UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN

Predicción del índice IBEX 35 mediante una red neuronal recurrente de múltiples ramas.

Martín Guimerà Castell

Tutora: Valery Naranjo Ornedo

Cotutor: Adrián Colomer Granero

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2019-20

Valencia, 10 de septiembre de 2020

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

VLC/ CAMPUS VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS Gestión ambiental verificada REG.NO. ES-CV-000030

Agradecimientos

A Adrián y a Valery, que me han permitido introducirme en el fascinante mundo de la inteligencia artificial, y a todos los profesores que de una manera u otra han participado en este trabajo.

A todos mis amigos que me han ayudado tanto de manera incondicional durante toda mi trayectoria académica. En especial a Raúl, que ha hecho que estos últimos años se me hiciesen mucho más llevaderos.

Resumen

La predicción de la dirección de las tendencias del precio de las acciones es vital para un óptimo desarrollo de estrategias para las transacciones de la bolsa. Debido a los riesgos y rendimientos variables de la bolsa, la predicción de esta es un tema de mucha importancia para los que quieren invertir en ella. Tener la capacidad de pronosticar la tendencia o el precio de las acciones sería una información muy valiosa para los inversores. En el presente trabajo se ha tratado de encontrar algún modelo que sea viable para la predicción del índice del Ibex-35. Para ello, se ha realizado el estudio de 864 modelos distintos, con uso de redes neuronales recurrentes simples, LSTM y una combinación de convolucionales de una dimensión con LSTM para cada una de las ocho bases de datos que se proponen. Se realiza, finalmente, una comparativa entre los distintos modelos partiendo como *baseline* los RMSE obtenidos por las medias móviles y una red neuronal simple. Se concluye que el modelo de combinación convolucional con LSTM es el mejor predictor para el índice.

Palabras clave: Ibex-35; redes neuronales artificiales; LSTM; aprendizaje profundo; predicción bursátil; mercado de valores; RNN; índice bursátil; predicción de series temporales

Resum

La predicció de la direcció de les tendències del preu de les accions és vital per a un òptim desenvolupament d'estratègies per a les transaccions de la borsa. A causa dels riscos i rendiments variables de la borsa, la predicció d'aquesta és un tema de molta importància per als quals volen invertir en ella. Tindre la capacitat de pronosticar la tendència o el preu de les accions seria una informació molt valuosa per als inversors. En el present treball s'ha tractat de trobar algun model que siga viable per a la predicció de l'índex de l'Ibex-35. Per a això, s'ha realitzat l'estudi de 864 models diferents, amb ús de xarxes neuronals recurrents simples, LSTM i una combinació de convolucionals d'una dimensió amb LSTM per a cadascuna de les vuit bases de dades que es proposen. Es realitza, finalment, una comparativa entre els diferents models partint com *baseline* els RMSE obtinguts per les mitjanes mòbils i una xarxa neuronal simple. Es conclou que el model de combinació convolucional amb LSTM és el millor predictor per a l'índex.

Paraules clau: Ibex-35; xarxes neuronals artificials; LSTM; aprenentatge profund; predicció borsària; mercat de valors; RNN; índex borsari; predicció de sèries temporals

Abstract

Predicting the direction of stock price trends is vital for optimal development of stock market trading strategies. Due to the variable risks and returns of the stock market, stock market prediction is a very important issue for those who want to invest in it. Having the ability to forecast the trend or price of shares would be very valuable information for investors. In this work we have tried to find some viable model for the prediction of the Ibex-35 index. To this end, 864 different models have been studied, using simple recurrent neural networks, LSTM and a combination of one-dimensional convolutionals layer with LSTM for each of the eight databases proposed. Finally, a comparison between the different models is made, using as a baseline the RMSE obtained by the moving averages and a simple neuronal network. It is concluded that the convolutional combination model with LSTM is the best predictor for the index.

Keywords: Ibex-35; artificial neural networks; LSTM; deep learning; stock market prediction; RNN; stock market index; time series prediction

Índice

I. Índice de figuras

II. Índice de tablas

III. Índice de ecuaciones

1. Introducción

1.1 Resumen

Desde hace décadas se han tratado de desarrollar métodos capaces de predecir de la mejor manera posible el mercado de valores. Tener la capacidad de conocer cómo se comportarán el precio de las acciones es de un valor incalculable para cualquier *trader* que quiera invertir en bolsa, debido a los riesgos y rendimientos variables de esta. Por ello muchos tipos de análisis y de predicción se han ido desarrollando, como el análisis técnico, estudiando las tendencias para predecir el futuro mediante el empleo de distintos indicadores técnicos, o como el análisis fundamental, en el que se intenta predecir el valor de la acción empleando componentes macroeconómicos y estados financieros de la empresa estudiada. Sin embargo, el mercado de valores es considerado muy incierto por las muchas variables que pueden afectar al precio de las acciones de una empresa como para que el valor de estas pueda ser predecible. (Huang & Lin, 2014).

En los últimos años, el empleo de redes neuronales artificiales ha crecido de manera exponencial con aplicación en muchos campos distintos. Las redes neuronales se están utilizando, solas o combinadas con otros métodos, para obtener los mejores resultados posibles en física, biología, medicina e ingeniería, entre muchos otros. (Bonrostro et al., 1997) y obviamente, también, en economía y finanzas.

Ante los comportamientos no lineales de las variables económicas que pueden afectar a la bolsa, se han propuesto nuevos métodos con empleo de redes neuronales ya que estas permiten establecer relaciones lineales y no lineales entre los *outputs* e *inputs* de un sistema. (Villada et al., 2012). Además, estos algoritmos se basan en su gran capacidad de reconocimiento de patrones para realizar predicciones a corto tiempo.

También se debe tener en cuenta que hay muchos tipos distintos de redes neuronales artificiales, cada una con sus ventajas y sus desventajas. En concreto, la literatura remarca el buen comportamiento que tienen las redes neuronales recurrentes para la predicción de series temporales (Hochreiter et al., 1997; Nelson et al., 2017; Olah, 2015; Selvin et al., 2017; Sermpinis et al., 2019).

En este trabajo se ha querido corroborar la literatura realizando un estudio de distintos modelos recurrentes aplicados a la predicción del índice bursátil del Ibex-35. Para ello se han empleado las librerías Tensorflow y Keras, dos librerías de Python que permiten programar redes neuronales artificiales con relativa facilidad.

Se han estudiado tres arquitecturas (cambiando la profundidad de la red neuronal o el número de neuronas de una capa) para cada sistema de red propuesto: red neuronal recurrente simple, red neuronal recurrente del tipo LSTM y una capa convolucional de una dimensión entre dos capas LSTM, dando así 9 modelos básicos.

Además, en cada uno de los modelos propuestos se han cambiado los parámetros de manera sistemática para así lograr el estudio de hasta 864 modelos distintos. En cada uno de estos modelos se han empleado 8 bases de datos diferentes que, a excepción de los datos base, todos tienen un indicador técnico distinto para ver la mejoría que se puede obtener.

Tras el estudio se puede observar que a pesar de que las redes neuronales recurrentes simples funcionan mejor que métodos clásicos de predicción y que redes neuronales simples, si los comparamos con los modelos LSTM y los modelos con una capa convolucional entre dos capas LSTM, son mucho peores. En particular, los últimos funcionan particularmente bien a nivel general, llegando a la conclusión que estos modelos son los mejores para la predicción de este problema.

1.2 Motivación

El aumento de la potencia de computación ha provocado que se desarrollen muchos más métodos y sistemas para la predicción de bolsa y la compraventa de acciones, un mercado altamente volátil y con alto riesgo. La capacidad de predecir de manera efectiva la tendencia o, aún mejor, el precio de las acciones o, como en el caso del presente trabajo, el valor del índice puede dar al inversor una gran ventaja a la hora de tomar decisiones. Con un buen sistema puede sacar una rentabilidad mucho mayor a corto plazo.

Sin embargo, esta no es la única motivación para la realización de este trabajo. Ha permitido al autor mezclar temas vinculados con las dos carreras que ha cursado, uno de los objetivos primeros que se propuso, para así dar más valor al haber formado en dos ámbitos, a priori, tan distintos disciplinalmente hablando.

A nivel académico, ha permitido al autor adentrarse en el campo de la inteligencia artificial, más concretamente en el *deep learning*, una disciplina que está en auge en muchas áreas distintas por su gran capacidad de adaptación en resolución de problemas. No solo eso, también ha permitido entrar en el campo del *trading* y la bolsa, un mundo altamente estudiado por los grandes beneficios que puede aportar.

1.3 Objetivos

El presente trabajo ha tenido como objetivo principal la creación y comparación de métodos de predicción de un índice bursátil mediante el uso de inteligencia artificial, más concretamente con técnicas de *deep learning* y redes neuronales artificiales aplicadas en a la predección del Ibex-35. Para llegar a dicho objetivo se plantearon una serie de objetivos más específicos que se debían resolver para poder completar el propósito final. Los objetivos que específicos que se han completado han sido los siguientes:

- Estudio y conocimiento general de sistemas de inteligencia artificial y *machine learning.* Esto se creyó necesario para tener un buen fundamento de estos conceptos antes de adentrarse en el mundo de las redes neuronales. Además, sirvió para entender algunos de los nuevos métodos de predicción y análisis de bolsa como los que presenta Kaufman (2005) en los que se emplean distintos métodos de inteligencia artificial.

- Estudio específico y meticuloso de cómo funcionan las redes neuronales y el *deep learning*, sus componentes principales y los distintos tipos principales que existen y se emplean.

- Estudio y revisión de la literatura sobre la predicción de bolsa para obtener una mejor visión de cómo tratar distintas ideas propuestas y así saber los distintos métodos, modelos y conclusiones que la comunidad científica ha conseguido llegar en este tema.

- Conocimiento específico de cómo funcionan los índices bursátiles, en concreto del Ibex-35, e introducción al mundo del *trading* y la predicción bursátil junto al estudio de distintos métodos de análisis técnico.

- Aprender a programar en Python. Toda la parte técnica del trabajo se realiza en este lenguaje de programación del cual el autor de este trabajo no tenía conocimiento previo.

- Obtención de una base de datos fiables, limpios y con un suficiente número para poder entrenar las redes.

- Diseño y desarrollo de distintos tipos de arquitecturas de redes neuronales recurrentes para poderlos comparar con una red neuronal básica que funcionará como *baseline*. Con esto se pretende observar si las redes neuronales recurrentes pueden ser mejores predictores en series temporales, y más en concreto para la predicción del Ibex-35.

- Elegir uno o varios de los modelos que se proponen. Tras el análisis de los resultados se escogerá qué modelo o modelos son los más precisos para la predicción del Ibex-35

1.4 Orden documental

El presente trabajo se ha organizado en siete capítulos

En el capítulo 1 se ha realizado una introducción y resumen de lo que se va a encontrar en este trabajo, junto a la motivación y los objetivos marcados para el desarrollo de este.

En el capítulo 2 presenta el marco teórico. Toda la teoría que se ha desarrollado y estudiado para entender mejor los modelos propuestos y los resultados obtenidos. Primero se describe qué es el Ibex-35 junto al concepto del índice bursátil. En el siguiente apartado se expone el concepto de la predicción de bolsa junto algunos de los métodos de análisis técnico más empleados. A continuación, se realiza una breve explicación de qué son las redes neuronales. Finalmente, se realiza un recorrido por distintos artículos y documentos científicos en los que nos apoyaremos para reforzar conceptos, elegir las metodologías y realizar los modelos y las medidas.

En el capítulo 3 se encuentra la metodología seguida para la realización del trabajo. Se exponen todas las decisiones tomadas para la realización de los modelos junto a los materiales y recursos empleados.

En el capítulo 4 se muestran los resultados que se han alcanzado.

En el capítulo 5 se presenta el análisis y la discusión de resultados, recomendando un modelo en concreto para la predicción del índice del Ibex-35.

En el capítulo 6 llegamos a las conclusiones que se han llegado a partir del análisis de resultados.

Finalmente, en el capítulo 7 se han desarrollado posibles futuras líneas de trabajo.

2. Marco teórico

2.1 Ibex 35

En una bolsa de valores se ponen en contacto demandantes e inversores de capital para la compraventa de valores, sean acciones (partes iguales en las que se divide el capital social de una empresa, es decir, el valor de los bienes que tiene esta) u otros instrumentos financieros. Las bolsas permiten que compañías que buscan financiación puedan negociar e intercambiar con personas físicas u organizaciones ahorradoras que buscan sacar rendimiento de su capital.

La primera bolsa de valores nació en Bélgica en el año 1460 donde una familia de banqueros organizaba en su casa operaciones comerciales. La primera bolsa oficial, sin embargo, no apareció hasta 1602 en Ámsterdam. En los siguientes siglos empezó a consolidarse la institución de la bolsa de Valores, surgiendo la de Londres, Nueva York, Paris y Madrid.

En España la primera bolsa nació el 10 de septiembre de 1831 en Madrid. Fue seguida por Bilbao (1890), Barcelona (1915) y Valencia (1980), formando las cuatro bolsas oficiales del país. Mediante el SIBE (Sistema de Interconexión Bursátil Español), estas cuatro bolsas de valores están enlazadas. El SIBE es un software que recibe todas las órdenes de compraventa de las bolsas en tiempo real y son clasificadas y almacenadas según precio y fecha de emisión. Cuando una orden de contrapartida entra y coincide con precio y volumen, se efectúa la orden y se cambian los valores de titular. (BME, 2019).

En una bolsa de valores existe un índice bursátil. Este índice es un sistema de medida estadística para mostrar la evolución en el tiempo de los precios de las acciones cotizadas en ese mercado de valores o de sus rendimientos.

Los índices bursátiles dan mucha información: muestran cómo va el mercado (al alza o en declive), muestran la rentabilidad y el riesgo que tiene el mercado, sirven como punto de referencia para comparar el rendimiento de un gestor de activos, crear carteras imitando el comportamiento del índice, etc.

Dentro del Ibex hay distintos índices, a lo que se llama familia de índices:

- El IBEX 35, de valores de alta capitalización.
- El IBEX Medium Cap, de valores de mediana capitalización.
- El IBEX Small Caps, de valores de baja capitalización.
- El IBEX Top Dividendos, formado por las empresas que más retribuyen a sus accionistas.
- Otros. Por sectores (financiero, energía, construcción, ...), IBEX 35 inverso, IBEX 35 apalancado, IBEX MAB, …

Sin embargo, dado que en este trabajo nos centraremos tan solo en el Ibex 35, ya que el resto de los índices dependen de una manera u otra de éste.

El 14 de enero de 1992 se puso en marcha el índice del Ibex 35 con 3000 puntos. En ese momento el valor de todas las empresas que la componían era de unos 52.000 millones de

euros. En el 2017, superaba el valor de los 650.000 millones de euros (Domingo, 2017). El Ibex 35 es el principal índice bursátil español en el que se recogen las 35 empresas (aunque dependiendo la época ha abarcado más o menos número como en 2011 que se componía por 36 empresas (El PAIS, 2011)) con más liquidez que cotizan en el SIBE en las cuatro bolsas españolas, es decir, aquellas cuyas acciones pueden transformarse de manera más rápida en dinero. Un equipo de expertos se reúne al menos dos veces al año para estudiar y revisar la composición del índice.

Para el cálculo del índice se usa la siguiente fórmula:

$$IBEX\ 35\ (t)\ =\ \frac{IBEX\ 35\ (t-1)\ \sum_{i\,=\,1}^{35} Cap\ (t)}{\sum_{i\,=\,1}^{35} Cap\ (t)\ \pm\ J}$$

Ecuación 1. Fórmula para el cálculo del valor del índice. Fuente: BME (2019)

Dónde *t* es el momento del cálculo del índice, *i* es la compañía incluida en el índice, Cap_i es la capitalización de la compañía *i* incluida en el índice y, por tanto, Σ Cap_i es la suma de la capitalización de todas las compañías incluidas en el índice.

La capitalización es tal que $S_i \cdot P_i$ dónde S_i es el número de acciones computables de la compañía *i* para el cálculo del valor del índice y P_i es el precio de las acciones de la compañía *i* incluida en el índice en el momento t. Es decir, la capitalización de una empresa indica el patrimonio disponible para la compraventa activa en la bolsa.

Finalmente, *J* es una cantidad utilizada para ajustar el valor del índice y, por tanto, trata de asegurar que, ante los posibles cambios de valores de las empresas, asegurar la continuidad del índice. La función del componente *J* es, entonces, evitar la posible alteración del índice por operaciones financieras como ampliación/reducción de capital, dividendos, emisión de instrumentos financieros convertibles o canjeables, variaciones del valor nominal, etc.

El Ibex 35 se caracterizaba por los componentes que se detallan en la Tabla 1 en septiembre de 2019, donde se puede ver el código, sector y ponderación de cada empresa (la ponderación es la influencia o peso que tiene cada empresa sobre el Ibex 35).

Empresa	Código	Sector	Ponderación septiembre 2019 (%)
Acciona	ANA	Infraestructura	0,65
Acerinox	ACX	Metalúrgica	0,37
Grupo ACS	ACS	Infraestructura	2,37
Aena	AENA	Transporte	3,08
Amadeus IT Group	AMS	Turismo	6,13
ArcelorMittal	MTS	Metalúrgica	0,59
Banco Sabadell	SAB	Finanzas	1,03
Banco Santander	SAN	Finanzas	13,06

Bankia	BKIA	Finanzas	0,67
Bankinter	BKT	Finanzas	1,10
Banco Bilbao Vizcaya Argentaria	BBVA	Finanzas	6,83
CaixaBank	CABK	Finanzas	3,00
Cellnex Telecom	CLNX	Telecomunicaciones	2,26
CIE Automotive	CIE	Siderurgia, petroquímica	0,49
Enagás	ENG	Energía	1,04
ENCE	ENC	Energía y celulosa	0,18
Endesa	ELE	Energía	2,09
Ferrovial	FER	Transporte	4,12
Grifols	GRF	Salud	2,36
IAG	IAG	Transporte	2,25
Iberdrola	IBE	Energía	12,33
Inditex	ITX	Textil	10,66
Indra Sistemas	IDR	Consultoría	0,31
Inmobiliaria Colonial	COL	Inmuebles	0,92
Mapfre	MAP	Finanzas	0,96
MasMovil Ibercom	MAS	Telecomunicaciones	0,41
Mediaset España Comunicación	TL5	Medios de comunicación	0,34
Meliá Hotels International	MEL	Turismo	0,28
Merlin Properties	MRL	Inmuebles	1,25
Naturgy	NTGY	Energía	3,00
Red Eléctrica Corporación	REE	Energía	2,07
Repsol	REP	Energía	4,75
Siemens Gamesa Renewable Energy	SGRE	Energía	1,15
Telefónica	TEF	Telecomunicaciones	7,5

Viscofan	VIS	Alimentos	0,42

Tabla 1. Empresas que forman el Ibex 35.
Fuente: Elaboración propia a partir de BME (2019) actualizados de septiembre de 2019.

Como se puede ver en la Tabla 1, las ponderaciones de las empresas que forma el IBEX son muy variadas. Solo con los once valores con mayor peso ya se representa casi un 75% del índice, y tan solo se necesitan cinco empresas (Banco Santander, Iberdrola, Inditex, Telefónica y BBVA) para superar el 50%. Esto nos indica que estas empresas son las que más influencia tienen sobre el índice.

Este hecho puede dar lugar a falsas interpretaciones. Que el IBEX 35 vaya al alza no tiene por qué dar por supuesto que la economía y la bolsa va evolucionando de manera positiva, porque puede ser que tan solo estén evolucionando así las grandes empresas.

No obstante, no suele ser así, ya que el Ibex suele ser un buen indicativo de cómo funciona la economía del país. En la Figura 1 vemos el histórico de cierre del Ibex 35 desde mitades del año 2000 hasta casi mitades del año 2020. Como se puede ver cuando había una crisis económica, el Ibex-35 reaccionaba cayendo, creando los mínimos representados por las flechas azules, frente al crecimiento económico siendo el máximo de la burbuja inmobiliaria en 2007 representado por la flecha verde.

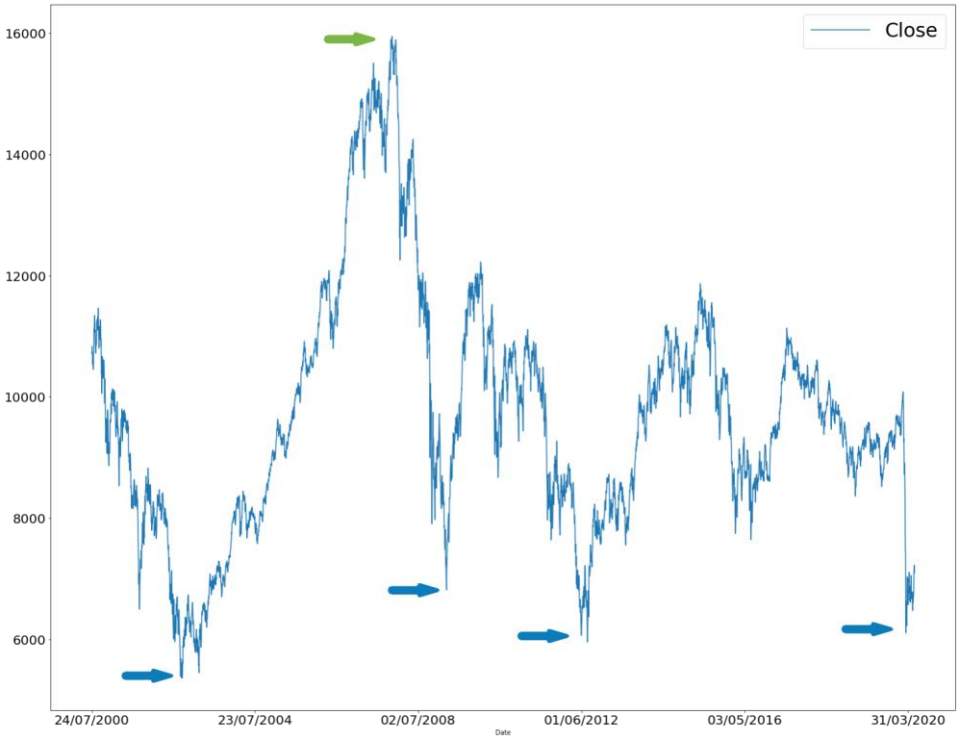


Figura 1. Histórico del Ibex-35 desde el 24 de julio del 2000 hasta el 31 de mayo del 2020.

La primera flecha azul representa la crisis causada por la crisis de Argentina y la depreciación de la moneda brasileña en 2002. La flecha verde es el pico de la burbuja inmobiliaria, viniendo de una época de crecimiento económico. En su contra, la siguiente flecha azul, es el mínimo causado por la crisis del 2008 al explotar la burbuja inmobiliaria y por la recesión mundial a causa de la caída de la economía estadounidense. El siguiente mínimo

fue por la crisis de Bankia arrastrando a todo el Ibex con ella. Finalmente, la última flecha representa la crisis económica causada por la pandemia mundial del Covid-19.

La sesión diaria del índice español abre a las 9 A.M. y cierra a las 17:30 P.M., horario nacional. En los índices se suelen tener varios valores indicadores de cómo ha ido la sesión de cada día. Estos son:

- *Close* o cierre. Este es el valor que tenía el índice cuando se cierra la sesión del día.
- *Open* o apertura. Este es el valor que tiene el índice al inicio de sesión. Para llegar al precio de apertura se realiza un proceso de subasta en el que se reciben órdenes de compraventa a precio limitado. Es decir, el precio de *Open* no es el mismo que el de *Close* del día anterior. En este valor se pueden encontrar las influencias de las otras economías o de sucesos que han ocurrido mientras el índice estaba cerrado.
- Máximo. Es el valor máximo al que se ha llegado en la sesión.
- Mínimo. Es el valor mínimo al que se ha llegado durante la sesión.
- *Adj Close* o cierre ajustado. Es el valor ajustado de cierre, donde se modifica el valor de cierre para reflejar la modificación de valor de esta, por ejemplo, un reparto de dividendos o una ampliación de capital.

En el siguiente apartado estudiaremos la motivación que existe detrás de la predicción de bolsa y algunos métodos clásicos para llevarla a cabo.

2.2 Predicción de bolsa

2.2.1 Introducción a la predicción de bolsa.

Predecir es formular una expectativa de lo que ocurrirá en el futuro. La dificultad radica en poder anticipar la incertidumbre del futuro a partir del conocimiento actual. Es decir, se debe intentar encontrar una relación entre el presente y los eventos que acontecerán en el futuro.

En la predicción del mercado bursátil se intenta calcular el valor futuro de las acciones de una compañía, otro instrumento financiero negociado en una bolsa o sus índices. Desde hace años los analistas han pretendido encontrar una solución factible al problema de predecir el valor de las acciones.

La hipótesis del mercado eficiente, la cual es una teoría de inversión, defiende que los precios reflejan toda la información. Por tanto, ni el análisis fundamental (el estudio de la información financiera) ni el análisis técnico (el estudio de los precios de las acciones en el pasado en un intento de predecir los precios futuros mediante el uso de indicadores técnicos) pueden producir rendimientos excesivos ajustados al riesgo de manera consistente, ya que los precios de mercado sólo deben reaccionar a la nueva información. Sugiere, entonces, que las acciones en las bolsas de valores siempre se negocian a su valor justo, lo que ofrece a los inversores la oportunidad de comprar acciones infravaloradas o vender acciones a precios inflados. Por lo tanto, los inversores no pueden vencer al mercado con la ayuda de la cronología de este y la selección experta de acciones (Malkiel, 2003).

> Los predictores buscan constantemente patrones predecibles y afectan a los precios cuando intentan explotar las oportunidades comerciales. Por lo tanto, es poco probable que las pautas de previsión estables persistan durante largos períodos de tiempo y se autodestruyan cuando sean descubiertas por un gran número de inversores. Esto da lugar a que las series temporales de rendimientos financieros no sean estacionarias y complica tanto las pruebas formales de la eficiencia del mercado como la búsqueda de enfoques de previsión satisfactorios (Timmermann & Granger, 2004, p. 15).

