt.xlabel('Tiempo')
plt.ylabel('Valor')
plt.plot(valores,Stackedyt['High'][0:100],label='Valor real')
plt.plot(valores,Stackedhat['High'][0:100],label='Predicción')
plt.legend()
plt.grid()
plt.savefig('tfm_pics/CompStacked_H-R.png',dpi=200)
plt.show()
```

[ ]:
```
plt.figure()

valores = [i for i in range(0,100)]
plt.xlabel('Tiempo')
plt.ylabel('Valor')
plt.plot(valores,Stackedyt['Low'][0:100],label='Valor real')
plt.plot(valores,Stackedhat['Low'][0:100],label='Predicción')
plt.legend()
plt.grid()
plt.savefig('tfm_pics/CompStacked_L-R.png',dpi=200)
plt.show()
```

[ ]:
```
plt.figure()

valores = [i for i in range(0,100)]
plt.xlabel('Tiempo')
plt.ylabel('Valor')
plt.plot(valores,Stackedyt['Close'][0:100],label='Valor real')
plt.plot(valores,Stackedhat['Close'][0:100],label='Predicción')
plt.legend()
plt.grid()
plt.savefig('tfm_pics/CompStacked_C-R.png',dpi=200)
plt.show()
```

[ ]:
```
Stacked['High'].summary()
```

4 Bidirectional LSTM

[ ]:
```
from keras.layers import Bidirectional
from keras.layers import LSTM
n_features = 1



MiBidirectional_mse=[]
```

```
MiBidirectional={}
MiBidirectional_history={}
MiBidirectionalx={}      #La X del entrenamiento de cada serie
MiBidirectionaly={}      #La y del entrenamiento de cada serie
MiBidirectionalxt={}     #La X del test de cada serie
MiBidirectionalyt={}     #La y del test de cada serie
fields = ['High','Low','Close']
early_stop = tf.keras.callbacks.EarlyStopping(monitor='mse',patience=20)
MiBidirectionalhat={}   #Guardamos las predicciones de High, Low, Close
num_epoch=50
for f in fields:
    dftmp = df[f]
    print("====================")
    print("Calculando Bidirectional-%s"%(f))
    print("====================")

    raw_seq_all = dftmp
    #Limitamos el conjunto de datos de entrenamiento a los últimos meses
    raw_seq_all = raw_seq_all[60000:]

    raw_seq_all = [ x for x in raw_seq_all]

    raw_seq = raw_seq_all[:70000]
    raw_seq_test = raw_seq_all[70000:]
    X, y = split_sequence(raw_seq, n_steps)
    Xt, yt = split_sequence(raw_seq_test, n_steps)
    MiBidirectional[f] = Sequential()
    MiBidirectional[f].add(Bidirectional(LSTM(50, activation='relu'),␣
 ↪input_shape=(n_steps, n_features)))
    MiBidirectional[f].add(Dense(1))
    MiBidirectional[f].compile(optimizer='adam',␣
 ↪loss='mse',metrics=['mse','mae'])

    Xlstm = X.reshape((X.shape[0], X.shape[1], n_features))
    XlstmT = Xt.reshape((Xt.shape[0], Xt.shape[1], n_features))
    MiBidirectional_history[f]=MiBidirectional[f].fit(Xlstm, y,␣
 ↪epochs=num_epoch,validation_split=0.2, callbacks=[early_stop],verbose=1)

    MiBidirectionalx[f] =  Xlstm
    MiBidirectionalxt[f] = XlstmT
    MiBidirectionaly[f] = y
    MiBidirectionalyt[f] = yt



    MiBidirectionalhat[f] = MiBidirectional[f].predict(XlstmT, verbose=0)
```

```
    MiBidirectional_mse.append(mean_squared_error(yt,MiBidirectionalhat[f]))
    MiBidirectional[f].save("Bidirectional_%s.h5"%(f))
```

[ ]:
```
MiBidirectional_mse
```

[ ]:
```
plot_history(MiBidirectional_history['High'],nombre='tfm_pics/
↪MiBidirectionalHerror.png')
```

[ ]:
```
plot_history(MiBidirectional_history['Low'],nombre='tfm_pics/
↪MiBidirectionalLerror.png')
```

[ ]:
```
plot_history(MiBidirectional_history['Close'],nombre='tfm_pics/
↪MiBidirectionalCerror.png')
```

[ ]:
```
plt.figure()

