Máster en Ciencias Actuariales y Financieras

TRABAJO FINAL DE MÁSTER

Título: Predicción de precios de activos financieros con *deep learning* en un entorno de *trading* de Alta Frecuencia y una gestión activa de riesgos.

Autoría: Juan Carlos Forero Carreño

Tutoría: Luis Ortiz Gracia

Curso académico: 2020-2021

UNIVERSITAT DE BARCELONA
Facultat d'Economia i Empresa

Màster de Ciències Actuarials i Financeres

Facultad de Economía y Empresa

Universidad de Barcelona

Trabajo Final de Máster

Máster en Ciencias Actuariales y Financieras

Predicción de precios de activos financieros con *deep learning* en un entorno de *trading* de Alta Frecuencia y una gestión activa de riesgos

Autoría: Juan Carlos Forero Carreño

Tutoría: Luis Ortiz Gracia

"El contenido de este documento es de exclusiva responsabilidad del autor, quien declara que no ha incurrido en plagio y que la totalidad de referencias a otros autores han sido expresadas en el texto".

Predicción de precios de activos financieros con *deep learning* en un entorno de *trading* de Alta Frecuencia y una gestión activa de riesgos

Juan Carlos Forero Carreño

7 junio de 2021

Resumen

El pronóstico de series financieras siempre ha sido un tópico de gran interés, ya que modelar los precios de los activos implica un gran reto. Esto se debe a que, en general, las series financieras tienen un comportamiento aleatorio y desde un punto de vista estrictamente econométrico, en muchas ocasiones, no podemos pronosticar los precios de los activos, pues es ruido blanco. De acuerdo con lo anterior, lo que busca este trabajo final de máster es realizar un montaje exploratorio de series financieras en periodos de muy corto tiempo y aplicar técnicas de *deep learning* como redes neuronales, para predecir el precio de los activos desde una perspectiva de clasificación y regresión y hacer el despliegue de esta estrategia de *trading* de alta frecuencia un entorno lo más real posible. De igual manera, se desea introducir el cálculo de medidas de riesgo como el valor en riesgo VaR y la pérdida esperada —ES— (*expected shortfall*) en cada uno de los instantes, para la gestión activa de riesgos de este portafolio. También se espera utilizarlo como una medida de riesgo para los *stop loss*.

Palabas clave: Trading de Alta Frecuencia, Deep Learning, —ES— (expected shortfall), Redes Neuronales, Python, LSTM, Redes Convolucionales.

Abstract

The forecast of financial time series has always been a topic of great interest since modeling asset prices implies a great challenge. This is because, in general, financial series have a random behavior and in an econometric point of view, in many occasions, we cannot foresee asset prices, as it is white noise. In accordance with the above, what this final master's thesis seeks is to carry out an exploratory study of financial series in very short periods of time and apply deep learning techniques such as neural networks, to predict the price of assets from a classification and regression perspective and make the deployment of this high frequency trading strategy as realistic as possible. Similarly, we want to introduce the calculation of risk measures such as Value at Risk (VaR) and the expected Shortfall —ES— at each moment, to actively manage the risks of this portfolio. It is also expected to be used as a risk measure for stop losses.

Tabla de contenido

1. INTRODUCCIÓN

Los mercados financieros están evolucionando hacia un entorno mucho más tecnológico, en donde la ejecución de las órdenes muchas veces ya no es ejecutada por una persona, sino por un algoritmo (Aldridge, 2013). Esto se ha dado en mayor medida por la aplicación de ciencias de la computación y la comunicación en los mercados financieros, lo que genera un gran nivel de tecnificación. Según diferentes informes, entre los que se encuentra el estudio realizado por el instituto Alan Turin (Buchanan, 2019), se estima que entre el 60 y 73 % de las operaciones diarias son llevadas a cabo por algún tipo de algoritmo automatizado. Esto de alguna manera incrementa la liquidez del mercado y disminuye los costes de *trading*; sin embargo, también se ha dicho que la tecnificación completa de los mercados financieros con algoritmos e inteligencia artificial puede ser perjudicial si no se toman las precauciones necesarias.

El *trading* de alta frecuencia, (HFT) por sus siglas en inglés, es definido por Roger Romero (Morral, 2020) en su tesis doctoral como un tipo de *trading* donde las decisiones de inversión tienen un horizonte temporal más pequeño, minutos o incluso segundos, allí las determinaciones de inversión deben ser ejecutadas por algoritmos precisamente por la estrechez de ese horizonte temporal. El *trading* de alta frecuencia usualmente se basa en la información pasada para realizar alguna decisión de inversión futura. Durante un largo tiempo se han utilizado modelos como la regresión lineal, para poder predecir el comportamiento de los precios de los activos financieros; no obstante, como se sabe, las series financieras tienen comportamientos no lineales muy complejos, que no pueden ser captados por un modelo lineal, por lo cual resulta natural pensar en utilizar algoritmos no lineales como las redes neuronales para obtener mejores predicciones.

Por lo anterior, las redes neuronales recurrentes (RNN) y las redes neuronales de memoria a largo y corto plazo (LSTM) son una buena herramienta para hacer una predicción con series financieras, pues estas redes en particular tienen en cuenta la información pasada y, a partir de esta información, ejecutan relaciones más complejas entre los datos y generan predicciones de mayor acierto con arquitecturas más complicadas como evidencia (Arthur Le Guennec, 2016).

También es de suma importancia tener métricas de riesgos en las inversiones y poder proteger el patrimonio ante una variación de mercado muy abrupta. Ahora bien, ya que el riesgo no es observable, se debe aproximar mediante alguna herramienta. Esta aproximación es la varianza. Para términos de este trabajo de final de máster, se explorará la estimación de variables de pérdida para activos financieros.

Para aproximar el riesgo de los activos comprados por el algoritmo, se emplearán el valor en riesgos (VaR) y la pérdida esperada (ES) (*Expected Shortfall*), de manera que podamos calcular el riesgo a un cuantil determinado y también explorar la profundidad de las colas de la distribución de pérdida.

1.1. Marco teórico

1.1.1. Trading algorítmico y de alta frecuencia

Tanto la banca como los mercados financieros se están tecnificando a pasos agigantados con el uso de diferentes tecnologías y tratan de tomar partida de la gran cantidad de datos que se generan en cada momento. En particular, como se ha visto en los últimos años, han surgido cientos de bancos electrónicos, plataformas de *trading* y de gestión de activos, que buscan democratizar la banca a las personas del común y ofrecer servicios con menores costos de transacción. Así pues, la utilización de grandes cantidades de información y la utilización de esta para la generación de algún tipo de *output* cada vez se hace más común.

Ahora bien, la disponibilidad de *big data*, tecnologías de computación en la nube y *hardware* optimizado han sido los impulsores clave de la última ola de innovación en inteligencia artificial. Las capacidades de IA y el aprendizaje automático (ML) están impulsando el crecimiento en el mercado emergente de *fintech*. En términos generales, fintech describe las nuevas tecnologías, servicios y empresas que han cambiado los servicios financieros (Buchanan, 2019). Entre los productos más relevantes encontramos las criptomonedas, *blockchain*, *robo-advising*, contratos inteligentes, *crowdfunding*, pagos móviles y plataformas de inteligencia artificial. En 2017, la IA encabezó la lista como tendencia clave en servicios financieros y *fintech*.

Para nuestro caso en particular, el *trading* de alta frecuencia (HFT), toma cada día más importancia en los mercados financieros internacionales, gracias a la gran cantidad de información disponible para crear y entrenar modelos de diferentes índoles. Se estima que entre el 60 y 70 % de todas las operaciones realizadas en los mercados de acciones —60 % del mercado de futuros y un 50 % en el mercado de *treasury bonds* americanos— son realizadas por algún tipo de algoritmo. Además, se estima que para el 2026 el valor de las operaciones diarias realizadas por algoritmos alcance US 21.52 billones tomando como base el 2019 con una cifra promedio de US 9.53 billones (Research, 2020).

De acuerdo con lo anterior, es posible hacer una definición más formal de *trading* algorítmico, que, a saber, es la implementación de un algoritmo con una lógica en particular para comprar o vender un activo en particular. De esta dinámica se deriva el *trading* de alta frecuencia (HFT), pues dada la rapidez con la que llega la información, la cantidad y el poco tiempo del que se dispone para realizar una decisión de inversión, esta es tomada por un ordenador previamente programado con una lógica o un algoritmo (Morral, 2020), porque es este el que tiene la capacidad de procesar esa gran cantidad de datos en milisegundos y tomar una decisión determinada.

Este tipo de sistemas automatizados de *trading* tienen diferentes ventajas, entre las que destacamos cinco enunciadas en el *Manual de Técnicas para el High Frecuency Trading* de Irene Aldridge (Aldridge, 2013). La primera ventaja alude a la realización de compras o ventas en los mejores niveles posibles. La segunda se refiere a tener la habilidad de comprobar diferentes condiciones de mercado simultáneamente. Tercera y cuarta, la capacidad de aumentar la precisión y reducir la probabilidad de cometer errores y, finalmente, reducir los sesgos cognitivos de los *traders* impulsados por el miedo o la felicidad (Buchanan, 2019). Así mismo, entre los defensores de esta perspectiva, se destaca que el gran volumen de negociación genera condiciones de mercado más eficientes y una liquidez mayor, lo que reduce los *spreads* de transacción y los costes involucrados. Por su parte, los detractores de este tipo de iniciativas sugieren que estos

algoritmos propenden al error en los precios, pues hay errores transitorios en estos, dada la gran cantidad de operaciones en un muy pequeño periodo de tiempo que no tiene sentido económico aparente (Aldridge, 2013).

1.1.2. Introducción al *machine learning*

El aprendizaje automático, de manera muy general, es una serie de métodos computacionales, en los cuales se utiliza la experiencia para hacer predicciones acertadas. La experiencia puede ser definida como la información disponible para el aprendiz, que, de manera usual, toma la forma de información electrónica recolectada para dicho análisis, en donde la calidad y la cantidad de la información es de suma importancia para la calidad de las predicciones. El éxito de estos algoritmos se basa en los datos utilizados y hay una inherente correlación entre esta disciplina, el análisis de datos y la estadística. Las técnicas de *machine learning* son orientadas hacia los datos, combinando conceptos fundamentales de la ciencia de la computación, ideas de la estadística básica, la probabilidad y la optimización (Mehryar Mohri, 2018).

De manera más específica y como lo de define el equipo de desarrollo de SAS, el *machine learning* es una serie de métodos para el análisis de datos y la construcción automática de modelos, por lo mismo, es una rama de la inteligencia artificial basada en la premisa de que un sistema en particular puede aprender de datos, identificar patrones y tomar decisiones con mínima intervención humana (SAS, 2021). En esta disciplina encontramos una gran cantidad de algoritmos que dependen de la tarea que queremos realizar y la complejidad de esta. Oriol Pujol en su libro *Introducción al machine learning* (Vila, 2014) divide los algoritmos de aprendizaje en tres grupos.

Aprendizaje supervisado

Este tipo de algoritmos aprenden de un conjunto de datos con etiquetas que generalizan todos los datos de entrada, es decir, los datos de entrada vienen preprocesados y son conocidos por el usuario. En esta categoría encontramos algoritmos como la regresión lineal y las *support vector machines*.

Aprendizaje no supervisado

En este tipo de aprendizaje, los datos de entrada vienen sin etiquetas y es el mismo algoritmo el que los categoriza de acuerdo con algún criterio estadístico. Entre los algoritmos más conocidos en esta rama encontramos algoritmos de *clustering* como el *K-means* y kernel density estimation (KDE).

Aprendizaje por refuerzo (*reinforcement learning*)

Estos algoritmos aprenden por medio del refuerzo, de acuerdo con un criterio seleccionado exploran el espacio de soluciones y por medio del refuerzo, buscan las mejores características para definirlo.

En los grupos anteriores podemos definir varias categorías que van en función del tipo de problema que se quiere solucionar y su complejidad. En general, cuando hay un conjunto relativamente pequeño y finito, se utilizan algoritmos de clasificación como si la respuesta fuera sí o no, o un numero discreto de posibilidades. Por otro lado, cuando se quiere predecir un número real, se utilizan principalmente algoritmos de regresión. Por ejemplo, predecir el precio de una casa de acuerdo con su metraje es más un modelo de regresión,

mientras que si una casa se vende de acuerdo con su barrio es más un modelo de clasificación.

En este caso en particular y por la naturaleza de los datos, nos vamos a enfocar en algoritmos de refuerzo, pues mantienen la estructura temporal de las series y aprenden de estas relaciones; sin embargo, también hay que recalcar que serán utilizados desde un punto de vista de regresión y de clasificación, si el precio sube o baja y el retorno específico. De acuerdo con lo anterior, este trabajo se centrará en tres algoritmos formales (*multi layer perceptrón* (MLP), *recurrent neural networks* (RNN) y *long short-term memory networks* (LSTM)), ya que estos, por su configuración, vienen muy bien para las series temporales y tienen de alguna manera una estructura de refuerzo entre los datos pasados y los datos futuros, lo cual para series de tiempo es justamente lo que buscamos: intentar hacer relaciones pasadas con el comportamiento futuro del precio de un activo.

1.1.3. Aprendizaje profundo *(Deep Learning*) y redes neuronales

Redes neuronales

De acuerdo con la definición de Jordi Vitrià (2020) en su *Introducción al deep learning*, las redes neuronales son un paradigma de programación inspirado en funciones biológicas que busca proveer la capacidad de aprender a los ordenadores.

Deep learning

Este término se refiere a una serie de técnicas que ayudan a aprender a las redes neuronales de una manera más eficiente (Vitrià, 2020).

Estas dos definiciones combinadas son una poderosa herramienta para hallar soluciones a problemas que antes eran casi imposibles de resolver por su intrincada estructura, además de que ofrecen gran eficiencia computacional.

TEORIA

En general todas las redes neuronales se componen de tres elementos fundamentales para su funcionamiento:

a) Función de pérdida:

Es la función $f_{(x,y)}(W)$ objetivo que queremos minimizar, que mide las diferencias entre los datos generados por el modelo y los datos reales.

Funciones

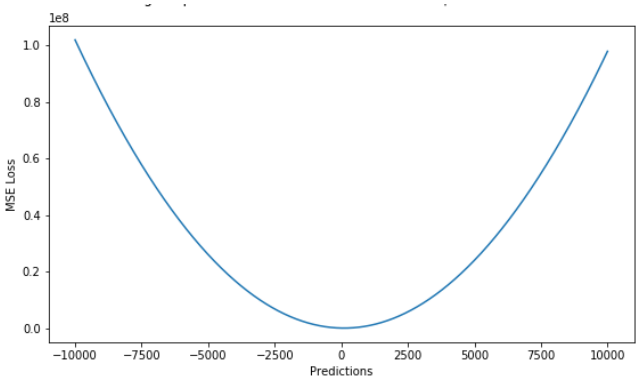
En este apartado mostraremos algunas de las funciones de pérdida más comunes y utilizadas para la optimización de modelos de *machine learning* y su utilidad.

- **Función cuadrática de pérdida euclidiana**

Esta función es usualmente utilizada en los problemas clásicos de regresión, donde la idea es minimizar las diferencias cuadradas medias de la función de aproximación con la observación real, donde y_i es la observación real, $f(x_i)$ la función de predicción y n el número de ejemplos.

$$L(y, f(x)) = \frac{1}{n}\sum_i (y_i - f(x_i))^2$$

Ilustración 1:Función cuadrática de pérdida euclidiana



Fuente: Elaboración propia a partir de datos propios

Funciones de pérdida para métodos de clasificación

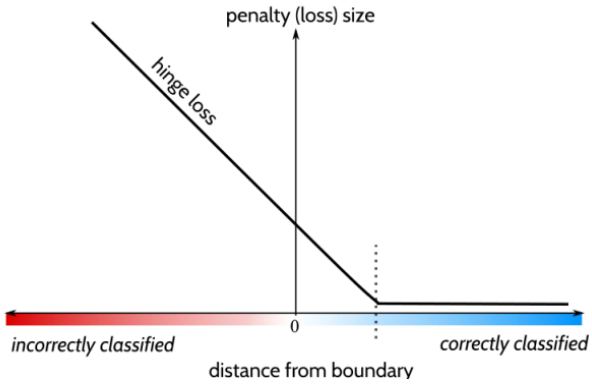
Función de regresión modificada: con esta función obtenemos probabilidades y no números, donde y_i es la observación real, $f(x_i)$ la función de predicción, usualmente la función sigmoidea, y n el número de ejemplos.

$$L(y, f(x)) = \frac{1}{n}\sum_i (1 - y_i f(x_i))^2$$

Función de pérdida de *hinge*: se utiliza en problemas mayormente de clasificación, de manera específica en algoritmos como las *support vector machines*.

$$L(y, f(x)) = \frac{1}{n}\sum_i \max(0, 1 - y_i f(x_i))$$

Ilustración 2: Función de pérdida de hinge

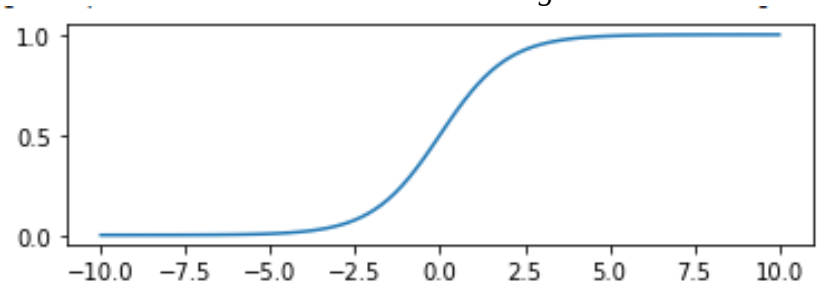


Fuente: tomado de https://towardsdatascience.com/a-definitive-explanation-to-hinge-loss-for-support-vector-machines-ab6d8d3178f1

Función logística: es una de las funciones más clásicas y de donde comenzaron muchos de los algoritmos de aprendizaje; sin embargo, como veremos más adelante, no es una buena aproximación, pues destruye el gradiente.

$$L\big(y, f(x)\big) = \frac{1}{n} log\big(1 + \exp\left(-y_i f(x_i)\right)\big)$$

Ilustración 3: Función logística



Fuente: Elaboración propia a partir de datos propios

Función sigmoidea de entropía cruzada: nos permite hacer clasificación multiclase, teniendo en cuenta que las clases son mutuamente excluyentes.

