E&A Revista Economía y Administración (E&A)
Volumen 17, Núm. 2, 2025

Redes neuronales recurrentes aplicadas al pronóstico de precios en el mercado de valores 2018-2024

DOI: https://doi.org/10.5377/eya.v17i2.21510

Recibido: 18/5/2024 Aceptado: 13/03/2025

Juan Ángel Isaula Mejía [1]

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 que se dan en los mercados bursátiles. Debido a los riesgos y rendimientos variables de la bolsa, la predicción de esta es un tema de mucha importancia para los que invierten en ella. Tener la capacidad de pronosticar la tendencia o el precio de las acciones es una información muy valiosa para los inversores. En el presente trabajo se han propuesto algunos modelos que pueden llegar a ser viables para la predicción de índices bursátiles, tales como AAPL y ^STOXX50E. Para ello, se ha realizado el estudio de dos modelos diferentes, los cuales pertenecen a la familia de redes neuronales recurrentes (LSTM y GRU). Finalmente, se realiza una comparativa entre los distintos modelos utilizando la métrica MSE (error cuadrático medio). Se concluye que ambos modelos son adecuados para la predicción de estos índices bursátiles, evidenciando el gran poder predictivo que tienen las redes neuronales artificiales, los modelos de tipo caja negra pueden ser muy ventajosos, dado que se pueden optimizarse aún más sus parámetros y de esa manera obtener mejores resultados en las predicciones.

Palabras clave: *Predicción*, *Redes Neuronales Recurrentes*, *Modelo LSTM*, *Modelo GRU*, *Error Cuadrático Medio (MSE)*.

JEL: C45, G17, C53

1 Juan Ángel Isaula Mejía, Universidad Nacional Autónoma de Honduras, Correo Electrónico: jisaula@unah.hn ORCID: https://orcid.org/0000-0003-3791-9232

E&A
Revista Economía y Administración (E&A)
Volumen 17, Núm. 2, 2025

Recurring neural networks applied to price forecasting in the stock market 2018-2024

DOI: https://doi.org/10.5377/eya.v17i2.21510

Received: 18/5/2024 Accepted: 13/03/2025

Juan Ángel Isaula Mejía[1]

Abstract

Predicting the direction of stock price trends is vital to optimally developing strategies for transactions that occur in the stock markets. Due to the risk and variable returns of the stock market, its prediction is a very important issue for those who invest in it. Having the ability to forecast the trend or price of stocks is very valuable information for investors. In the present work, some models have been proposed that may become viable for the prediction of stock market indicators, such as the case of the S&P 500. For this, the study of two different models has been carried out, which belong to the family of recurrent neural networks (LSTM and GRU). Finally, a comparison is made between the different models using the MSE (mean square error) metric. It is concluded that both models are suitable for predicting these stock indices, evidencing the great predictive power that artificial neural networks have. Black box models can be very advantageous, given that their parameters can be further optimized and thus way to obtain better results in predictions.

Key Words: *Prediction*, *Recurrent Neural Networks*, *LSTM Model*, *GRU Model*, *Mean Square Error* (MSE).

JEL: C45, G17, C53

1 Juan Ángel Isaula Mejía, Universidad Nacional Autónoma de Honduras, E-mail: jisaula@unah.hn ORCID: https://orcid.org/0000-0003-3791-9232

I. Introducción

Desde hace décadas se ha 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 la bolsa debido a los riesgos y rendimientos variables de esta. 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.

La mayoría de los operadores en los mercados bursátiles buscan a través del tiempo encontrar algún método o criterio que les facilite predecir de la manera más cercana posible la evolución futura de las acciones, los bonos, los contratos derivativos y el mercado de divisas, en la medida que ese punto se logre se obtendrían altos márgenes de ganancia y disminuir el riesgo y las pérdidas.

En los últimos años el uso de redes neuronales artificiales ha crecido de manera exponencial con aplicación en diferentes campos. Las redes neuronales se están utilizando, solas o combinadas con otros métodos, para obtener los mejores resultados posibles en física, biología, medicina, economía e ingeniería, entre muchos otros.

El objetivo principal de este artículo radica en la comprensión de los modelos de Redes neuronales artificiales y posteriormente su aplicabilidad en la predicción del precio de cierre de los índices AAPL *y* ^STOXX50E de la bolsa de valores utilizando dos modelos de predicción, *Red de Memoria de Corto y Largo Plazo* (*LSTM*), *Unidades Recurrentes Cerradas* (*GRU*). No se está abogando por el uso indiscriminado de dichos modelos. Sin embargo, se provee una visión sobre su alcance en comparación a los modelos tradicionales utilizados para dicha tarea. Esto abre la posibilidad de tener a mano una herramienta para observar algún pronostico en un campo específico, a partir de los cuales se puede tener una idea de que decisión tomar para la posterioridad.

II. Marco teórico

Desde los tiempos antiguos los seres humanos se han preocupado por algunos comportamientos de la naturaleza, los babilónicos se preocupaban por la posición relativa de las estrellas y planetas para predecir eventos astronómicos, las series de tiempo pudieron haber jugado un papel importante para dar respuesta a este tipo de eventos. El análisis de series de tiempo, utilizando como dominio el tiempo y vistas como procesos estocásticos (sucesión de variables aleatorias que evolucionan en función de otra variable), surgen en 1970 con la obra pionera de Box-Jenkins (Jenkins, 1970). Los personajes más importantes a lo largo del desarrollo de la teoría sobre series de tiempo han sido:

(Jenkins, 1970) En colaboración, desarrollaron un esquema de clasificación para una amplia familia de modelos de series de tiempo. En esta clasificación se distinguen, modelos autorregresivos de

orden p (AR(p)), modelos de promedio móvil de orden q (MA(q)), modelos autorregresivos de promedio móvil (ARMA (p,q)) y modelo autorregresivo e integrado de promedio móvil (ARIMA (p,d,q)).

(Reinsel, 2020) Aparecieron las series de tiempo univariante y multivariante, estudiadas por Gregory C. Reinsel. Mientras Peter J. Brockwell trabajaba en el análisis de las series de tiempo y procesos estocásticos, Richard A. Davis estaba trabajando en la inferencia, estimación, predicción y las propiedades generales de las series de tiempo. Los dos personajes en colaboración desarrollaron las obras: Times series: Theory and Methods (Davis, 1987), y desarrollo y análisis de series de tiempo con el software 2000 (Strazicich, 2002).

El uso más común en la actualidad de las series temporales es su análisis para la predicción y pronostico. Resulta difícil imaginar una rama de las ciencias en la que no aparezcan datos que puedan ser considerados como series temporales. Las series temporales se estudian en estadística, procesamiento de señales, econometría y muchas otras áreas.

En 2018, se llevó a un desafío histórico en inteligencia artificial (IA), el desafío de aprendizaje automático explicable. El objetivo de la competencia era crear un modelo de redes neuronales artificiales (modelo de caja negra) complicado para un conjunto de datos y explicar cómo funcionaba (Rudin, 2019). Un equipo no siguió las reglas, en lugar de enviar una caja negra, crearon un modelo que era completamente interpretable. En el presente trabajo se usan algunos modelos de caja negra, explícitamente estos modelos son *Red de Memoria de Corto y Largo Plazo* (*LSTM*), *Unidades Recurrentes Cerradas* (*GRU*).

1936 – Alan Turing. Fue el primero en estudiar el cerebro como una forma de ver el mundo de la computación. Sin embargo, los primeros teóricos que coincidieron en los fundamentos de la computación neural fueron Warren McCulloch, un neurofisiólogo, y Walter Pitts, un matemático, quienes, en 1943 lanzaron una teoría acerca de la fortuna de trabajar de las neuronas (Pitts, 1990). Ellos modelaron una red neuronal simple mediante circuitos eléctricos. La primera conferencia sobre inteligencia artificial (IA) en la cual se discutió sobre la capacidad de las computadoras para simular el aprendizaje tuvo lugar en Dartmouth 1956. A partir de ahí investigadores han desarrollado distintos tipos de redes neuronales:

1957 – Frank Rosenblatt. Comenzó el desarrollo del Perceptron (Mitchell, 1997). Esta es la red neuronal más antigua; utilizándose hoy en día para aplicación como identificar de patrones. Este modelo era capaz de generalizar, es decir, después de haber aprendido una serie de patrones podía reconocer otros similares, aunque no se hubiesen presentado en el entrenamiento. Sin embargo, tenía una serie de limitaciones, por ejemplo, era incapaz de clasificar clases no separables linealmente.

