UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Facultad de Administración y Dirección de Empresas /UPV

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Facultad de Administración y Dirección de Empresas

Predicción de la volatilidad de acciones bursátiles: Modelo GARCH vs Red neuronal LSTM

Trabajo Fin de Grado

Grado en Administración y Dirección de Empresas

AUTOR/A: Méndez López, Raúl

Tutor/a: Oliver Muncharaz, Javier

CURSO ACADÉMICO: 2023/2024

Agradecimientos

En primer lugar, quiero agradecer a mis padres por su constante apoyo y confianza. A mi pareja, por su comprensión, paciencia y por ser mi pilar fundamental en los momentos difíciles. Gracias por estar siempre a mi lado, apoyándome y animándome a alcanzar mis metas. Y a mi familia, por su cariño incondicional y por siempre creer en mí.

En segundo lugar, a mi tutor Javier Oliver, por su dedicación, conocimiento y paciencia. Gracias por su constante disponibilidad, por su capacidad para resolver mis dudas y por su confianza en mi trabajo.

Por último, quiero agradecer a mis amigos y compañeros de clase por su apoyo y por compartir este camino conmigo.

A todos ellos, muchas gracias.

Resumen

En el presente trabajo se realiza una comparación entre dos metodologías para la predicción de la volatilidad de las acciones bursátiles: el modelo GARCH (*Generalized Autoregressive Conditional Heteroskedasticity*) y la red neuronal recurrente LSTM (*Long Short-Term Memory*). Para ello, se analiza la volatilidad de diferentes acciones bursátiles, mediante la programación de los modelos con el lenguaje de Python. En primer lugar, se analiza las características y estadísticas descriptivas de la volatilidad diaria de las acciones seleccionadas en este trabajo. En segundo lugar, se describen de forma resumida, los modelos a evaluar. Por último, una vez estimados cada uno de ellos, se calculan las correspondientes predicciones de la volatilidad diaria y se evalúa la precisión y eficacia de cada modelo mediante medidas del error de predicción como el MAPE (*Mean Absolute Percentage Error*) y el RMSE (*Root Mean Square Error*).

Abstract

The objective of this work is to perform a comparison between two time series prediction models used in the financial field: the GARCH (*Generalized Autoregressive Conditional Heteroskedasticity*) model and the LSTM (*Long Short-Term Memory*) model. The research aims to evaluate and compare the performance of these two models over a certain period of time, using programming techniques and data analysis in Python. The theoretical concepts are developed, data from different companies will be gathered, and both models will be implemented in Python. Finally, the accuracy and effectiveness of each model will be evaluated using evaluation measures such as MAPE (*Mean Absolute Percentage Error*) and RMSE (*Root Mean Square Error*).

Resum

En el present treball es realitza una comparació entre dues metodologies per a la predicció de la volatilitat de les accions borsàries: el model GARCH (*Generalized Autoregressive Conditional Heteroskedasticity*) i la xarxa neuronal recurrent LSTM (*Long Short-Term Memory*). Per a això, s'analitza la volatilitat de diferents accions borsàries, mitjançant la programació dels models amb el llenguatge de Python. En primer lloc, s'analitzen les característiques i estadístiques descriptives de la volatilitat diària de les accions seleccionades en aquest treball. En segon lloc, es descriuen de forma resumida els models a avaluar. Finalment, una vegada estimats cadascun d'ells, es calculen les corresponents prediccions de la volatilitat diària i s'avalua la precisió i eficàcia de cada model mitjançant mesures de l'error de predicció com ara el MAPE (*Mean Absolute Percentage Error*) i el RMSE (*Root Mean Square Error*).

Índice de contenido

Índice de figuras

Índice de tablas

Índice de ecuaciones

1. Introducción y objetivos

1.1 Introducción

En el ámbito financiero, para los inversores, es importante tener herramientas y modelos que puedan predecir tendencias y precios futuros de los activos en los que invierte, con el objetivo de reducir el riesgo asociado y aumentar la rentabilidad que percibe por estos.

La volatilidad se conoce como el cambio de valor o variación que sufre la rentabilidad de los activos en la bolsa, si esta aumenta, también aumenta el riesgo de la inversión, ya que la posibilidad de que fluctúe ese valor es más alta. Es por esto por lo que, a lo largo de los años se han desarrollado diversos modelos y técnicas con el objetivo de proporcionar información sobre las futuras tendencias del mercado y ayudar a tomar decisiones de inversión con más detalle.

Existen diferentes mercados de valores, sin embargo, hay uno que destaca por albergar a las principales empresas de tecnología, como Apple, Microsoft, Amazon, Meta, Google (a través de Alphabet) o Tesla. Estas empresas han tenido un gran crecimiento a raíz de la pandemia de la COVID-19. El mercado en el que operan es conocido como NASDAQ (*National Association of Securities Dealers Automated Quotation*), nació a principio de los años setenta en el Congreso de los Estados Unidos ante la falta de transparencia de los mercados y hoy en día opera en más de 20 mercados de todo el mundo [12].

El modelo econométrico clásico de predicción de series temporales más conocido es el modelo ARIMA, estos se centran en la estacionariedad y no tienen en cuenta la volatilidad. Más adelante, surgieron los modelos ARCH, los cuales, tienen en cuenta diferentes características asociadas a la volatilidad, especialmente la extensión GARCH, la cual permite una modelización más precisa de la volatilidad al incorporar componentes de media móvil al mismo tiempo que tiene en cuenta la autocorrelación de la serie, y es el que se desarrollará en el presente trabajo.

Sin embargo, en los últimos años el uso de las redes neuronales para predecir series temporales ha sido muy exitosa, tanto que algunos modelos econométricos clásicos se están dejando de utilizar. El tipo de red neuronal ideal para predecir acciones bursátiles es la red neuronal recurrente LSTM (*Long Short-Term Memory*), este tipo de red es la que se desarrollará más adelante y cuyos resultados se compararán con el modelo clásico GARCH, con el objetivo de concluir que modelo resulta más acertado.

1.2 Objetivos

El objetivo principal de este trabajo es realizar una comparación exhaustiva entre el modelo GARCH y la red neuronal LSTM. Se plantean los siguientes objetivos:

1. Evaluar el rendimiento del modelo GARCH y de la red LSTM en la predicción de la volatilidad diaria, semanal y mensual del precio de cierre de diferentes empresas, utilizando sus datos históricos.
2. Comparar y analizar los resultados obtenidos respecto a cada empresa en cuanto a su precisión y eficacia utilizando medidas de comparación como el RSME o el MAPE.
3. Identificar las diferencias, fortalezas y limitaciones de cada modelo.

Al alcanzar estos objetivos, se pretende contribuir al conocimiento y comprensión de las capacidades y limitaciones de los modelos GARCH y de la red neuronal LSTM, brindando a los inversores y profesionales del mercado una base sólida para la toma de decisiones financieras.

1.3 Estructura del trabajo

En el presente apartado se ha proporcionado una visión general del proyecto, donde se ha introducido el tema y se han presentado los objetivos que se pretenden alcanzar.

En el segundo apartado se establecen los fundamentos teóricos necesarios para comprender los modelos de predicción utilizados en el estudio, introduciendo en primer lugar el modelo ARIMA, seguidamente los modelos de la familia ARCH, en el que se encuentran el modelo GARCH y el EGARCH. Finalmente, se hace una introducción a las redes neuronales, se explica la red LSTM y se presentan los conceptos teóricos de las diferentes medidas del error de predicción que se utilizarán para comparar ambos modelos.

En el tercer apartado, en primer lugar, se realiza un estudio analítico de los datos, así como se detallan las empresas que forman el conjunto de datos, el periodo muestral o el entorno de programación elegido. A continuación, se explica cómo se han implementado los diferentes modelos en Python, describiendo los parámetros óptimos de cada uno.

En el cuarto apartado se analizan los resultados de ambos modelos, se comparan y se discute sobre qué modelo ha resultado el más acertado para cada tipo de volatilidad.

En el quinto apartado se proponen dos alternativas de mejora. En la primera se compara la eficacia de cada modelo ante diferentes valores de ventana. Mientras que en la segunda se simulan las predicciones con el modelo EGARCH.

En el sexto apartado se resumen las principales conclusiones del trabajo y se introducen varias líneas de estudio futuras.

Finalmente, en el séptimo apartado se describen los ODS ligados al trabajo y en el octavo apartado se muestran las referencias bibliográficas utilizadas.

Además, el presente Trabajo Fin de Grado consta de tres Anexos. El primero hace referencia a los objetivos de desarrollo sostenible, en el segundo se incluyen todas las gráficas de los resultados obtenidos por ambos modelos y en el tercero se incluyen los resultados de la simulación del modelo EGARCH del quinto apartado.

2. Metodología

En este apartado se proporciona una breve descripción sobre las acciones bursátiles y se explora el concepto de volatilidad en el contexto financiero, se establecen los fundamentos teóricos necesarios para comprender el modelo GARCH, introduciendo en primer lugar el modelo ARCH y el ARIMA. Además, se introducirá el funcionamiento de una red neuronal y de la red LSTM, así como las medidas de comparación que se utilizarán para evaluar el rendimiento de ambos modelos.

2.1 Acciones bursátiles y volatilidad

Las acciones bursátiles son activos financieros que representan un pequeño porcentaje de participación en una empresa. Cotizan en el mercado o bolsa de valores y la suma de todas ellas representa el valor económico de la compañía. Los inversores compran y venden acciones con la expectativa de obtener beneficios a través de la apreciación del valor de las acciones o de los posibles dividendos.

El mercado de valores es conocido por su dinamismo y su volatilidad. La mayoría de los mercados de valores operan en días hábiles regulares de lunes a viernes, excluyendo los días festivos. Cada mercado tiene su horario de cotización y está sujeto al calendario festivo de cada país en cuestión, aunque, se suele establecer de media un número de 252 días hábiles de negociación en un año y de 21 días en un mes.

La cotización de una acción es la tasación diaria de un activo cuyo valor varía en función de las órdenes de compra y venta que haya tenido ese activo a lo largo de la jornada de negociación. El precio o cotización de la acción al finalizar la jornada se le conoce como precio de cierre y es el valor que se suele tomar como referencia en finanzas a la hora de realizar predicciones.

Una medida importante en el ámbito financiero es la rentabilidad o rendimiento de un activo. Esta se refiere al porcentaje de variabilidad de un activo en un tiempo determinado y se expresa como:

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}} \tag{1}$$

Ecuación 1. *Expresión de la rentabilidad simple*

Sin embargo, en finanzas se suele utilizar la rentabilidad logarítmica en vez de la simple, ya que presenta varias ventajas como la normalización o la estacionariedad [2], y su expresión matemática es la siguiente:

$$R_t = \ln\left(\frac{P_t}{P_{t-1}}\right) = \ln(P_t) - \ln(P_{t-1}) \tag{2}$$

Ecuación 2. *Expresión de la rentabilidad logarítmica*

Siendo R_t el rendimiento del activo en el tiempo t, P_t el precio de cierre en el tiempo t y P_{t-1} el precio de cierre en un periodo de tiempo anterior. Dependiendo del periodo de tiempo t que se desee medir, la rentabilidad puede ser diaria, semanal, mensual, trimestral o anual.

En cuanto a la volatilidad, se refiere a la medida de la variabilidad de los rendimientos en un determinado intervalo de tiempo. Es una medida importante para los inversores, ya que indica la

incertidumbre y el riesgo asociado a una acción en particular. Una alta volatilidad implica que los precios de las acciones fluctúan significativamente, lo que puede presentar oportunidades de ganancias, pero también implica un mayor riesgo. Al contrario, una baja volatilidad significa que el activo es más estable y por lo tanto presenta menor riesgo.

Existen dos medidas para medir la volatilidad, la primera es la varianza de los rendimientos, cuya fórmula matemática es la siguiente:

$$\sigma^2 = \frac{\sum_{i=1}^{N}(r_i - \mu)^2}{N} \tag{3}$$

Ecuación 3. *Fórmula de la varianza*

Donde:

- σ^2 es la varianza
- N es el número total de valores o periodo de tiempo
- r_i es el rendimiento aritmético o logarítmico
- μ es el promedio o media de los rendimientos en el periodo N

Y la segunda es la desviación típica, cuya expresión es:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(r_i - \mu)^2}{N}} \tag{4}$$

Ecuación 4. *Fórmula de la desviación típica*

Siendo $\sigma = \sqrt{\sigma^2}$, tal como se puede deducir de las expresiones anteriores.

Ambas miden cuánto se alejan los rendimientos de su promedio. Cuanto mayor sea esta, mayor será la dispersión de los valores y, por lo tanto, mayor será la volatilidad del activo financiero.

Para la realización de este trabajo se ha usado la desviación típica como medida de volatilidad, ya que suele ser la más usada en el ámbito financiero [10].

2.2 Modelo ARIMA

El modelo ARIMA es uno de los modelos clásicos más utilizados para analizar y predecir series temporales. Fue propuesto por Box & Jenkins en 1970 [9] y es una extensión del modelo ARMA, el cual combina las componentes de auto regresión (AR) y de media móvil (MA), añadiéndole una componente de integración (I) con el objetivo de manejar la estacionalidad y tendencia de la serie.

El concepto de estacionariedad hace referencia a que la media y la varianza de las observaciones de la serie temporal tienen que ser constantes a lo largo del tiempo, es decir, no deben tener tendencia, su media y varianza debe ser constante. Este concepto es importante a la hora de realizar las predicciones, ya que los modelos se comportan mejor ante series estacionarias.

Existen diferentes pruebas para comprobar la existencia de estacionariedad, como la prueba de raíces unitarias (Dickey & Fuller, 1979) [13], más conocida como prueba de Dickey-Fuller (DF). Esta se basa en comprobar la existencia de una raíz unitaria en la serie de tiempo, ya que esta indica que la serie no es estacionaria al poder modelarse como un modelo autorregresivo. Se

establecen dos hipótesis: H0 y H1, la primera supone que la serie no es estacionaria y la segunda que sí que es estacionaria. La prueba calcula un valor de probabilidad p que mide la evidencia en contra de la hipótesis nula, si este valor es menor al 5% hay evidencia suficiente para rechazar la hipótesis nula, en caso contrario no se puede rechazar H0 y por tanto la serie no sería estacionaria. Además, el hecho de utilizar rendimientos logarítmicos hace que la serie sea estacionaria con carácter general.

El modelo ARIMA se basa en la metodología Box-Jenkins (Box & Jenkins, 1973) [8], la cual consiste en considerar varios modelos para el análisis e identificar y escoger el orden de retrasos de la parte autorregresiva (AR) y de la parte de medias móviles (MA) que mejor se ajuste a los datos. Posteriormente se tienen que estimar las fases o parámetros del modelo y finalmente se verifica el modelo ajustado a través de un análisis de los residuos.

Se denota como $ARIMA(p, d, q)$ y matemáticamente se puede expresar como [21]:

$$(1 - \emptyset_1 B - \emptyset_2 B^2 - \cdots - \emptyset_p B^p)(1 - B)^d X_t = (1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_q B^q)\varepsilon_t \qquad (5)$$

Ecuación 5. *Expresión del modelo ARIMA*

Donde:

- p es el orden del modelo AR
- q es el orden del modelo MA
- d es el orden de la componente de integración, que representa el número de diferencias necesarias para hacer que la serie sea estacionaria.
- B es el operador de retardo, que representa el desplazamiento hacia atrás en el tiempo.
- X_t es el valor de la serie temporal en el tiempo t.
- $\emptyset_1$, …, $\emptyset_p$ son los coeficientes autorregresivos que representan la influencia de las p observaciones anteriores en el valor actual del modelo AR.
- θ_1 , …, θ_q son los coeficientes de media móvil que representan la influencia de los q errores pasados en el valor actual del modelo MA.
- ε_t es el error aleatorio en el tiempo t

En la expresión anterior, el primer término representa a la componente autorregresiva (AR), el segundo a la componente de integración (I) y el tercero representa la componente de media móvil (MA).

En cuanto al cálculo del operador B, este se puede expresar como:

$$B^n X_t = X_{t-n} \qquad (6)$$

Ecuación 6. *Expresión del operador B del modelo ARIMA*

Es decir, $B^n X_t$ representa el valor de la serie temporal en $t - n$.

Para estimar los parámetros p y q óptimos del modelo se suele hacer uso de la función de autocorrelación (ACF) para la componente de media móvil (q) y de la función de autocorrelación parcial (PACF) para la componente autorregresiva (p). Aunque, también se puede encontrar la combinación óptima usando el cálculo AIC (*Akaike Information Criterion*).

Por otro lado, los métodos más comunes para la estimación de los parámetros de los modelos ARIMA son: el método de mínimos cuadrados ordinario (MCO), el de máxima verosimilitud (MV) y el de la estimación por cuadrado mínimos generalizados (GLS).

Si la componente de integración d es igual a 0, el proceso se define como $\mathrm{ARIMA(p, 0, q)}$ o $\mathrm{ARMA(p, q)}$, donde no se requiere diferenciación de la serie. El proceso es estacionario si lo es su componente autorregresiva, y es invertible si lo es su componente de medias móviles.

Las series con parte regular y variaciones cíclicas pueden representarse mediante los modelos $\mathrm{ARIMA(p, d, q)(P, D, Q)}$. El primer paréntesis se refiere a la parte regular de la serie y el segundo paréntesis se refiere a las variaciones estacionales o cíclicas.

Además, dentro de la familia ARIMA también se encuentran el modelo ARIMAX, que incorpora variables exógenas o el VARIMA para predecir múltiples series temporales relacionadas entre sí.

El modelo ARIMA no se utilizará en el presente trabajo, ya que existen modelos más específicos para predecir la volatilidad de una serie temporal. Aunque, es importante entender el concepto de este para comprender mejor los modelos que se explican en el siguiente apartado.

2.3 Modelos ARCH

El modelo ARCH (*Autoregressive Conditional Heteroskedasticity*) es un modelo estadístico utilizado en el análisis y la predicción de series temporales financieras desde los años 80 (Robert F. Engle, 1982) [6]. A diferencia de los modelos ARIMA, los modelos ARCH se centran en modelar la volatilidad condicional. Este término se refiere a que la volatilidad no es constante, sino que varía a lo largo del tiempo en función de observaciones previas. Mientras que el modelo ARIMA se centra solo en la autocorrelación y tendencia de la serie, asumiendo una variabilidad constante.

Este modelo permite capturar y modelar la heteroscedasticidad condicional presente en los datos financieros, lo que implica que la varianza de los errores puede aumentar o disminuir a lo largo del tiempo, dependiendo de los errores pasados de la serie temporal.

El término de heterocedasticidad condicional se refiere a la variabilidad no constante de la volatilidad, es decir, la varianza cambia a lo largo del tiempo.

La expresión de la varianza del modelo ARCH se define como:

$$y_t = \sigma_t \varepsilon_t \tag{7}$$

Ecuación 7. *Expresión de la varianza del modelo ARCH*

Donde:

- y_t es el error en el tiempo t.
- σ_t es la volatilidad condicional en el tiempo t.
- ε_t es un error estándar independiente e idénticamente distribuido.

La volatilidad condicional σ_t^2 se modela como una función lineal de los errores pasados al cuadrado:

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i y_{t-i}^2 \quad (8)$$

Ecuación 8. *Expresión de la volatilidad condicional al cuadrado del modelo ARCH.*

Donde:

- σ_t^2 es la varianza condicional en el tiempo t.
- ω es la varianza constante.
- α_i son los coeficientes del modelo ARCH que capturan la autocorrelación en la varianza condicional.
- y_{t-i}^2 son los errores pasados al cuadrado.

Dado que el objetivo es predecir datos financieros con épocas o eventos de inestabilidad, provocando que volatilidad aumente o disminuya sustancialmente, resulta de gran utilidad aplicar este modelo para predecir acciones en la bolsa. Sin embargo, ante la falta de flexibilidad, la asimetría de la volatilidad o las dificultades de pronóstico a largo plazo del modelo ARCH, llevó a la necesidad de desarrollo de modelos más avanzados como el GARCH o el EGARCH.

2.3.1 Modelo GARCH

El modelo GARCH (*Generalized Autoregressive Conditional Heteroskedasticity*) es la ampliación del modelo ARCH. Fue introducido por Tim Bollerslev en 1986 [1] [20] y permite capturar tanto la autocorrelación condicional en la varianza como la autocorrelación condicional en los errores, proporcionando así una descripción más completa de la volatilidad de una serie temporal. Mientras que, el modelo ARCH solo se enfoca en la relación de autocorrelación en la varianza condicional. Matemáticamente se puede expresar como:

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i y_{t-i}^2 + \sum_{j=1}^{q} \beta_i \sigma_{t-j}^2 \quad (9)$$

Ecuación 9. *Expresión del modelo GARCH*

Donde β_i son los coeficientes del modelo GARCH que capturan la autocorrelación en los errores. Mientras que, los demás parámetros son los mismos que en el modelo ARCH e y_t se calcula utilizando la Ecuación 7.

La estimación de los parámetros del modelo GARCH como del modelo ARCH se realiza utilizando el método de máxima verosimilitud condicional (MLE), este se centra en escoger el valor estimado del parámetro que mayor probabilidad tiene de ocurrir o que mejor se ajusta a los resultados observados.

En cuanto a los pasos del MLE, en primer lugar, se define el modelo estadístico. Seguidamente se especifica la función de verosimilitud y se maximiza, es decir, se encuentran los parámetros que más se ajustan al modelo con la ayuda de algoritmos de optimización. Finalmente se interpretan los resultados o estimaciones con los parámetros elegidos.

El modelo GARCH mejora al modelo ARCH, especialmente en las predicciones a largo plazo al incorporar la componente de media móvil. Sin embargo, este modelo no refleja completamente la naturaleza de la volatilidad de algunos activos financieros ya que sigue presentando el problema de asimetría, es decir, la volatilidad en los modelos GARCH se ve afectada simétricamente ante

cambios positivos y negativos de la volatilidad porque depende del cuadrado de los errores. Aunque, a pesar de no ser el más completo, ha sido el modelo escogido en el presente Trabajo Fin de Grado.

2.3.2 Modelo EGARCH

El modelo EGARCH (*Exponential Generalized AutoRegressive Conditional Heteroskedasticity*) o GARCH exponencial fue propuesto por Nelson en 1991 para permitir efectos asimétricos entre los rendimientos positivos y negativos de los activos financieros [1] [20]. Resulta un modelo más complejo y su expresión matemática es la siguiente:

$$ln(\sigma_t^2) = \omega + \sum_{i=1}^{p} \alpha_i \ln(\sigma_{t-i}^2) + \sum_{j=1}^{q} \beta_k g(\varepsilon_{t-1-k}) \quad (10)$$

Ecuación 10. *Expresión del modelo EGARCH*

Donde:

- σ_t^2 es la varianza condicional en el tiempo t.
- ω es la varianza constante.
- α_i y β_i son los coeficientes de autoregresión y media móvil, respectivamente.
- p y q son los órdenes de los términos autorregresivos y de media móvil, respectivamente.
- $g(\varepsilon_{t-1-k})$ es una función que captura el efecto asimétrico de los errores pasados.

El modelo EGARCH mejora los problemas de simetría del modelo GARCH. Más adelante, en el apartado 5 se realiza una comparación entre ambos modelos clásicos.

La familia ARCH es mucho más completa, a parte de los modelos GARCH y EGARCH comentados se encuentran los modelos de media, linealización y varianza multiplicativa como el GARCH-M, LGARCH y el MGARCH, o los que introducen una función no lineal o umbrales en el modelo GARCH como el NGARCH y el TGARCH [7].

2.4 Introducción a las redes neuronales

Las redes neuronales son modelos que simulan el comportamiento del sistema nervioso, es decir, emulan el modo en el que el cerebro humano procesa la información. Las unidades básicas son las neuronas, que generalmente se organizan en capas las cuales se conectan entre si formando una red neuronal. La estructura básica de una red neuronal se muestra en la Figura 1 y se divide en: una capa de entrada, con unidades que representan los campos de entrada; una o varias capas ocultas**,** donde se realizan una serie de operaciones matemáticas; y una capa de salida, con una unidad que representa el campo de destino.

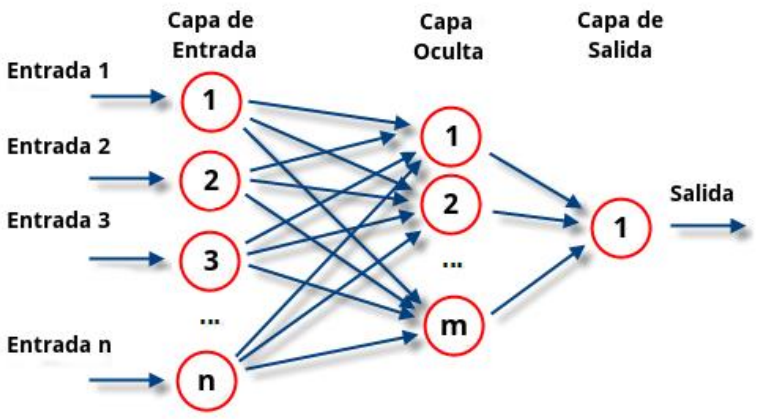


Figura 1. *Estructura básica de una red neuronal [4]*

Cada una de las neuronas de la red posee a su vez un peso, un valor numérico, con el que modifica la entrada recibida. Los nuevos valores obtenidos salen de las neuronas y continúan su camino por la red hasta llegar a la capa de salida, donde se realiza una suma ponderada de los valores y cuyo resultado se pasa por una función de activación, donde se obtiene un valor final que será la salida de la red neuronal. Este funcionamiento se basa en el modelo de perceptrón [17], el cual fue inventado en 1957 por Frank Rosenblatt [23], y sienta las bases de las redes neuronales actuales.

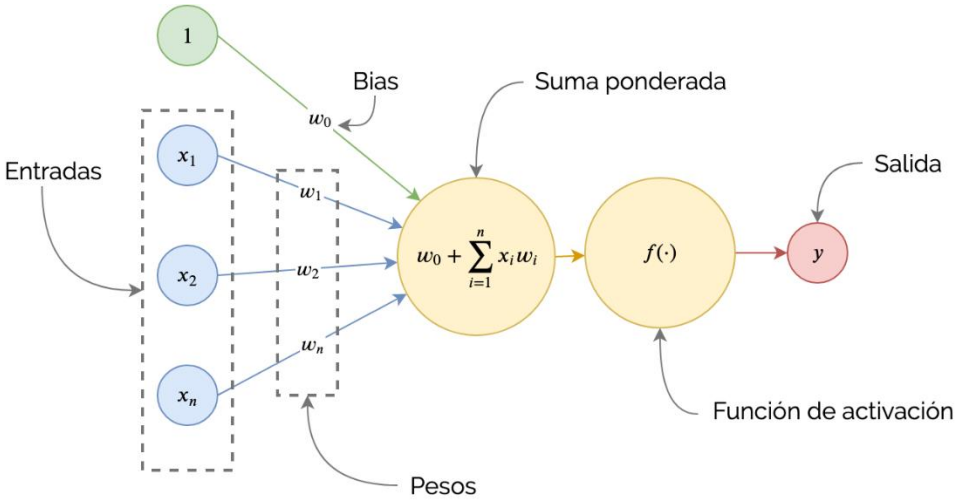


Figura 2. *Funcionamiento de una red neuronal monocapa o perceptrón [11]*

Siguiendo el esquema mostrado en la figura anterior se puede deducir que la expresión general de una red neuronal es:

$$y = f(W_0 + \sum_{i=1}^{n} X_i W_i) \tag{10}$$

Ecuación 11. *Expresión general de una red neuronal*

Las funciones de activación (f) más comunes de una red neuronal son: la sigmoide, la Unidad Lineal Rectificada (ReLU) y la tangente hiperbólica, las cuales están representadas en la Figura 3.