El párrafo de Timmermann & Granger (2004), defensores de la teoría del mercado eficiente, nos explica como predecir bolsa es inviable si esta es descubierta por muchos predictores, simplemente dejarán de funcionar, por ello, al final no existe un método válido para predecir. No obstante, esta hipótesis, a pesar de su fama, es rechazada por numerosos investigadores ((Basu, 1977),(Chan et al., 1997)). Hay estudios, que se verán más adelante, que demuestran que sí se pueden predecir precios, no de manera exacta pero sí lo suficiente como para reducir notablemente los riesgos de invertir en algo tan volátil como la bolsa.

Sin embargo, debido a la cantidad de variables que se tiene en un mercado es tremendamente difícil poder encontrar un sistema viable. 'Hay una clara diferencia entre predecir la tendencia y encontrar la tendencia actual. Predecir el precio futuro es mucho más deseable pero mucho más complejo.' (Kaufman, 2005)

Durante años, estudiosos e inversores han tratado de predecir el comportamiento de las bolsas. La posesión de información y un buen análisis técnico puede ponerte muy por delante de otros *traders*. Por ello, la predicción de bolsa ha sido siempre algo muy buscado, ya que proporciona una ventaja frente a otros inversores al dar la posibilidad de generar unos beneficios mucho mayores.

Muchas de las técnicas para la predicción de bolsa dan por hecho que los datos del pasado se pueden emplear para poder predecir el movimiento de los precios del futuro. En su mayor parte, así es. Estos métodos, desde un punto de vista práctico, son mucho más flexibles y dan mejores resultados que los métodos tradicionales de regresión lineal. No obstante, siempre hay un *lag* o un retardo, es decir, un retraso en la identificación de una tendencia.

Durante años se han empleado dos tipos distintos de análisis para la predicción cada uno con sus ventajas y sus carencias. El análisis fundamental intenta calcular el valor real de un título mediante información de la empresa y de la economía a nivel macroeconómico, y así determinar si en un momento el precio de este está por encima o por debajo del valor que le correspondería. El análisis técnico, en su contraposición, se basa en identificar tendencias mediante indicadores estadísticos y matemáticos, los llamados indicadores técnicos.

En este trabajo vamos a estudiar algunos de los métodos de análisis técnico más usados ya que luego serán utilizados en el modelo propuesto.

2.2.2 Métodos de predicción con análisis técnico

2.2.2.1 Medias móviles

Uno de los indicadores técnicos clásicos más conocidos y usados es el de las Medias Móviles o *Moving Averages* (MA). Este método nos da una idea general de la tendencia.

Se usan para eliminar el ruido del mercado y confirmar la tendencia de precios. La media móvil proporciona, de manera aproximada, el valor del activo.

Hay varios tipos de medias móviles, pero las más usadas son la Media Móvil Simple o *Simple Moving Average* (SMA) y la Media Móvil Exponencial o *Exponential Moving Average* (EMA).

Media Móvil Simple (SMA)

La SMA es el cálculo de la media aritmética de un valor o precio sobre un número de períodos.

$$\mathrm{SMA}_t = \frac{A_1 + A_2 + A_3 + \cdots + A_n}{n}$$

Ecuación 2. Fórmula de la Media Móvil Simple o SMA

Dónde n es el número de períodos y A_n es la precio o valor en el período n.

Así, cuanto mayor sea la n los datos antiguos tendrán mayor peso. La decisión de n, entonces, puede cambiar de manera drástica la predicción. Cuanto más baja sea n, mayor será la capacidad de contestar ante cambios u oscilaciones significativos del valor de los datos, en cambio también estarán más influidos por efectos aleatorios o datos incorrectos. Por el contrario, si la n es alta responderá de manera más lenta ante fluctuaciones significativas de períodos anteriores, aunque permita filtrar mejor los datos aleatorios y sea más insensible al ruido

En la Figura 2 hay un ejemplo de SMA elaborado por el autor del TFG. Se han usado tres meses de datos del precio de Bitcoin con distinto número de períodos n. Como se puede ver, cuando se usa n =5 sigue de mejor manera la tendencia del precio de los Bitcoins. En n = 15, por el contrario, la predicción está menos afectada por el ruido que puede tener el precio. Eso es porque el *lag* depende del número de períodos. Además, se puede ver también que cambia el *lag* de un valor a otro. Se ha representado en el ejemplo del máximo más claro cada retardo con una flecha de su respectivo color. Cuanto mayor sea n, mayor será el retardo.

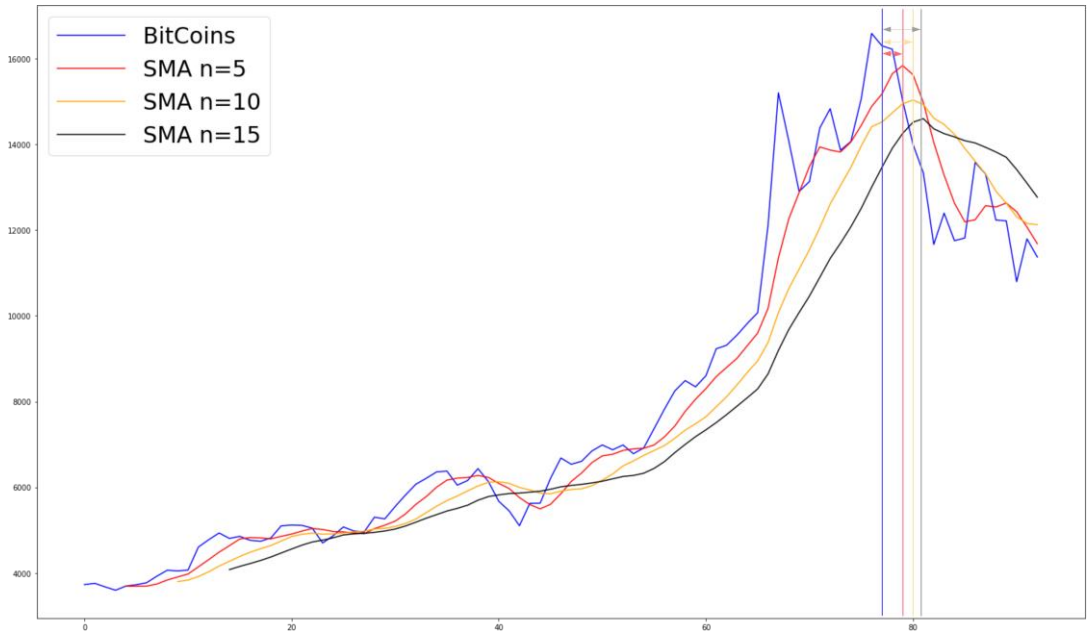


Figura 2. Ejemplo de SMA de distintos n (5,10,15) desde el 01/10/2017 hasta el 01/01/2018 sobre el precio de Bitcoins junto con los retardos respecto a este.
Fuente: elaboración propia

Media Móvil Exponencial (EMA).

La Media Móvil Exponencial o EMA se basa en asignar un peso a cada valor, teniendo más importancia los valores más recientes y menos los más lejanos. Es decir, a diferencia de SMA, no le influencian de igual manera los datos antiguos, ya que éstos tendrán un menor peso a medida que se vayan usando los datos más recientes.

$$EMA_t = A_t \cdot \alpha + EMA_{t-1} \cdot (1 - \alpha)$$

Ecuación 3. Fórmula para el cálculo de la Media Móvil Exponencial.

Dónde A_t es el valor o precio en el periodo t , EMA_{t-1}es el cálculo del EMA anterior (el primer EMA_{t-1} será A_1) y α es un coeficiente que representa el número de descenso del peso (entre 0 y 1). Cuanto mayor sea el coeficiente menos se tiene en cuenta los datos menos recientes. Para α se suele usar $\frac{2}{n+1}$, siendo n el número de periodos.

En la Figura 3 tenemos un ejemplo de EMA, con la misma base de datos de Bitcoin que teníamos para calcular las SMA. Como se puede ver al aumentar n y, por tanto, reducir α se tienen más en cuenta los datos más lejanos.

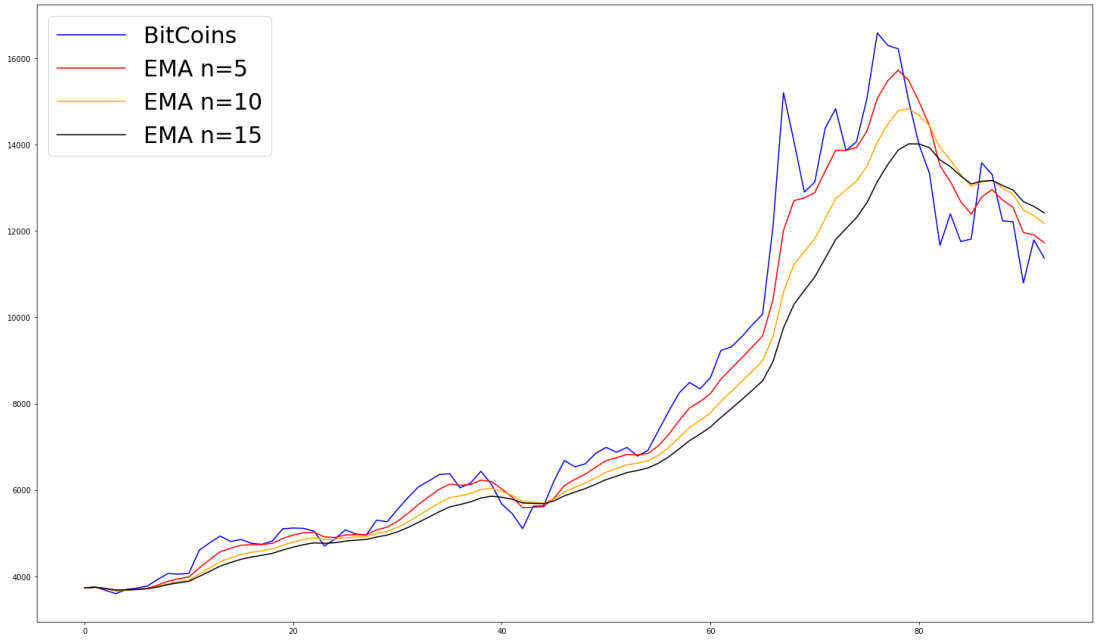


Figura 3. Ejemplo de EMA de distintos n (5,10,15) desde el 01/10/2017 hasta el 01/01/2018 sobre el precio de Bitcoins.
Fuente: elaboración propia

Comparación entre EMA y SMA

En la Figura 4 tenemos una comparación entre EMA y SMA para n =5. Vemos que son muy parecidas, pero se podría decir que la EMA es ligeramente mejor, ya que el retraso parece ligeramente inferior.

En la Figura 5 tenemos lo mismo que en la anterior, pero con n =15

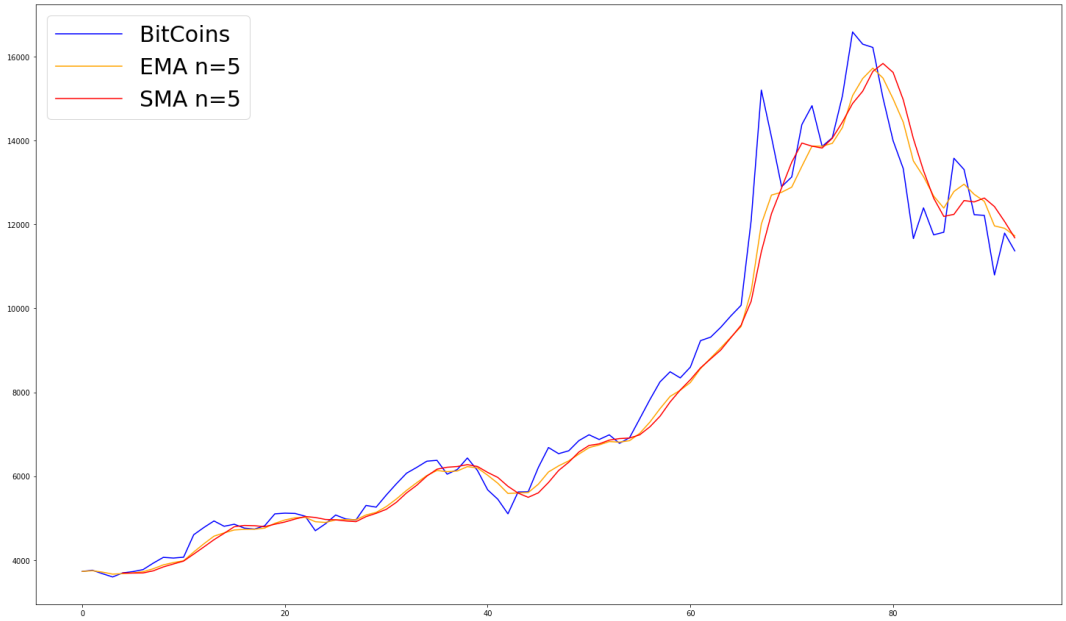


Figura 4. Comparación entre SMA y EMA de n=5.
Fuente: Elaboración propia.

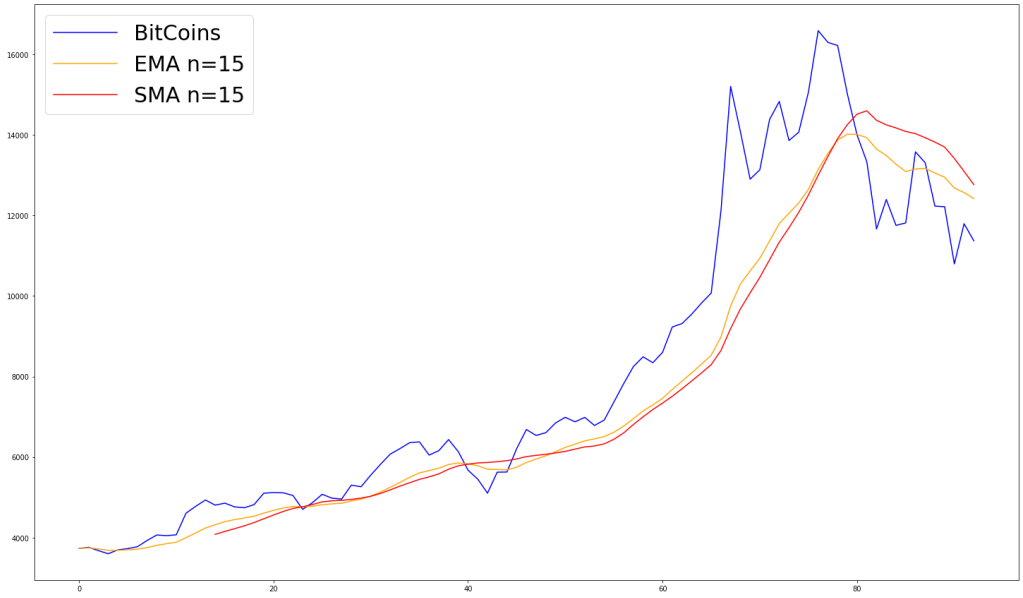


Figura 5. Comparación entre SMA y EMA de n=15.
Fuente: Elaboración propia.

Como podemos ver en la Tabla 2 la raíz del error cuadrático medio de EMA es menor que el de SMA indicando que la EMA es un mejor predictor. Más adelante veremos que es el RMSE en profundidad, pero básicamente es un indicador de error entre el valor predicho y el valor real.

n	RMSE de EMA	RMSE de SMA
5	628.599792820659 [1]	784.2411961556462
15	1338.9961028996975	1691.5554367201948

Tabla 2. Error cuadrático medio de EMA y SMA para n=5 y n =15.
Fuente elaboración propia

2.2.2.2 *Commodity Channel Index* (CCI)

Índice de canales de materias primas (*Commodity Channel Index*, CCI), es un indicador técnico que es usado para identificar tendencias y avisar de condiciones extremas. Este indicador mide el precio en un momento relativo al precio medio de un periodo. El CCI se calcula de la siguiente manera:

$$\mathrm{CCI}(n) = \frac{PT_t - SMA\ PT(n)}{0.0015 * \sigma(n)}$$

Ecuación 4. Cálculo de CCI de n periodos.

Siendo PT_t el precio típico, es decir, la media aritmética entre el valor máximo, el mínimo y el valor de cierre, $SMA\ PT(n)$ es la media móvil de los precios típicos de los últimos n periodos, 0.0015 es una constante que se elige para que entre el 70%-80% de los datos se sitúen entre -100 y 100 y $\sigma(n)$ la desviación típica de los últimos n períodos.

Cuando el CCI se mueve por encima de +100, una nueva y fuerte tendencia alcista comienza. Cuando el CCI se mueve por debajo de -100, una nueva y fuerte tendencia descendente está comenzando.

2.2.2.3 *Relative Strength Index* (RSI)

Índice de fuerza relativa (*Relative Strength Index*, RSI) es un indicador técnico que permite medir con qué fuerza sube o baja el precio de un valor en relación con sus precios habituales para un periodo de tiempo determinado. El RSI se muestra como un oscilador (un gráfico lineal que se mueve entre dos extremos) y puede tener una lectura de 0 a 100.

Este indicador cuenta con dos ecuaciones:

$$\mathrm{RS} = \frac{EMA\ de\ n\ períodos\ alcistas}{EMA\ de\ n\ periodos\ bajistas}$$

Ecuación 5. Fuerza relativa (Relative Strength).

Con esta ecuación calculamos la fuerza relativa inicial (*RS*) que es la media móvil exponencial de cierres con el valor superior al del día anterior de n periodos entre la media móvil exponencial de cierres con el valor inferior al del día anterior de *n* periodos. Finalmente, para el cálculo de *RSI* se emplea la siguiente fórmula:

$$\mathrm{RSI} = 100 - \frac{100}{1 + RS}$$

Ecuación 6. Índice de fuerza relativa.

[1] Todos los resultados serán dados con notación anglosajona, es decir, punto como separador decimal.

2.2.2.4 *Moving Average Convergence Divergence (*MACD)

Media Móvil de Convergencia Divergencia (Moving Average Convergence Divergence, MACD) es un indicador técnico diseñado para revelar cambios de fuerza, dirección momento y duración de una tendencia.

Para el cálculo del MACD se emplea una media móvil exponencial con un período corto de cálculo y otra media móvil exponencial con un período de tiempo medio. Por tanto, para el cálculo del MACD se emplea la siguiente fórmula:

$$\mathrm{MACD} = \mathrm{EMA}(n_0) - \mathrm{EMA}\,(n_1)$$

Ecuación 7. Media Móvil de Convergencia Divergencia.

Normalmente, para la media corta ($EMA(n_0)$)se emplean 12 períodos y 26 períodos para la media de tiempo medio ($EMA\,(n_1)$). Cuanto más corto es el período de cálculo, más sensible es la media móvil a la variación del precio.

2.2.2.5 Williams R%

Williams R% o R% es un indicador técnico que muestra dónde está el último precio de cierre en relación con los precios más altos y bajos de un período de tiempo determinado. Su propósito es decir si un mercado de valores o de productos básicos está operando cerca del máximo o del mínimo, o en algún punto intermedio, de su rango de negociación reciente. El cálculo de su fórmula es el siguiente:

$$Williams\ \%R = \frac{Máximo\ más\ alto_{Ndías} - Cierre}{Máximo\ más\ alto_{Ndías} - Mínimo\ más\ bajo_{Ndías}}$$

Ecuación 8. Williams R%

Normalmente se eligen los 14 últimos períodos para el cálculo de este índice.

Otro tipo de indicadores técnicos que se están usando actualmente son los basados en redes neuronales artificiales (Yao et al., 1999). En el apartado siguiente veremos que son estos modelos y cómo funcionan.

2.3 Redes Neuronales Artificiales

Inteligencia artificial, *Machine Learning* y *Deep Learning* son conceptos que, aunque están fuertemente unidos, no son lo mismo. Hay bastante confusión entre estos términos. De hecho, *Deep Learning* es un subcampo dentro del *Machine Learning* que es, a su vez, un subcampo de la inteligencia artificial (ver Figura 6). *Deep Learning* o, en castellano, aprendizaje profundo, es la metodología central que se empleará en el presente trabajo y que se detallará más adelante. En este apartado, se detallarán los fundamentos del *Deep Learning* para profundizar más en las Redes Neuronales Artificiales. Para ello vamos a delimitar el alcance de estos tres conceptos básicos.

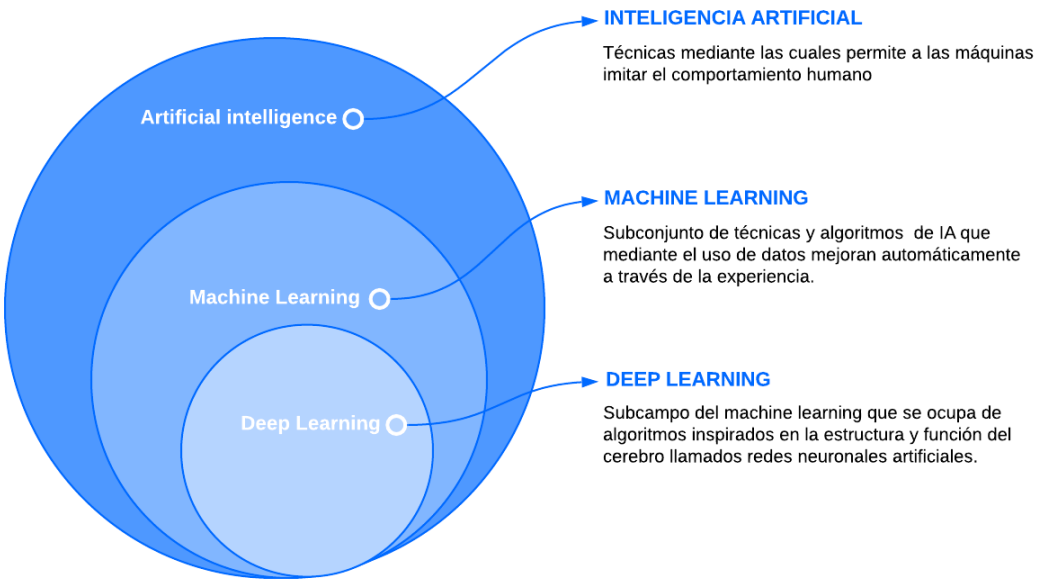


Figura 6. Comparativa entre IA, ML y Deep Learning.
Fuente: Elaboración propia a partir de González (2019)

2.3.1 Evolución conceptual: de la Inteligencia Artificial al Deep Learning

La Inteligencia Artificial (IA) es todo el conjunto de técnicas y métodos las cuales permite que una máquina imite el comportamiento humano. Esta tecnología ha avanzado a pasos agigantados desde mitad del siglo pasado. La evolución de la tecnología con procesadores mucho más potentes junto a la invención de nuevos algoritmos capaces de analizar millones de datos en minutos son los que han llevado a la Inteligencia Artificial a ser tan valorada en diferentes áreas.

Los primeros trabajos que se pueden considerar de IA datan de mitad del siglo XX. Alan Turing, el considerado padre de la informática y de la IA, saca en 1950 un artículo llamado *"Computing Machinery and Intelligence"* a partir del cual se empiezan a tomar más en serio el concepto de que una máquina pueda imitar el comportamiento de la mente humana. Sin embargo, no fue hasta 1956 que se acuñó el término de Inteligencia Artificial en la conferencia de Dartmouth (Buchanan, 2005). Desde entonces la IA ha evolucionado gracias a nuevos métodos y algoritmos que han dado paso al desarrollo de técnicas inteligentes capaces de

mejorar todo tipo de áreas: en agricultura prediciendo en qué momento específico una cosecha estará lista para recolectar (INTEL, 2019), monitorización de la tierra y de los campos (Kumba & Sennaar, 2019); en sanidad en detección de enfermedades, asistencia en diseño de tratamientos, creación de fármacos, etc. (The Medical Futurist, 2018); en defensa (Schweikhard et al., 2001) y aviación no controlada (Baomar & Bentley, 2017), y un largo etcétera. Y, por supuesto, en finanzas y *trading* automático.

La IA engloba a Machine Learning junto a otras técnicas como el razonamiento aproximado, la lógica difusa o métodos bayesianos entre otros.

El *Machine Learning* pretende que los ordenadores realicen un aprendizaje a partir de los datos que se le introducen mediante el uso de algoritmos. La diferencia con la inteligencia artificial es que esta no tiene porque "aprender", sino que puede estar previamente programada a través de una serie de reglas para que actúe de una manera "inteligente".

En ML hay muchos tipos de algoritmos distintos, como *clustering*, modelos lineales o redes neuronales artificiales y *Deep Learning* por poner algunos ejemplos.

El *Deep Learning* o aprendizaje profundo es un subcampo dentro de las redes neuronales artificiales. Se habla de DL cuando hay varias capas de profundidad en el modelo propuesto, es decir, cuánto más compleja sea la red neuronal más "profundo" es el aprendizaje. A continuación, vamos a estudiar qué son las redes neuronales artificiales.

2.3.2 Redes Neuronales Artificiales

Las Redes Neuronales Artificiales (RNA) se han convertido en estos últimos años en la familia de algoritmos de *Machine Learning* más populares, aunque desde mediados del siglo pasado ya existían. Sin embargo, no ha sido hasta hace poco con la mejora de los algoritmos y de la tecnología que no han crecido el uso de estos métodos.

Hay muchos campos en los que se están usando Redes Neuronales artificiales:

- Reconocimiento de voz y texto y traducción automática a otros idiomas
- Conducción autónoma.
- Análisis genético.
- Detección de enfermedades vía imagen.
- Reconocimiento facial y ciberseguridad.
- Minería de datos.
- Prevención de fraudes.
- Filtrado de email y spam.
- Prevención de accidentes
- Predicción bursátil.

Estos son solo unos cuantos ejemplos de los miles de usos que se le dan a las RNA.

Las RNA son algoritmos computacionales de *Machine Learning*, que se basan en las Redes Neuronales Biológicas y que pretenden modelar comportamientos inteligentes. Tratan, entonces, de replicar simplificadamente la organización física del cerebro, para reproducir sus características computacionales (Buchanan, 2005).

