# LSTM y Random Forest aplicados a la predicción de stocks

Julián Toledano Díaz

Julio 2018

# Índice

# 1. Resumen

Las técnicas de Inteligencia Artificial son aplicadas hoy día a la resolución de numerosos problemas en diversos campos de la actividad humana. Las empresas intentan extraer la máxima información de los grandes volúmenes de datos que manejan y recopilan diariamente y tratan de anticipar decisiones de la forma más rápida y eficiente posible dado que una buena parte de sus resultados o equilibrios financieros pueden depender de ello.

El presente trabajo se enfoca al análisis de aplicabilidad de algunas técnicas de Inteligencia Artificial para predecir el valor en bolsa de los activos de grandes empresas. Concretamente, se trata de estudiar si es posible la predicción de los valores en bolsa de las acciones de dos gigantes tecnológicos, Apple y Google.

En el trabajo se estudia la aplicabilidad de algunas de las técnicas de redes neuronales recurrentes (LSTM) y de árboles de decisión (Random Forest), utilizadas con éxito en el campo de la predicción de stocks, al objetivo establecido. Se explican las principales característica de ambas técnicas, se profundiza en sus problemáticas de aplicación y, tomando como referencia trabajos previos, se adaptan sus planteamientos de base para intentar predecir el valor que tomarán en bolsa las acciones de las empresas analizadas. Finalmente se detallan los resultados obtenidos y se relacionan las pertinentes conclusiones.

# 2. Introducción

Podríamos definir teóricamente el mercado como aquel lugar en que se encuentran la oferta y demanda de productos y servicios. El resultado de ese encuentro es la determinación del precio, que es, por definición, el valor que iguala oferta y demanda. Así pues, podríamos decir que los mercados son fascinantes, sobreentendiéndose que en los mercados el valor de los activos varía constantemente y, en principio, de forma aleatoria. Sin embargo, realmente el precio no hace nada más que ajustarse de forma constante a la nueva información que llega al mercado de forma continua. Ahora bien, partiendo de tal situación, ¿puede ser posible anticipar la variación del precio?.

Si se estudia un stock minuciosamente pueden llegar a descubrirse patrones ocultos que se repiten. ¿Es factible establecer criterios de análisis de los stocks que puedan servirnos para predecir un movimiento futuro, ya sea de subida o de bajada?. A este tipo de estudio se le llama análisis técnico y utiliza diferentes indicadores, basándose en datos del pasado, para predecir el futuro. Los diferentes indicadores se dividen en distintas categorías según su uso. Así, entre otros, citaremos:

Indicadores de superposición: Bollinger Bands, Double Exponential Moving Average, Simple Moving Average, Weighted Moving Average.

Indicadores de momento: Commodity Channel Index, Relative Strength Index.

Indicadores de volumen: Chaikin A/D Oscillator, Chaikin A/D Line.

Indicadores de volatilidad: Average True Range, Normalized Average True Range.

Aunque estos indicadores pueden ser de gran ayuda a la hora de tomar decisiones, todos ellos tienen en común que siempre utilizan valores ya pasados por lo que poseen una gran dependencia de éstos y un retraso frente al presente. Por este motivo son de limitada fiabilidad y nunca pueden ser tomados como ciertos, únicamente nos darán una idea aproximada de cómo puede evolucionar un valor a corto, medio o largo plazo (los indicadores técnicos utilizados en el presente trabajo pueden verse, con más detalle, en el epígrafe 2.3 y en la Tabla 4).

Por la naturaleza aleatoria e impredecible de la variación de los precios de los activos, es común que inversores e investigadores estudien maneras de predecir el comportamiento

de éstos mediante diversas técnicas. Las más antiguas se basan en los indicadores ya mencionados. Combinando el estudio de varios de estos indicadores conjuntamente sobre un único valor se pueden llegar a crear alertas de compra o venta. Este tipo de análisis a partir de indicadores se hicieron realmente populares a finales de los ochenta y principios de los noventa, período en el que cualquiera podía escribir un programa de compraventa gracias a la expansión de los ordenadores personales. Durante esa época se produjo el cambio de las antiguas plazas de bolsa, en las que las compraventas se hacían de manera presencial, a las actuales, en las que se opera mayoritaria y casi exclusivamente a través de un ordenador. Por ese motivo desde mediados de los noventa y principios del nuevo milenio las grandes empresas desarrollaron potentes programas de inversión automática. Estos programas capaces de realizar cientos de operaciones por segundo, negociación de alta frecuencia, dejan poco espacio para pequeños inversores que deseen operar personalmente u obtener una rentabilidad extra de forma individualizada.

La inteligencia artificial ha tenido un gran auge en los últimos años, muchos optimistas creen que esta tecnología se puede aplicar a cualquier campo y obtener mejores resultados que con las técnicas tradicionales propias. Automatización, publicidad, asistentes personales, entre otros muchos campos de actividad, son algunos de los terrenos donde se ha utilizado. Obviamente, los mercados financieros no pueden quedar fuera de esta ola de optimismo y existen infinidad de empresas que venden una elevada rentabilidad gracias a su software de compraventa basado en inteligencia artificial. Existen a su vez una gran cantidad de estudios, con mejores o peores resultados, donde se aplican técnicas de inteligencia artificial y deep learning al mundo de la bolsa.

En este trabajo se utilizarán, a tal efecto, dos técnicas muy populares en los últimos años, las LSTM —Long Short-Term Memory—[14] y los Random Forest[6]. Ambas se explicarán con detalle más adelante. El motivo por el que se han escogido estos dos algoritmos es porque se han aplicado con éxito en el campo de la predicción de stocks. Aunque se han aplicado con éxito tanto las LSTM[12] como los Random Forest[10], en ambos casos resuelven el problema planteando éste como un ejercicio de clasificación. Esto es, si sube una acción la salida del algoritmo será 1 y si baja será 0. El objetivo de este

trabajo es ir un paso más allá y resolver la cuestión planteándola como un problema de regresión. Es decir, se intentará predecir un valor continuo que, en este caso, es el precio de cierre del día siguiente de una determinado acción (valor en bolsa). En este trabajo se van a realizar pruebas sobre los precios de Apple y Google.

## 2.1. Redes de Neuronas Recurrentes

En una red de neuronas tradicional —Feedforward Neural Network— se asume que todas las entradas y salidas son independientes entre si. Las redes de neuronas recurrentes, RNN —Recurrent Neural Networks, por sus siglas en inglés-, por el contrario han sido ideadas para hacer uso de información secuencial. Las RNN son llamadas recurrentes porque realizan la misma tarea para cada elemento dentro de una secuencia, siendo su salida dependiente de cálculos previos. En concreto la salida de su capa oculta $S_t$ es reintroducida como un parámetro de entrada como muestra la figura 1.

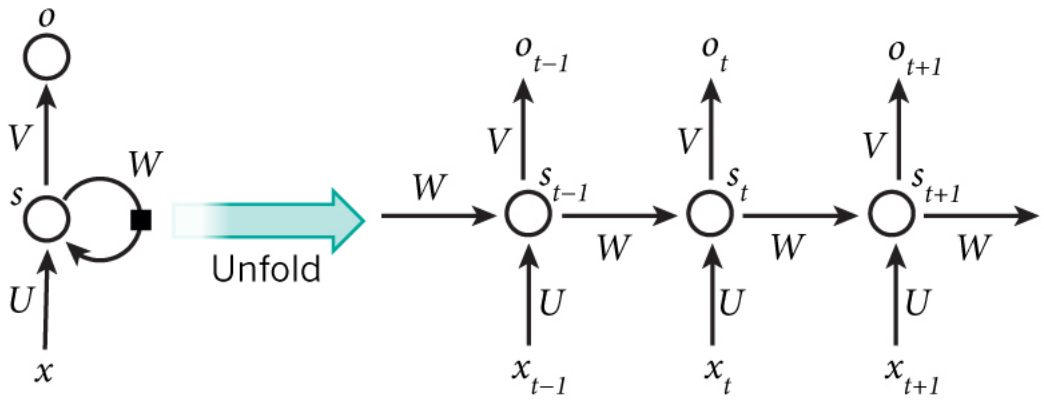



Figura 1: Una RNN y el despliegue en el tiempo de sus cálculos.

En teoría las redes de neuronas recurrentes pueden hacer uso de secuencias de longitud arbitraria, pero en la práctica están limitadas a mirar solo unos pocos pasos más atrás por el problema del desvanecimiento de gradiente [13], tal y como se comenta a continuación.

### 2.1.1. Desvanecimiento de Gradiente

Este problema surge por la propia naturaleza del algoritmo de back-propagation utilizado para optimizar las redes de neuronas. Durante el entrenamiento de un red neuronal sus pesos son actualizados respecto a la función de pérdida de la siguiente manera:

$$\frac{\partial C}{\partial W_l} \tag{1}$$

Donde $W_l$ representa los pesos de la capa $l$ y $C$ la función de coste o pérdida en la capa de salida.

En la última capa este cálculo es sencillo, sin embargo, en capas anteriores, es necesario utilizar el algoritmo de back-propagation. En la capa final, el término de error se expresa:

$$\delta_i^{(n_l)} = -(y_i - h_i^{(n_l)}) \cdot f'(z_i^{(n_l)}) \tag{2}$$

Esta función $\delta_i^{(n_l)}$ demuestra que el error está relacionado con la diferencia entre la salida de la red $h_i^{(n_l)}$ y la salida esperada $y_i$. Es además proporcional a la derivada de la función de activación $f'(z_i^{(n_l)})$. Los pesos en la última capa cambian directamente en proporción al valor de $\delta$. Para capas anteriores, el error de su capa posterior a de ser propagado. Asimismo, esta propagación se realiza con la siguiente ecuación:

$$\delta^{(l)} = ((W^{(l)})^T \delta^{l+1}) \cdot f'(z^{(l)}) \tag{3}$$

De nuevo esta vez, se encuentra la derivada de la función de activación $f'(z_i^{(n_l)})$. La diferencia está en que $\delta^{(l)}$ es proporcional al error propagado por la capa posterior $\delta^{l+1}$. Esta cadena de valores de $\delta$ incluyen su propia derivada de la función de activación; de esta forma, el gradiente de los pesos en una capa determinada respecto a la función de pérdida, que controla cómo estos son actualizados, es proporcional a la multiplicación de

las derivadas de la función de activación. Así:

$$\frac{\partial C}{\partial W_l} \propto f'(z^l)f'(z^{l+1})f'(z^{l+2})...f'(z^{l+n}) \tag{4}$$

El problema del desvanecimiento del gradiente llega cuando los términos $f'$ son valores menores que uno. Si se multiplican estos valores, se termina con un desvanecimiento, que acaba con un valor muy bajo de $\frac{\partial C}{\partial W_l}$ y por lo tanto la red no aprende nada.

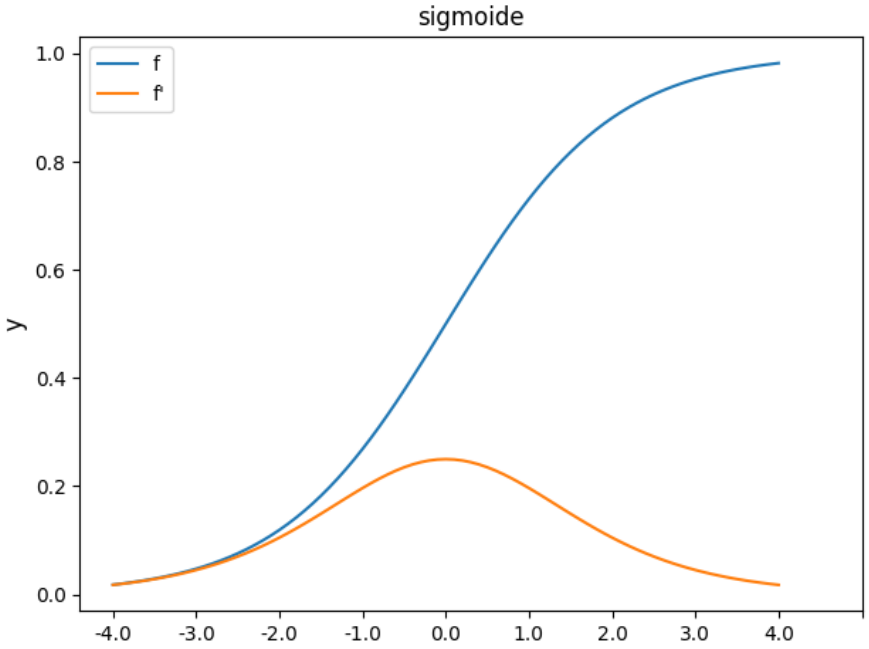
En la figura 2 se puede observar, tanto en la función sigmoide $\sigma = \frac{e^x}{e^x+1}$ como en la tangente hiperbólica $\tanh x = \frac{e^x-e^{-x}}{e^x+e^{-x}}$ que cuando toman valores muy grandes o muy pequeños, la derivada tiende a cero. Esto causa resultados de aprendizaje muy pobres. Incluso en una red de neuronas con cuatro capas, donde la derivada de su función de activación está en torno al máximo 0,2, el resultado del producto de sus derivadas $0{,}2^4 = 0{,}0016$ es demasiado pequeño.