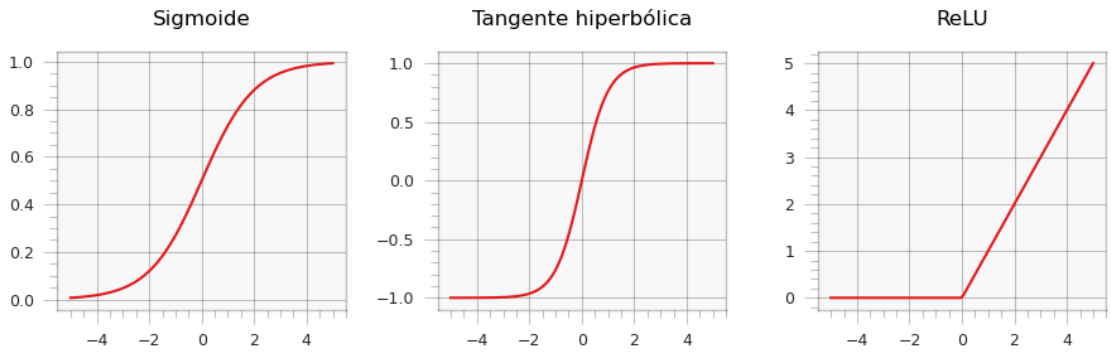


Figura 3. *Funciones de activación [11]*

En la actualidad, el uso de las redes neuronales es muy amplio. Sin embargo, para su correcto funcionamiento es necesario entrenarlas. Este entrenamiento se realiza modificando los pesos de sus neuronas para que consiga extraer los resultados deseados. Para ello lo que se hace es introducir datos de entrenamiento en la red y en función del resultado que se obtenga, se modifican los pesos de las neuronas según el error obtenido y de la contribución de cada una a dicho resultado.

La información que maneja la red se puede dividir en información volátil y no volátil, la primera se refiere a los datos que se están usando y es la que varía con la dinámica de la computación de la red. La segunda hace referencia a los datos que no varían, estos son importantes para el aprendizaje de la red, por lo que, se mantienen para recordar los patrones aprendidos.

Respecto al tipo de aprendizaje, se puede dividir en tres tipos:

- Aprendizaje supervisado: consiste en que la red dispone de los patrones de entrada y los patrones de salida que deseamos para esa entrada y en función de ellos se modifican los pesos de las sinapsis para ajustar la entrada a esa salida.
- Aprendizaje no supervisado: consiste en no presentar patrones objetivos, sino solo patrones de entrada, y dejar a la red clasificar dichos patrones en función de las características comunes de los patrones.
- Aprendizaje reforzado: donde el supervisor no enseña patrones objetivos si no que solo le dice se acierta o falla en su respuesta ante un patrón de entrada. Es un hibrido entre el aprendizaje supervisado y el no supervisado.

Hay que destacar que la regla de aprendizaje de una red neuronal consiste en algoritmos basados en fórmulas matemáticas, que usando técnicas como minimización del error modifican el valor de los pesos sinápticos en función de las entradas disponibles y con ello optimizan la respuesta de la red a las salidas que deseamos.

En cuanto al número de capas existen dos tipos:

- Redes monocapa: es el tipo más sencillo de red neuronal artificial. Solo puede analizar objetos linealmente separables con resultados binarios, es decir, 1 o 0. En la Figura 4 se muestra el esquema de este tipo de red, donde se puede apreciar que solo existe una capa, la capa de salida.

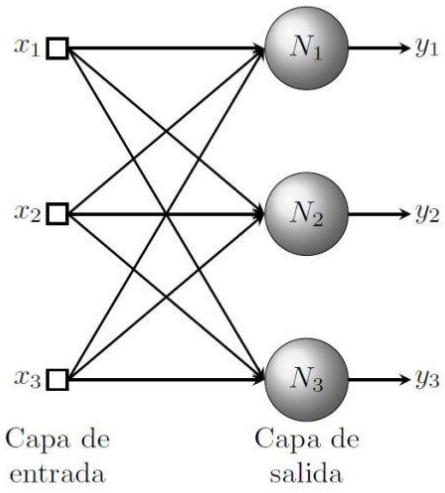


Figura 4. *Red monocapa [19]*

- Redes multicapa: están formadas por varias capas (Figura 5), a diferencia de la red monocapa, esta dispone de un conjunto de capas intermedias (capas ocultas) entre la capa de entrada y la de salida. Contra más capas ocultas presente la red, más profundo y complejo será su aprendizaje.

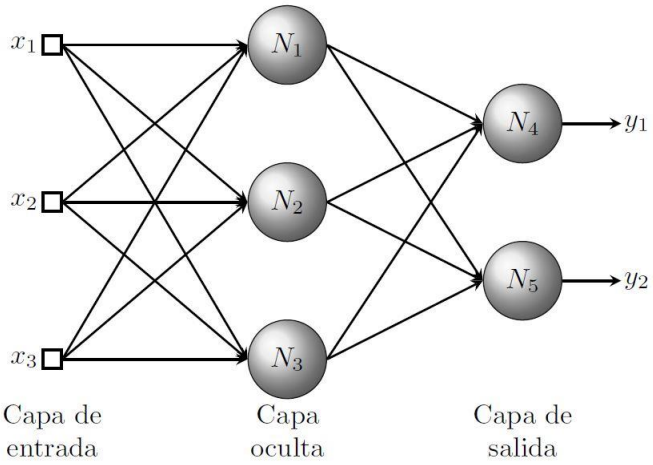


Figura 5*. Red multicapa [19]*

Según el sentido de la información, las redes se clasifican en:

- Redes Neuronales Feedforward (FNN): son las redes neuronales más básicas, donde la información fluye en una sola dirección desde la capa de entrada a través de las capas ocultas hasta la capa de salida, sin ciclos o conexiones retroactivas. Estas redes son ampliamente utilizadas en tareas de clasificación y regresión, donde la entrada se procesa de manera secuencial y se produce una salida sin considerar el contexto temporal.

- Redes Neuronales Recurrente (RNN): estas redes tienen conexiones retroactivas o recurrentes que les permiten mantener y utilizar información anteriormente procesada. Esto significa que la información fluye tanto hacia adelante como hacia atrás a través de la red, formando bucles y permitiendo que la red tenga una especie de "memoria" interna.

Son el tipo de redes más usadas actualmente y es en la que se basa la red LSTM. En el apartado 2.4.1 se explicará más detalladamente el funcionamiento de este tipo de redes.

Es importante tener en cuenta que existen variantes y extensiones de estas dos categorías principales de redes neuronales, como las redes neuronales convolucionales (CNN) que se utilizan específicamente para estructuras espaciales, como imágenes y videos. Las CNN se basan en la idea clave de la convolución y reducción, la primera es una operación matemática que implica deslizar un filtro o ventana de tamaño NxN sobre una imagen para extraer características locales. Mientras que la segunda se utiliza para reducir el tiempo y memoria de computación. Su estructura se muestra en la Figura 6 y son especialmente útiles para detectar objetos o generar y clasificar imágenes.

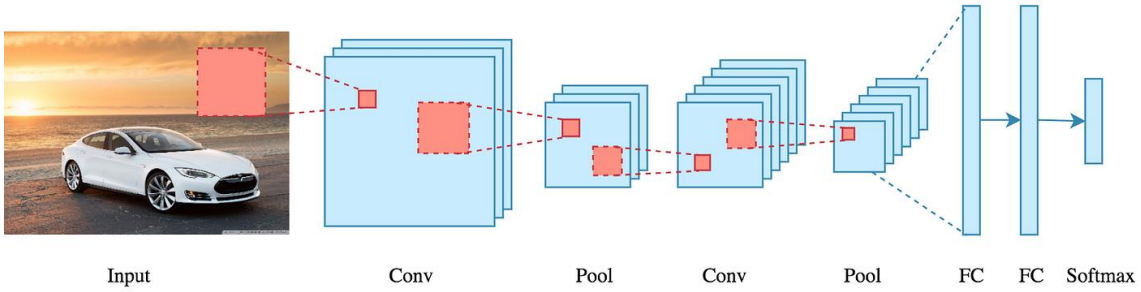


Figura 6. *Estructura de una Red Neuronal Convolucional [28]*

Estas redes son muy interesantes, sin embargo, no se hará más hincapié en ellas al no estar del todo relacionadas con el presente trabajo.

2.4.1 Redes Neuronales Recurrentes (RNN)

Como se ha introducido anteriormente, las redes neuronales recurrentes presentan bucles, lo que hace que la información persista en el tiempo. El bucle permite que la información pase de un paso de la red al siguiente, tal como se puede apreciar en la Figura 7, donde se puede ver que la salida h_1 depende de X_1 y del estado anterior A_0. Este funcionamiento se asemeja al de una máquina de estados finitos [31].

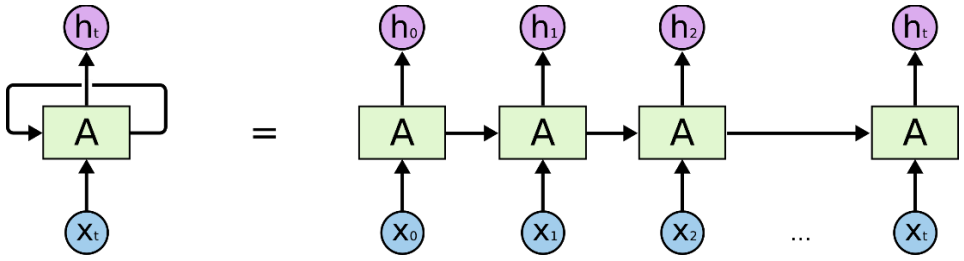


Figura 7. *Funcionamiento de una Red Neuronal Recurrente [22]*

La función de activación de un RNN suele ser una tangente hiperbólica, con el fin de estabilizar los valores a 1 o -1 independientemente del valor de entrada, impidiendo que los valores crezcan indefinidamente y que la red se vuelva inestable. Ampliando el bloque A se tiene:

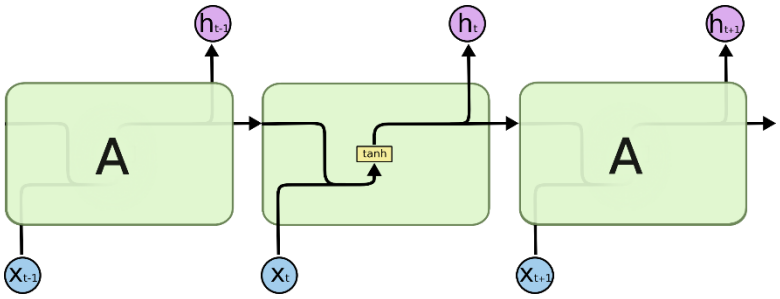


Figura 8. *Bloque de una Red Neuronal Recurrente [22]*

Y la expresión matemática de una RNN se deduce como:

$$y(t) = \tanh\left(Wx(t) + Uh(t-1) + b\right) \tag{12}$$

Ecuación 12. *Expresión matemática de un RNN*

Donde se ha separado la matriz de pesos de las entradas actuales Wx y las entradas pasadas Uh.

El principal problema de las RNN es que un valor bastante alejado en el tiempo se habrá multiplicado por un peso varias veces, por lo que ese valor tenderá a cero. Es decir, las RNN tradicionales no son capaces de almacenar datos muy espaciados en el tiempo. Tal como se aprecia en la Figura 9, la salida h_{t+1} no puede estar relacionada con las entradas X_0 y X_1, ya que habrán desaparecido debido a la gran brecha temporal entre la salida y las entradas.

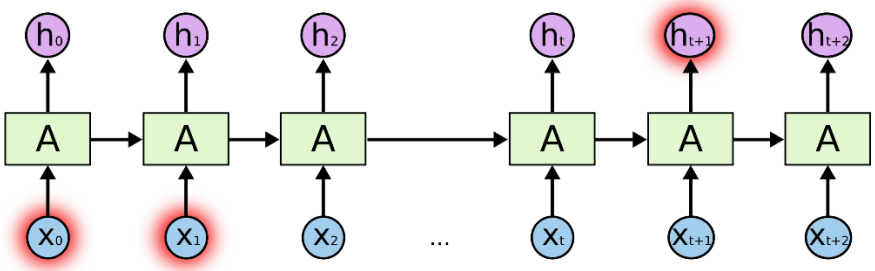


Figura 9. *Bloque de una Red Neuronal Recurrente [22]*

Para evitar este problema de dependencia a largo plazo surgieron las redes LSTM, que se desarrollarán a continuación.

2.4.2 Redes LSTM

Las redes de memoria a largo plazo o LSTM (*Long Short-Term Memory*) es un tipo especial de red neuronal recurrente. Fueron introducidas por Sepp Hochreiter y Jürgen Schmidhuber en 1997 [26] y están diseñadas explícitamente para recordar información durante largos períodos de tiempo.

A diferencia de las RNN tradicionales, las cuales tienen solo una capa, las LSTM presentan cuatro ramas [22] [31] que interactúan entre si de una manera muy especial:

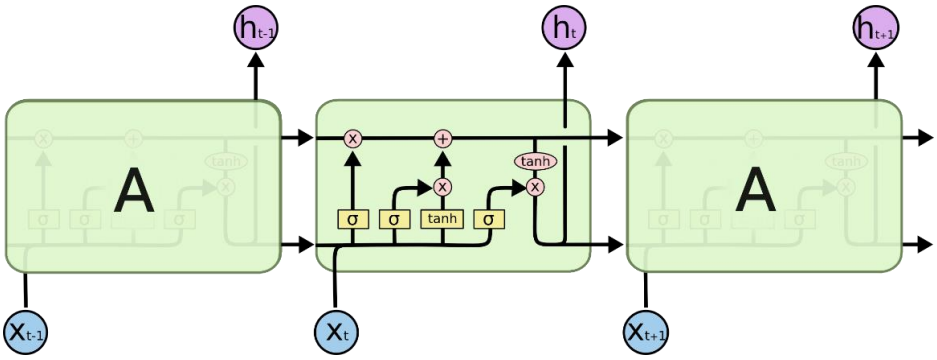


Figura 10. *Bloque de una Red LSTM [21]*

Rama de olvidar o recordar

La primera rama hace referencia a que información ya almacenada se quiere recordar o se quiere olvidar teniendo en cuenta la nueva entrada X_t. Es decir, permite eliminar elementos de la memoria. Como se aprecia en la Figura 10, esta rama viene dada por una función sigmoide, esto quiere decir que, si el valor de la sigmoide es 1, se recordará esa información y si es 0, se olvidará completamente. Su mecanismo se asemeja al de un interruptor, multiplicando el estado anterior por el valor de esta rama.

Rama básica y rama de ignoración

La rama básica es la rama de una RNN tradicional, su función es una tangente hiperbólica cuya función es estabilizar o normalizar los valores a 1 o -1. Esta rama crea un vector de nuevos valores candidatos, que podría agregarse al estado.

La rama de ignoración decide que valores candidatos serán incorporados al estado y cuáles no. Su función es una sigmoide, al igual que en la primera rama, esta actúa de interruptor multiplicando por 1 o por 0 los candidatos de la rama básica.

Finalmente se combinan estas dos para crear una actualización del estado, añadiendo los candidatos seleccionados mediante el operador suma.

Rama de selección

En esta última rama se genera la salida h_t de la red. Para ello se ejecuta una capa sigmoidea que decide qué partes del estado de la celda actual se van a generar. Luego se pasa el estado de la celda o memoria de la red (en la Figura 10 es la línea horizontal superior) por una tangente hiperbólica y se multiplica por la salida de la puerta sigmoidea, de modo que solo se emitan las partes deseadas, dando lugar a la salida h_t que a su vez es la entrada del siguiente módulo.

Las expresiones matemáticas de las diferentes ramas y de los estados de la red LSTM se resumen en la Tabla 1:

Rama básica	$z(t) = tgh(W_z x(t) + U_z h(t-1) + b_z$
Rama de ignoración	$i(t) = sig(W_i x(t) + U_i h(t-1) + b_i$
Rama de olvidar o recordar	$f(t) = sig(W_f x(t) + U_f h(t-1) + b_f$
Rama de selección	$o(t) = sig(W_o x(t) + U_o h(t-1) + b_o$
Rama de estado	$c(t) = c(t-1) * f(t) + i(t) * z(t)$
Rama de salida	$h(t) = o(t) * tgh(c(t))$

Tabla 1. *Ecuaciones red LSTM*

2.5 Métricas del error

Existen varias medidas para calcular el error de predicción en series temporales, sin embargo, en este proyecto solo se emplearán el RMSE y el MAPE [3][12]. La elección de usar más de una medida es conveniente para contrastar y determinar con mayor exactitud la bondad de los modelos.

2.5.1 RMSE

El RMSE (*Root Mean Squared Error*) es la raíz del error cuadrático medio y mide la diferencia promedio entre los valores predichos de un modelo estadístico y los valores reales. Su fórmula matemática es la siguiente:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2} \tag{13}$$

Ecuación 13. *Fórmula del RMSE*

Donde n es el número de muestras y e_i^2 es el error de predicción al cuadrado, el cual se expresa como:

$$e_i^2 = (P_i - O_i)^2 \tag{14}$$

Ecuación 14. *Error de predicción al cuadrado*

Siendo P_i el valor previsto y O_i el valor observado.

Como su nombre indica, el término que se encuentra dentro de la raíz es el MSE (*Mean Squared Error*) y mide la diferencia promedio al cuadrado entre los valores estimados y el valor real.

Matemáticamente, el RMSE es la desviación estándar de los residuos. Esta métrica es utilizada en muchos campos. Además, su interpretación es intuitiva, ya que es una métrica simple que no requiere un gran conocimiento en estadística. Sin embargo, el RMSE es sensible a valores atípicos, al sobreajuste o a la escala.

2.5.2 MAPE

El MAPE (Mean Absolute Percentage Error) o error porcentual absoluto medio es una medida de error relativo que usa valores absolutos para evitar que los errores positivos y negativos se cancelen entre sí. Su expresión matemática es la siguiente:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n} \left|\frac{O_i - P_i}{O_i}\right| \tag{15}$$

Ecuación 15. *Expresión del MAPE*

Donde O_t es el valor medido u observable, P_t es el pronóstico para la muestra i y n el número de muestras.

A diferencia del RMSE, el MAPE mide el error en términos de porcentaje en lugar de unidades absolutas. Cuanto menor sea su valor, mejor será la precisión del modelo.

El MAPE es útil para comparar el rendimiento de modelos en diferentes escalas o para evaluar la precisión en función de la magnitud de los valores observados. Sin embargo, el MAPE es sensible a divisiones por cero y puede ser afectado por valores extremos o atípicos en los datos.

3. Análisis empírico y modelización en Python

3.1 Conjunto de datos y entorno de programación

Para la elaboración de este trabajo, se ha escogido como conjunto de datos al histórico de las acciones bursátiles de las seis grandes empresas tecnológicas que encabezan el mercado NASDAQ, así como otros índices financieros como el S&P 500 o el mismo NASDAQ-100. Estas empresas mundialmente conocidas son las siguientes:

- Amazon (AMZN): Fundada en 1994 por Jeff Bezos, es una multinacional tecnológica estadounidense especializada en comercio electrónico, computación en la nube, streaming digital e inteligencia artificial. En los últimos años se ha convertido en el mayor minorista en línea del mundo, consiguiendo unos ingresos de 513.500 millones de dólares en 2022.
- Apple (AAPL): Marca y compañía estadounidense que produce y diseña artículos electrónicos y software. En las últimas décadas se ha convertido en una de las marcas de tecnología más importantes del mundo, consiguiendo unos ingresos de 394.330 millones de dólares en 2022.
- Alphabet (GOOGL): Grupo de empresas formado por principalmente por Google, desarrolla productos y servicios relacionados con internet, software o electrónica, entre otros. Es una de las empresas más conocidas, generando más de 282 mil millones de dólares en 2022.
- Microsoft (MSFT): Compañía estadounidense vinculada al desarrollo y fabricación de productos de software y servicios para diferentes tipos de dispositivos electrónicos. En 2022 obtuvo unos ingresos de 198.270 millones de dólares.
- Meta (META): Nombre que recibe el grupo de empresas y servicios formado por Facebook, liderada por el empresario Mark Zuckerberg. Recibe este nombre por la reciente reorientación de la compañía hacia el metaverso. Sus ingresos anuales en 2022 fueron de 116.609 millones de dólares.
- Tesla (TSLA): Empresa norteamericana creada en 2003 con sede en California. Fabrica y comercializa vehículos eléctricos, así como componentes y baterías para otros fabricantes. Su CEO es el conocido Elon Musk y durante 2022, la compañía ingresó más de 81.400 millones de dólares.

El entorno de programación escogido para el desarrollo de los modelos ha sido Google Colab, ya que es una plataforma orientada a ciencia de datos y a inteligencia artificial (IA), por lo que resulta el entorno ideal para desarrollar este Trabajo Fin de Grado. En un primer momento, se empezó a trabajar en Visual Studio, sin embargo, gracias a la búsqueda de información se descubrió esta plataforma y se decidió migrar a ella, ya que cuenta con bibliotecas de análisis de datos preinstaladas y puede trabajar con diversas fuentes de datos, además de la flexibilidad que te ofrece el poder acceder desde cualquier ordenador gracias al guardado automático en la nube. Aunque, la principal ventaja, es la ejecución de código en un servidor de Google, aumentando considerablemente la capacidad computacional.

En cuanto a la muestra recogida consta de 4152 días, correspondientes a los datos diarios de cierre desde el 18 de mayo de 2012 hasta el tercer trimestre de 2023, es decir, hasta el 30 de septiembre de 2023. La elección de la fecha de partida es debido a que la última empresa en salir a bolsa fue META (Facebook) ese mismo día y, con el fin de obtener registros en todas las empresas se ha decidido partir desde esa fecha, dando lugar a un horizonte temporal de más de 11 años.

3.2 Análisis y procesamiento de los datos

Los datos se han recogido de la plataforma web Yahoo Finance, la cual ofrece servicios como cotizaciones en tiempo real, información de empresas, gráficos, indicadores y seguimiento de acciones entre otros.

La opción más común para importar datos de una serie temporal en Python es a través de un fichero Excel o 'csv'. Sin embargo, existe una librería denominada 'yfinance' que permite importar estos datos directamente de Yahoo Finance a Python. Hay que destacar que esta librería ya está preinstalada en Google Colab, por lo que no será necesaria su instalación.

Al ejecutar el comando 'yf.download', con la etiqueta de la empresa deseada se importa la siguiente información:

- **Open:** Precio de apertura en un día o fecha concreta
- **High:** Precio máximo alcanzado en dicha fecha
- **Low:** Precio mínimo alcanzado en dicha fecha
- **Close:** Precio de cierre de la cotización en la fecha concreta
- **Adj Close:** Precio ajustado de cierre en la fecha concreta
- **Volume:** Volumen de negocio en un día o fecha concreta

Para el análisis de este proyecto, se ha centrado únicamente en el precio de cierre (Close) de cada compañía. En la Figura 11, se muestra un gráfico del precio de cierre de cada empresa durante el periodo de tiempo de estudio.

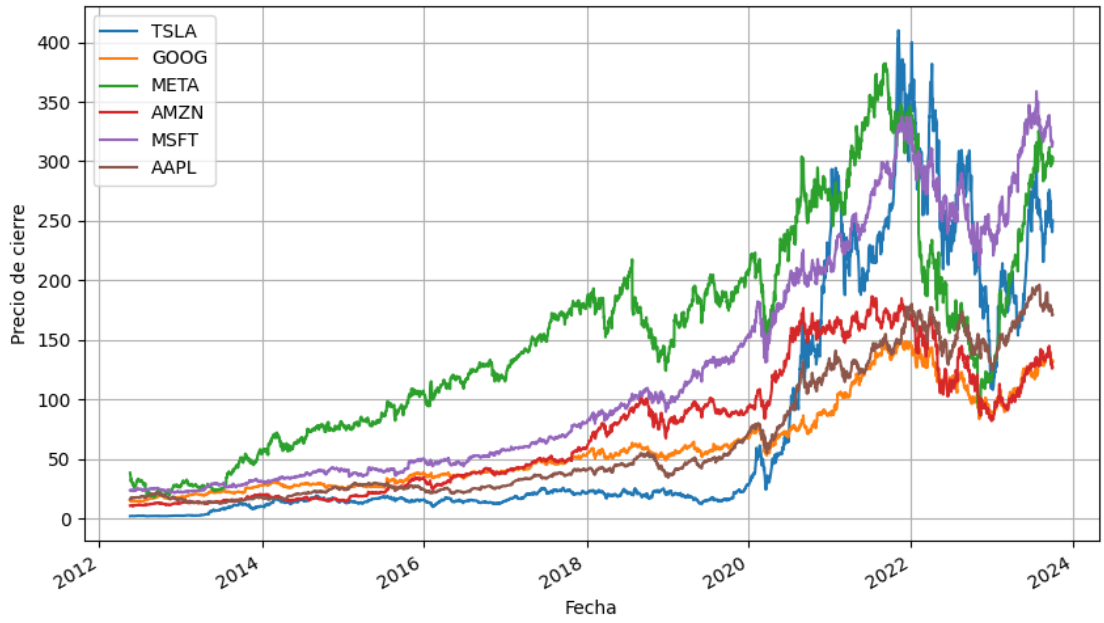


Figura 11. *Precio de cierre de cada empresa*

Fuente: *Elaboración propia a partir de los datos obtenidos de Yahoo Finance*

En primer lugar, lo que más llama la atención a la hora de analizar el gráfico es el gran crecimiento que han tenido las tecnológicas desde el año 2012, cuyo precio por acción no superaba los 50 dólares. También, se puede observar que hasta el año 2018 hay una tendencia ligeramente creciente en la mayoría de las empresas a excepción de META, que es la que presenta un crecimiento mayor, alcanzando casi los 200 dólares por acción al inicio de 2018, mientras que las demás apenas superaban los 100 dólares.