Como en las Redes Neuronales Biológicas, las RNA están formadas por muchas neuronas unidas entre sí mediante los inputs (dendritas) y los outputs (axones) formando las redes neuronales. En el interior de la neurona se procesa la información (soma). Además, cada

neurona tiene un peso correspondiente que indica qué cantidad de información pasa de cada neurona a la siguiente (sinapsis). Aunque estén inspiradas en el cerebro humano y compartan nombre, ni funcionan igual ni aprenden de la misma manera.

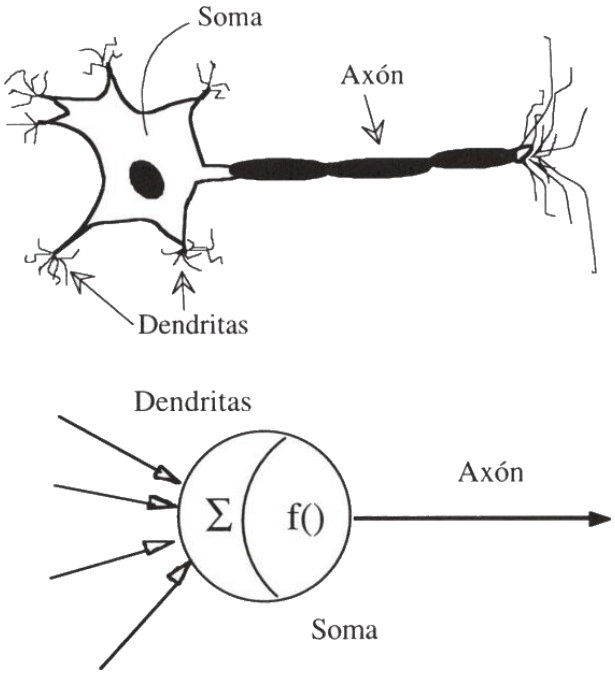


Figura 7. Comparativa entre Neurona Biológica y Neurona Artificial.
Fuente: Akgün & Demir (2018)

Sin embargo, se usan ya que son excepcionalmente buenas en encontrar patrones y modelos en una base de datos (G. P. Zhang, 2012). Con las RNA se pretende hacer que un programa aprenda a realizar una tarea sin haber sido previamente programada para ello. Esto se hace mediante el uso de ejemplos. La RNA a base de estos ejemplos debe encontrar unos patrones que le indiquen una solución válida para cuando reciba entradas nuevas.

2.3.3 Tipos

Se pueden clasificar RNA según:

Mecanismos de aprendizaje

Según el tipo de aprendizaje se puede hablar de aprendizaje supervisado, no supervisado o por refuerzo.

- Aprendizaje supervisado. El entrenamiento se realiza mediante el control de un agente externo que determina la respuesta a la que debería llegar la RNA a partir del *input*.
- Aprendizaje no supervisado. El entorno no da ninguna señal de si el *output* respecto a un *input* específico es correcto.
- Por refuerzo. Da una señal de +1 si el ajuste ha tenido éxito y de -1 si el ajuste ha sido un fracaso.

Topologías básicas

Dependiendo de la organización o disposición de las neuronas en la red se puede hablar de distintos tipos de RNA.

- Monocapa o perceptrón simple. Hay una sola capa la que constituye la red. Se usan de manera general cuando los *inputs* están de manera incompleta o distorsionados con la intención de regenerar información.
- Multicapa. Cuando se emplean varias capas de profundidad

Topologías más avanzadas

- *Multi-Layer Perceptron* o Perceptrón Multicapa (MLP). Son el tipo de Redes Neuronales más clásicas. Están compuestas por dos o más capas. Se emplean tanto para problemas de regresión como de clasificación.
- *Recurrent Neural Network* o Red neuronal Recurrente (RNN). En este tipo de redes neurales se ha implementado una especie de memoria para mejorar la capacidad de predecir series temporales. Dentro de este tipo también se sitúan las redes neuronales del tipo LSTM, una mejora del tipo de redes neuronales recurrentes, ya que emplean un tipo de memoria que permite olvidar.
- *Convolutional Neural Network* o Red Neuronal Convolucional (CNN). Estas redes neuronales están diseñadas para trabajar en tareas de visión por computador en las que la imagen es el tipo de entrada.
- Otras. Hay muchos tipos de redes neuronales y de tipos de neuronas. Pero las anteriores son las más empleadas.

2.3.4 Datos para las redes neuronales.

Para entrenar a las redes, los datos que con los que se trabajan se suelen separar en tres grupos: datos de entrenamiento, datos de validación y datos de test.

Los datos de entrenamiento es el conjunto de datos de ejemplos que se usan para el aprendizaje de la red, es decir, para adaptar los parámetros de configuración de una red neuronal.

El conjunto de datos de test en su contraparte, son un conjunto de datos independientes del aprendizaje de la red. En estos datos se pone a prueba el resultado del entrenamiento de la red y evaluar el rendimiento de esta.

Un conjunto de datos de validación es una muestra de datos retenidos durante el entrenamiento de su modelo que se utiliza para dar una estimación de la habilidad del modelo mientras se afinan los hiperparámetros del mismo. El conjunto de datos de validación es diferente del conjunto de datos de test que también se retiene del entrenamiento del modelo, pero se utiliza en cambio para dar una estimación imparcial de la habilidad del modelo afinado final al comparar modelos finales.(Ligeza, 1995)(Kuhn & Johnson, 2013)

Con esto se pretende evitar tanto el *underfitting* (subajuste) como el *overfitting* (sobreajuste). Ambos son altamente perjudiciales para la red neuronal (ver Figura 8).

El *underfitting* se produce cuando la red neuronal no puede captar la tendencia subyacente de los datos. Se produce cuando el modelo o el algoritmo no se ajusta lo suficientemente bien a los datos. No se suelen encontrar tanto estos casos como el *overfitting.*

El *overfitting* ocurre cuando la red neuronal captura el ruido de los datos. Se produce cuando el modelo o el algoritmo se ajusta demasiado bien a los datos. Utilizando validación puede evitar este problema.

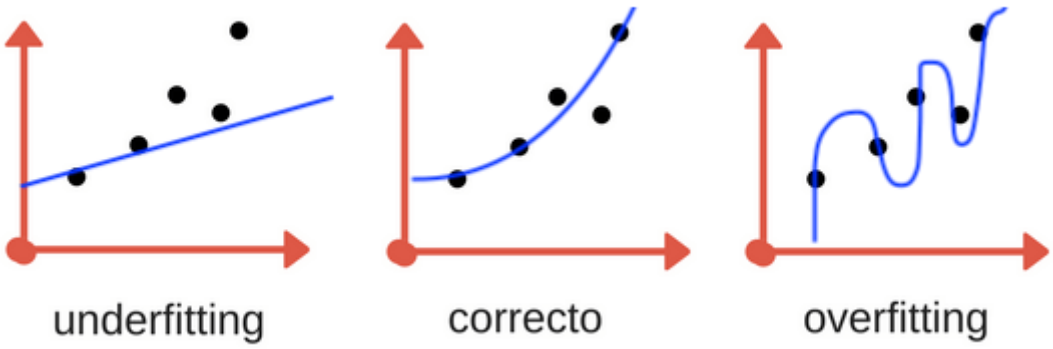


Figura 8. Underfiting, aprendizaje correcto y overfitting.
Fuente: Bagnato (2018).

Para saber cómo de bien está aprendiendo la red neuronal se debe tener un medidor o alguna referencia. Ahí entra la función de coste.

2.3.5 Función de coste o 'loss function'

En el cálculo del error del modelo se emplean distintas funciones de coste. Estas tratan de determinar el error entre el valor estimado y el valor real. A partir de dichos errores es posible optimizar los parámetros de la red neuronal.

Para el cálculo del error se emplean distintos métodos dependiendo del tipo de problema.La mayoría de problemas para resolver con RNA son de dos tipos: tareas de clasificación y tareas de regresión.

En los problemas de clasificación las RNA producen una distribución de probabilidades a través de varias categorías. Se emplea mucho para clasificación de imágenes. En este tipo de problemas se suele usar el *sparse categorical crossentropy* (entropía cruzada categórica). Es fácil de interpretar pero dificil para diferenciación y convergencia.

$$\int(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij} log\left(p_{ij}\right)$$

Ecuación 9. Sparse categorical crossentropy.

En los problemas de regresión se produce una salida con un solo valor. Este es el problema que trataremos de resolver para la realización del trabajo. En estos casos se suele usar el *root mean squared error (RMSE)* (raíz del error cuadrático medio) o *mean squared error (MSE)*. La primera función de coste mide la raíz cuadrada error promedio al cuadrado entre la diferencia de la predicción y de los datos reales y la segunda lo mismo pero sin la raíz cuadrada.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

Ecuación 10. Root mean squared error.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

Ecuación 11. Mean squared error.

Siendo n el número de predicciones, $\hat{y}_i$el valor predicho e $y_i$el valor real.

También hay otras funciones de coste que se pueden emplear para comprobar la efectividad de la Red Neuronal: como el MAE, MAPE, FPE, R^2, etc. El MAE y el MAPE son los indicadores que más se emplean después de RMSE y MSE para problemas de regresión.

MAE, que significa *Mean Absolute Error* (Error Absoluto Medio), tiene la siguiente fórmula:

$$\text{MAE} = \frac{\sum_{i=1}^{n}|\hat{y}_i - y_i|^2}{n}$$

Ecuación 12. Mean Absolute Error

MAPE, *Mean Absolute Percentage error* (Error medio de porcentaje absoluto), tiene la siguiente fórmula:

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

Ecuación 13. Mean Absolute Percentage Error

2.3.6 Neurona o Perceptrón.

Similar a una neurona biológica, la Neurona Artificial (NA) tiene conexiones de entrada con las que recibe "estímulos" externos (valores de entrada).

En el perceptrón se realiza un cálculo interno y genera un valor de salida. Este cálculo no es más que una suma ponderada de los valores de entrada con el peso que se le asigna a cada entrada. Estos pesos son los parámetros del modelo y son los que se deben ajustar para que opere la RNA.

Además, se le añade otro parámetro que se llama *bias* (sesgo), en el que la variable siempre está asignada a 1. El sesgo sirve para que el modelo se adecúe lo máximo posible a los datos. Esencialmente lo que hace cada neurona es un problema de regresión lineal (más adelante veremos que con las funciones de activación no es exactamente así pero sí que es una buena aproximación).

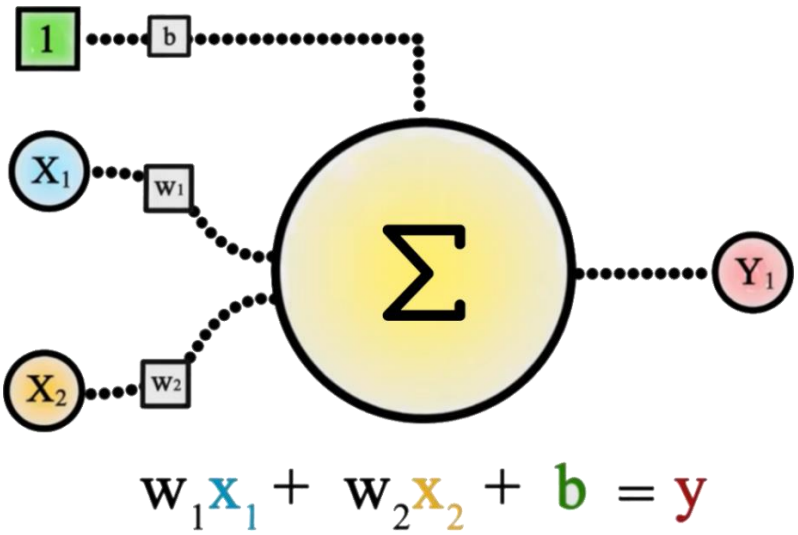


Figura 9. Perceptrón Simple.
Fuente: Elaboración propia a partir de Santana (2017).

En la Figura 9 se puede ver un ejemplo con los componentes principales de un perceptrón simple. Interiormente en una NA para calcular la salida (Y_1) se realiza un sumatorio de las entradas o "estímulos" exteriores (X_n) por su peso (W_n) más el sesgo (b_1):

$$Y_i = \sum_n W_{in} * X_n + b_i$$

Ecuación 14. Sumatorio que se produce dentro de la neurona (sin función de activación).

2.3.6.1 Función de activación.

Como antes hemos dicho en el apartado de la neurona, lo que hace cada neurona es básicamente un problema de regresión. Sin embargo, la suma de muchas regresiones lineales sigue resultando una regresión lineal. Por lo que no podrían sumarse varios perceptrones para crear un esquema más complejo, ya que llegaríamos a un solo perceptrón de igual manera. Para cambiar este resultado y que pueda ampliarse la profundidad de una RNA se añade al perceptrón una función de activación.

Es un componente que también forma parte de la neurona, el cual básicamente es una función que "deforma" la suma ponderada que hace la neurona con la intención de quitarle la linealidad de la regresión. Así se le añaden distorsiones no lineales en cada capa, dando así la capacidad de concatenar varias neuronas.

En la práctica se usan muchos distintos tipos de funciones de activación. La única condición es que esta sea derivable. A continuación, se detallan algunas de las funciones de activación más usadas:

1. **Función escalón unitario** (o escalón binario). Es una función discontinua que vale 0 para cualquier argumento negativo y 1 para 0 o cualquier argumento positivo. Su rango es {0,1}. En las Figura 10 y Figura 11 vemos la representación gráfica de la función escalón.

$$\mathrm{f}(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geq 0 \end{cases}$$

Ecuación 15. Función escalón unitario

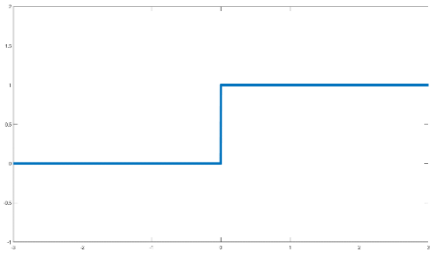


Figura 10. Gráfico 2D de la función escalón unitario. Fuente: Elaboración propia.

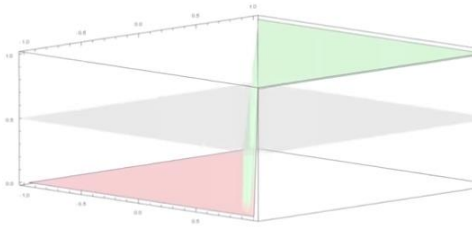


Figura 11. Gráfico 3D de la función escalón unitario. Fuente: Elaboración propia.

2. **Función Sigmoide**. Es una función real que tiene una forma parecida a una "S". Su rango es (0,1). En las Figura 12 y Figura 13 vemos la representación gráfica de la función sigmoide.

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

Ecuación 16. Función Sigmoide.

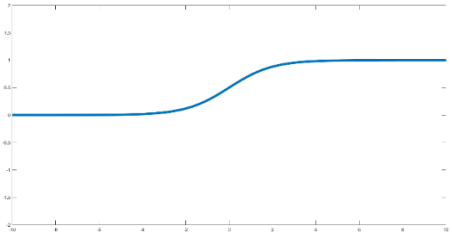


Figura 12. Gráfico 2D de la función Sigmoide. Fuente: Elaboración propia.

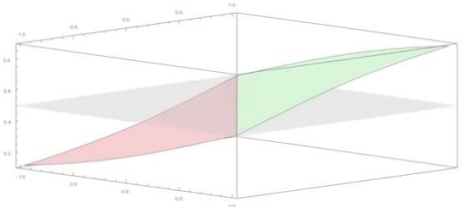


Figura 13. Gráfico 3D de la función Sigmoide. Fuente: Elaboración propia.

3. **Función Tangente Hiperbólica.** Se define como el cociente entre el seno y el coseno hiperbólicos. Su rango es (-1,1). En las Figura 14 y Figura 15 vemos la representación gráfica de la función tangente hiperbólica.

$$\mathrm{f}(x) = \tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$$

Ecuación 17. Función Tangente Hiperbólica.

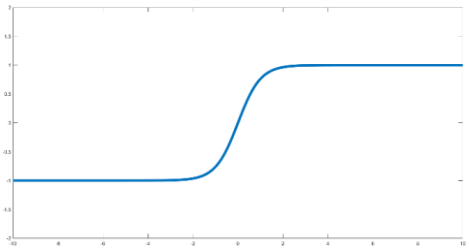


Figura 14. Gráfico 2D de la función tangente hiperbólica. Fuente: Elaboración propia.

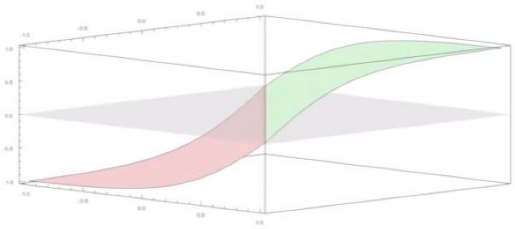


Figura 15. Gráfico 3D de la función tangente hiperbólica. Fuente: Elaboración propia.

4. **Función ReLU (*Rectified Linear Units*).** Esta función se define como la parte positive de su argumento. Su rango es $[0, \infty)$. En las Figura 16 y Figura 17 vemos la representación gráfica de la función ReLU.

$$f(x) = \begin{cases} 0 & for\ x < 0 \\ x & for\ x \geq 0 \end{cases}$$

Ecuación 18. Función ReLU

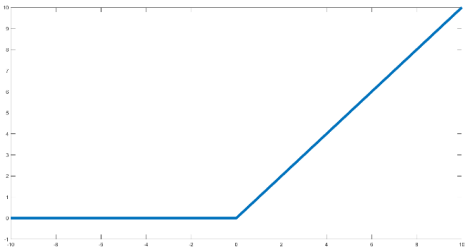


Figura 16. Gráfico 2D de la función ReLU. Fuente: Elaboración propia.

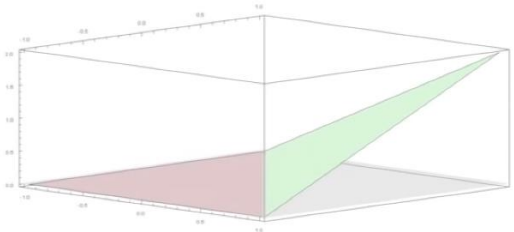


Figura 17. Gráfico 3D de la función ReLU Fuente: Elaboración propia.

Dentro de la ReLU hay muchas variantes interesantes como la ReLU paramétrica (dentro de la cual está la Leaky ReLU o a la Random ReLU) o la ELU (Exponential Linear Unit). La primera con rango (∞, ∞) y la segunda con rango (α, ∞) ,

La **ReLU paramétrica** se define mediante la siguiente función:

$$f(\alpha, x) \begin{cases} \alpha x & for\ x < 0 \\ x & for\ x \geq 0 \end{cases}$$

Ecuación 19. Función ReLU Paramétrica.

En la **Leaky ReLU** el parámetro α toma un valor de 0.01 y en **Random ReLU** un valor aleatorio como su propio nombre indica.

La **ELU** (exponential linear unit) viene definida por la siguiente función:

$$f(\alpha, x) \begin{cases} \alpha(e^x - 1)\ for \leq 0 \\ x \qquad\quad for\ x > 0 \end{cases}$$

Ecuación 20. Función ELU

2.3.7 Redes Neuronales Artificiales del tipo Multi Layer Perceptron (MLP).

Multi Layer Perceptron (MLP) o perceptrón multicapa es un tipo de red neuronal consistente en juntar varios perceptrones de tal manera que estén organizados en capas. Son dos o más neuronas organizadas de tal manera que dichas neuronas reciben la misma información de entrada que la salida de la capa anterior. Los cálculos que realizan pasarán a la siguiente capa. El conjunto de capas forma la red neuronal.

La primera capa de todas se denomina capa de entrada. Estará compuesta por tantas neuronas como variables se desea utilizar para el problema. Las capas intermedias (que pueden ser tantas como se quiera) se denominan capas ocultas o *Hidden Layers* en inglés. La profundidad en la cantidad de capas es lo que lleva al *Deep Learning.* Por último, la capa de salida donde se tendrán tantas neuronas de salida como número de categorías en las que se quiera en el caso de que sea un problema de clasificación (si se tratara de una clasificación binaria solo haría falta una neurona) y una salida en un problema de regresión.

En la Figura 18 vemos la disposición de una Red Neuronal del tipo MLP.

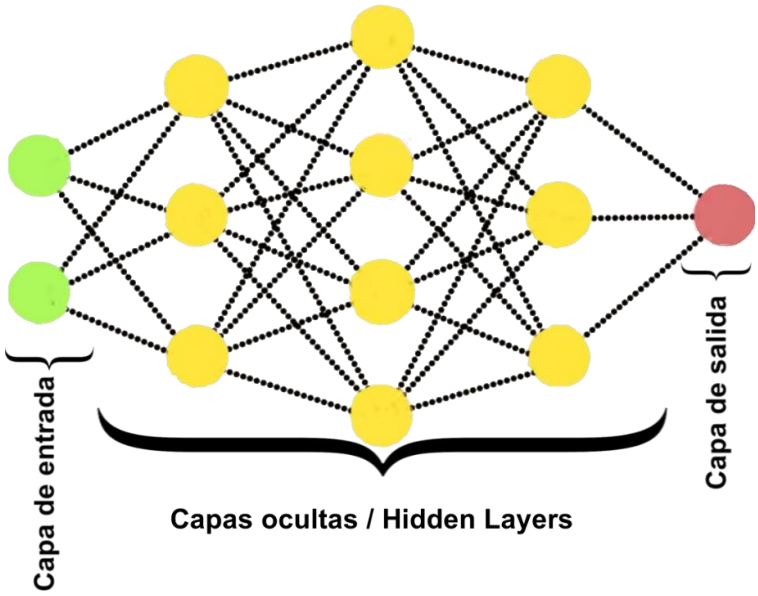


Figura 18. Disposición de las capas en una Red Neuronal MLP.
Fuente: Elaboración propia a partir de Santana (2017*)*

2.3.8 Proceso de aprendizaje. Algoritmo del descenso del gradiente.

Para saber el grado de adaptación de nuestro modelo al problema se debe estudiar una función de coste, es decir, una función que para cada una de las combinanciones de parámetros nos diga qué cantidad de error tenemos.

El objetivo es, entonces, minimizar lo máximo posible dicha función. Aquí entra el descenso del gradiente. El descenso del gradiente es un algoritmo que nos permite encontrar el mínimo en cualquier función de *n* dimensiones.

Cuando se empieza a entrenar el modelo, se va a tener un error de valor x. Como lo que se pretende es minimizar dicho error, se tiene que encontrar una pendiente en el punto de la función de coste en la que se esta calculando. Para calcular dicha pendiente, dado que posiblemente en la RNA se esta trabajando con bastantes parámetros por lo que la función de coste será multidimensional, se hace mediante el calculo del gradiente. Consiste en las derivadas parciales de cada uno de los parámetros de la función. El gradiente de una función f evaluado en un punto x del dominio de f indica la dirección en la cual el campo varía más rapidamente. Es una generalización de la derivada en un espacio multidimensional.

$$\nabla f(r) = \left(\frac{\partial f(r)}{\partial x_1}, \frac{\partial f(r)}{\partial x_2}, \dots, \frac{\partial f(r)}{\partial x_n}\right)$$

Ecuación 21. Gradiente de una función.

En la Figura 19 vemos un ejemplo del descenso del gradiente. Tenemos una función de coste con la forma mostrada donde el eje de las *Z* es la cantidad de error. Como habíamos dicho antes buscamos minimizar dicho error por lo que tenemos que buscar cómo descender desde un punto aleatorio (punto rojo) al que llamaremos punto *a*. Para ello empleamos las derivadas parciales de ese punto respecto al error representadas con la flecha morada y la flecha naranja.

$$\frac{\partial error(a)}{\partial \theta_1}$$

Ecuación 22. Derivada parcial del error respecto al parámetro 1 en el punto a (flecha morada en Figura 19)

$$\frac{\partial error(a)}{\partial \theta_2}$$

Ecuación 23. Derivada parcial del error respecto al parámetro 2 en el punto a (flecha naranja Figura 19)

La flecha negra representa el gradiente en el punto *a*, que es el vector de las dos derivadas parciales anteriormente mencionadas. La flecha blanca es la representación del menos gradiente en el punto *a*.

$$\nabla f(a) = \begin{pmatrix} \frac{\partial error(a)}{\partial \theta_1} \\ \frac{\partial error(a)}{\partial \theta_2} \end{pmatrix}$$

Ecuación 24. Gradiente de la función de coste en el punto a (flecha negra en la Figura 19)

$$-\nabla f(a)$$

Ecuación 25. Menos gradiente de la función de coste en el punto a (flecha blanca en la Figura 19)

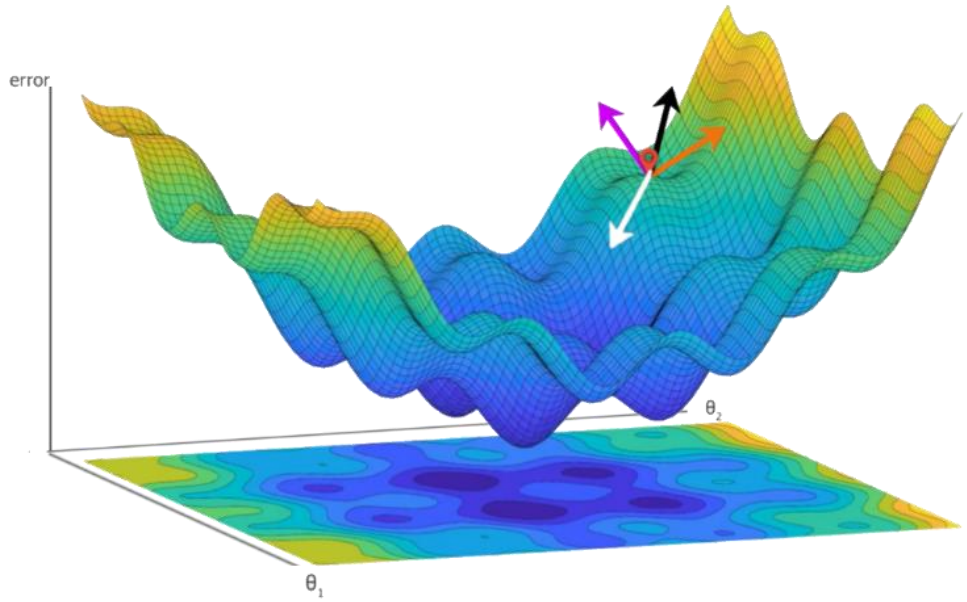


Figura 19. Ejemplo descenso del gradiente.
Fuente: Elaboración propia.