### 2.1.2. LSTM

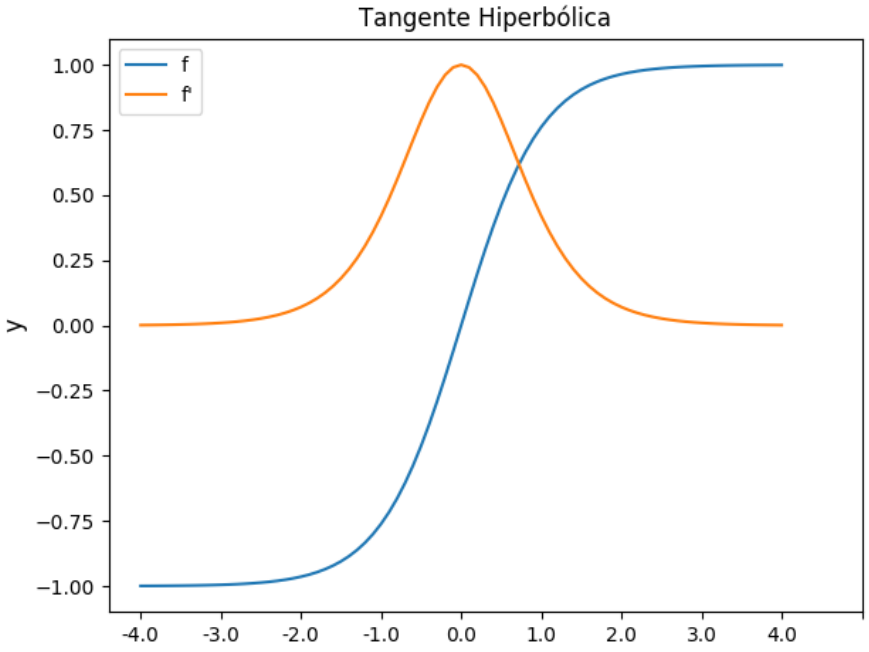
Las LSTM —Long Short Term Memory— son un tipo de redes de neuronas recurrentes que resuelven el problema del desvanecimiento de gradiente. Para ello este tipo de redes introducen una "memoria". La figura 3 muestra el diagrama de la estructura de una unidad de LSTM.

Esta unidad tiene tres entradas: $X_t$ es la entrada de los datos de este momento, $h_{t-1}$ y $C_{t-1}$ son la salida y la "memoria"de la unidad previa, respectivamente. Además, esta unidad posee dos salidas: $h_t$ es el resultado obtenido y $C_t$ es la memoria de esta unidad. De esta manera se toman decisiones considerando la entrada de los datos actuales, la salida previa y la "memoria", generando una salida y alterando también la memoria de la LSTM. Asímismo, la figura 4 muestra una LSTM desglosada en el tiempo.

La manera en que la memoria $C_t$ cambia depende principalmente de dos operaciones llamadas puerta de olvido y puerta de nueva memoria. La figura 5 muestra el flujo de memoria dentro de la unidad. La entrada es la memoria de la unidad anterior $C_{t-1}$ y la primera operación $\times$ es la puerta de olvido. Esta operación es una multiplicación de




(a) Comparación de la función sigmoide y su derivada.




(b) Comparación de la función tang. hiperbólica y su derivada.

Figura 2: Comparación de las funciones sigmoide y tangente hiperbólica con sus derivadas.

element-wise. De esta manera si se multiplican $C_{t-1}$ y un vector cercano a 0 significa que se olvida casi toda la memoria antigua; por el contrario se mantiene la memoria si se

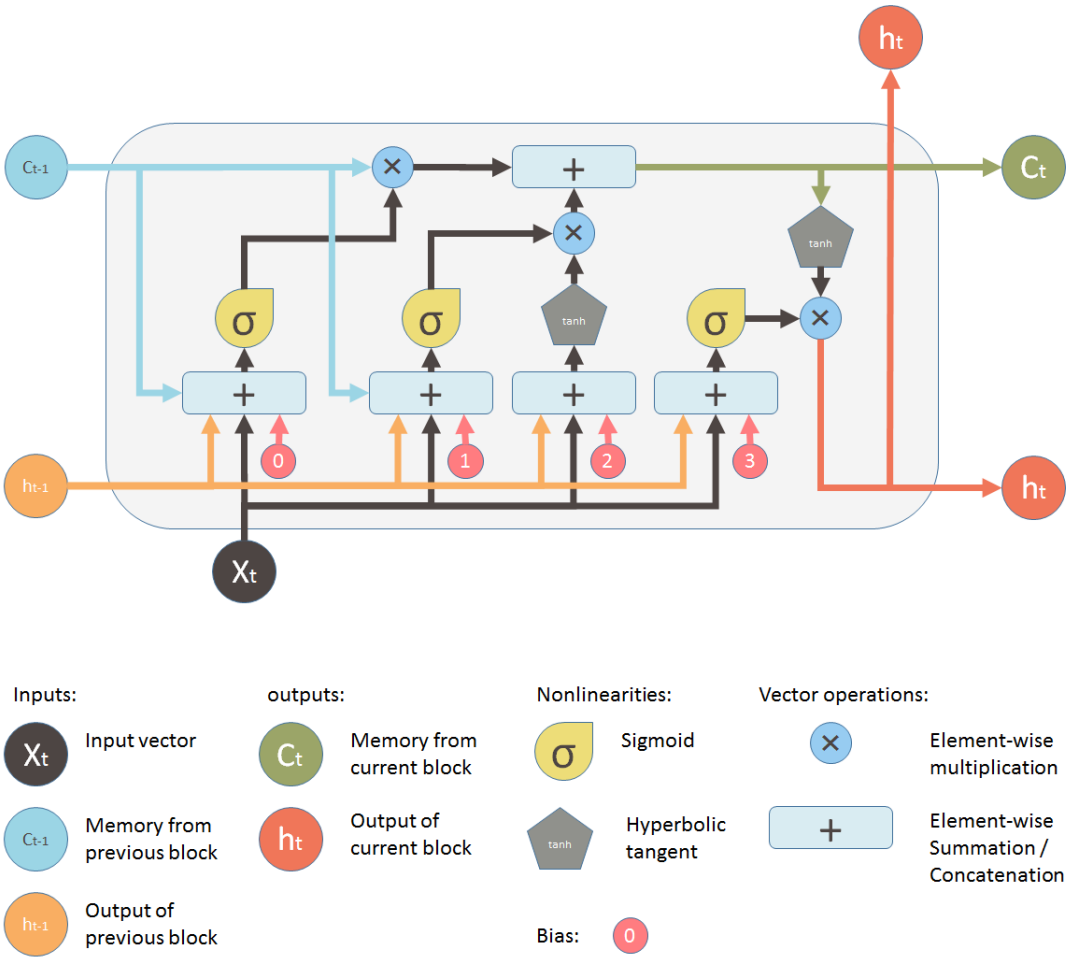



Figura 3: Unidad de una LSTM.[15]

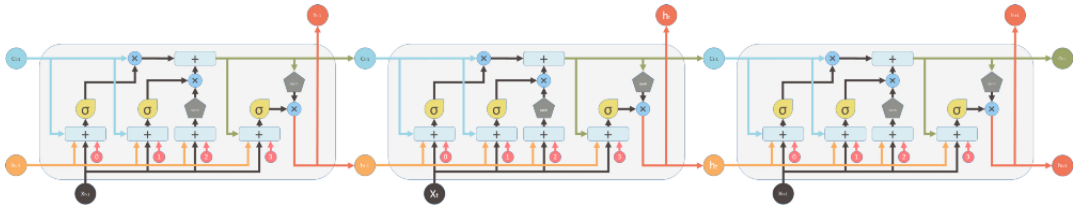


Figura 4: LSTM en el tiempo.[15]

multiplica por un vector cercano a 1. La siguiente operación en el flujo de memoria es el operador +, que calcula una suma de piece-wise. La nueva memoria y la memoria antigua se funden con esta operación; cuánta nueva memoria es añadida a la vieja se controla con la operación × que está justo debajo. Después de estas dos operaciones la nueva memoria $C_t$ ya ha sido calculada.

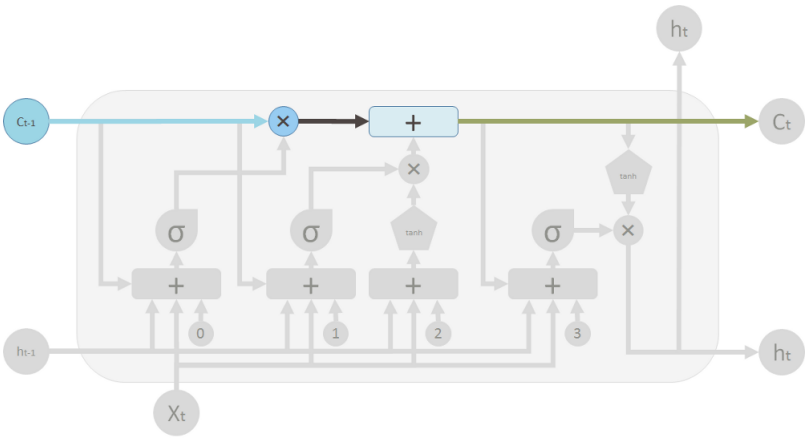



Figura 5: Flujo de memoria.[15]

La figura 6 muestra cómo se calcula el vector por el que se multiplica $C_{t-1}$ en la puerta de olvido. Este vector es la salida de una red de neuronas simple de una sola capa que tiene como entrada $h_{t-1}$, la salida del bloque previo, $X_t$, los datos de entrada en este momento, $C_{t-1}$, la memoria del bloque previo y $b_0$ un vector de bias. Esta red tiene como función de activación a la función sigmoide.

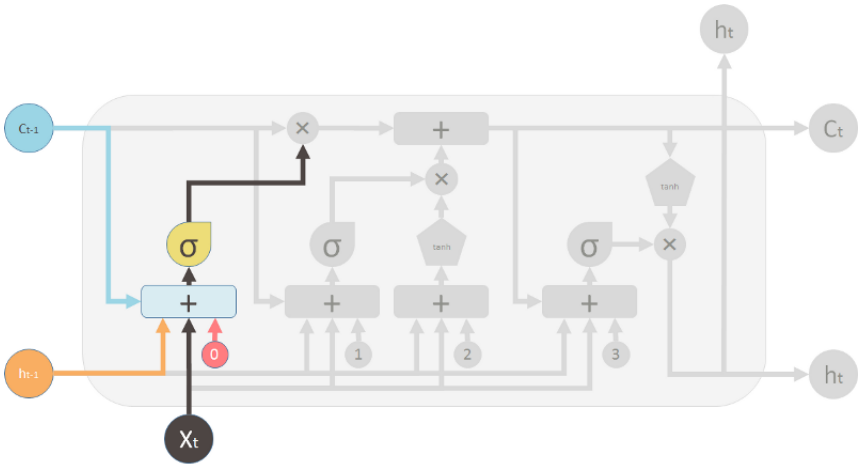



Figura 6: Cálculo del vector de olvido.[15]

En la figura 7 se muestra cómo se calcula el vector de nueva memoria. En esta ocasión también se trata de una red de neuronas simple que toma las mismas entradas que la anterior. Esta red controla cuánto debe influir la nueva memoria a la vieja. La nueva memoria es calculada con otra red simple, pero en esta ocasión no toma la antigua

memoria por entrada y utiliza una función tangente hiperbólica como activación. Las salidas de estas dos operaciones son multiplicadas por una operación de element-wise y añadidas a la antigua memoria para formar la nueva. La figura 8 esquematiza las puertas de olvido y nueva memoria.

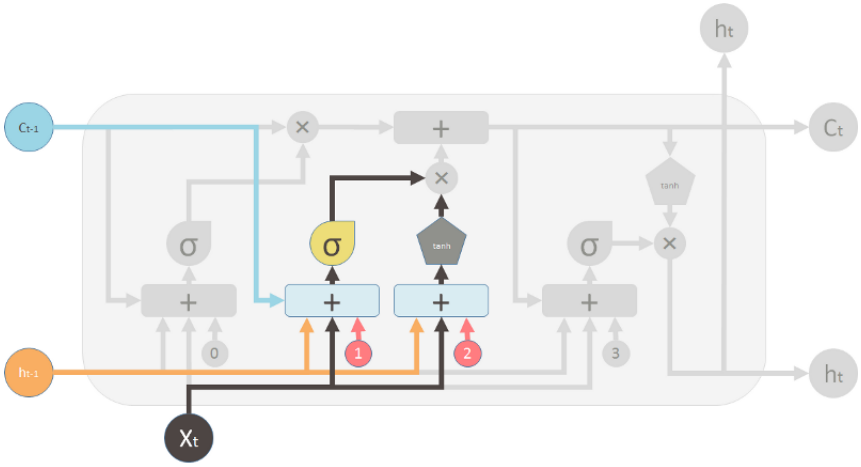



Figura 7: Cálculo de la nueva memoria.[15]

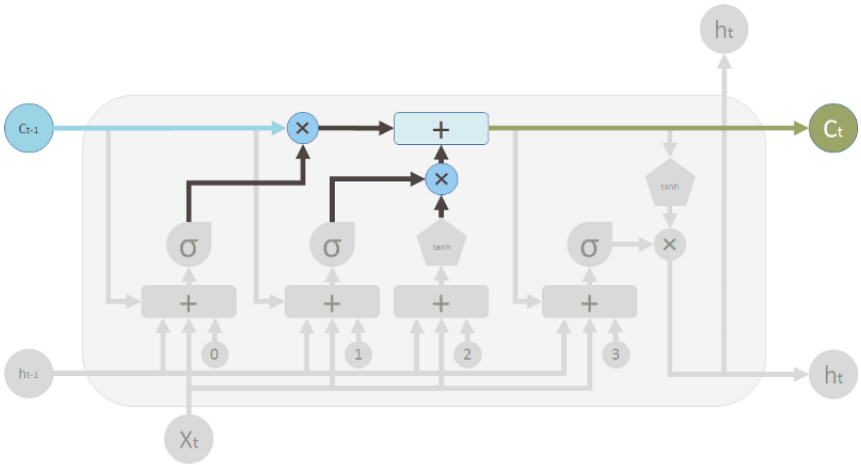



Figura 8: Puertas de olvido y de nueva memoria.[15]

Por último la figura 9 muestra cómo se genera la salida. Así, $h_t$ es calculado con la entrada de los datos en este momento, $X_t$, la salida del bloque anterior, $h_{t-1}$, el cálculo de la nueva memoria y un vector de bias $b_3$. La salida se obtiene, por tanto, tras realizar una multiplicación de element-wise a la salida de una red simple que toma los datos antes mencionados por la tangente hiperbólica de $c_t$.