En julio de 2018 hay una gran caída en META, cuyas acciones descendieron un 19% provocando una pérdida de 120.000 millones de dólares en un solo día [15]. Esta caída es debida a la filtración masiva de datos personales de clientes de esta empresa, que provocó el descontento y la venta de las acciones de los inversores ante las posibles consecuencias.

Hasta inicios de 2020 se mantuvo la tendencia creciente, sin embargo, en marzo de 2020 llegó uno de los eventos más importantes de los últimos años: la pandemia de la COVID-19. Este suceso provocó de primeras una gran caída en bolsa de todas las empresas de estudio ante la incertidumbre del momento. Aunque, debido al confinamiento, al teletrabajo y al aumento del consumo digital, las tecnológicas, tras el transcurso de los meses tuvieron más importancia en la sociedad. Es por esto por lo que hasta finales de 2022 aumentaron su valor considerablemente, alcanzando máximos históricos como es el caso de TESLA, logrando superar los 400 dólares en noviembre de 2021. Posteriormente, se observa una tendencia negativa en todas las empresas hasta finales de 2022 y desde el inicio de 2023 hasta el último registro de estudio se vuelve a ver una tendencia positiva.

Para profundizar más en el análisis de la serie, se han calculado los parámetros estadísticos más significativos como la media, la desviación típica, el máximo y el mínimo de cada compañía, los cuales se reflejan en la siguiente Tabla:

	Tesla	**Google**	**Meta**	**Amazon**	**Microsoft**	**Apple**
Media	78.24	60.58	153.99	73.12	123.16	65.06
Desv. típica	103.79	37.41	88.36	53.27	99.06	54.43
Mínimo	1.74	13.92	17.73	10.41	21.55	11.99
Máximo	409.97	150.71	382.18	186.57	358.00	195.92

Tabla 2. *Resumen parámetros estadísticos*

Fuente: *Elaboración propia a partir de los resultados obtenidos en Python*

Se observa que Tesla es la que mayor desviación típica presenta en comparación con las demás empresas. Además, es la compañía con el máximo y mínimo valor de cotización absoluto, esto nos indica, tal como se puede apreciar en la Figura 11, que es la empresa que más ha crecido respecto al año 2012. Mientras que, la empresa que más constante se ha mantenido a lo largo del tiempo ha sido Google, con una desviación típica de 36,20 y una media de 59,47. También se puede observar en los mínimos y máximos el crecimiento de META y MSFT, cuya media ha sido de 151,93 y 119,7, y su desviación típica de 86,02 y 95,26 respectivamente.

A continuación, se procede a calcular la rentabilidad logarítmica diaria en porcentaje (Ecuación 2), que es la que se utilizará como entrada en el modelo GARCH (se ha decidido tomar el porcentaje de las rentabilidades como entrada debido al problema de magnitud que presentaba el modelo GARCH a la hora de estimar los parámetros, siendo el intervalo ideal entre 1 y 1000). En Python se ha usado la función ‘log’ de la librería Numpy y luego se ha aplicado la función ‘diff’ para diferenciar el valor actual con el anterior, siguiendo los pasos descritos en la Ecuación 2. Por último, se multiplica por 100 para representarla en términos de porcentaje. Esta rentabilidad se muestra en la Figura 12.

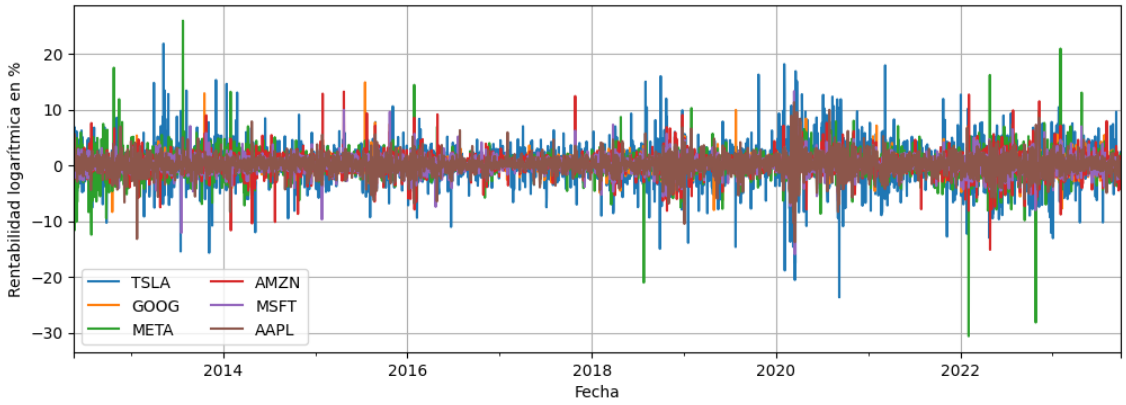


Figura 12. *Rentabilidad logarítmica diaria de cada empresa*

Se observa que Tesla es la que mayor variación presenta a lo largo del tiempo, con picos de rendimiento muy elevados respecto a las demás, sin embargo, es Meta la compañía que presenta el rendimiento máximo y mínimo absoluto. La caída que se aprecia a inicios de 2022 tiene que ver con el anuncio de cambio de nombre de Facebook a META y el nuevo rumbo que tomó esta compañía hacia el metaverso, que vino acompañada de una gran caída de ingresos provocando una crisis dentro de la empresa.

Con la información analizada hasta ahora, se puede ver que Tesla y Meta son las empresas con mayor volatilidad y riesgo para los inversores. Sin embargo, la rentabilidad diaria introduce bastante ruido y resulta difícil de analizar. Es por esto por lo que se ha decidido calcular y estudiar también la rentabilidad semanal y mensual, con tal de hacer más robusto el análisis y ver cómo se comportan ambos modelos ante diferentes medidas o periodos de tiempo. Dichas rentabilidades se muestran en las Figuras 13 y 14 respectivamente.

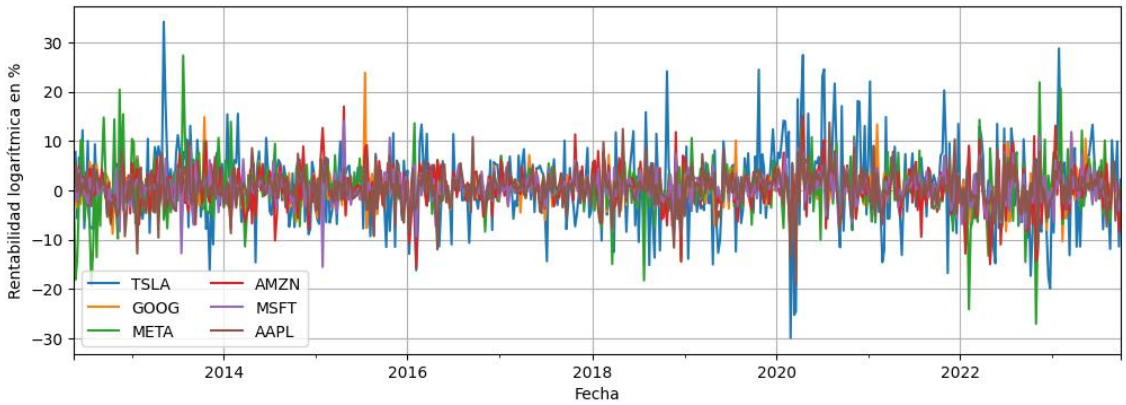


Figura 13. *Rentabilidad logarítmica semanal de cada empresa*

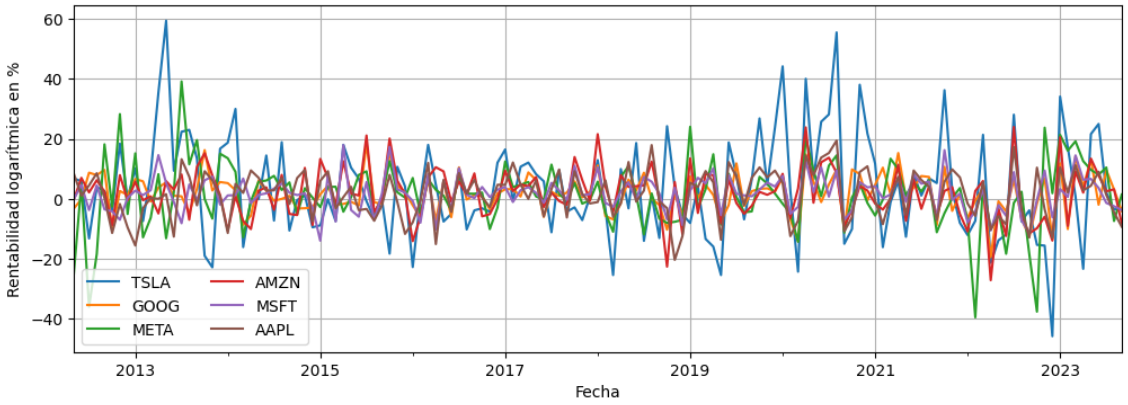


Figura 14. *Rentabilidad logarítmica mensual de cada empresa*

El siguiente paso es ver si se cumple el principio de estacionariedad introducido en el apartado 2.2. Partiendo de las gráficas de rentabilidad (Figuras 12, 13 y 14) se puede ver que en las muestras diarias y semanales la media es contante e igual a cero y a priori no existe ningún tipo de tendencia, sin embargo, en la muestra mensual no se ve del todo claro. Para ello, se ha aplicado la prueba Dickey-Fuller (DF) a cada serie de datos de rentabilidad, cuyos valores p se muestran en la Tabla 3. Donde se puede observar que ningún valor p es mayor a 0,05, por lo que las tres series cumplen el criterio de estacionariedad para todas las empresas del estudio.

Muestras	Tesla	Google	Meta	Amazon	Microsoft	Apple
Diarias	0	2.35e-30	0	0	1.14e-27	6.51e-26
Semanales	1.84e-21	0	2.11e-20	0	0	0
Mensuales	3.88e-4	1.73e-25	2.87e-20	5.54e-24	2.07e-25	1.95e-14

Tabla 3. *Prueba Dickey-Fuller para cada serie de tiempo*

Fuente: *Elaboración propia a partir de los resultados obtenidos en Python*

Tras comprobar que las series de entrada son válidas para el primer modelo, se procede a calcular la volatilidad diaria, semanal y mensual, que serán las series de entrada de la red neuronal LSTM. Para ello se ha calculado la volatilidad de cada registro con el siguiente a través de la desviación típica, tal como se ha visto en el apartado 2.1, para así tener una serie de datos comparable a lo largo del tiempo (en Python se ha aplicado la función 'rolling' a cada rentabilidad con una ventana de dos valores seguida de la función 'std'). Estas series se usarán para comparar y medir los errores de predicción de ambos modelos. La volatilidad diaria se muestra en la Figura 15, la volatilidad semanal en la Figura 16 y la volatilidad mensual en la Figura 17.

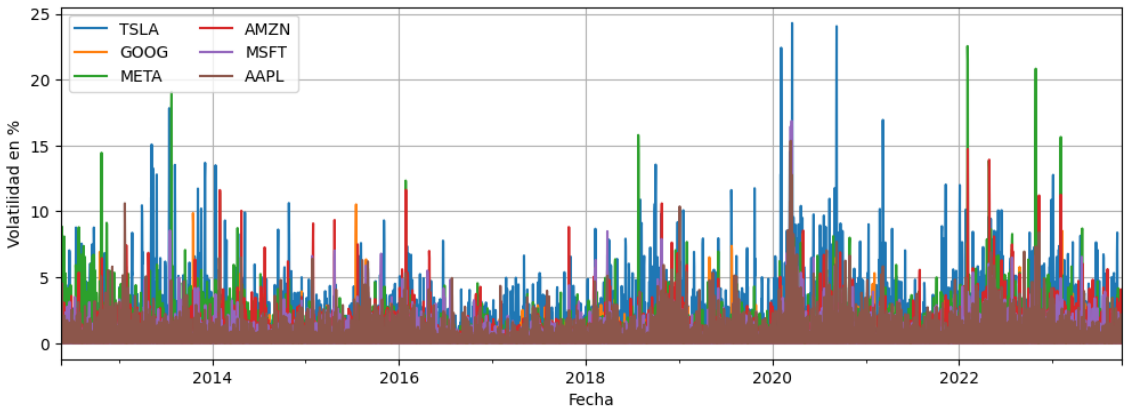


Figura 15. *Volatilidad diaria de cada empresa*

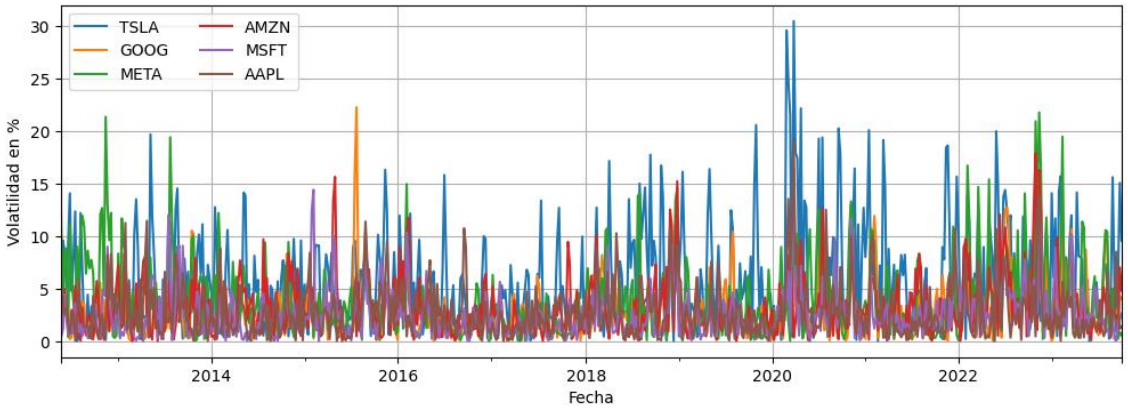


Figura 16. *Volatilidad semanal de cada empresa*

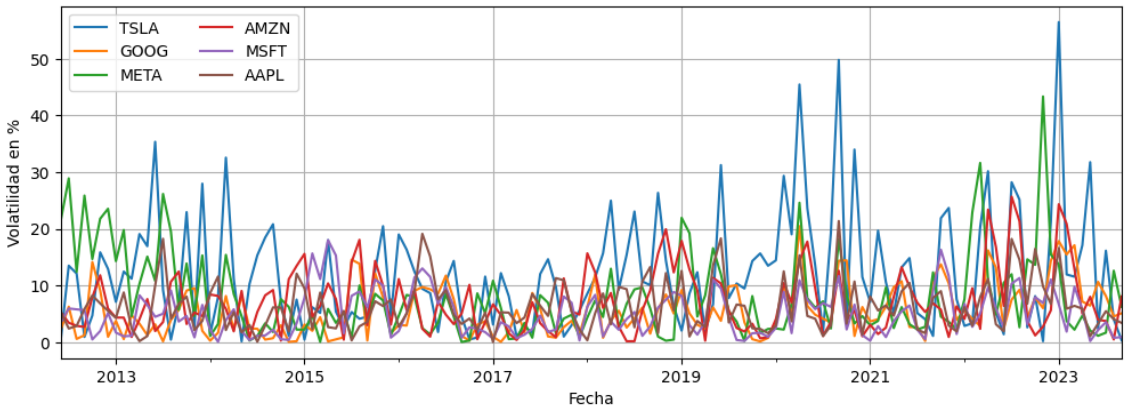


Figura 17. *Volatilidad mensual de cada empresa*

3.3 Modelización en Python

En este apartado se va a explicar cómo se ha implementado o programado cada modelo en Python. Las librerías que se han utilizado en ambos modelos, así como en el análisis de datos previo han sido las siguientes: Numpy, Pandas y Math para el procesamiento de los datos, Matplotlib para graficar los resultados y Sklearn para importar las medidas del error.

Respecto a los datos de entrenamiento y prueba, variarán dependiendo del tipo de volatilidad a predecir, ya que el hecho de incluir datos semanales y mensuales hace que se tenga un menor número de muestras en cada modelo de predicción. Así pues, para la volatilidad diaria se tomarán las últimas 150 muestras como datos de prueba, para la volatilidad semanal las últimas 100 muestras y para la volatilidad mensual las últimas 50 muestras. Además, el número total de muestras será inferior al número total de días, ya que como se ha introducido anteriormente, los mercados solo operan en días hábiles de lunes a viernes.

En la Tabla 4 se resume el tamaño de cada conjunto de prueba y entrenamiento para cada tipo de datos de rentabilidad.

Datos	Entrenamiento	Prueba	Total
Diarios	2705	150	2855
Semanales	494	100	594
Mensuales	87	50	137

Tabla 4. *Muestras para cada tipo de datos*

Fuente: *Elaboración propia*

Por otro lado, debido al tiempo computacional de la red LSTM se ha decidido predecir con una ventana deslizante de 5 muestras, es decir, por cada iteración del bucle '*for'* de cada modelo se pronosticarán 5 valores, en vez de solo un valor. Es por esto también por lo que se ha escogido un tamaño de prueba múltiplo de cinco. Además, en la siguiente iteración el conjunto de datos de entrenamiento se desplazará también 5 posiciones, con la finalidad de que su tamaño se mantenga fijo. En la Figura 18 se ilustra un ejemplo de dicho mecanismo para las dos primeras iteraciones del bucle, donde N es el tamaño del conjunto de entrenamiento y '*pred'* el vector de predicciones.

i = 1	1	2	3	4	5	6	7	8	9	10	…	N-5	N-4	N-3	N-2	N-1
i = 2	6	7	8	9	10	11	12	13	14	15	…	N	N+1	N+2	N+3	N+4
pred				1	2	3	4	5	6	7	8	9	10			

Figura 18. *Ejemplo del mecanismo de ventana deslizante*

Fuente: *Elaboración propia*

En cuanto a la recopilación y procesamiento de los datos, en ambos modelos ha sido la misma. Tal como se ha explicado en el apartado anterior, se emplea la función '*download'* de la librería yfinance para descargar y guardar los datos en un dataframe llamado '*data'*. Posteriormente, se calculan la rentabilidad y la volatilidad diaria, semanal y mensual.

3.3.1 Modelo GARCH en Python

Para implementar el modelo GARCH en Python se ha hecho uso de la función '*arch_model*' de la librería Arch. Esta librería no está preinstalada en Google Colab, por lo que el primer paso será instalarla en la plataforma a través del comando '!pip install arch', y, una vez instalada, se importa junto con el resto de las librerías necesarias. A continuación, se tiene que indicar el tipo de dato o volatilidad a predecir en la variable '*td'* como: 'D' si son datos diarios, 'W' si son semanales o 'M' si son mensuales. Dependiendo del tipo de dato introducido en '*td*' se tomará el tamaño de prueba correspondiente (véase Tabla 4) en la variable '*test_size'* Se indicará también el tamaño de la ventana de predicción en 'N', cuyo valor será igual a 5, tal como se ha mencionado anteriormente.

Como se ha comprobado en el apartado 3.2 (véase Tabla 3), las series de rentabilidad para cada tipo de datos cumplen el criterio de estacionariedad al presentar un p-valor menor a 5%. Haciendo que la serie de entrada sea estacionaria para que el modelo sea lo más eficiente posible, ya que, en caso contrario, el modelo puede generar estimaciones inadecuadas, ya que supone que la volatilidad se relaciona con las fluctuaciones en torno a una media constante.

El siguiente paso es estimar los parámetros p y q que mejor se ajustan al modelo. Para ello se ha calculado el parámetro AIC para cada combinación, siendo el número máximo de p y q igual a 4, con tal de no aumentar el tiempo de computación y de no sobrecargar al modelo. En la Tabla 5 se muestran los resultados de los índices (p,q) que menor AIC han proporcionado para cada empresa y para cada serie de tiempo.

Muestras	Tesla	Google	Meta	Amazon	Microsoft	Apple
Diarias	(2,4)	(1,2)	(3,2)	(4,2)	(1,1)	(1,1)
Semanales	(1,1)	(1,2)	(2,4)	(1,1)	(1,1)	(1,2)
Mensuales	(1,2)	(1,1)	(4,1)	(3,1)	(1,1)	(1,1)

Tabla 5. *Parámetro p y q óptimos del modelo GARCH*

Fuente: *Elaboración propia a partir de los resultados obtenidos en Python*

A continuación, se define dentro de un bucle '*for'* el modelo GARCH a través de la función '*arch_model*', introduciendo los datos de entrenamiento y los parámetros p y q, indicando también la media constante y la distribución normal. Posteriormente, se entrena el modelo utilizando la función '*fit'* y se predecirán los siguientes valores con h = 5 (siendo h el horizonte de predicción) mediante la función '*forecast*'. Esta función estima la variancia esperada en cada valor de predicción, por lo que, se calcula la raíz cuadrada a esta para obtener la volatilidad, luego se almacena en un vector y se vuelve a ejecutar el bucle '*for'* hasta llegar a la última muestra.

Cuando finaliza el bucle, se transforma el vector de los valores predichos a una serie de tiempo con el mismo índice temporal que la serie de volatilidad calculada a través de la función '*Series'* de la librería Pandas.

Por último, se calcula el RMSE y el MAPE, haciendo uso de la librería Sklearn. Mediante la función '*mean_squared_error*' se calcula el MSE y aplicando la raíz cuadrada se obtiene el RMSE. En cuanto al MAPE, se ha calculado usando la función '*mean_absolute_percentage_error*' de la misma librería.

Finalmente, se gráfica el vector de predicciones y el vector de la volatilidad real empleando la función '*plot'* de la librería matplotlib.

En la Figura 19 se muestra un diagrama de flujo del modelo GARCH diseñado, con todos los pasos descritos.

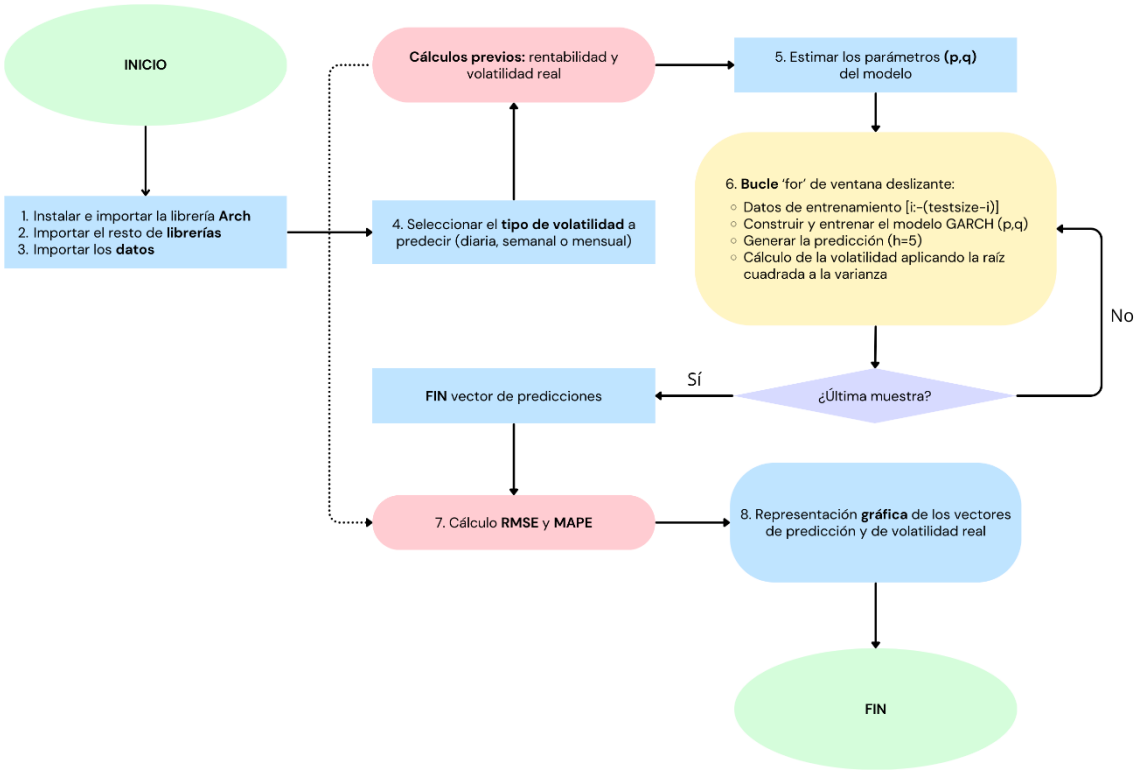


Figura 19. *Diagrama de flujo del modelo GARCH*

Fuente: *Elaboración propia en Canva*

3.3.2 Red Neuronal LSTM en Python

Para implementar la red neuronal LSTM se ha hecho uso de la librería Keras, la cual ya se encuentra instalada en el entorno de programación e incluye funciones de aprendizaje automático y Deep Learning muy intuitivas sin necesidad de ser un experto en la materia, ideal para diseñar con facilidad redes neuronales. Sin embargo, a diferencia del modelo GARCH, se necesita manipular los datos con el fin de adaptarlos a la red neuronal, ya que esta trabaja con matrices de tres dimensiones.

El modelo GARCH es un modelo entrenado para calcular la varianza de una serie dada, en cambio, las redes neuronales LSTM trabajan basándose en los datos de entrada, y en función de varios parámetros se ajustan los pesos de cada rama (véase apartado 2.4). Una opción es introducir como entrada la serie de los rendimientos logarítmicos diarios, semanales o mensuales y posteriormente calcular la volatilidad. Sin embargo, se ha optado por introducir las series de la volatilidad directamente como entrada. Esto hace que los datos muestrales disminuyan en un valor, aunque se ha mantenido fijo el tamaño de la ventana de prueba para poder comparar ambos modelos.

Al igual que en el modelo GARCH, la serie de entrada es conveniente que sea estacionaria para que la red se ajuste mejor y propicie resultados más acertados. Aplicando la prueba de Dickey-

Fuller a las series de volatilidad se obtienen los resultados mostrados en la Tabla 6, donde se puede observar que todos los valores p son menores a 0,05 en todas las empresas de estudio.

Muestras	Tesla	Google	Meta	Amazon	Microsoft	Apple
Diarias	1.34e-12	6.56e-12	9.33e-12	2.32e-13	3.95e-13	1.68e-11
Semanales	1.7e-3	7.52e-19	1.3e-3	3.49e-19	1.99e-19	2e-29
Mensuales	4.33e-18	2.24e-12	3.92e-11	3.72e-12	1.62e-10	3.62e-18

Tabla 6. *Prueba Dickey-Fuller para las series de volatilidad*

Fuente: *Elaboración propia a partir de los resultados obtenidos en Python*

El siguiente paso es dividir los datos de entrenamiento y prueba en una matriz. Para ello se ha creado la función llamada '*convert2matrix*'. Finalmente, se almacenan los datos de entrenamiento en un vector llamado 'X' y los datos de prueba en un vector llamado 'T'.