Debemos movernos, entonces, en la dirección de la flecha blanca. Una vez lleguemos a otro punto de la función repetiremos el proceso y así de manera iterativa hasta que nos encontremos en una zona donde movernos ya no represente una variación notable del coste. Esto será cuando lleguemos en una zona donde la pendiente sea próxima a 0, por lo que nos encontraremos en un mínimo local. El algoritmo que se empleará es entonces el siguiente:

$$Repetir\ hasta\ la\ convergencia\ \{$$
$$\theta_i = \theta_{i-1} - \eta * \nabla f\ (i-1)\ \}$$

Ecuación 26. Algoritmo del descenso del gradiente.

El parámetro η es la ratio de aprendizaje. Este parámetro se multiplica por el gradiente para saber el cuanto de lejos estará el siguiente punto. El problema está en que si fijas este parámetro demasiado pequeño el algoritmo realizará numerosas iteraciones para encontrar el punto mínimo. En el ejemplo de la Figura 20 vemos que sí que llega a un mínimo, pero requiere de un gran número de iteraciones. También podría darse situaciones en las que el punto se encuentre encerrado en un mínimo más ineficiente y sea incapaz de salir. Por el contrario, si pones una ratio demasiado grande puede que ni siquiera encuentre un punto mínimo por tener unas oscilaciones demasiado grandes como vemos en la Figura 21.

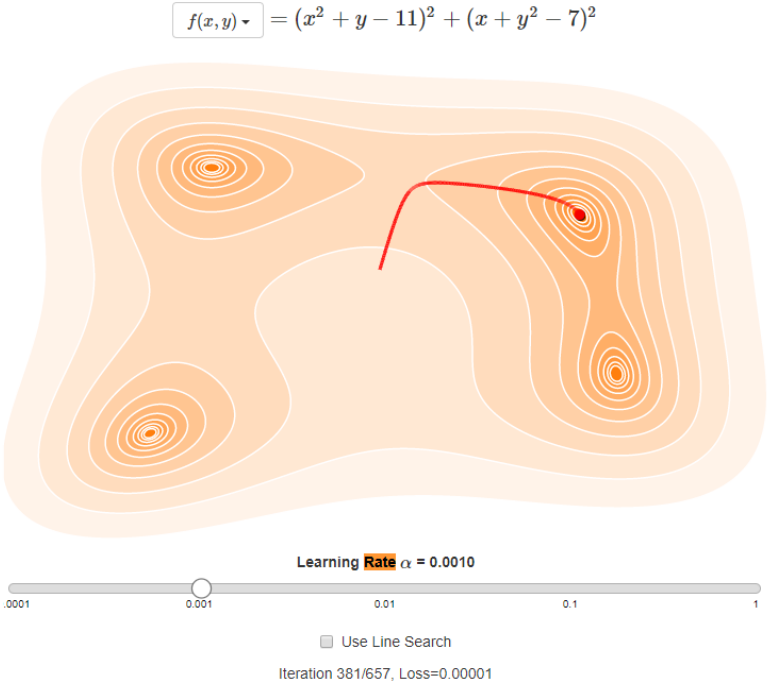


Figura 20. Ejemplo de descenso de gradiente con una ratio de aprendizaje demasiado bajo. Fuente: Elaboración propia a partir de Frederickson (2016)

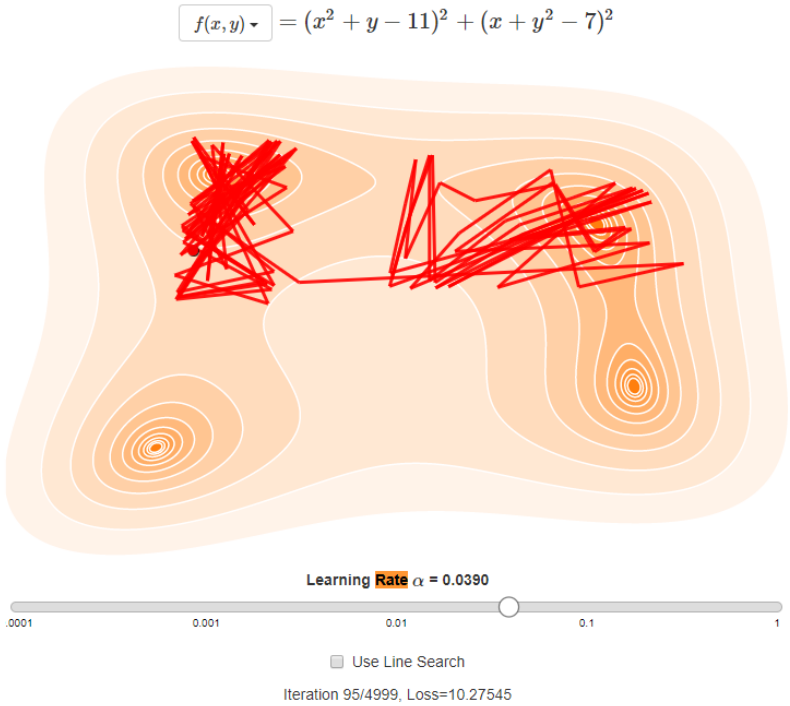


Figura 21. Ejemplo de descenso de gradiente con una ratio de aprendizaje demasiado alto. Fuente: Elaboración propia a partir de Frederickson (2016)

El problema de este algoritmo es que cuando estamos trabajando con RNA tenemos muchas variables y, por tanto, muchos caminos distintos. El cambio de un solo peso de una rama de la RNA puede influir a toda la red. Ahí es donde entra en juego el algoritmo de *backpropagation*.

2.3.9 Algoritmo de backpropagation.

El algoritmo de *Backpropagation,* o de retropropagación en español, es un algortimo para entrenar redes neuronales artificiales de manera eficiente. La idea de este algoritmo es saber qué cantidad de error es la que introduce cada neurona de manera iterativa y recursiva para así recalcular los pesos de cada neurona.

Hasta que se creó este algoritmo en 1970 se usaba un método donde se estudiaba cada posible camino de manera aleatoria e iba cambiando ligeramente el peso de la neurona. Esto llevaba a que fuera tremendamente ineficiente por la cantidad de iteraciones que debían llevarse a cabo. Sería imposible entrenar la mayoría de redes de *Deep Learning* existentes a día de hoy sin el algoritmo de *backpropagation*.

El algoritmo de *backpropagation* consta de dos fases(Bernacki & Włodarczyk, 2005):

1. Propagación hacia adelante donde las entradas pasan através de la RNA dando como salida una predicción.
2. El paso hacia atrás, donde los pesos de la Red Neuronal se actualizan para acercarse más a la realidad.

2.3.9.1 Propagación hacia adelante

Como hemos visto en apartados anteriores, las RNA vienen conformadas por distintas capas y componentes. Ahora estudiaremos la propagación hacia delante de las RNA.

Para entrenar una red neuronal requerimos de un conjunto de datos de entrenamiento. Esto consiste en una cantidad de parametros $[x_1, x_2, \ldots, x_n]$ signados con un valor deseado z. Con este conjunto de datos se pasa por la RNA, que calcula la salida, y repite este proceso de manera iterativa. A continuación, veremos un ejemplo simple de cómo se propaga por la red.

La Figura 22 es una RNA bastante simple compuesta por dos *hidden layers* donde a_n es la salida de la neurona n , f_n la señal de salida del elemento no lineal de la neurona (función de activación) $f_n(z) = a_n$, la $z = x_1 w_{m1} + x_2 w_{m2} + \cdots x_n w_{mn}$, w_{mn} es el peso de cada neurona e y es la salida de todo el sistema.

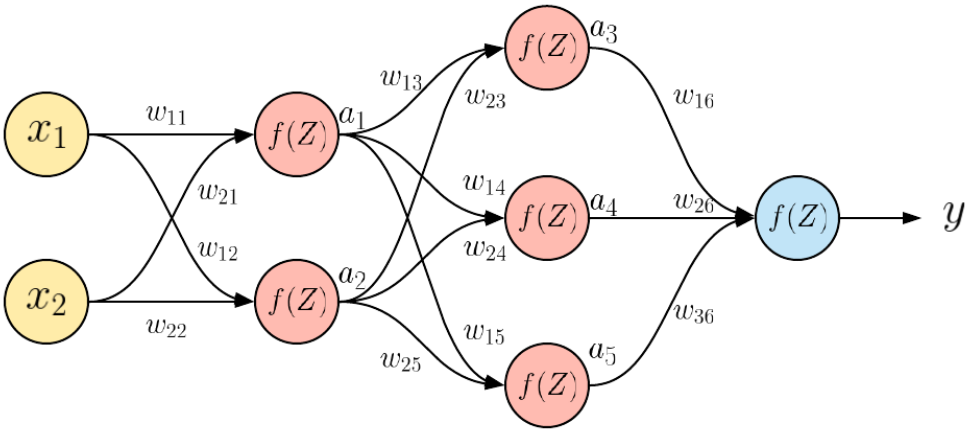


Figura 22. Ejemplo de propagación hacia adelante en una Red Neuronal artificial.
Fuente: Elaboración propia.

La salida de la primera capa oculta es tal que:

$$a_1 = f_1(w_{11}x_1 + w_{21}x_2) = f_1(s)$$
$$a_2 = f_2(w_{12}x_1 + w_{22}x_2) = f_2(s)$$

Ecuación 27. Salida de la primera y segunda neurona respectivamente de la primera capa oculta de la RNA del ejemplo de la Figura 22.

La salida de la segunda capa oculta viene dada por:

$$a_3 = f_3(w_{13}a_1 + w_{23}a_2)$$
$$a_4 = f_4(w_{14}a_1 + w_{24}a_2)$$
$$a_5 = f_5(w_{15}a_1 + w_{25}a_2)$$

Ecuación 28. Salida de la primera, segunda y tercera neurona respectivamente de la segunda capa oculta de la RNA del ejemplo de la Figura 22.

Por último, la capa de salida:

$$\hat{y} = f_6(w_{16}a_3 + w_{26}a_4 + w_{36}a_5)$$

Ecuación 29. Salida de la RNA del ejemplo de la Figura 22.

Una vez se llega al final se debe comparar el resultado de la salida con el valor deseado y para ver cuánto nos hemos alejado de solución y en caso de estar muy lejos deberemos cambiar el peso de cada neurona. Ahí es donde entra la propagación hacia atrás.

2.3.9.2 Propagación hacia atrás.

Vamos a estudiar el algoritmo de *backpropagation* en el ejemplo de la Figura 22 donde habiamos visto la propagación hacia adelante.

La idea es ver cómo varía el coste o la diferencia del error ante un cambio de cada parametro w.

Como se ha visto en el apartado de los componenetes de la neurona, cada neurona tiene también el término de sesgo o *bias*. Por lo que también se tendrá que tener en cuenta cómo varía el coste ante un cambio del bias.

Se verá un ejemplo en una red neuronal de L capas para luego estudiar el algoritmo de *backpropagation* en el ejemplo de la Figura 22 donde habíamos visto la propagación hacia adelante.

La función de coste se concatena con la función de activación, en la cual se ha pasado la suma ponderada de los pesos por la salida de la anterior neurona más el *bias*. Por lo que tenemos una composición de funciones. Al hacer la derivada parcial debe aplicarse la regla de la cadena.

$$Z^L = W^L a^{L-1} + b^L$$

Ecuación 30. Vector suma

$$\mathrm{C}\left(a^L(Z^L)\right)$$

Ecuación 31. Composición de funciones que forman la última capa.

$$\frac{\partial C}{\partial w^L} = \frac{\partial C}{\partial a^L}\frac{\partial a^L}{\partial z^L}\frac{\partial z^L}{\partial w^L}$$

Ecuación 32. Derivada parcial respecto al peso en la última capa (capa L)

$$\frac{\partial C}{\partial b^L} = \frac{\partial C}{\partial a^L}\frac{\partial a^L}{\partial z^L}\frac{\partial z^L}{\partial b^L}$$

Ecuación 33.Derivada parcial respecto al bias en la última capa (capa L).

Como se ve en la Ecuación 32 y en la Ecuación 33, aplicando la regla de la cadena se queda tres derivadas parciales. La primera de las derivadas parciales $\left(\frac{\partial C}{\partial a^L}\right)$ nos indica cómo varía el coste variando la activación de las neuronas en la última capa. La siguiente $\left(\frac{\partial a^L}{\partial z^L}\right)$ en qué grado se modifica el *output* de la neurona cuando variamos la suma ponderada de la neurona. Finalmente, la última $\left(\frac{\partial z^L}{\partial w^L}, \frac{\partial z^L}{\partial b^L}\right)$ es cuánto varía la suma ponderada respecto a una variación de los parametros, pesos o *bias* respectivamente.

Para simplificar, podemos juntar las dos primeras derivadas parciales:

$$\frac{\partial C}{\partial z^L} = \frac{\partial C}{\partial a^L}\frac{\partial a^L}{\partial z^L} = \delta^L$$

Ecuación 34.Error imputado a la neurona.

La Ecuación 34 representa en qué grado se modifica el coste cuando se produce un cambio en el valor de la neurona. Por ello, lo que nos da esta función es la responsabilidad de cada neurona o el error imputado a cada neurona. Además, la derivada de $\frac{\partial z^L}{\partial w^L} = a_i^{L-1}$, es decir, la activación de la capa previa y la derivada de $\frac{\partial z^L}{\partial b^L} = 1$.

Si sustituimos nos quedan las siguientes expresiones:

$$\frac{\partial C}{\partial w^L} = \delta^L\, a_i^{L-1}$$

Ecuación 35. Derivada parcial respecto al peso en la última capa (capa L).

$$\frac{\partial C}{\partial b^L} = \delta^L$$

Ecuación 36.Derivada parcial respecto al bias en la última capa (capa L).

Para las capas anteriores (capas L-1, L-2,etc.) solo necesitaremos una expresión más, ya que aplicando la regla de la cadena se nos queda una expresión muy parecida.

$$C\left(a^L(W^L a^{L-1}(W^L a^{L-2} + b^{L-1}) + b^L)\right)$$

Ecuación 37.Composición de funciones que forman la penúltima capa.

$$\frac{\partial C}{\partial w^{L-1}} = \frac{\partial C}{\partial a^L}\frac{\partial a^L}{\partial z^L}\frac{\partial z^L}{\partial a^{L-1}}\frac{\partial a^{L-1}}{\partial z^{L-1}}\frac{\partial z^{L-1}}{\partial w^{L-1}}$$

Ecuación 38. Derivada parcial respecto al peso en la penúltima capa (capa L-1).

$$\frac{\partial C}{\partial b^{L-1}} = \frac{\partial C}{\partial a^L}\frac{\partial a^L}{\partial z^L}\frac{\partial z^L}{\partial a^{L-1}}\frac{\partial a^{L-1}}{\partial z^{L-1}}\frac{\partial z^{L-1}}{\partial b^{L-1}}$$

Ecuación 39.Derivada parcial respecto al bias en la penúltima capa (capa L-1).

Si lo estudiamos vemos que $\frac{\partial z^L}{\partial a^{L-1}} = W^L$, es decir, es la matriz de parámetros W que conecta una capa con la anterior. También $\frac{\partial z^{L-1}}{\partial w^{L-1}} = a^{L-2}$ como hemos visto antes es la activación de la capa previa y $\frac{\partial z^{L-1}}{\partial b^{L-1}} = 1$. La derivada parcial $\frac{\partial a^{L-1}}{\partial z^{L-1}}$ es la derivada de la función de activación.

Si sustituimos se nos quedan las siguientes expresiones:

$$\frac{\partial C}{\partial w^{L-1}} = \delta^L W^L \frac{\partial a^{L-1}}{\partial z^{L-1}} a^{L-2} = \delta^{L-1} a^{L-2}$$

Ecuación 40. Derivada parcial respecto al peso en la penúltima capa (capa L-1).

$$\frac{\partial C}{\partial b^{L-1}} = \delta^L W^L \frac{\partial a^{L-1}}{\partial z^{L-1}} = \delta^{L-1}$$

Ecuación 41.Derivada parcial respecto al bias en la penúltima capa (capa L-1).

Esto se repite hasta llegar a la última capa. Entonces habremos calculado el gradiente que necesitábamos para aplicar el algoritmo del descenso del gradiente. Una vez se finaliza mediante dicho algoritmo se actualizan los pesos y se repite el proceso de manera iterativa hasta conseguir el resultado deseado. Por lo que nos queda:

$$Repetir\ hasta\ la\ convergencia\ \{$$
$$w_i = w_{i-1} - \alpha \nabla C\ (w_{i-1})\ \}$$

2.3.10 Redes Neuronales Artificiales Recurrentes (RNN)

Las Redes Neuronales del tipo MLP tienen un problema en cuanto al análisis de secuencias (por ejemplo, series temporales) se refiere. Estas no tienen memoria, es decir, en cada iteración aprenden sin tener ningún tipo de contexto previo. Por poner un ejemplo, queremos implementar un sistema de predicción de texto y tenemos la siguiente frase: 'Siempre me han gustado más los números que las letras. Por eso la asignatura que más me gusta es matemáticas'. Si hubiésemos intentado predecir a que asignatura se refiere pues es bastante obvio que diría matemáticas o física porque tenemos el contexto de la frase anterior. Pero sin ese contexto podría haberse dicho inglés, lengua, música, etc. Ese es el inconveniente de las MLP, al no tener memoria no puede tener en cuenta entornos anteriores. Este problema se puede solucionar mediante Redes Neuronales Recurrentes (RNN).

Las Redes Neuronales Artificiales Recurrentes son un tipo de RNA con bucles que permiten que se mantenga información, de tal forma que la red tenga una especie de memoria. Son redes con bucles que permiten que la información persista. Esto se hace permitiendo que outputs previos se puedan usar como inputs.

En la Figura 23 vemos una red neuronal recurrente con el bucle desenrollado siendo $y_t$el *output*, x_t el *input* y la A un segmento de red neuronal. Con esta estructura tipo cadena se puede observar lo buenas que pueden ser para secuencias y series de tiempo. Se puede pensar que se le añade muchísima dificultad a la RN, pero nada más lejos de la realidad. Se tiene que ver como una sucesión de numerosas réplicas de la misma red, cada una de las cuales se van pasando información a su sucesora.

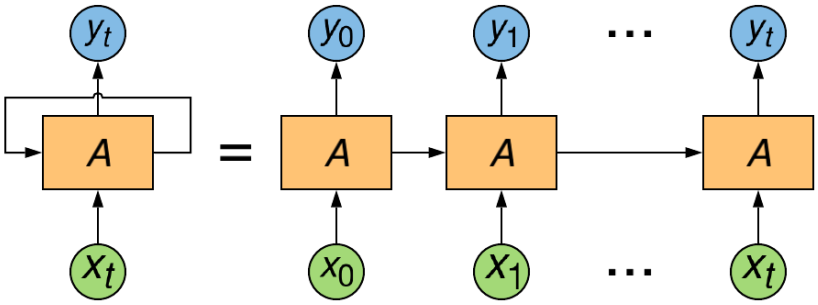


Figura 23. RNN con el bucle 'desenrollado'.
Fuente: Elaboración propia a partir de Olah (2015)

Este tipo de redes, basadas a partir del trabajo de Rumelhart et al. en 1986, junto con la invención de Hochreiter & Schmidhuber en 1997 de las redes neuronales recurrentes tipo LSTM (*Long-short Term Memory*), han supuesto una revolución en la resolución de distintos tipos de problemas: reconocimiento de voz, modelación del lenguaje, predicción y creación de textos, subtitulación automática, traducción, etc.

Hay muchos tipos de redes neuronales recurrentes: redes bidireccionales, redes de tipo Hopfield, redes de estados de eco, recursivas, etc. Sin embargo, tan solo se estudiarán dos tipos de redes neuronales: las recurrentes estándar y las LSTM, ya que son las que se van a emplear en la parte metodológica de este trabajo. Las primeras han sido elegidas ya que son el tipo más clásico de redes neuronales y para entender el resto de las redes recurrentes deben entenderse éstas. Las LSTM se han elegido por la aplicabilidad que tienen estas en predicción de basadas en series de tiempo.

2.3.10.1 Recurrente estándar.

La forma más clásica de las redes neuronales recurrentes tiene la forma de la Figura 24, siendo $\mathrm{x_t}$ el *input*, A un segmento de red neuronal e y_t el *output*. Esta forma viene dada por la repetición de módulos repetitivos de la red neuronal. En las RNN estándar este módulo será muy simple. En este caso una sola capa de *tanh*, pero podría ser cualquier otra función de activación. La salida del módulo anterior se concatena junto a la entrada del periodo anterior para sacar la predicción del paso de tiempo actual.

Con esto podemos ayudar a la red neuronal a que tenga esa memoria que se estaba buscando. Sin embargo, esta red neuronal tiene un problema: la memoria es a corto plazo. Cuando tienes dependencias a largo plazo este tipo de red neuronal no es tan buena.

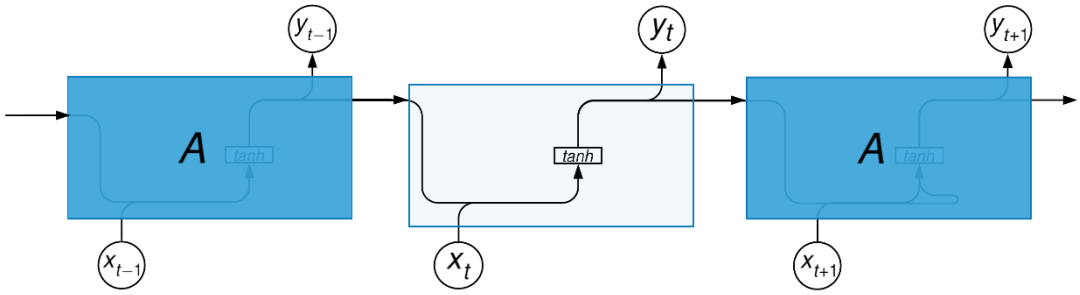


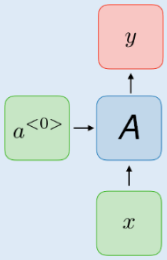
Figura 24. Esquema de una Red Neuronal Recurrente.
Fuente: Elaboración propia a partir de Olah (2015)

Por supuesto, este tipo de redes neuronales presentan ciertos tipos de ventajas y de inconvenientes frente a las redes neuronales clásicas. Estas particularidades se muestran en la Tabla 3.

Ventajas	Inconvenientes
• Se tiene en cuenta información histórica. • El tamaño del modelo no aumenta con el tamaño de la entrada • Los pesos se comparten a través del tiempo • Se pueden procesar entradas de cualquier longitud	• La computación pude ser más lenta • Tiene dificultad para acceder a la información a largo plazo. • No se puede considerar ningún aporte futuro para el estado actual.

Tabla 3. Ventajas y desventajas de las redes neuronales recurrentes frente a las redes neuronales clásicas.
Fuente: Elaboración propia a partir de Amidi & Amidi (2018).

Dentro de las RNN se pueden clasificar según las aplicaciones o, más bien, su estructura, como se puede ver en la Tabla 4 .

Tipo de RNN	Estructura	Ejemplo de Aplicación
Uno a uno $T_x = T_y = 1$		Red neuronal tradicional

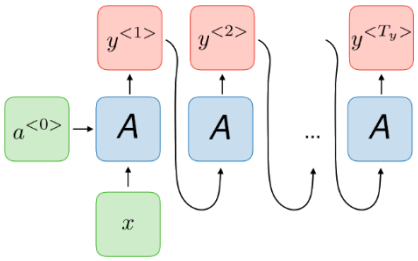
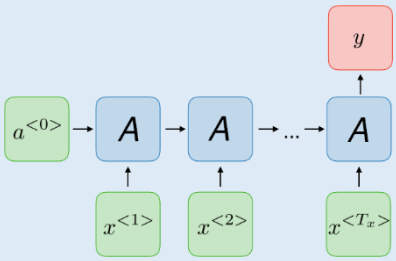
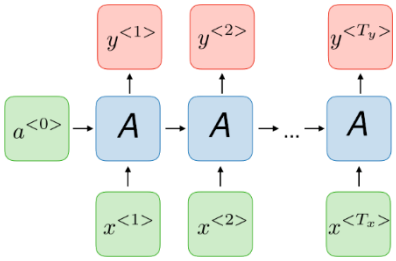
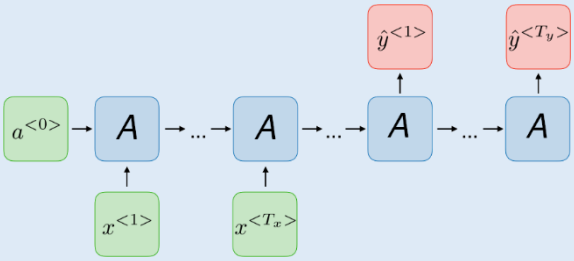
Uno a muchos $T_x = 1,\ T_y > 1$		Generación de música
Muchos a uno $T_x > 1,\ T_y = 1$		Clasificación sentimientos Predicción de bolsa
Muchos a muchos $T_x = T_y$		Reconocimiento de entidades nombradas (localizar y clasificar categorías predefinidas)
Muchos a muchos $T_x \neq T_y$		Traducción automática

Tabla 4. Clasificación de tipos de RNN según su estructura. Siendo T_x y T_y el número de entradas y salidas respectivamente.
Fuente: Elaboración propia a partir de Amidi & Amidi (2018).