1974 – Paul Werbos. Desarrolló la idea básica del algoritmo de aprendizaje de propagación hacia atrás (backpropagation) (Werbos, 1990).

1997 – Schmidhuber and Hochreiter. Desarrollaron la red de Memoria a corto plazo y largo plazo (LSTM) (Hochreiter, 1997); la cual en la actualidad tiene un gran potencial para predicción de series temporales y es uno de los tipos de redes neuronales que estudiaremos en este artículo.

En la actualidad son numerosos los trabajos que se realizan y publican cada año con el uso de redes neuronales.

2020 – OpenAI. Creó un tipo de algoritmo (GPT-3) que emplea redes neuronales para producir texto que simulan la redacción humana (Hopkins, 2020). Tiene una capacidad de 175,000 millones de parámetros de aprendizaje automatizado y 96 capas (términos que se define más adelante).

III. Método

El enfoque de este estudio es exploratorio y predictivo, con un alcance centrado en el análisis de series temporales financieras. El diseño incluye la selección de modelos de redes neuronales recurrentes (LSTM y GRU), su entrenamiento, y la evaluación comparativa utilizando métricas estándar para evaluar su desempeño como el error cuadrático medio (MSE).

Series Temporales

Una serie temporal es una colección de observaciones indexadas por la fecha de cada observación, denotada por y_t

$$\{y_1, y_2, \ldots, y_T\}$$

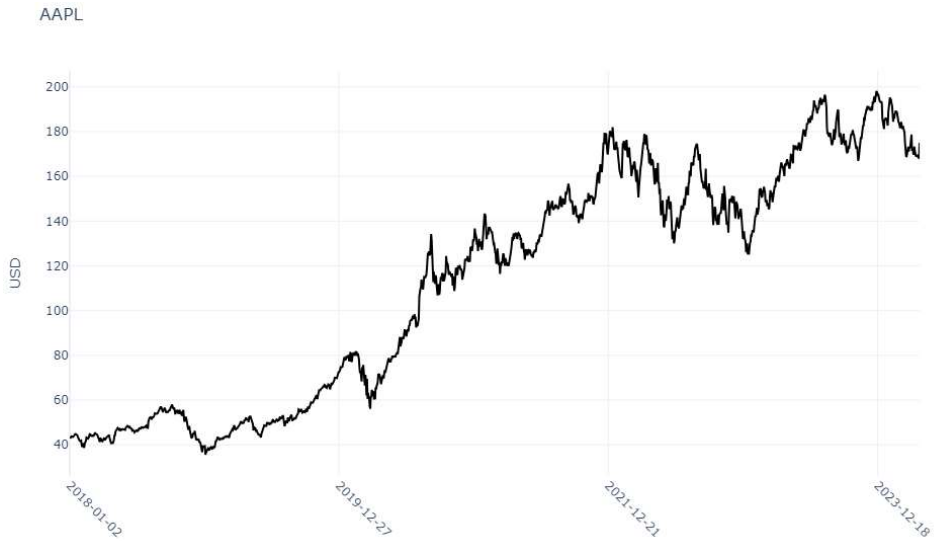
En la práctica se asume que lo anterior es sólo una muestra, pero que la serie pudo haber sido observada en más períodos.

$$\{y_t\}_{t=-\infty}^{\infty} = \{\ldots\ y_{-1}, y_0, y_1, y_2, \ldots, y_T, y_{T+1}, y_{T+2}, \ldots\}$$

En el análisis de series temporales el objetivo es extraer parámetros relevantes o características de ellas. Estas características pueden ser luego utilizadas para generar un modelo matemático que describa la serie y pueda ser utilizado para realizar predicciones.

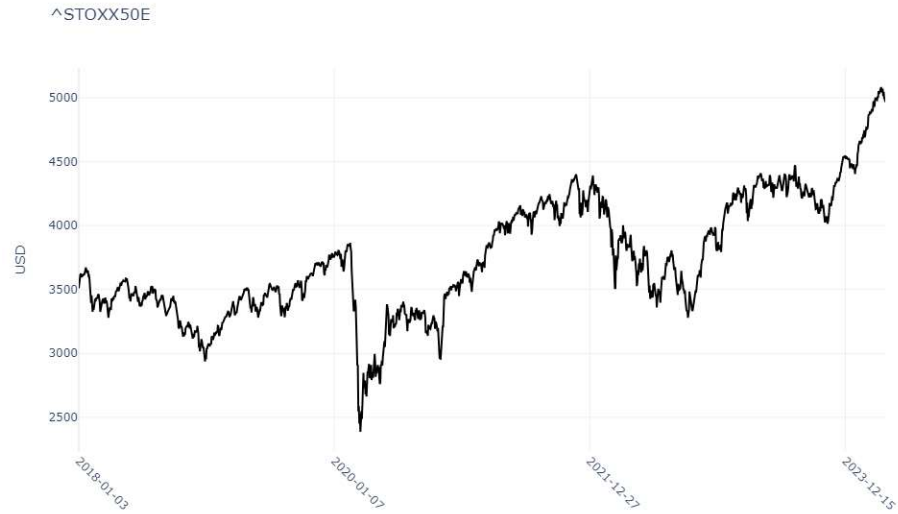
De acuerdo con la cantidad de variables o características que la serie temporal contenga se considera invariada para el caso de una variable, o multivariada para el caso de múltiples variables. De esta fundamental característica depende los pasos a seguir para su análisis y predicción (Tsay, 2005). En este artículo se tratará del caso invariado ya que las series temporales (AAPL y ^STOXX50E) que se utilizarán únicamente interesa el precio de cierre (*Close*) diario de cada índice. En la **Figura 1 y Figura 2** muestran el comportamiento de las series AAPL y ^STOXX50E respectivamente.

Figura 1 Serie histórica del precio de cierre de AAPL.



Fuente: Elaboración propia con datos históricos del índice AAPL.

Figura 2 Serie histórica del precio de cierre de ^STOXX50E



Fuente: Elaboración propia con datos históricos del índice ^STOXX50E.

Redes Neuronales Artificiales

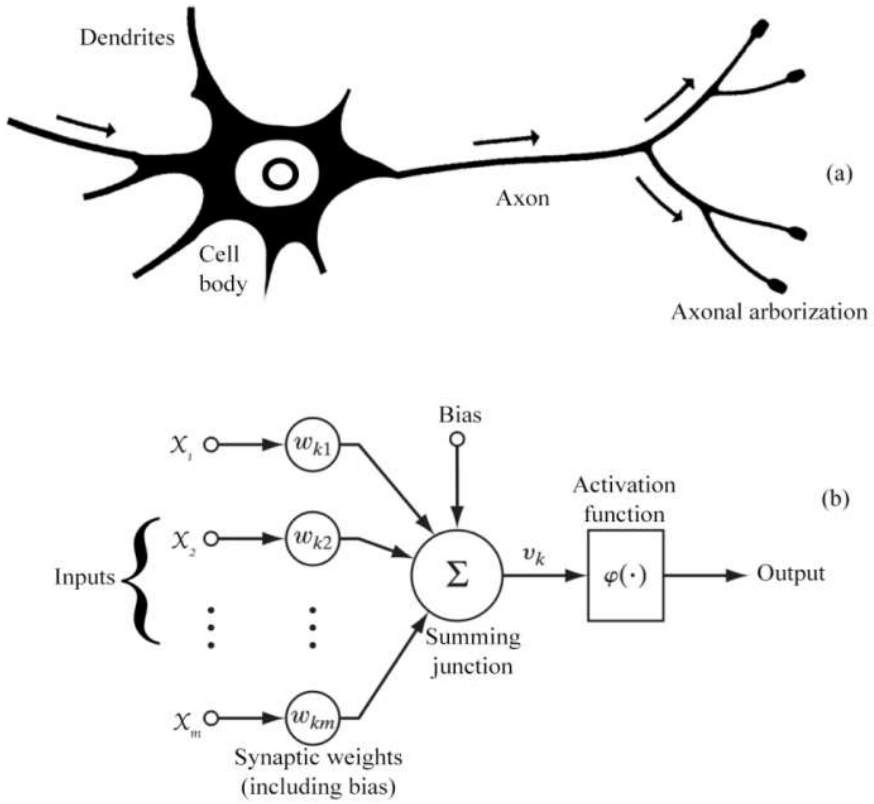
Las Redes Neuronales Artificiales (RNA) se han convertido en estos últimos años en la familia de algoritmos de Deep Learning más populares, aunque desde mediados del siglo pasado ya existían. Sin embargo, no ha sido hasta hace poco con la mejora de los algoritmos y de la tecnología que han crecido el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