A continuación, se genera una matriz que contenga los valores de entrenamiento de la red LSTM. Esta matriz presenta un formato en tres dimensiones, cuya nomenclatura es [i, j, k], donde i es el número total de muestras, j es el número de observaciones pasadas que se tienen en cuenta para predecir el valor, este parámetro será igual a 1, ya que solo se va a tener en cuenta el valor anterior. Por último, el parámetro k hace referencia al número de variables que se tienen en cuenta. Al ser un modelo univariante, es decir, el modelo se basa en las propias observaciones pasadas, este valor k será igual a 1.

Una vez los datos están estructurados, se tiene que definir el modelo. Para ello se utilizan las funciones Sequential, LSTM y Dense de la librería Keras. El primer paso es crear un modelo secuencial de capas mediante la función Sequential, seguidamente se agrega una capa LSTM especificando el número de neuronas óptimas y finalmente se agrega una capa densa utilizando la función Dense, esta capa se utiliza para realizar la regresión del modelo y será la capa de salida de la red.

Para la elección del número de neuronas se ha experimentado con potencias de 2. El valor escogido ha sido de 128 neuronas, ya que a partir de este se lograba la convergencia de la red neuronal (teniendo en cuenta el resto de los parámetros), haciendo que un mayor número de neuronas no aumentara el rendimiento de la red, sino lo contrario, llegándose a producir *overfitting* (sobreajuste) para valores superiores a 256 neuronas.

En cuanto a la función de activación escogida ha sido la tangente hiperbólica, ya que es la que mejores resultados ha proporcionado. Mientras que el 'batch size' o tamaño de lote empleado ha sido de 10. Este parámetro se refiere al número de ejemplos de entrenamiento que se utilizan en una iteración para actualizar los pesos del modelo. Un 'batchsize' elevado disminuirá el tiempo de computación, pero a su vez suavizará la predicción al proporcionar una estimación del gradiente más estable. Además, introduce un componente de aleatoriedad, ya que los ejemplos que toma no son continuos, sino que son aleatorios.

El número de épocas escogido ha sido de 20 epochs. Este término hace referencia al número de veces que se pasan los datos de entrenamiento en una iteración ajustando en cada una los pesos de la red. Por lo tanto, cabe esperar que al emplear más epochs el modelo se ajuste más. Sin embargo, cabe la posibilidad de que se produzca *overfitting*, es decir que los datos estén sobreajustados y que por tanto los resultados no sean los esperados. Al contrario, también se puede producir un subajuste (*underfitting*) al emplear un número reducido de epochs.

Tras definir el modelo se tiene que compilar, para ello se hace uso de la función 'compile', cuya función de pérdidas ha sido el MSE y el optimizador escogido ha sido el Adam.

Los parámetros finales escogidos para la red LSTM son los siguientes:

Neuronas	Epochs	Función de pérdidas	Función de activación	Optimizador	Batch size
128	20	MSE	tanh	Adam	10

Tabla 7. *Parámetros de la red LSTM*

Fuente: *Elaboración propia*

Cabe destacar que la elección de estos parámetros ha sido a base de prueba y error, priorizando que los resultados sean correctos y el tiempo de computación no fuera demasiado elevado. Es posible que pueda haber otros parámetros óptimos que incluso mejoren el rendimiento de los actuales, sin embargo, para este Trabajo Fin de Grado se han escogido los que se muestran en la Tabla 7.

El último paso es predecir los valores de prueba o validación, para ello se emplea la función 'predict' al modelo cuya entrada serán los valores de prueba. Por último, se calcula el RMSE y el MAPE y se grafican los resultados de predicción junto con el vector de volatilidad real utilizando la función 'plot' al igual que en la implementación del modelo GARCH.

En la Figura 20 se muestra el diagrama de flujo de la red neuronal LSTM con todos los pasos detallados en este apartado.

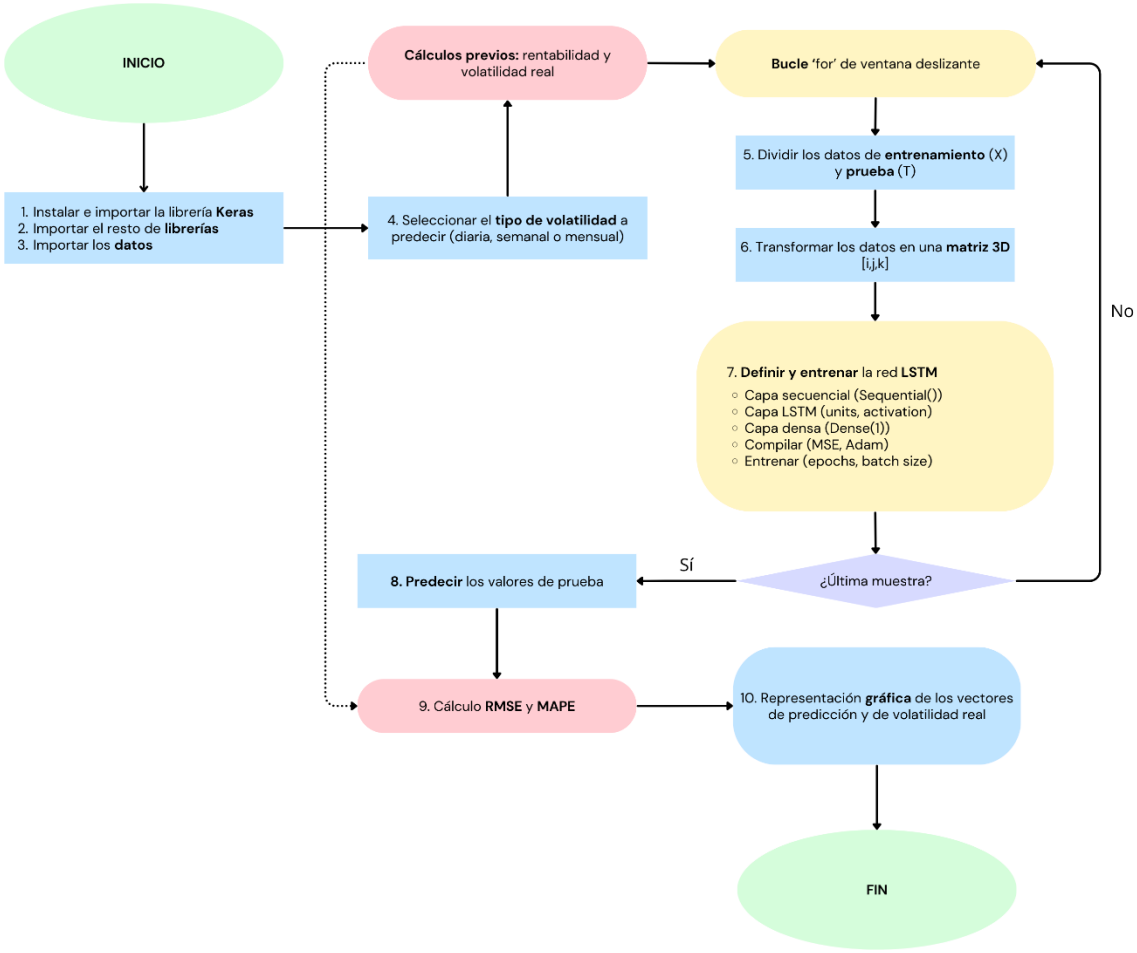


Figura 20. *Diagrama de flujo de la red neuronal LSTM*

Fuente: *Elaboración propia en Canva*

4. Resultados y discusión

En este apartado se evaluarán y compararán los resultados obtenidos por cada modelo, con el fin de obtener una conclusión sobre cuál es el más acertado para predecir tanto la volatilidad diaria como la volatilidad semanal o mensual. Para ello se utilizará el RMSE y el MAPE, y se graficará la volatilidad predicha con la real para visualizar mejor el error o acierto de cada modelo.

4.1 Evaluación de los resultados diarios

Al analizar series con datos de tiempo diarios se debe tener en cuenta la cantidad de muestras u observaciones que se han utilizado para entrenar los modelos, en comparación con las series de tiempo semanales o mensuales. Este gran número de muestras incrementa en gran medida el tiempo de computación, aunque también nos ofrece información más detallada.

Desde el punto de vista del inversor, analizar la volatilidad diaria resulta idónea si se está invirtiendo en el corto plazo o en empresas tan volátiles como Tesla o Meta. Por ello, se van a analizar los resultados de estas dos compañías en primer lugar, donde en color gris se muestra la volatilidad real y en líneas discontinuas se muestra la predicción de ambos modelos, siendo el color verde el que ilustra la predicción realizada por el modelo GARCH y el naranja la de la red LSTM.

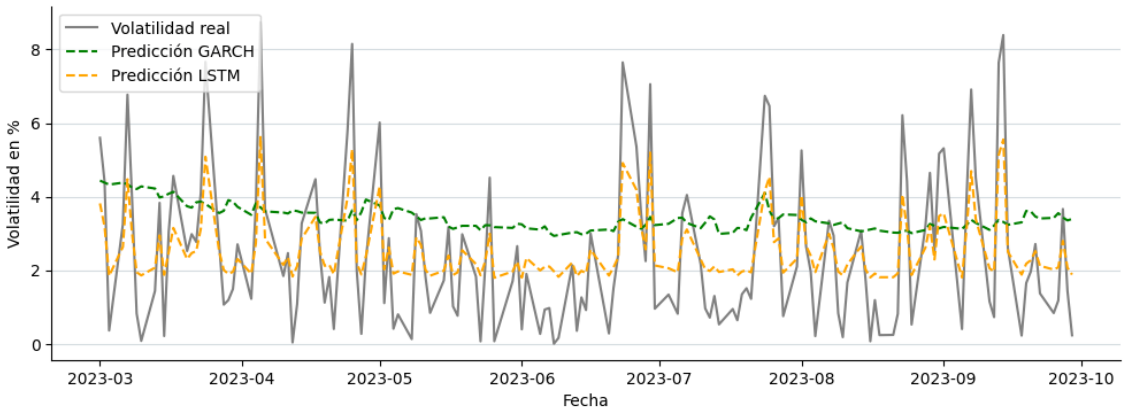


Figura 21. *Predicción de la volatilidad diaria de Tesla*

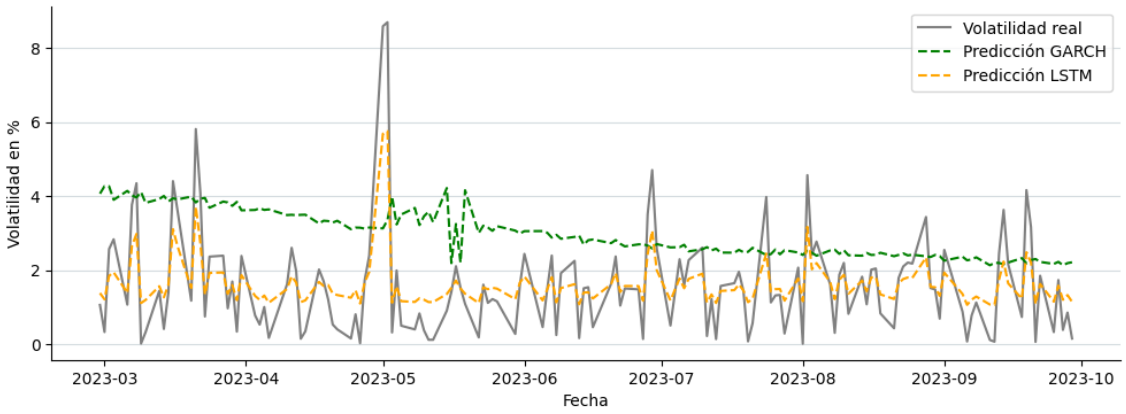


Figura 22. *Predicción de la volatilidad diaria de Meta*

Se aprecia que la predicción de la red LSTM es más acertada en ambas empresas. En cambio, se puede ver que la predicción de la volatilidad de Tesla con el modelo GARCH ha capturado bastante bien la media y la tendencia, logrando acertar algunos incrementos de volatilidad. Sin embargo, la predicción de Meta no ha sido acertada, apreciándose un comportamiento anómalo. Esto puede ser debido a lo inestable y volátil que ha sido siempre esta empresa, por lo que, ante datos de entrada de rentabilidad elevados y cercanos a la fecha de testeo el modelo calcula una media superior a la real, que a medida que pasa el tiempo se va estabilizando, siempre que la rentabilidad se estabilice también.

A diferencia de las dos empresas anteriores, el modelo GARCH resulta más acertado ante empresas menos volátiles como Google (Figura 23) o Amazon (Figura 24), donde el modelo captura mejor la volatilidad, acertando en las fluctuaciones de esta. Aunque, como se puede apreciar, la red LSTM sigue siendo mejor opción.

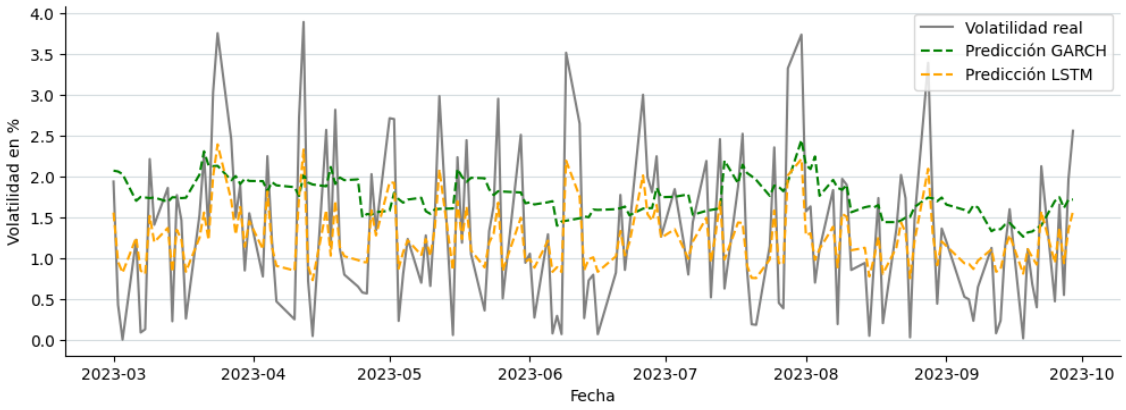


Figura 23. *Predicción de la volatilidad diaria de Google*

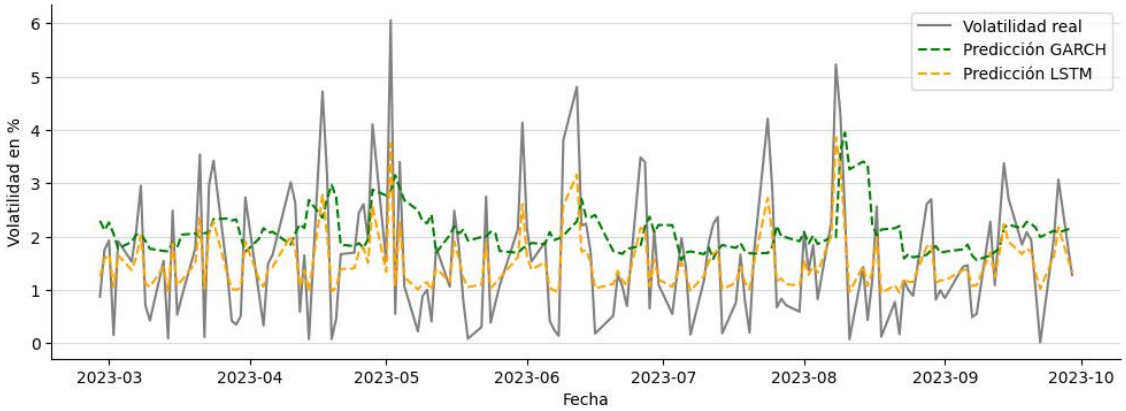


Figura 24. *Predicción de la volatilidad diaria de Amazon*

Las métricas del error RMSE y MAPE cometidas por la predicción de la volatilidad diaria de ambos modelos se muestran en la Tabla 8. Además, también se muestra la diferencia en porcentaje entre los errores del modelo GARCH y los de la red LSTM. Donde, resultados negativos indican que la red LSTM resulta más acertada y viceversa.

	GARCH		LSTM		Diferencia	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
Tesla	2.2286	0.5599	1.2826	0.4342	-42,45%	-22,45%
Google	0.9948	0.4837	0.618	0.4242	-37,88%	-12,30%
Meta	2.0546	0.5728	0.8204	0.4077	-60,07%	-28,82%
Amazon	1.3237	0.5125	0.6955	0.3824	-47,46%	-25,39%
Microsoft	1.0586	0.5276	0.5589	0.3999	-47,20%	-24,20%
Apple	0.8975	0.5076	0.5385	0.4042	-40,00%	-20,37%

Tabla 8. *Métricas del error de predicción de la volatilidad diaria*

Fuente: *Elaboración propia a partir de los resultados obtenidos en Python*

Ante los resultados de la tabla anterior, se puede afirmar que la red neuronal LSTM es la mejor opción, ya que reduce en gran medida el error respecto al modelo GARCH, mejorando los resultados para todas las empresas del estudio. Además, también se observa que Tesla y Meta son las empresas con mayor error asociado en comparación a las demás.

Por último, respecto a los errores cometidos por las empresas Microsoft y Apple son similares a los de Google o Amazon, cuyas gráficas de predicción se encuentran en el Anexo II (Figuras A.1 y A.2).

4.2 Evaluación de los resultados semanales

En cuanto a la predicción de datos semanales, hay que tener en cuenta que su horizonte es a medio o largo plazo y que la cantidad de datos de entrada ha sido menor. Sin embargo, los datos de rentabilidad semanal agrupan a los datos de rentabilidad diaria de ese periodo de tiempo, por lo que responden mejor ante grandes fluctuaciones diarias, ya que las suaviza al tener en cuenta todos los valores de esa semana.

Este efecto se puede apreciar en las empresas Tesla y Meta, donde los resultados de la volatilidad diaria con el modelo GARCH no han sido los esperados. Las predicciones de ambas empresas se muestran en las Figuras 25 y 26, donde se puede ver que la predicción del modelo clásico GARCH resultada más acertada que en las predicciones diarias.

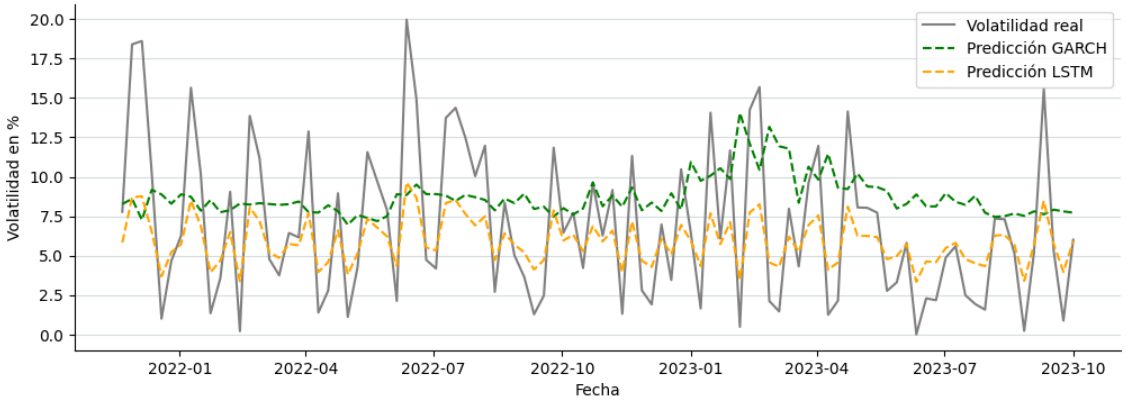


Figura 25. *Predicción de la volatilidad semanal de Tesla*

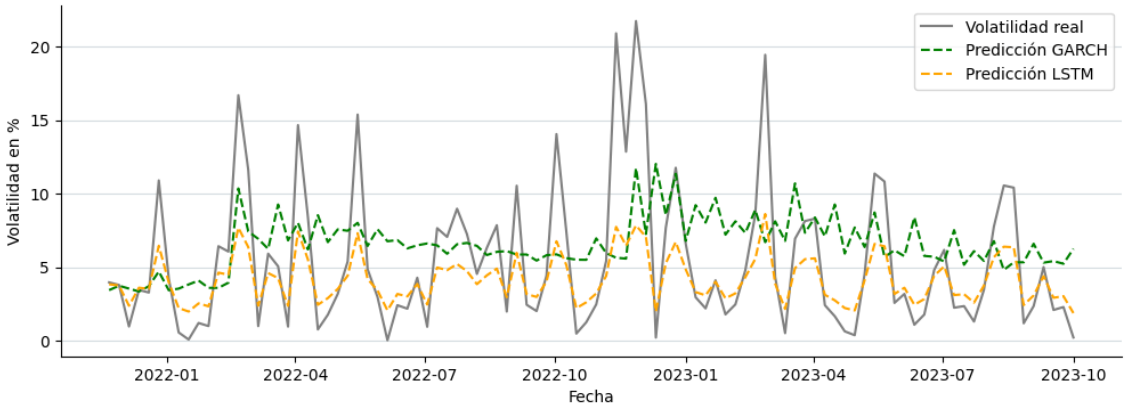


Figura 26. *Predicción de la volatilidad semanal de Meta*

Además, también se puede observar un ligero retardo en la predicción del modelo GARCH. Esto es debido al tamaño de la ventana de predicción, al predecir con un horizonte de 5 valores y una ventana móvil de igual valor, hay ocasiones donde un valor de entrada de rentabilidad elevado que puede cambiar los parámetros del modelo aún no ha sido introducido en este, por lo que no se tiene en cuenta provocando ese retardo. En cambio, se observa que el hecho de utilizar dicha ventana no afecta en ninguna medida a la red LSTM, que no provoca ningún retardo. En el apartado 5.1 se analiza el comportamiento de ambos modelos ante diferentes ventanas, donde para una ventana igual a la unidad el retardo desaparece en el modelo GARCH.

Los errores cometidos por la predicción de la volatilidad semanal de ambos modelos, así como la diferencia entre ellos se muestra en la Tabla 9.

	GARCH		LSTM		Diferencia	
	RMSE	**MAPE**	**RMSE**	**MAPE**	**RMSE**	**MAPE**
Tesla	5,2995	0,5092	3,52	0,4529	-33,58%	-11,06%
Google	3,0962	0,6082	2,0697	0,4134	-33,15%	-32,03%
Meta	4,8685	0,6014	3,5235	0,4835	-27,63%	-19,60%
Amazon	3,3116	0,5334	2,6317	0,4503	-20,53%	-15,58%
Microsoft	2,4571	0,5362	1,6015	0,36	-34,82%	-32,86%
Apple	2,7586	0,5857	1,7069	0,4159	-38,12%	-28,99%

Tabla 9. *Métricas del error de predicción de la volatilidad semanal*

Fuente: *Elaboración propia a partir de los resultados obtenidos*

Donde se observa que la red LSTM sigue siendo la mejor opción. Aunque, la diferencia de los errores cometidos es menor a la del análisis de la volatilidad diaria, por lo que el modelo GARCH parece ser más acertado para series de tiempo semanales.

Sin embargo, llama la atención el gran error que presenta Google en el modelo GARCH, más de un 10% en comparación al análisis anterior. La Figura 27 muestra la predicción de esta empresa, donde se aprecia que el modelo si ha detectado las fluctuaciones que sufre la volatilidad, pero no en la magnitud correspondiente. Esto puede ser debido a la baja volatilidad asociada a esta empresa, que hace que sea menos sensible ante cambios repentinos de la rentabilidad.

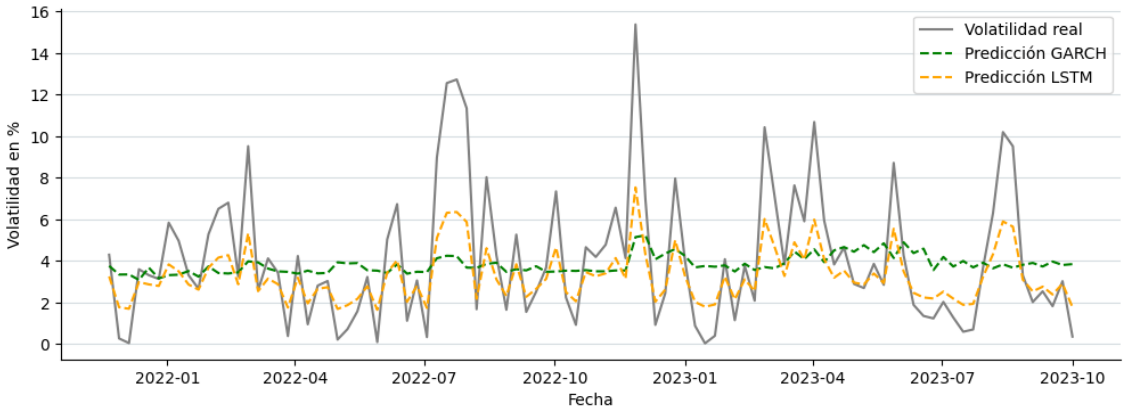


Figura 27. *Predicción de la volatilidad semanal de Google*

En cuanto a Amazon, los resultados de ambos modelos son bastante acertados, y tal como se observa en la Figura 28, el modelo GARCH ha detectado la tendencia y las fluctuaciones con gran acierto a pesar del ligero retardo ya comentado. Aun así, es la red LSTM la que sigue presentando ligeramente mejores resultados.

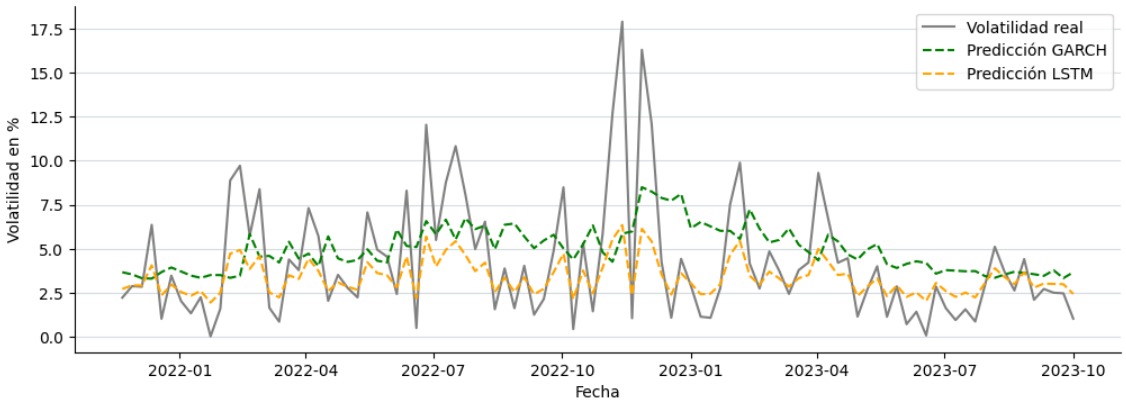


Figura 28. *Predicción de la volatilidad semanal de Amazon*

Respecto a Microsoft y Apple, los resultados son bastante similares y las gráficas de predicción de estas se encuentran en el Anexo II (Figuras A.3 y A.4).