A continuación, se va a estudiar cómo funciona una Red Neuronal Recurrente del tipo Estándar (ver Figura 25)

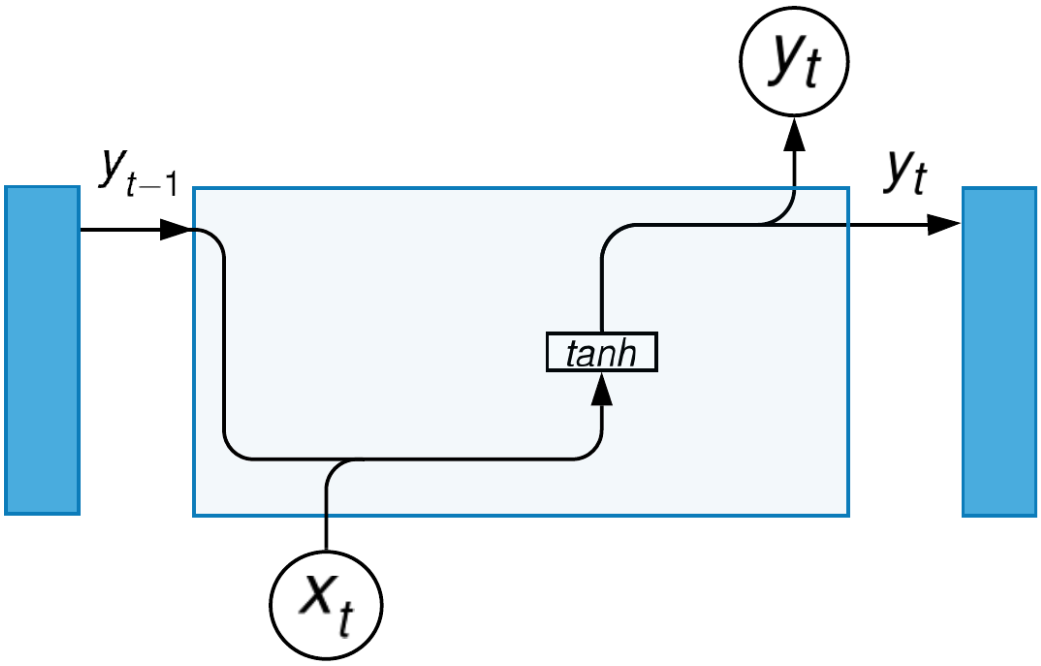


Figura 25. Red Neuronal Recurrente.
Fuente: Elaboración propia a partir de Olah (2015).

La salida del módulo anterior se concatena con el input del módulo actual. Este se pasa por una capa de red neuronal con función de activación tanh calculando así el *output* del módulo anterior y parte del *input* de la siguiente. El *output* por tanto viene representado por la Ecuación 42.

$$y_t = \tanh(W_C \cdot [y_{t-1}, x_t] + b_C)$$

Ecuación 42. Salida de Red Neuronal Recurrente estándar de la Figura 25.

2.3.10.2 LSTM. Long-Short Term Memory.

Las redes neuronales del tipo LSTM (Memoria de Largo-Corto Plazo) fueron desarrolladas para tener la capacidad de aprender dependencias a largo plazo (Hochreiter & Schmidhuber, 1997). Más tarde han sido mejoradas permitiendo que funcionen muy bien en una gran variedad de problemas.

Este tipo de redes tienen cuatro capas de redes neuronales en lugar de una, que tiene la estándar, como se puede ver en la Figura 26.

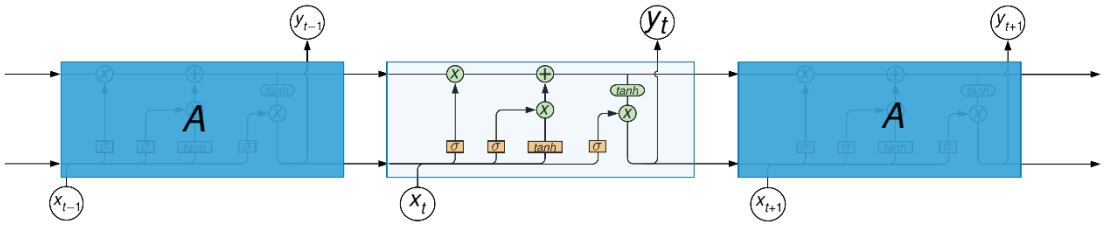


Figura 26. Esquema de una Red Neuronal Recurrente del tipo LSTM.
Fuente: Elaboración propia a partir de Olah (2015).

La transferencia de vectores viene representada por las flechas. Cada uno de los rectángulos naranjas representan una compuerta de red neuronal, ya sean sigmoides o tanh. Cada uno de dichos rectángulos acaban en un operador punto a punto.

La razón por la cual este tipo de redes neuronales han crecido en fama por su utilidad es debido a la línea horizontal que cruza el diagrama. Esta línea horizontal llamada *Cell State* (estado de las células) dónde se añade información nueva o elimina información que ya no se necesita regulado todo por otras estructuras llamadas *gates (*puertas*).* Así se puede mantener información a largo plazo e ir eliminando información que ya no se requiere.

En una LSTM se pueden diferenciar distintas partes en las cuales cada una tiene una función concreta. Estas son *Forget Gate Layer, Input Gate Layer and tanh Layer, Cell State* y *Output Layer.*

Forget Gate Layer.

En esta puerta se decide qué información previa y actual se va a eliminar de la de la *Cell State*. Esta decisión se hace mediante una red neuronal con una función de activación sigmoide. Mira a y_{t-1} y a x_t, sacando un número entre 0 y 1 para cada número en la *Cell State*, siendo 1 mantener esto completamente y 0 deshacerse de esto completamente.

$$f_t = \sigma\left(W_f \cdot [y_{t-1}, x_t] + b_f\right)$$

Ecuación 43. Salida de 'Forget Gate Layer'.

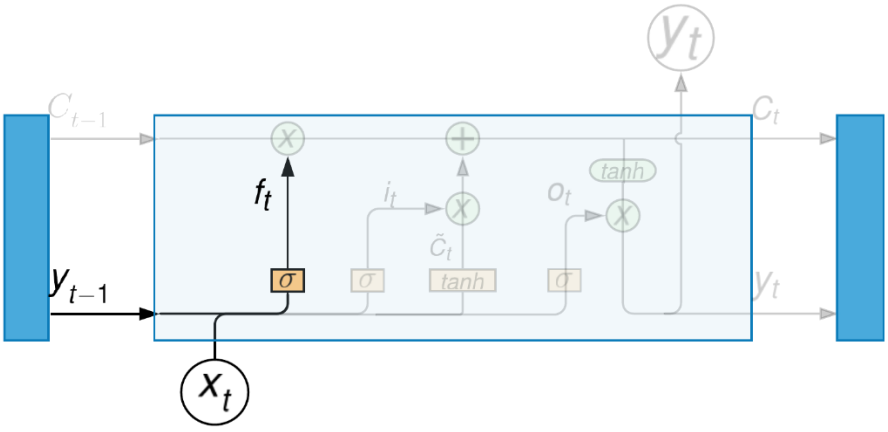


Figura 27. Capa de la puerta del olvido (Forget gate layer).
Fuente: Elaboración propia a partir de Olah (2015).

Input Gate Layer and tanh Layer

En este paso, representado gráficamente en Figura 28, se decide la nueva información que se añadirá a la *Cell State.* Se divide en dos partes:

- La primera, una capa densa con activación sigmoide donde se decide que nueva información se actualizará.
- A continuación, un vector de nuevos candidatos $\tilde{C}_t$ se crea a partir de una capa densa con activación *tanh.*

$$i_t = \sigma(W_i \cdot [y_{t-1}, x_t] + b_i)$$

Ecuación 44. Salida de la función sigmoide donde se decide qué información se actualizará.

$$\tilde{C}_t = tanh(W_C \cdot [y_{t-1}, x_t] + b_C)$$

Ecuación 45. Salida de la función tanh donde se decide que nuevos candidatos se añaden.

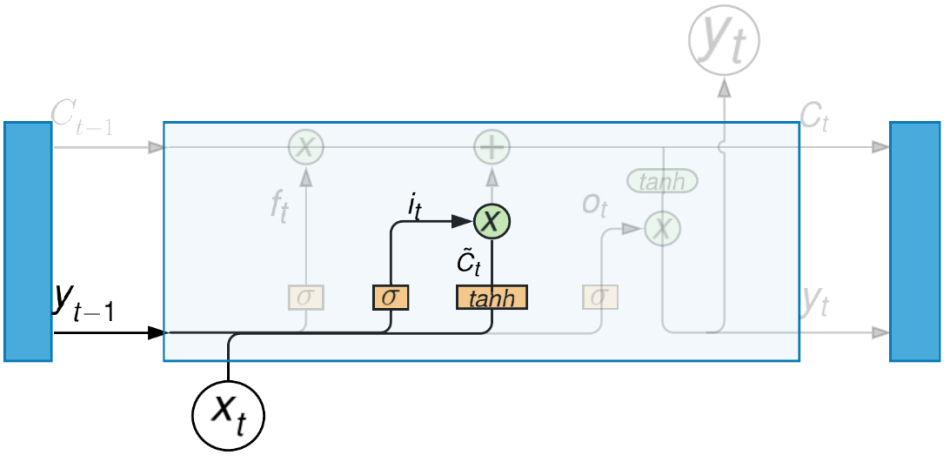


Figura 28. Capa de Puerta de Entrada y Capa de tanh (Input Gate Layer and tanh Layer).
Fuente: Elaboración propia a partir de Olah (2015).

Cell State

La siguiente parte es actualizar la *Cell State* (ver Figura 29). Previamente se ha decidido qué olvidar, qué actualizar y qué añadir.

Multiplicamos punto a punto el *Cell State* anterior C_{t-1} por f_t para omitir los datos que ya no son necesarios. A continuación, añadimos $i_t * \tilde{C}_t$ donde se añaden los nuevos valores candidatos por cómo queremos actualizar cada valor estado. Por tanto, el nuevo valor C_t vendrá dado por la Ecuación 46.

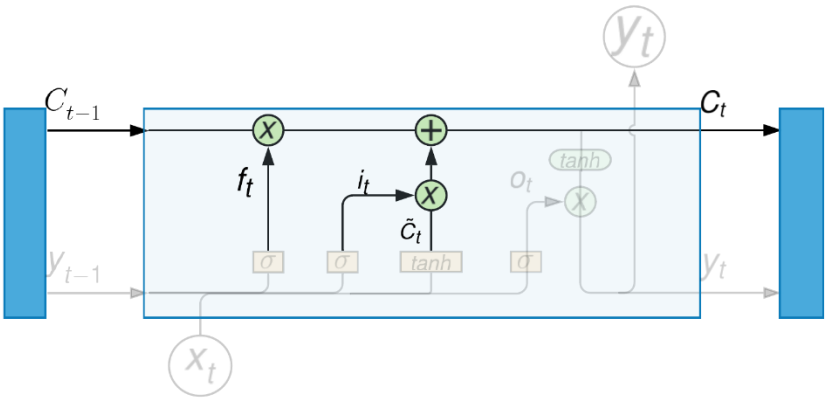


Figura 29. Estado de célula (Cell State).
Fuente: Elaboración propia a partir de Olah (2015).

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

Ecuación 46. Actualización del Cell State.

Output Layer

En esta capa se decide qué será el output. Dicho output será dependiente del *Cell State*. Primero se pasará la convergencia del output anterior con el input actual por una capa sigmoide. Esta capa decide que partes del *Cell State* van a salir. Pasamos a través de una tanh la *Cell State* para que los valores estén entre -1 y 1. Estos datos los multiplicamos por la salida de la puerta sigmoide o_t, saliendo así solo las partes que se hayan decidido (ver Figura 30).

$$o_t = \sigma(W_o \cdot [y_{t-1}, x_t] + b_o)$$

Ecuación 47. Salida de la función sigmoide.

$$y_t = o_t * \tanh(C_t)$$

Ecuación 48. Output.

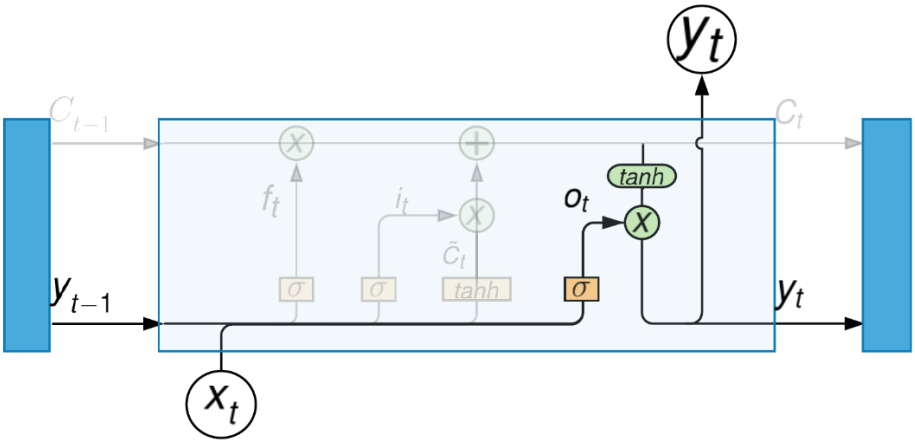


Figura 30. Capa de Salida (Output Layer).
Fuente: Elaboración propia a partir de Olah (2015).

2.4 Predicción de índices de bolsa con RNA

Como se ha comentado en capítulos anteriores, debido a los riesgos y rendimientos variables de la bolsa, la predicción de valores es un tema de mucha importancia para los que quieren invertir en ella. Tener la capacidad de pronosticar la tendencia o el precio de las acciones sería una información muy valiosa para los inversores. Por ello, este problema ha sido estudiado por numerosos economistas, matemáticos e ingenieros.

Anteriormente hemos visto las medias móviles como sistema de predicción, pero hay bastantes metodologías que han sido utilizadas por la comunidad académica en el entorno de la economía como ARIMA o regresión múltiple entre otras; y en el campo del Machine Learning se han empleado técnicas como algoritmos genéticos, *K-nearest neighbor*, **redes neuronales artificiales**, *Support Vector Machines*, *Clustering*, etc. (M. C. García et al., 2013). Pero no solo se emplean dichos métodos sino también indicadores técnicos para tratar de predecir tendencias.

Si hay tantos métodos, ¿por qué vamos a usar redes neuronales artificiales?

Para contestar mejor a dicha pregunta de la se han leído más de cuarenta documentos científicos, capítulos de libro, tesis y escritos de simposios, todos ellos relacionados con el uso de redes neuronales artificiales aplicados a la economía, y más concretamente la predicción bursátil. La mayoría de estos escritos fueron encontrados en Google Académico y en Polibuscador, la herramienta de búsqueda de la Universitat Politècnica de València. Después se ha aplicado un criterio de selección de textos basados en la complejidad, es decir, trabajos extremadamente complicados o sencillos han sido descartados. También por la aplicabilidad y utilidad a este trabajo. Los textos se han dividido en tres tablas:

- En la Tabla 5 se estudia la importancia y la aplicabilidad que tienen las redes neuronales en cara a la economía, no solo a las finanzas.

- Para la Tabla 6 se han analizado tres documentos que aplican redes neuronales artificiales para la bolsa española. Se ha creído oportuno añadir de columnas la estructura o tipo de redes que se han empleado, que métodos se han empleado para medir los resultados y finalmente los resultados y conclusiones que ha llegado cada artículo.

- Finalmente, en la Tabla 7 se han examinado otros diecisiete documentos donde se explora el uso de redes neuronales para el cálculo de distintos índices de alrededor del mundo y, en algunos casos, la predicción de precios de algunas empresas que cotizan en distintas bolsas. En esta tabla se ha creído oportuno incluir también las mismas columnas que en la Tabla 6, más otra en la que indica el mercado en el que se realiza el estudio.

TÍTULO	AUTOR Y FECHA	RESUMEN
NEURAL NETWORKS FOR TIME-SERIES FORECASTING	L. Zhang et al. (2017)	En este capítulo del libro se trata como las redes neuronales se han convertido en uno de los métodos más importantes para la predicción de series temporales. Se discuten ventajas sobre modelos tradicionales y desarrollos recientes. Llega a la conclusión de que se requiere un estudio cuidadoso de cada uno de los problemas que se quieran investigar para adaptarse de mejor manera a este. Aunque dependiendo del problema no siempre las redes neuronales llegan a superar a los métodos tradicionales en tener mejores resultados.
APLICACIONES DE REDES NEURONALES EN ECONOMÍA	Güemes et al., n.d.	Este documento trata de como las redes neuronales han encontrado un hueco en el mundo de la economía y presenta las principales líneas de investigación existentes. Explica también tres motivos por lo que se está subiendo el uso de RNA frente a métodos tradicionales. La primera es su efectividad en series temporales. Segundo, es que no es necesaria una especificación del tipo de relación funcional entre variables permitiendo así un trabajo más cómodo con modelos multiperiodos y multivariables. Y, por último, su facilidad en la introducción de distintos indicadores. También duda su efectividad en predicciones a muy corto plazo en los mercados financieros y la dificultad de entrenar y elegir la mejor estructura para cada problema.
ARTIFICIAL NEURAL NETWORKS IN TRADING SYSTEMS	Voegt (2017)	El autor destaca la eficiencia de las RNA como técnica de predicción de finanzas por encima de otros métodos, por su capacidad de reconocer patrones con conjuntos de datos ruidosos. También hace hincapié en su habilidad de trabajar con la no-linealidad y la complejidad caótica que componen los mercados financieros. Resalta, también, que en la mayoría de la literatura se emplean RNA del tipo MLP, aunque hay otros tipos de RNA, como las LSTM, que pueden dar mejores resultados. Explica la dificultad que se tiene en optimizar y elegir bien el número de capas y neuronas en cada capa ya que se debe hacer a partir de prueba y error. Concluye que, a pesar de ser un muy buen indicador, se debe tener en cuenta las limitaciones que tienen las RNA y como entrenar y asegurarse de que se emplean técnicas de entrenamiento correctas para evitar el *overfitting.*

Tabla 5. Documentos sobre la importancia redes neuronales artificiales. Elaboración propia.

TÍTULO	AUTOR Y FECHA	METODOLOGÍA	MEDICIÓN DE RESULTADOS	RESULTADOS
STAR AND ANN MODELS: FORECASTING PERFORMANCE ON THE SPANISH "IBEX-35" STOCK INDEX	Pérez-Rodríguez et al. (2005)	Smooth transition autoregression (STAR)y RNA del tipo MLP	RMSE, MAE, MAPE	RNA da mucho mejor resultados que STAR incluyendo gastos de transacciones.
REDES NEURONALES Y SU APLICACIÓN PREDICTIVA EN LA BOLSA DE VALORES ESPAÑOLA	Aparicio et al. (1999)	RNA del tipo MLP	-	-
ON THE PROFITABILITY OF TECHNICAL TRADING RULES BASED ON ARTIFICIAL NEURAL NETWORKS: EVIDENCE FROM THE MADRID STOCK MARKET	Fernández-Rodríguez et al. (2000)	RNA del tipo MLP	Sharpe Ratio	A la larga RNA da peores resultados que *buy-and hold- strategy*, pero a corto da muchos mejores resultados.

Tabla 6. Documentos de predicción de IBEX-35 mediante RNA. Elaboración propia.

Se debe de comentar que el último documento (Fernández-Rodríguez et al., 2000) no es del IBEX-35 sino de la bolsa de Madrid. Sin embargo, por cercanía se ha decidido colocar en la misma tabla.

TÍTULO	AUTOR Y FECHA	MERCADO	METODOLOGÍA	MEDICIÓN DE RESULTADOS	RESULTADOS
AN EMPIRICAL METHODOLOGY FOR DEVELOPING STOCKMARKET TRADING SYSTEMS USING ARTIFICIAL NEURAL NETWORKS	Vanstone & Finnie (2009)	General	RNA del tipo MLP junto a análisis fundamental y técnico. Con selección de variables.	MAE junto a *benchmarking* ideado por los autores.	Presenta una metodología general para diseñar un sistema de predicción funcional, aunque destaca que este sistema no tiene por qué funcionar en todos los mercados
NEURAL NETWORK TECHNOLOGY FOR STOCK MARKET INDEX PREDICTION	Komo et al. (1994)	Índice industrial Dow Jones	RNA del tipo MLP backpropagation y Radial Basis Function	MSE	Hasta un 90% de aciertos en la predicción en ambos tipos de redes. No obstante, la RBF parece tener unos resultados ligeramente mejores.
APPLICATION OF NEURAL NETWORKS TO AN EMERGING FINANCIAL MARKET: FORECASTING AND TRADING THE TAIWAN STOCK INDEX	Chen et al. (2003)	Índice de la bolsa de Taiwan	RNA probabilísticas	FPE (Akaike's Final Prediction Error)	Aciertos más del 50% de las veces. También demuestra que obtiene más beneficios que empleando otras estrategias de inversión.

PRONÓSTICO DE BOLSA DE VALORES EMPLEANDO TÉCNICAS INTELIGENTES	Mirledy et al. (2006)	Bolsa de Valores de Colombia	RNA del tipo MLP y redes neurodifusas	Error promedio	Las redes neuronales presentan mejores resultados y menor tiempo computacional que las redes neurodifusas.
APLICACIÓN DE LAS REDES NEURONALES AL PRONÓSTICO DE PRECIOS EN EL MERCADO DE VALORES	Villada et al. (2012)	Dos empresas que negocian tanto en la bolsa de Valores de Colombia como en la de Nueva York	RNA del tipo MLP	RMSE y MAPE	Se emplean distintas redes neuronales con distintos números de capas y neuronas. Se llega a la conclusión de que es un buen predictor.
APPLICATION OF INTEGRATED DATA MINING TECHNIQUES IN STOCK MARKET FORECASTING	Huang & Lin (2014)	Dos empresas de la bolsa de Taiwan: Taiwan Semiconductor Manufacturing Company y Evergreen Marine Corporation	Un sistema de Top-Down trading con Análisis técnico finalmente pasando por una RNA del tipo MLP	RMSE	El rendimiento de la inversión de la cartera fue del 54% y 128% (durante la recesión de 2008-2009) con más de 80% y 96% de aciertos respectivamente.

APPLYING ARTIFICIAL NEURAL NETWORKS TO PREDICTION OF STOCK PRICE AND IMPROVEMENT OF THE DIRECTIONAL PREDICTION INDEX - CASE STUDY OF PETR4, PETROBRAS, BRAZIL	De Oliveira et al. (2013)	Empresa de la bolsa de Brasil: Petrobas	RNA del tipo MLP	MAPE, RMSE y el coeficiente de Theil	Un MAPE y RMSE muy reducidos. Demuestra que es un método fiable para la predicción del mercado brasileño.
A LSTM-BASED METHOD FOR STOCK RETURNS PREDICTION: A CASE STUDY OF CHINA STOCK MARKET	K. Chen et al. (2015)	Índice de Shangai y de Shenzen	Redes Neuronales recurrentes del tipo LSTM	*Accuracy*	Comprueban que a medida que iban eligiendo variables distintas se llegaba a mejores resultados. Aunque hayan podido demostrar que la LSTM es bueno para aprender, la impredecibilidad del mercado china ha provocado que tengan resultados no tan buenos como los esperados.
PREDICTING STOCK INDEX USING NEURAL NETWORK COMBINED WITH EVOLUTIONARY COMPUTATION METHODS	El-Henawy et al. (2010)	Índice de la bolsa de Corea KOSPI 200	RNA del tipo MLP con tres métodos para optimizar las variables sustituyendo *backpropagation*.	RMSE	Dos de los tres métodos buenos resultados con menos del 50% de error.

HYBRID FUZZY NEURAL NETWORK TO PREDICT PRICE DIRECTION IN THE GERMAN DAX-30 INDEX	F. García et al. (2018)	Índice alemán DAX-30	RNA difusa hibrida en concreto un sistema híbrido de inferencia neural difusa	Hit Ratio	Concluye que aplicando este tipo de redes junto el análisis de los *inputs* son una manera de obtener unos resultados más que buenos con un Hit Ratio de hasta el 76.24%.
INTEGRATING METAHEURISTICS AND ARTIFICIAL NEURAL NETWORKS FOR IMPROVED STOCK PRICE PREDICTION	Göçken et al. (2016)	Índice de la bolsa de valores de Turquía	Método hibrido donde compara RNA mezcladas con algoritmos genéticos y algoritmos de la búsqueda de la armonía con una RNA normal	MAE, MSE, RMSE, MAPE, MSPE	La RNA normal daba mejores resultados en la muestra de entreno, pero en la de testeo las otras dos hacían mejor trabajo. Aunque se tiene que decir que en el trabajo se limitaron a una sola capa oculta.
STOCK MARKET'S PRICE MOVEMENT PREDICTION WITH LSTM NEURAL NETWORKS	Nelson et al. (2017)	Distintas empresas de la bolsa de valores de Brasil	Redes Neuronales recurrentes del tipo LSTM	*Accurarcy, precision, recall* y *F-mesure*	Mejora las *baselines* propuestas y demuestra que los algoritmos elegidos tienen una capacidad aceptable de aprendizaje. Comparado con otros modelos de RNA mejora de manera considerable los resultados.

NEURAL NETWORKS IN FINANCIAL TRADING	Sermpinis et al. (2019)	Índices de DJIA, NASDAQ 100 y NIKKEI 225	RNA del tipo MLP, RBF y recurrentes (RNN)	FDR	Todos los métodos tienen resultados positivos para predecir dentro y fuera de la muestra. RNN es el que tiene mejor capacidad de predicción en ambas pruebas.
PRONOSTICO DEL ÍNDICE GENERAL DE LA BOLSA DE VALORES DE COLOMBIA USANDO REDES NEURONALES	Cruz et al. (2009)	Índice Igbc de Colombia	RNA	R^2	Demuestra que las RNA dan mucho mejor resultado que los métodos tradicionales de predicción
THE USE OF DATA MINING AND NEURAL NETWORKS FOR FORECASTING STOCK MARKET RETURNS	Enke & Thawornwong (2005)	Índice S&P 500	RNA del tipo MLP, GRNN (Generalized Regression Neural Network) y PNN (Probabilistic Neural Network) diferenciando entre modelos de estimación (MLP_1 y GRNN) y modelos de clasificación (MLP_2 y PNN)	RMSE, WER	En los modelos de estimación, a pesar de que GRNN tiene mejor desempeño, el rendimiento es mayor para las MLP_1 Los modelos de clasificación son perores, en general, que los de estimación. No obstante, siguen dando resultados positivos.