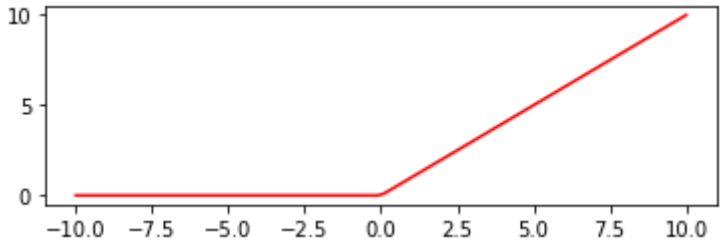
$$P(y_i = j|x_i) = -\log\left(\frac{e^{f_j(x_i)}}{\sum_k e^{f_k(x_i)}}\right)$$

Rectificador de unidad lineal (ReLu)

$$f(x) = \max(0, x)$$

Donde X puede ser cualquier función, por ejemplo, una sigmoidea $log\big(1 + \exp\left(-y_i f(x_i)\right)\big)$

Ilustración 4:Rectificador de unidad lineal (ReLu)



Fuente: Elaboración propia a partir de datos propios

b) Algoritmos de optimización

Para poder minimizar nuestra función de pérdida es necesario utilizar algún método de optimización. Los métodos más utilizados en redes neuronales y clasificadores son de la familia de gradiente descendente. Este tipo de algoritmos se basan principalmente en moverse a lo largo de una función por medio del gradiente, es decir, la dirección en cada punto en donde la función decrece más rápido, ya que, como se mencionó anteriormente, lo que se quiere es minimizar una función de pérdida. El principio básico de estos algoritmos de gradiente descendente tiene que ver con que en cada punto en donde

estemos en la función calculamos las derivadas parciales de cada una de las componentes, y a partir de ahí nos vamos al siguiente punto. Es importante anotar que no hay ningún método de optimización que siempre converja a un óptimo global; sin embargo, con varios trucos numéricos que se han desarrollado a lo largo de los años se ha disminuido la propensión de estos algoritmos de caer en óptimos locales. Estos trucos numéricos los discutiremos un poco más adelante. Actualmente se utilizan cinco métodos de gradiente descendente que dependen mucho de la cantidad de datos de entrenamiento que se tengan.

Principales métodos y submétodos

- Gradiente descendente
- *Batch gradiente descend*
- *Mini batch gradient descent*
- L-BFGS *Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm*
- *Stochastic gradient descent* (SGD)

Gradiente descendente

Nociones básicas

Supongamos una función

$$f(w): R - R$$

Donde el objetivo es hallar los pesos w que minimizan dicha función $-f(w)$

En primer lugar, hay que calcular la derivada con respecto al parámetro, para saber el cambio de la función a partir de un cambio en la variable de esta manera:

Definición clásica de derivada

$$f'(w) = \lim_{h-0} \frac{f(h+h) - f(w)}{h}$$

Actualización de coordenada: esta expresión arroja la nueva ordenada en función de la rata de cambio de la función en ese punto, multiplicada por una escalar h partiendo del punto inicial $f(w)$:

$$f(w+h) \approx f(w) + hf'(w)$$

Luego:

$$w^{i+1} = w^i - hf^i(w^i)$$

En la ecuación anterior podemos ver cómo se actualiza la nueva coordenada w en función de las derivadas.

Nota: este método resulta muy costoso computacionalmente, además de que la escogencia de la escalar h para moverse a lo largo de la función es crítica para poder alcanzar el óptimo de la función, ya que es esta la que define el paso en función de la derivada en cada punto, pero resulta que, si en ese lugar en particular la derivada es muy negativa y está multiplicada por una h relativamente grande, el movimiento puede ser tan abrupto que puedo terminar pasando el mínimo de la función.

La determinación de h o *alpha* ha sido ampliamente discutida en la literatura y no se tiene un consenso general sobre esta, ya que si ese parámetro es muy pequeño la velocidad de convergencia puede ser muy lenta y tomar mucho tiempo, y si h si es muy grande, como se mencionó en el párrafo anterior, podemos no encontrar el óptimo. Si bien es cierto que hay criterios y algoritmos para determinar una *alpha* que cambia a lo largo del tiempo, lo que se hace usualmente en la práctica es observar el comportamiento de la función de pérdida a medida que pasan las épocas para saber si es un buen o mal parámetro.

Gradiente

Consideremos una función $f(x)$ de la cual queremos hallar el mínimo

$$f(x) = \sum_i x_i^2$$

Donde el gradiente de $f(x)$ es un vector de i componentes de las derivadas parciales de $f(x)$ y, por lo tanto, el gradiente es la dirección en donde más rápido crece la función. $f(x)$

$$\nabla f = \left(\frac{\delta f}{\delta x_1}, \dots, \frac{\delta f}{\delta x_n}\right)$$

Algoritmia

Se cuenta con una serie de datos (x, y) con una cantidad de n datos y representan una situación en particular:

- se quiere buscar un modelo que aproxime el comportamiento de los datos (x, y)
- para aproximar el comportamiento de los datos se genera una función arbitraria, puede ser cualquiera de las mencionados en el apartado 1 o cualquier otra$f_{(x,y)}(w)$ que nos permita de alguna manera minimizar el error de nuestro modelo predictivo con el real.
- $f_{(x,y)}(w)$ es la función que queremos minimizar con los pesos w que minimizan el error
- Z va a ser el modelo de representación del modelo
- $\nabla f_{(x,y)}$ es el gradiente de la función
    1. Inicializar el modelo con pesos aleatorio W
    2. Calcular la función de pérdida $f_{(x,y)}(w)$
    3. Calcular el gradiente de la función $\nabla f_{(x,y)}$

4. Repetir pasos 2 y 3 hasta llegar a un error aceptable

c) Backpropagation algorithm

Como ya se ha ilustrado en los apartados anteriores de una de una función $f_{(x,y)}(w)$, se buscan los pesos óptimos que minimicen la función de pérdida, pero esto tiene un costo, ya que debemos calcular el gradiente en cada punto y por cada neurona donde hagamos el cálculo, por lo tanto, si tenemos 1.000.000 de parámetros, deberemos calcular el mismo número de derivadas más 1, lo cual, en modelos difíciles de entrenar con grandes cantidades de capas ocultas y gran cantidad de neuronas, se convierte en una tarea muy pesada computacionalmente.

De manera muy simplista y como resume Roger Romero (Morral, 2020), el funcionamiento de este algoritmo es básicamente calcular el gradiente de la función de pérdida con respecto a cada peso utilizando la regla de la cadena, calculando el gradiente una capa a la vez. Este proceso se repite iterativamente hacia atrás desde la última capa, para evitar cálculos redundantes y cálculos intermedios en la regla de la cadena.

2. METODOLOGÍA

2.1.Serie de datos

Para este trabajo de final de máster, se ha decidido utilizar OANDA como bróker para la obtención de datos. En particular, OANDA es una plataforma de *trading* para *traders* de *retail* (*traders* no institucionales). Se ha escogido esta plataforma por tres razones fundamentales:

En primer lugar, es una plataforma que tiene una gran cantidad de activos.Tiene 150 diferentes tipos de activos de los que podemos obtener información, de allí podremos adquirir sus series históricas con diferentes intervalos de tiempo, es decir, *tickers* de un segundo, un minuto, una hora, etcétera. Esto es muy importante, pues, dentro de la estrategia de compra y venta, el *timing* de la operación también es muy importante, ya que, como explicaremos más adelante, por los costos de transacción es muy difícil alcanzar rendimientos positivos en el retorno total en una estrategia real, porque los costos de transacción pueden agotar una estrategia de *trading* rápidamente.

En segundo lugar, se escogió esta plataforma porque la profundidad del mercado también es importante a la hora de evaluar el comportamiento de un activo, por lo cual es importante tener una plataforma que tenga grandes volúmenes de transacciones y que de alguna manera el precio de los activos refleje los volúmenes de negociación.

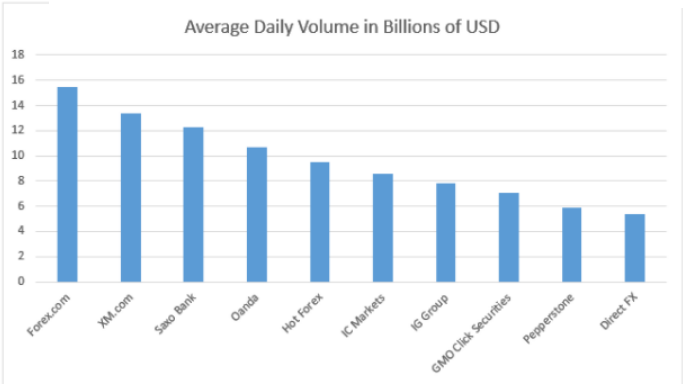
Ilustración 5: Biggest brokers by capitalization

	Broker	Country	Volume	1 Year Change
1	Forex.com	United States	15.5	+41%
2	XM.com	Cyprus	13.4	+151%
3	Saxo Bank	Denmark	12.3	-4%
4	Oanda	United States	10.7	+16%
5	Hot Forex	St Vincent	9.5	+91%
6	IC Markets	Australia	8.6	+118%
7	IG Group	United Kingdom	7.8	-3%
8	GMO Click Securities	Hong Kong	7.1	-17%
9	Pepperstone	Australia	5.9	+9%
10	Direct FX	Australia	5.4	+34%

Fuente: tomado de https://fairreporters.net/economy/largest-forex-brokers-by-volume-in-2015

En tercer lugar, he escogido esta plataforma por las facilidades de brinda para la extracción de datos y las API que ofrece para su extracción en diferentes en lenguajes. Toda la estrategia estará programada en *Python* y las peticiones al servidor de la aplicación se realizarán por medio de un *script* también programado en este lenguaje

Ilustración 6: Daily volume



Fuente: tomado de https://fairreporters.net/economy/largest-forex-brokers-by-volume-in-2015

.

De acuerdo con nuestro segundo criterio para escoger un bróker, es posible ver que OANDA no tiene el promedio diario de negociación más alto de todos. De hecho, está por debajo de plataformas como Forex y *Saxo Bank*, a pesar de esto, OANDA provee una API para *Python* mucho más robusta y con más funcionalidades, por lo cual es la plataforma escogida.

2.2.Medidas de desempeño

Redes neuronales

Las medidas de desempeño son de fundamental importancia para monitorear la calidad del modelo, por tal motivo se debe observar el comportamiento de este tanto en los datos de entrenamiento como en los datos de prueba y, finalmente, en los datos de validación.

- **Datos de entrenamiento:** estas series de datos para las redes neuronalesson escogidos inicialmente para fijar los parámetros o pesos, con los cuales se va a predecir la salida deseada, en este caso en particular son los datos de la serie financiera en el periodo t+1.

- **Datos de prueba:** esta serie de datos busca observar qué tan efectivo es el modelo en datos que la red nunca ha visto; sin embargo, hay que aclarar que la selección del modelo se hace en base en estos, ya que la idea es buscar los parámetros que mejor ajustan los datos de entrenamiento, pero que no tienen un sobre ajuste (*overfitting)* sobre los datos de prueba, pues, cuando esto sucede, el modelo no se comporta de una manera adecuada a parámetros nuevos, ya que memorizó los datos de entrada y no le es posible predecir nuevos datos.
- **Datos de validación:** son aquellos que la red nunca ha visto y es cuando se prueba el modelo en condiciones casi reales, pues son datos con los cuales no se escogió ni el modelo ni se entrenaron los parámetros, por lo tanto, son la mejor estimación de cómo se comportará el modelo en condiciones normales.

Nota: En la práctica, lo que se suele hacer después de pasar toda la fase de entrenamiento y validación es entrenar nuevamente el modelo con todos los datos disponibles y estos parámetros de modelo son los que se pasarán a producción en un ámbito real.

2.2.1. Serie de tiempo

El total de la serie de tiempo para este este trabajo final de máster son *tickers* de precio cada 5 minutos sobre la serie de EUR/USD en un periodo desde enero del 2007 hasta mayo del 2021. En total la serie de datos cuenta con las siguientes características:

- Serie: EUR/USD
- Granularidad: 5 minutos
- Total, datos = 1’056.000 *ticks*
- Datos de validación: 60 % = 633.000 *ticks*
- Datos de prueba: 20 % = 211.200
- Datos de validación: 10 % = 105.600

2.2.2. Métricas para evaluar el entrenamiento de los modelos neuronales

Para este trabajo de grado, se decidió que los parámetros por monitorizar dentro de los diferentes modelos utilizados van a ser comunes para todos, es decir, las funciones van a ser las mismas independiente del modelo utilizado.

Función de pérdida

Esta función, como se mencionó en el apartado 1 de teoría, es con la cual se quiere minimizar el error de generalización, es decir, el error entre la función con la que aproximamos el valor deseado y el valor real.

Modelo de regresión

Huber loss

$$L_\delta\big(y, f(x)\big) = \begin{cases} \frac{1}{2}\big(y - f(x)\big)^2 & \forall |y - f(x)| \leq \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2 & en\ cualquier\ otro\ caso \end{cases}$$

Esta función en particular resulta muy útil en datos de regresión, pues es menos sensible para casos atípicos, como sí lo son los mínimos cuadrados medios, pues estos penalizan mucho más los errores muy lejanos, mientras que esta es más permisiva y ayuda a un mejor ajuste del modelo.

Función de error

En general, estas funciones se utilizan para saber si el modelo está sobre ajustado o no, pues, aunque la función de pérdida llegue un valor muy pequeño, esto no implica que el modelo se va a comportar de una manera correcta en datos nuevos, dado que una función de pérdida muy pequeña puede decir que estamos sobreajustando los datos y, por lo tanto, el modelo no tiene poder predictivo, sino, por el contrario, aprendió los datos de entrada, por lo cual se monitorizan las funciones de error en el conjunto de entrenamiento y de prueba.

Error absoluto medio (MAE)

Esta función es la suma de todas las diferencias absolutas entre el valor deseado y el predicho, con los cual evitamos el efecto de los valores fuera de muestra perjudiciales para el ajuste.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - y_i^p|}{n}$$

$$y_i : valor\ real$$

$$y_i^p : valor\ predicho$$

2.2.3. Nociones financieras de rendimiento

Para determinar si una estrategia de inversión es buena o mala o si una estrategia es mejor que otra debe haber una medida general que mida el desempeño de esta. Por tal motivo, se introduce una noción básica para medir el rendimiento de una cartera. Para evaluar el comportamiento de una cartera o un activo en particular, podemos partir de los diferenciales de precios de dicho activo en el momento t y componerlos en un determinado periodo y determinar cuál fue la ganancia total en dicho periodo.

En general, el rendimiento de un activo puede ser definido como su cambio de un periodo a otro:

$$\boldsymbol{r_t = \frac{p_i - p_{t-1}}{p_{i-1}} = \frac{p_i}{p_{i-1}} - 1}$$

Para las series financieras, se utilizan los retornos logarítmicos, dado que se asume que los retornos de los activos son normales, con una expresión de la siguiente forma:

$$r_t = log\left(\frac{p_t}{p_{t-1}}\right)$$

Esta expresión es útil, ya que los retornos logarítmicos son escindibles, por lo cual nos permiten hacer la composición de los retornos como la sumatoria de retornos intermedios o como un salto entre precios de periodos más largos como en el ejemplo siguiente:

Ejemplo

$$r_t = log\left(\frac{p_5}{p_{t-5}}\right)$$

O como

$$\sum_{i=1}^{5} log\left(\frac{p_{i+1}}{p_i}\right) \quad \forall i \leq 5$$

De lo anterior podemos computar el retorno total para cualquier activo de la siguiente manera:

Retorno total

$$TR_i = \begin{cases} \sum_{i=1}^{t} c_{i-1} + c_{i-1}(1 + r_i) * k & k = 1 \; si \; tiene \; posicion \; en \; el \; activo \\ \sum_{i=1}^{t} c_{i-1} & k = 0 \; si \; no \; tiene \; posicion \; en \; el \; activo \end{cases}$$

Donde:

$$TR_i: Retorno \; total \; en \; el \; momento \; t$$

$$c_{i-1} = valor \; de \; la \; cartera \; en \; el \; momento \; t - 1$$

$$r_i: retorno \; entre \; el \; momento \; t \; y \; t - 1$$

$$k: define \; si \; tenemos \; el \; activo \; en \; el \; cartera \; en \; el \; monento \; t$$

2.3. Medidas de riesgo de cartera

El análisis de la volatilidad de las series financieras es de suma importancia, pues estimar de alguna manera las pérdidas posibles sobre alguna inversión en particular provee una buena intuición del riesgo que se está corriendo. Así mismo, como una definición más formal en el ámbito financiero, como menciona Gracia (2020), podemos decir que una medida de riesgos asociada a una posición de cartera tiene una función L con un número real de observaciones que describen el riesgo de dicha cartera.

Así pues, con esta definición podemos determinar varias medidas de riesgo que nos servirán para aproximar una pérdida esperada de nuestra cartera.