Las RNA son sistemas de procesamiento de la información cuya estructura y funcionamiento está inspirado en las redes neuronales biológicas y que pretenden modelar comportamientos inteligentes. En la **Figura 3** se puede ver que las redes neuronales biológicas, las RNA están formadas por muchas neuronas unidas entre si mediante los inputs (Dendrites) y los outputs (Axon) formando las redes neuronales. En el interior de la neurona se procesa la información (Cell body). 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 funcional ni aprenden de la misma manera.

Con las RNA se pretende hacer que un programa aprenda a realizar una tarea sin haber sido previamente programada para ello. Se usan ya que son excepcionalmente buenas en encontrar patrones y modelos en una base de datos (Zhang, 2017). Esto se hace a base de ejemplos. La RNA a base de estos ejemplos debe encontrar unos patrones que le indiquen una solución válida para cuando reciba entradas nuevas.

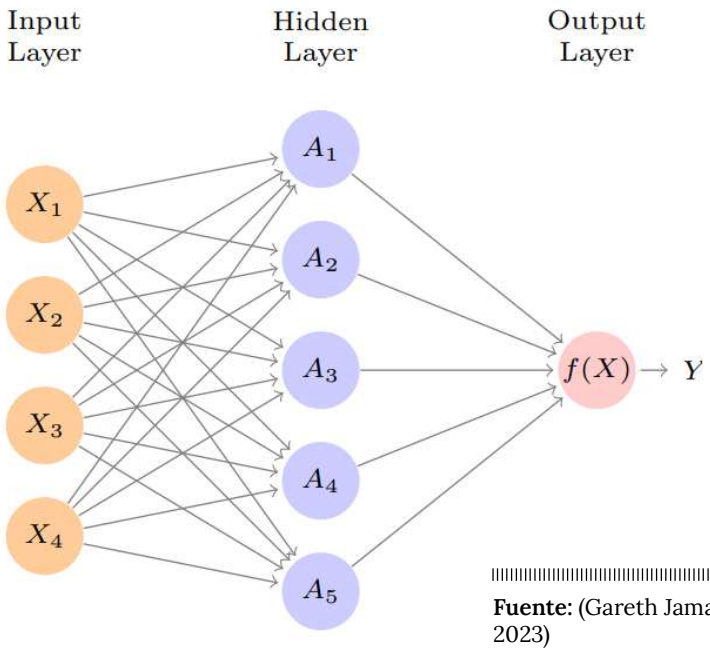
Un ejemplo básico de una RNA se puede apreciar en **Figura 4.** Se puede notar que la RNA tiene una única capa oculta (columna morada/lila). La capa oculta calcula activaciones A_k=h_k (X) que son transformaciones no lineales de combinaciones lineales de las entradas X_1,X_2,...,X_p. Por lo tanto, estas A_k no se observan directamente. La capa de salida es un modelo lineal que usan estas activaciones A_k como entradas, resultando en una función f(X).

Figura 3 Comparativa entre neurona biológica y neurona artificial.



Fuente: (Akgün, 2018)

Figura 4 Red neuronal de una sola capa oculta. Input Layer (capa de entrada), Hidden Layer (Capa oculta) y Output Layer (Capa de salida).



Fuente: (Gareth Jamas, 2023)

Modelo Matemático de las Redes Neuronales Artificiales

De manera general, una red neuronal toma un vector de entrada de p variables $X=(X_1,X_2,...,X_p)$ y construye una función no lineal f(X) para predecir la respuesta Y. La Figura 4 muestra una red neuronal de avance simple para modelar una respuesta cuantitativa utilizando p=4 predictores. En la terminología de las redes neuronales, las cuatro características $(X_1,...,X_4)$ forman las unidades en la capa de entrada. Las flechas indican que cada una de las entradas de la cada de entrada alimentan cada una de las K unidades ocultas (color azul) de las cuales podemos elegir K; aquí elegimos 5. El modelo de red neuronal tiene la forma:

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k h_k(X) = \beta_0 + \sum_{k=1}^{K} \beta_k g(\omega_{k_0} + \sum_{j=1}^{p} \omega_{k_j} X_j)$$

Se construye aquí en dos pasos. Primero las K activaciones A_k,k=1,...,K, en la capa oculta se calculan como funciones de las características de entrada $X_1,...,X_p$,

$$A_k = h_k(X) = g\,(\,\omega_{k_0} + \sum_{j=1}^{p} \omega_{k_j} X_j)$$

Donde g(z) es una función de activación (la cual se definirá posteriormente) no lineal que se especifica de antemano. Se puede pensar en cada A_k como una transformación diferente de h_k (X) de

las características originales, estas activaciones K de la capa oculta alimentan la capa de salida, lo que da como resultado:

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k A_k$$

Un modelo de regresión lineal en las activaciones K=5. Todos los parámetros $\beta_0,...,\beta_k$ y $\omega_{10},...,\omega_{Kp}$ debe estimarse a partir de los datos.

Con base en lo descrito previamente, si se analiza la Figura 4, se observa que la capa oculta calcula activaciones $A_k=h_k(X)$ que son transformaciones no lineales de combinaciones lineales de las entradas $X_1,....,X_p$. Por tanto, estas A_k no se observan directamente. Las funciones $h_k(.)$ no están fijadas de antemano, sino que se aprenden durante el entrenamiento de la red. La capa de salida es un modelo lineal que usa estas activaciones A_k como entradas, resultando en una función f(X).

Función de Activación

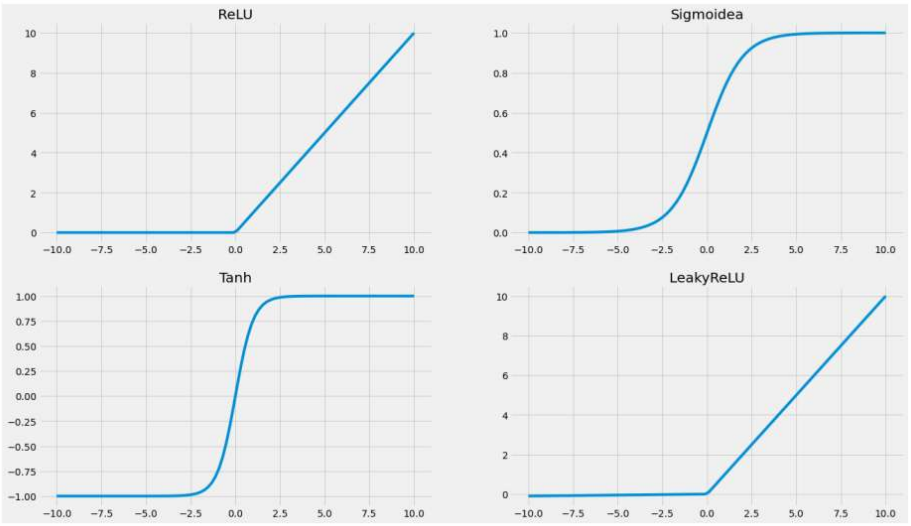
Las funciones de activación son un tipo de función que se agrega a una red neuronal para ayudar a la red a aprender dependencias no lineales complejas. Una función de activación típica debe ser diferenciable y continua en todas partes. La **Tabla 1** muestra algunas funciones de activación de mayor uso.