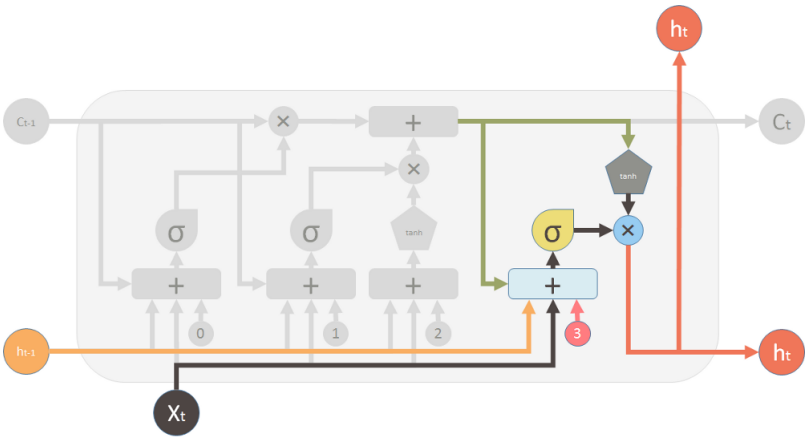



Figura 9: Salida de la LSMT.[15]

En los diagramas anteriores se muestra la memoria *C* como una entrada de la red. Sin embargo, la mayoría de la literatura trata a esta memoria como algo interno en la red. La figura 10 representa la LSTM con la memoria interna y la figura 11 el proceso descrito anteriormente paso por paso.

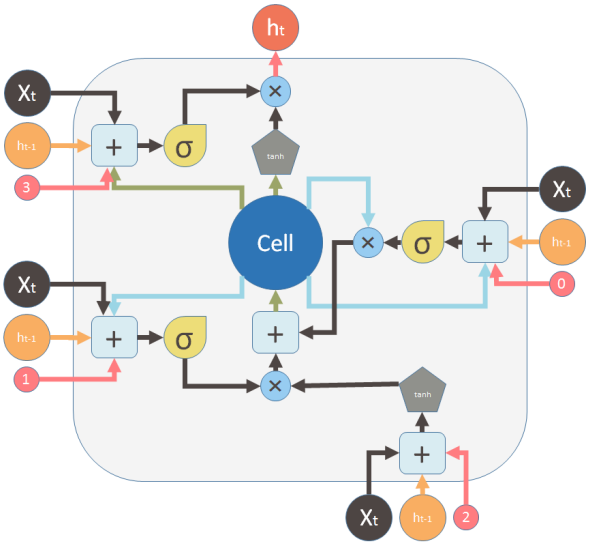
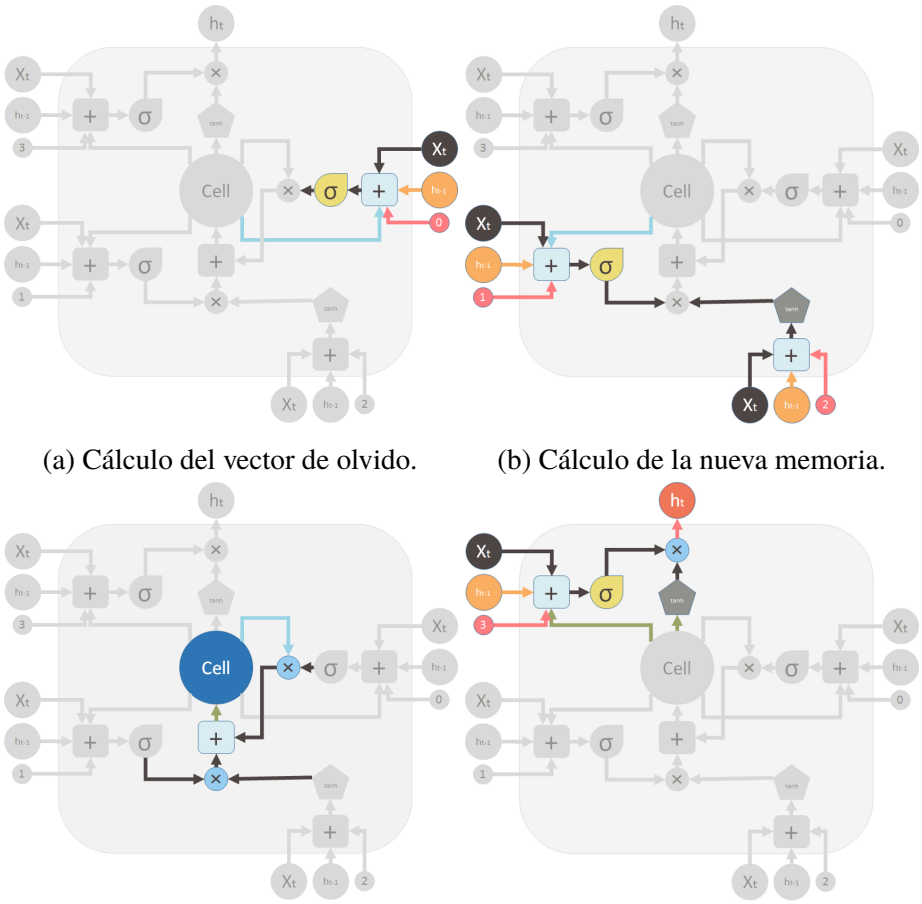



Figura 10: LSTM con la memoria interna.[15]

## 2.2. Árboles de Decisión

Un árbol de decisión es un sistema de clasificación o regresión de datos que representa el conocimiento en forma de árbol. Rompe un conjunto de datos en subconjuntos cada vez




(a) Cálculo del vector de olvido.

(b) Cálculo de la nueva memoria.

(c) Puertas de olvido y nueva memoria.

(d) Salida de la LSTM.

Figura 11: Fases de una LSTM con la memoria interna.[15]

más pequeños mientras que, al mismo tiempo, se desarrolla incrementalmente un árbol de decisión asociado. El resultado final es un árbol con nodos de decisión y nodos hoja o terminales. Un nodo de decisión posee dos o más hijos, cada uno de los cuales representa valores para el atributo probado, mientras que cada nodo terminal representa un valor de retorno, esto es uno de los posibles resultados que el árbol puede devolver.

El algoritmo para construir árboles de decisión se llama ID3. Creado por J. R. Quinlan emplea una búsqueda codiciosa de arriba hacia abajo a través del espacio de posibles ramas sin retroceso. El algoritmo comienza con el conjunto $S$ como el nodo raíz; en cada iteración comprueba cada atributo no usado de $S$ y calcula su entropía o ganancia de información; después selecciona el atributo con menor entropía o mayor ganancia de información; el conjunto $S$ es entonces dividido por el atributo seleccionado para producir subconjuntos

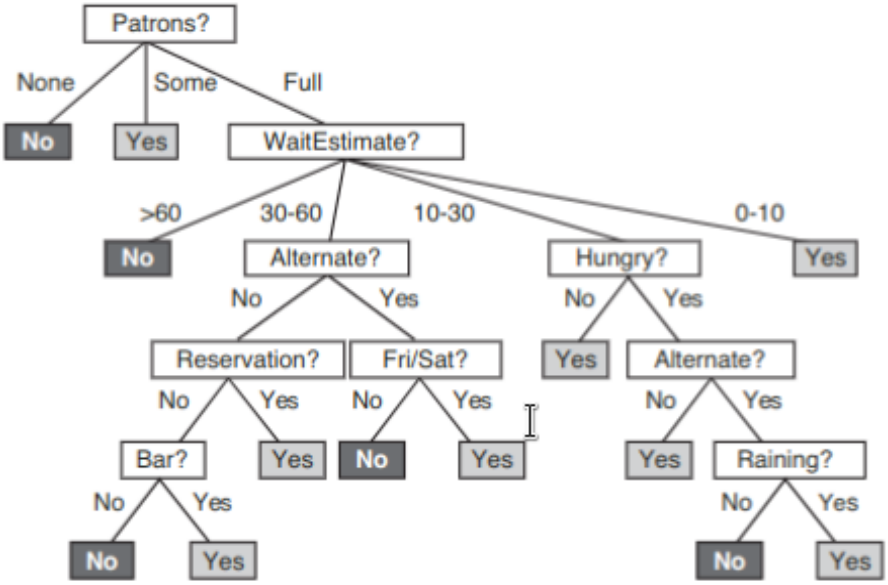



Figura 12: Árbol de decisión [17]

(p. ej. tiempo de espera en un restaurante, menor que 10, entre 10 y 30, entre 30 y 60, o más de 60 minutos; figura 12). El algoritmo continúa iterando tomando atributos que no han sido considerados anteriormente.

El árbol es construido de arriba hacia abajo desde el nodo raíz e incluye particionar los datos en subconjuntos que contienen instancias con valores similares. El algoritmo ID3 puede ser usado para construir un árbol que resuelva un problema regresión. Para calcular la homogeneidad de conjuntos numéricos se utiliza la derivación estándar. Si el conjunto es completamente homogéneo su derivada estándar será cero.

$$\begin{aligned} Count &= n \\ Average = a &= \frac{\sum x}{n} \\ StandardDerivation = S &= \sqrt{\frac{\sum (x-a)^2}{n}} \\ Coefficient of Variation = CV &= \frac{S}{a} * 100 \end{aligned} \tag{5}$$

La reducción de desviación estándar se basa en la disminución de la desviación estándar

después de que un conjunto de datos se divide en un atributo.

### 2.2.1. Random Forest

Random Forest es un algoritmo de aprendizaje supervisado. Como dice su nombre crea un bosque que, de alguna manera, es aleatorio. Ese bosque es un conjunto de árboles de decisión, la mayor parte de las veces entrenados con un método ”bagging” [5]. La idea general de este método es que una combinación de modelos de aprendizaje mejora sustancialmente el resultado final. Una de las ventajas de los random forest es que, al carecer de funciones de activación, pueden ser usados tanto para problemas de clasificación como de regresión sin necesidad de normalizar los datos.

Los random forest crean una cantidad determinada de árboles de decisión. La aleatoriedad es introducida en el modelo mientras se divide un nodo. En vez de buscar la mejor característica global, busca la mejor dentro de un conjunto aleatorio. Este proceso crea una gran diversidad generalmente produciendo mejores resultados.

La gran diferencia entra los árboles de decisión y los random forest es que los árboles de decisión formularán una serie de reglas que resultarán en una predicción. El algoritmo de random forest selecciona aleatoriamente observaciones y características para generar un número de árboles de decisión y luego promediar los resultados.

## 2.3. Indicadores Técnicos

Los datos de la bolsa que aquí se utilizan están representados por cinco columnas de datos con valores numéricos y cada fila representa a un día. La primera columna representa el precio de apertura, la segunda el máximo a lo largo de ese día, la tercera el mínimo de ese día, la cuarta el precio de cierre y la última el volumen.

Los indicadores técnicos son cálculos matemáticos que pueden ser aplicados a los datos pasados de diversos atributos de un valor de la bolsa (tales como el volumen y/o precio) o incluso sobre otros indicadores técnicos. Estos indicadores no analizan ninguno de los fundamentales de la empresa y, en la práctica, son principalmente utilizados para analizar movimientos a corto o medio plazo. El resultado de estos indicadores es usado

para alertar, predecir o confirmar. Existen diversos tipos de indicadores técnicos tal y como ya se ha mencionado al inicio del presente trabajo.

Los indicadores de tendencia miden la dirección y fuerza de una tendencia utilizando algún tipo de precio medio en su proceso de cálculo (Moving averages, MACD, Parabolic SAR, etc.); cuando el precio cruza la línea media se pueden identificar tendencias.

Los indicadores de momento (Commodity ChanelIndex, Relative strenght Index, etc.) ayudan a identificar la velocidad del movimiento del precio comparando el precio a lo largo del tiempo; también pueden ser usados para analizar el volumen. Se calculan comparando el precio de cierre con precios de cierre anteriores, cuando existe una divergencia entre el precio y un indicador de momento puede significar un cambio en los precios futuros.