2.3.1. Value at risk (VaR)

El VaR es una medida de riesgo que nos dice cuál es el nivel de pérdida que podemos llegar a tener en un determinado nivel de confianza o en un determinado cuantil. Así pues, podemos definir el VaR según Gracia (2020) para un portafolio con una función de pérdida L a un nivel de confianza α, como el menor número l cuya probabilidad que la pérdida L exceda l y que no es más grande que 1- α

formalmente:

$$VaR_{\propto} = VaR_{\propto}(L) = \inf\{l \in \mathbb{R}: P(L > l)1 - \alpha\} = \inf\{l \in \mathbb{R}: F_L(l) \geq \alpha\}$$

De acuerdo con esta definición debemos definir también la función de los cuantiles o la inversa generalizada.

Inversa generalizada: dada una función creciente $T: \mathbb{R} \to \mathbb{R}$ la generalizada inversa de T es definida por $T \leftarrow (y) \coloneqq \inf\{x \in \mathbb{R}: T(x) \geq y\}$ (Gracia, 2020).

Función de cuantía: dada una función de pérdida df F, la inversa generalizada df F^{-}es llamda la función de cuantía de F para todo $\alpha \in (0,1)$ y el cuantil $\alpha\ de\ F$ esta dado por:

$$q_{\alpha}(F) \coloneqq F^{\leftarrow}(\alpha) = \inf\{x \in \mathbb{R}: F(x) \geq \alpha\}$$

De manera más general, como lo expresa Gracia (2020) en el *quantitative finance* si F es continua y estrictamente creciente podemos simplemente definir $q_{\alpha}(F) = F^{-1}(\alpha)$ donde F^{-1} es la inversa ordinaria de F.

Para que la definición de VaR se cumpla, se tienen que cumplir dos condiciones:

Un punto $x_0 \in \mathbb{R}$ es el $\alpha\ cuantil$ de alguna función F si y solamente si las siguientes dos condiciones se cumplen: $F(x_0) \geq \alpha: y\ F(x) < \alpha\ \forall\ x < x_0$

VaR distribución de pérdida para una normal

Supongamos que F_L es normal con una media μ con una varianza σ^2 y un $\alpha \in (0,1)$

$$VaR_{\alpha} = \mu + \sigma\Phi^{-1}(\alpha)$$

Donde Φ es la normal estándar y $\Phi^{-1}(\alpha)$ es la función inversa de Φ

2.3.2. Expected shortfall

La pérdida esperada por su nombre en español es una medida de riesgo que busca ir un poco más allá de una medida fija de VaR a un determinado nivel de confianza y explorar en las colas de las distribuciones. Así, una definición intuitiva de esta medida de riesgo es calcular el promedio de todos los VaR por encima del VaR un determinado α.

Más formalmente y como se define en el *quantitativa finance* (Gracia, 2020) para una función de pérdida $L\ con\ E(|L|) < \infty\ y\ una\ funcionde\ perdida\ F_L$ el ES se relaciona con el VaR de la siguiente manera:

$$ES_{\alpha} = \frac{1}{1-\alpha}\int_{\alpha}^{\infty} VaR_u(L)\ du$$

Donde en lugar de un nivel particular de α, se calcula el promedio de los VaR por encima de $u \geq \alpha$ y por lo tanto $ES_\alpha \geq VaR_\alpha$

En particular para una distribución normal la función cerrada para el ES:

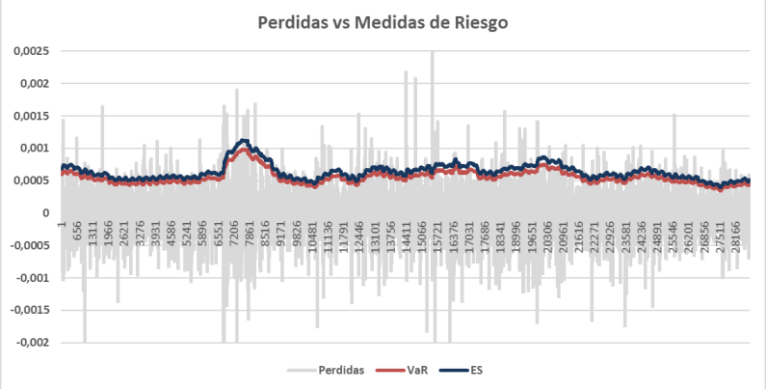
$$ES_\alpha = \mu + \sigma \frac{\phi\left(\Phi^{-1}(\alpha)\right)}{1-\alpha}$$

ϕ denota la función de densidad en el punto de la inversa de la distribución normal en el punto $\Phi^{-1}(\alpha)$

2.3.3. Medida de riesgo financiero para los riesgos escogidos

Para los activos escogidos lo que se pretende es calcular el ES como medida de riesgo en cada momento del tiempo, es decir, cada nuevo precio en este caso cada 5 minutos.

Ilustración 7: Medidas de riesgo VaR y ES



Fuente: Elaboración propia a partir de datos propios

Tabla 1: Eur/Usd VaR y ES

Medida	Alfa	Perdidas > L	TOTAL	% Error
VaR (L)	99%	281	28785	0,98%
ES (L)	99%	443	28785	1,54%

Como medida de riesgo, el ES (*expected shrotfall)* resulta una medida más conveniente para la medición de riesgo de esta cartera, pues en general tiende a subestimar menos la pérdida esperada y se ajusta mejor a la variable de pérdida. Esto lo podemos corroborar en la tabla anterior, en donde se muestra el porcentaje de las veces que la pérdida ha superado la medida de riesgo, y teniendo en cuenta que estamos calculando las medidas de riesgos a un 99 % de confianza, implica que como máximo deberíamos equivocarnos en la medición el 1 % de las veces, lo cual no ocurre en el caso del VaR, pues es sobrepasado 1,54 % mientras que el ES un 0,98% por lo cual será nuestra medida de riesgo escogida.

2.4. Estrategia de inversión

Dentro del mundo de las inversiones existen miles de tipos de estrategias para determinar si se invierte o no en un activo determinado. Entre las técnicas más conocidas podemos resaltar el análisis técnico, el análisis fundamental, el arbitraje estadístico, técnicas econométricas, entre otras Para este trabajo de final de máster, la idea es combinar varias de estas técnicas dentro de un solo algoritmo que tome todos estos datos de entrada y con base en estas determinar de una forma mucho más acertada si debemos comprar o vender un activo en un determinado momento t.

Componentes

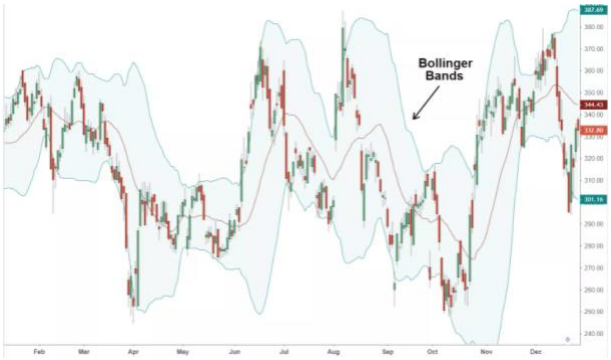
Análisis técnico

El análisis técnico es el estudio de la realización de los mercados financieros. Como menciona Suisse (2019), el analista técnico observa los cambios de los precios en un intervalo de tiempo determinado, sea día a día, mes a mes o cualquier intervalo de tiempo en forma de tablas o gráficos, para, a partir de estas, poder hacer inferencias. Estas inferencias se hacen no son solo a partir de las tablas y las gráficas, sino también con diferentes indicadores técnicos como veremos en el siguiente párrafo.

Bollinger bands

Las bandas de Bollinger son un indicador técnico que tiene una serie de líneas de tendencia que se basa en una línea central de media móvil, usualmente de 22 periodos hacia a atrás y otras dos líneas más de tendencia igual a la central, pero con una distancia de dos desviaciones estándar, tanto hacia arriba como hacia abajo. Este indicador técnico lo que intenta predecir son las condiciones de mercado en donde un activo está sobrecomprado o esta sobrevendido, así pues, si el precio real del activo toca la línea superior quiere decir que está llegando a una condición de sobrecompra y por lo tanto empezaría a bajar y por lo tanto se debería vender. Por su parte cuando el precio real del activo alcana la línea inferior quiere decir que está en una condición de sobre venta, es decir, que debería empezar a subir (Investopedia, 2021).

Ilustración 8: Bollinger bands



Fuente: tomado de https://www.investopedia.com/terms/b/bollingerbands.asp

Relative strength index (RSI)

El indicador de fuerza relativa mide la fuerza o el impulso en los cambios de los precios de un activo. En el manual de *trading* de (Avatrade, 2020) se explica que este indicador está categorizado dentro de los osciladores, lo cual significa que su valor se sitúa entre 0 y 100. De igual manera al indicador anterior, este indicador tiene como propósito medir qué tan sobre comprado o sobre vendido pueden estar un activo, en particular se dice que en promedio los activos llegan a una condición de sobrecompra cuando llegan a un valor de 70 y un valor de sobreventa en un valor de 30, ahora bien, esto es en general pero estos niveles pueden varíar de activo en activo.

Ilustración 9: Relative Strenght Index

Fuente: tomado de https://www.avatrade.es/educacion/professional-trading-strategies/rsi-trading-strategies

Exponential moving average

Este indicador en particular es una media móvil exponencial, que da menos peso a las entradas más antiguas y más peso a las más recientes. Este indicador busca corregir el hecho de que la información más antigua es menos relevante en una serie de tiempo y la reciente tiene más poder en los precios siguientes (Capital, 2019).

Ilustración 10:Exponential moving average

Fuente: tomado de https://www.investopedia.com/ask/answers/122314/what-exponential-moving-average-ema

2.4.1. Características y datos de entrada

Inicialmente, la base de datos original descargada del servidor de OANDA contenía 1’056.000 *ticks* de precio con una granularidad de 5 minutos entre ellas, desde febrero de 2007 hasta abril de 2021, sin embargo, a lo largo de este trabajo de final de master, se han presentado diferentes inconvenientes en cuanto al poder de computación y el tiempo de cálculo, ya que para entrenar redes neuronales relativamente grandes se necesita gran poder computacional y arquitecturas que son muy costosas.

De acuerdo con lo anterior, se ha decidido tomar solo una décima parte del data set, pues en intentos previos de entrenar un solo algoritmo para una sola serie tomaba alrededor 12 horas con una infraestructura en la nube ejecutándose sobre un GPU NIVIDIA P100, que en general es bastante poderosa, pero para este caso no era suficiente y aumentar el poder de computación resultaría muy costoso para entrenar todas las redes, por este motivo se acorta la serie y solo se tomará en cuenta el periodo desde 23 de mayo del 2019.

Características principales

- EUR/USD: Cantidad de Euros por dólar americano
- EUR/SGD: Cantidad de Euros por dólar de Singapur
- EUR/GBP: Cantidad de Euros por Libras Esterlinas
- EUR/CAD: Cantidad de Euros por dólar canadiense
- EUR/HKD: Cantidad de Euros por dólar Hongkonés

Sub características: Estas Características son calculadas para cada uno de los 5 activos anteriores

Además de las series de tiempo iniciales se extraen características adicionales relacionadas con cada una de las series para darle más fuerza al poder predictivo de los modelos

- **V: Volumen de operaciones:** el monto total operado entre el momento $t \ y \ t+1$, es
- **Media móvil 10 periodos:** promedio simple de los precios de los 10 periodos anteriores,

$$M_{i,10} = \frac{1}{10} \sum_{i-1-10}^{t-1} p_i$$

- **Media móvil 20 periodos:** promedio simple de los precios de los 20 periodos anteriores,

$$M_{i,20} = \frac{1}{20} \sum_{i-1-20}^{t-1} p_i$$

- **Media móvil 30 periodos:** promedio simple de los precios de los 30 periodos anteriores,

$$M_{i,30} = \frac{1}{30} \sum_{i-1-30}^{t-1} p_i$$

Siendo p_i los precios del activo.

- RSI: *Relative Strenght Index* 14 días
- BBM: Banda de Bollinger media
- BBI: Banda de Bollinger inferior
- BBS: Banda de Bollinger superior
- EMA: Media móvil exponencial 20 días

Características autorregresivas

Para estos modelos en particular también se utiliza una ventana temporal de toda la serie de características, es decir, se toma la característica inmediatamente anterior más una venta definida hacia atrás en el tiempo. En particular utilizamos una ventana de 15 periodos hacia atrás, es decir, que por cada una de las características anunciadas anteriormente tenemos 15 más de ellas hacia atrás en el tiempo

$$\Omega: [\mathrm{EUR/USD}_t, \mathrm{EUR/SGD}_t, \mathrm{EUR/SGD}_t, \mathrm{EUR/CAD}_t, \mathrm{EUR} \\ /\mathrm{HKD}_t, V_t, M_{i,10,t}, M_{i,20,t}, M_{i,30,t}, RSI_t, BBM_t, BBI_t, BBS_t, EMA_t]$$

$$\Omega: Espacio\ de\ caracteristicas$$

2.5. Estrategia de *trading*

En este apartado se mostrará la estrategia que se ha escogido con base en las características escogidas, modelos predictivos y algoritmia basada en los pronósticos para la compra y venta del activo.

En cada momento del tiempo, nuestra estrategia de *trading* tendrá una señal que puede tener 3 posibles valores, comprar el activo, mantener la posición o vender el activo

for t in tiempo_*t :*
Señal = Predicción()
Posición = get_position()
if (Señal == Comprar and posición == 0) :
comprar una unidad del activo al precio a(t)
Posición = 1
Elseif (Señal == Comprar and posición == 1):
Posición = Posición
No hacer nada mantener el activo
Elseif(Señal == Vender and posición == 1):
Vender una unidad del activo al precio a(t)
Elseif(Señal == Vender and posición == 0):
Posición= posición
No hacer nada, no tenemos activo
End if
End For

Conceptualización

- *Bid Price*: precio al que se compra el activo en el mercado
- *Ask Price*: precio al que se vende al activo en el mercado

Para esta estrategia de *trading* haremos varias conceptualizaciones acerca de los costos de transacción. En primer lugar, tomaremos la serie de tiempo del BID_PRICE EURO/USD que usualmente es el precio al que compramos el activo. En segundo lugar, para el precio de venta no utilizaremos el ASK_PRICE, pues implicaría también hacer un pronóstico de esta serie para cada uno de los modelos, por lo cual haremos una aproximación, en donde sabemos lo siguiente:

Promedio *Spread Bid-Ask histórico* = 0.00006

Lo anterior quiere decir que en el mismo instante t, el precio de compra *bid price* estará 0.6 puntos básicos por encima del precio de compra A*sk Price.* En el caso contrario cuando se vaya a vender tomaremos el bid Price y le restaremos el spread histórico para de alguna manera simular los costos de transacción de compra y venta

$$Precio\ de\ compra = Bid\ price$$

$$Precio\ de\ venta = Bid\ price - 0.000061$$

Señal:

Compra:

- Cuando retorno del precio pronosticado por el modelo de regresión sobre el precio actual conocido sea mayor a un determinado k se compra el activo.

$$Compra_i = \begin{cases} \text{compra} & \mathrm{r}_{i+t} = \ln\left(\frac{f_{i+1}(w)}{x_i}\right) > k \\ no\ hace\ nada\ \text{compra} & \mathrm{r}_{i+t} \quad \ln\left(\frac{f_{i+1}(w)}{x_i}\right) \leq k \end{cases} \qquad \forall\, x_i, \forall r_i > 0$$

Donde r_{i+t} es el retorno pronosticado en base a $f_{i+1}(w)$ que es el precio pronosticado por el modelo de regresión y x_i es el precio conocido en el momento t_i

- Cuando la clasificación del pronóstico del retorno por el modelo de clasificación caiga en el intervalo que sea mayor igual a k compra el activo

$$Compra_i = \begin{cases} compra & f_{i+1}(w)\ \epsilon\ kg_i > k \\ no\ hace\ nada & f_{i+1}(w)\ \epsilon\ kg_i < k \end{cases} \qquad \forall\, x_i$$

Donde kg_i el intervalo en el que se ha categorizado el retorno pronosticado

- Para hacer la compra del activo las dos condiciones tienen que cumplirse

Venta:

- Cuando retorno del precio pronosticado por el modelo de regresión sobre el precio actual conocido sea menor a un determinado k se vende el activo, pero si el retorno pronosticado no es muy negativo no hace nada.

$$vende_i = \begin{cases} \text{vende} & \mathrm{r}_i = \ln\left(\frac{f_{i+1}(w)}{x_i}\right) \leq ES \quad \forall\, x_i, \forall r_i < 0 \\ \text{vende} & \mathrm{r}_i = \ln\left(\frac{f_{i+1}(w)}{x_i}\right) \leq RT \quad x_i, \forall r_i < 0 \\ \text{vende} & \mathrm{r}_i = \ln\left(\frac{f_{i+1}(w)}{x_i}\right) < k \quad \forall\, x_i\,, \forall r_i < 0 \\ \textit{no hace nada} & \mathrm{r}_i\ \mathrm{k} < \ln\left(\frac{f_{i+1}(w)}{x_i}\right) \leq 0 \end{cases}$$

Donde r_{i+t}es el retorno pronosticado en base a $f_{i+1}(w)$ que es el precio pronosticado por el modelo de regresión y x_i es el precio conocido en el momento t_i, ES (spected Shortfall) y si es igual o mayor a este valor debe vender de inmediato. RT representa el retorno total y si el acumulado de las perdidas sobrepasa cierto valor de RT debe vender

- Cuando la clasificación del pronóstico del retorno por el modelo de clasificación caiga en el intervalo que sea mayor igual a k compra el activo

$$vende_i = \begin{cases} vende & f_{i+1}(w) \in kg_i < RT \quad \forall\, x_i\, \forall r_i < 0 \\ vende & f_{i+1}(w) \in kg_i < ES \quad \forall\, x_i\, \forall r_i < 0 \\ vende & f_{i+1}(w) \in kg_i < k \quad \forall\, x_i\, \forall r_i < 0 \\ no\ hace\ nada & k < f_{i+1}(w) \in kg_i < 0 \end{cases}$$

Donde kg_i el intervalo en el que se ha categorizado el retorno pronosticado

- Para hacer la compra del activo las dos condiciones tienen que cumplirse

3. RESULTADOS

En esta sección analizaremos tanto los datos como la arquitectura de cada uno de los modelos. Así mismo, se plantea hacer un análisis exploratorio de cada una de las series de datos utilizadas, en cuanto a estadística descriptiva y funciones de distribución, pues esta información es de suma importancia tanto para el correcto funcionamiento de los algoritmos, como para la estimación de los riesgos, ya que en el ámbito de gestión de riesgos hay varias presunciones fuertes acerca la distribución de los retornos para el cálculo de funciones de perdida. Adicionalmente, para la convergencia más rápida de los algoritmos de aprendizaje automático, es necesario que se cumplan ciertas condiciones.