STOCK PRICE PREDICTION USING LSTM, RNN AND CNN-SLIDING WINDOW MODEL	Selvin et al. (2017)	1721 empresas del índice NSE de India	Emplean tres tipos distintos de arquitecturas de *deep learning*, RNN simple, LSTM y CNN	*Error percentage*	Todos los modelos dan muchísimos mejores resultados que modelos como el ARIMA, siendo las redes convolucionales las que dan un mejor resultado seguido, de la LSTM y finalmente la RNN, para la metodología seguida.
A PERFORMANCE COMPARISON OF NEURAL NETWORKS IN FORECASTING STOCK PRICE TREND	Wu & Duan (2017)	Índice CSI 300	RNA del tipo BP y del tipo Elman	MSE	Concluyen que RNA del tipo Elman funciona significativamente mejor que del tipo BP.

Tabla 7. Documentos de predicción de mercados financieros usando RNA. Elaboración propia.

La lectura y estudio de toda esta literatura ha permitido al autor entender mucho mejor distintos aspectos en los que se centra este trabajo y se va a intentar responder a la pregunta que se había planteado previamente del porque se van a emplear las redes neuronales artificiales en los siguientes párrafos y profundizar en las conclusiones que se han podido sacar.

Hay tres tipos de predicciones: las lineales, las no lineales y las híbridas. Las lineales para la predicción de series de tiempo financieras son más simples y tienen mayor facilidad para interpretar. Por ello se siguen usando, como medias móviles o regresión múltiple (M. C. García et al., 2013). Sin embargo, se ha demostrado en numerosos estudios que las RNA cuentan con la capacidad de reflejar las características no lineales, además de las lineales, de índices de bolsa en diferentes países (Villada et al., 2012). En la mayoría de los casos modelos de RNA dan mejor resultado que otros métodos (Atsalakis & Valavanis, 2009).

No solo eso, dentro de las redes neuronales hay distintos tipos que funcionan mejor que otros. En trabajos como los de K. Chen et al. (2015), Nelson et al. (2017), y Sermpinis et al (2019) se señala la superioridad de las redes neuronales del tipo recurrente, en los dos primeros destacando el rendimiento de las LSTM. Por otro lado, Selvin et al. (2017) remarca que a pesar de que RNN y LSTM les han dado muy buenos resultados en predicción de 10 minutos en el futuro (extremadamente a corto plazo), las redes convolucionales han sido capaces de realizar un mejor trabajo en detectar las tendencias. Esto es debido a que no siempre los cambios que ocurren en un mercado de valores pueden ser en un patrón regular, lo que le permite a la CNN realizar una mejor tarea en un modelo tan a corto plazo.

En la mayoría de estos trabajos, hacen particular hincapié en la selección de *inputs* y la limpieza de datos como base para que se pueda crear un buen modelo. En especial K. Chen et al. (2015) que prueban el mismo modelo con distintos número de variables comprobando la mejoría de la predicción según se añadían. Enke & Thawornwong (2005) también reafirman que no hay ningún modelo consistente para elegir las variables más útiles en la predicción de los mercados bursátiles. Esto puede deberse a que el comportamiento de estos datos no es bien conocido.

La lectura de estas investigaciones también ha permitido discernir mejor los límites del presente trabajo, ya que en un principio se plantearon muchos tipos de variables, de modelos y de tipos de redes neuronales, lo que era inviable poder crear un trabajo de tales dimensiones. Todo esto se ha descrito el siguiente capítulo, donde se describe la metodología seguida para crear los modelos, todas las decisiones tomadas

3. Metodología

En este capítulo se realiza un repaso por todas la elecciones tomadas y materiales que se han empleado para la realización del presente trabajo, al igual que las decisiones de los modelos elegidos apoyándose en la literatura, la limpieza de los datos y la elección de las medidas de resultados.

3.1 Elección de lenguaje de programación

Para programar las redes neuronales artificiales se plantearon distintas alternativas: Matlab, R y Python 3. A continuación, se analiza cada uno de los lenguajes propuestos para llegar a la decisión que finalmente se tomó.

- Matlab. Es un entorno de cálculo numérico multiparadigma y un lenguaje de programación propio desarrollado por MathWorks. Su ventaja principal es que es una herramienta muy potente en la realización de cálculos con grandes cantidades de datos, en concreto con matrices. Tiene una gran gama de librerías. Además, durante la carrera el autor ha trabajado bastante con Matlab así que no habría una curva de aprendizaje alta el emplear nuevas librerías.

- R. Es un lenguaje de programación y un entorno de software libre para la computación estadística y los gráficos. El lenguaje R se utiliza ampliamente entre los estadísticos para desarrollar programas informáticos estadísticos y análisis de datos. Tiene buena gama de librerías para IA, *Deep Learning* y procesamiento de datos. Sin embargo, es más lento y requiere más memoria que otros como Python. Respecto a R, el autor de este trbnajo lo había usado anteriormente pero no tenía un nivel muy alto en este lenguaje, así que la curva de aprendizaje podría ser un poco mayor.

- Python. Es un lenguaje de muy alto nivel que permite expresar algoritmos de forma casi directa. Python viene con una gran cantidad de librerías. Muchas de ellas son para la Inteligencia Artificial y el Aprendizaje Automático. El autor de trabajo nunca había usado Python anteriormente, sin embargo, sí que sabía programar con otros lenguajes orientados a objetos y, en teoría, es un lenguaje relativamente sencillo de aprender.

Finalmente se decidió emplear Python a pesar de que el autor del Trabajo Fin de Grado no tenía formación previa en este lenguaje. La razón de ello es que Google dispone de una herramienta llamada *Google Colaboratory* dónde se puede programar y ejecutar Python en cualquier navegador. Colab es un servicio en la nube, basado en los Notebooks de Jupyter, que permite el uso gratuito de las GPUs (*Graphics Processor Unit*, coprocesador dedicado al procesamiento de gráficos u operaciones muy grandes) y TPUs (*Tensor Processing Unit,* un acelerador de IA de circuito integrado desarrollado por Google específicamente para el aprendizaje redes neuronales) .Esto supone una gran ventaja frente al resto de lenguajes debido al hardware del que se dispone para hacer el trabajo, ya que los servidores de Google funcionan mucho más rápidos que el ordenador portátil del que se dispone. Además, poder

emplear GPU como acelerador por *hardware* es una gran ventaja para ahorrar tiempo de ejecución. Para obtener los conocimientos necesarios en Python se realizaron tres cursos:

- Introducción a Python. Para poder hacer más tarde programas más complejos se debía tener una base en el lenguaje. Para ello se realizó un curso de 22 horas por la página Udemy.

- Conocimiento de las distintas librerías para el tratamiento, limpieza y operaciones de manejo de datos para el uso de estos en *machine learning.* Para ello se realizó otro curso en Udemy siendo este de 25 horas.

- Introducción y perfeccionamiento al uso de la librería tensorflow y keras para *deep learning*. En este caso se realizó un curso en Udacity de 16 horas.

3.2 Hardware

Dado que la mayoría de los cálculos se realizaron en la nube, y el procesamiento de datos previos se realizaron en Excel, con un ordenador con características técnicas media serviría. Para ello se ha realizado el proyecto enteramente con un ordenador portátil MSI Modern 14 A10M con un Intel Core i5 con 4 núcleos a velocidad 1.6GHz, 16 Gb de memoria RAM de DDR4 a 2666MHZ y un disco duro SSD de 512GB.

Aparte, desde Google Colab se tiene acceso gratuito a una GPU Tesla T4 además de 13.7 Gb disponibles de RAM.

3.3 Software

En lo que respecta al sistema operativo se ha empleado Windows 10 de 64 bits. También se han empleado otros programas de Microsoft como el Excel para el tratamiento de datos y el Word para la realización de la memoria.

Para programar se ha empleado, como ya se ha dicho, Python en Google Colab, que emplea la versión 3.6.7. También se han empleado distintas librerías de Python para el diseño, entrenamiento y testeo de las redes neuronales. A continuación, se hace un pequeño resumen de las librerías que se han empleado.

3.3.1 Numpy

NumPy es el paquete fundamental para la computación científica en Python. Es una biblioteca de Python que proporciona un objeto de matriz multidimensional, varios objetos derivados (como matrices y matrices enmascaradas), y un surtido de rutinas para operaciones rápidas en matrices. (https://numpy.org/install/)

Este paquete ha servido para gestionar y estructurar las bases de datos para que la red neuronal pueda usarlos de manera correcta.

3.3.2 pandas

pandas (todas en minúsculas) es un kit de herramientas de análisis de datos basado en Python. Presenta una amplia gama de utilidades, que van desde el análisis sintáctico de múltiples formatos de archivo hasta la conversión de una tabla de datos completa en una matriz NumPy. Estas funcionalidades son de mucha ayuda para el tratamiento de datos. (https://pandas.pydata.org/getting_started.html)

Esta librería se ha usado tanto para importar correctamente los ficheros *.csv* (*coma separated values*) donde se encontraban los datos como para la limpieza primera de datos y estructurar los estos para la entrada y la salida de la red neuronal.

3.3.3 Matplotlib

Matplotlib es una librería de software para generar gráficos a partir de datos contenidos en listas, o vectores, en el lenguaje de programación Python y su extensión matemática NumPy. (https://matplotlib.org/users/installing.html)

Los gráficos de los resultados han sido generados con esta librería.

3.3.4 Keras

Keras es una API de redes neuronales de alto nivel desarrollada con el objetivo de permitir una rápida experimentación. Ser capaz de ir de la idea al resultado con el menor retraso posible es la clave para hacer una buena investigación. Permite que el mismo código se ejecute en la CPU o en la GPU, sin problemas. Es una API fácil de usar que facilita la creación rápida de prototipos de modelos de aprendizaje profundo. Soporta arquitecturas de red arbitrarias: modelos multientrada o multisalida, compartición de capas, compartición de modelos, etc. Esto significa que Keras es apropiado para construir esencialmente cualquier modelo de aprendizaje profundo. Es capaz de correr sobre múltiples back-ends incluyendo TensorFlow, que es el que se emplea en este caso. (https://keras.io/)

Con esta herramienta se han desarrollado, entrenado y modelizado los distintos tipos de redes neuronales que se han empleado en el trabajo.

3.3.5 Scikit-learn

Scikit-learn, o abreviado Sklearn, es una libreria de aprendizaje de máquinas de software libre para el lenguaje de programación Python. Presenta varios algoritmos de clasificación, regresión y agrupación que incluyen *support machine vectors*, *random forest*, *k-means* y DBSCAN, y está diseñado para interoperar con las bibliotecas numéricas y científicas de Python NumPy y SciPy. Sin embargo, en nuestro trabajo nos hemos centrado en sklearn.metrics y sklearn.prepocessing. El primer módulo implementa varias funciones de coste, puntuación y utilidad para medir el rendimiento de la clasificación. El segundo paquete proporciona varias funciones de escalado, centrado, normalización y binarización. (https://scikit-learn.org/stable/install.html)

Esta librería se ha usado para dos cosas:

- El cálculo de las funciones de coste RMSE, MSE y MAE.
- Estandarización de los datos entre -1 y 1.

La estandarización de los datos es muy importante para el Deep Learning, ya que optimiza y acelera el entrenamiento llegando antes a la convergencia. También para poder comparar datos variados. En nuestro caso, el volumen o capitalización está en valores de millones, mientras que el resto de los valores pueden estar en valores de centenas o miles. El llevarlos todos a la misma escala permite que no haya pesos que se actualicen a mucha más velocidad que otros.

3.3.6 Keras tuner

Keras Tuner es una librería que permite definir redes neuronales con Keras, determinar un espacio de búsqueda tanto para los parámetros del modelo (es decir, la arquitectura) como para los hiperparámetros del modelo (es decir, las opciones de configuración), y así optimizar de mejor manera antes de entrenar el modelo final.
(https://www.tensorflow.org/tutorials/keras/keras_tuner)

Esta librería se empleó para optimizar el número de neuronas, units o filtros y la ratio de aprendizaje de los modelos que se proponen.

3.3.7 Otras librerías

Se han empleado otros como pydrive, oauth2client y google.colab para autorizar acceso e importar los datos desde mi Google Drive.

3.4 Base de datos

Los datos del histórico del Ibex-35 fueron sacados en su totalidad de Yahoo! finanzas. Esta página ofrece información financiera, cotizaciones de bolsa, índices bursátiles y sus respectivos históricos y comunicados de prensa tanto corporativos como financieros, entre otras muchas funcionalidades.

Desde la página se puede elegir el intervalo de fechas para la descarga de datos, así como la frecuencia, ya sea diaria semanal o mensual. Dado a que el objetivo del presente trabajo es predecir el cierre del mismo día, se escogió la opción de frecuencia diaria.

Los datos se descargan en fichero .csv. Lo que implica que no viene separado en columnas, sino que se deberá hacer mediante el comando de separación de columnas de Excel. Así se quedan las siguientes columnas: *Date* (la fecha del día), *Open* (el valor del índice al abrir el día), *High* (el máximo al que llegó el índice en ese día), *Low* (el mínimo al que llegó el índice en ese día), *Close* (el valor del índice al cierre del día), *Adj Close* (el valor del índice al cierre del día ajustado por dividendos y splits) y *Volume* (la cantidad de títulos negociados)

En un principio se descargó el histórico diario en su totalidad. Sin embargo, dado que los primeros datos no son estables hasta pasado 1994 se fueron recortando. Una vez pasado 1994 los datos eran mucho más estables, pero seguía habiendo problemas en algunas columnas principalmente en el volumen. Hasta el día 25 de Julio del año 2000 no se tiene una estabilización de esta columna, así que se eliminaron los valores anteriores a esta fecha. Teniendo entonces de fechas entre el 24/07/2000 hasta el 31/12/2019.

Una vez determinada las fechas entre las que se iba a trabajar se debía acabar de realizar la limpieza de datos. Entre fechas había muchos valores que tenían valor *NaN* (*Not a Number*, son valores vacíos no computables). Estos valores podían aparecer por fines de

semana o fiestas nacionales en los que el Ibex-35 no estaba abierto, pero sí estaba puesta la fecha. Mediante el uso del comando de panda *.dropna* todas las filas que contenían al menos un valor *NaN* eran eliminadas.

Una vez los datos limpios se debían preparar para la red neuronal y elegir si era necesario poner otras variables. Se concluyó que lo más lógico era crear un fichero de datos .csv base del que el resto de las bases de datos partieran.

Este fichero base iba a contener la fecha y el *Close* del día junto con todos los datos del día anterior tal como se ve en el ejemplo de la Tabla 8.

Date	Close	Open-1	High-1	Low-1	Close-1	Adj Close-1	Volume-1
25/07/2000	10714.6	10879.6	10924.4	10812.5	10831.5	10831.49	5545600
26/07/2000	10743.8	10831.5	10842.2	10703.4	10714.6	10714.59	7368000
27/07/2000	10726	10722.6	10840.2	10722.6	10743.8	10743.79	6643800
28/07/2000	10560	10743.8	10856.6	10713.4	10726	10725.99	6989400
31/07/2000	10531.6	10726	10735	10526.8	10560	10559.99	6670000

Tabla 8. Ejemplo de los primeros datos del fichero base.

Una vez con el fichero base se empezaron a plantear distintas variables que podrían ser interesantes añadir a los datos:

Datos de empresas.

Los primeros que se plantearon fueron los valores de las empresas del mismo Ibex. Es decir, el mismo sistema de datos que en el fichero base (con los valores del día anterior), pero de aquellas empresas con una ponderación superior al 5% o al 7%, por ejemplo, ya que estás son las que más influencia tendrán sobre el índice. De hecho, ya se habían limpiado los datos del Banco Santander, Inditex, Iberdrola y Telefónica. Sin embargo, finalmente se rechazó la idea, ya que en el propio valor del Ibex ya se encuentran todas las influencias de estas empresas, y añadirlas solamente influenciaría en el aumento de ruido en la propia red.

Precio del oro y del petróleo.

Otros valores interesantes que se plantearon fueron el precio del oro y del petróleo. Estos son dos indicadores importantes de cómo va la economía a nivel global afectando, así también, a la bolsa española. El problema de estos valores fue la imposibilidad de encontrarlos de manera gratuita en las distintas páginas que se han usado, así que finalmente se tuvieron que rechazar estos indicadores. Estos indicadores fueron pensados tras revisar los artículo de De Oliveira et al. (2013) y K. Chen et al. (2015) que emplean el precio del petróleo.

Otros índices internacionales

Los principales índices mundiales también eran tremendamente interesantes colocarlos. Es innegable la influencia que tienen la economía de otros países sobre el nuestro. Se plantearon Nasdaq-100 (EE. UU.), Dax 30 (Alemania), Dow Jones (EE. UU.), Nikkei 225 (Japón), CAC 40 (Francia), FTSE 100 (Inglaterra), S&P 500 (EE. UU.), SSE Composite (China), Bovespa (Brasil), Euro Stoxx (Europa), entre otros. Sin embargo, una vez ya limpios de valores NaN, y al colocarlos junto al fichero de datos base, se vio un gran problema. Cada uno de los países tiene distintos días de fiestas nacionales y por tanto distinto días en los que la bolsa está cerrada. Requería de muchísimo trabajo colocar cada uno de los índices con las fechas que

coincidieran con las españolas. Además, quedaba una duda: con los valores que quedaban vacíos porque no existían datos de ese día en un índice extranjero, pero sí en el español, ¿cómo se debía proceder? Una opción era eliminar los datos del Ibex de ese día también llegando así a tener muchos menos datos. O, tal vez, valía la pena rellenar esos vacíos con la media de toda la columna. Tras deliberar bastante se decidió sustituir todos estos valores por el *Open* del propio día del índice español ya que, en teoría, recoge todos los movimientos durante la noche, incluyendo las influencias de otros índices. Aunque este puede que no sea tan valioso como todos los índices internacionales, se ganaba en optimización de los datos y en un ahorro masivo de trabajo, así que se añadió al fichero base.

Datos macroeconómicos.

Otros datos que se plantearon fueron valores macroeconómicos como el IPC, el Euribor, el PIB o el paro. El problema de estos datos es que en su mayoría son como mucho mensuales llegando alguno a ser trimestrales. Serían datos mucho más interesantes para calcular el valor del índice mensual. Estos se pensaron tras repasar el artículo de De Oliveira et al. (2013) y Vanstone & Finnie, (2009) donde se plantea el uso de algunos componentes macroeconómicos los cuales son descartados, el primero por el estudio de correlaciones y el segundo por el posible *lag* que pueden surgir.

Indicadores técnicos.

Finalmente, los últimos datos que se plantearon fueron indicadores técnicos. Con estos se pretendía intentar capturar ciertas tendencias que por lo que fuera, la red neuronal no fuese capaz de predecir. Así se eligieron 6 de los indicadores técnicos más empleados: Media Móvil simple, Media Móvil exponencial (EMA), *Commodity Channel Index* (CCI), *Relative Strength Index* (RSI), *Moving Average Convergence Divergence* (MACD) y Williams %R. Varios de los indicadores planteados son también utilizados en los trabajos de Chavarnakul & Enke (2008), De Oliveira et al. (2013), F. García et al. (2018), Göçken et al. (2016), Huang & Lin, (2014), K. Chen et al. (2015), Nelson et al. (2017) Patel et al. (2015) y Vanstone & Finnie (2009)

Juntando los indicadores con el fichero *.csv* se creó un archivo que contenía todos los indicadores. Desde este se empezó a trabajar y se separó en ocho subconjuntos de datos cada uno con características distintas. Como base todos los subconjuntos tienen 7 columnas: Fecha, Open (valor de apertura del mismo día), Open-1 (valor de apertura del día anterior), High-1, (valor máximo del día anterior), Low-1 (valor mínimo del día anterior), Close-1 (valor de cierre del día anterior), Adj Close-1 (valor del cierre ajustado del día anterior), Volume-1 (número de títulos negociados en el mercado en el día anterior) y Close (el valor que se trata predecir). En la Tabla 9 vemos las diferencias de cada una de las bases de datos[2] que se han creado.

[2] A partir de ahora el subconjunto de bases de datos serán llamados bases de datos por comodidad.

Base de datos	Contenido
Base	7 columnas base
SMA	7 columnas base + la media móvil simple con n=5
EMA	7 columnas base + la media móvil exponencial con n=5
CCI	7 columnas base + CCI con n=5
RSI	7 columnas base + RSI con n=5
MACD	7 columnas base + MACD con n=5 y n=12
Williams R%	7 columnas base + Williams con n=5
Todos	7 columnas base + todos los indicadores anteriormente calculados

Tabla 9. Bases de datos creadas con sus respectivos contenidos.

Todos los indicadores fueron calculados mediante Excel. Como todos los indicadores técnicos requieren de datos anteriores, finalmente se recortaron los datos para que todas las bases de datos dispusieran del mismo número de datos. Después de todo el proceso de limpieza e integración, los datos recopilados comprenden desde el 28 de agosto del año 2000 hasta 31 de diciembre de 2019. En la Tabla 10 se puede ver la base de datos donde se encuentran todos los indicadores técnicos.

Date	Close	Open	Open-1	High-1	Low-1	Close-1	Adj Close-1	Volume-1	Media_Movil	EMA	CCI	MACD	RSI	R%
29/08/2000	10840.6	10907.5	10845.8	10919.6	10845.8	10907.5	10907.49	3808300	10835.72	10924.4213	-0.18867465	64.0347481	48.8206483	-52.0805066
30/08/2000	10869.6	10840.6	10907.5	10922	10783.6	10840.6	10840.59	5063500	10816.92	10812.6596	61.2222296	52.7458169	41.8476615	-43.9071567
31/08/2000	10884.7	10869.6	10840.6	10911.1	10783	10869.6	10869.59	6173400	10831.22	10888.5801	24.946053	45.613511	46.0249123	-25.1439539
01/09/2000	11170.1	10938.6	10869.6	10937.1	10753.1	10884.7	10884.69	8104500	10864.14	10883.4066	23.6882047	40.7102684	48.4357132	-24.6704332
04/09/2000	11246.3	11170.1	10938.6	11191	10938.6	11170.1	11170.09	9607600	10934.5	11265.6645	166.666667	59.1717135	74.9109318	-4.77277917
05/09/2000	11188.2	11246.3	11170.1	11297.7	11143.3	11246.3	11246.29	7462900	11002.26	11239.8452	132.267578	79.0401195	78.581192	-9.43811972
06/09/2000	11238.5	11188.2	11246.3	11296.3	11128.5	11188.2	11188.19	9589600	11071.78	11170.9849	87.8613345	89.0710346	68.9656002	-20.1065002
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
19/12/2019	9617.2	9622.6	9606.1	9659.2	9606.1	9621.8	9621.8	156463700	9590.1	9634.32188	43.4615496	83.1815197	72.1273385	-21.898263
20/12/2019	9675.5	9620.4	9622.6	9635.4	9567.4	9617.2	9617.2	165383700	9619.84	9611.49271	-62.2818996	88.3278289	70.8291939	-54.3745481
23/12/2019	9659.6	9650.2	9620.4	9675.5	9605.8	9675.5	9675.5	366705600	9642.2	9696.83576	67.027759	96.0039649	77.3012914	-13.52
24/12/2019	9661.8	9632.1	9650.2	9670.9	9639	9659.6	9659.6	119328700	9638	9647.18808	140.640039	99.6555932	71.8655829	-14.7086032
27/12/2019	9700.5	9673	9632.1	9661.8	9607.8	9661.8	9661.8	35436200	9647.18	9666.67064	18.3997557	101.556378	72.2036422	-12.6734505
30/12/2019	9612.6	9672.5	9673	9700.5	9657.5	9700.5	9700.5	103106600	9662.92	9711.77645	124.609939	104.975439	78.0129058	0
31/12/2019	9549.2	9564.9	9672.5	9682.1	9612.6	9612.6	9612.6	95100900	9662	9579.54118	24.4536598	99.4459225	48.9611359	-92.8194298

Tabla 10. Primeros y últimos datos de la base de datos Ibex_todos.csv

3.4.1 División de los datos

Para conseguir unos resultados robustos en el entrenamiento y después en la predicción o test es importante dividir los datos.

Para la división de datos hay dos preocupaciones que compiten entre sí. Con menos datos de entrenamiento, las estimaciones de los parámetros tienen mayor variabilidad. Con menos datos de test, su estadística de rendimiento tendrá mayor varianza. En términos generales, se busca un punto medio dónde ninguna de ambas se sobreponga una encima de la otra.

Para un número relativamente alto de datos se recomienda una separación entre el 70% y el 80% para los datos de entrenamiento y el 30% y el 20% para los datos de test. Para este trabajo se ha decidido elegir una división del 80%-20%. Además, se dividió la parte del entrenamiento en 80%-20% para validación quedando entonces un 64% de entrenamiento, un 16% de validación y un 20% de test. Todo esto apoyándonos en el trabajo de Vanstone & Finnie, (2009)

3.5 Modelos

En muchos de los textos se ha remarcado la dificultad que existe en crear un modelo válido para cada problema:

> Desarrollar un modelo de red neuronal para la aplicación de la predicción de series temporales no es una tarea trivial. Aunque existen muchos paquetes para facilitar el esfuerzo de los usuarios en la construcción de un modelo de red neuronal, es fundamental que los pronosticadores comprendan muchas cuestiones importantes en torno al proceso de construcción del modelo. Crear un modelo de red neuronal exitoso es una combinación de arte y ciencia, y tan solo *software* no es suficiente para resolver todos los problemas en el proceso. Es un escollo arrojar a ciegas datos en un paquete de software y luego esperar que éste proporcione automáticamente un pronóstico satisfactorio Los problemas del modelado de redes neuronales incluyen la elección del tipo y la arquitectura de la red, el algoritmo de entrenamiento, así como la validación, evaluación y selección del modelo. (G. P. Zhang, 2012, p. 466-467)

En este caso, se han elegido los tipos de redes aplicando tanto la literatura estudiada como por recomendaciones de los tutores del trabajo.