Tras analizar los resultados semanales, la red LSTM sigue siendo la mejor opción, aunque esta vez, la diferencia respecto al modelo GARCH es menor, haciendo de este un modelo también válido.

4.3 Evaluación de los resultados mensuales

Por último, se va a evaluar los resultados de las predicciones mensuales. Estas resultan más atractivas a inversores que operan a largo plazo, ya que cada muestra representa o resume la información de la volatilidad durante un mes. Es la serie con menos muestras introducidas para entrenar los modelos y esto puede afectar a los resultados.

En primer lugar, al igual que en los dos apartados anteriores, se analizarán los resultados obtenidos para las empresas Tesla y Meta, los cuales se muestran en las Figuras 29 y 30 respectivamente.

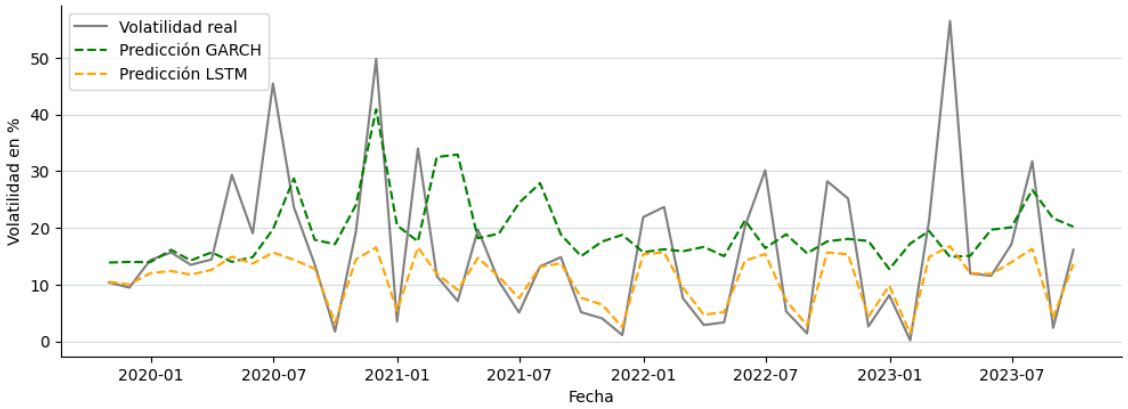


Figura 29. *Predicción de la volatilidad mensual de Tesla*

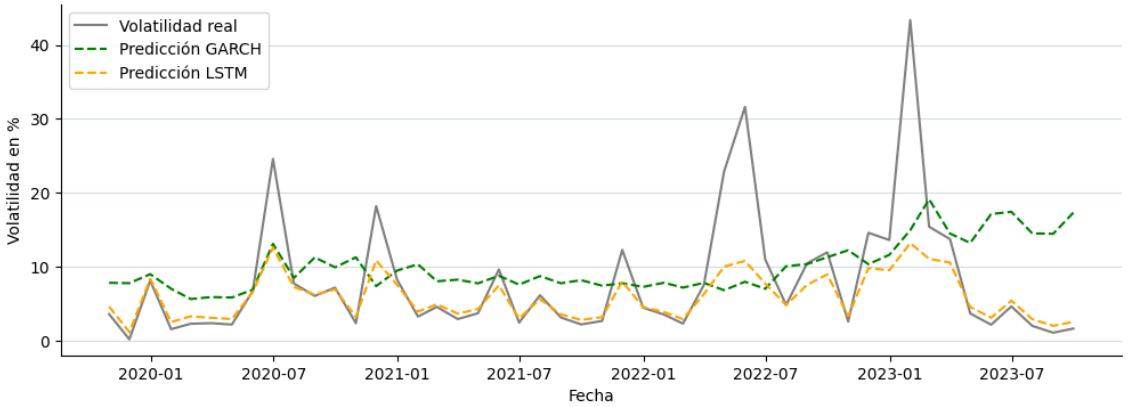


Figura 30. *Predicción de la volatilidad mensual de Meta*

Llama bastante la atención el resultado obtenido por ambos modelos en estas dos empresas, donde, en comparación con las predicciones diarias y semanales analizadas anteriormente, la diferencia entre modelos es mínima. Además, se puede apreciar en los resultados de Tesla que el modelo GARCH logra captar mejor la magnitud de los cambios de la volatilidad.

En la Tabla 10 se muestran las métricas del error cometido por ambos modelos, donde se aprecia esa diferencia mínima del 20% en el RMSE y del 14% en el MAPE para la compañía de Elon Musk. Hay que destacar que el RMSE aumenta respecto al análisis diario o semanal ya que la volatilidad también aumenta, al ser la suma de los datos diarios de un mes.

	GARCH		LSTM		Diferencia	
	RMSE	**MAPE**	**RMSE**	**MAPE**	**RMSE**	**MAPE**
Tesla	13.1239	0.5506	10.419	0.4728	-20,61%	-14,13%
Google	4.9216	0.5975	3.5625	0.3501	-27,62%	-41,41%
Meta	8.7192	0.6404	6.1988	0.3575	-28,91%	-44,18%
Amazon	5.9485	0.5463	4.0727	0.303	-31,53%	-44,54%
Microsoft	4.045	0.5453	1.5918	0.2621	-60,65%	-51,93%
Apple	4.971	0.4854	3.1229	0.2852	-37,18%	-41,24%

Tabla 10. *Métricas del error de predicción de la volatilidad mensual*

Fuente: *Elaboración propia a partir de los resultados obtenidos en Python*

En cuanto a Google, la predicción del modelo GARCH no es buena, ya que no es capaz de captar las fluctuaciones de la volatilidad, tal como se puede ver en la Figura 31. En cambio, la predicción de la red LSTM sí que capta estas alteraciones, pero no en la medida adecuada.

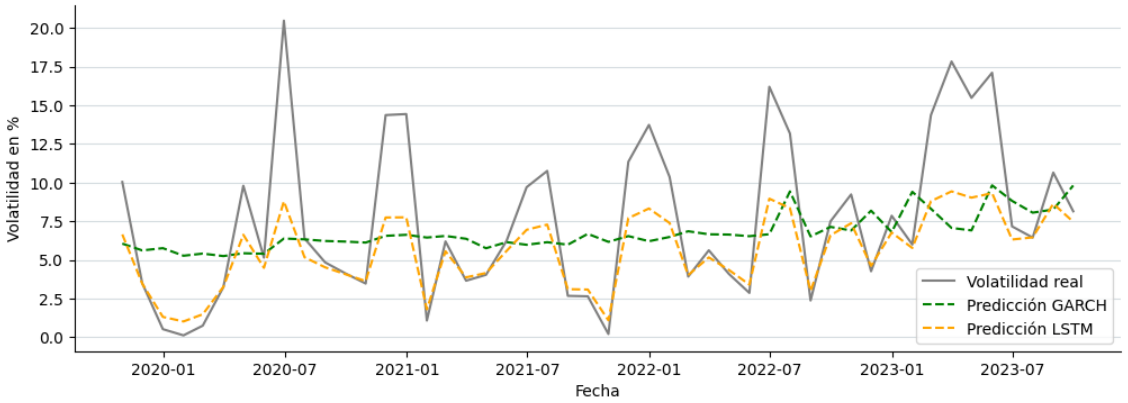


Figura 31. *Predicción de la volatilidad mensual de Google*

Por otro lado, en Amazon, el modelo GARCH sí que capta las fluctuaciones de la volatilidad, acertando en algunas de ellas con bastante exactitud, tal como se aprecia en la Figura 32.

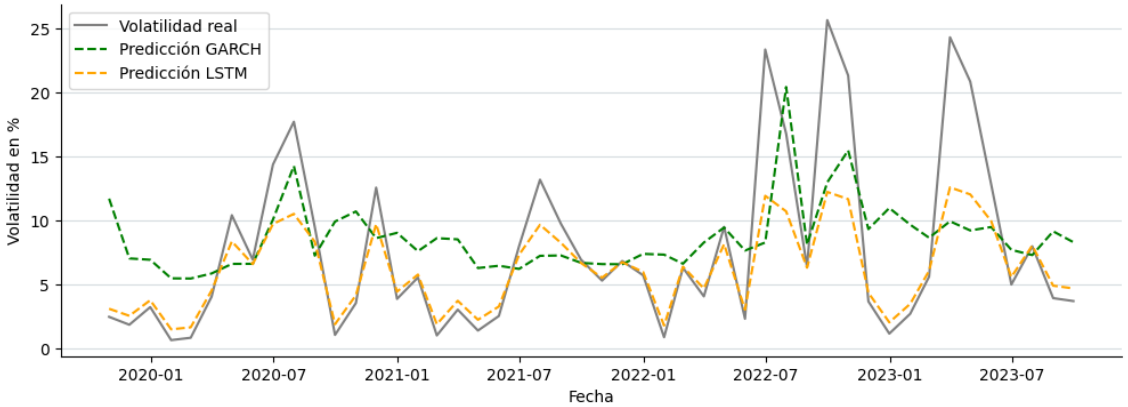


Figura 32. *Predicción de la volatilidad mensual de Amazon*

Respecto a Microsoft y Apple, las predicciones de la red LSTM son bastantes acertadas en comparación a las del modelo GARCH, donde sí que capta algunos movimientos, pero no de la forma adecuada. Al igual que en la evaluación de la volatilidad diaria y semanal, las gráficas de la predicción de la volatilidad de estas empresas se encuentran en el Anexo II (Figura A.5 y A.6).

A la vista de los resultados analizados en este apartado y en los apartados 4.1 y 4.2, se puede ver que el modelo GARCH ha resultado valido y se ha comportado mejor para series de tiempo semanales. No obstante, la red LSTM ha demostrado ser la mejor opción para la predicción de la volatilidad diaria, semanal y mensual, con una notable mejora en la precisión respecto al modelo GARCH, llegando a reducir el error hasta en un 40%.

En cambio, el tiempo que tarda en simular la red neuronal es mucho mayor al del modelo GARCH. Este problema se resuelve en el siguiente apartado, donde se proponen varias alternativas de mejora.

5. Propuestas de mejora

En este apartado se van a comentar varias alternativas de mejora a los modelos definidos en el anterior apartado, así como las ventajas y desventajas de cada una.

5.1 Variación de la ventana de predicción

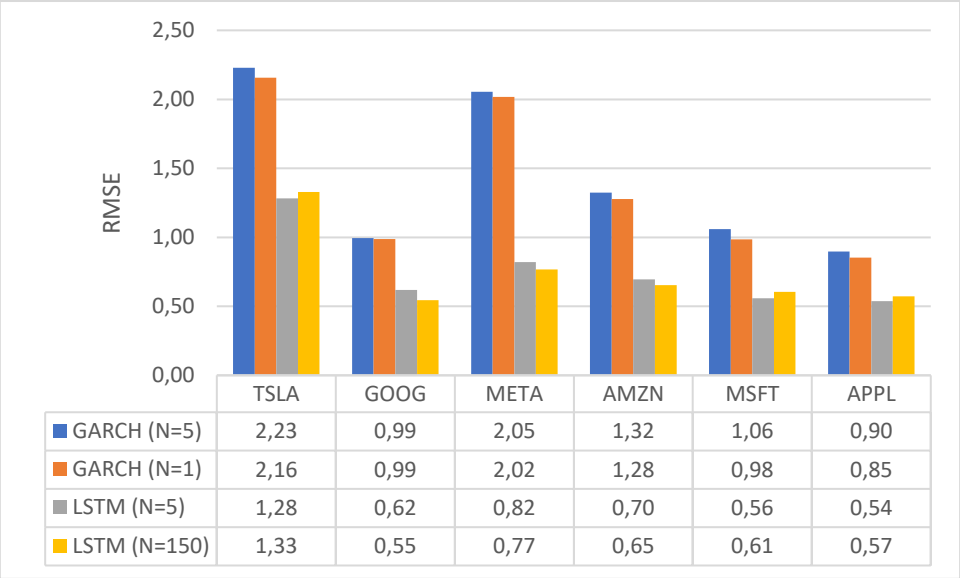
Como se ha mencionado anteriormente, se ha empleado una ventana de predicción para reducir el tiempo de simulación, especialmente en la red LSTM, llegando a 15 minutos de simulación por empresa en el caso de la volatilidad diaria con la ventana de 5 muestras empleada en el anterior apartado.

Tras analizar el comportamiento de ambos modelos ante la variación del tamaño de la ventana se ha observado que el modelo GARCH ante una ventana igual a la unidad los resultados mejoran para todas las empresas de estudio. Sin embargo, al aplicarla en la red LSTM, el tiempo computacional aumenta considerablemente y, los resultados no son tan favorables, sino que más bien son similares a los comentados en el apartado 4.

Este tiempo de computación varía en función del número de observaciones introducidas para entrenar al modelo, así como del número de epochs seleccionado y de las iteraciones del bucle '*for'* de predicción, que a su vez está vinculado con el tamaño de la ventana.

Por lo general, los resultados tienden a ser peores cuanto menor sea el tiempo computacional. Sin embargo, se ha simulado la red neuronal LSTM con una ventana igual al tamaño de prueba, o, lo que es lo mismo sin utilizar ventana de predicción. Los resultados, sorprendentemente, no difieren con los obtenidos con ventana igual a 5, consiguiendo reducir en gran medida el tiempo de simulación. Al contrario, los resultados del modelo GARCH con ventana igual a tamaño de prueba no son correctos, ya que solo capta la media y no las variaciones.

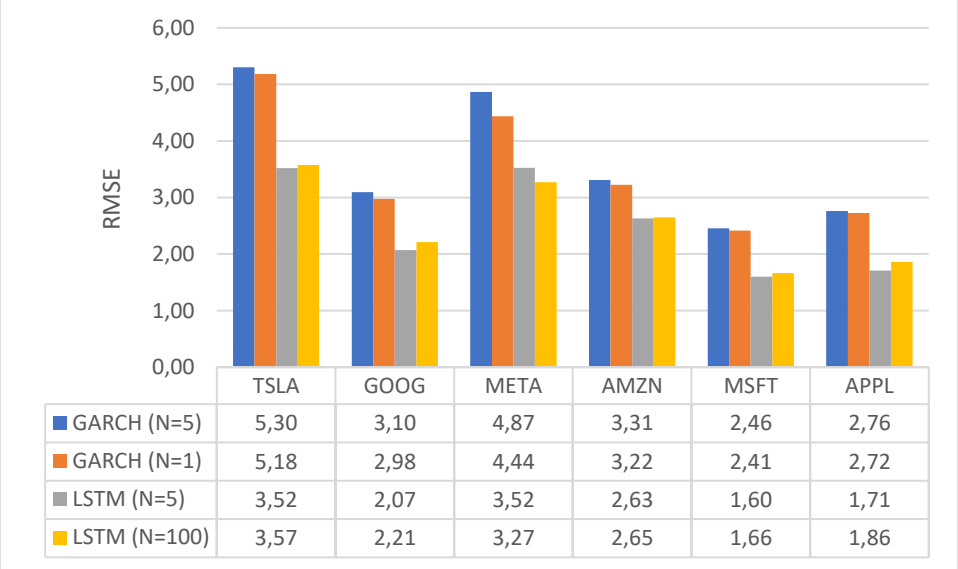
Así pues, se han comparado el RMSE del modelo GARCH con N=1 y de la red LSTM con N igual al tamaño de prueba junto con los resultados obtenidos en el apartado 4. Los RMSE asociados a la volatilidad diaria se ilustran en la Figura 33, mientras que en las Figuras 34 y 35 se ilustran los resultados para la volatilidad semanal y mensual respectivamente, siendo N igual al tamaño de ventana.



	TSLA	GOOG	META	AMZN	MSFT	APPL
GARCH (N=5)	2,23	0,99	2,05	1,32	1,06	0,90
GARCH (N=1)	2,16	0,99	2,02	1,28	0,98	0,85
LSTM (N=5)	1,28	0,62	0,82	0,70	0,56	0,54
LSTM (N=150)	1,33	0,55	0,77	0,65	0,61	0,57

Figura 33. *Comparación RMSE volatilidad diaria*

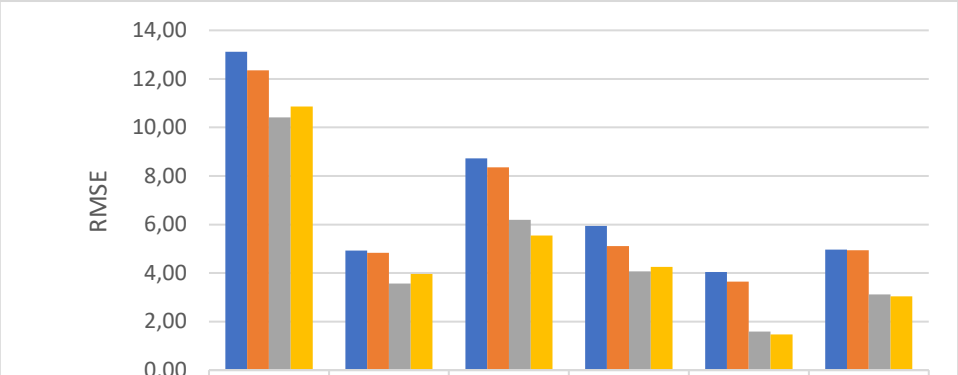
Fuente: *Elaboración propia en Excel a partir de los resultados obtenidos*



	TSLA	GOOG	META	AMZN	MSFT	APPL
GARCH (N=5)	5,30	3,10	4,87	3,31	2,46	2,76
GARCH (N=1)	5,18	2,98	4,44	3,22	2,41	2,72
LSTM (N=5)	3,52	2,07	3,52	2,63	1,60	1,71
LSTM (N=100)	3,57	2,21	3,27	2,65	1,66	1,86

Figura 34. *Comparación RMSE volatilidad semanal*

Fuente: *Elaboración propia en Excel a partir de los resultados obtenidos*



	TSLA	GOOG	META	AMZN	MSFT	APPL
GARCH (N=5)	13,12	4,92	8,72	5,95	4,05	4,97
GARCH (N=1)	12,35	4,84	8,35	5,12	3,65	4,94
LSTM (N=5)	10,42	3,56	6,20	4,07	1,59	3,12
LSTM (N=50)	10,86	3,97	5,55	4,25	1,47	3,04

Figura 35. *Comparación RMSE volatilidad mensual*

Fuente: *Elaboración propia en Excel a partir de los resultados obtenidos*

A la vista de los resultados obtenidos, se observa que el modelo GARCH con ventana igual a la unidad mejora ligeramente los resultados respecto al de ventana con valor 5 en los tres tipos de volatilidad. Sin embargo, esta mejora no es suficiente en comparación con los resultados de la red LSTM, donde se aprecia que la simulación sin ventana aporta resultados similares a la de N=5, hasta en algunos casos consigue mejorarla. Las gráficas de los resultados de esta simulación se encuentran en la Parte II del Anexo II.

Tras esta comprobación, se puede reafirmar que la red LSTM es el modelo más apropiado. Dónde el único inconveniente que tenía, el cual era el tiempo de simulación, se ha visto reducido drásticamente al no aplicar ningún tipo de ventana, cuyos resultados prácticamente no varían respecto al analizado en los apartados 4.1, 4.2 y 4.3.

5.2 Simulación con el modelo EGARCH

En este apartado se ha simulado el modelo EGARCH y comparado respecto al modelo GARCH, con tal de ver las diferencias entre ambos, ya que, como se ha introducido en el apartado 2.3.2, el modelo EGARCH mejora los problemas de simetría del modelo GARCH.

Respecto a la implementación en Python ha sido la misma que el modelo GARCH, simplemente se ha cambiado el atributo ‘vol’, el cual hace referencia al modelo de volatilidad, de ‘GARCH’ a ‘EGARCH’.

Los resultados del error de predicción de la volatilidad diaria para una ventana igual a uno se muestran en la Tabla 11. Donde se puede apreciar que los resultados son similares a los del modelo GARCH. Sin embargo, llama la atención que el error de META para la volatilidad diaria mejora bastante. Esto se puede deber a que los precios de cierre sufren una gran caída, provocando rendimientos negativos. Como se ha comentado, el modelo GARCH no se comporta igual ante movimientos negativos que positivos. Este problema de simetría se ve solucionado con el modelo EGARCH. En la Figura 36 se puede ver claramente este ejemplo, donde el nuevo modelo mostrado con una línea discontinua de color azul capta mejor las fluctuaciones de la volatilidad de esta compañía.

	GARCH		EGARCH		Diferencia	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
Tesla	2,1559	0,5453	2,1801	0,5348	-1,12%	1,93%
Google	0,9871	0,4807	1,0022	0,4827	-1,53%	-0,42%
Meta	2,0185	0,5722	1,7096	0,5447	15,30%	4,81%
Amazon	1,2773	0,5152	1,2625	0,5058	1,16%	1,82%
Microsoft	0,9844	0,4965	1,0320	0,5141	-4,84%	-3,54%
Apple	0,8522	0,4893	0,8660	0,4965	-1,62%	-1,47%

Tabla 11. *Métricas del error de predicción de la volatilidad diaria GARCH vs EGARCH*

Fuente: *Elaboración propia a partir de los resultados obtenidos en Python*

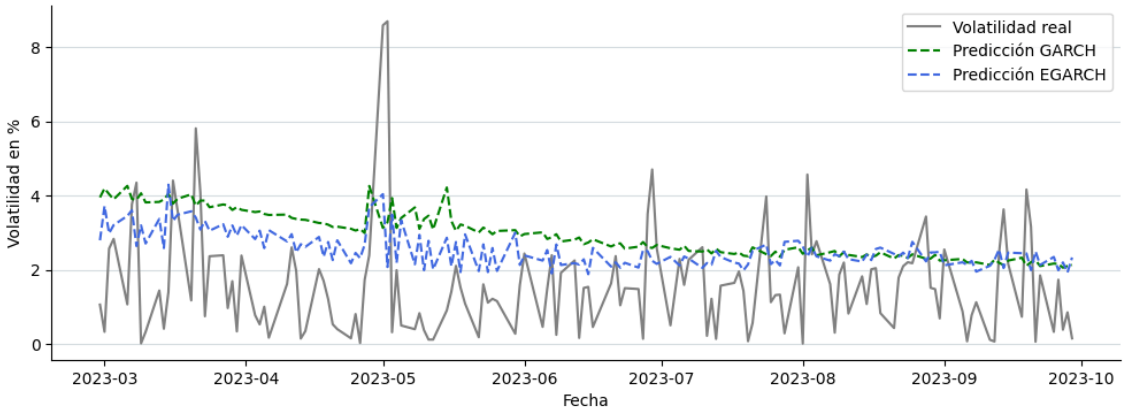


Figura 36. *Predicción de la volatilidad diaria de Meta (GARCH vs EGARCH)*

El resto de las gráficas y resultados de la volatilidad semanal y mensual se encuentran en el Anexo III. Donde cabe destacar que los resultados obtenidos por el modelo EGARCH para la volatilidad semanal mejoran a los del anterior modelo. Aunque, los resultados de la volatilidad mensual empeoran.

Este modelo mejora ligeramente los resultados obtenidos por el modelo GARCH para una ventana de uno, especialmente en los casos donde se aprecian los problemas de simetría. Sin embargo, sigue sin alcanzar los resultados que se logran con la red neuronal LSTM.

6. Conclusiones y líneas futuras

6.2.1 Conclusiones

En primer lugar, atendiendo a los resultados obtenidos por cada modelo, se llega a la conclusión de que la red neuronal LSTM tiene un mejor rendimiento, logrando reducir en gran medida el error cometido por el modelo GARCH en todos los casos de estudio. Aunque, como se ha visto en el apartado 4.2, el modelo clásico ha resultado válido en la predicción de la volatilidad semanal.

Atendiendo a las tablas de los errores RMSE y MAPE y analizando las gráficas de predicción de la volatilidad, se observa que la red LSTM presenta una mejor capacidad de predicción, independientemente de la empresa o del tipo de volatilidad estudiada. Según los resultados obtenido en el apartado 4, la red LSTM logra reducir entorno a un 50% el RMSE cometido por el modelo GARCH para la volatilidad diaria y alrededor de un 30% para la volatilidad semanal y mensual. En cuanto al MAPE, se aprecia una reducción media respecto al modelo clásico del 25% para los resultados diarios y semanales, y aproximadamente un 40% para los resultados mensuales.

Sin embargo, aunque los resultados de la red neuronal son bastante buenos, cabe destacar el principal conveniente de esta, que es el tiempo de computación excesivo que presenta al simular muestra a muestra. Esto no sucede con el modelo GARCH, cuyo tiempo es aceptable. Es por esto por lo que en el apartado 4 se ha utilizado una ventana de predicción de 5 valores, con tal de conseguir un tiempo de simulación más bajo.

En segundo lugar, con tal de mejorar los resultados obtenidos por el modelo GARCH, se han propuesto dos alternativas de mejora. En la primera, se experimenta con la variación del tamaño de ventana. Para ello, se ha probado con el caso mínimo y máximo, es decir con tamaño igual a la unidad y sin ventana, es decir introduciendo todos los datos de entrenamiento en vez de por lotes. Analizando los resultados mostrados en el apartado 5.1, se llega a la conclusión que con N=1, los resultados del modelo GARCH mejoran ligeramente en los tres tipos de volatilidad. Sin embargo, esta mejora no se acerca a los resultados de la red LSTM, donde se aprecia que la simulación sin ventana aporta resultados similares a la de N=5, hasta en algunos casos consigue mejorarla, solucionando a su vez el problema del tiempo de computación, reduciéndose a tiempos similares a los del modelo clásico.

Por otro lado, como segunda alternativa se ha utilizado el modelo EGARCH, con el objetivo de solucionar los problemas de simetría que presenta el modelo GARCH. Según los resultados del apartado 5.2, este modelo consigue mejorar de forma general al anterior, aunque sigue sin poder competir con los resultados obtenidos por la red LSTM.

Un ejemplo de mejora de este modelo se puede ver en los resultados de la volatilidad diaria para la empresa META, donde el modelo GARCH proporcionaba resultados incoherentes (Figura 22) al no capturar bien los movimientos de esta. Como se muestra en la Figura 36, los resultados obtenidos por el modelo EGARCH son más coherentes, pero no mejoran los de la red neuronal.