Tratamiento de datos

En particular para el entrenamiento de las redes neuronales, es conveniente que todos los datos estén bajo una misma escala como menciona (Vitrià, 2020) en su investigación, esto dado que así los algoritmos aprenden mejor, así mismo es ideal que todas las series de datos y las caracterices sean de una escala pequeña, preferiblemente centradas el rededor del cero o entre cero y uno.

De acuerdo con lo anterior, se utilizó un método de estandarización de datos de mínimos y máximos, usualmente se utiliza la normalización de las variables restando la media y dividendo por la desviación estándar para quietarles la escala, sin embargo, en este caso en particular como estamos hablando de precios de activos si hacemos este procedimiento podríamos llegar a tener precios negativos, lo cual no es conveniente una vez se re escale el modelo a producción.

Estandarización MinMax

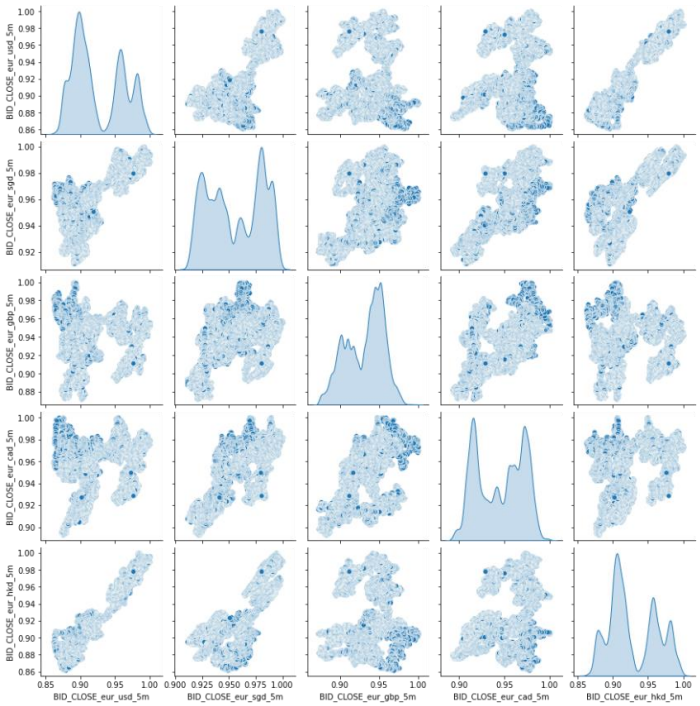
$$x_{rescalado} = \frac{(x_i - x_{\min})}{(x_{max} - x_{min})}$$

Con este sencillo algoritmo lo que logramos es que todos los datos estén entre cero y uno, si aplicamos este método a todas las variables a utilizar, tendremos todos los datos en la misma escala y será más fácil para el modelo entender la magnitud de las variables, así el modelo es más eficiente a la hora de prender

3.1. Estadística descriptiva

Series de precios

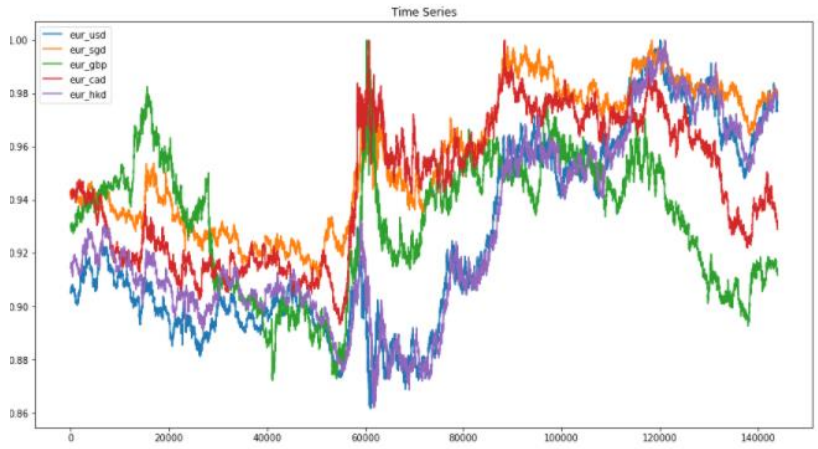
Ilustración 11: Distribución de Precios y Correlaciones



Fuente: Elaboración propia a partir de datos propios

En la imagen anterior, podemos ver tanto los gráficos de distribución de los precios de cada uno de los 5 activos, como también sus correlaciones con cada uno de los otros. Así cómo, en la diagonal se hace evidente la distribución de los precios de cada uno de los activos. Ahora bien, es notable que la mayoría de las series tienen un comportamiento bimodal y parcialmente difícil de ajustar a una distribución paramétrica, hay que tener en cuenta que estamos hablando sobre los precios y no sobre los retornos, por lo cual, los resultados anteriores son esperables. Así pues, podemos ver que las correlaciones en algunos casos son muy erráticas, pero en otros tiene un comportamiento predecible.

Ilustración 12: Series de precios Normalizadas



Fuente: Elaboración propia a partir de datos propios

De igual modo, en el grafico anterior podemos observar las 5 series financieras graficadas a lo largo de todo le horizonte de tiempo, que como ya se mencionó va desde 23 de mayo del 2019 a 5 de mayo de 2021. Igualmente hay que resaltar que todas las series tiene un dominio entre cero y 1, que es precisamente lo que estábamos buscando con la estandarización de los datos con el algoritmo MinMax.

Ilustración 13:Resumen Estadística Descriptiva

	count	mean	std	min	25%	50%	75%	max
BID_CLOSE_eur_usd_5m	144000.0	0.925605	0.036107	0.861562	0.896119	0.911442	0.959159	1.0
BID_CLOSE_eur_sgd_5m	144000.0	0.955972	0.025268	0.911297	0.932517	0.956334	0.980088	1.0
BID_CLOSE_eur_gbp_5m	144000.0	0.931720	0.024074	0.872102	0.910762	0.937716	0.950982	1.0
BID_CLOSE_eur_cad_5m	144000.0	0.944944	0.025670	0.893220	0.918467	0.948316	0.969388	1.0
BID_CLOSE_eur_hkd_5m	144000.0	0.928235	0.034094	0.861939	0.902814	0.916310	0.958724	1.0

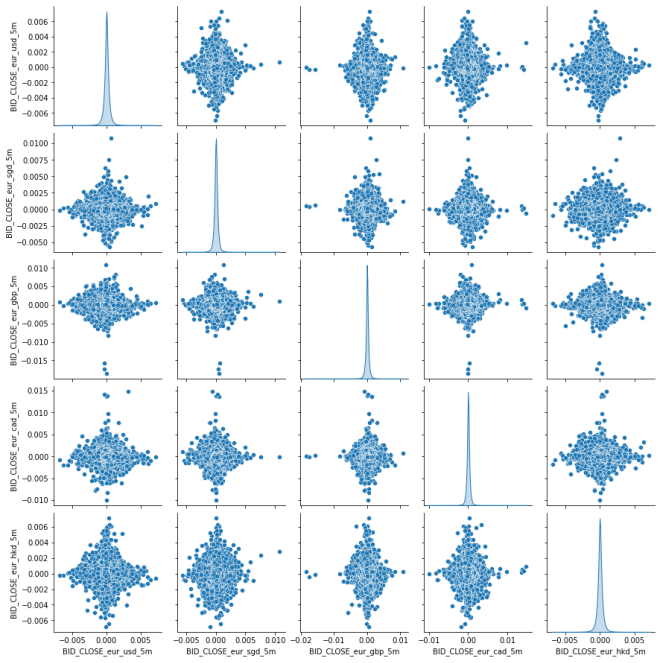
Fuente: Elaboración propia a partir de datos propios

Ahora bien, en cuanto a estadísticas generales podemos ver que la media para el dólar de singapur, la libra esterlina y el dólar canadiense con base a los euros es muy similar. Evidenciando que es fácilmente observable que él euros Euro/dólar y el Euro/hkd tienen una desviación típica muy similar. Si bien en la vida real sus cotizaciones tienen escalas diferentes, cuando revisamos el grafico de cotizaciones anterior podemos ver que su

comportamiento tiene una similitud muy particular lo cual en un futuro nos podría llevar a hacer más inferencias en la creación de otros algoritmos.

Retornos

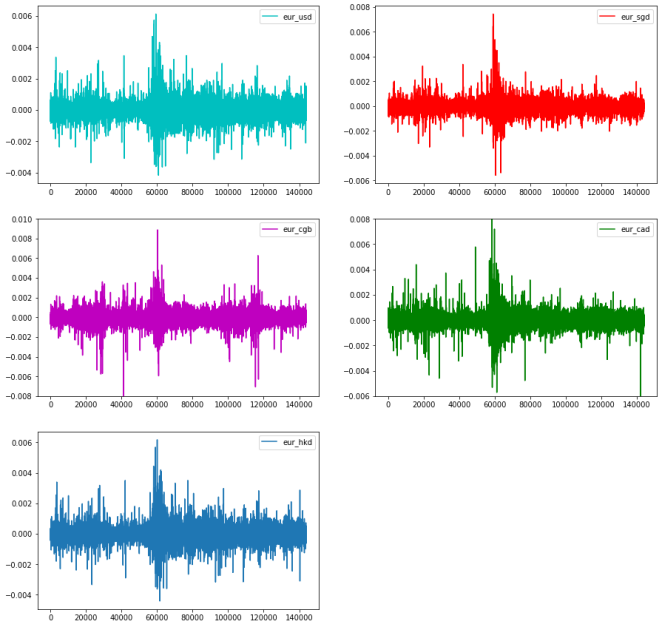
Ilustración 14:Distribucion retornos y Correlaciones



Fuente: Elaboración propia a partir de datos propios

Según la imagen previa, las distribuciones de los retornos de cada uno de los cinco activos se encuentran muy junto a la media y de igual forma la mayoría de la masa de probabilidad. Además, tienen colas muy pesadas y se hace evidente que todas las distribuciones tienen una correlación bastante similar entre unas y otras; con una forma de diamante, lo cual luego nos puede ser útil para la medición de los riesgos.

Ilustración 15: Retornos por Activo



Fuente: Elaboración propia a partir de datos propios

En cuanto al comportamiento de los retornos sobre la serie temporal, podemos ver que todas tienen un comportamiento similar. Incluso se hace notorio que en todas las series de retornos, antes de la mitad, tienen un pico de volatilidad generalizado. Todo lo anterior ocurre alrededor de finales del 2019, lo cual podría indicarnos un cambio abrupto del mercado en ese momento o incertidumbre generalizada en los mercados.

Ilustración 16:Estadsitica Descriptiva Retornos

	count	mean	std	min	25%	50%	75%	max
BID_CLOSE_eur_usd_5m	143999.0	5.254346e-07	0.000268	-0.004179	-0.000107	0.000000	0.000108	0.006133
BID_CLOSE_eur_sgd_5m	143999.0	2.774496e-07	0.000233	-0.005575	-0.000099	0.000000	0.000099	0.007435
BID_CLOSE_eur_gbp_5m	143999.0	-1.426614e-07	0.000327	-0.016586	-0.000125	0.000000	0.000127	0.008884
BID_CLOSE_eur_cad_5m	143999.0	-1.035405e-07	0.000291	-0.006540	-0.000116	0.000000	0.000117	0.014493
BID_CLOSE_eur_hkd_5m	143999.0	4.672426e-07	0.000269	-0.004410	-0.000108	0.000001	0.000110	0.006175

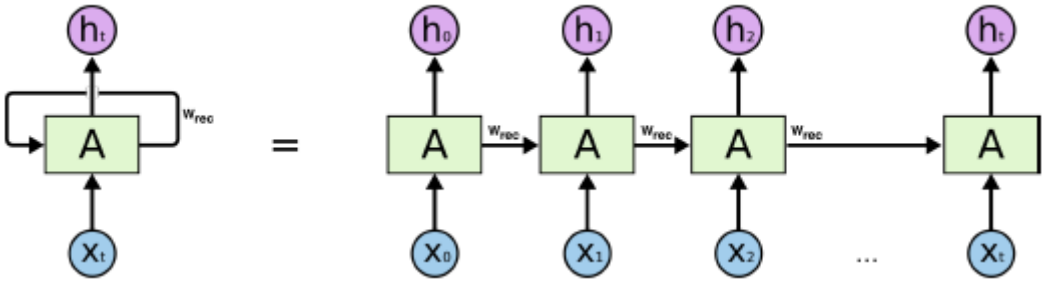
Fuente: Elaboración propia a partir de datos propios

Según lo anterior, se han calculado los retornos lo que nos permite observar que el euro dólar y el euro dólar hongkonés ya no son los activos más volátiles, si no que por el contrario lo son la libra esterlina y el par de dólares canadienses.

3.2. Redes Neuronales Recurrentes RNN y LSTM (Long short-term memory)

Redes Neuronales Recurrentes RNN(General):

Ilustración 17:Red Neuronal Recurrente



Fuente: (Morral, 2020)

Las redes neuronales recurrentes en general, son una clase de redes neuronales que mantienen el comportamiento temporal de una serie, por las conexiones directas entre las neuronas de una misma capa. Como explica (FAZLE KARIM1, 2017) una red neuronal mantiene un vector oculto h, que se actualiza en el tiempo t de la siguiente manera:

$$h_t = \tanh\left(Wh_{t-1} + Ix_t\right)$$

Donde: tanh es la función de la tangente hiperbólica, W son los pesos actuales de la matriz de recurrencia y I es la matriz de proyección. El estado h es usado para hacer predicciones.

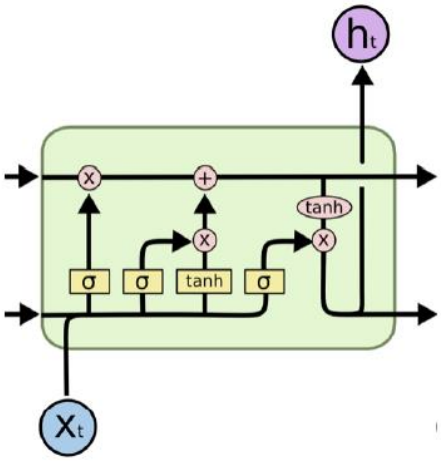
$$y_t = softmax(Wh_{t-1})$$

La función softmax es muy útil, ya que provee una función de distribución normalizada para cada una de las clases, σ representa en este caso la función logística y W es una matriz de pesos. Así pues, con el uso de h como parámetro de entrada para otra capa recurrente, podemos agregar más capas, para poder crear arquitecturas más complejas.

$$h_t^l = \sigma\left(Wh_{t-1}^l + Ih_t^{l-1}\right)$$

Redes Neuronales Recurrentes LSTM (Long short-term memory)

Ilustración 18:LSTM Neurona



Fuente: (Morral, 2020)

Es por esto por lo que, este tipo de redes neuronales recurrentes son una mejora notable a las redes recurrentes usuales, pues estas solucionan de alguna forma el problema de la desaparición del gradiente. Ahora bien, este problema se soluciona, de acuerdo con (S. Hochreiter and J. Schmidhuber, 1997), con la incorporación de funciones puente entre los estados dinámicos. Por lo que, en cada uno de los pasos de estas redes se mantiene un vector oculto h y un vector de memoria m responsable de controlar los cambios de estado de las salidas. Siendo así qué, en específico (FAZLE KARIM1, 2017) explica que el cálculo en tiempo t, se da de la siguiente manera:

$$g^u = \sigma(\ W^u\ h_{t-1} + I^u x_t)$$

$$g^f = \sigma(\ W^u\ h_{t-1} + I^f x_t)$$

$$g^o = \sigma(\ W^o\ h_{t-1} + I^o x_t)$$

$$g^c = \sigma(\ W^c\ h_{t-1} + I^c x_t)$$

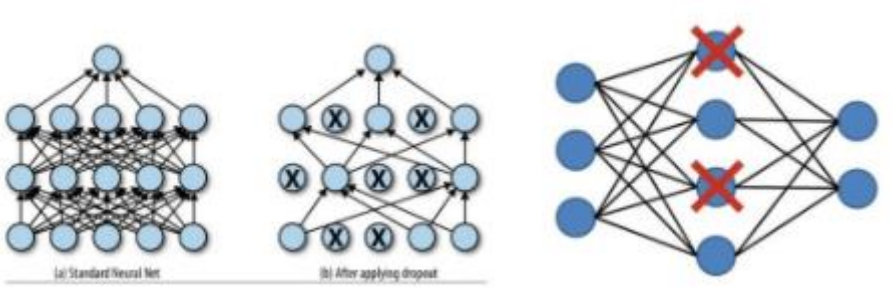
$$m_t = g^f \emptyset_t\ m_{t-1} + g^u \emptyset_t g^c$$

$$h_t = \tanh\ (g^0 \emptyset_t m_t)$$

Donde: σ representa en este caso la función logística, $\emptyset_t$ es la multiplicación Hadamard (element-wise), es decir, elemento a elemento, W^u, W^f, W^o, W^cson matrices de pesos recurrentes y finalmente I^u, I^f, I^o, I^c