Los indicadores de volatilidad (Average True Range, Normalized Average True Range) miden la variación del precio independientemente de la dirección; generalmente basados en el cambio entre el máximo y el mínimo de los precios históricos proporcionan información útil sobre el rango de compra venta.

Los indicadores de volumen (Chaikin A/D Oscillator, Chaikin A/D Line) miden la fuerza de una tendencia o confirman una dirección basándose en algún tipo de media sobre el volumen.

# 3. Predicción de stocks utilizando LSTM

Gracias a su capacidad de recordar, las LSTM han sido objeto de estudio a la hora de resolver problemas con series temporales[1][2][7]. Algunos estudios comparan a estas redes recurrentes con otro tipo de algoritmos obteniendo mejores resultados[9][16].

La complejidad del problema que se quiere resolver aquí reside en el tipo de datos. Intentar predecir un valor continuo, problema de regresión, con un conjunto de datos que posee gran cantidad de ruido y que ha de ser reducido. Aun así, el mayor problema se debe a la naturaleza de estos datos, ya que debido a sus condiciónes de aleatoriedad son difíciles de modelar. A este tipo de datos se les llama no estacionarios y uno de sus mayores inconvenientes es el de la normalización.

## 3.1. Trabajo previo

Para resolver el problema se toman como referencia varios trabajos previos, intentando reproducir sus resultados o adaptando su planteamiento de clasificación a uno de regresión.

En el estudio que se indica, [12], se intenta predecir tendencias de subida o bajada en los próximos quince minutos calculando para ello más de 175 indicadores técnicos que son usados como entrada de la red; después de calcular esas señales, con los datos se realizan una serie de operaciones, primero un alisamiento exponencial (exponential smoothing) (ecuación 6) que permite que los datos antiguos reciban progresivamente menos peso relativo mientras los datos más nuevos obtienen progresivamente mayor peso. Donde $X_i$ es el valor real en $t$ que se quiere alisar y $Z_{i-1}$ es el valor alisado en $t_{-1}$. Así, $\alpha$ es un valor comprendido entre (0, 1), cuanto más bajo es el valor de $\alpha$ menos valor tendrán los datos antiguos. Esto se puede observar en la figura 13.

$$z_i = \alpha x_i + (1 - \alpha) z_{i-1} \tag{6}$$

Seguidamente se efectúa una transformación logarítmica.

$$\log(p_i) - \log(p_{i-1}) \tag{7}$$

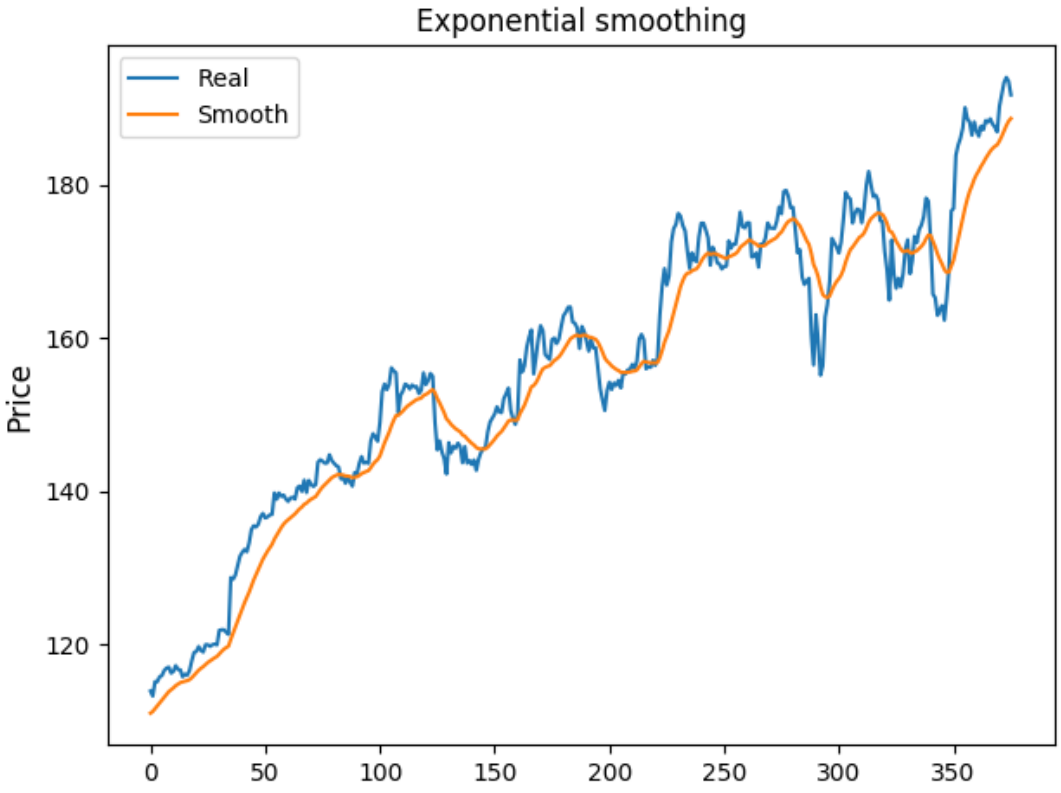



Figura 13: Alisamiento exponencial. $\alpha = 0{,}1$

Esta transformación es realizada tanto como medida de normalización como para estabilizar la media y la varianza a lo largo de la serie temporal. Con estas operaciones ya realizadas una clase binaria $y$ es calculada como salida para cada una de estas entradas; un 1 indicará entonces que el precio subirá en los próximos 15 minutos mientras que un 0 significará una bajada en el precio.

$$y = \begin{cases} 1 \rightarrow close_j > close_i \\ 0 \rightarrow close_j \leq close_i \end{cases} \tag{8}$$

Por lo tanto, dado que $i$ es el momento actual y $j$ es el siguiente, entonces $j = i + timestep$, siendo timestep 15 minutos.

Para las predicciones, la red de neuronas toma una ventana de $k$ instancias de $X$ como entrada $(X_{i-k}, ..., X_{i-2}, X_{i-1}, X_i)$, donde $X$ son tuplas de datos.

Este modelo fue diseñado como un modelo cambiante. Al final de cada día una nueva red es generada, esto significa que una matriz de pesos es nuevamente definida usando un nuevo conjunto de entrenamiento. Los resultados obtenidos con este planteamiento rondan el 55.9 % de acierto de media.

En otro nuevo estudio considerado, [8], se pretende predecir la tendencia de subida o bajada de las acciones de Google. Se toman para ello como datos el precio de salida, el precio máximo, el mínimo, el precio de cierre y el volumen de todos los días durante cinco años, creando ventanas de 30 días. El primer paso es reescalar las ventanas, para lo cual se normalizan los datos de la siguiente manera:

$$\mathrm{z} = \frac{x - \mu}{\sigma}; \mu = \frac{1}{N}\sum_{i=1}^{N}(x_i); \sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{9}$$

o con una normalización Min-Max:

$$\mathrm{X}_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{10}$$

Estas normalizaciones se escalan luego dentro del rango [-1;1] ya que se utiliza una función tangente hiperbólica como activación. El objeto de predicción de este trabajo consiste en la diferencia entre el precio de cierre del último día de la ventana con el siguiente día, el quinto próximo día o el décimo. La diferencia es entonces convertida a un valor binario para mostrar mejor si el precio sube o baja. La red es entrenada durante 100 iteraciones, epochs, obteniendo un porcentaje de aciertos del 66 % a la hora de predecir la tendencia en el día siguiente mientras que para la tendencia en el quinto y décimo día se

obtienen un 72 % y 71 % de aciertos, respectivamente.

Asimismo, en el siguiente trabajo considerado,[11], se explica qué arquitectura se utiliza para realizar las predicciones. Tras procesar los datos, discretizar y normalizar, y utilizando un 5-10 % del total como conjunto de testeo, se introducen en una red cuyos pesos y bias han sido inicializados de forma aleatoria. La red cuenta con dos capas de LSTM de 128 y 64 neuronas respectivamente y una tercera capa Dense de 16 unidades con una función de activación ReLU —Unidad Lineal Rectificada—, siendo entonces:

$$\mathrm{f}(x) = max(0, x) \tag{11}$$

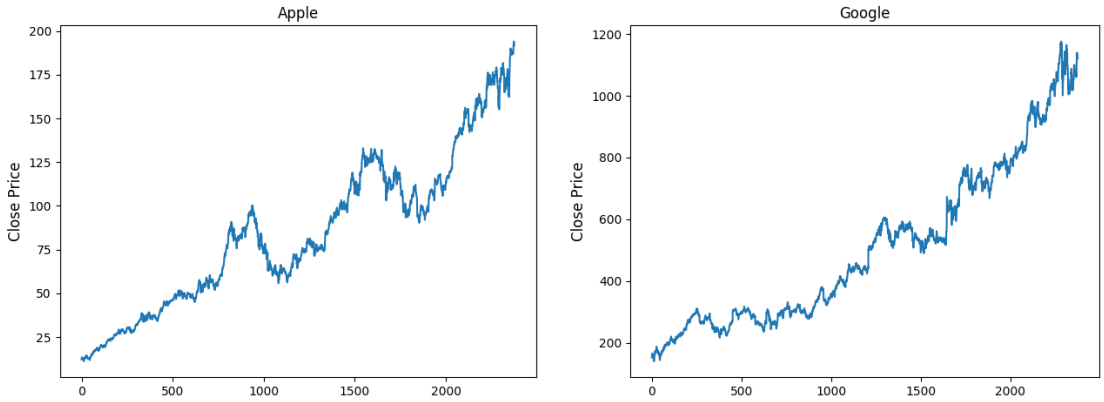
Por último utiliza como salida una capa Dense de una unidad con una activación lineal.

## 3.2. Pruebas

Tomando las ideas de los trabajos anteriores como base se intenta predecir el valor que tomarán en bolsa las acciones de Google y Apple (véase la figura 14). Para ello se utilizarán diversas arquitecturas, formas de normalización, tamaño de ventanas, indicadores técnicos y funciones de activación.

Todas las pruebas de este trabajo han sido realizadas en python utilizando la librería Keras. Ésta es una API, desarrollada por el grupo Google, de alto nivel de redes neuronales que funciona como envoltura de TensorFlow, CNTK o Theano. De esta forma la librería permite una rápida implementación y experimentación. En este caso Keras funciona como capa de TensorFlow, librería de código abierto, para la computación numérica de alto rendimiento; desarrollada por los ingenieros del equipo Google Brain dentro de la organización AI de Google, TensorFlow posee muchas herramientas para el aprendizaje automático y el deep learning.

En los gráficos siguientes el eje horizontal representará el tiempo en días mientras que el eje vertical será el precio.




(a) Precios de Apple.

(b) Precios de Google.

Figura 14: Precio de cierre de Apple y Google desde el 31 de Diciembre de 2008 hasta el 8 de Junio de 2018.

Tabla 1: Comparación de la función de pérdida.

| Tamaño de ventana | RMSE Apple | RMSE Google |
|---|---|---|
| 2 | 0.0130 | 0.0118 |
| 5 | 0.0114 | 0.0111 |

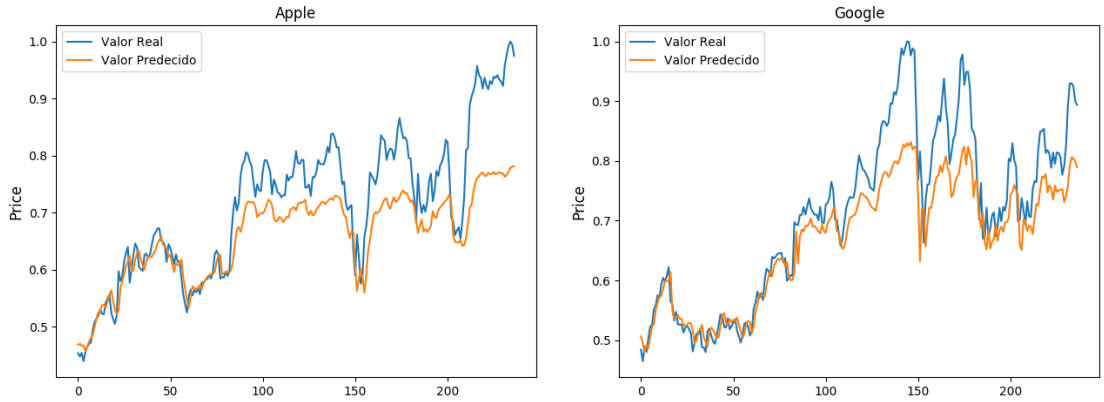
### 3.2.1. Caso 1

La primera prueba que se realiza es utilizando únicamente los precios de apertura, máximo, mínimo, cierre y la volatilidad, normalizando éstos con una función Min-Max antes de ser divididos en ventanas. La red presenta una arquitectura similar a [11], utilizando RMSD (fórmula 12) como función de pérdida.

$$\text{RMSD} = \sqrt{\frac{\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}{T}} \tag{12}$$

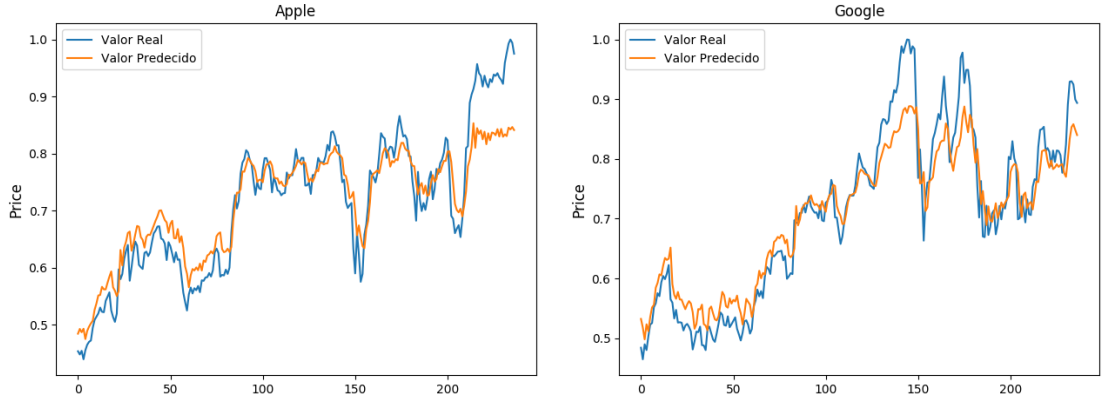
Para esta caso, ventanas de 2 días, figura 15, y 5 días, figura 16, han sido utilizadas. La comparación de la función de pérdida se encuentra en la tabla 1.