Los resultados obtenidos en este Trabajo Fin de Grado son consistentes con otros estudios similares, como por ejemplo el de S. Siami-Namini, N. Tavakoli y A. Siami Namin [27], titulado 'A Comparison of ARIMA and LSTM in Forecasting Time Series'. Que compara el modelo clásico ARIMA con la red LSTM, resaltando el beneficio de aplicar algoritmos y técnicas basadas en aprendizaje profundo. Donde en su caso, se consiguió mejorar en promedio un 85% la predicción de series temporales.

En resumen, se han alcanzado los objetivos propuestos en el primer apartado, donde, tras analizar todos los resultados obtenidos se puede afirmar que la red neuronal es el mejor modelo a la hora de predecir la volatilidad de acciones bursátiles. Haciendo de esta una gran herramienta para inversores y profesionales del mercado, con tal de generar predicciones de volatilidad que ayuden en la toma de decisiones financieras.

6.2.2 Líneas futuras

A pesar de los resultados satisfactorios obtenidos por la red neuronal LSTM, existen diversas vías de investigación que podrían ser exploradas en futuros estudios. A continuación, se proponen algunas líneas futuras que podrían contribuir al avance en el campo de la predicción de volatilidad financiera, así como en la mejora del actual estudio:

- Optimización de los parámetros de la red neuronal: Como se ha comentado, los parámetros escogidos dan resultados bastante acertados, sin embargo, no tienen por qué ser los óptimos. Por lo que se propone seguir experimentando con diferentes arquitecturas de red, tamaños de capa, funciones de activación y otros parámetros relevantes para mejorar aún más la precisión del modelo.

- Uso de otros modelos: Se pueden emplear otros modelos de aprendizaje profundo como Redes Neuronales Residuales o Redes Neuronales Encoder-Decoder. Así como explorar otros modelos econométricos como el TGARCH o el IGARCH.

- Aplicación en otros mercados y activos: Extender el estudio a otros mercados financieros, como el mercado de las criptomonedas, conocido por su gran volatilidad, o también a otros activos, como índices bursátiles, bonos o divisas, con tal de evaluar la capacidad de generalización de los modelos. Cada mercado puede tener características únicas, y adaptar los modelos a diferentes contextos podría ser crucial. Cada mercado presenta características únicas y sería de gran utilidad investigar y adaptar los modelos a diferentes contextos.

- Análisis de sensibilidad: Se propone realizar un análisis de sensibilidad para comprender cómo diferentes factores afectan al rendimiento del modelo, proporcionando una mayor robustez al estudio, así como una mejor generalización de los modelos.

Además, una interesante vía de investigación abierta para futuros estudios es la propuesta de un modelo híbrido entre GARCH y LSTM. Este enfoque se ha explorado en profundidad en el trabajo de E. Koo y G. Kim [16], titulado 'A Hybrid Prediction Model Integrating GARCH Models With a Distribution Manipulation Strategy Based on LSTM Networks for Stock Market Volatility,'. Donde, logran un rendimiento destacado en comparación con modelos GARCH, aunque también sugieren posibles investigaciones futuras para mejorar la precisión predictiva.

7. ODS

Este proyecto se enmarca en los Objetivos de Desarrollo Sostenible (ODS) 8 y 9, impulsando la innovación y el crecimiento económico a través de la mejora de las prácticas en econometría y análisis de series temporales. Específicamente, se centra en la predicción de la volatilidad de las seis grandes tecnológicas del NASDAQ-100, un área de gran relevancia para la economía global.

El proyecto demuestra un profundo conocimiento de los problemas económicos y sociales actuales que afectan a las empresas y generan incertidumbre en los inversores. En este contexto tan complejo, el desarrollo de nuevos modelos y técnicas como las que se presentan se vuelve crucial para facilitar la toma de decisiones.

En primer lugar, el proyecto contribuye al ODS 9: Industria, innovación e infraestructuras. El uso de la inteligencia artificial para crear modelos predictivos representa un avance significativo en el campo de la econometría.

En segundo lugar, el proyecto también se alinea con el ODS 8: Trabajo decente y crecimiento económico. Las técnicas desarrolladas en este trabajo pueden ser utilizadas por inversores para mejorar sus decisiones financieras, lo que a su vez puede conducir a una mayor estabilidad y crecimiento en los mercados financieros. Además, la mayor precisión en las predicciones de volatilidad y el campo de la inteligencia artificial en general puede fomentar la creación de empleos, impulsando el crecimiento económico sostenible.

En definitiva, este proyecto es un ejemplo de cómo la innovación tecnológica puede contribuir al crecimiento económico y lograr un futuro más próspero y sostenible.

Figura 37. *Objetivos de desarrollo sostenible alcanzados*

8. Bibliografía

[1] Andersen, T., Bollerslev, T., & Hadi, A. (2014). *ARCH and GARCH models*. John Wiley & Sons.

[2] Almaraz. J. S (2014, 31 marzo). *"¿Por qué usar rendimientos logarítmicos?"* en Quantdare.com <https://quantdare.com/por-que-usar-rendimientos-logaritmicos/>

[3] Cosio, N. A. L. (2021, 21 diciembre). *Métricas en regresión.* Medium. <https://medium.com/@nicolasarrioja/m%C3%A9tricas-en-regresi%C3%B3n-5e5d4259430b>

[4] Techtalks. (2019, 22 octubre). *Qué son las redes neuronales y sus funciones*. ATRIA Innovation. <https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/>

[5] Balasubramanian, R. (2020, 13 diciembre). *Univariate Forecasting for the Volatility of the Stock Data using Deep Learning.* Medium. <https://medium.com/analytics-vidhya/univariate-forecasting-for-the-volatility-of-the-stock-data-using-deep-learning-6c8a4df7edf9>

[6] Bollerslev, T., Engle, R. F., & Nelson, D. B. (1994). ARCH models. *Handbook of econometrics*, *4*, 2959-3038.

[7] Bollerslev, T. (2008). Glossary to arch (garch). *CREATES Research paper*, *49*.

[8] Box, G. E. P., & Jenkins, G. M. (1973). Some Comments on a Paper by Chatfield and Prothero and on A Review by Kendall. *Journal of the Royal Statistical Society. Series A (General)*, *136*(3), 337–352. https://doi.org/10.2307/2344995

[9] Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

[10] Boyte-White, C. (2023, 31 octubre). *What Is the Best Measure of Stock Price Volatility? Investopedia.* <https://www.investopedia.com/ask/answers/021015/what-best-measure-given-stocks-volatility.asp>

[11] Castillo, L., Maldonado, D. (2020). Redes Neuronales. <https://rstudio-pubs-static.s3.amazonaws.com/599210_4306c5d3d6a34a6c829566ca11b7e27a.html#8>

[12] Chicco D, Warrens MJ, Jurman G. 2021. The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. *PeerJ Computer Science* 7:e623 https://doi.org/10.7717/peerj-cs.623

[13] David A. Dickey & Wayne A. Fuller (1979) Distribution of the Estimators for Autoregressive Time Series with a Unit Root, Journal of the American Statistical Association, 74:366a, 427-431, DOI: 10.1080/01621459.1979.10482531

[14] Evo Banco. *"¿Qué empresas están en el Nasdaq 100?"* en evobanco.com <https://www.evobanco.com/ayuda/al-dia-con-EVO/finanzas/que-empresas-estan-en-el-nasdaq-100/>

[15] Expansión (2018, 27 julio). *Facebook pierde 120.000 millones en un día, la mayor caída en la historia de la Bolsa*. <https://www.expansion.com/mercados/2018/07/27/5b5ad882e5fdeacc6a8b4659.html>

[16] E. Koo and G. Kim, "A Hybrid Prediction Model Integrating GARCH Models With a Distribution Manipulation Strategy Based on LSTM Networks for Stock Market Volatility," in *IEEE Access*, vol. 10, pp. 34743-34754, 2022, doi: 10.1109/ACCESS.2022.3163723

[17] González, L. (2022, 14 septiembre). *¿Qué es el Perceptrón? Perceptrón Simple y Multicapa.* Aprende IA. <https://aprendeia.com/que-es-el-perceptron-simple-y-multicapa/>

[18] Herrera, D (2020). *Predicción para el mercado de acciones con redes neuronales LSTM*. Trabajo Fin de Grado. Bogotá: Universidad Jorge Tadeo Lozano.
[19] Inteligenciartificiallmca. (2017, 11 junio). *1.2.- Clasificación de las Redes Neuronales.* Inteligencia Artificial. <https://inteligenciartificiallmca.wordpress.com/2017/06/10/1-2-clasificacion-de-las-redes-neuronales/>
[20] Monsegny, M. C., & Cuervo, E. C. (2008). Modelos ARCH, GARCH y EGARCH: aplicaciones a series financieras. *Cuadernos de economía*, *27*(48), 287-320.
[21] Nau, R. (2014). The mathematical structure of arima models. Duke University Online Article, 1(1), 1-8.
[22] Olah, C. (2015, 27 agosto). *Understanding LSTM Networks*. Colah's Blog. <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>
[23] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6), 386.
[24] Sullivan, J. *Stock Price Volatility Prediction with Long ShortTerm Memory Neural Networks.* Proyecto. Stanford, CA: Stanford University.
[25] S. C. Hillmer & G. C. Tiao (1982) An ARIMA-Model-Based Approach to Seasonal Adjustment, Journal of the American Statistical Association, 77:377, 63-70, DOI: 10.1080/01621459.1982.10477767
[26] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in Neural Computation, vol. 9, no. 8, pp. 1735-1780, 15 Nov. 1997, doi: 10.1162/neco.1997.9.8.1735
[27] S. Siami-Namini, N. Tavakoli and A. Siami Namin, "A Comparison of ARIMA and LSTM in Forecasting Time Series," *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Orlando, FL, USA, 2018, pp. 1394-1401, doi: 10.1109/ICMLA.2018.00227
[28] Silva, S., Freire, E. *Intro a las redes neuronales convolucionales.* Bootcamp AI-Medium. <https://bootcampai.medium.com/redes-neuronales-convolucionales-5e0ce960caf8>
[29] Thai, K. L. C. (2022, 30 noviembre). How to Predict Stock Volatility Using GARCH Model In Python. Medium. <https://medium.datadriveninvestor.com/how-to-predict-stock-volatility-using-garch-model-in-python-df5ba46bae35>
[30] Unipython Blog (2019). *"Predicción con series temporales con LSTM, redes neuronales recurrentes"* en Unipython.com <https://unipython.com/prediccion-con-series-temporales-con-lstm-redes-neuronales-recurrentes/>
[31] Vennerød, C. B., Kjærran, A., & Bugge, E. S. (2021). Long short-term memory RNN. *arXiv preprint arXiv:2105.06756*.
[32] Xpikuos. (2021, 28 febrero). REDES NEURONALES LSTM RECURRENTES (RNN) IA - NO MÁS DUDAS! 2022 [Vídeo]. YouTube. https://www.youtube.com/watch?v=ppJA9iyByNo

9. Anexos

9.1 Anexo I

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ADE Facultat d'Administració i Direcció d'Empreses /UPV

ANEXO I. RELACIÓN DEL TRABAJO CON LOS OBJETIVOS DE DESARROLLO SOSTENIBLE DE LA AGENDA 2030

Anexo al Trabajo de Fin de Grado y Trabajo de Fin de Máster: Relación del trabajo con los Objetivos de Desarrollo Sostenible de la agenda 2030.

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**			X	
ODS 2. **Hambre cero.**			X	
ODS 3. **Salud y bienestar.**			X	
ODS 4. **Educación de calidad.**		X		
ODS 5. **Igualdad de género.**			X	
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**			X	
ODS 8. **Trabajo decente y crecimiento económico.**	X			
ODS 9. **Industria, innovación e infraestructuras.**	X			
ODS 10. **Reducción de las desigualdades.**		X		
ODS 11. **Ciudades y comunidades sostenibles.**			X	
ODS 12. **Producción y consumo responsables.**			X	
ODS 13. **Acción por el clima.**		X		
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**		X		
ODS 17. **Alianzas para lograr objetivos.**			X	

Descripción de la alineación del TFG/TFM con los ODS con un grado de relación más alto.

***Utilice tantas páginas como sea necesario.

El presente trabajo se alinea con el ODS 8: Trabajo decente y crecimiento económico , y con el ODS 9: Industria, innovación e infraestructuras.

El capítulo 7. ODS (Página 49) describe estos ODS y su alineacion con el presente TFG.

Editor: Secretaría General / UPV · D.L.: V-5092-2006 · ISSN: 1887-2298 · www.upv.es/secgen

Anexo al Trabajo de Fin de Grado y Trabajo de Fin de Máster: Relación del trabajo con los Objetivos de Desarrollo Sostenible de la agenda 2030. (Numere la página)

Este proyecto se enmarca en los Objetivos de Desarrollo Sostenible (ODS) 8 y 9, impulsando la innovación y el crecimiento económico a través de la mejora de las prácticas en econometría y análisis de series temporales. Específicamente, se centra en la predicción de la volatilidad de las seis grandes tecnológicas del NASDAQ-100, un área de gran relevancia para la economía global.

El proyecto demuestra un profundo conocimiento de los problemas económicos y sociales actuales que afectan a las empresas y generan incertidumbre en los inversores. En este contexto tan complejo, el desarrollo de nuevos modelos y técnicas como las que se presentan se vuelve crucial para facilitar la toma de decisiones.

En primer lugar, el proyecto contribuye al ODS 9: Industria, innovación e infraestructuras. El uso de la inteligencia artificial para crear modelos predictivos representa un avance significativo en el campo de la econometría.

En segundo lugar, el proyecto también se alinea con el ODS 8: Trabajo decente y crecimiento económico. Las técnicas desarrolladas en este trabajo pueden ser utilizadas por inversores para mejorar sus decisiones financieras, lo que a su vez puede conducir a una mayor estabilidad y crecimiento en los mercados financieros. Además, la mayor precisión en las predicciones de volatilidad y el campo de la inteligencia artificial en general puede fomentar la creación de empleos, impulsando el crecimiento económico sostenible.

En definitiva, este proyecto es un ejemplo de cómo la innovación tecnológica puede contribuir al crecimiento económico y lograr un futuro más próspero y sostenible.

Editor: Secretaria General / UPV · D.L.: V-5092-2006 · ISSN: 1887-2298 · www.upv.es/secgen

9.2 Anexo II

El presente anexo se compone de dos partes. La primera parte está compuesta por las gráficas de los resultados de predicción de las empresas Microsoft y Apple, analizadas en el apartado 4, cuya ventana deslizante ha sido de 5 registros para ambos modelos.

En cuanto a la segunda parte, se compone de todas las gráficas de cada una de las empresas analizadas en este trabajo, correspondientes a los resultados analizados en el apartado 5.1. Cuya simulación del modelo GARCH ha sido con una ventana de valor 1, mientras que la simulación de la red LSTM ha sido con una ventana igual al tamaño de prueba correspondiente.

Las gráficas siguen el mismo orden de aparición de las empresas y del tipo de volatilidad analizada en el trabajo.