En un principio se realizó un modelo con solo capas de tipo Dense y los datos base, es decir, una red neuronal del tipo. Las capas de tipo Dense tienen neuronas simples y se realizó para tener una *baseline* desde la que partir y, así, poder comparar los próximos modelos más complejos. Se probaron distintas arquitecturas cambiando la profundidad hasta que se llegó a que la mejor estructura de las distintas que se examinaron era de 5 *hidden layers,* con decremento potencial en base de dos de número de neuronas, es decir, 64-32-16-8-4. En la Figura 31 se puede ver un esquema de dicha arquitectura. A partir de ahí se desarrollaron el resto de las estructuras propuestas. Además, para tener un *baseline* clásico, también se calculó el error de la media móvil simple y de la media móvil exponencial.

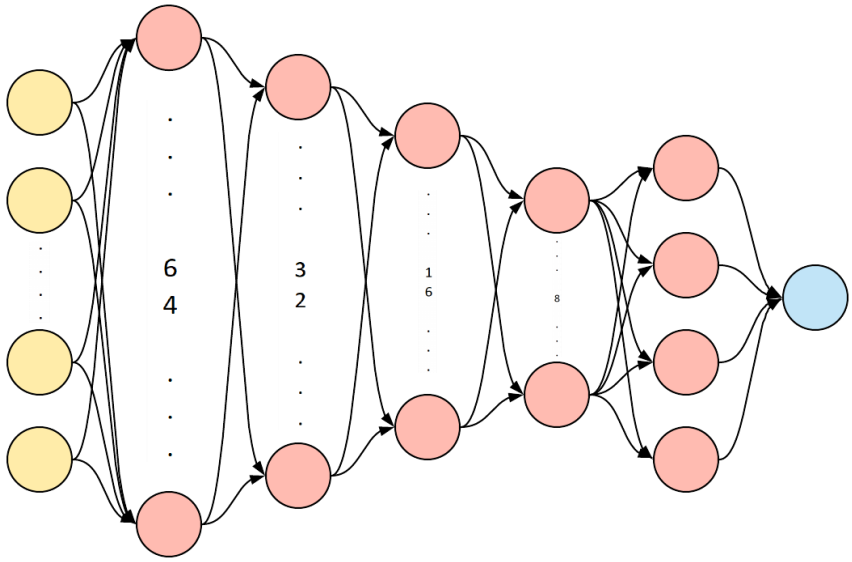


Figura 31. Esquema de la arquitectura de la MLP básica para la baseline. Los círculos amarillos representan los inputs, el azul el output y los rojos las hidden layers.

Respecto a los modelos se han usado tres tipos distintos de redes y dentro de estos tres tipos diferentes de distribución. En todos los modelos tanto la longitud de una de las capas como el *learning rate* son variables. La longitud de estos se elige mediante el uso de keras tuner. Esta librería permite elegir el valor más optimo entre unos valores que se le han establecido para que pruebe. El número de *units*/filtros varía de 32 a 512 en saltos de 32 dando así 16 posibilidades. Los valores que podía elegir del *learning rate* eran 0.01, 0.005, 0.001, 0.0005, 0.0001 o 0.00005, dando 6 posibilidades de ratio de aprendizaje. Por lo tanto, llegamos así hasta 96 posibilidades por cada modelo.

- Redes del tipo Redes neuronales Recurrentes (RNN). Basándonos en los trabajos de Selvin et al. (2017) y Sermpinis et al. (2019) que emplean este tipo de redes y obtienen mejores resultados que con solamente capas básicas del tipo Dense:
  - RNN 0: Esta red neuronal consta de dos capas RNN Simples variando el número de *units* de la primera, manteniendo la segunda en 16 *units*, más tres capas del tipo Dense de 8, 4 y 1 en ese orden.
  - RNN 1: Esta red neuronal consta de dos capas RNN Simples variando el número de *units* de la segunda, manteniendo la primera en 64 *units*, más tres capas del tipo Dense de 8, 4 y 1 en ese orden.
  - RNN 2: Esta red neuronal consta de una capa RNN Simple variando el número de *units* de esta, más tres capas del tipo Dense de 8, 4 y 1 en ese orden.

- Redes del tipo *Long short term memory* (LSTM). K. Chen et al. (2015), Nelson et al. (2017) y Selvin et al. (2017) emplean tipo de redes LSTM en sus modelos obteniendo resultados más que satisfactorios:
  - LSTM 0: Esta red neuronal consta de dos capas LSTM variando el número de *units* de la primera, manteniendo la segunda en 16 *units*, más tres capas del tipo Dense de 8, 4 y 1 en ese orden.

  - LSTM 1: Esta red neuronal consta de dos capas LSTM variando el número de *units* de la segunda, manteniendo la primera en 64 *units*, más tres capas del tipo Dense de 8, 4 y 1 en ese orden.
  - LSTM 2: Esta red neuronal consta de una capa LSTM variando el número de *units* de esta, más tres capas del tipo Dense de 8, 4 y 1 en ese orden.

- Redes del tipo LSTM con una capa convolucional de 1 dimensión. Este modelo fue recomendado por los tutores del trabajo:
  - CONV1D 0: Esta red neuronal consta de dos capas LSTM variando el número de *units* de la primera manteniendo la segunda en 16 *units*, colocando entre ellas una capa CONV1D de 32 filtros, más tres capas del tipo Dense de 8, 4 y 1 en ese orden.
  - CONV1D 1: Esta red neuronal consta de dos capas LSTM fijas de 64 y 16 fijas variando el número de filtros una capa CONV1D que se encuentra entre las dos capas LSTM, más tres capas del tipo Dense de 8, 4 y 1 en ese orden.
  - CONV1D 2: Esta red neuronal consta de dos capas LSTM variando el número de *units* de la segunda manteniendo la primera en 64 *units*, colocando entre ellas una capa CONV1D de 32 filtros, más tres capas del tipo Dense de 8, 4 y 1 en ese orden.

Se eligió cambiar dos hiperparámetros solo por el aumento de cálculos que tendría que soportar el programa. Con solo 96 posibilidades por cada base de datos tardaba una media de aproximadamente media hora en realizar los cálculos para sacar el resultado. Se intentó variar 4 y 3 variables (el tamaño del *batch*, que es el número de muestras con las que se entrena por iteración, fijado en 32 si no se indica nada, y el número de capas), sin embargo, el tiempo de cálculo lo hacía totalmente inviable. La mayoría de los artículos estudiados realizan un estudio de 3 a 10 modelos distintos, a excepción de dos que realizan uno de 27 (Pérez-Rodríguez et al,. 2005) y otro de 16 (Villada et al., 2012).En este trabajo en cambio se han estudiado 864 modelos distintos para cada una de las 8 bases de datos propuestas para obtener el mejor resultado posible.

Para obtener los hiperpárametros, en este caso *units*/filtros y *learning rate*, se realizaron 30 épocas, es decir, cada uno de los posibles 96 modelos recorrieron 30 veces cada uno de los *datasets*, con un *batch size* fijado de 32. Una vez los hiperparámetros calculados, se entrenaba la red con los resultados que keras tuner nos proporcionaba.

3.6 Medidas de resultados

Hay muchas maneras distintas que se pueden medir los resultados. De los muchos métodos que hay finalmente se decidió emplear el MSE, el RMSE y MAE. Esta decisión se tomó por dos razones: la facilidad que se tiene en calcular estos y por el respaldo que se tiene en la bibliografía. Göçken et al. (2016) han empleado los tres indicadores entre otros. MAE ha sido empleado por Vanstone & Finnie (2009). MSE ha sido utilizado tanto por Darmadi Komo et al., (1994) y Wu & Duan (2017). Finalmente, RMSE ha sido aplicado por De Oliveira et al. (2013), El-Henawy et al. (2010), Huang & Lin (2014), Villada et al.(2012) y Enke & Thawornwong (2005).

4. Resultados

En las siguientes tablas veremos los resultados obtenidos. Se ha divido según el tipo de red empleado y dentro de cada tipo de red, en cada una de las distribuciones propuestas. En cada una de las distribuciones y por cada base de datos se muestra el número de *units*/filtros y *learning rate* que, con ayuda del keras tuner, se han calculado. También el RMSE, MSE y MAE que se han obtenido en el conjunto de test, tras cinco veces de entrenamiento de cada una con los hiperparámetros calculados, se ha elegido aquella con mejor resultado. En la Tabla 11 vemos los resultados obtenidos de las redes neuronales recurrentes simples con cada una de las arquitecturas, respecto a la base de datos empleada.

Tipo de RNA	Modelo	Base de datos	Mejor Unit	Mejor Learning Rate	RMSE	MSE	MAE
RNN	RNN0	Base	192	0.001	154.647836	23915.9533	127.506709
		SMA	384	0.00005	186.6398	34834.4151	153.594052
		EMA	224	0.0001	200.855887	40343.0873	173.69309
		CCI	416	0.00005	194.093955	37672.4632	164.802116
		RSI	384	0.00005	208.199302	43346.9494	176.298793
		MACD	480	0.00005	211.697699	44815.9158	185.759913
		Williams	352	0.0001	221.451923	49040.9544	183.186815
		Todos	320	0.00005	246.878473	60948.9804	213.611484
	RNN1	Base	160	0.0005	195.154293	38085.198	161.92639
		SMA	480	0.00005	233.863696	54692.2282	209.57318
		EMA	512	0.00005	207.281737	42965.7186	176.317177
		CCI	320	0.0001	214.206213	45884.3016	180.840209
		RSI	416	0.00005	227.612332	51807.3735	195.285024
		MACD	512	0.00005	195.728486	38309.6403	165.900962
		Williams	448	0.00005	200.392022	40156.9623	171.512239
		Todos	96	0.005	174.909808	30593.4408	142.139069
	RNN1	Base	512	0.001	140.999284	19880.7981	110.890397
		SMA	320	0.001	154.855779	23980.3124	120.485846
		EMA	416	0.0005	148.77634	22134.3995	119.28596
		CCI	480	0.0005	177.458425	31491.4927	149.574505
		RSI	480	0.0005	164.761837	27146.4629	133.16247
		MACD	128	0.0005	238.809014	57029.7453	207.615419
		Williams	320	0.0001	212.778003	45274.4788	183.542195
		Todos	512	0.0005	225.351822	50783.4438	184.867054

Tabla 11. Resultados obtenidos de las redes neuronales recurrentes simples

En la Tabla 12 vemos los resultados obtenidos de las redes neuronales recurrentes del tipo LSTM con cada una de las arquitecturas, respecto a la base de datos empleada.

Tipo de RNA	Modelo	Base de datos	Mejor Unit	Mejor Learning Rate	RMSE	MSE	MAE
LSTM	LSTM 0	Base	352	0.005	128.483231	16507.9407	96.0168594
		SMA	320	0.005	147.658322	21802.9802	116.966214
		EMA	448	0.01	137.360999	18868.044	100.199122
		CCI	256	0.01	177.412423	31475.1677	138.081125
		RSI	228	0.01	183.109902	33529.2361	144.113549
		MACD	64	0.005	142.2837	20244.6512	112.152042
		Williams	192	0.01	221.451923	49040.9544	183.186815
		Todos	384	0.005	198.380368	39354.7703	164.528861
	LSTM 1	Base	224	0.01	135.091343	18249.6709	98.8541159
		SMA	448	0.0005	176.392217	31114.2143	137.193961
		EMA	320	0.005	134.128547	17990.4672	96.0450098
		CCI	192	0.01	221.451923	49040.9544	183.186815
		RSI	224	0.0005	184.333228	33978.7389	146.114661
		MACD	256	0.01	158.654785	25171.3409	114.20024
		Williams	320	0.0005	176.466247	31140.3362	144.164775
		Todos	480	0.001	185.321251	34343.966	153.582391
	LSTM 2	Base	416	0.05	179.166443	32100.6143	135.372535
		SMA	320	0.01	228.331963	52135.4854	192.441879
		EMA	128	0.01	243.594618	59338.3381	181.99505
		CCI	192	0.005	143.619112	20626.4493	107.165829
		RSI	384	0.0005	172.67866	29817.9198	137.185807
		MACD	352	0.005	149.097898	22230.1832	109.85971
		Williams	448	0.005	179.223177	32120.9471	141.119266
		Todos	128	0.01	243.594618	59338.3381	181.99505

Tabla 12. Resultados obtenidos de las redes neuronales recurrentes del tipo LSTM.

En la Tabla 13 vemos los resultados obtenidos de las redes neuronales recurrentes LSTM con una capa convolucional entre ellas con cada una de las arquitecturas, respecto a la base de datos empleada.

Tipo de RNA	Modelo	Base de datos	Mejor Unit/ Filter	Mejor Learning Rate	RMSE	MSE	MAE
CONV1D	CONV1D 0	Base	288	0.005	145.412582	21144.819	107.274569
		SMA	384	0.01	146.858705	21567.4794	106.554068
		EMA	480	0.01	137.642569	18945.4767	99.5751011
		CCI	352	0.001	140.038441	19610.7649	103.840008
		RSI	128	0.01	168.899368	28526.9965	134.886179
		MACD	384	0.001	147.743131	21828.0329	112.555187
		Williams	128	0.005	196.543715	38629.432	148.910653
		Todos	416	0.001	199.85705	39942.8405	163.132148
	CONV1D 1	Base	416	0.005	136.922803	18747.8538	99.9485868
		SMA	128	0.005	140.28311	19679.351	102.468593
		EMA	64	0.01	135.170683	18271.1135	97.4110792
		CCI	416	0.005	141.745034	20091.6546	104.808965
		RSI	352	0.005	143.848153	20692.2912	106.91921
		MACD	96	0.01	135.945439	18481.1624	97.763775
		Williams	384	0.005	197.422944	38975.8187	149.349993
		Todos	64	0.005	211.178667	44596.4292	172.479044
	CONV1D 2	Base	384	0.005	142.080086	20186.7508	107.010662
		SMA	96	0.01	133.108216	17717.7971	95.2230732
		EMA	352	0.005	138.984021	19316.5582	99.6400959
		CCI	96	0.01	149.343018	22303.337	113.773981
		RSI	352	0.01	146.956338	21596.1652	109.579907
		MACD	480	0.005	130.698137	17082.003	97.8828135
		Williams	352	0.005	220.756443	48733.4073	186.314529
		Todos	128	0.01	296.479315	87899.984	255.383776

Tabla 13. Resultados obtenidos de las redes neuronales recurrentes LSTM con una capa convolucional entre ellas.

En la Tabla 14 como punto de referencia o *baseline* tenemos los resultados de la red neuronal simple con los siguientes resultados:

		Datos base
MLP DENSE	RMSE	207.232933
	MSE	42945.488651
	MAE	176.7618612

Tabla 14. Resultados red neuronal simple MLP.

Finalmente, para poder entender mejor como de buenos son los sistemas propuestos se han calculado también el RMSE MSE y MAE de la media móvil simple y de la exponencial de 5 días de ventana. Estos resultados se recogen en la Tabla 15.

		Datos base
Media Móvil Simple	RMSE	283.913003
	MSE	80606.5933
	MAE	210.827102
Media Móvil Exponencial	RMSE	245.683226
	MSE	60360.2478
	MAE	181.130679

Tabla 15. Resultados de la SMA y EMA.

Con estos resultados compararemos y contrastaremos los resultados de las demás tablas para comprobar la efectividad y rendimiento de los modelos propuestos.

En la Figura 32 y Figura 33 se pueden observar las gráficas de las dos mejores predicciones frente a la Figura 34 y Figura 35 donde se pueden ver las dos peores. Se puede apreciar como las dos primeras se adaptan muy bien a los datos reales a falta de cubrir algunas zonas. Las dos siguientes en cambio, aunque tienen forma similar están muy por debajo del rendimiento que debería tener.

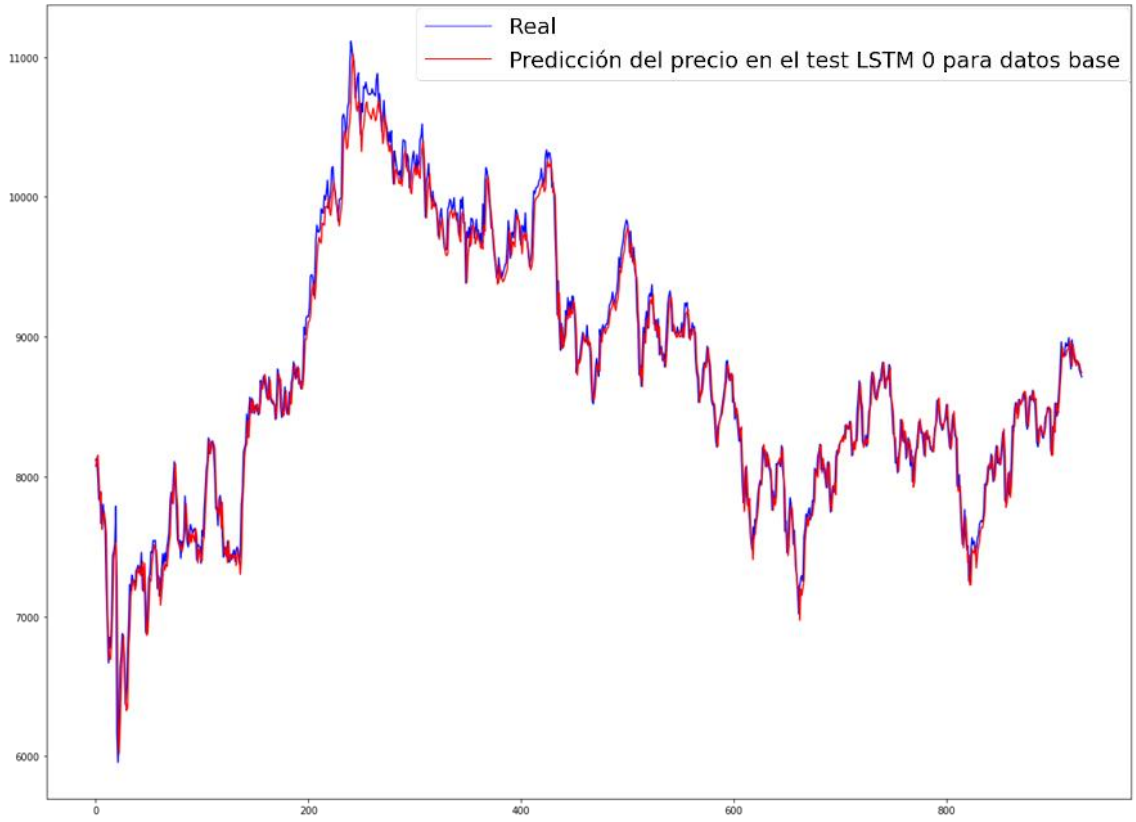


Figura 32. Gráfica real (azul) vs Test (rojo) en LSTM 0 para los datos base. Mejor resultado.

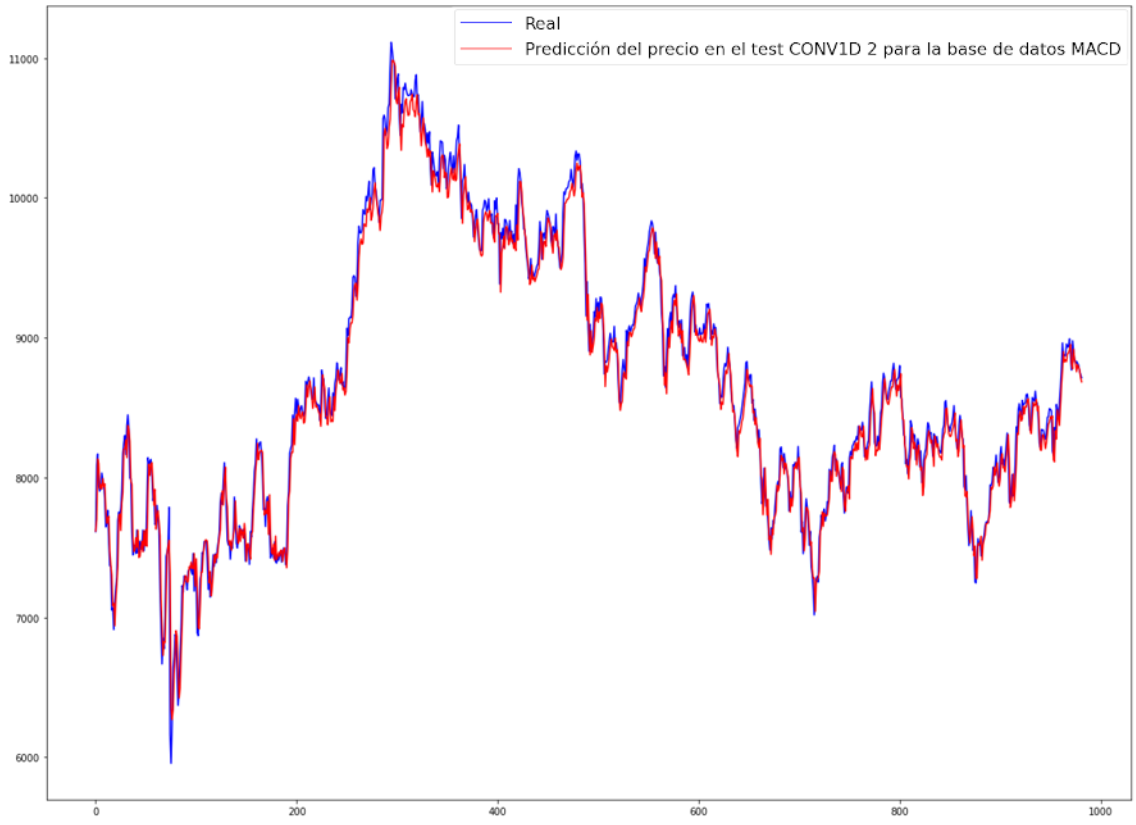


Figura 33. Gráfica real (azul) vs test (rojo) en CONV1D 2 para la base de datos MACD. Segundo mejor resultado.

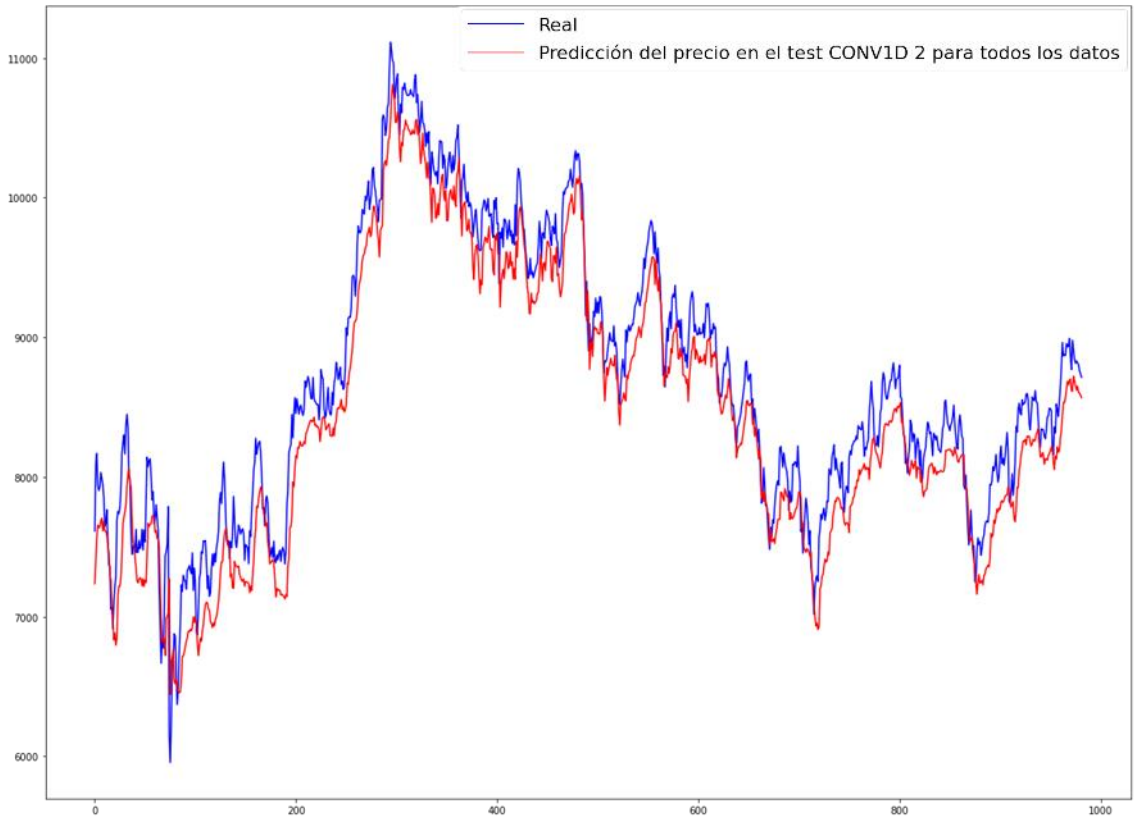


Figura 34. Gráfica real (azul) vs test (rojo) en CONV1D 2 para todos los datos. Peor resultado

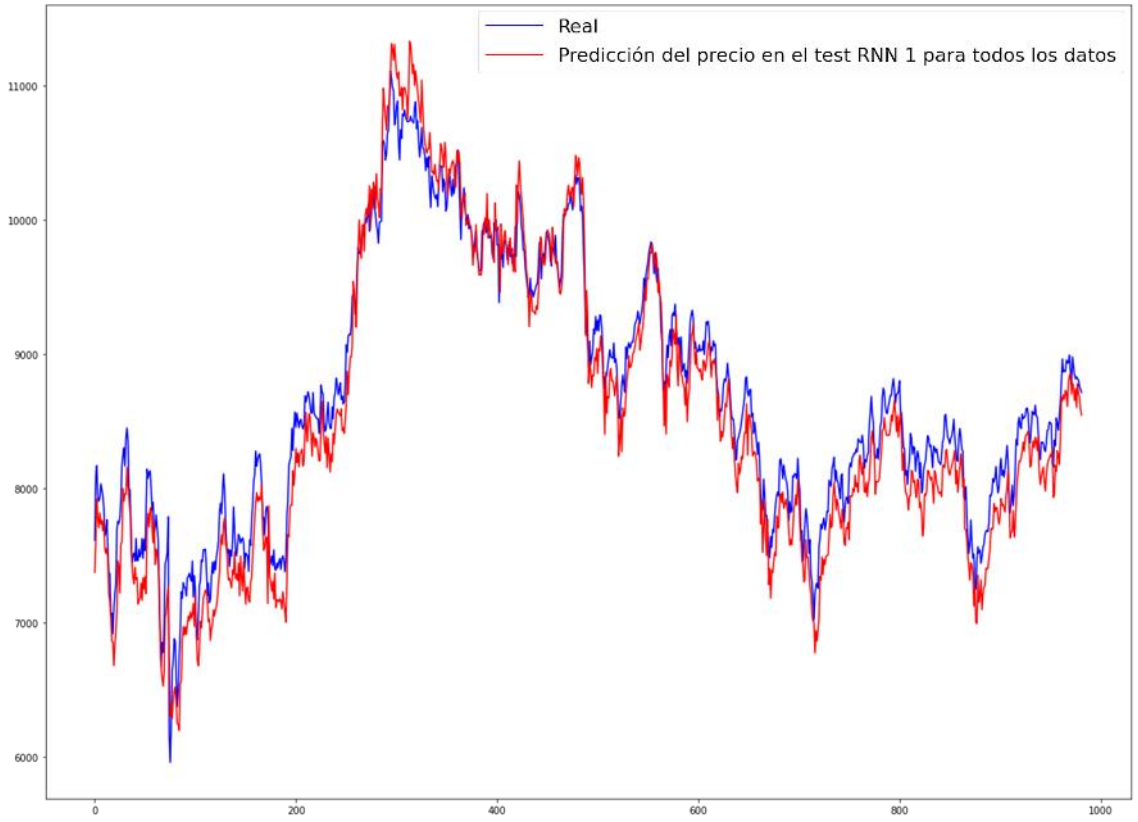


Figura 35. Gráfica real (azul) vs test (rojo) en RNN 1 para todos los datos. Segundo peor resultado.