Se puede observar cómo la red realiza mejores predicciones con una ventana de cinco días. El problema de este planteamiento reside en la forma en la que se han normalizado los datos. Al normalizar éstos con la fórmula Min-Max los datos quedan reducidos a un rango entre [0,1]; el problema de este planteamiento reside en su carácter no estacionario y




(a) Predicción de Apple. (b) Predicción de Google.

Figura 15: Predicción del precio con una ventana de dos días




(a) Predicción de Apple. (b) Predicción de Google.

Figura 16: Predicción del precio con una ventana de cinco días

secuencial. En la figura 14 se puede apreciar que ambos valores tienen una fuerte tendencia creciente y puesto que se debe normalizar los conjuntos de testeo y entrenamiento de forma independiente para no filtrar información al primero, este tipo de normalización dejaría a ambos conjuntos, entrenamiento y testeo, con los primeros números de la serie cercanos a 0 y los últimos a 1. De esta forma, en consecuencia, no se representa de manera fiel la secuencia y ello es el principal motivo por el que los resultados no son muy acertados.

El inconveniente arriba reflejado inspira el planteamiento de la segunda prueba.

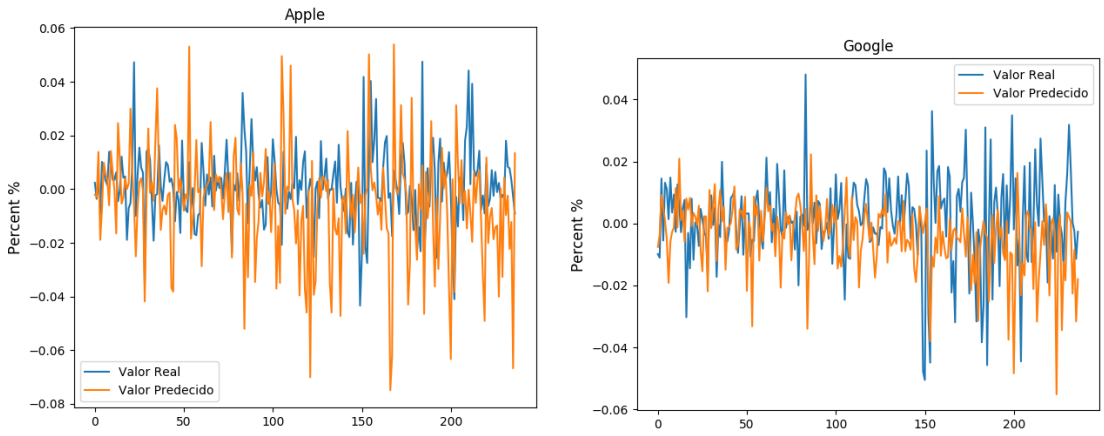
### 3.2.2. Caso 2

En este segundo estudio, utilizando los mismos datos que en el anterior, no se intenta predecir el precio de cierre del día siguiente sino el retorno diario (ecuación 13).

$$\mathrm{DR} = \frac{Close_t}{Close_{t-1}} - 1 \tag{13}$$

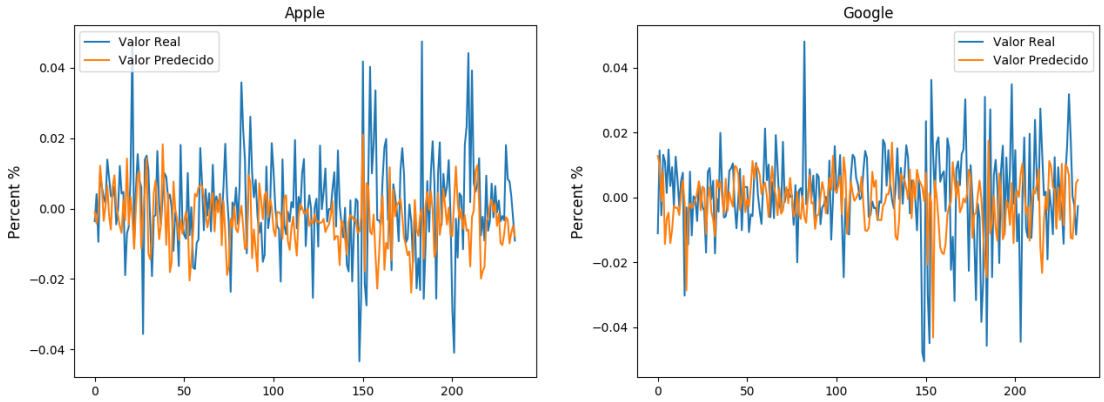
El motivo por el que se elige este método es porque de esta manera no es necesario normalizar la salida de la red, la salida expresa la variación del precio en ese día. Teniendo en cuenta que los precios no presentan una gran volatilidad, la mayoría de los datos de salida se sitúan en el rango [-0.5,0.5]. Así, las ventanas pueden ser normalizadas de forma independiente, representando cada ventana la variación del precio en ese tiempo. Para normalizar las ventanas, los datos son divididos por el máximo de cada columna. Con esta solución todos los datos están presentes en un rango de [0,1]. La arquitectura de la red consta de una capa de LSTM con 128 unidades y función de activación tangencial y una capa de salida Dense que posee una activación lineal. Ventanas de 5 días (figura 17), 15 días (figura 18) y 30 días (figura 19), han sido elegidas para este supuesto.

Los resultados se muestran en la tabla 2. Esta tabla muestra el porcentaje de acierto de la red y ha sido calculado dividiendo el número de aciertos obtenidos entre el total.



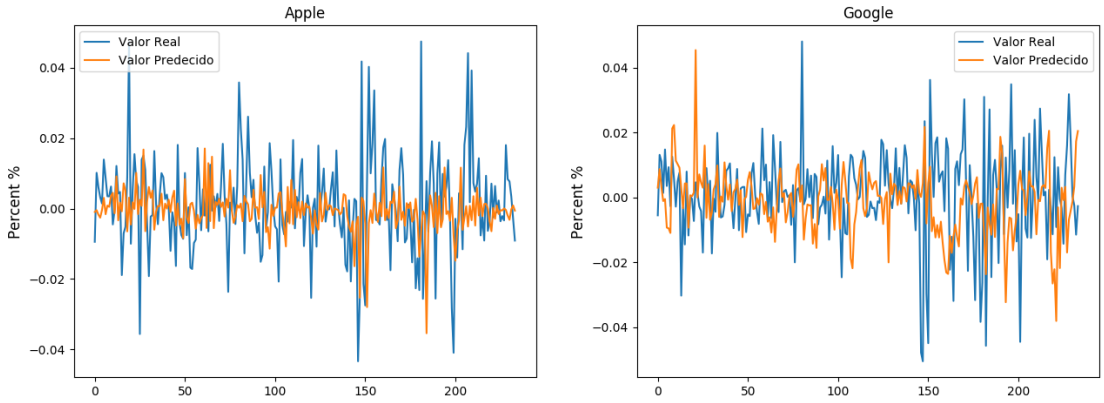

(a) Predicción de Apple. (b) Predicción de Google.

Figura 17: Predicción del retorno diario con una ventana de 5 días.




(a) Predicción de Apple. (b) Predicción de Google.

Figura 18: Predicción del retorno diario con una ventana de 15 días.




(a) Predicción de Apple. (b) Predicción de Google.

Figura 19: Predicción del retorno diario con una ventana de 30 días.

A diferencia del planteamiento anterior los datos son normalizados por ventanas, sin tener en cuenta datos futuros o pasados. De esta manera la normalización de éstos no está condicionada por los valores del futuro.

Para comprobar el resultado de la red, se compara la salida de ésta con el resultado real. Si ambos resultados coinciden en la tendencia, retorno positivo o negativo, se toma por correcto y, en caso contrario, por incorrecto. De esta manera se obtiene la tabla 2. Atendiendo a éste, podemos ver que la red no es capaz de predecir hacia donde variarán los activos, obteniendo unos pobres resultados en torno al 50 %.

Tabla 2: Porcentaje de aciertos DR.

| Tamaño de ventana | Apple | Google |
|---|---|---|
| 5 | 52 % | 47 % |
| 15 | 48 % | 51 % |
| 30 | 52 % | 45 % |

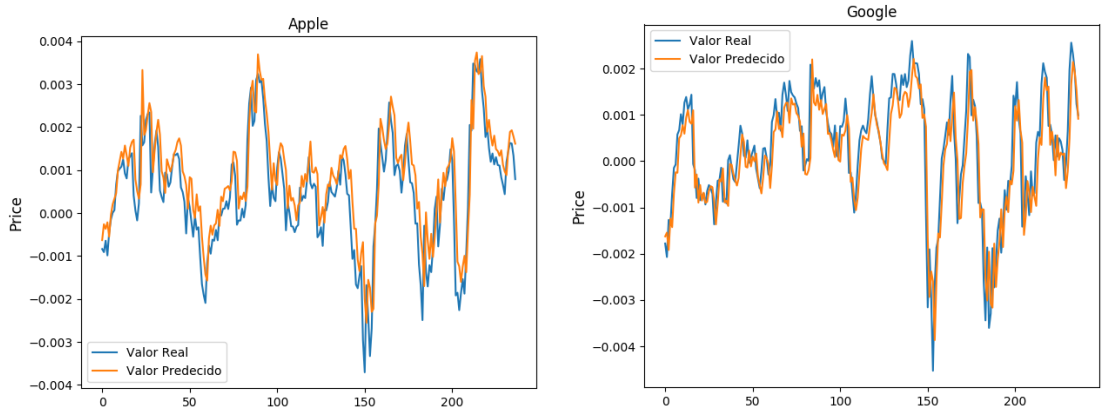
Tabla 3: Comparación de una fila de datos antes y depués de ser normalizada.

| | Open | High | Low | Close | Volume |
|---|---|---|---|---|---|
| Sin Normalizar | 185.96 | 187.34 | 184.97 | 186.30 | 2.42+07 |
| Log return Normalization | 0.0014 | 0.0015 | 0.0016 | 0.0014 | 0.003 |

### 3.2.3. Caso 3

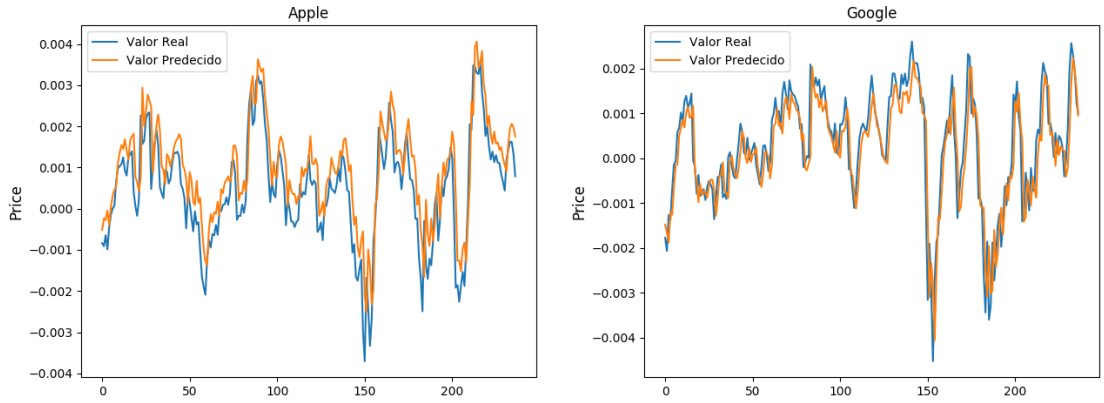
Lo más importante a la hora de plantear el problema es cómo se van a normalizar los datos. Para este caso se va a optar por normalizarlos como en [12]. Primero se realiza un alisamiento exponencial, (fórmula 6), para mejorar la capacidad de predicción, como muestra [4].