I Parte

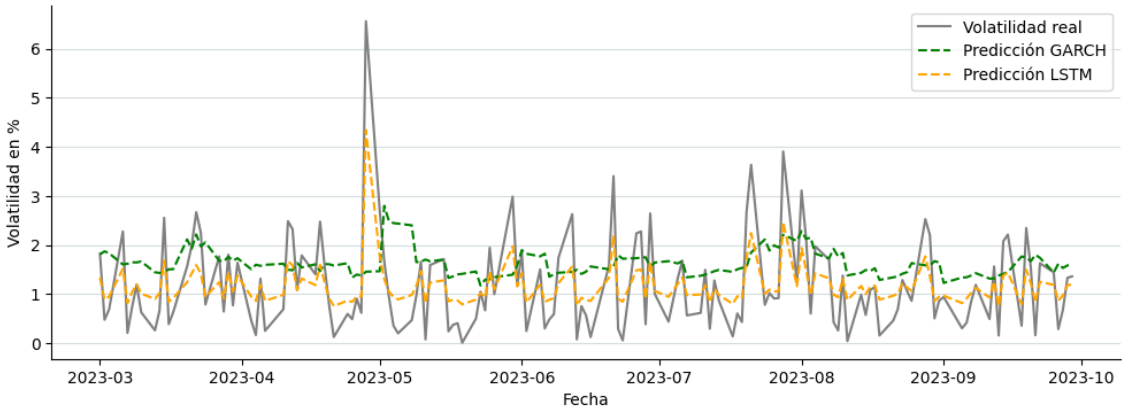


Figura A. 1. *Predicción de la volatilidad diaria de Microsoft*

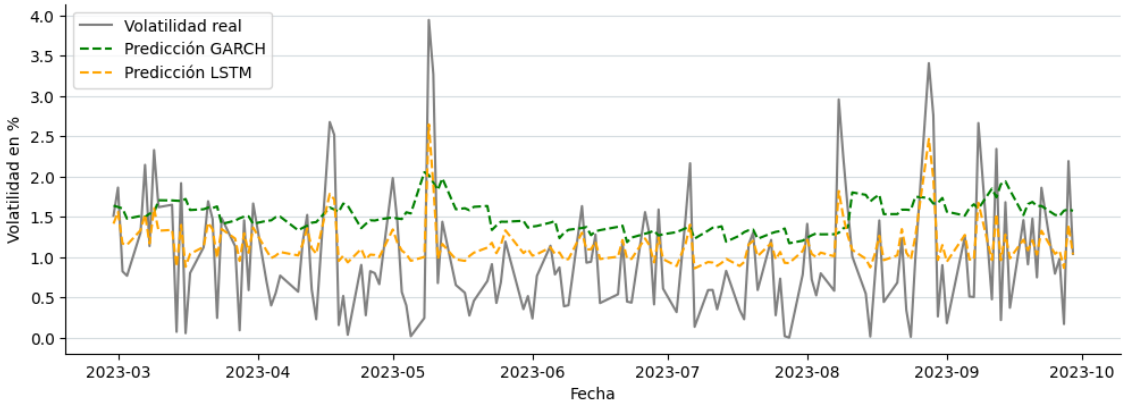


Figura A. 2. *Predicción de la volatilidad diaria de Apple*

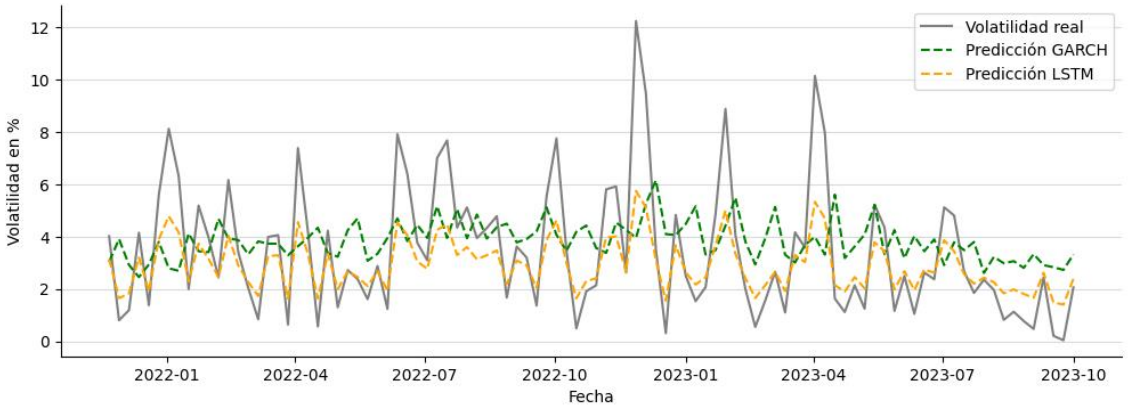


Figura A. 3. *Predicción de la volatilidad semanal de Microsoft*

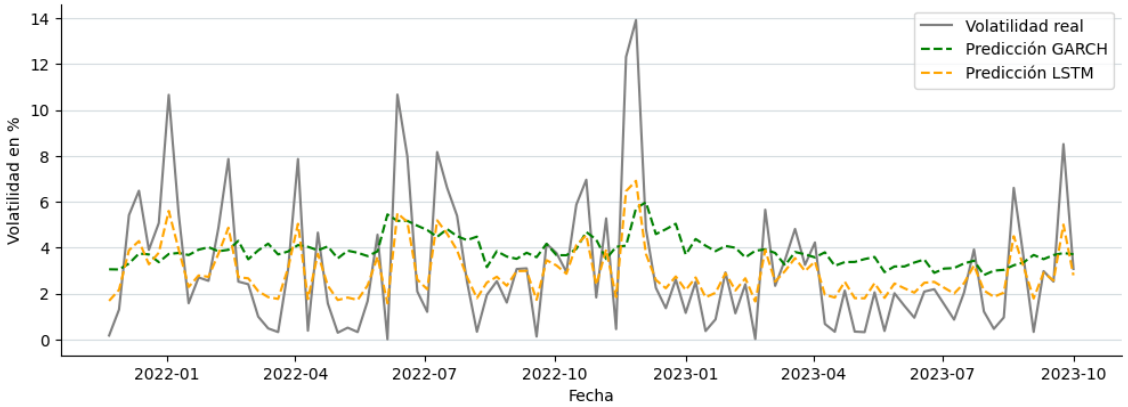


Figura A. 4. *Predicción de la volatilidad semanal de Apple*

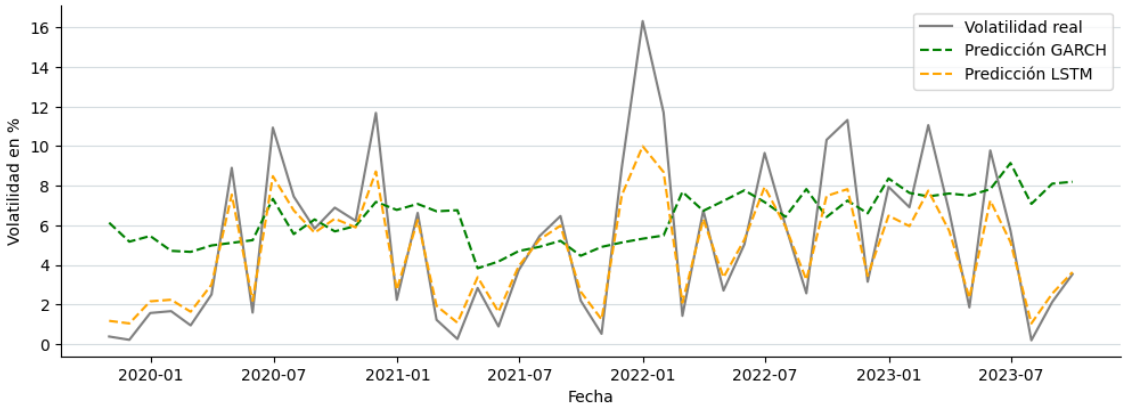


Figura A. 5. *Predicción de la volatilidad mensual de Microsoft*

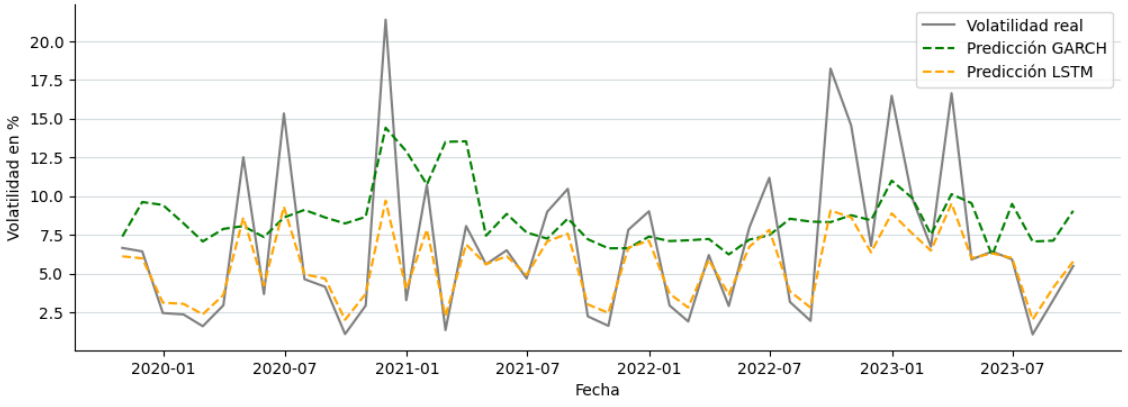


Figura A. 6. *Predicción de la volatilidad mensual de Apple*

II Parte

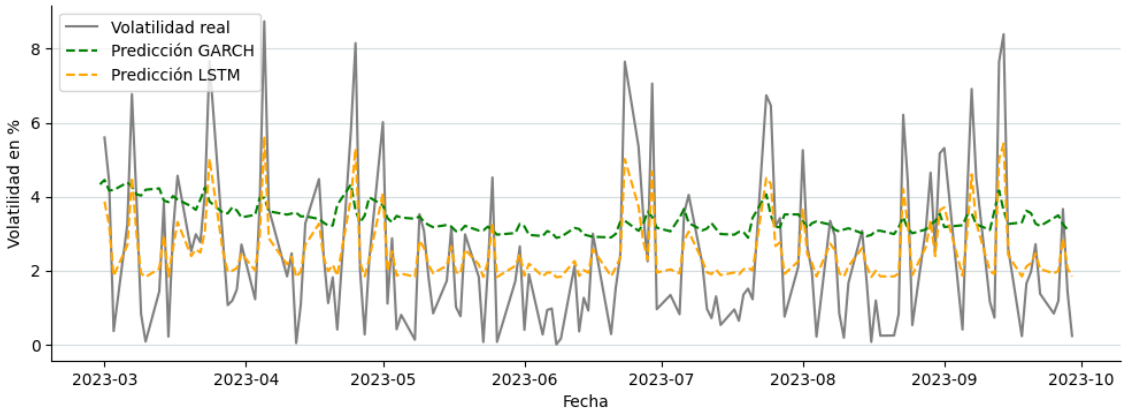


Figura A. 7. *Predicción de la volatilidad diaria de Tesla II*

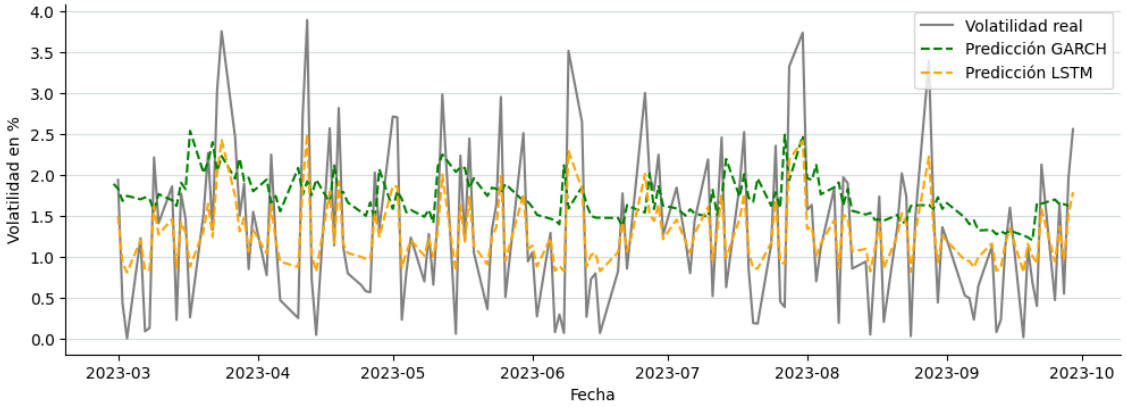


Figura A. 8. *Predicción de la volatilidad diaria de Google II*

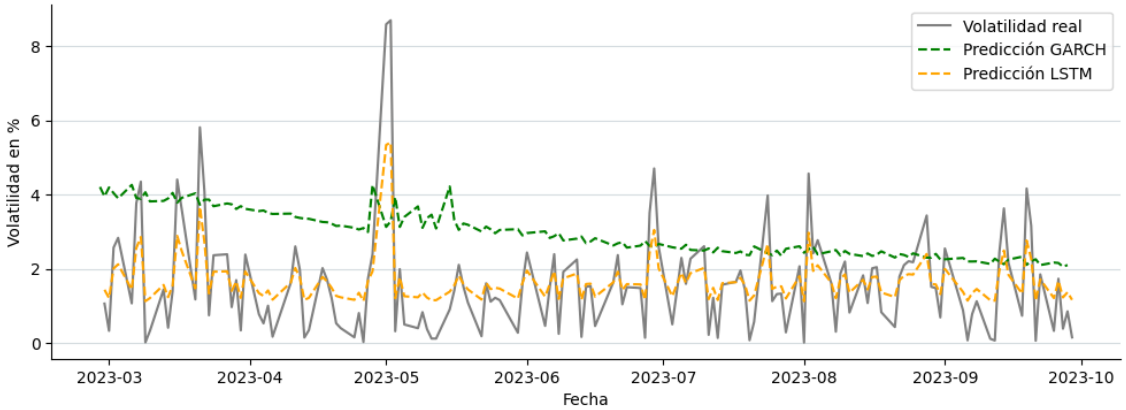


Figura A. 9. *Predicción de la volatilidad diaria de Meta II*

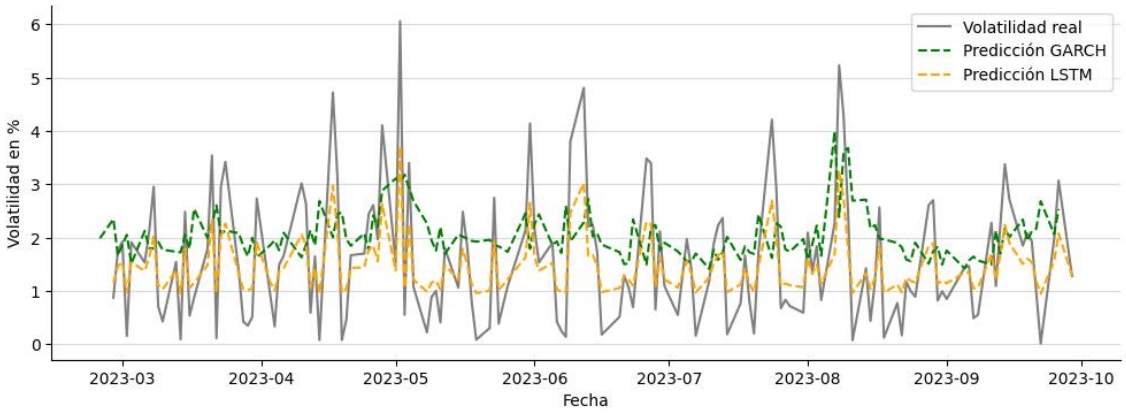


Figura A. 10. *Predicción de la volatilidad diaria de Amazon II*

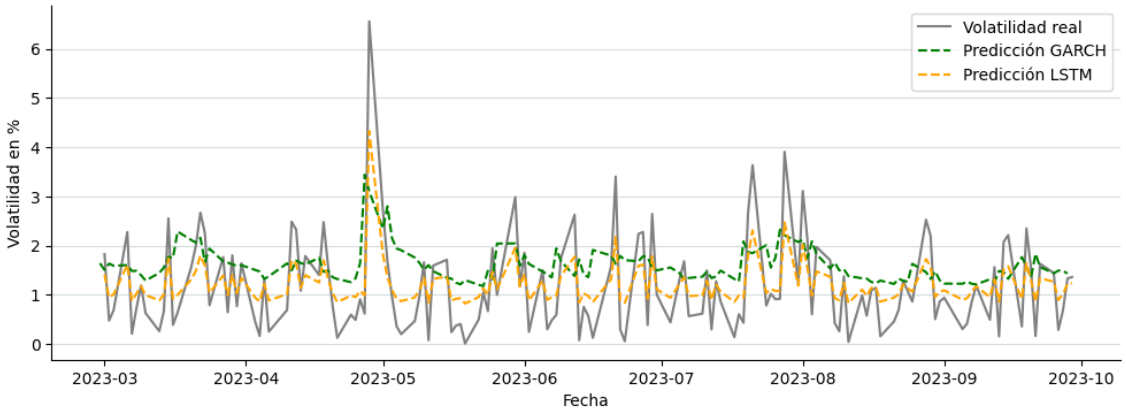


Figura A. 11. *Predicción de la volatilidad diaria de Microsoft II*

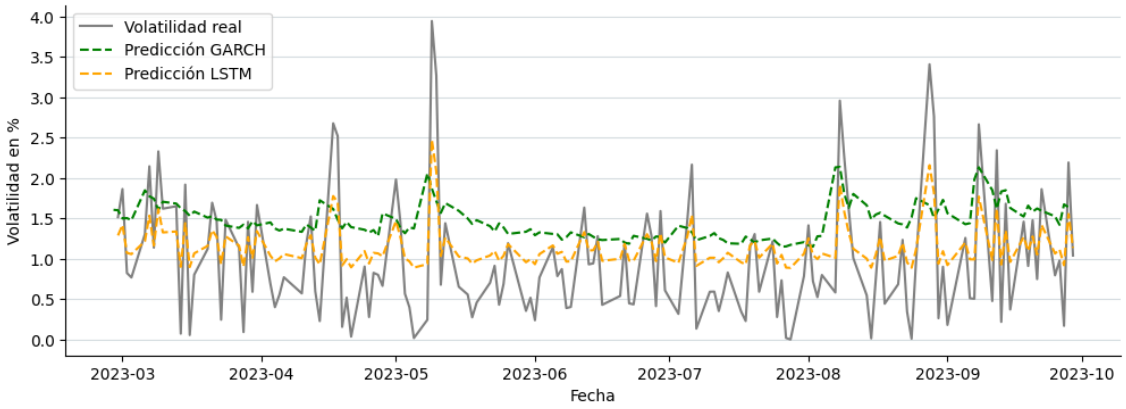


Figura A. 12. *Predicción de la volatilidad diaria de Apple II*

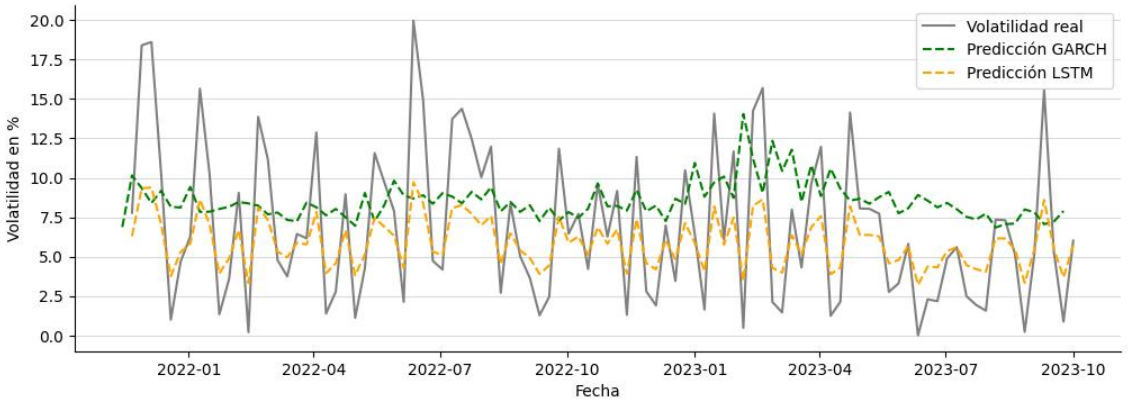


Figura A. 13. *Predicción de la volatilidad semanal de Tesla II*

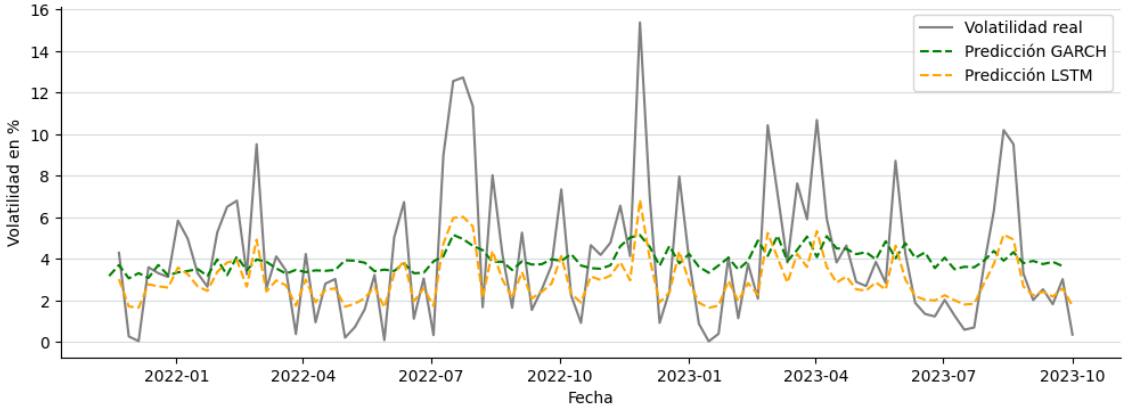


Figura A. 14. *Predicción de la volatilidad semanal de Google II*

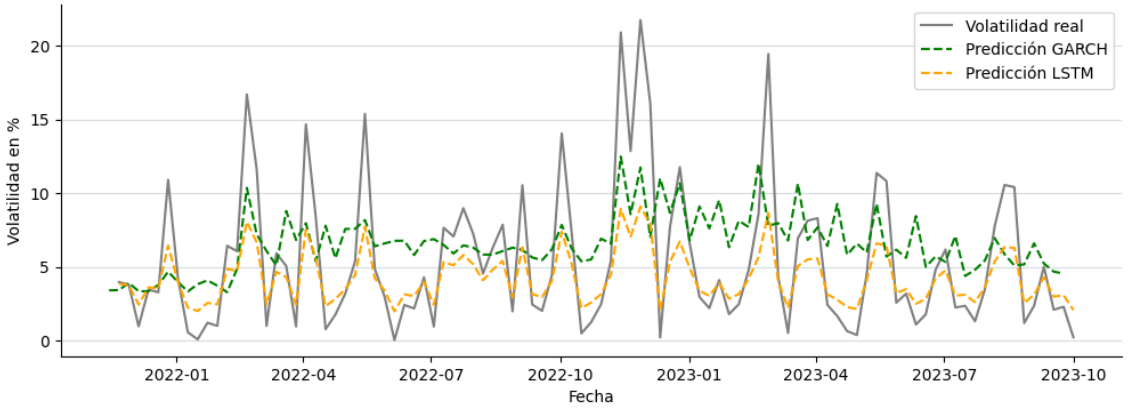


Figura A. 15. *Predicción de la volatilidad semanal de Meta II*

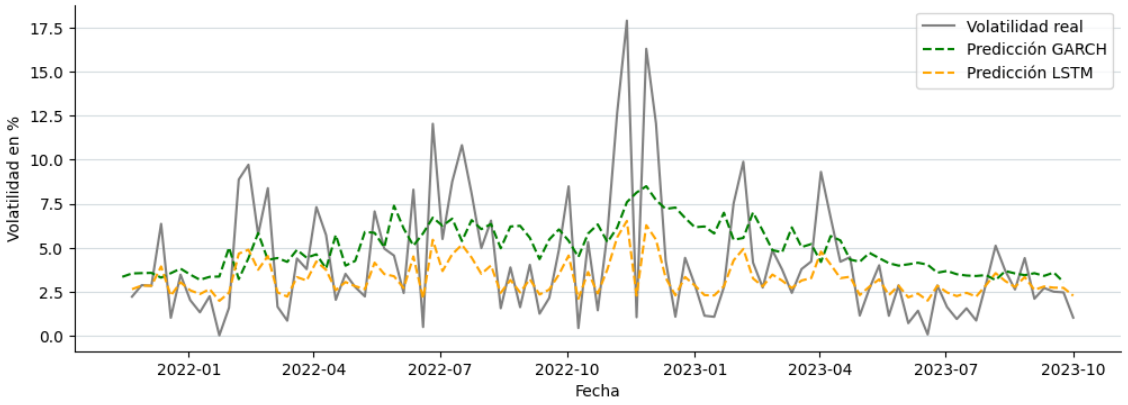


Figura A. 16. *Predicción de la volatilidad semanal de Amazon II*

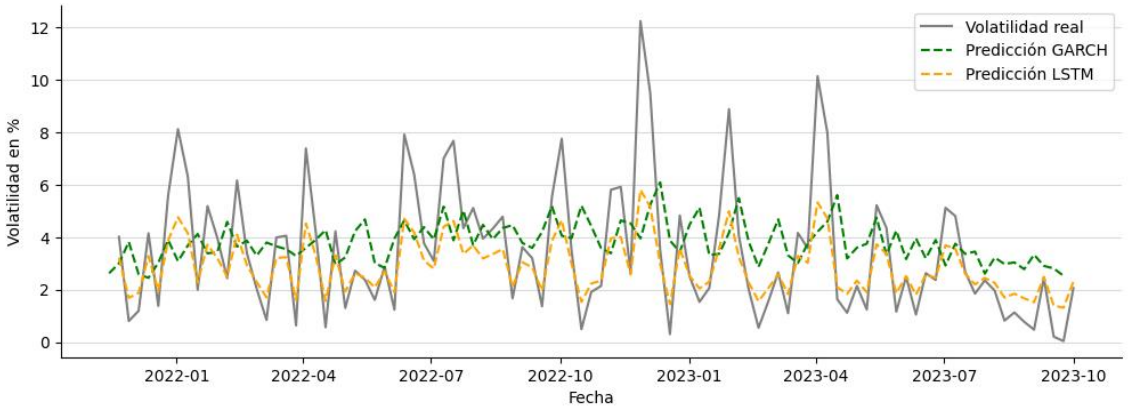


Figura A. 17. *Predicción de la volatilidad semanal de Microsoft II*

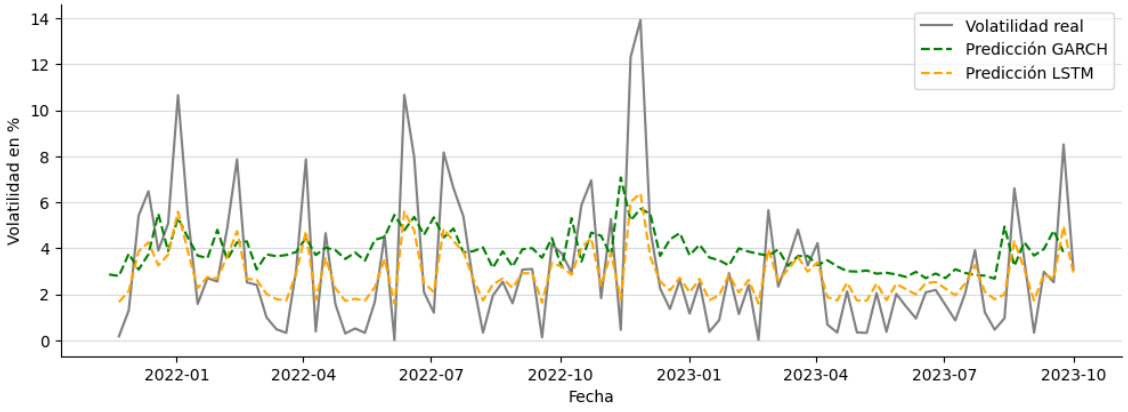


Figura A. 18. *Predicción de la volatilidad semanal de Apple II*

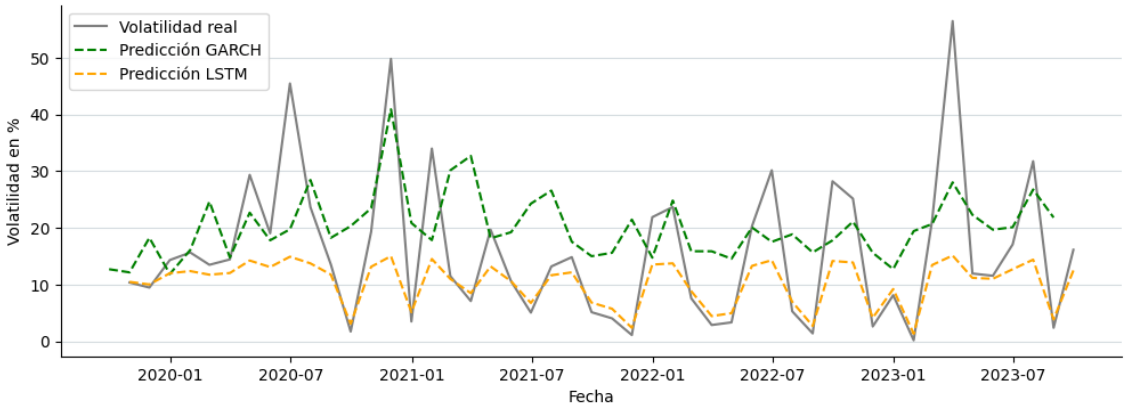


Figura A. 19. *Predicción de la volatilidad mensual de Tesla II*

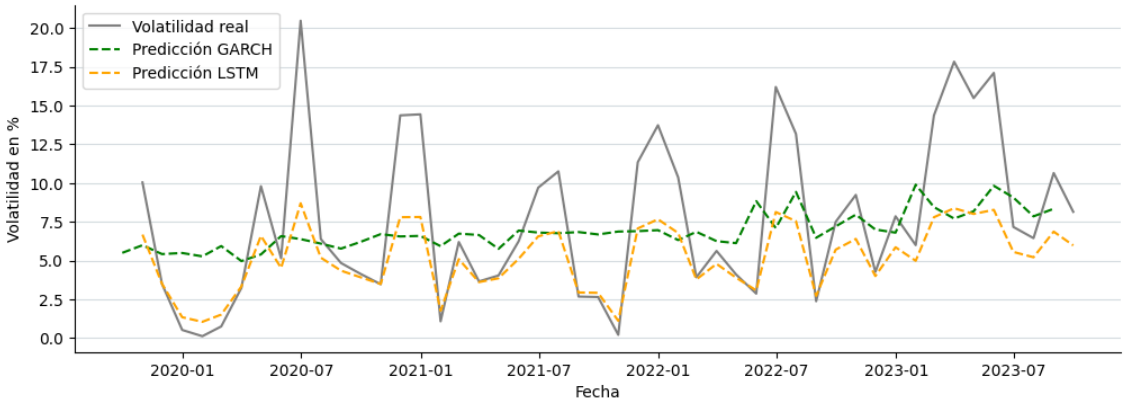


Figura A. 20. *Predicción de la volatilidad mensual de Google II*

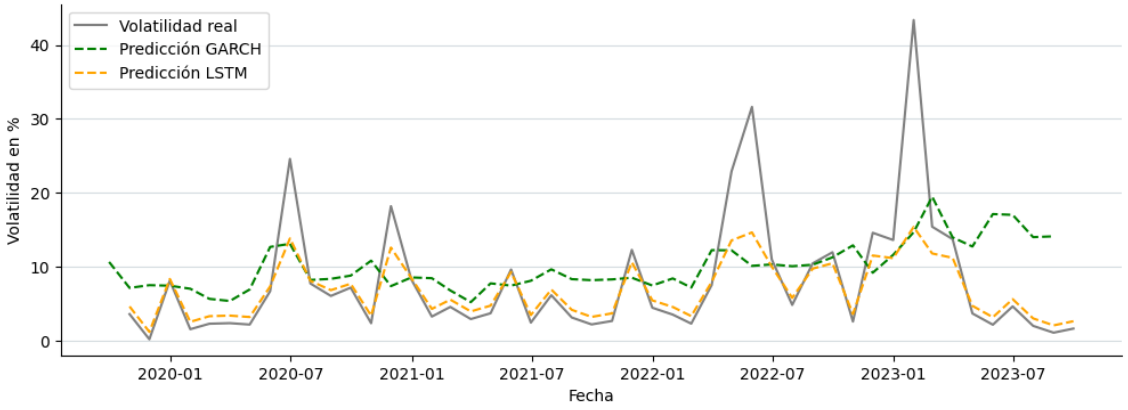


Figura A. 21. *Predicción de la volatilidad mensual de Meta II*

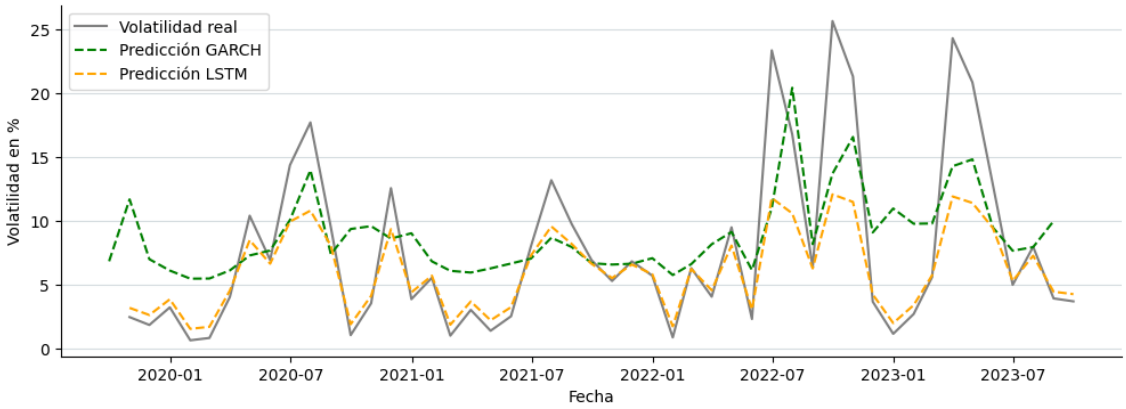


Figura A. 22. *Predicción de la volatilidad mensual de Amazon II*

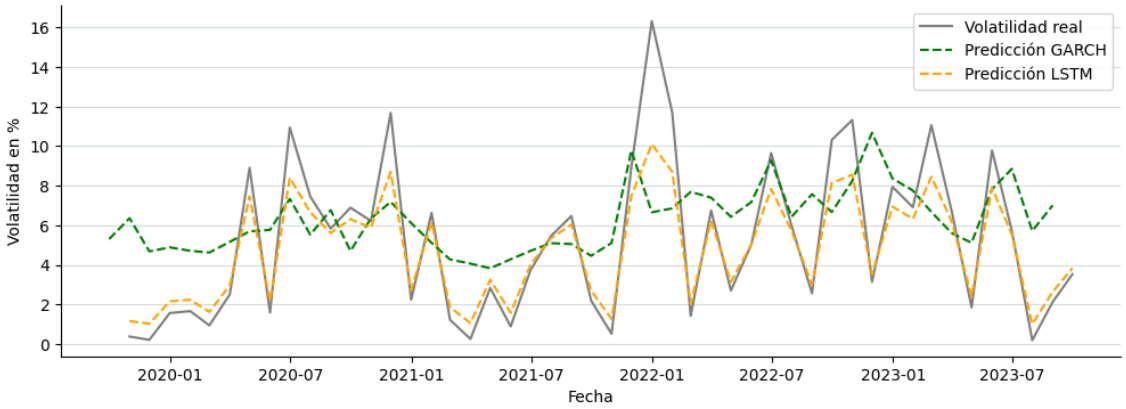


Figura A. 23. *Predicción de la volatilidad mensual de Microsoft II*

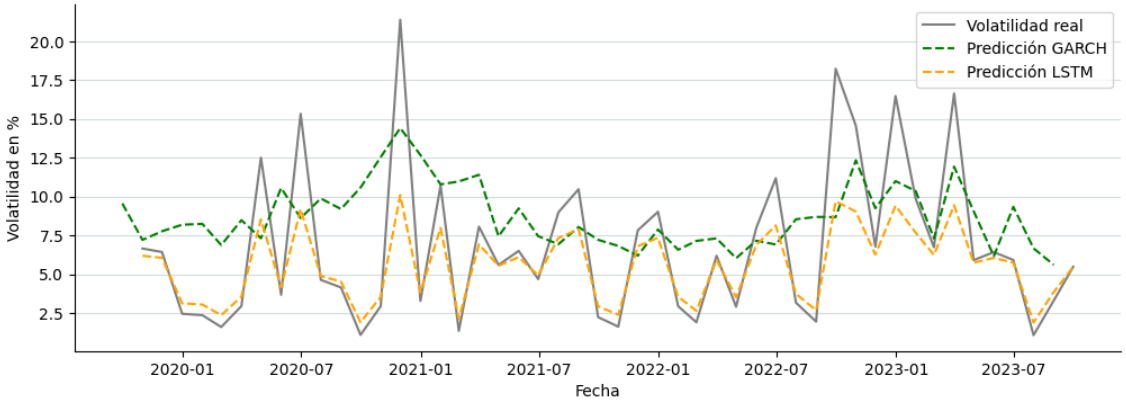


Figura A. 24. *Predicción de la volatilidad mensual de Apple II*

9.3 Anexo III

Este último Anexo está compuesto por los resultados y las gráficas obtenidas de la simulación del modelo EGARCH comentada en el apartado 5.2. El orden de aparición de las empresas y del tipo de volatilidad es el mismo seguido durante todo el trabajo.