3.3. Capas de abandono (Drop Out)

Ilustración 19:Drop Out Layer



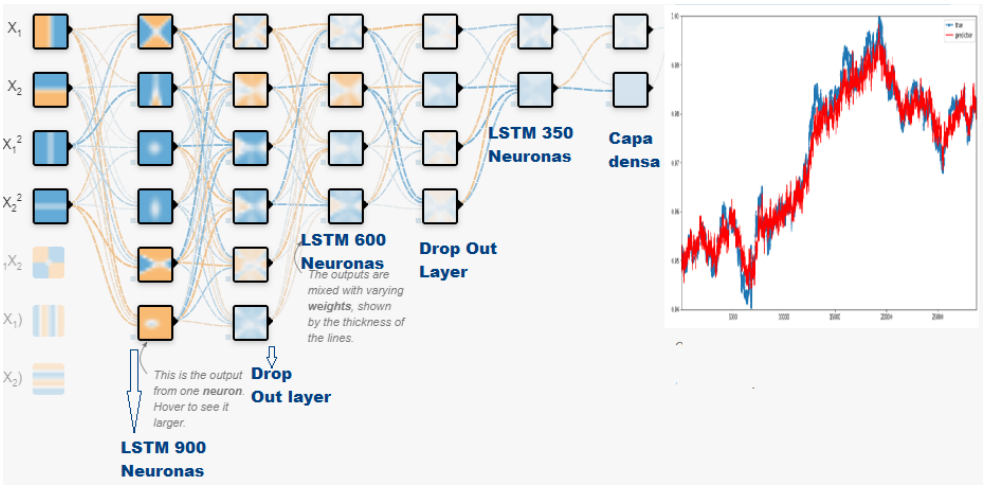
Fuente: tomado den https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning

Las capas de abandono (Drop Out), según explica (Qun Zhuge, 2017), son unas capas que anulan la contribución de algunas neuronas hacia las siguientes capas y deja sin modificar todas las demás. Por lo anterior, podemos aplicar una capa Dropout al vector de entrada en cuyo caso, anula algunas de sus características. Pero, a su vez también podemos aplicarlo a una capa oculta y lo que ocurriría es que este anularía algunas neuronas ocultas.

Es así como, es importante aclarar el valor que las capas de abandono tienen en el entrenamiento de CNN; ya que, son estas las que evitan el sobreajuste de los datos de entrenamiento. De lo contrario, al no estar presentes se evidencia que, el primer lote de muestras de entrenamiento será el responsable de incluir en el aprendizaje de una manera desproporcionadamente alta. Lo que al final evitaría el aprendizaje de características que aparecen solo en muestras o lotes posteriores:

Arquitectura

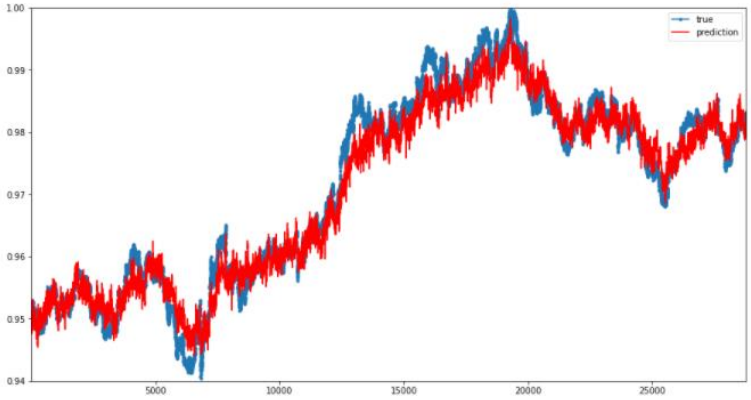
Ilustración 20:configuración LSTM



Fuente: Elaboración propia a partir de datos propios

Datos de prueba

Ilustración 21:Pronostico LSTM



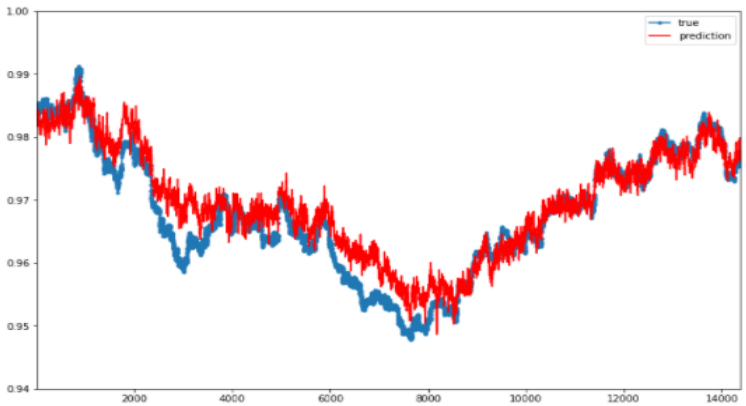
Fuente: Elaboración propia a partir de datos propios

Según la gráfica anterior, siendo la línea azul representativa de la serie de datos real y la línea roja la serie de datos pronosticados para el periodo del 10 de Octubre de 2020 – 22 de Febrero de 2020 en la serie de test, la serie con la que validamos el modelo y seleccionamos los parámetros de entrenamiento. Si bien es cierto que estos datos nunca los ha visto el modelo, es decir, no fueron utilizados para ajustar los parámetros. Si es la serie para escoger la mejor infraestructura en cuanto a diseño de la red y sus parámetros de optimización, como el ratio de aprendizaje, la cantidad de neuronas, y el número de capas entre otras.

Por lo anterior, se puede notar que el comportamiento de la serie pronosticada es bastante similar al de la de la serie real y es evidente que la serie de datos pronosticados mantiene características de la serie original como; la tendencia, media móvil y la amplitud. Lo que finalmente, hace positivo a este como buen modelo de predicción, teniendo en cuenta que son 28.400 datos que pronostica y que nunca había visto.

Datos de validación

Ilustración 22:Pronostico LSTM validación

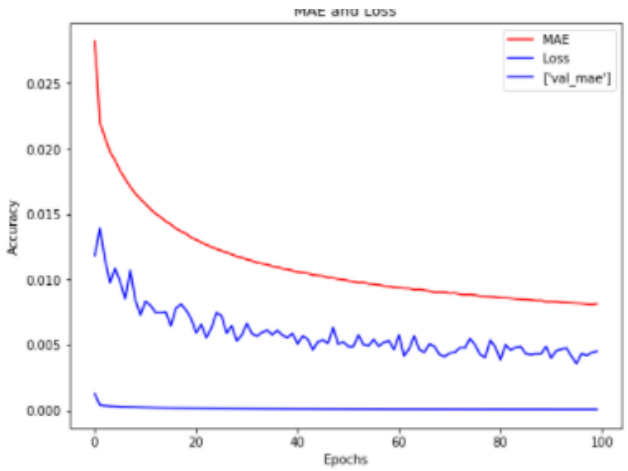


Fuente: Elaboración propia a partir de datos propios

En la gráfica inmediatamente anterior, se puede apreciar el comportamiento de la serie original y la pronosticada sobre los datos de validación, esto quiere decir, que son las predicciones sobre datos que realmente el modelo nunca ha visto, por lo tanto, es la aproximación más real al comportamiento del algoritmo en producción, pues estos datos no fueron utilizados para el entrenamiento y fijación de los pesos, pero tampoco para la escogencia del modelo, por lo cual estos datos no están contaminados y es una muy buena aproximación al comportamiento que podría llegar a tener el algoritmo en la vida real. Ahora bien, dicho lo anterior podemos ver que en general el algoritmo alcanza a captar los movimiento y tendencia de la serie original, si bien es cierto que en la primera mitad del periodo las aproximaciones están un poco por encima de la serie original, la dirección del movimiento, es decir, hacia arriba o hacia abajo se respeta y es bastante acertado, con lo cual logramos un poder de predicción bastante bueno, como se verá en el resultado del algoritmo de trading.

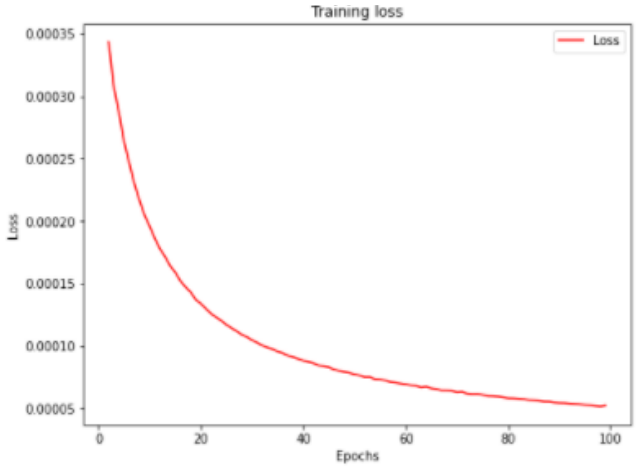
Convergencia

Ilustración 24:LSTM MAE



Fuente: Elaboración propia a partir de datos propios

Ilustración 23:LSTM Loss



Fuente: Elaboración propia a partir de datos propios

Para tener buenos resultados en el pronóstico de los modelos, es de fundamental importancia que a lo largo de la fase de entrenamiento el modelo tenga un comportamiento determinado. De acuerdo con lo anterior hay ciertos comportamientos que son deseables cuando se entrena el modelo. En primer lugar, como podemos ver en el grafica del lado izquierdo, las funciones de error absoluto del pronóstico del modelo y el error en el conjunto de datos de pruebas disminuyen a lo largo de cada una de las épocas, hasta que en teoría se encuentran y se vuelven constantes, como es posible visualizar en el modelo. Por otra parte, cuando la gráfica de error de entrenamiento y la de prueba se interceptan y la de prueba tiene una tendencia hacia arriba, quiere decir que se está sobre ajustando el modelo ("Overfitting"[1]) y por lo tanto el modelo está aprendiendo los parámetros y no tiene poder predictivo.

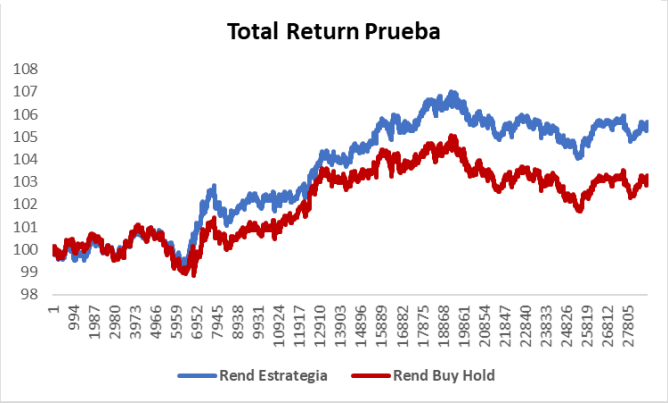
De acuerdo con lo anterior, es posible observar que el comportamiento de este modelo es el correcto, ya que las funciones de error absoluto van disminuyendo sin interceptarse a lo largo de las 100 épocas. Por otro lado, en la gráfica derecha podemos ver cómo va disminuyendo la función de perdida, en este caso la función es la (Hubber – loss) expuesta en el numeral 1, en teoría se dice que la función de perdida debe llegar a un punto donde

el ratio de cambio ya es casi imperceptible, para decir que el modelo converge, sin embargo como ya se mencionó, por poder computacional y tiempo de cálculo, se decidió que el modelo termina en la época 100, con lo cual se logra un muy buen rendimiento en el pronóstico y no exponemos el modelo a un sobre ajuste.

Finalmente, hay que resaltar el comportamiento del error medio absoluto en el conjunto de datos de prueba, ya que debería ser un curva que baja suavemente como en el caso de la curva de entrenamiento, esto puede deberse a que el parámetro del ratio de aprendizaje no está bien calibrado, faltan más ejemplos para entrenar o más variables de entrada, sin embargo después de haber probado varias arquitecturas, esta fue la que mejor resultado arrojo y como ya mencionamos, aumentar el conjunto de datos no era una opción, dado el tiempo de computación.

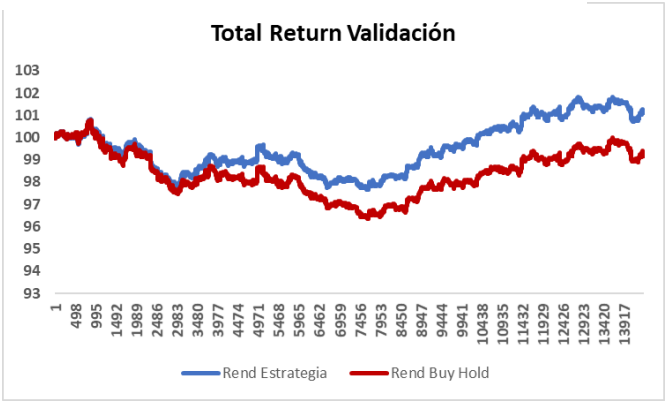
Resultados de la estrategia.

Ilustración 26:Retorno Toral Prueba



Fuente: Elaboración propia a partir de datos propios

Ilustración 25:Retorno Total validación



Fuente: Elaboración propia a partir de datos propios

Tabla 2: Retorno Toal LSTM Nominal

Retorno Nominal					
Datos	inicio	Fin	Dias	Estrategia	Buy Hold
Train	23/05/2019	01/10/2020	497,00	no aplica	no aplica
Test	01/10/2020	22/02/2021	144,00	5,58%	3,20%
Validation	22/02/2021	06/05/2021	73,00	1,16%	-0,75%

Tabla 3:Retorno Total Anual LSTM

Retorno Anual					
Datos	inicio	Fin	Dias	Estrategia	Buy Hold
Train	23/05/2019	01/10/2020	497,00	no aplica	no aplica
Test	01/10/2020	22/02/2021	144,00	14,77%	8,31%
Validation	22/02/2021	06/05/2021	73,00	5,93%	-3,70%

3.4. Red Convolucional

Convoluciones

Una convolución pude ser definida de manera general para dos señales de una dimensión según (Anastasia Borovykh, 2018) como $f\ y\ g$ escritas como $f * g$

$$(f * g)(i) = \sum_{j=-\infty}^{\infty} f(j)j(i-j),$$

En donde dependiendo de la definición de convolución, para ejemplos inexistentes estos deben tomar el valor de cero, o solo calcular los valores donde hay puntos en las dos señales. Además, las convoluciones son conmutativas $(f * g) = g * f$. Ahora asumamos que las señales son finitas, en este caso la convolución infinita debe ser truncada. Así pues, si suponemos que $f = |f(0), \dots, f(N-1)|$ Y $g = |g(0), \dots., g(M-1)|$ entonces la convolución de dos señales de puede definir como:

$$(f * g)(i) = \sum_{j=0}^{M} f(j)j(i-j),$$

Red Neuronal Convolucional

De acuerdo con la definición de (Arthur Le Guennec, 2016) de una red convolucional en el foro de analítica avanzada y aprendizaje de datos temporales, son modelos de clasificación compuestos por unidades fundamentales llamadas neuronas, cada una de estas neuronas está asociada a un peso y una serie de pesos que derivan de otras neuronas. Una de las arquitecturas más utilizadas es la de conexión completa entre capas, donde las neuronas son organizadas en capas, donde los parámetros de entrada de una capa son lo de salida de otra. Estas arquitecturas son muy poderosas ya que permiten hallar relaciones muy complejas.

De manera más formal, los parámetros de entrada para una capa de una red neural convolucional son de tres dimensiones: altura, peso y numero de parámetros. Ahora consideremos una entrada de una sola dimensión $x = (x)_{t=0}\ ^{N-1}$ de un tamaño N donde no se admiten valores nulos. Según (Anastasia Borovykh, 2018) los parámetros de salida de la primera capa están dados por hacer la convolución entre cada uno de los filtros w_h^1 para $g = 1, \dots, M$

$$a^1(i,h) = (w_h^1 * x)(i) = \sum_{j=-\infty}^{\infty} w_h^1(j)x(i-j)$$

Donde $w_h^1 \in \mathbb{R}^{1xkx1}$ y $a^1 \in \mathbb{R}^{1xN-k+1xM_1}$ nótese que el número de dimensiones de entrada, en este caso son uno, por lo tanto, el matriz de pesos solo tiene un canal, muy parecido a una red neural recurrente, donde luego el resultado pasa por una función no lineal $h(\cdot)$ para poder calcular $f^1 = h(a^1)$.