Acto seguido una transformación logarítmica, (fórmula 7), se efectúa como forma de normalizar los datos y estabilizar la varianza. La tabla 3 muestra una tupla antes y después de ser normalizada. La salida esperada de la red será, por tanto, la diferencia logarítmica del día siguiente. De esta forma los datos de salida están dentro del rango de los datos de entrada. La arquitectura de esta red presenta una capa LSTM con 128 unidades y una función de activación tangencial, seguida de una capa de salida Dense con una única neurona y una función de activación lineal. Ya que se ha realizado un alisamiento exponencial, en este caso se ha optado por una ventana de 2 días (figura 20), además de ventanas de 5 y 10 días (figuras 21 y 22, respectivamente). Se eligen estos tamaños en relación al valor $\alpha$ utilizado en el alisamiento exponencial; en este caso se ha optado por $\alpha = 0{,}1$ por lo que cuanto más grande es la ventana posee menos información útil, tal y como se muestra en la figura 23 (calculada para un caso de 60 días). Los resultados de este acercamiento son bastante buenos.




(a) Predicción de Apple. (b) Predicción de Google.

Figura 20: Ventana de dos días, transformación logarítmica.

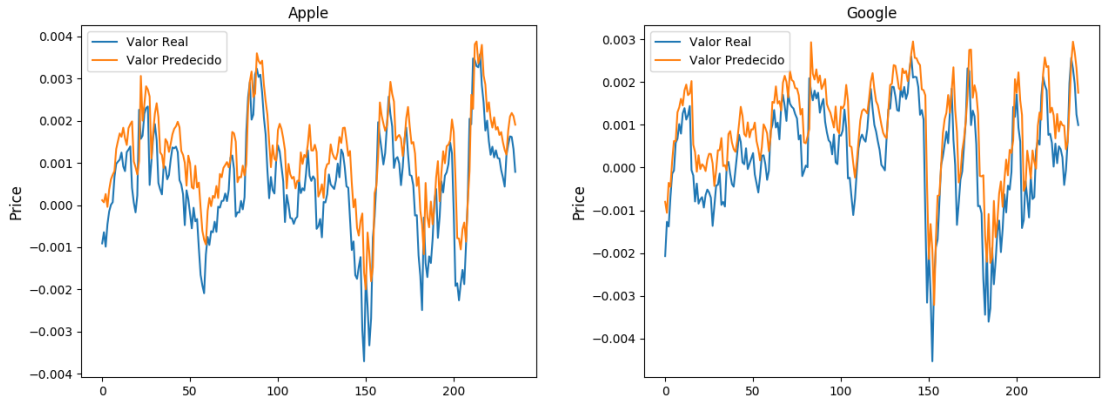



(a) Predicción de Apple. (b) Predicción de Google.

Figura 21: Ventana de cinco días, transformación logarítmica.

### 3.2.4. Caso 4

Sabiendo que el alisamiento exponencial sirve de gran ayuda para mejorar los resultados de las redes, en esta ocasión, son calculados una serie de indicadores técnicos tras efectuarlo. Los conjuntos son normalizados con una función Min-Max, después los datos son procesados por una red que consta de dos capas LSTM, de 256 unidades la primera y 126 la segunda, ambas con una función de activación sigmoide, y otras dos capas Dense, la primera de 16 unidades y activación ReLU y la segunda, de salida, de 1 unidad y una función de activación lineal. Como el alisamiento exponencial da más valor a los precios recientes que a los viejos y siendo $\alpha = 0{,}1$, las ventanas elegidas para este planteamiento




(a) Predicción de Apple.

(b) Predicción de Google.

Figura 22: Ventana de diez días, transformación logarítmica.

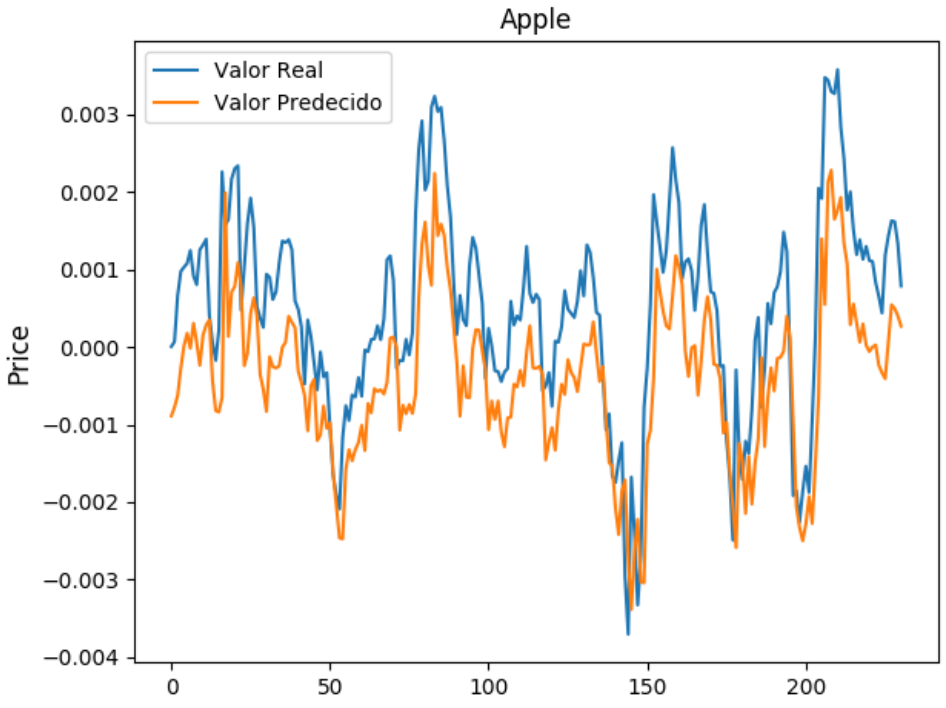



Figura 23: Ventana de sesenta días, transformación logarítmica.

son de 2, 5 y 10 días (figuras 24, 25 y 26, respectivamente).

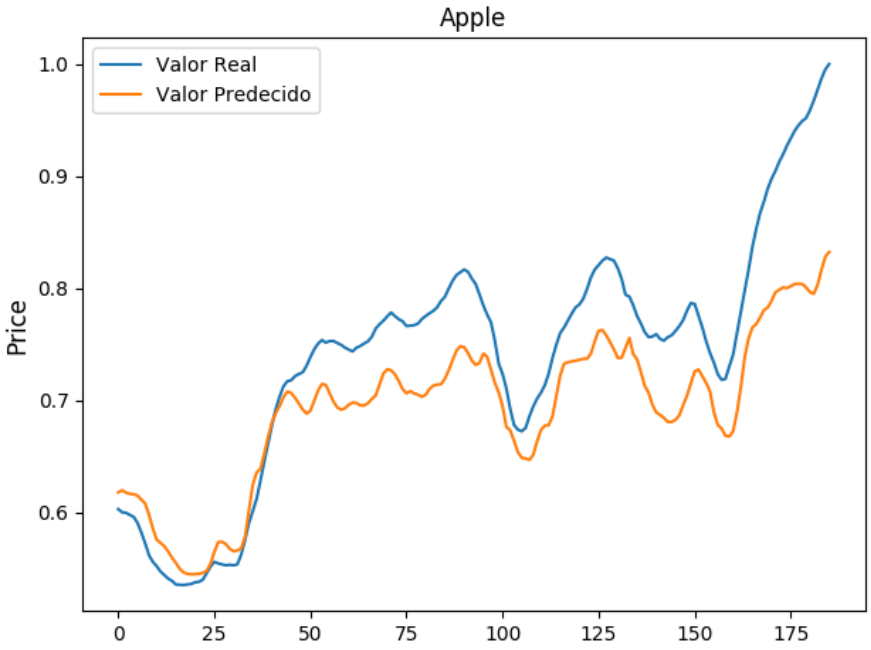



Figura 24: Ventana de 2 días, alisamiento exponencial con indicadores técnicos.

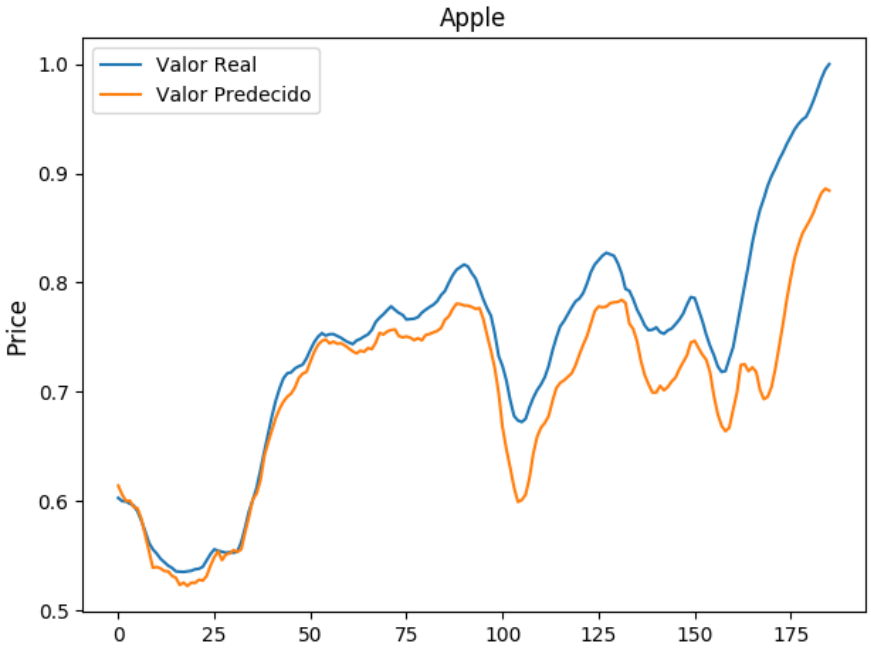



Figura 25: Ventana de 5 días, alisamiento exponencial con indicadores técnicos.

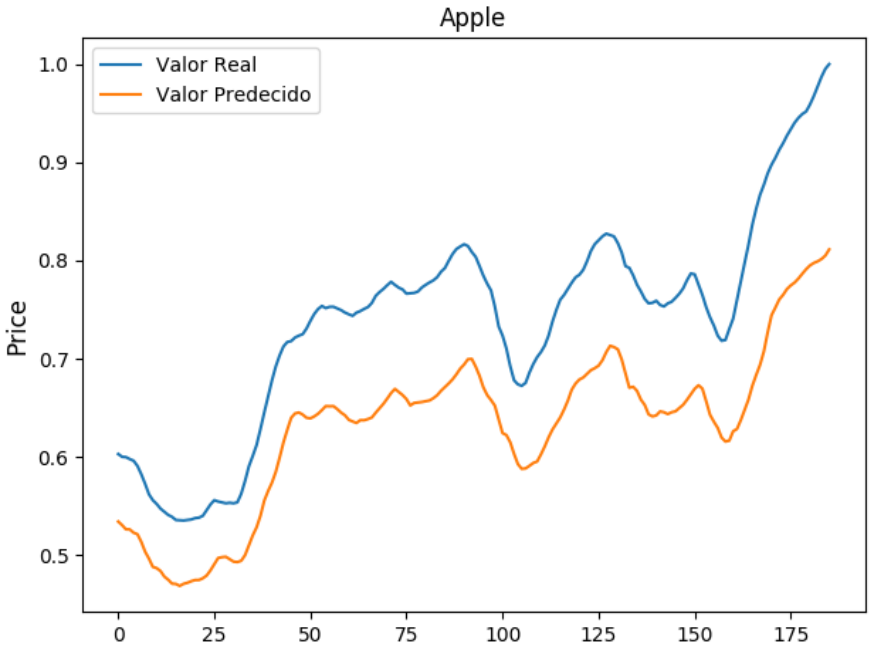



Figura 26: Ventana de 10 días, alisamiento exponencial con indicadores técnicos.

# 4. Predicción de stocks utilizando Random Forest

Al igual que con las LSTM, se tomará como punto de partida trabajos anteriores, intentando mejorar los resultados obtenidos y planteando éstos como un problema de regresión.

## 4.1. Trabajo previo

En [10] el algoritmo random forest (RF) es utilizado para predecir si el precio de un valor subirá o bajará en el futuro. Específicamente se predice la tendencia en el quinto y décimo día calculando para ello doce indicadores técnicos. Estos indicadores son elegidos en base a estudios anteriores y sus fórmulas están descritas en la tabla 4.