valores = [i for i in range(0,100)]
plt.xlabel('Tiempo')
plt.ylabel('Valor')
plt.plot(valores,MiBidirectionalyt['High'][0:100],label='Valor real')
plt.plot(valores,MiBidirectionalhat['High'][0:100],label='Predicción')
plt.legend()
plt.grid()
plt.savefig('tfm_pics/CompMiBidirectional_H-R.png',dpi=200)
plt.show()
```

[ ]:
```
plt.figure()

valores = [i for i in range(0,100)]
plt.xlabel('Tiempo')
plt.ylabel('Valor')
plt.plot(valores,MiBidirectionalyt['Low'][0:100],label='Valor real')
plt.plot(valores,MiBidirectionalhat['Low'][0:100],label='Predicción')
plt.legend()
plt.grid()
plt.savefig('tfm_pics/CompMiBidirectional_L-R.png',dpi=200)
plt.show()
```

[ ]:
```
plt.figure()

valores = [i for i in range(0,100)]
plt.xlabel('Tiempo')
plt.ylabel('Valor')
plt.plot(valores,MiBidirectionalyt['Close'][0:100],label='Valor real')
plt.plot(valores,MiBidirectionalhat['Close'][0:100],label='Predicción')
plt.legend()
plt.grid()
plt.savefig('tfm_pics/CompMiBidirectional_C-R.png',dpi=200)
plt.show()
```

[ ]:
```
MiBidirectional['High'].summary()
```

5 CNN

[ ]:
```
from keras.layers import Conv1D, MaxPooling1D, Flatten

from keras.layers import LSTM
n_features = 1



MiCNN_mse=[]
MiCNN={}
MiCNN_history={}
MiCNNx={}       #La X del entrenamiento de cada serie
MiCNNy={}       #La y del entrenamiento de cada serie
MiCNNxt={}      #La X del test de cada serie
MiCNNyt={}      #La y del test de cada serie
fields = ['High','Low','Close']
early_stop = tf.keras.callbacks.EarlyStopping(monitor='mse',patience=20)
MiCNNhat={}   #Guardamos las predicciones de High, Low, Close
num_epoch=50
for f in fields:
    dftmp = df[f]
    print("======================")
    print("Calculando MiCNN-%s"%(f))
    print("======================")

    raw_seq_all = dftmp
    #Limitamos el conjunto de datos de entrenamiento a los últimos meses
    raw_seq_all = raw_seq_all[60000:]

    raw_seq_all = [ x for x in raw_seq_all]

    raw_seq = raw_seq_all[:70000]
    raw_seq_test = raw_seq_all[70000:]
    X, y = split_sequence(raw_seq, n_steps)
    Xt, yt = split_sequence(raw_seq_test, n_steps)
    MiCNN[f] = Sequential()
    MiCNN[f].add(Conv1D(filters=64, kernel_size=2, activation='relu',␣
 ↪input_shape=(n_steps,n_features)))
    MiCNN[f].add(MaxPooling1D(pool_size=2))
    MiCNN[f].add(Flatten())
    MiCNN[f].add(Dense(50, activation='relu'))
    MiCNN[f].add(Dense(1))
    MiCNN[f].compile(optimizer='adam', loss='mse',metrics=['mse','mae'])

    Xlstm = X.reshape((X.shape[0], X.shape[1], n_features))
    XlstmT = Xt.reshape((Xt.shape[0], Xt.shape[1], n_features))
```

```
    MiCNN_history[f]=MiCNN[f].fit(Xlstm, y, epochs=num_epoch,validation_split=0.
↪2, callbacks=[early_stop],verbose=1)
    MiCNNx[f] = Xlstm
    MiCNNxt[f] = XlstmT
    MiCNNy[f] = y
    MiCNNyt[f] = yt

    MiCNNhat[f] = MiCNN[f].predict(XlstmT, verbose=0)

    MiCNN_mse.append(mean_squared_error(yt,MiCNNhat[f]))
    MiCNN[f].save("MiCNN_%s.h5"%(f))
```

[ ]:
```
plot_history(MiCNN_history['High'],nombre='tfm_pics/MiCNNHerror.png')
```

[ ]:
```
plot_history(MiCNN_history['Low'],nombre='tfm_pics/MiCNNLerror.png')
```

[ ]:
```
plot_history(MiCNN_history['Close'],nombre='tfm_pics/MiCNNCerror.png')
```

[ ]:
```
MiCNN_mse
```

[ ]:
```
plt.figure()