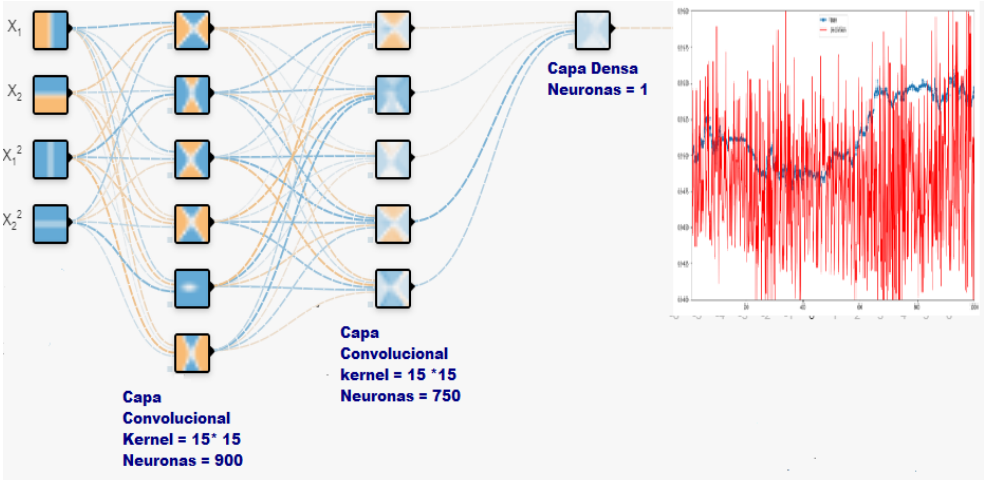
En cada una de las capas siguientes $l = 2, \dots, L$ la entrada que debe ser transformada, $f^{-1} \in \mathbb{R}^{1xN_{t-1}-k+1xM_{t-1}}$, donde $1\ x\ N_{t-1}\ x\ M_{t-1}$ es la dimensión de la salida de la función de transformación de la convolución anterior con $N_{t-1} = N_{t-2} - k + 1$ es convolucionada con un conjunto de M_t filtros $w_h^1 \in \mathbb{R}^{1xk-M_t}$, $h = 1, \dots, M_t$ esto para poder crear la siguiente tranformacion $a^l \in \mathbb{R}^{1xN_txM_t}$

$$a^l(i,h) = \left(w_h^l * f^{l-1}\right)(i) = \sum_{j=-\infty}^{\infty} \sum_{m=1}^{M_{t-1}} w_h^l(j,m) f^{l-1}(i-j,m)$$

Finalmente, la salida es a través de una función $f^l = h(a^l)$. El tamaño del filtro k y una salida de cada una de las capas con una anchura de $N_t = N_{t-1} - k + 1$ para todo $l = 1, \dots, L$

Arquitectura

Ilustración 27:configuración Convolucional

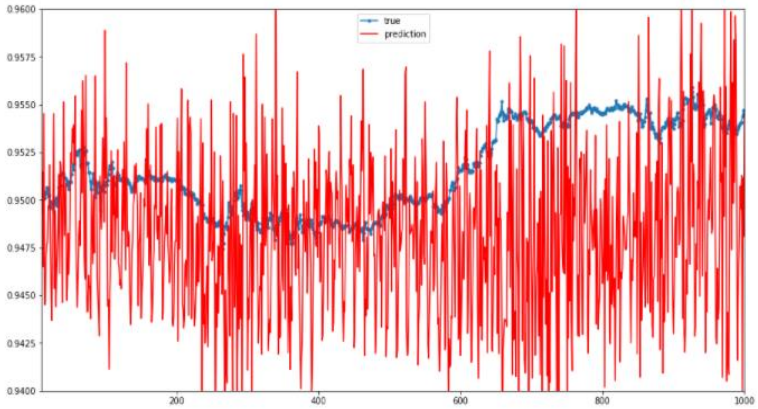


Fuente: Elaboración propia a partir de datos propios

Resultados

Datos de prueba

Ilustración 28:Pronostico Convolucional prueba

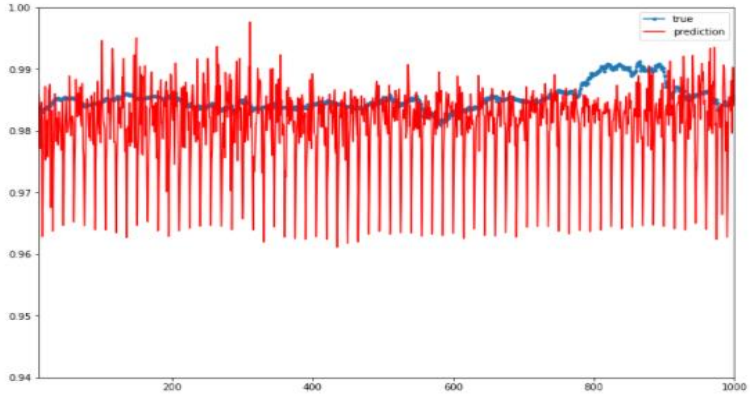


Fuente: Elaboración propia a partir de datos propios

En lo figura 28, se puede visualizar el resultado de las predicciones bajo un modelo con una capa convolucional y una variable de salida lineal densa. En el modelo sugerido podemos ver como los resultados de la predicción no son muy acertados, pues como podemos ver los puntos pronosticados oscilan alrededor de la serie de precios con una longitud muy amplia, lo cual logra que el error absoluto medio sea muy pequeño, ya que anula los picos altos con los bajos, pero no es para nada una buena aproximación, ya que no representa ninguna de las propiedades de la serie, ni la magnitud de los cambios ni la tendencia de la misma, por esta razón esta arquitectura no es una buena estrategia de predicción y por lo tanto no es beneficiosa para el algoritmo.

Datos de Validación

Ilustración 29:Pedriccion Convolucional validación

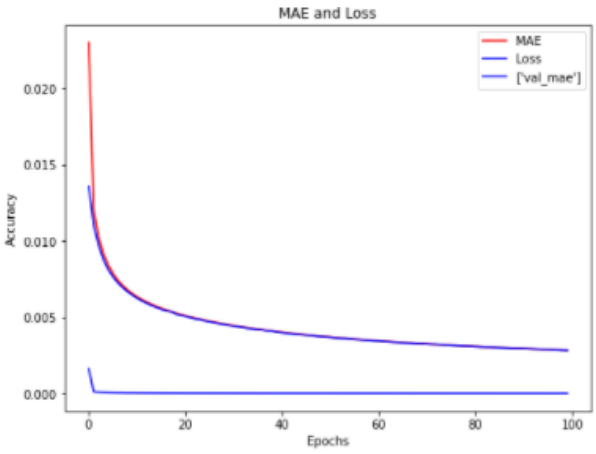


Fuente: Elaboración propia a partir de datos propios

Como es perceptible en los datos de validación, el pronóstico tampoco se comporta de una manera adecuada, pues al igual que en los datos de prueba, los puntos pronosticados solo oscilan alrededor de una media, minimizando el error, mas no tiene ningún sentido económico para la inversión y por lo cual descartamos esta arquitectura

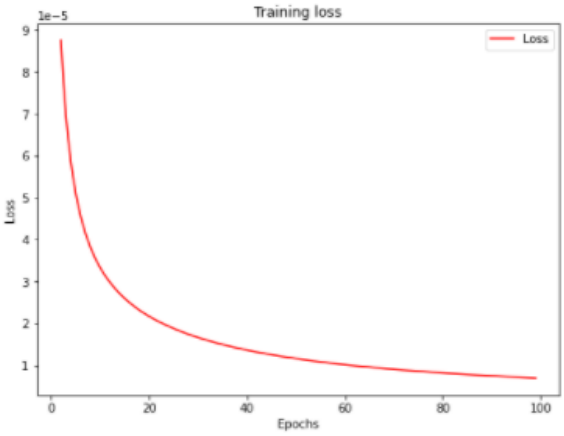
Convergencia

Ilustración 31:Convolutional MAE



Fuente: Elaboración propia a partir de datos propios

Ilustración 30:Convolucional Loss



Fuente: Elaboración propia a partir de datos propios

En cuanto al desempeño del modelo en la fase de entrenamiento, podemos observar en las gráficas superiores, que tiene un comportamiento ideal, pues las curvas decrecen paulatinamente a medida que aumenta el número de épocas. Sin embargo y como ya mencionamos, en este modelo en particular logramos disminuir el error y que la función de perdida también decrezca, pero aparentemente es por la oscilación de los pronósticos sobre la serie real, logrando que los errores muy grandes negativos sean compensados por lo positivos, pero el modelo tiene un comportamiento pobre en términos de predicción.

Resultados estrategia.

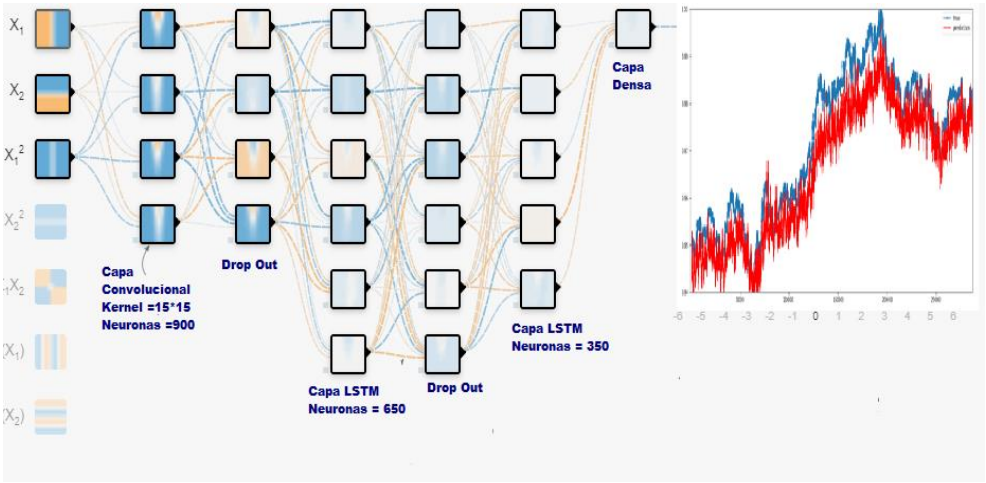
Dado que los resultados de las predicciones no tienen ningún sentido económico aparente, no se evalúa el comportamiento del algoritmo de trading con este método de predicción, pues los resultados no tendrían ninguna validez.

3.5. Red Neuronal Convolucional Completamente conectada con LSTM

La idea de básica de una red neuronal convolucional con un LSTM, es básicamente que la selección de características se haga en la capa convolucional y una vez se hayan depurado estas características haya un análisis más detallado de la serie en las capas de la LSTM. Se toma este modelo en base a la investigación (FAZLE KARIM1, 2017) pues determina que es más eficiente tener una capa convolucional para la escogencia de características, pues es computacionalmente más eficiente y luego dado la recurrencia de una serie temporal ajustar esta relaciones mediante un LSTM resulta ideal.

Arquitectura

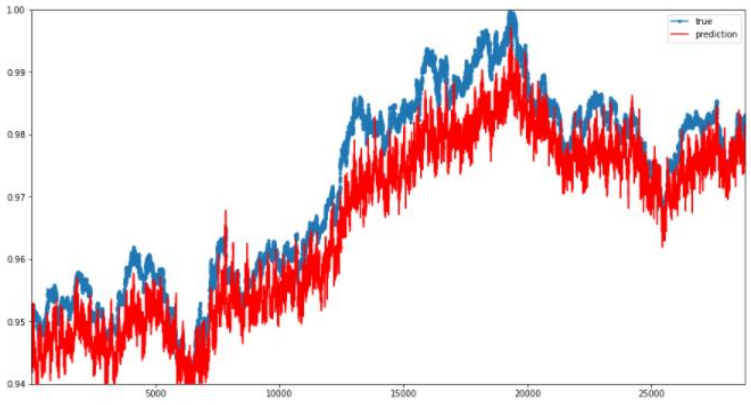
Ilustración 32:Configuracion Conv LSTM



Fuente: Elaboración propia a partir de datos propios

Datos de Prueba

Ilustración 33:Prediccion Conv LSTM Prueba

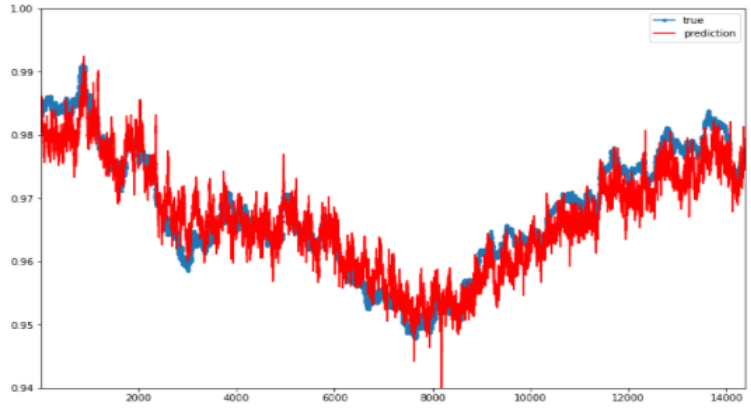


Fuente: Elaboración propia a partir de datos propios

Como se pude ver en la figura 33, el comportamiento de las series de tiempo tanto del pronóstico, como de la serie original son muy similares, si bien la serie pronosticada está un poco por debajo de la serie original, cuando se evalúa el modelo, dentro de la estrategia de trading, observamos que la dirección del pronóstico en correcta, es decir, si sube o baja, logrando un desempeño aceptable en términos del retorno total.

Datos de validación

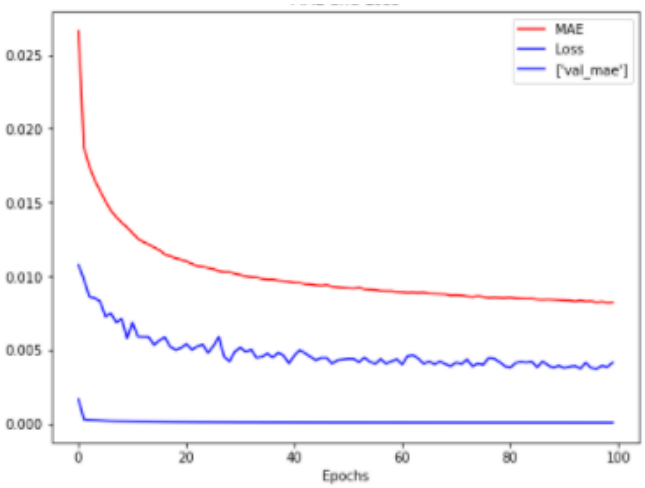
Ilustración 34:Prediccion Conv LSTM validación



Fuente: Elaboración propia a partir de datos propios

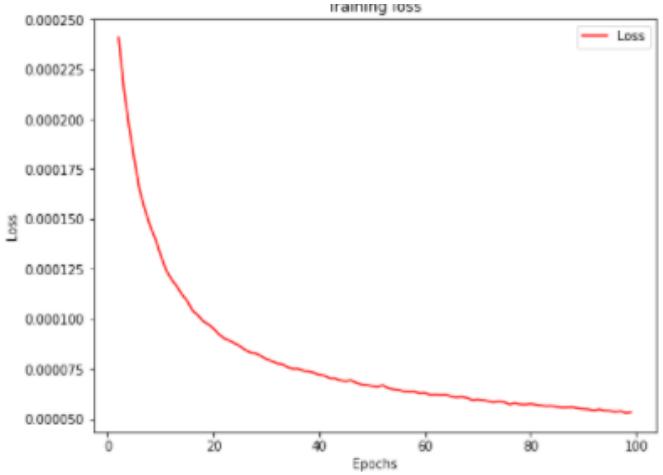
En la gráfica anterior, podemos ver el pronóstico de cada uno de los puntos de la serie en el conjunto de datos de validación. Hay que resaltar que en este caso en particular la serie pronosticada se ajusta mejor a los datos que nunca ha visto que en el modelo de los LSTM, sin una capa convolucional, pues podemos observar que capta toda la tendencia de la serie y sus métricas desempeño están bastante ajustadas.

Ilustración 36:Conv LSTM MAE



Fuente: Elaboración propia a partir de datos propios

Ilustración 35:Conv LSTM Loss

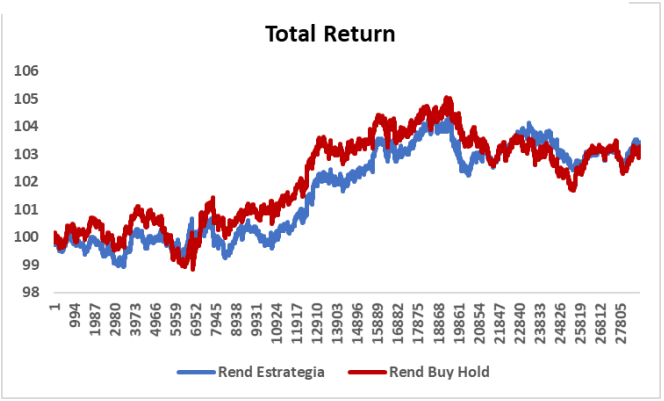


Fuente: Elaboración propia a partir de datos propios

En cuanto al ajuste del modelo, podemos ver que sus medidas de desempeño se comportan de acuerdo a lo esperado, es decir, el valor de la función de perdida va decreciendo paulatinamente, los errores absolutos también decrecen a medida que pasan la épocas, pero como sucedida en el modelo de los LSTM el error en el conjunto de prueba va decreciendo de una manera sinuosa, esto como ya mencionaos se puede deber al ratio de aprendizaje, a falta de más ejemplos de entrenamiento o falta de una espacio de características mayor, si bien no es lo más deseable el comportamiento no deja de ser bueno para el entrenamiento del modelo.