Los dos primeros indicadores son medias móviles y son utilizados como alisamientos del precio, así las tendencias son más fáciles de detectar. Los dos siguientes son osciladores estocásticos; %K es usado para comparar el precio de cierre en un momento $t$ en relación al máximo-mínimo en un periodo de $n$ días mientras que %D es una simple media de %K durante $n$ días. Los valores estocásticos varían entre 0 y 100; normalmente se acepta que cuando uno de éstos está por encima de 80 una acción ha sido sobrecomprada, mientras que si se sitúa por debajo de 20 ha sido sobrevendida. El indicador MACD se calcula restando a la media móvil exponencial rápida la lenta (la rápida suele ser una media de 12 días, mientras la lenta suele ser de 26) y es normalmente usado junto con los indicadores anteriores para sugerir cuándo comprar o vender. El indicador CCI es usado para detectar cuándo un stock ha sido sobrecomprado o sobrevendido; se calcula restando el promedio móvil simple de $n$ días del precio típico y luego dividiéndolo por la desviación media absoluta del precio típico. RSI es otro indicador utilizado para averiguar cuando se han sobrevendioo o sobrecomprado las acciones; su valor varía entre 0 y 100, si está por encima de 70 indica que el stock ha sido sobrecomprado, mientras que si está por debajo de 30 ha sido sobrevendido. La disparidad y ROC son indicadores similares. La disparidad es el ratio del precio de cierre en un momento $t$ y la media simple en un periodo de $n$ días, mientras que el ROC es la división entre el precio de cierre en un momento $t$ y el precio

Tabla 4: Indicadores técnicos usados como entrada

| Nombre | Fórmula | Parámetros |
|---|---|---|
| SMA(5)<br>SMA(10) | $\frac{C_t+C_{t-1}+...C_{t-n+1}}{n}$ | n = 5<br>n = 10 |
| WMA(5)<br>WMA(10) | $\frac{(n)C_t+(n-1)C_{t-1}+...+C:t-m+1}{(n+(n-1)+...+1)}$ | n = 5<br>n = 10 |
| Stochastic %K | $\frac{C_t-LL_n}{HH_n+LL_n}$ | n = 14 |
| Stochastic %D | $\frac{\sum_{i=0}^{n-1}(\%K_{t-i})}{n}$ | n = 3 |
| MACD | $EMA(fast)_t - EMA(slow)_t$ | fast = 12<br>slow = 26 |
| CCI | $\frac{M_t-SM_t}{(0{,}0015D_t)}$ | n = 20 |
| 5-days and 10-days disparity | $\frac{C_t}{SMA(n)_t}100$ | n = 5<br>n = 10 |
| OSCP | $\frac{SMA(fast)_t-SMA(slow)_t}{SMA(fast)_t}100$ | fast = 5<br>slow = 10 |
| ROC | $\frac{C_t}{C_{t-m}}100$ | n = 10 |
| 10-days momentum | $C_t - C_n$ | n = 10 |
| RSI | $100-\frac{100}{1+\frac{\frac{\sum_{i=0}^{n-1}Up_{t-i}}{n}}{\frac{\sum_{i=0}^{n-1}Dw_{t-i}}{n}}}$ | n = 14 |
| 5-days standard deviation | $\sqrt{\frac{\sum_{i=0}^{n-1}(C_{t-1}-SMA(n)_t)^2}{n}}$ | n = 5 |

de cierre en un momento $t-n$. Momentum es la diferencia entre el precio en un momento $t$ y el precio en un momento $t-n$; un momentum negativo es normalmente utilizado para generar una señal de compra, un momentum positivo para una de venta. Por último, OSCP es un indicador técnico que muestra la diferencia entre la media móvil rápida y lenta, dividida entre la media móvil rápida.

Los mejores resultados de este planteamiento se obtuvieron con un máximo de cinco indicadores por árbol, creando 100 árboles con una profundidad ilimitada. Los resultados de este planteamiento rondan el 76.5 % de aciertos para la predicción de cinco días y el 80.8 % para la de diez días.

En [3] los random forest son utilizados junto a otros algoritmos para predecir el movimiento de stocks en el mercado iraní. El trabajo demuestra que los random forest funcionan mejor que los árboles de decisión o los clasificadores bayesianos. Para este

trabajo se calculan de nuevo una serie de indicadores técnicos. No obstante, la principal diferencia es que, además, se utilizan unos indicadores fundamentales macroeconómicos. Estos indicadores son el precio del petróleo, el precio del oro y el cambio de moneda USD/IRR debido al papel fundamental que juegan éstos en la economía iraní. De esta manera, se obtiene hasta un 78.81 % de aciertos.

La investigación realizada en [4] es muy similar a [10]. También se calculan una serie de indicadores técnicos: RSI —Relative Strength Index—, %K —Stochastick Oscillator—, %R —Williams %R—, MACD —Moving Average Convergence Divergence—, PROC —Price Rate of Change— y OVB —On Balance Volume—. La principal diferencia está en que, en este caso, antes de calcular todos estos indicadores un alisamiento exponencial es realizado. De esta manera se alcanzan hasta un 93.96 % de aciertos.

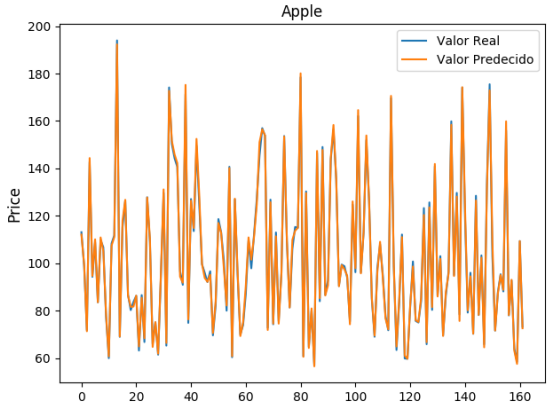
## 4.2. Pruebas

Como en el apartado anterior, todo el código está escrito en python, utilizando la librería scikit-learn, siendo ésta una librería de machine learning de código abierto que posee algoritmos tanto de clasificación como de regresión y agrupamiento —clustering—. Algunos de los algoritmos que posee la librería son k-means, random forest, BBSCAN, etc. Scikit-learn fue diseñado para interoperar con las librerías científicas NumPy y SciPy.

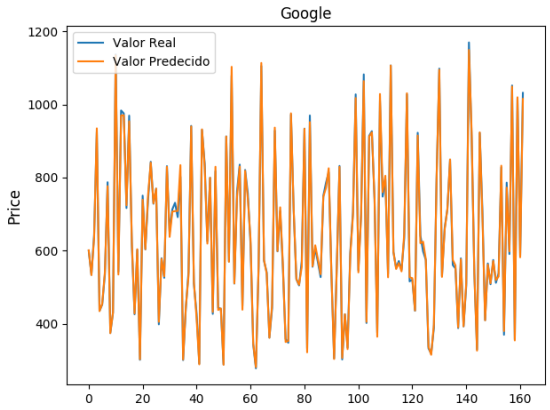
### 4.2.1. Caso 1

Utilizando únicamente los mismos indicadores que en la tabla 4 se intenta predecir el precio de cierre del día siguiente. Para ello se crean 100 árboles con un máximo de cinco atributos elegidos de manera aleatoria por el algoritmo. Los datos que se utilizan perteneces al periodo situado entre el 1 de Enero de 2012 hasta el ocho de Junio de 2018; un 10 % del conjunto es utilizado como muestra de testeo y el otro 90 % como conjunto de entrenamiento. Los datos son barajados de forma aleatoria. De esta manera el conjunto de testeo puede estar compuesto por datos de cualquier momento. Los resultados obtenidos se muestran en la figura 27.

Si no barajamos los datos los resultados que se obtiene se muestran en la figura 28.




(a) Predicción de Apple.



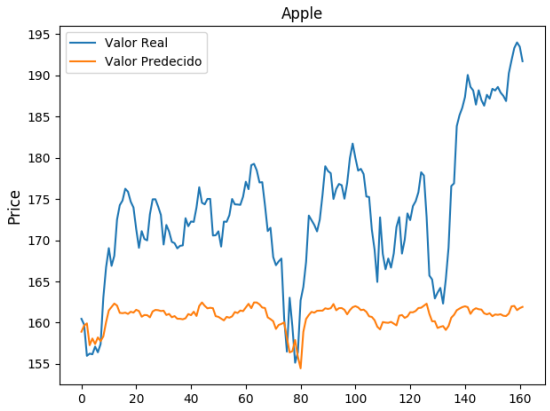

(b) Predicción de Google.

Figura 27: Árboles de decisión, 100 árboles, 5 características, barajando datos.

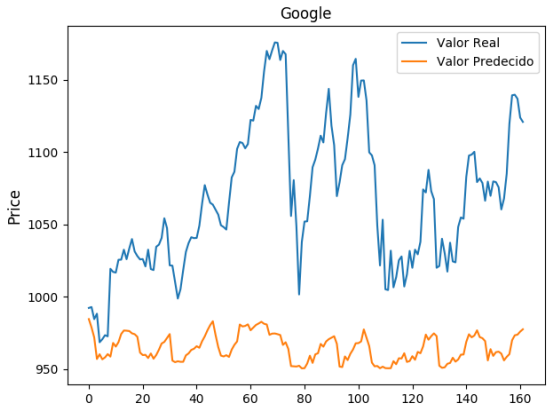
Se puede apreciar que los datos se sitúan muy por debajo de su valor real. Como ya se ha dicho, los datos de estos conjuntos tienen una gran tendencia creciente, por lo que los datos del conjunto de entrenamiento no representan a los datos del conjunto de testeo. Para solucionar este inconveniente se toma la idea de [12] diseñando un modelo cambiante. De esta manera un conjunto de árboles es generado cada día para predecir el día siguiente. Los resultados de este planteamiento se muestran en la figura 29.

### 4.2.2. Caso 2

Aun con un modelo cambiante las predicciones poseen un cierto retraso frente al presente. Para intentar mejorar las predicciones se realiza un alisamiento exponencial




(a) Predicción de Apple.



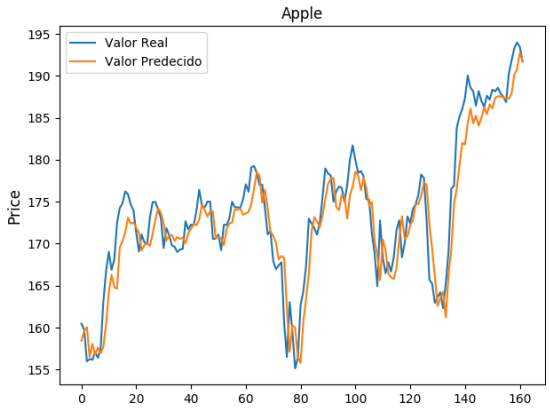

(b) Predicción de Google.

Figura 28: Árboles de decisión, 100 árboles, 5 características, no barajando datos.

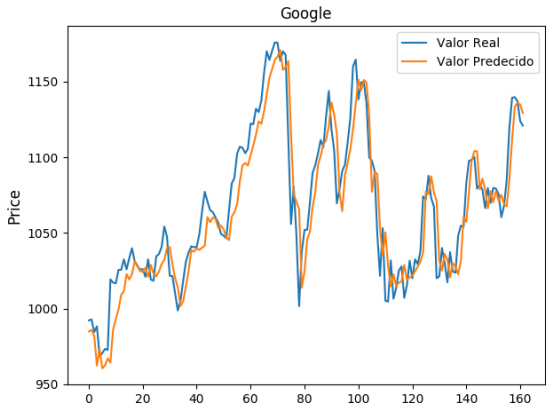
(fórmula 6). Con esto se pretende priorizar el valor de los últimos días respecto de los primeros. Las figuras 30 y 31 ilustran los resultados de este planteamiento. Las imágenes (a) de cada figura muestran la salida del algoritmo, siendo el resultado de las predicciones el valor de cierre con el alisamiento exponencial efectuado; de otra parte, las imágenes (b) muestran la salida de la red siendo el valor real el objetivo de las predicciones.

### 4.2.3. Caso 3

Si se aplica el mismo planteamiento que en [10], clasificar si el precio va subir o bajar, pero en vez de predecir la tendencia en el quinto y décimo día se predice la del día siguiente, se obtienen unos pobres resultados (véase tabla 5).

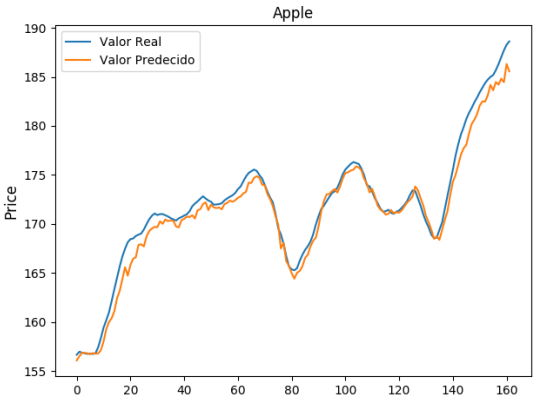



(a) Predicción de Apple.

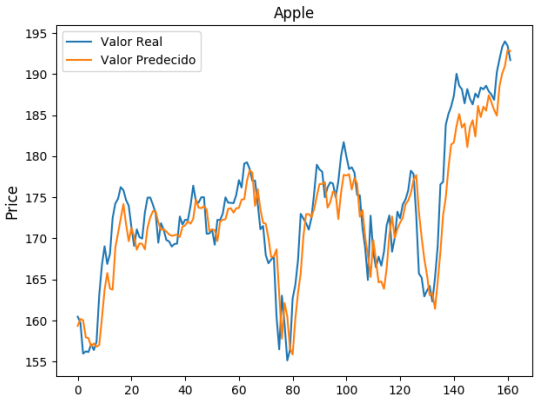



(b) Predicción de Google.

Figura 29: Árboles de decisión, modelo cambiante.




(a) Predicción de Apple respecto el alisamiento.




(b) Predicción de Apple respecto el valor real.

Figura 30: Predicción del precio de Apple.