valores = [i for i in range(0,100)]
plt.xlabel('Tiempo')
plt.ylabel('Valor')
plt.plot(valores,MiCNNyt['High'][0:100],label='Valor real')
plt.plot(valores,MiCNNhat['High'][0:100],label='Predicción')
plt.legend()
plt.grid()
plt.savefig('tfm_pics/CompMiCNN_H-R.png',dpi=200)
plt.show()
```

[ ]:
```
plt.figure()

valores = [i for i in range(0,100)]
plt.xlabel('Tiempo')
plt.ylabel('Valor')
plt.plot(valores,MiCNNyt['Low'][0:100],label='Valor real')
plt.plot(valores,MiCNNhat['Low'][0:100],label='Predicción')
plt.legend()
plt.grid()
plt.savefig('tfm_pics/CompMiCNN_L-R.png',dpi=200)
plt.show()
```

[ ]:
```
plt.figure()

valores = [i for i in range(0,100)]
plt.xlabel('Tiempo')
plt.ylabel('Valor')
plt.plot(valores,MiCNNyt['Close'][0:100],label='Valor real')
```

```
plt.plot(valores,MiCNNhat['Close'][0:100],label='Predicción')
plt.legend()
plt.grid()
plt.savefig('tfm_pics/CompMiCNN_C-R.png',dpi=200)
plt.show()
```

[ ]:
```
MiCNN['High'].summary()
```

[ ]:

[ ]:

[ ]:

[ ]:
```
lista_mse = [ MLP_mse, Vanilla_mse, Stacked_mse, MiBidirectional_mse, MiCNN_mse␣
↪]
```

[ ]:
```
df_mse = pd.DataFrame(lista_mse, columns = ['High','Low','Close'],␣
↪index=['MLP', 'Vanilla', 'Stacked', 'Bidirectional','CNN'])
```

[ ]:
```
df_mse.values
```

[ ]:

[ ]:
```
ax = df_mse.plot.bar()
ax.figure.savefig('tfm_pics/mse_todos.png',dpi=200)
```

[ ]:

[ ]:
```
df_mse.to_latex()
```

[ ]:

[ ]:
```
for f in fields:
    plt.figure()
    tam=50
    valores = [i for i in range(tam)]
    plt.xlabel('Tiempo')
    plt.ylabel('Valor')

    plt.title='Comparatión resultados Serie Temporal %s'%(f)
    plt.plot(valores,MiCNNyt[f][0:tam],label='Valor real')
    plt.plot(valores,MLPhat[f][0:tam],label='MLP')
    plt.plot(valores,Vanillahat[f][0:tam],label='Vanilla LSTM')
    plt.plot(valores,Stackedhat[f][0:tam],label='Stacked LSTM')
    plt.plot(valores,MiBidirectionalhat[f][0:tam],label='Bidirectional LSTM')
    plt.plot(valores,MiCNNhat[f][0:tam],label='CNN')
    plt.legend()
    plt.grid()
    plt.savefig('tfm_pics/Comp_%s.png'%(f),dpi=200)
    plt.show()
```

[ ]:

[ ]:
```
%%capture
!wget -nv https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('Modelos.ipynb')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id
=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redire
ct_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&scope=email%20https%3a%2f%2fwww.googl
eapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%2
0https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2
fwww.googleapis.com%2fauth%2fpeopleapi.readonly%20https%3a%2f%2fwww.googleapis.c
om%2fauth%2fdrive.activity.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2f
experimentsandconfigs%20https%3a%2f%2fwww.googleapis.com%2fauth%2fphotos.native&
response_type=code

Enter your authorization code:
4/1AX4XfWgNNqPIrLVyQHIcAQGvHF1sZJGhTpTxSlP9SzUYDLTqv2Ecrg9komc
```

[ ]:

D. Código fuente *Streamlit*

```
import streamlit as st
import pandas as pd
import numpy as np
import plotly.express as px
import pymongo
import altair as alt
import plotly.graph_objects as go
from tensorflow import keras
from keras.models import load_model
import re
import datetime
from datetime import timedelta


# Funcion que devuelve el ticker entre paréntesis del activo de la lista
def getTicker(cad):
    m = re.match("[A-Za-z]+ \(([A-Za-z]+)\)", cad)
    if m:
        return m.group(1).lower()
#En el caso de que no se cumpla la "realidad necesaria" de las velas hay que reajustar. High es el ma
def Normaliza(valores):
    #Al realizar la estimación, hay valores que pueden quedar fuera de rango, por lo que ajustamos hi
    valores[1] = max(valores)
    valores[2] = min(valores)


def main():
    menu = ["Nasdaq (NQ)"]
    timeframe = ["5m"]
    choice_activo = st.sidebar.selectbox("Activos",menu)
    choice_tf = st.sidebar.selectbox("Marco Temporal",timeframe)
    st.title("Pronósticos Redes Neuronales")