Tabla 1 **Algunas funciones de activación.**

Función de Activación	Descripción
Función Lineal Rectificadora (ReLU)	$f(x) = f(x) = \{0, \quad x < 0 \; a * x, \quad x \geq 0$
Sigmoidea	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
Tanh	$f(x) = \frac{1}{1 + e^{-x}}$
Leaky ReLU	$f(x) = máx(0, x) == \{0, \quad x < 0 \; x, \quad x \geq 0$

Fuente: Elaboración propia.

Figura 5 Serie histórica del precio de cierre de ^STOXX50E



Fuente: Gráfico de las funciones de activación de la Tabla 1.

Función de Pérdida

La función de pérdida es la función objetivo que describe la diferencia entre las aproximaciones ($\hat{y}_i$) las verdades fundamentales y_i de la muestra dada. Medirá el error de la discrepancia entre cada par ($\hat{y}_i$,y_i) . A partir de dichos errores es posible optimizar los parámetros de la red neuronal.

Existen varias funciones de pérdida, cada una de ellas destinada a una tarea particular (clasificación, regresión, etc.). Para el análisis de series temporales que será el caso de estudio, se suele utilizar 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 del error promedio entre la diferencia de la predicción y de los datos reales y la segunda lo mismo, pero sin la raíz cuadrada.

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

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

Siendo n el número de predicciones.

También existen 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. De todos estos métodos se ha decidido emplear el RMSE por dos razones: la facilidad que se tiene en calcular estos y pro

el respaldo que se tiene en la bibliografía donde RMSE ha sido aplicado por (De Oliveira, 2013), (El-Henawy, 2010), (Huang, 2014) y (Enke, 2005).

Sobreajuste

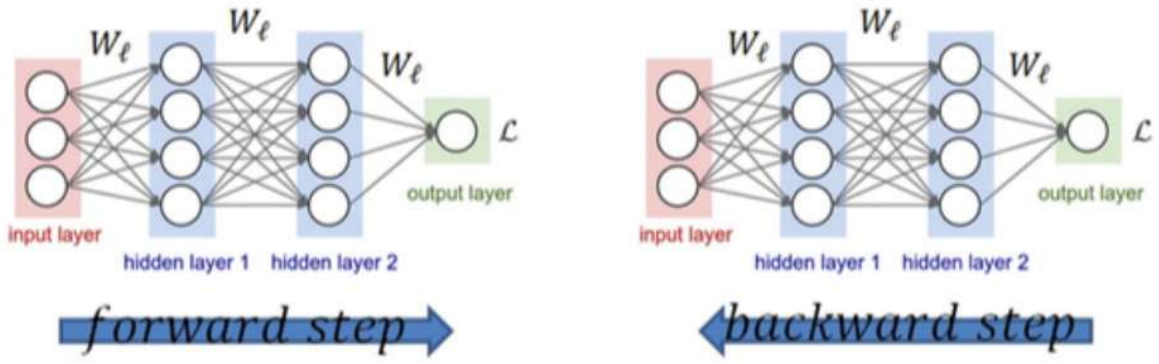
Cuando un método dado produce un error (MSE, RMSE) de entrenamiento pequeño pero un MSE o RMSE de prueba grande, se dice que se están sobre ajustando los datos.

Cuando se sobre ajustan los datos de entrenamiento, el MSE o RMSE de prueba será muy grande dado que los supuestos patrones que el método encontró en los datos de entrenamiento simplemente no existen en dos datos de prueba. Tenga en cuenta que independientemente de si se ha producido o no un sobreajuste, casi siempre se espera que el MSE o RMSE de entrenamiento sea más pequeño que el error de prueba. El sobreajuste se refiere específicamente al caso en el que el modelo menos flexible habría producido un MSE de prueba más pequeño.

Backpropagation (Retropropagación)

Backpropagation es el mecanismo que se utiliza para calcular los parámetros de la red neuronal (pesos ω y sesgos b) a partir de la función de pérdida. Este método es la esencia del entrenamiento de redes neuronales. Es el método de ajuste fino de los pesos de una red neuronal en función de la tasa de error obtenida en una iteración previa. El ajuste de los pesos le permite reducir las tasas de error y hacer que el modelo sea confiable al aumentar su generalización.

Figura 6 Paso hacia adelante (forward step) y hacia atrás (backward step).



- **Forward Step (Paso hacia adelante):** La entrada X (de la capa de entrada) proporciona la información inicial que luego se propaga a las unidades ocultas en cada capa y finalmente produce la salida y^.
- **Backward Step (Paso hacia atrás):** Se utiliza para actualizar los pesos mediante el descenso del gradiente **(ver Figura 7)** usando la ecuación (2). Calcula el gradiente de la función de pérdida con respecto a los pesos de la red neuronal. El cálculo procede hacia atrás a través de la red.

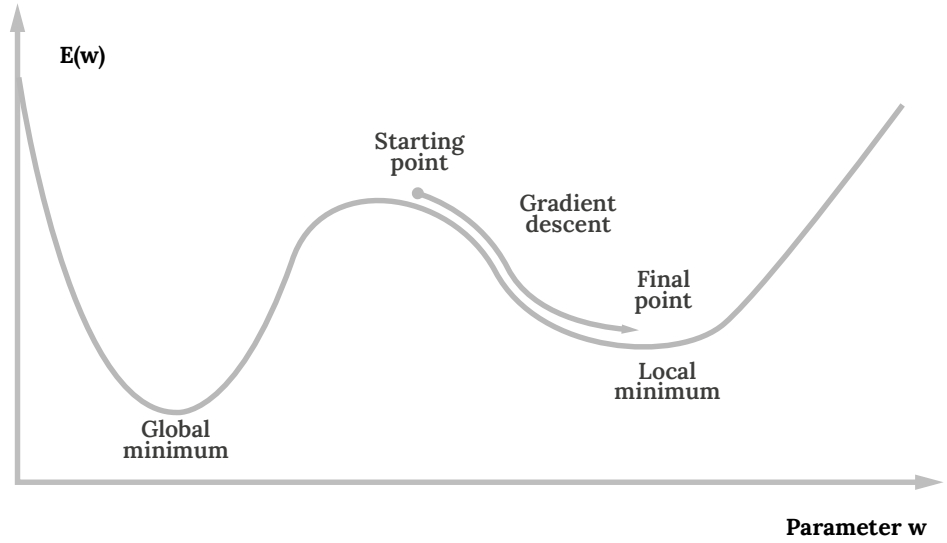
$$\omega_{peso\ actualizado} = \omega_{peso\ anterior} - \alpha_{tasa\ aprendizaje} \cdot \frac{\partial L}{\partial \omega_i} \quad (2)$$

La idea de la tasa de aprendizaje es introducir una constante que permita forzar que los pesos se actualicen de manera suave. Aunque este parámetro suele estar relacionado con una función que realiza esta tarea, denominados optimizadores **(Ver Tabla 2)**.

Formalmente el gradiente es simplemente el vector de las derivadas de la función de error con respecto a los parámetros, es decir:

$$\nabla L_w = \left(\frac{\partial L}{\partial \omega_1}, \frac{\partial L}{\partial \omega_2}, \ldots, \frac{\partial L}{\partial \omega_n}\right)^T$$

Figura 7 **Idea geométrica de descenso del gradiente.**



Fuente: (Torre A. F., 2020)

La **Figura 7** muestra la idea del descenso de gradiente la cual es comenzar en algún punto y luego simplemente caminar cuesta abajo. En este ejemplo hay dos mínimos y el proceso termina en el equivocado. Si hubiese comenzado un poco más a la izquierda, habría encontrado el global. De esta forma se reduce el error de salida de cada neurona al modificar los valores de entrada en la misma, a partir del vector gradiente previamente calculado con el algoritmo de Backpropagation.

Entrenamiento

Una vez diseñado y, por tanto, definido el modelo, es el momento de entrenarlo para poder ver la actualización de este sobre los datos disponibles. Aunque conceptualmente la red neuronal sea impecable a nivel de diseño, hasta que no es probada con los datos no puede deducir lo bien que se ajusta a estos y a la predicción que se espera obtener. Por consiguiente, sería el momento de efectuar el inicio del proceso de aprendizaje de la red. Sin embargo, es necesario definir previamente los siguientes parámetros que definen el modo de realización de este proceso, con el objetivo de maximizar la precisión y minimizar el tiempo de entrenamiento.