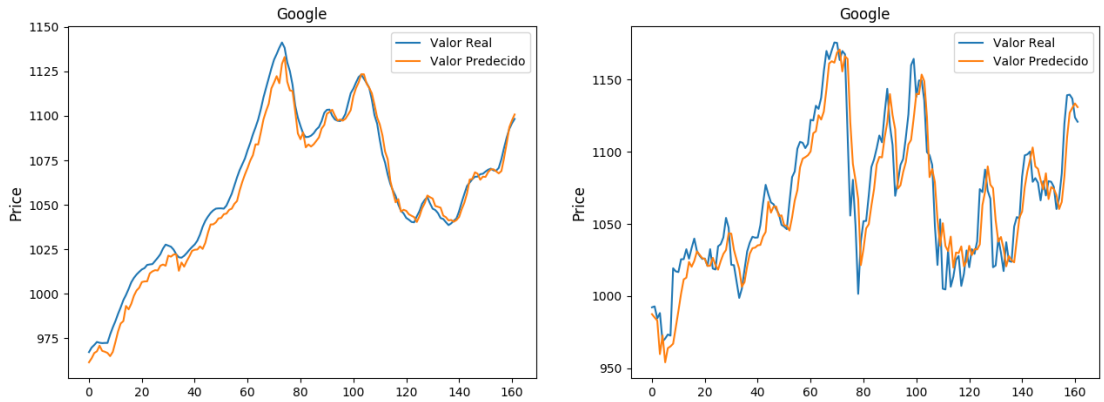
Tabla 5: Porcentajes de aciertos RF un día.

| Aciertos | Subida | Bajada |
|---|---|---|
| 0.52 % | 0.53 % | 0.51 % |

Tabla 6: Porcentajes de aciertos RF un día con alisamiento exponencial.

| Aciertos | Subida | Bajada |
|---|---|---|
| 0.83 % | 0.94 % | 0.78 % |

Sin embargo, esto cambia en gran medida si se aplica un alisamiento exponencial (tabla 6). Esto se debe a que el alisamiento exponencial reduce en gran medida el ruido de los datos; con este ruido reducido, los datos posee una distribución más normalizada y por ese motivo, como puede comprobarse, los árboles de decisión son capaces de aumentar muy sensiblemente el porcentaje de aciertos. (Las columnas de las tablas 5 y 6 hacen referencia al porcentaje de aciertos totales, al porcentaje de aciertos cuando el algoritmo marca una subida y al porcentaje de aciertos cuándo el algoritmo marca una bajada).




(a) Predicción de Google respecto el alisamiento. (b) Predicción de Google respecto el valor real.

Figura 31: Predicción del precio de Google.

# 5. Conclusión

En este trabajo se intenta predecir el precio de los activos de Google y Apple en bolsa. Para ello se utilizan dos técnicas diferentes, LSTM y Random Forest. El primero es un tipo de red neuronal recurrente mientras que el segundo inicializa una cantidad de árboles de decisión tomando un número determinado de atributos de forma aleatoria. La gran diferencia entre estos dos algoritmos está en que las Redes de Neuronas necesitan normalizar los datos según sus funciones de activación y los Árboles de Decisión no. Los datos de la bolsa son datos no estacionarios y la normalización presenta un gran problema debido a la dificultad de modelar los mismos.

Respecto a la utilización de LSTM, diferentes formas de normalización son usadas. La principal forma utilizada es la de Min-Max (3.2.1); ésta presenta un gran problema y es que acota los resultados a un rango [0, 1], siendo 0 el valor más bajo del conjunto y 1 el más alto. El inconveniente de este método consiste en que los valores del conjunto de testeo y entrenamiento han de ser normalizados de manera independiente para evitar el traspaso de información. En el caso de Apple, por ejemplo, los datos del conjunto de testeo se sitúan cerca del rango [150, 200] mientras que el de entrenamiento se sitúan entre [10, 150]; de esta manera cuando la red es entrenada los datos cercanos a 10 habrán sido normalizados con números cercanos a 0, y a 1 en el caso de los datos cercanos a 150. Por contra, en el conjunto de testeo, los datos cercanos a 150 serán representados por números cercanos a 0 mientras que los cercanos a 200 por números alrededor de 1. De esta forma los datos no se representan de manera adecuada y la red no es capaz de predecir de forma muy acertada. Si por el contrario se normalizara el conjunto antes de ser dividido, existiría un paso de información ya que el conjunto de entrenamiento no tendría números cercanos a 1 y el conjunto de testeo no los tendría cercanos a 0.

En el planteamiento del retorno diario (3.2.2), se opta por normalizar los datos en tamaños de ventana. Con este acercamiento, cada conjunto de entrada, ventana, estará acotada entre 0 y 1 ya que el máximo de cada ventana divide al resto de datos. De esta manera las ventanas representan la variación del precio en ese periodo, independientemente del valor de los activos. La idea de este método es que la red fuese capaz de identificar

Tabla 7: DR con indicadores técnicos.

| Tamaño de ventana | Apple | Google |
|---|---|---|
| 5 | 51 % | 48 % |
| 15 | 49 % | 50 % |
| 30 | 53 % | 49 % |

patrones en los cambios del precio, sin embargo, los resultados demuestran que la red no es capaz de identificar éstos. El rendimiento de la red tampoco mejora si se le añaden indicadores técnicos, tal y como se muestra en la tabla 7.

Los mejores resultados son obtenidos cuando se realiza un alisamiento exponencial. En las figuras 20, 21, 22, además, se utilizó la diferencia logarítmica como forma de normalización (3.2.3). Al utilizar este método, se representa la diferencia entre el día $t$ y $t_{-1}$, por lo que se aumenta el total de información que la red recibe. Este método obtiene grandes resultados, sin embargo, su funcionamiento está lejos de ser perfecto ya que la salida de la red no representa el valor real de las acciones, objetivo principal del trabajo, si no la diferencia entre el día de hoy y mañana. Si se observa la figura 23, diferencia logarítmica con una ventana de 60 días, se puede observar cómo al tomar valores pequeños de $\alpha$ los datos pasados poseen menos peso frente a los recientes. Como en el caso anterior, la red no mejora si se le añaden indicadores técnicos.

Los Random Forest poseen la gran ventaja de que los datos no han de ser normalizados. Utilizando los indicadores técnicos de la tabla 4 se obtienen las figuras 27, 28 y 29 (apartado 4.2.1). La primera de éstas es el resultado de entrenar el algoritmo barajando los datos, esto es, mezclando todos los datos sin tener en cuenta el orden temporal. De esta manera, los conjuntos de testeo y de entrenamiento están formados por datos de todo el espectro temporal. La figura 28 muestra el mismo planteamiento que la anterior pero sin barajar los datos. De esta forma el conjunto de testeo está formado únicamente por los datos más cercanos al presente. Si se observa la figura se comprueba que el algoritmo no es capaz ni de acercarse al precio real. El motivo de este pobre resultado es que los árboles no han conocido ningún dato cercano a éstos en el momento de su creación por lo que no son capaces de predecir en ese rango de datos. Para solucionar este problema se utiliza

un modelo cambiante; cada día el algoritmo de Random Forest es ejecutado para crear un nuevo conjunto de árboles que tomen en consideración los datos del último día. Los resultado de este planteamiento se muestran en la figura 29. En ésta se aprecia que las predicciones poseen un gran retraso y que los árboles realizan un salida cercana al precio del día anterior.

Tras realizar numerosas pruebas con ambos algoritmos, se ha comprobado la dificultad de predecir el valor de las acciones en bolsa ya que debido al carácter aleatorio de los datos es muy difícil realizar predicciones exactas.

Después de muchos y distintos intentos, diferentes arquitecturas y funciones de activación, en las LSTM se ha comprobado que éstas no son capaces de predecir el precio exacto de los activos. En la mayoría de los casos muestran un resultado muy similar al del día anterior y son incapaces de predecir cuándo la tendencia va a cambiar. El motivo de esto reside en el carácter aleatorio de los datos.

Los Random Forest a su vez presenta unos resultados similares a las LSTM. El mismo retraso está presente en ellos y son a su vez incapaces de predecir cuándo la tendencia va a cambiar, aunque en tareas de clasificación pueden obtener grandes resultados, como muestra la tabla 6.

Sin embargo, ambos algoritmos pueden ser muy útiles como ayuda para tomar decisiones. En el ejemplo de la diferencia logarítmica de LSTM, la predicción de Google con una ventana de cinco días es bastante acertada. Un estudio con diversas redes de arquitecturas y tamaños de ventana diferentes podrían mejorar aún más la predicción.

Uno de los caminos para mejorar los resultados obtenidos podría estar en el estudio de los datos del intermercado. Así, partiendo del concepto de que existe una economía globalizada en la que todo está interconectado se podría aplicar la idea a este caso y considerar adicionalmente una gran cantidad de factores macroeconómicos diversos (movimientos en bolsa de grandes empresas, precios de materias primas, luz, petróleo, divisas, etc.) intentando encontrar patrones ocultos con los que predecir la tendencia del mercado. Asimismo, otra de las posibles soluciones para mejorar el rendimiento podría estar en el estudio del análisis de sentimiento (sentiment analysis); cada día, en redes sociales, prensa,

historiales de navegación, consultas en buscadores, se generan millones de nuevos datos y el tratamiento de toda está cantidad de información podría ser de gran ayuda a la hora de identificar las acciones que van a llevar a cabo las personas; con un análisis adecuado de redes sociales (twitter, buscadores, foros profesionales y similares) de los inversores se podría llegar a anticipar reacciones de las personas con influencia en los cambios de los precios a corto plazo. Si esta información fuese utilizada por las redes de neuronas y/o los sistemas basados en árboles de decisión, junto a los propios datos de la empresa sobre la que se quiere efectuar la predicción, más el análisis de los datos del intermercado, es muy probable que los resultados mejorasen notablemente.

# Referencias


[1] W Du, Shuyang, Pandey, Madhulima, Xing, and Cuiqun. Modeling approaches for time series forecasting and anomaly detection. 2017.

[2] Felix A. Gers, Douglas Eck, and Jürgen Schmidhuber. Applying lstm to time series predictable through time-window approaches. In Roberto Tagliaferri and Maria Marinaro, editors, *Neural Nets WIRN Vietri-01*, pages 193–200, London, 2002. Springer London.

[3] Sadegh Bafandeh Imandoust and Mohammad Bolandraftar. Forecasting the direction of stock market index movement using three data mining techniques : the case of tehran stock exchange. 2014.

[4] Luckyson Khaidem, Snehanshu Saha, and Sudeepa Roy Dey. Predicting the direction of stock market prices using random forest. *CoRR*, abs/1605.00003, 2016.

[5] Breiman L. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996.

[6] Breiman L. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[7] Zachary Lipton, David Kale, and Randall Wetzel. Phenotyping of clinical time series with lstm recurrent neural networks. 10 2015.

[8] Di Persio Luca and Honchar Oleksandr. Recurrent neural networks approach to the financial forecast of google assets. In *INTERNATIONAL JOURNAL OF MATHEMATICS AND COMPUTERS IN SIMULATION*, volume 11, pages 7–13, May 22 2017.

[9] Xiaolei Ma, Zhimin Tao, Y Wang, Haiyang Yu, and Yunpeng Wang. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. 54, 05 2015.

[10] T. Manojlović and I. Štajduhar. Predicting stock market trends using random forests: A sample of the zagreb stock exchange. In *2015 38th International Convention*

*on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1189–1193, May 2015.

[11] Shraddha Varma Murtaza Roondiwala, Harshal Patel. Predicting stock prices using lstm. In *International Journal of Science and Research (IJSR)*, pages 1754–1756, 2015.

[12] D. M. Q. Nelson, A. C. M. Pereira, and R. A. de Oliveira. Stock market's price movement prediction with lstm neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1419–1426, May 2017.

[13] Razvan Pascanu, Tomas Mikolov, and Y Bengio. On the difficulty of training recurrent neural networks. 11 2012.

[14] J. Schmidhuber S. Hochreiter. Long short-term memory. 9:1735–80, 12 1997.

[15] Shi-yan. Understanding LSTM and its diagrams. `https://github.com/shi-yan/FreeWill/blob/master/Docs/Diagrams/LSTM_diagram.pptx`.

[16] Sima Siami-Namini and Akbar Siami Namin. Forecasting economics and financial time series: Arima vs. lstm. *CoRR*, abs/1803.06386, 2018.

[17] Peter Norvig Stuart Russell. *Artificial Intelligence A Modern Approach*. 2014.

# Bibliografía


Russel, S., Norvig, N. (2014). *Artificial Intelligence a Modern Approach.* England: Pearson.

Nilsson, N. J. (1998). *Artificial Intelligence a New Synthesis.* San Francisco: Morgan Kaufmann Publishers, Inc.

Mendelsohn, L. B. (2018). *Profitable Trading with Artificial Intelligence.* United States of America:Market Technologies, LLC.

Hilpiscg, Y. (2014). *Python for Finance.* United States of America: O'Reilly Media, Inc.

Karpathy, A. (Mayo 21, 2015). *The Unreasonable Effectiveness of Recurrent Neural Networks.* Recuperado en Abril 2018, de https://karpathy.github.io/2015/05/21/rnn-effectiveness/

Harris, L. (2003). *Trading & Exchanges Market Microstructure For Practitioners.* India: Oxford University Press.

Olah, C. (Agosto 27, 2015). *Understanding LSTM Networks.* Recuperado en Abril 2018, de https://colah.github.io/posts/2015-08-Understanding-LSTMs/

# Índice de figuras

# Índice de fórmulas

# Índice de cuadros