    # Conexión con la base de datos para obtener los datos a descargar
    client = pymongo.MongoClient("mongodb+srv://tfmunedesm:Ab123456@cluster0.c1gnb.mongodb.net/test?r
    db = client.test
    resul=db.nq5m.find().sort("Date",-1).limit(20)

    dfdb = pd.DataFrame(resul)
    dfdb.drop("_id",axis=1,inplace=True)
    dftmp = dfdb[['Date','Open','High','Low','Close']]
    dftmp.set_index('Date',inplace=True)
    dftmp.sort_index(inplace=True)
    st.dataframe(dftmp)


    # Inicializacion de diccionarios que contendra los modelos
    fields = ['High','Low','Close']
    valoresP = {} #Almacena los n ultimos valores para pasar al modelo y hacer el pronostico
    model_mlp = {}
    model_vanilla = {}
    model_stacked = {}
```

```
model_bidirectional = {}
model_cnn = {}
#Carga de modelos y valores para las predicciones
for f in fields:
    valoresP[f] = np.array( [dftmp.iloc[-3][f], dftmp.iloc[-2][f], dftmp.iloc[-1][f]])
    model_mlp[f] = load_model("modelos/"+getTicker(choice_activo)+"_"+choice_tf+"/MLP_"+f+".h5")
    model_vanilla[f] = load_model("modelos/"+getTicker(choice_activo)+"_"+choice_tf+"/Vanilla_"+f
    model_stacked[f] = load_model("modelos/"+getTicker(choice_activo)+"_"+choice_tf+"/Stacked_"+f
    model_bidirectional[f] = load_model("modelos/"+getTicker(choice_activo)+"_"+choice_tf+"/MiBid
    model_cnn[f] = load_model("modelos/"+getTicker(choice_activo)+"_"+choice_tf+"/MiCNN_"+f+".h5"


#Realizacion de predicciones.
mlpp = []
vanillap = []
stackedp = []
bidirectionalp = []
cnnp = []

mlpp.append(dftmp.iloc[-1]['Close'])
vanillap.append(dftmp.iloc[-1]['Close'])
stackedp.append(dftmp.iloc[-1]['Close'])
bidirectionalp.append(dftmp.iloc[-1]['Close'])
cnnp.append(dftmp.iloc[-1]['Close'])

for f in fields:
    mival = model_mlp[f].predict(valoresP[f].reshape((1, 3)),verbose=0)
    mlpp.append(float(mival[0][0]))

    mival = model_vanilla[f].predict(valoresP[f].reshape((1, 3, 1)),verbose=0)
    vanillap.append(float(mival[0][0]))
    mival = model_stacked[f].predict(valoresP[f].reshape((1, 3, 1)),verbose=0)
    stackedp.append(float(mival[0][0]))
    mival = model_bidirectional[f].predict(valoresP[f].reshape((1, 3, 1)),verbose=0)
    bidirectionalp.append(float(mival[0][0]))
    mival = model_cnn[f].predict(valoresP[f].reshape((1, 3, 1)),verbose=0)
    cnnp.append(float(mival[0][0]))

#Nos aseguramos del cumplimiento de High valor máximo y Low valor mínimo de los 4 de la vela
Normaliza(mlpp)
Normaliza(vanillap)
Normaliza(stackedp)
Normaliza(bidirectionalp)
Normaliza(cnnp)



# Creación de los dataframes con las predicciones MLP
st.write("## MLP")
df_mlp = dftmp.copy()
df_mlp.reset_index(inplace=True)
mlpp.insert(0,df_mlp.iloc[-1]['Date']+timedelta(minutes=5))
dfaux = pd.DataFrame(mlpp)
dfaux=dfaux.transpose()
dfaux.columns=['Date','Open','High','Low','Close']
```

```
df_mlp=df_mlp.append(dfaux)
df_mlp.set_index("Date")


# Representación del DataFrame MLP
base = alt.Chart(df_mlp).encode(
alt.X('Date:T', axis=alt.Axis(labelAngle=-45)),
color=alt.condition("datum.Open <= datum.Close",
                    alt.value("#06982d"), alt.value("#ae1325")))

chart = alt.layer(base.mark_rule().encode(alt.Y('Low:Q', title='Price',
                                scale=alt.Scale(zero=False)), alt.Y2('High:Q')), base.mark_bar()
st.altair_chart(chart, use_container_width=True)