Optimizador

Se usa a la hora de entrenar la red neuronal para optimizar la función de pérdida. El valor $\hat{y}$ se obtiene mediante el proceso de propagación hacia adelante y hace uso de los pesos ω y sesgos b de la red. Por tanto, con la ayuda de los algoritmos de optimización se minimiza el valor de la función de pérdida actualizando las variables de entrenamiento ω y b (Torre A. F., 2020).

Iterar hasta que converja

Desde que se actualizan los pesos hasta que la red aprende se necesitan una serie de iteraciones. Por tanto, se puede cuestionar cuantas son necesarias para conseguir la convergencia del modelo. Este hecho depende de muchos factores:

- **La definición de la red** (número de capas, como de complejas son las funciones lineales, etc.). Cuanto mayor sea el número de variables más tiempo tardará en converger, pero la precisión será mucho mayor.
- El método de **optimización** usado, algunas reglas de actualización de la red no están probadas que sean más rápidas que otras.
- La **inicialización** aleatoria de la red. Puede que genere un valor cercano a la solución óptima.
- La **calidad de los datos de entrenamiento**: si no hay ninguna correlación entre los datos de entrada y los de salida, la red será capaz de aprender una correlación aleatoria, por lo que será muy complicado obtener buenos resultados.

Una vez establecido el proceso básico para poder obtener una red funcional, entendiendo este término como un modelo que nos ofrece unos resultados que se ajustan en cuanto a datos y tiempo de ejecución a nuestra problemática y salida esperadas. Por ello, hay que tener en cuenta que la definición de la topología de la red neuronal resulta extremadamente importante para su posterior éxito.

Tabla 2 **Funciones de optimización en RNA.**

Función	Descripción
Descenso de gradiente	$\omega = \omega - \alpha \cdot \frac{\partial L}{\partial \omega_i}$, $b = b - \alpha \cdot \frac{\partial L}{\partial b}$ donde, α es la tasa de aprendizaje (Developers, 2019)
Adam (Adaptative Moment Estimation)	Calcula las tasas de aprendizaje adaptativo para cada parámetro. Combina dos funciones RMSProp y Stochastic Gradient Descent (Developers, 2019)
Adagrad	Algoritmo basado en gradientes que adapta la tasa de aprendizaje a los parámetros.
Adadelta	Es una versión avanzada de Adagrad, busca minimizar su tasa de aprendizaje.

Fuente: Elaboración propia.

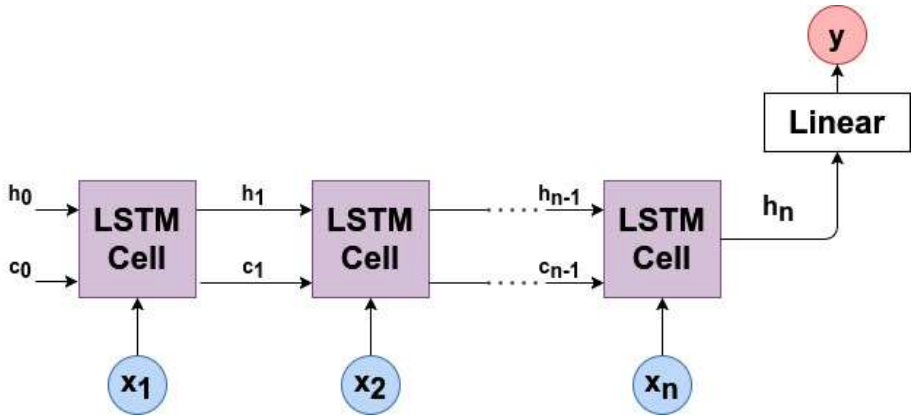
Redes Neuronales Recurrentes (RNN)

Este tipo de red se utiliza ante fuentes de datos que son de naturaleza secuencial y requieren un tratamiento especial al crear un modelo predictivo. Las RNN tienen un concepto de estado oculto. Un estado oculto puede ser tratado como memoria interna. El estado oculto no intenta recordar todos los valores pasados de la secuencia sino sólo su efecto. Las RNN sufren de memoria a corto plazo. Es decir, si una secuencia es lo suficientemente larga, le resultará difícil enviar información de pasos anteriores a los posteriores. La Red de Memoria de Corto y Largo Plazo (LSTM) y Red de Unidades Recurrentes Cerradas (GRU) se crearon como solución a la memoria a corto plazo. Tienen mecanismos internos llamados puertas que pueden regular el flujo de información tal como se aprecia en la **Figura 8**.

Red de Memoria de Corto y Largo Plazo (LSTM)

Las redes LSTM (Long Short – Term Memory) son un tipo especial de redes neuronales recurrentes diseñadas con celdas de memoria que mantienen su estado a largo plazo. Globalmente, el flujo computacional de la red LSTM se puede apreciar en **Figura 8**.

Figura 8 **Flujo computacional de la red LSTM**



Fuente: (Gridin, 2022)

Las RNN pasó sólo un estado oculto h_t a través de cada iteración. Pero LSTM pasa dos vectores h_t- estado oculto (memoria a corto plazo) y c_t - estado celular (memoria a largo plazo).

Las salidas de la celda LSTM se calculan a través de las fórmulas que se muestran a continuación:

$$
\begin{aligned}
i_t &= \sigma(\omega_{ii}x_t + b_{ii} + \omega_{hi}h_{(t-1)} + b_{hi}) \\
f_t &= \sigma(\omega_{if}x_t + b_{if} + \omega_{hj}h_{(t-1)} + b_{hf}) \\
g_t &= tanh(\omega_{ig}x_t + b_{ig} + \omega_{hg}h_{(t-1)} + b_{hn}) \\
o_t &= \sigma(\omega_{io}x_t + b_{io} + \omega_{h0}h_{(t-1)} + b_{ho}) \\
c_t &= f_t \circ c_{t-1} + i_t \circ g_t \\
h_t &= o_t \circ tanh(c_t)
\end{aligned}
$$

donde:

σ es la función sigmoidea.

∘ es el producto de Hadamard, que es

$$[a_1\ a_2\ a_3\ ] \circ [b_1\ b_2\ b_3\ ] = [a_1 b_1\ a_2 b_2\ a_3 b_3\ ]$$

En cuanto a las variables se considera:

- i_t ***(puerta de entrada)*****:** es la variable que se utiliza para actualizar el estado de la celda c_t. El estado previamente oculto c_t y la entrada secuencial actual x_t se dan como entrada a una función sigmoidea. Si la salida está cerca a 1, más importante es la información.
- f_t ***(puerta de olvido)*****:** variable que decide qué información debe olvidarse en el estado de celda c_t. El estado previamente oculto h_t y la entrada de secuencia x_t se dan como entradas a una función sigmoidea. Si la salida f_t está cerca de cero, entonces la información puede

olvidarse, mientras que, si la salida está cerca de uno, la información debe almacenarse.

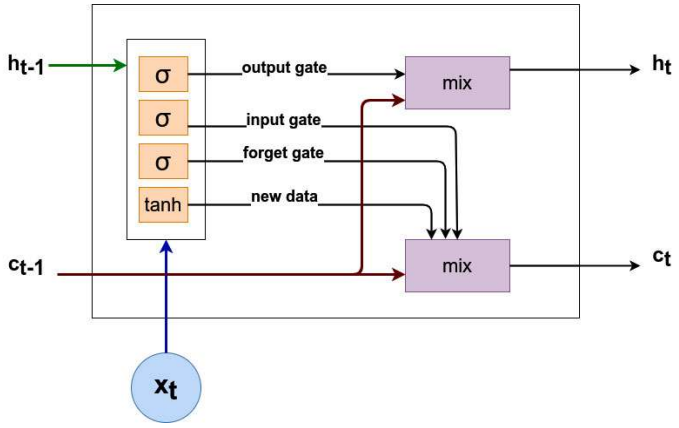
- g_t: representa información importante potencialmente nueva para el estado celular c_t.
- c_t: es una suma de:
    - Estado de celda anterior $c_{(t-1)}$ con alguna información olvidada f_t.
    - Nueva información de g_t.
- o_t ***(puerta de salida)***: variable para actualizar el estado oculto h_t.
- h_t ***(estado oculto)***: es el siguiente estado oculto que se calcula seleccionando la información importante o_t del estado de celda c_t.