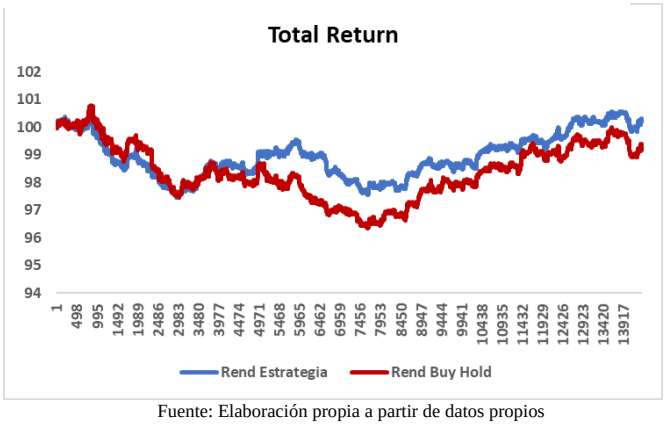
Resultados

Ilustración 38:Conv LSTM Retorno datos prueba



Fuente: Elaboración propia a partir de datos propios

Ilustración 37:Conv LSTM Retorno Datos validación



Fuente: Elaboración propia a partir de datos propios

Tabla 4:Conv LSTM Retorno nominal

Retorno Nominal					
Datos	inicio	Fin	Dias	Estrategia	Buy Hold
Train	23/05/2019	01/10/2020	497,00	no aplica	no aplica
Test	01/10/2020	22/02/2021	144,00	3,41%	3,20%
Validation	22/02/2021	06/05/2021	73,00	0,27%	-0,75%

Tabla 5:Conv LSTM retorno Anualizado

Retorno Anual					
Datos	inicio	Fin	Dias	Estrategia	Buy Hold
Train	23/05/2019	01/10/2020	497,00	no aplica	no aplica
Test	01/10/2020	22/02/2021	144,00	8,86%	8,31%
Validation	22/02/2021	06/05/2021	73,00	1,37%	-3,70%

3.6. Análisis de retornos

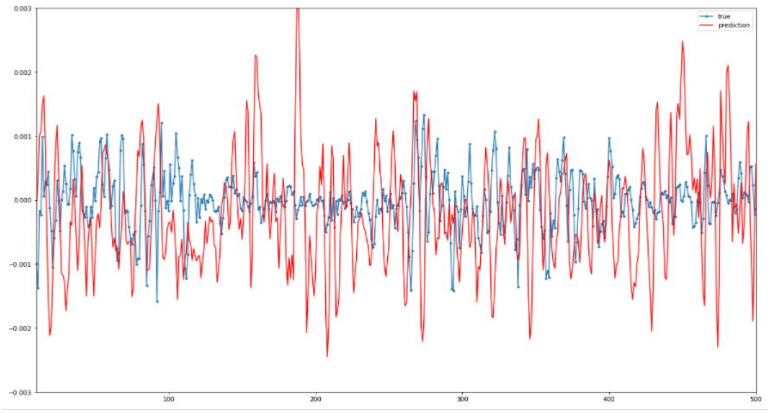
En el análisis clásico de series de tiempo y series financieras, usualmente se habla sobre el análisis de los retornos, dado que estos son estacionarios en media y desde un punto de vista econométrico clásico son más adecuados para el análisis. De acuerdo con lo anterior y teniendo en cuenta que usualmente las series no son estacionarias, al diferenciarlas tenemos varios efectos beneficiosos, tanto para el pronóstico clásico como en redes neuronales, esto ya que los datos de la serie van a estar centrados alrededor del cero por lo que el modelo podrá aprender mejor.

De acuerdo con lo anterior, el propósito de esta sección es realizar un pronóstico de los retornos, en lugar de la serie de precios, con lo cual se espera obtener un mejor resultado en la predicción. Así pues, tomaremos el modelo con el mejor desempeño de la sección anterior, dado el análisis exploratorio previo y aplicaremos el mismo modelo, pero sobre la serie diferenciada.

3.6.1. Redes Neuronales Recurrentes LSTM (Long short-term memory) Retornor

Datos de prueba

Ilustración 39:LTSM Retornos datos Prueba



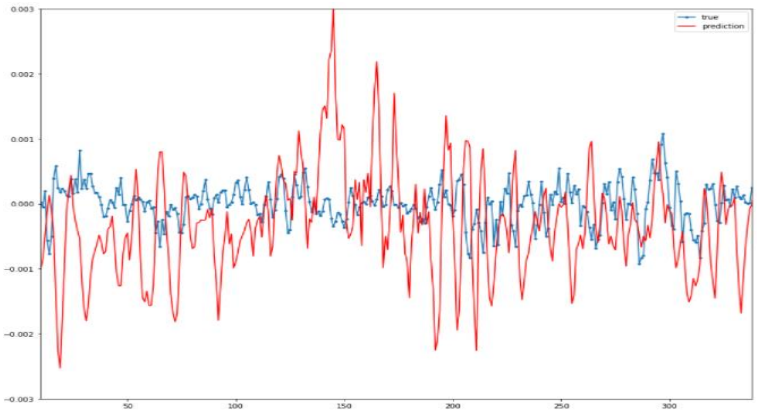
Fuente: Elaboración propia a partir de datos propios

Cuando aplicamos el modelo convolucional combinado con capas recurrentes LSTM sobre los retornos, encontramos que la predicción tiene una amplitud en general más grande que los retornos reales, es decir, el valor pronosticado siempre esta más a arriba o

más abajo del valor real, si bien es cierto que en general la predicción capta la señal, el movimiento de la serie tiene un desfasé.

Datos de validación

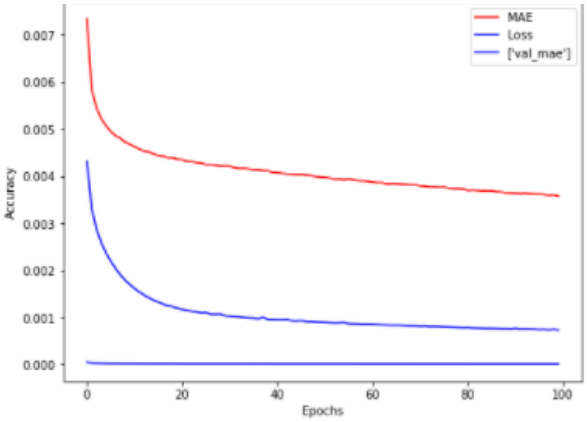
Ilustración 40: LSTM Retorno Datos validación



Fuente: Elaboración propia a partir de datos propios

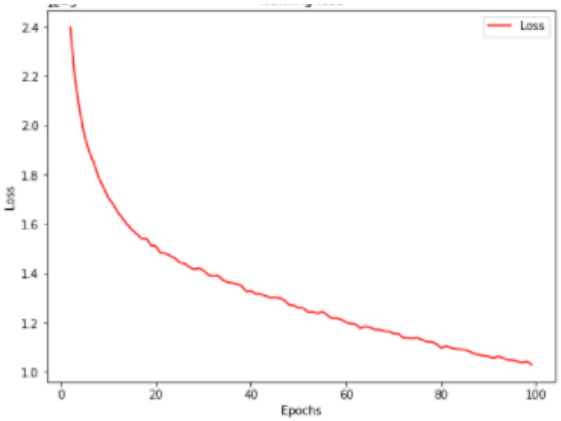
Al igual que en el conjunto de datos de prueba, los retornos pronosticados tienen una amplitud mucho más grande, por lo cual no calzan en la misma magnitud que los retornos reales. Por tal motivo y al igual que en el conjunto de prueba, vamos a utilizar estos retornos como un variable dicotómica, si el valor del activo sube o baja en determinado tiempo t.

Ilustración 42:LSTM Retornos MAE



Fuente: Elaboración propia a partir de datos propios

Ilustración 41:LSTM Retornos Loss



Fuente: Elaboración propia a partir de datos propios

Las variables de control del progreso del entrenamiento de la red sobre los retornos están bastante ajustadas a lo que podría esperarse, pues como ya se ha mencionado repetidas veces el valor de la función de perdida decrece paulatinamente de acuerdo con lo esperado y los errores absolutos medios también decrecen como se esperan. En particular como en el modelo convolucional, los errores decrecen y son pequeños dado que los valores están por encima o por debajo de la serie cancelándose.

3.7. Modelo de clasificación

3.7.1. Red Neuronal Convolucional Completamente conectada con LSTM

En este apartado lo que se busca es hacer una predicción sobre los retornos de la serie de tiempo, pero desde un punto de vista de clasificación, es decir, trazar una frontera que pueda dividir los datos de acuerdo con unos parámetros establecidos. Para este efecto se ha decidió clasificar los retornos en tres subconjuntos

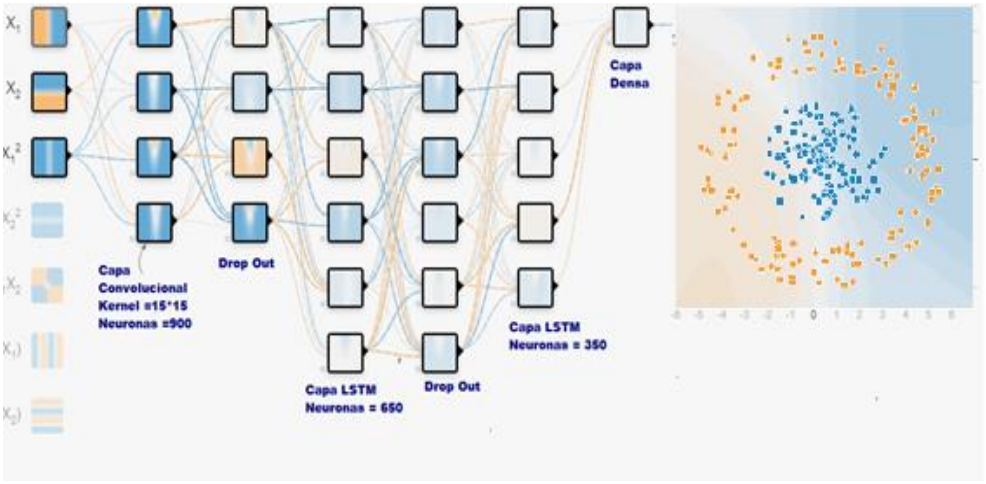
$$r_i \begin{cases} 0 & \forall, r_i > 0.00006 \\ 1 & -0.00003 \leq r_i \leq 0.00006 \\ 2 & \forall, r_i < -0.00003 \end{cases}$$

$$r_i : Clasificacion\ del\ retorno\ i$$

Las clasificaciones anteriores se hacen principalmente para poder trazar una frontera que divida los retornos de acuerdo con estos rangos y con los datos de entrada poder clasificarlos. Esto resulta muy beneficioso ya que tenemos un gran conjunto de datos para clasificar solo tres categorías. Esta estrategia resulta muy beneficiosa, ya que, aunque no sabemos exactamente el retorno esperado, podemos tener una intuición si el retorno va a estar en un determinado nivel y si va ser una buena compra.

Arquitectura

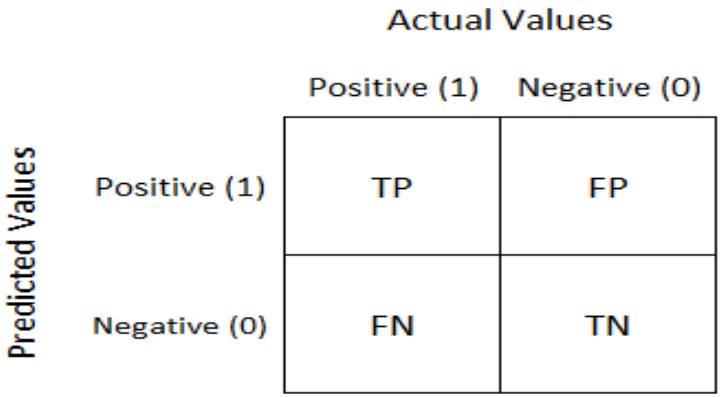
Ilustración 43:Clasificacion Conv LSTM



Fuente: Elaboración propia a partir de datos propios

Matriz de confusión

Ilustración 44: confusión Matrix explicación

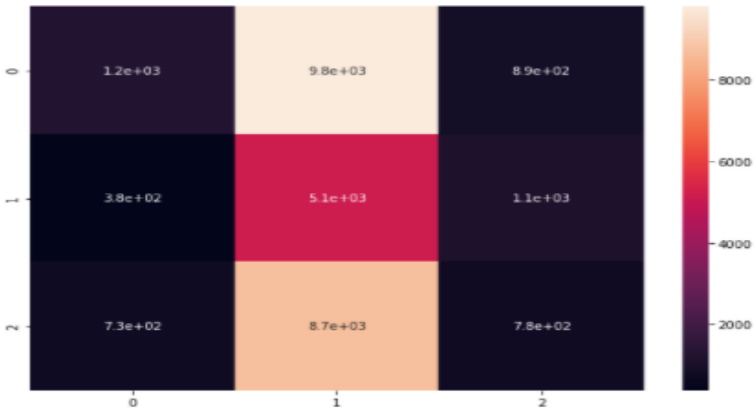


Fuente: Tomado de https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

En los modelos de clasificación la métrica sobre el desempeño del modelo es diferente al utilizado al predecir las series de precios, pues en estos modelos en particular lo que buscamos es saber qué cantidad de los datos fueron categorizados en la categoría a la realmente corresponden, es decir, si el retorno está por encima de 0,00006 el modelo sea capaz de identificar que el retorno siguiente estará por encima de ese nivel. Para este efecto utilizamos una herramienta que se llama la matriz de confusión, que básicamente mide el nivel de acierto y desacierto del modelo en la clasificación, de esta manera la matriz de confusión tiene en su diagonal los ejemplos que fueron categorizados efectivamente en su categoría real y en la parte inferior y superior de la diagonal los ejemplo que pertenecen una categoría pero que el modelo los clasifico en otra.

Datos de prueba

Ilustración 45: confusión Matrix Datos Prueba



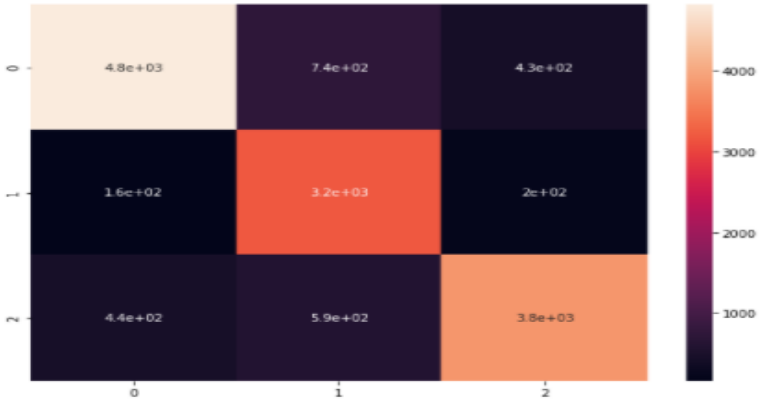
Fuente: Elaboración propia a partir de datos propios

los resultados observados en la matriz de confusión anterior, hacen referencia a la cantidad de veces que el algoritmo ha clasificado correctamente la información, como ya mencionamos, en la diagonal podemos observar el número de veces sobre el total de muestras que corresponden a una categoría que el algoritmo ha etiquetado de una manera correcta. así pues con los resultados arrojados tenemos un nivel de aciertos (Acuracy) de

81.93%, lo cual quiere decir que sobre todas las muestras del conjunto de prueba, el modelo clasifico este porcentaje en la categoría correcta y se equivocó el 19% de las veces, esto va a ser de suma importancia en el rendimiento del algoritmo, dado que dividimos el conjunto en tres categorías y podría confundir un retorno positivo, cuando realmente es negativo, tomando un decisión de inversión negativa perjudicando el retorno total.

Datos de validación

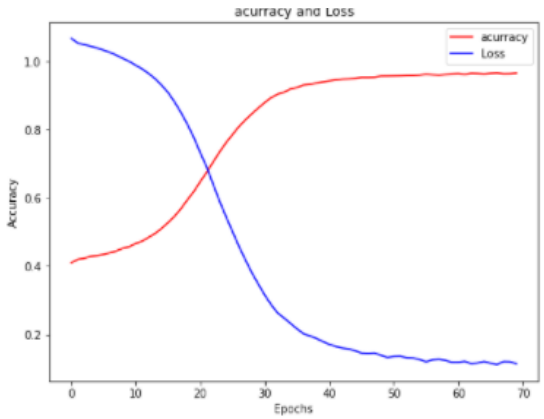
Ilustración 46:Confusion Matrix datos Validacion



Fuente: Elaboración propia a partir de datos propios

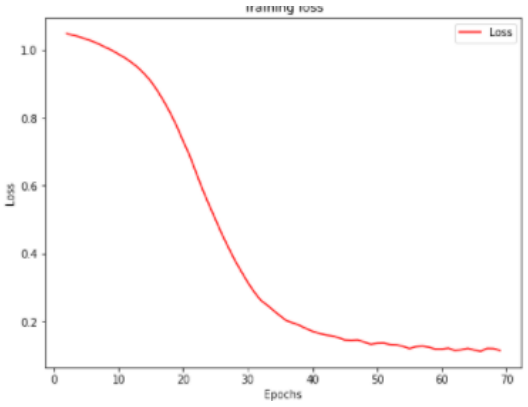
cómo podemos ver en la matriz de confusión anterior sobre los datos de validación, la cantidad de aciertos sobre el conjunto de datos fue en este caso de 83.25%, muy similar a lo observado en el conjunto de prueba, lo cual nos dice que el modelo ha generalizado bien las características de entrada. Ahora bien, veremos en el algoritmo de trading cual es la influencia de ese 17% equivocación y cómo influye en el retorno total.