# Creación de los dataframes con las predicciones Vanilla
st.write("## Vanilla LSTM")
df_vanilla = dftmp.copy()
df_vanilla.reset_index(inplace=True)
vanillap.insert(0,df_vanilla.iloc[-1]['Date']+timedelta(minutes=5))
dfaux = pd.DataFrame(vanillap)
dfaux=dfaux.transpose()
dfaux.columns=['Date','Open','High','Low','Close']

df_vanilla=df_vanilla.append(dfaux)
df_vanilla.set_index("Date")

# Representación del DataFrame Vanilla
base = alt.Chart(df_vanilla).encode(
alt.X('Date:T', axis=alt.Axis(labelAngle=-45)),
color=alt.condition("datum.Open <= datum.Close",
                    alt.value("#06982d"), alt.value("#ae1325")))

chart = alt.layer(base.mark_rule().encode(alt.Y('Low:Q', title='Price',
                                scale=alt.Scale(zero=False)), alt.Y2('High:Q')), base.mark_bar()
st.altair_chart(chart, use_container_width=True)

modeloMLPH = keras.models.load_model("./Documents/MLPH.h5")

# Creación de los dataframes con las predicciones Stacked
st.write("## Stacked LSTM")
df_stacked = dftmp.copy()
df_stacked.reset_index(inplace=True)
stackedp.insert(0,df_stacked.iloc[-1]['Date']+timedelta(minutes=5))
dfaux = pd.DataFrame(stackedp)
dfaux=dfaux.transpose()
dfaux.columns=['Date','Open','High','Low','Close']

df_stacked=df_stacked.append(dfaux)
df_stacked.set_index("Date")
# Representación del DataFrame Stacked
base = alt.Chart(df_stacked).encode(
alt.X('Date:T', axis=alt.Axis(labelAngle=-45)),
color=alt.condition("datum.Open <= datum.Close",
                    alt.value("#06982d"), alt.value("#ae1325")))
```

```
    chart = alt.layer(base.mark_rule().encode(alt.Y('Low:Q', title='Price',
                                    scale=alt.Scale(zero=False)), alt.Y2('High:Q')), base.mark_bar()
    st.altair_chart(chart, use_container_width=True)

    # Creación de los dataframes con las predicciones Bidirectional
    st.write("## Bidirectional LSTM")
    df_bidirectional = dftmp.copy()
    df_bidirectional.reset_index(inplace=True)
    bidirectionalp.insert(0,df_bidirectional.iloc[-1]['Date']+timedelta(minutes=5))
    dfaux = pd.DataFrame(bidirectionalp)
    dfaux=dfaux.transpose()
    dfaux.columns=['Date','Open','High','Low','Close']
    # Representación del DataFrame Bidirectional
    df_bidirectional=df_bidirectional.append(dfaux)
    df_bidirectional.set_index("Date")
    base = alt.Chart(df_bidirectional).encode(
    alt.X('Date:T', axis=alt.Axis(labelAngle=-45)),
    color=alt.condition("datum.Open <= datum.Close",
                        alt.value("#06982d"), alt.value("#ae1325")))

    chart = alt.layer(base.mark_rule().encode(alt.Y('Low:Q', title='Price',
                                    scale=alt.Scale(zero=False)), alt.Y2('High:Q')), base.mark_bar()
    st.altair_chart(chart, use_container_width=True)

    # Creación de los dataframes con las predicciones CNN
    st.write("## CNN")
    df_cnn = dftmp.copy()
    df_cnn.reset_index(inplace=True)
    cnnp.insert(0,df_cnn.iloc[-1]['Date']+timedelta(minutes=5))
    dfaux = pd.DataFrame(cnnp)
    dfaux=dfaux.transpose()
    dfaux.columns=['Date','Open','High','Low','Close']

    df_cnn=df_cnn.append(dfaux)
    df_cnn.set_index("Date")
    # Representación del DataFrame CNN LSTM
    base = alt.Chart(df_cnn).encode(
    alt.X('Date:T', axis=alt.Axis(labelAngle=-45)),
    color=alt.condition("datum.Open <= datum.Close",
                        alt.value("#06982d"), alt.value("#ae1325")))

    chart = alt.layer(base.mark_rule().encode(alt.Y('Low:Q', title='Price',
                                    scale=alt.Scale(zero=False)), alt.Y2('High:Q')), base.mark_bar()
    st.altair_chart(chart, use_container_width=True)




if __name__ == "__main__":
    main()
```