La red LSTM tiene los siguientes parámetros, que se ajustan durante el entrenamiento:

$$\omega_{ii}, \omega_{hi}, \omega_{if}, \omega_{hf}, \omega_{ig}, \omega_{hg}, \omega_{io}, \omega_{ho} \rightarrow pesos$$

$$b_{ii}, b_{hi}, b_{if}, b_{hf}, b_{ig}, b_{io}, b_{ho} \rightarrow Sesgos$$

Figura 9 **Gráfico computacional de la celda de red LSTM.**



Fuente: (Gridin, 2022)

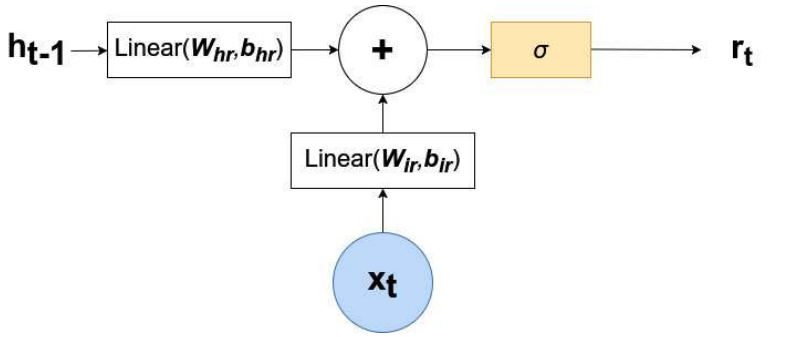
Unidades Recurrentes Cerradas (GRU)

GRU es esencialmente una red LSTM simplificado. Tiene exactamente el mismo papel. La principal diferencia está en el número de puertas y pesos: GRU es algo más simple. Las siguientes identidades ayudaran a comprender el funcionamiento de GRU.

$$r_t = \sigma(w_{ir}x_t + w_{hr}h_{t-1} + b_{hr})$$
$$z_t = \sigma(w_{iz}x_t + b_{iz} + w_{hz}h_{t-1} + b_{hz})$$
$$n_t = tanh(w_{in}x_t + b_{in} + r_t \circ (w_{hn}h_{t-1} + b_{hn}))$$
$$h_t = (1 - z_t) \circ n_t + z_t \circ h_{t-1}$$

donde, σ es la función sigmoidea y ∘ es el producto de Hadamard.

Figura 10 **Gráfico computacional de variable r_t**

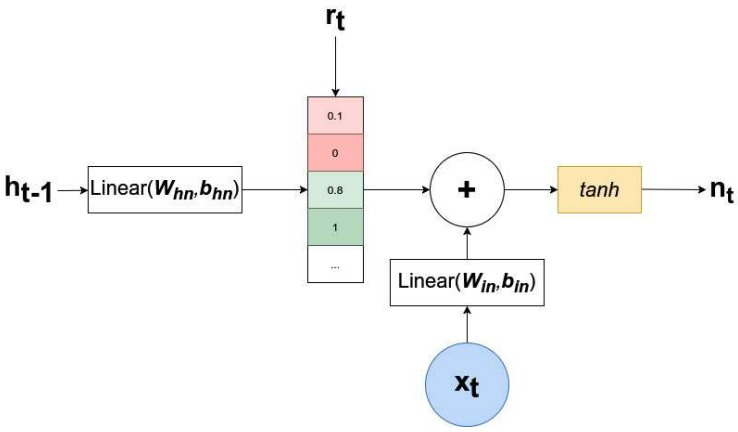


Fuente: (Gridin, 2022)

r_t Es una salida de una red neuronal simple en dos vectores de entrada h_t y x_t tal como se muestra en **Figura 10**. La variable r_t es responsable de olvidar datos no importantes en el estado oculto de GRU.

La **Figura 11** ayuda a considerar el caso para n_t. El cual representa el vector de estado oculto basado en $h_{(t-1)}$, que restablece algunos valores anteriores. Exactamente la variable n_t proporciona almacenamiento de información a largo plazo porque la variable r_t no siempre permite cambiar algo en el estado oculto $h_{(t-1)}$.

Figura 11 **Gráfico computacional de variable n_t.**

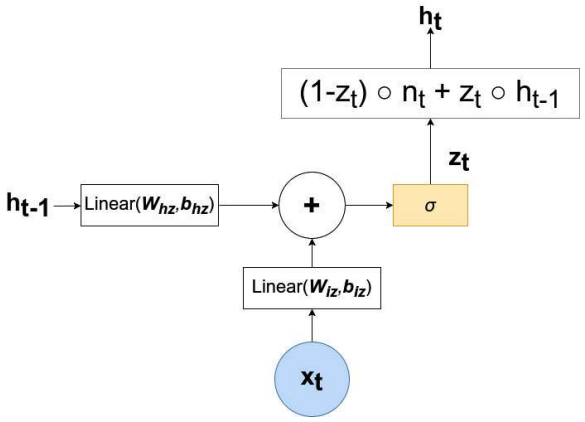


Fuente: (Gridin, 2022)

La **Figura 12** muestra el gráfico computacional para las variables z_t y h_t. Note que h_t es una combinación lineal simple de dos estados ocultos: $h_{(t-1)}$ estado oculto previo y n_t estado oculto candidato con memoria a largo plazo. Y la variable z_t decide en que proporción mezclarlos.

Por tanto, podemos concluir que GRU selecciona características a largo plazo r_t y forma un estado oculto con memoria n_t a largo plazo y lo mezcla con el estado $h_{(t-1)}$ previamente oculto con relación z_t.

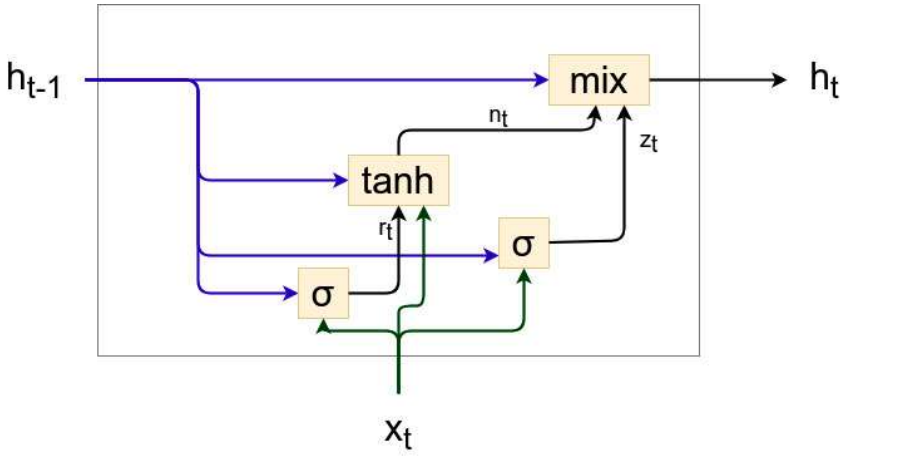
Figura 12 **Gráfico computacional de variable n_t.**



Fuente: (Gridin, 2022)

Por último, y con base en lo descrito previamente la Figura 13 muestra el gráfico computacional de GRU.

Figura 13 **Gráfico computacional de GRU.**



Fuente: (Gridin, 2022)

GRU tiene los siguientes parámetros, que se ajustan durante el entrenamiento:

$$w_{ir}, w_{hr}, w_{iz}, w_{in}, w_{hn} \rightarrow \text{pesos.}$$

$$b_{ir}, b_{hr}, b_{iz}, b_{hz}, b_{in}, b_{hn} \rightarrow \text{sesgos.}$$

Fuentes de datos

Los datos del histórico de AAPL (Apple inc. Empresa tecnológica más reconocida a nivel mundial no sólo por sus productos sino también por sus acciones financieras) y ^STOXX50E (índice bursátil que representa a las 50 mayores empresas de la eurozona en términos de capitalización bursátil) fueron obtenidos de *Yahoo! Finanzas*. Esta página ofrece información financiera, cotizaciones de la 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 que el objetivo del presente artículo 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 la función *read_csv*() de Python. Así quedan las siguientes columnas: Date (la fecha del día), Open (el valor del índice al abrir el día), High (el máximo al que llego el índice), Low (el mínimo al que llego el índice 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). Ver ejemplo en **Tabla 3**.