5. Análisis y discusión de resultados

En el análisis y discusión de resultados solo se han tenido en cuenta el RMSE calculado de cada uno por la facilidad de trabajar con datos más pequeños que con el MSE y dado al apoyo de los artículos estudiados. Recordamos que de *baseline* se tiene los resultados de una red neuronal simple siendo el valor del RMSE de 207.23, como se ha visto en la Tabla 14 y los resultados de la media móvil simple y exponencial siendo el valor del RMSE de estas de 283.91 y 245.68 respectivamente, visto en la Tabla 15.

Se ha realizado con ayuda de Excel un método de ayuda visual para la interpretación de resultados, ordenando los valores en un sistema de colores siendo verde el mejor de los casos y rojo el peor, asociando un color entre medias dependiendo de la distancia entre el máximo y el mínimo. Es decir, si los valores tienen color verde es porque están cerca del mejor dato, amarillo es que tiene un valor intermedio y finalmente rojo si están cerca del dato más deficiente.

En la Tabla 16 tenemos los resultados de todos los modelos, las arquitecturas y las bases de datos planteados. Como podemos ver el mejor resultado obtenido es la estructura LSTM 0 con la base de datos base con una RMSE de 128.48 y el peor con un valor de 296.479 con la estructura de CONV1D 2 con la base de datos con todos los indicadores propuestos.

Resulta curioso la cantidad de buenos valores que tiene la estructura con una capa convolucional, en prácticamente todas las bases de datos, a excepción de las dos últimas. Esto puede deberse por la capacidad que tienen las redes neuronales convolucionales en suavizar los parámetros que se les introduce aislando así la variables 'buenas' del ruido que se pueda introducir (Moreno, 2020).No obstante, también esta estructura contiene el peor resultado de los obtenidos. A lo mejor, la introducción de tantos índices ha provocado tanto ruido que la red no puede suavizarla correctamente.

Si se compara con los resultados obtenidos se puede ver que incluso el peor RMSE de los modelos propuestos se acerca bastante al de la media móvil simple siendo esta RMSE = 283.91. También se puede observar que la gran mayoría de resultados están por debajo de los valores de la media móvil exponencial y de la red neuronal MLP básica indicando que todos los modelos propuestos son mejores indicadores que los modelos base.

También se puede ver la que mejor se adapta a niveles generales son los datos base, no obstante, para los modelos CONV1D tanto EMA como MACD actúan de mejor manera, seguido de muy cerca por la media móvil simple y los datos base. En LSTM, la base de datos de MACD también da buenos resultados. Para todos los datos se puede ver que los resultados no mejoran prácticamente ninguno de los *baselines* propuestos y el indicador Williams superándolos a duras penas. Finalmente, parece ser que añadir distintas variables en las redes neuronales recurrentes simples provoca un peor rendimiento de estas, esto puede darse porque este tipo de redes no tienen ningún tipo de capacidad de olvidar información que no sea valiosa, al contrario que las otras dos estructuras que emplean LSTM, provocando que se 'sobrescriban' datos que ya han sido empleados. Por ejemplo, se ha empleado la media móvil de 5 días, en el día 15 tendrá la media móvil del día 10 al 14, pero como no puede 'olvidar' tendrá en memoria también del día 1 al 5, del 2 al 6, del 3 al 7, etc., datos que probablemente ya no tengan tanta influencia pudiendo provocar que empeoren los resultados.

En su mayoría se puede observar como las redes neuronales recurrentes son un peor tipo de neurona para este tipo de problemas. Sin embargo, se tiene que mencionar que en su mayoría supera al RMSE de la red neuronal simple, el cual tenía un RMSE de 207.23. Tanto para Williams como para todos los indicadores el mejor tipo de red neuronal es la LSTM, no obstante, ambos resultados son bastante altos si los comparamos con los *baselines*. También se puede ver como en su mayoría LSTM actúa entre los otros dos tipos propuestos. Esto puede darse porque LSTM tiene la capacidad de olvidar frente a la red neuronal recurrente pero no tiene la capacidad de suavizar los parámetros como tiene la combinación de LSTM con la capa Convolucional. Finalmente, el que mejor se ajusta al problema con la mayoría de las variables parece ser el que tiene la estructura con una capa convolucional entre dos capas LSTM. Esta combinación da unos resultados más que positivos si los comparamos con los resultados de las medias móviles y de la red neuronal Dense.

En términos generales las redes neuronales recurrentes solas son un peor predictor que los otros dos tipos de redes propuestas estando la mayoría de los resultados, tanto RNN 0 como RNN 1, muy cerca de la red neuronal simple. Finalmente, vemos que en términos generales teniendo en cuenta todas las variables estudiadas el CONV1D 1 es el que mejor se adapta a la mayoría de las bases de datos seguido del CONV1D 0. Esto puede darse porque en el CONV1D 1 la capa que cambia de tamaño es la capa convolucional, adaptándose así a una mejor suavización de los datos permitiendo actuar, así también, mejor las capas LSTM.

		Normal	SMA	EMA	CCI	RSI	MACD	Williams	Todos
RNN	RNN 0	154.6478364	186.6398004	200.8558869	194.0939545	208.1993022	211.6976991	221.4519234	246.878473
	RNN 1	195.1542929	233.8636958	207.2817372	214.2062127	227.6123315	195.7284861	200.3920216	174.9098077
	RNN 2	140.9992841	154.8557793	148.7763405	177.4584252	164.7618369	238.8090142	212.7780035	225.3518224
LSTM	LSTM 0	**128.4832311**	147.6583224	137.3609987	177.4124227	183.1099017	142.2836997	221.4519234	198.3803677
	LSTM 1	135.0913429	176.3922174	134.1285473	221.4519234	184.3332279	158.6547852	176.4662465	185.3212507
	LSTM 2	179.1664431	228.3319632	243.5946183	143.6191118	172.6786604	149.0978981	179.2231769	243.5946183
CONV1D	CONV1D 0	145.4125819	146.8587054	137.6425688	140.0384407	168.899368	147.7431315	196.5437151	199.8570502
	CONV1D 1	136.9228025	140.28311	135.1706827	141.7450337	143.8481533	135.945439	197.4229437	211.1786665
	CONV1D 2	142.0800857	133.1082158	138.9840212	149.3430178	146.9563378	130.698137	220.7564435	**296.4793146**

Tabla 16. Comparación de RMSE de los resultados obtenidos. Siendo verde el mejor resultado y rojo el peor.

Se ha podido ver que a pesar de que el mejor dato obtenido es de la LSTM 0 con la base datos base, parece ser que en términos generales la estructura con la capa convolucional entre dos capas LSTM funciona mejor. Ese dato mejor que el resto puede haberse dado simplemente porque la *seed (*es una secuencia de números aleatoria que cada vez que se ejecuta el programa se crea asegurando que si empleas esa misma serie de vas a obtener los mismos resultados) para este entrenamiento se ha ajustado mucho, pero no tiene por qué ser así en la mayoría de las ocasiones.

También hemos visto como al entrenar las redes con el fichero de datos con todos los indicadores ha descendido el rendimiento de la red, en términos generales. Esto puede darse porque al añadir tantas variables se crea tanto ruido que la red no es capaz de discernir entre que es ruido y no.

Finalmente, se ha podido comprobar que se han creado diversos modelos que en su mayoría actúan mucho mejor que métodos clásicos de predicción de tendencias y que modelos simples de redes neuronales.

Con todo esto, se ha visto que el mejor predictor es la LSTM 0 para los datos base con un RMSE de 128.48, siendo muy superior a la media móvil simple y exponencial y a la rede neuronal simple. Sin embargo, se ha visto que las redes con una capa convolucional dan muy buenos resultados para la mayoría de las bases de datos. Por ello, la recomendación que el autor da es el uso del modelo CONV1D 1, por la estabilidad que tiene en el mayor número de bases de datos, teniendo la mejor media de RMSE de todos los modelos.

Respecto a la base de datos, es más complicado de identificar cual es la mejor opción. En algunos casos el uso de la media móvil exponencial o del MACD nos da mejores resultados que los datos base. Sin embargo, si nos fijamos en la **¡Error! No se encuentra el origen de la r eferencia.**, los mejores datos a nivel general son los que se ha empleado de base con una RMSE media de 150.88.

6. Conclusión

Se han comparado la predicción de precio de cierre a un día del Ibex-35 de cerca de 900 modelos distintos planteados con distinta profundidad, topología, estructura y *learning rate* para cada una de las 8 bases de datos propuestas.

Tras el entrenamiento de todos los modelos planteados con sus respectivas bases de datos, podemos extraer distintas conclusiones. La primera y principal es que, a excepción de algunos casos, todos los modelos propuestos son mejores que los *baselines* propuestos, y, por tanto, que algunos modelos clásicos de predicción como las medias móviles simples y exponenciales.

Las redes neuronales recurrentes funcionan mejor que las redes neuronales simples a nivel general. Sin embargo, dejan que desear si las comparamos con los otros dos tipos de red neuronales propuestas.

La capacidad de olvidar que tienen las redes neuronales LSTM permite que se deshaga de datos que ya no necesita, haciendo que la predicción mejore de manera importante respecto a la red neuronal simple y a la recurrente simple. Sin embargo, parece ser que tiene mucho en cuenta el ruido que tiene el propio sistema, provocando que la predicción no sea tan correcta como se esperaba.

Finalmente, se ha comprobado la creación de modelos híbridos con capas convolucionales y de tipo LSTM ha dado unos resultados más que favorables respecto a las otras dos topologías propuestas, y, por ello, a los *baselines* propuestos. Juntar la capacidad de recordar de las LSTM junto a la capacidad de suavizar el ruido de la capa convolucional ha provocado que sea la mejora estructura a nivel general.

También se ha comprobado que las variables juegan un papel bastante importante respecto a las predicciones. Los indicadores técnicos no han mejorado de manera sustancial la predicción, pero sí que han hecho que el modelo sea superior en algunos casos y mucho más deficientes en otros. Por lo que la elección correcta de las variables puede provocar tener una predicción más que aceptable a una cercana a lo mediocre.

7. Líneas futuras

Tras la realización de este trabajo se puede establecer una guía de actuaciones futuras que permitan contribuir a mejorar en distintos aspectos los modelos y resultados propuestos, y así poder emplear los conocimientos obtenidos para aplicar a la vida real.

Poder cambiar la profundidad y el número de *units* de todas las capas de las arquitecturas propuestas fue una de las cosas que lamentablemente se tuvo que descartar por la cantidad de tiempo de ejecución que requería. Sin embargo, sería más que interesante poder cambiar estos parámetros y así encontrar la mejor estructura. Otro de los parámetros que no se cambiaron y que sería ideal cambiar sería el *batch size.* Este es un hiperparámetro que puede afectar también de gran manera al rendimiento de la red.

Aumentar el número de las variables empleadas que fueron descartadas por distintas razones, como índices de otros países o el precio del oro y del petróleo, podrían ser interesantes ver cómo afectan a las distintas redes. También sería valioso el poder probar de cambiar las variables juntándolas en parejas o tríos para comprobar si en algunos casos estos indicadores juntos son capaces de captar ciertas tendencias que individualmente no son capaces de interpretar.

Finalmente, otra línea de trabajo interesante sería aplicar estos modelos para empresas que cotizan en bolsa y, junto a un estudio y análisis técnico y fundamental, tratar de sacar beneficios invirtiendo con la mayor información posible.

Bibliografía

Akgün, E., & Demir, M. (2018). Modeling Course Achievements of Elementary Education Teacher Candidates with Artificial Neural Networks. *International Journal of Assessment Tools in Education*, *5*. https://doi.org/10.21449/ijate.444073

Amidi, A., & Amidi, S. (2018). *VIP Cheatsheet : Recurrent Neural Networks*. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Aparicio, A., Abia, R., Rodriguez–Piñero, T., & Alfaraz, J. A. (1999). Redes Neuronales y su aplicación predictiva en la Bolsa de Valores española. *Rect@*, *Actas_7*, 56–67.

Atsalakis, G., & Valavanis, K. (2009). Surveying stock market forecasting techniques - Part II: Soft computing methods. *Expert Syst. Appl.*, *36*, 5932–5941. https://doi.org/10.1016/j.eswa.2008.07.006

Bagnato, J. I. (2018). *Sets de Entrenamiento, Test y Validación*. https://www.aprendemachinelearning.com/sets-de-entrenamiento-test-validacion-cruzada/

Baomar, H., & Bentley, P. J. (2017). An Intelligent Autopilot System that learns flight emergency procedures by imitating human pilots. In *2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016*. https://doi.org/10.1109/SSCI.2016.7849881

Basu, S. (1977). Investment performance of common stocks in relation to their price-earnings ratios: a test of the efficient market hypothesis. *The Journal of Finance*, *32*(3), 663–682. https://doi.org/10.1111/j.1540-6261.1977.tb01979.x

Bernacki, M., & Włodarczyk, P. (2005). *Principles of training multi-layer neural network using backpropagation*. http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

BME, B. y mercados E. (2019). *Bolsas y Mercados Españoles (BME) calcula , publica y distribuye en tiempo real los índices IBEX . BME , la compañía que integra los principales mercados de valores y sistemas financieros de España , es el quinto mayor operador de mercados de Europa por*. http://www.bolsamadrid.es/docs/SBolsas/InformesSB/FS-Ibex35_ESP.pdf

Bonrostro, P., Joaquín, G., & Güemes, A. (1997). Aplicaciones de redes neuronales en economía. *Revista Electrónica de Comunicaciones y Trabajos de ASEPUMA*, 1–8.

Buchanan, B. G. (2005). A (Very) Brief History of Artifcial Intelligence. *AI Magazine*, *26*(4), 53–60. https://doi.org/10.1609/aimag.v26i4.1848

Chan, K. C., Gup, B. E., & Pan, M.-S. (1997). International stock market efficiency and integration: a study of eighteen nations. *Journal of Business Finance & Accounting*, *24*(6), 803–813. https://doi.org/10.1111/1468-5957.00134

Chavarnakul, T., & Enke, D. (2008). Intelligent technical analysis based equivolume charting for stock trading using neural networks. *Expert Systems with Applications*, *34*(2), 1004–1017. https://doi.org/10.1016/j.eswa.2006.10.028

Chen, A.-S. S., Leung, M. T., & Daouk, H. (2003). Application of neural networks to an emerging financial market: Forecasting and trading the Taiwan Stock Index. *Computers and*

Operations Research, *30*, 901–923. https://doi.org/10.1016/S0305-0548(02)00037-0

Chen, K., Zhou, Y., & Dai, F. (2015). A LSTM-based method for stock returns prediction: A case study of China stock market. In *2015 IEEE International Conference on Big Data (Big Data)* (pp. 2823–2824). IEEE. https://doi.org/10.1109/BigData.2015.7364089

Cruz, T., Arturo, E., Restrepo, H., & Varela, M. (2009). Pronostico del índice general de la bolsa de valores de Colombia usando redes neuronales. *Scientia et Technica*, *1*(41), 129–134. https://doi.org/10.22517/23447214.2889

De Oliveira, F. A., Nobre, C. N., & Zárate, L. E. (2013). Applying Artificial Neural Networks to prediction of stock price and improvement of the directional prediction index - Case study of PETR4, Petrobras, Brazil. *Expert Systems with Applications*, *40*(18), 7596–7606. https://doi.org/10.1016/j.eswa.2013.06.071

Domingo, R. (2017, December 18). *¿Qué es el IBEX 35? ¿Qué empresas lo forman?* Blog. https://www.zaplo.es/blog/que-es-el-ibex-35/

El-Henawy, I. M., Kamal, A. H., Abdelbary, H. A., & Abas, A. R. (2010). Predicting stock index using neural network combined with evolutionary computation methods. *INFOS2010 - 2010 7th International Conference on Informatics and Systems*, 1–6.

El PAIS. (2011). El Ibex 35 tendrá temporalmente 36 valores. *12 Diciembre*. https://elpais.com/economia/2011/12/12/actualidad/1323678787_850215.html

Enke, D., & Thawornwong, S. (2005). The use of data mining and neural networks for forecasting stock market returns. *Expert Systems with Applications*, *29*(4, November 2005), 927–940. https://doi.org/10.1016/j.eswa.2005.06.024

Fernández-Rodríguez, F., González-Martel, C., & Sosvilla-Rivero, S. (2000). On the profitability of technical trading rules based on artificial neural networks: Evidence from the Madrid stock market. *Economics Letters*. https://doi.org/10.1016/S0165-1765(00)00270-6

Frederickson, B. (2016). *An Interactive Tutorial on Numerical Optimization*. http://www.benfrederickson.com/numerical-optimization/

Garces, A., Toro, E., Molina-Cabrera, A., Mirledy, E., Ocampo, T., Cabrera, A. M., Ruiz, A. G., Garces, A., Toro, E., & Molina-Cabrera, A. (2006). Pronóstico de bolsa de valores empleando técnicas inteligentes. *Tecnura*, *9*, 57–66. https://doi.org/10.14483/22487638.6234

García, F., Guijarro, F., Oliver, J., & Tamošiūnienė, R. (2018). Hybrid fuzzy neural network to predict price direction in the German DAX-30 index. *Technological and Economic Development of Economy*. https://doi.org/10.3846/tede.2018.6394

García, M. C., Jalal, A. M., Garzón, L. A., & López, J. M. (2013). Métodos para predecir índices Bursátiles. *Ecos de Economía*, *17*(37), 51–82. https://doi.org/10.17230/ecos.2013.37.3

Göçken, M., Özçalici, M., Boru, A., & Dosdoɨru, A. T. (2016). Integrating metaheuristics and Artificial Neural Networks for improved stock price prediction. *Expert Systems with Applications*, *44*, 320–331. https://doi.org/10.1016/j.eswa.2015.09.029

González, N. (2019). *How Deep Learning is Different from Machine Learning?* https://noeliagorod.com/2019/11/14/how-deep-learning-is-different-from-machine-learning/

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8),

1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hochreiter, S., Schmidhuber, J., & Urgen Schmidhuber, J. J. (1997). Long Short-Term Memory. *MEMORY Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Huang, C. Y., & Lin, P. K. P. (2014). Application of integrated data mining techniques in stock market forecasting. *Cogent Economics and Finance*, *2*(1), 1–18. https://doi.org/10.1080/23322039.2014.929505

INTEL. (2019). *The Future of AI in Agriculture - Intel*. https://www.intel.com/content/www/us/en/big-data/article/agriculture-harvests-big-data.html

Kaufman, P. J. (2005). *New trading systems and methods*. John Wiley & Sons, Inc.

Komo, D., Chang, C. I., Ko, H., Kong, H., Network, N., For, T., Market, S., Prediction, I., Komo, D., Chang, C. I., & Ko, H. (1994). Neural Network Technology For Stock Market Index Prediction. *ISSIPNN 1994 - 1994 International Symposium on Speech, Image Processing and Neural Networks, Proceedings*, *April*, 543–546. https://doi.org/10.1109/SIPNN.1994.344854

Kuhn, M., & Johnson, K. (2013). Applied predictive modeling. In *Applied Predictive Modeling* (pp. 27–59, 61–92). https://doi.org/10.1007/978-1-4614-6849-3

Kumba, S., & Sennaar, K. (2019). *AI in Agriculture – Present Applications and Impact | Emerj*. Emerj. https://emerj.com/ai-sector-overviews/ai-agriculture-present-applications-impact/

Ligeza, A. (1995). Artificial Intelligence: A Modern Approach. *Neurocomputing*, *9*(2), 215–218. https://doi.org/10.1016/0925-2312(95)90020-9

Malkiel, B. G. (2003). The Efficient Market Hypothesis and Its Critics. *Journal of Economic Perspectives*, *17*(1), 59–82. https://doi.org/10.1257/089533003321164958

Moreno, A. (2020). *1-d Convolutional Neural Networks for Time Series: Basic Intuition*. BoostedML. https://boostedml.com/2020/04/1-d-convolutional-neural-networks-for-time-series-basic-intuition.html

Nelson, D. M. Q., Pereira, A. C. M., & De Oliveira, R. A. (2017). Stock market's price movement prediction with LSTM neural networks. *Proceedings of the International Joint Conference on Neural Networks*, *2017-May*(Dcc), 1419–1426. https://doi.org/10.1109/IJCNN.2017.7966019

Olah, C. (2015, August 27). *Understanding LSTM Networks*. Colah's Blog. http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock market index using fusion of machine learning techniques. *Expert Systems with Applications*, *42*(4), 2162–2172. https://doi.org/10.1016/j.eswa.2014.10.031

Pérez-Rodríguez, J. V., Torra, S., & Andrada-Félix, J. (2005). STAR and ANN models: Forecasting performance on the Spanish "Ibex-35" stock index. *Journal of Empirical Finance*, *12*(3), 490–509. https://doi.org/10.1016/j.jempfin.2004.03.001

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Santana, C., & Dot CSV. (2018, October 3). *¿Qué es una Red Neuronal? Parte 3 : Backpropagation | DotCSV*. Archivo de Video. https://www.youtube.com/watch?v=eNIqz_noix8&

Schweikhard, K. a., Richards, W. L., Theisen, J., Mouyos, W., & Garbos, R. (2001). *Flight Demonstration of X-33 Vehicle Health Management System Components on the F/A-18 Systems Research Aircraft* (Issue December). https://ntrs.nasa.gov/search.jsp?R=20010055588

Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017). Stock price prediction using LSTM, RNN and CNN-sliding window model. *International Conference on Advances in Computing, Communications and Informatics, ICACCI 2017*, *2017-Janua*, 1643–1647. https://doi.org/10.1109/ICACCI.2017.8126078

Sermpinis, G., Karathanasopoulos, A., Rosillo, R., & de la Fuente, D. (2019). Neural networks in financial trading. *Annals of Operations Research*. https://doi.org/10.1007/s10479-019-03144-y

The Medical Futurist. (2018). *Artificial Intelligence Will Redesign Healthcare - The Medical Futurist*. The Medical Futurist. http://medicalfuturist.com/artificial-intelligence-will-redesign-healthcare/

Timmermann, A., & Granger, C. W. J. (2004). Efficient market hypothesis and forecasting. *International Journal of Forecasting*, *20*(1), 15–27. https://doi.org/https://doi.org/10.1016/S0169-2070(03)00012-8

Vanstone, B., & Finnie, G. (2009). An empirical methodology for developing stockmarket trading systems using artificial neural networks. *Expert Systems with Applications*, *36*(3 PART 2), 6668–6680. https://doi.org/10.1016/j.eswa.2008.08.019

Villada, F., Muñoz, N., & García, E. (2012). Aplicación de las Redes Neuronales al Pronóstico de Precios en el Mercado de Valores. *Informacion Tecnologica*, *23*(4), 11–20. https://doi.org/10.4067/S0718-07642012000400003

Voegt, T. (2017). *Artificial Neural Networks in Trading Systems* (Issue 1).

Wu, B., & Duan, T. (2017). A performance comparison of neural networks in forecasting stock price trend. *International Journal of Computational Intelligence Systems*, *10*(1), 336–346. https://doi.org/10.2991/ijcis.2017.10.1.23

Yao, J., Tan, C. L., & Poh, H.-L. (1999). Neural networks for technical analysis: a study on KLCI. *International Journal of Theoretical and Applied Finance*, *02*(02). https://doi.org/10.1142/s0219024999000145

Zhang, G. P. (2012). Neural Networks for Time-Series Forecasting. In G. Rozenberg, T. Bäck, & J. N. Kok (Eds.), *Handbook of Natural Computing* (pp. 461–477). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-92910-9_14

Zhang, L., Aggarwal, C., & Qi, G. J. (2017). Stock price prediction via discovering multi-frequency trading patterns. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *Part F1296*, 2141–2149. https://doi.org/10.1145/3097983.3098117