Ilustración 48:Accuracy



Fuente: Elaboración propia a partir de datos propios

Ilustración 47:Clasificacion loss

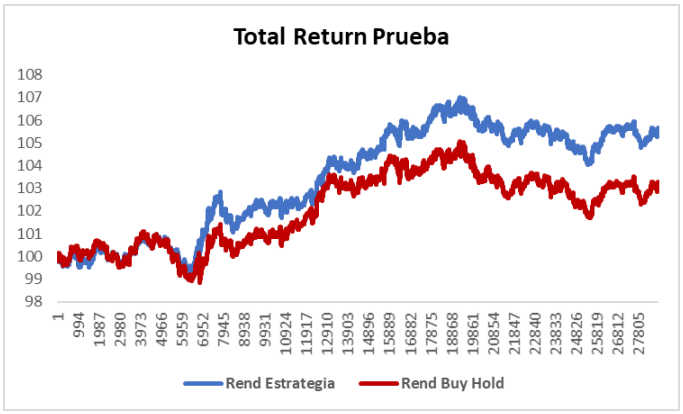


Fuente: Elaboración propia a partir de datos propios

Como podemos ver en las gráficas anteriores el modelo tiene unas métricas bastante buenas y un comportamiento deseado en el periodo de entrenamiento, pues como vemos la función de pérdida del lado izquierdo, decrece paulatinamente hasta empezar a disminuir muy poco. Por su parte, el asertividad del modelo (Acuracy) va aumentando a medida que las épocas aumentan, lo cual es esperable y nos dice que el modelo está aprendiendo de los inputs de entrada.

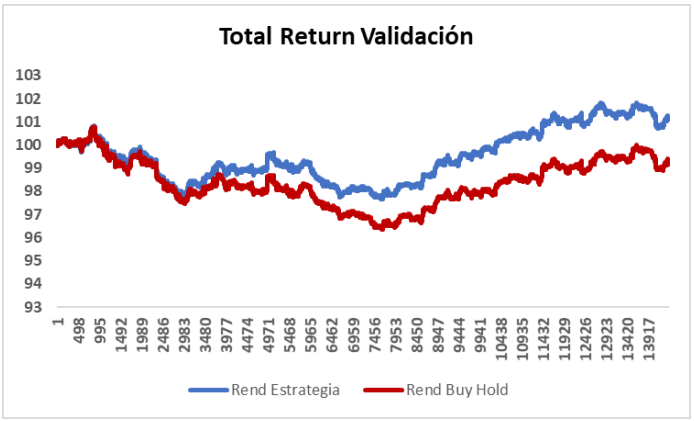
Resultados estrategia

Ilustración 50::Retorno clasificación datos Prueba



Fuente: Elaboración propia a partir de datos propios

Ilustración 49:Retorno clasificación datos validación



Fuente: Elaboración propia a partir de datos propios

Ilustración 51:Retorno Nominal y Anual Estrategia

Retorno Nominal					
Datos	inicio	Fin	Dias	Estrategia	Buy Hold
Train	23/05/2019	01/10/2020	497,00	no aplica	no aplica
Test	01/10/2020	22/02/2021	144,00	4,13%	3,20%
Validation	22/02/2021	06/05/2021	73,00	1,16%	-0,75%

Retorno Anual					
Datos	inicio	Fin	Dias	Estrategia	Buy Hold
Train	23/05/2019	01/10/2020	497,00	no aplica	no aplica
Test	01/10/2020	22/02/2021	144,00	10,79%	8,31%
Validation	22/02/2021	06/05/2021	73,00	5,93%	-3,70%

3.8. Emsemble learning y Selección de parámetros de modelo.

Los modelos de Esembel segun (Zhou, 2009) es un paradigma del aprendizaje automático, donde múltiples aprendices son entrenados para resolver el mismo problema. Contrario al paradigma clásico de aprendizaje automático donde la idea es tomar una hipótesis para entrenar los datos, por su parte los modelos de ensamble tratan de construir un conjunto de hipótesis para luego combinarlas y tener un mejor desempeño.

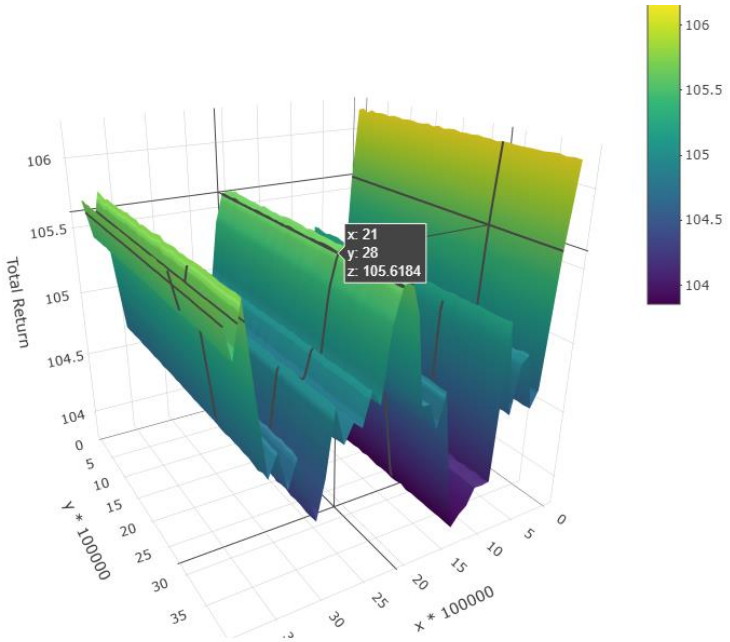
De acuerdo con lo anterior, lo que buscamos en esta sección con los modelos de ensemble es combinar tanto el modelo de regresión creado con el modelo LSTM visto exclusivamente desde una perspectiva de regresión y el modelo de clasificación de los retornos que determina en que rango están los mismo y observar si el siguiente precio está en el rango óptimo de compra.

Selección de parámetros

En esta sección exploraremos los mejores parámetros para el algoritmo de trading como tal, es decir, variar los parámetros en los cuales el algoritmo decide comprar y vender dentro de los datos de prueba, para determinar en qué niveles el algoritmo alcanza su mejor desempeño, en este caso el mejor retorno total. Para lo anterior se plantea hacer los siguientes cambios al modelo.

1. Variar el parámetro de pérdida acumulada que determina cuando se vende el activo cuando tiene posición en este.
2. Variar el nivel de retorno requerido para efectivamente tomar posición en el activo o no tomar posición en el.

Ilustración 52:Superficie optimización estrategia



Fuente: Elaboración propia a partir de datos propios

cómo podemos ver en la gráfica anterior, en el eje x tenemos el nivel de retorno requerido en el pronóstico para realizar una orden de compra, y en el eje y podemos ver los retornos negativos con los cuales tomamos la decisión de vender el activo de inmediato, estas dos variables están en función del retorno total. Así pues, evaluamos en todo el periodo de prueba todas las combinaciones de estas dos variables aumentando 0.00001 cada vez al nivel de compra y disminuyendo la misma cantidad para la venta del activo, con esto lo que se desea es encontrar el par de parámetros que maximiza el algoritmo ya implementando y posteriormente evaluarlo en el conjunto de datos de validación para finalmente evaluar el desempeño de todo el modelo general.

Parámetros

$$Retorno\ Acumulado\ venta\ RT < 0.009$$

$$Retorno\ pronosticdo < ES_i$$

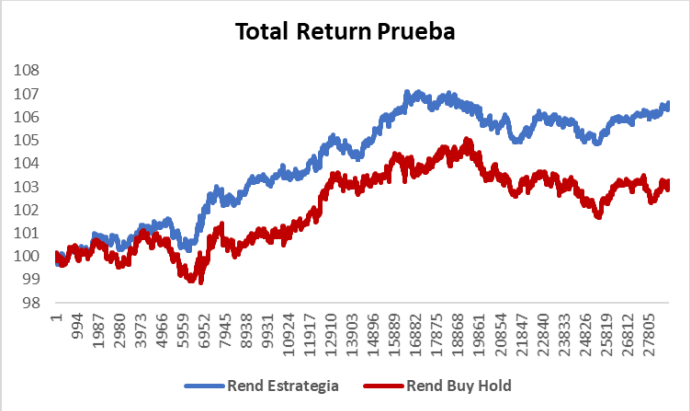
$$Compra\ a\ partir\ de\ nivel\ de\ retorno\ pronosticado\ r_i > 0.0007$$

$$Venta\ a\ partir\ de\ nivel\ de\ retorno\ pronosticado\ r_i < -0.0006$$

Modelo Optimizado

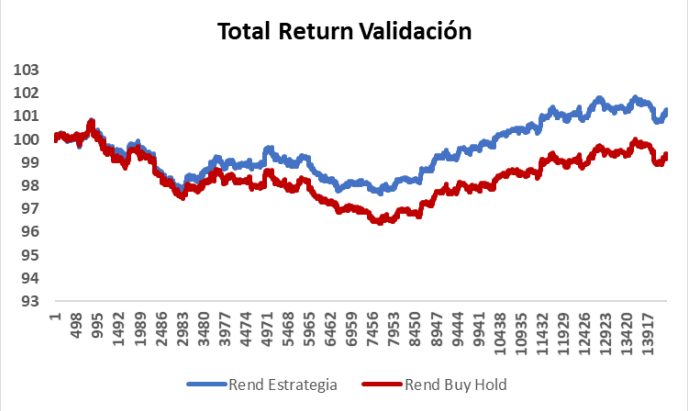
en este modelo tomamos como base la combinación del mejor modelo de regresión y el modelo de clasificación como un conjunto, donde la decisión de compra está basada en un ponderado de los dos pronósticos, es decir, se pondera la decisión de compra o venta de un activo con respecto si ambas predicciones son hacia arriba o hacia abajo, en este sentido, si el modelo de regresión predice un precio más alto que satisface las condiciones mínimas de retorno y adicionalmente si el modelo de clasificación predice una subida que se clasifica en el tercer grupo de clasificaciones un retorno por encima de 0.00007, entonces el algoritmo va a comprar, por el contrario, si el modelo de regresión predice una bajada y el retorno negativo resultante es igual o más bajo que el límite que hemos impuesto y además el modelo de clasificación categoriza el retorno siguiente por debajo de -0,00006 entonces el modelo deberá vender el activo.

Ilustración 54:Retorno estrategia final datos prueba



Fuente: Elaboración propia a partir de datos propios

Ilustración 53:Retorno estrategia final datos validación



Fuente: Elaboración propia a partir de datos propios

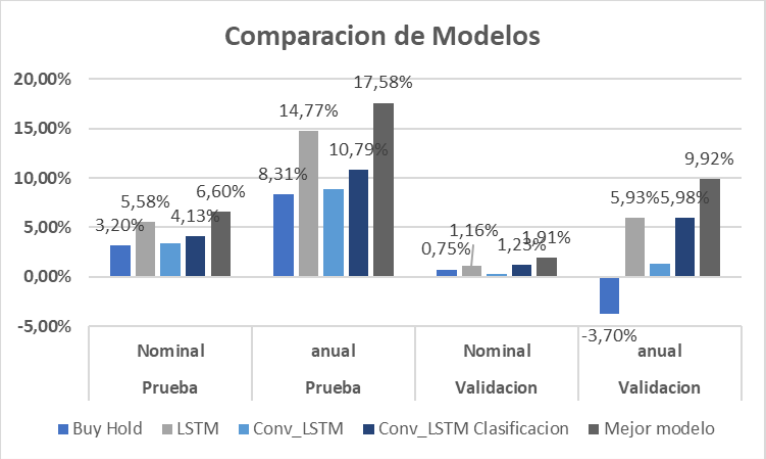
Tabla 6: Retorno nominal y anualizado Estrategia

Retorno Nominal					
Datos	inicio	Fin	Dias	Estrategia	Buy Hold
Train	23/05/2019	01/10/2020	497,00	no aplica	no aplica
Test	01/10/2020	22/02/2021	144,00	6,60%	3,20%
Validation	22/02/2021	06/05/2021	73,00	1,91%	-0,75%

Retorno Anual					
Datos	inicio	Fin	Dias	Estrategia	Buy Hold
Train	23/05/2019	01/10/2020	497,00	no aplica	no aplica
Test	01/10/2020	22/02/2021	144,00	17,58%	8,31%
Validation	22/02/2021	06/05/2021	73,00	9,92%	-3,70%

Comparación de modelos

Ilustración 55:Comparacion de modelos



Fuente: Elaboración propia a partir de datos propios

4. CONCLUSIONES

Las redes neuronales y el *machine learning* son herramientas muy poderosas para el pronóstico de diferentes tipos de datos, ya que logran hacer relaciones muy complejas y no lineales entre estos. Adicionalmente, estos algoritmos son muy robustos cuando se tiene una gran cantidad de datos para entrenar los modelos, ya que así tienen una gran cantidad de información para hacer inferencias y generar pronósticos más acertados. En particular y como se ha desarrollado a lo largo de este trabajo final de máster, resultan una herramienta muy sólida para el pronóstico de series financieras, teniendo en cuenta su complejidad y aleatoriedad.

Se encontró que el pronóstico de series financieras desde un punto de vista clásico de regresión utilizando la serie de precios y redes neuronales es un método muy efectivo para hacer predicciones directamente sobre los precios y sin que la serie sea estacionaria como sí es un requerimiento en el pronóstico econométrico. Como se evidenció con una arquitectura de redes neuronales basadas en LSTM **(***long short-term memory***)**, se logra captar la tendencia y movimiento de la serie. Este efecto en particular es logrado por la estructura de refuerzo y la interconexión que tiene cada una de las neuronas dentro de la misma capa y las relaciones no lineales que logra hacer entre los datos.

Por su parte, cuando vemos el problema de regresión desde un punto de vista de clasificación, este tiene un enfoque diferente, pues lo que se busca no es directamente predecir el retorno o el precio del activo en el momento t+1, sino agrupar suficientes ejemplos en clases más grandes, es decir, agrupamos en un rango una serie de datos y luego los clasificamos, en este caso se divide todo el rango de los retornos en 3 para ser clasificados. Para esta tarea sí que fue muy beneficioso el uso de una capa convolucional y luego capas de LSTM, pues como se mencionó en diferentes artículos, una capa convolucional resulta muy apropiada en problemas de clasificación de series temporales,

dado que es muy eficiente a la hora de escoger los parámetros necesarios, que luego serán entrenados en las capas LSTM que sí buscan las relaciones en el tiempo de las variables.

En cuanto al desempeño del algoritmo de *trading* planteado, logró un desempeño mayor en promedio, en todas las arquitecturas de un 1,3 % nominal, a una estrategia (*buy and hold*) comprar un activo en un periodo y venderlo n periodos después, comprobando que el algoritmo es válido. Ahora bien, esto no hubiese sido posible sin los modelos de predicción, pues es con base en estos que se toman las decisiones de compra y venta, de ahí la importancia de la construcción y validación de los modelos de predicción, pues como ya se discutió, si la capacidad predictiva es baja llevaría a tomar decisiones de inversión equivocas, afectando directamente al retorno total y a la rentabilidad para el inversionista.

Finalmente, las tecnologías de la información como el *big data* y el *machine learning* hacen que cada vez más industrias como el *trading* sean migrados a tecnologías autónomas con inteligencia artificial, dada la gran cantidad de datos que hay disponibles hoy y que permiten crear modelos muy robustos; sin embargo, esto no podría ser posible sin el conocimiento previo de la persona que crea el modelo, las características de entrada y las arquitecturas necesarias para entrenarlos, pues sin este se necesitaría un poder de computación infinito donde entran millones de ejemplos de entrada y características y es el mismo modelo quien escoge todos los parámetros, pero en su mayoría si no se tiene esa noción previa y un sentido de lo que se está haciendo, se cae en el los modelos basura entra basura sale *(gargabe in garbage out*), pues las relaciones intermedias no tienen un sentido aparente.

5. BIBLIOGRAFÍA

Aldridge, I. (2013). *High-Frequency Trading A Practical Guide to Algorithmic.* New Jersey: Wiley.

Anastasia Borovykh, S. B. (18 de 9 de 2018). Conditional time series forecasting with convolutional neural networks. New York, Ithaca, Estados Unidos.

Arthur Le Guennec, S. M. (2016). Data Augmentation for Time Series Classification using Convolutional Neural Networks. Italia.

Avatrade. (2020). Indicador RSI. de https://www.avatrade.es/educacion/professional-trading-strategies/rsi-trading-strategies

Buchanan, B. G. (Abril de 2019). *The Alan Turing Institute.*

Capital, E. (2019). Mastering the basics of technical. Londres, Inglaterra.

Charpentier, A., & Fermanian, J.-D. (2006). The Estimation of Copulas:. Francia.

FAZLE KARIM1, S. M. (22 de 10 de 2017). LSTM Fully Convolutional Networks. Illinois, Estados Unidos

Gracia, L. O. (2020). Quantitative Finance. Barcelona, España.

Haugh, M. (2016). An Introduction to Copulas. New York.

Investopedia. (03 de 2021). Bollinger Band Definition., de https://www.investopedia.com/terms/b/bollingerbands.asp

M.Merc`e Claramunt, A. C. (2020). Solvencia. Barcleona.

Mehryar Mohri, A. R. (2018). *Foundations of Machine Learning, second edition.* Massachusetts Institute of Technology.

Morral, R. R. (2020). High Frequency Trading via Deep. Barcelona, España.

Novales, A. (2017). Copulas. Madrid.

Qun Zhuge, L. X. (2017). LSTM neural network with emotional analysis for prediction of stock price. California, Estados Unidos.

Research, V. M. (2 de 2020). *Verified Market Research.*

S. Hochreiter and J. Schmidhuber. (1997). *Neural Computation, Volume 9.* Berlin.

SAS. (2021). *SAS*. de https://www.sas.com/es_es/insights/analytics/machine-learning.html

Suisse, C. (2019). Technical Analysis - Explained. Zurich, Switzerland.

Vila, O. P. (2014). *Machine Learning I: Introduction to Supervised Classification Methods.* Barcelona, España.

Vitrià, J. (2020). Introduction to deep learning. Barcelona, España. Recuperado el 21 de 05 de 2021

Zhou, Z.-H. (Septiembre de 2009). Ensemble Learning. Nanjing, China.