Una vez determinada las fechas (2018-01-01 a 2024-04-12) entre las que se iba a trabajar se debía realizar el tratamiento de la data. Entre fechas había muchos valores que tenían *NaN (Not a Number, son valores vacíos no computables*). Estos valores podían aparecer por fines de semana o fiestas nacionales en los que AAPL o ^STOXX50E no estaba abierto, pero sí estaba puesta la fecha.

Tabla 3 **Ejemplo de los primeros datos del fichero de base.**

Date	Open	High	Low	Close	Adj Close	Volumen
2018-01-02	42.540001	43.075001	42.314999	43.064999	40.670979	102223600
2018-01-03	43.122500	43.637501	42.990002	43.057499	40.663891	118071600
2018-01-04	43.134998	43.367500	43.020000	43.257500	40.852787	89738400
2018-01-05	43.360001	43.842499	43.262501	43.750000	41.317894	94640000
2018-01-08	43.587502	43.902500	43.482498	43.587502	41.164433	82271200

Elaboración propia con datos de AAPL de Yahoo! Financie.

División de los datos

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

Para la división de los 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 prueba, 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 (train) y el 30% y 20% para los datos de prueba. Para este trabajo se ha decidido elegir una división del 80% - 20%. Todo esto apoyándose en el trabajo de (Vanstone, 2009).

Los índices APPL y ^STOXX50E no sólo son una herramienta de análisis para los inversionistas que desean invertir en ellos, sino que también son una estadística de la situación en un momento determinado de tiempo. Por ello, son seguidos por los gestores de carteras, traders, hedge funds, más importantes del mundo.

IV. Discusión

En la siguiente tabla se observan los resultados obtenidos. Se han dividido tal que se pueda apreciar cada uno de los tipos de red neuronal estudiados (LSTM y GRU). Dentro de la tabla se ha dividido los dos modelos propuestos. Estas arquitecturas, luego se han dividido por bases de datos. Por cada archivo de datos se ha remarcado la cantidad de capas ocultas utilizadas, el número de epoch y la tasa de aprendizaje. Tras ello se detalla el MSE obtenido al comparar la predicción y el valor real en los datos de prueba con los hiperparámetros anteriores.

En la **Tabla 4** vemos los resultados del entrenamiento de LSTM y GRU, respecto a las bases de datos empleadas.

Tabla 4 **Resultados obtenidos de las redes neuronales recurrentes LSTM y GRU.**

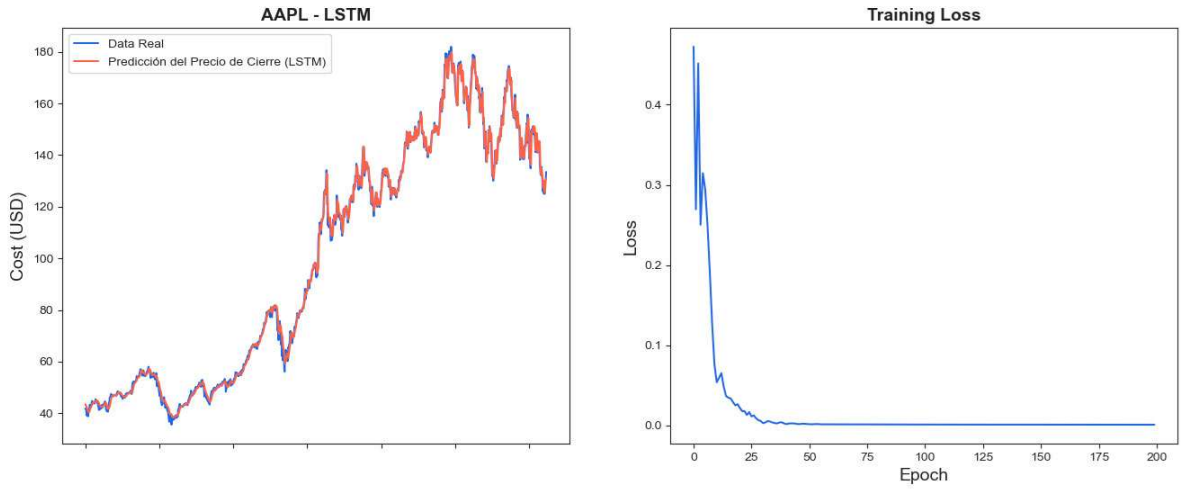
Tipo RNN	índice bursátil	Capas ocultas	Numero epoch	Tasa aprendizaje	Train MSE	Test MSE
LSTM	AAPL	30	200	0.03	2.38	2.69
		24	200	0.03	2.33	2.90
		20	200	0.03	2.39	2.89
		50	200	0.02	2.33	2.77
		50	200	0.04	2.76	12.54
GRU	AAPL	30	200	0.03	2.23	2.79
		24	200	0.03	2.23	2.95
		20	200	0.03	2.24	2.98
		50	200	0.02	2.23	2.64
		50	200	0.04	2.27	2.96
LSTM	^STOXX50E	30	200	0.03	44.91	139.33
		24	200	0.03	44.12	72.72
		20	200	0.03	44.29	69.77
		50	200	0.02	44.94	69.64
		50	200	0.04	44.80	109.02
GRU	^STOXX50E	30	200	0.03	43.54	75.32
		24	200	0.03	43.86	93.40
		20	200	0.03	43.82	88.11
		50	200	0.02	42.88	193.22
		50	200	0.04	43.81	39.60

Fuente: Elaboración propia con los resultados obtenidos del entrenamiento de LSTM y GRU

De la **Figura 14 y Figura 16** se pueden observar las mejores y peores predicciones en ambos modelos y para ambos índices bursátiles.

De la **Tabla 4** se puede notar que si se comienza a realizar variaciones en los hiperparámetros se podrían encontrar resultados óptimos en cuanto a las predicciones. En este caso se enfoca en variar la tasa de aprendizaje y la cantidad de capaz ocultas en la red neuronal, a la cual se puede notar que si son hiperparámetros bastante sensibles y se obtienen mejores resultados. Como se puede apreciar los resultados con un Train MSE de 2.38 y Test MSE de 2.69 para el índice de AAPL usando la red LSTM, dicho resultado está sujeto a un número de capaz ocultas de 30 y una tasa de aprendizaje de 0.03. Con GRU para predicción de AAPL se obtienen resultados bastante similares con un Train MSE de 2.23 y Test MSE de 2.64.

Figura 14 **Gráfica real (azul) vs predicciones (rojo) LSTM para los datos AAPL, mejor resultado.**



Fuente: Elaboración propia.

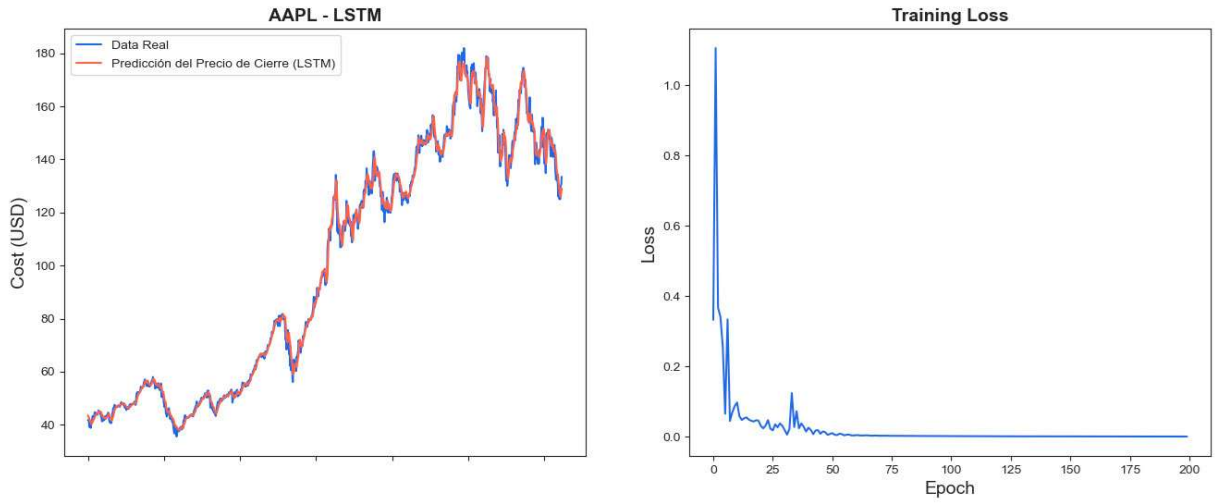
Figura 15 **Visualización de Valores Reales, Train Predicciones y Test Predicciones sobre mejor resultado de LSTM con datos de AAPL.**

Fuente: Elaboración propia en Python.

Como se observa en la **Figura 16** se obtuvo uno de los peores resultados siempre con LSTM para los datos AAPL, donde se obtuvo un Train MSE de 2.76 y Test MSE de 12.54 (**ver Tabla 4**). Dicho resultado se dio cuando se aumentó el número de capaz ocultas y la tasa de aprendizaje se pasó de 0.03 a 0.04.

Figura 16

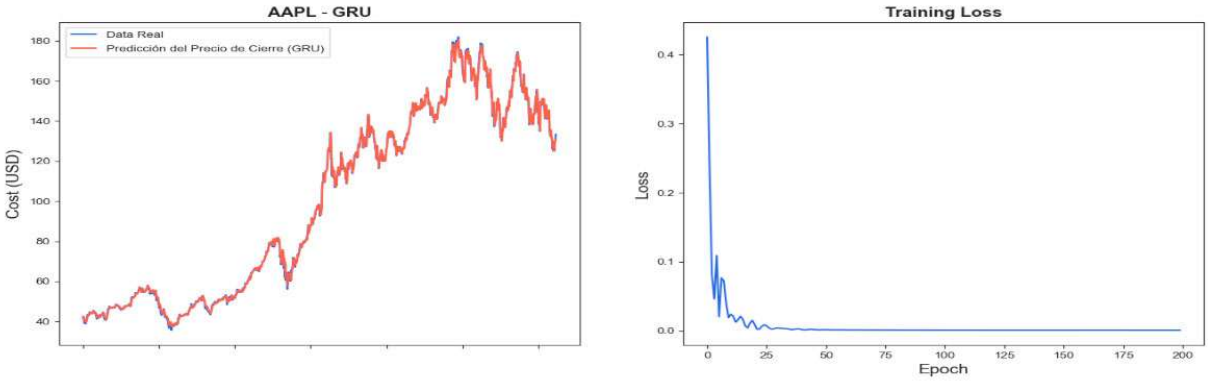
Peor resultado LSTM con datos de AAPL, datos reales (curva azul) vs predicciones (curva roja).



Fuente: Elaboración propia.

Figura 17

Gráfica real (azul) vs predicciones (rojo) en GRU para la base de datos AAPL. Mejor resultado



Fuente: Elaboración propia.

Figura 18 **Visualización de Valores Reales, Train Predicciones y Test Predicciones sobre mejor resultado de GRU con datos de AAPL.**

Fuente: Elaboración propia en Python.

En caso del modelo GRU y como se puede apreciar en la **Tabla 4** no se obtuvo mucha diferencia en cuanto al Test MSE respecto al Train MSE, tal como si ocurrió con el LSTM en el último caso de estudio con 50 capas ocultas y una tasa de aprendizaje de 0.04 donde se obtuvo un Test MSE de 12.54.

V. Conclusiones

Las redes neuronales artificiales tienen una aplicación práctica en el mercado bursátil sobre los modelos estadísticos tradicionales porque no dependen de supuestos teóricos sobre los que se basan las técnicas estadísticas (normalidad, homocedasticidad, independencia, etc.).

Tras el entrenamiento de los modelos planteados, se pudo observar que ambos modelos propuestos siguen un ajuste semejante al de las observaciones reales.

Las redes neuronales recurrentes funcionan muy bien para la predicción de índices bursátiles. Sin embargo, se podrían continuar optimizando los hiperparámetros para reducir la diferencia de las funciones de pérdidas.

El uso exitoso de las redes neuronales artificiales para el pronóstico del precio de las acciones del mercado bursátil demuestra su aplicabilidad en los diferentes mercados. Se concluyen las ventajas de las redes neuronales al ser modelos más sencillos de implementar y permitir obtener bajos errores en el pronóstico.

A pesar de que las aplicaciones mostradas se dieron en acciones diferentes las cuales pueden ser de sectores diferentes, se encontró que prácticamente una misma estructura de red neuronal utilizando sólo la serie de precios, puede representar de forma confiable las dos acciones utilizadas.

IV. Referencias

Akgün, E. &. (2018). Modeling Course Achievements of Elementary Education Teacher Candidates with Artificial Neural Networks. *International Journal of Assessment Tools in Education*, https://dergipark.org.tr/en/pub/ijate/issue/37036/444073.

Breiman, L. (2001). *Statistical Modeling: The two cultures. Institute of Mathematical Statistics.*

Davis, P. J. (1987). *Time Series: Theory and Methods.*

De Oliveira, F. A. (2013). Applying Artificial Neural Networks to prediction of Stock price and improvement of the directional prediction index - case study of PETRO, petrobras, Brazil. *Expert Systems with Applicationes*, https://www.sciencedirect.com/science/article/abs/pii/S0957417413004703.

Developers, G. (2019). *Optimization Algorithms for Deep Learning*. Obtenido de medium.

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

Enke, D. &. (2005). The use of data mining and neural networks for forecasting stock market returns. *Expert Systems with Applications*, https://www.sciencedirect.com/science/article/abs/pii/S0957417405001156.

Garet Jammes, D. W. (2021). *An Introduction to Statistical Learning.*

Gareth Jamas, D. W. (2023). *An Introduction to Statistical Learning*. https://link.springer.com/book/10.1007/978-3-031-38747-0.

George E.P. Box and Gwilym M. Jenkins. (1970). *Time Series Analysis: Forecasting and Control.* Canada.

Gridin, I. (2022). *Time Series Forecasting Using Deep Learning.*

Hochreiter, J. S. (1997). *Long Short-Term Memory.*

Hopkins, O. a. (2020). *Language models are few-shot learners.*

Huang, C. Y. (2014). *Application of integrated data mining techniques in stock market forecasting.* Cogent Economics and Finance, https://www.tandfonline.com/doi/full/10.1080/23322039.2014.929505.

Jenkins, G. E. (1970). *Time Series Analysis: Forecasting and Control*. Canada: http://repo.darmajaya.ac.id/4781/1/Time%20Series%20Analysis_%20Forecasting%20and%20Control%20%28%20PDFDrive%20%29.pdf.

Mitchell, T. M. (1997). *Machinne Learning*.

Pitts, W. S. (1990). A *logical calculus of gthe ideas immanent in nervous activity*.

Reinsel, G. C. (2020). *Elements of Multivariate Time Series Analysis*. New York.

Rudin, J. R. (2019). *Why are we using black box models in ai when we dont need to a lesson from an explainable ai competition*.

Strazicich, J. L. (2002). *Software review: ITMS 2000 professional version* 6.0.

Tom M, M. (1997). *Machinne Learning. McGraw.Hill Science/Engineering/Math*.

Torre, A. F. (2020). *Portability and Optimization of a Neural network for rapid damage detection in earthquakes using OpenVINO toolkit*.

Trevor Hastie Robert Tibshirani, J. F. (2017). *The Elements of Statistical Learning*.

Tsay, R. S. (2005). *Analysis of Financial Time Series*.

Vanstone, B. &. (2009). Efficient market hypothesis and forecasting. *International Journal of Forecasting*, https://www.sciencedirect.com/science/article/abs/pii/S0169207003000128.

Werbos, P. (1990). *Backpropagation through time: what it does and how to do it*.

Zhang, L. A. (2017). Stock price prediction via discovering multifrequency trading patterns. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (pág. https://doi.org/10.1145/3097983.3098117).