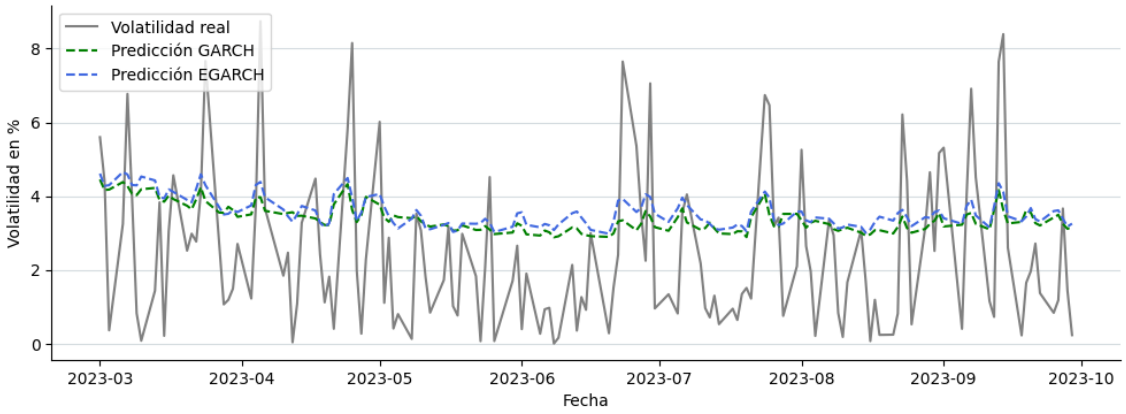


Figura A. 25. *Predicción de la volatilidad diaria de Tesla (GARCH vs EGARCH)*

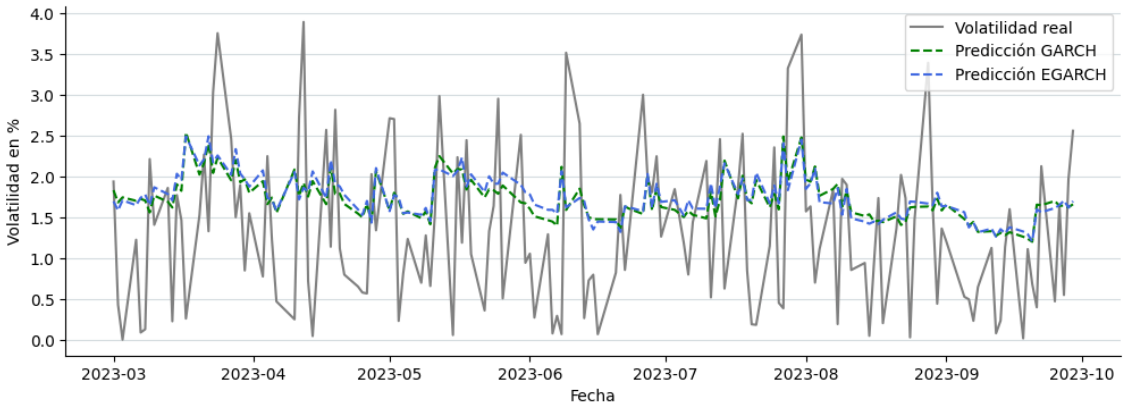


Figura A. 26. *Predicción de la volatilidad diaria de Google (GARCH vs EGARCH)*

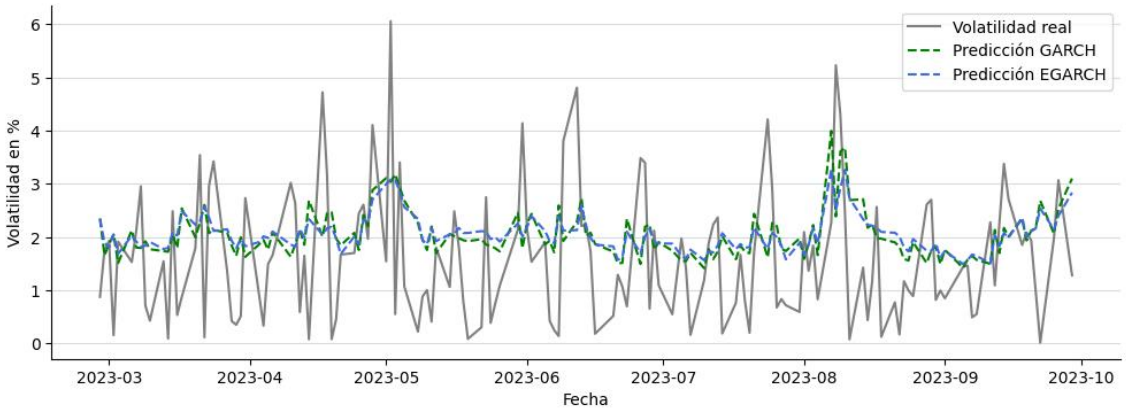


Figura A. 27. *Predicción de la volatilidad diaria de Amazon (GARCH vs EGARCH)*

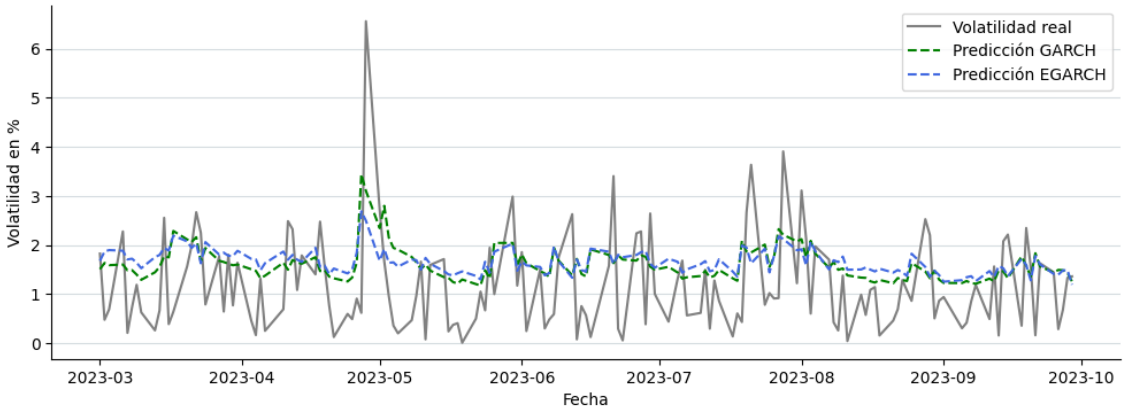


Figura A. 28. *Predicción de la volatilidad diaria de Microsoft (GARCH vs EGARCH)*

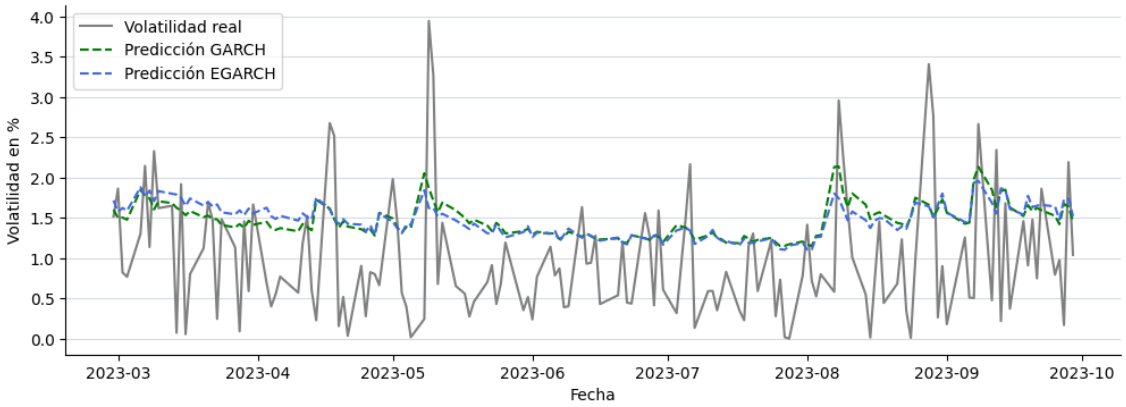


Figura A. 29. *Predicción de la volatilidad diaria de Apple (GARCH vs EGARCH)*

Tabla 12. *Métricas del error de predicción de la volatilidad semanal GARCH vs EGARCH*

	GARCH		EGARCH		Diferencia	
	RMSE	**MAPE**	**RMSE**	**MAPE**	**RMSE**	**MAPE**
Tesla	5,1822	0,5098	5,3258	0,5131	-2,77%	-0,65%
Google	2,9765	0,5664	3,0064	0,5677	-1,00%	-0,23%
Meta	4,4386	0,557	4,1383	0,5248	6,77%	5,78%
Amazon	3,2248	0,5107	3,2216	0,5184	0,10%	-1,51%
Microsoft	2,4144	0,5232	2,2644	0,4833	6,21%	7,63%
Apple	2,7248	0,5745	2,8758	0,5886	-5,54%	-2,45%

Fuente: *Elaboración propia a partir de los resultados obtenidos en Python*

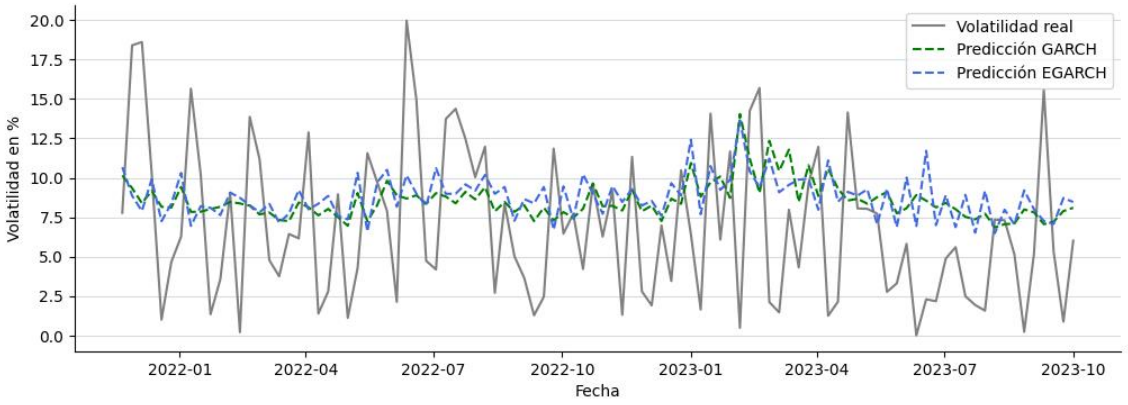


Figura A. 30. *Predicción de la volatilidad semanal de Tesla (GARCH vs EGARCH)*

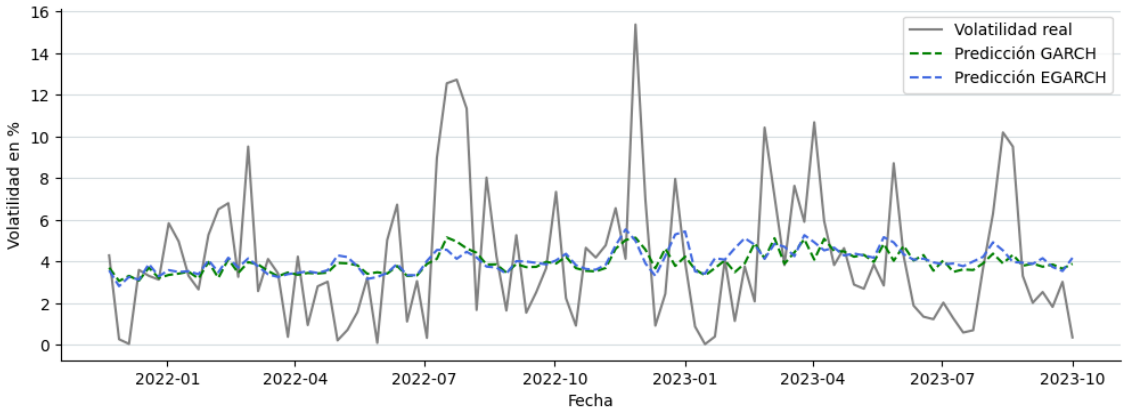


Figura A. 31. *Predicción de la volatilidad semanal de Google (GARCH vs EGARCH)*

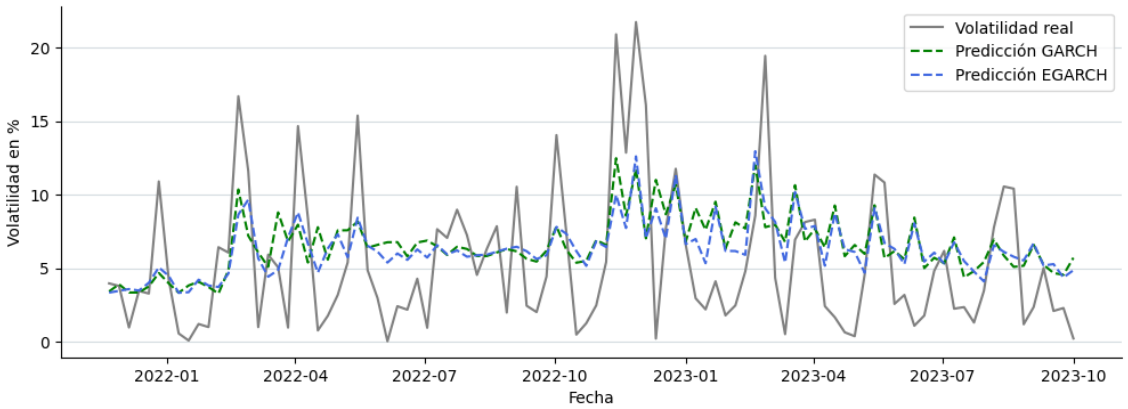


Figura A. 32. *Predicción de la volatilidad semanal de Meta (GARCH vs EGARCH)*

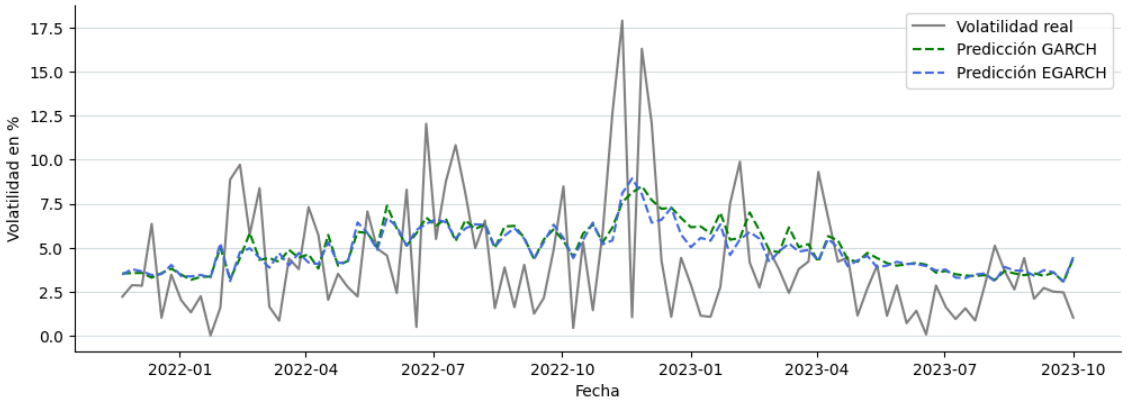


Figura A. 33. *Predicción de la volatilidad semanal de Amazon (GARCH vs EGARCH)*

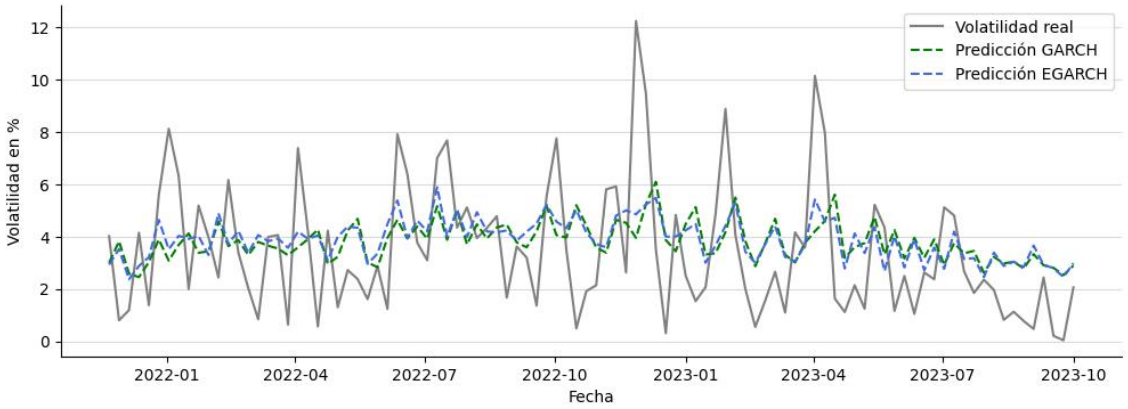


Figura A. 34. *Predicción de la volatilidad semanal de Microsoft (GARCH vs EGARCH)*

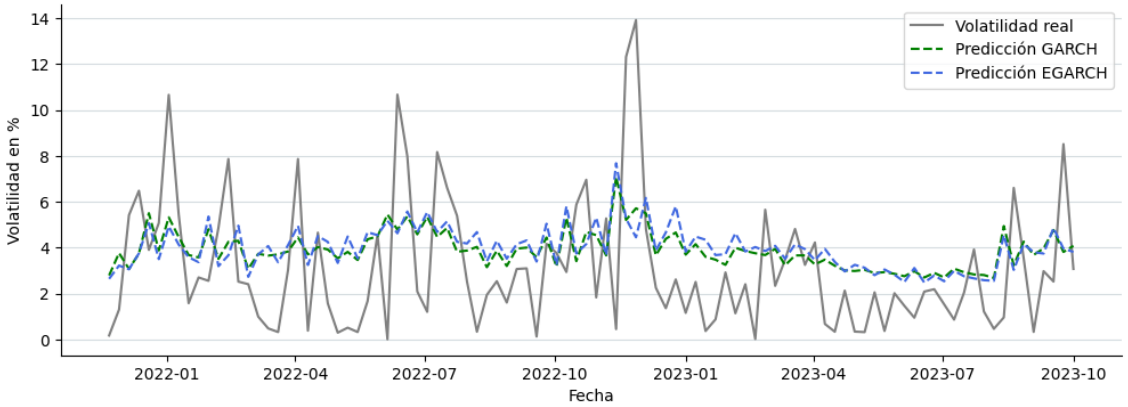


Figura A. 35. *Predicción de la volatilidad semanal de Apple (GARCH vs EGARCH)*

Tabla 13. *Métricas del error de predicción de la volatilidad mensual GARCH vs EGARCH*

	GARCH		EGARCH		Diferencia	
	RMSE	**MAPE**	**RMSE**	**MAPE**	**RMSE**	**MAPE**
Tesla	12,3478	0,4908	12,323	0,5165	0,20%	-5,24%
Google	4,8394	0,5709	5,2087	0,6771	-7,63%	-18,60%
Meta	8,3495	0,5907	9,0347	0,7753	-8,21%	-31,25%
Amazon	5,1158	0,4696	5,3779	0,5096	-5,12%	-8,52%
Microsoft	3,6506	0,5058	3,5383	0,5262	3,08%	-4,03%
Apple	4,9397	0,4732	4,8717	0,5166	1,38%	-9,17%

Fuente: *Elaboración propia a partir de los resultados obtenidos en Python*

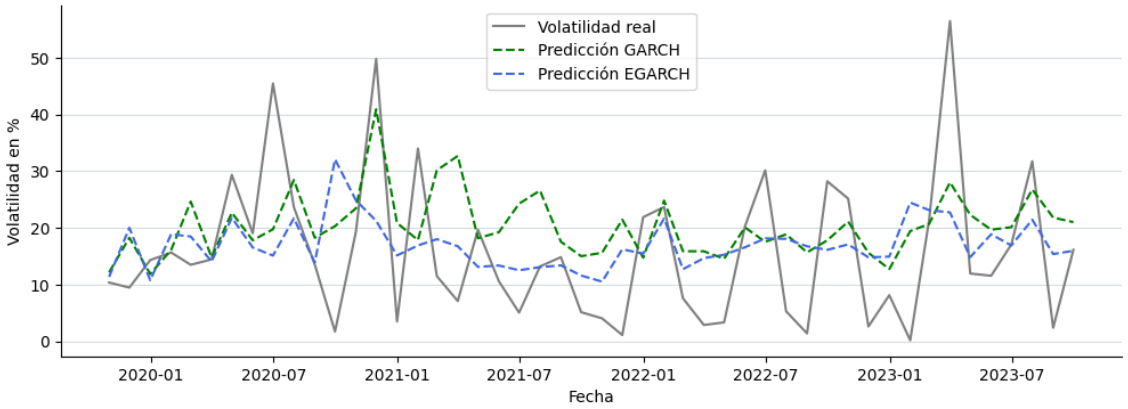


Figura A. 36. *Predicción de la volatilidad mensual de Tesla (GARCH vs EGARCH)*

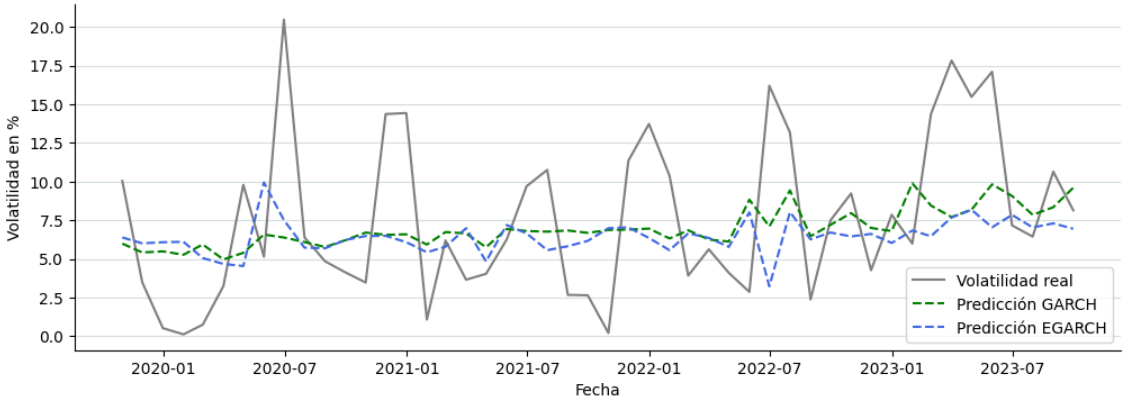


Figura A. 37. *Predicción de la volatilidad mensual de Google (GARCH vs EGARCH)*

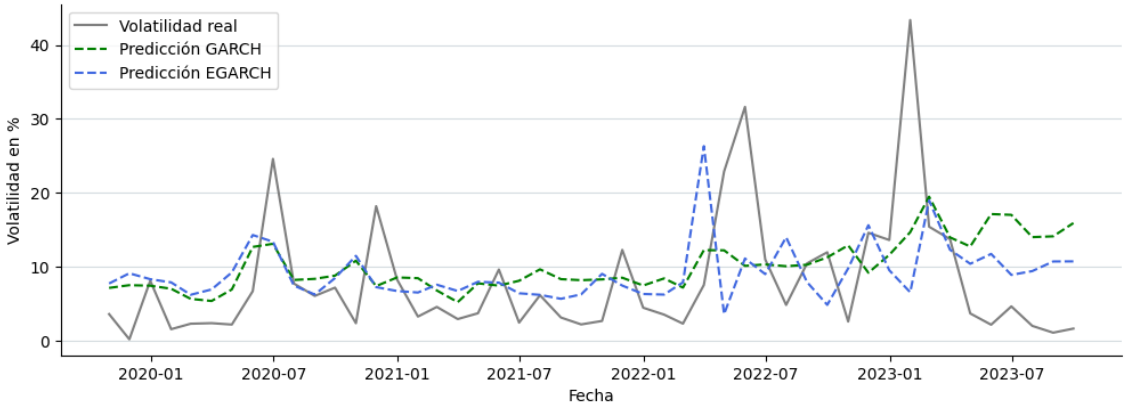


Figura A. 38. *Predicción de la volatilidad mensual de Meta (GARCH vs EGARCH)*

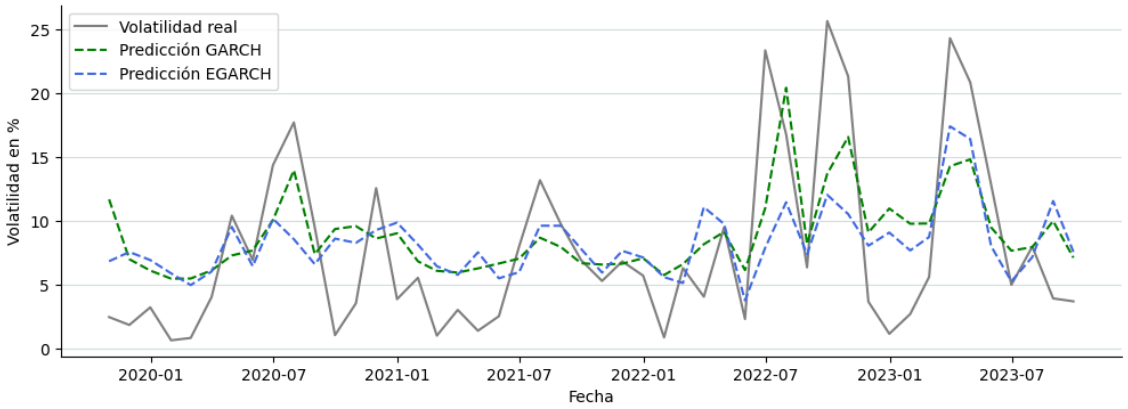


Figura A. 39. *Predicción de la volatilidad mensual de Amazon (GARCH vs EGARCH)*

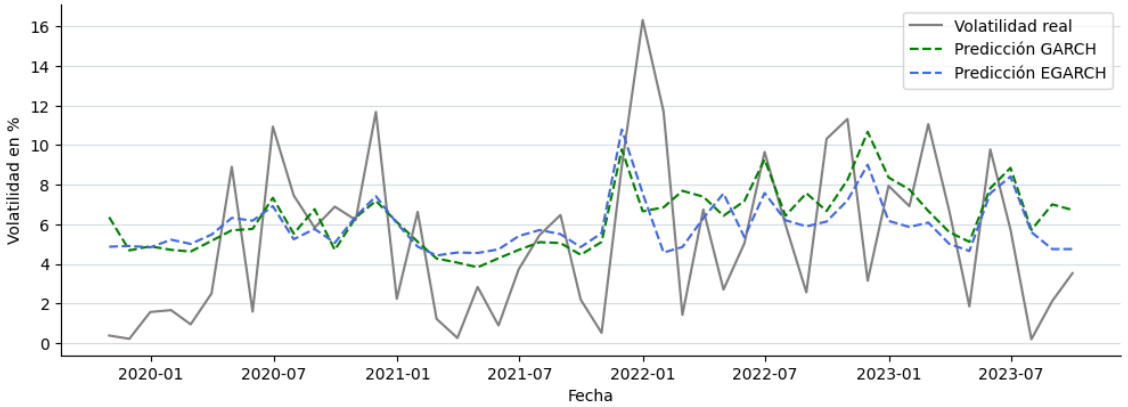


Figura A. 40. *Predicción de la volatilidad mensual de Microsoft (GARCH vs EGARCH)*

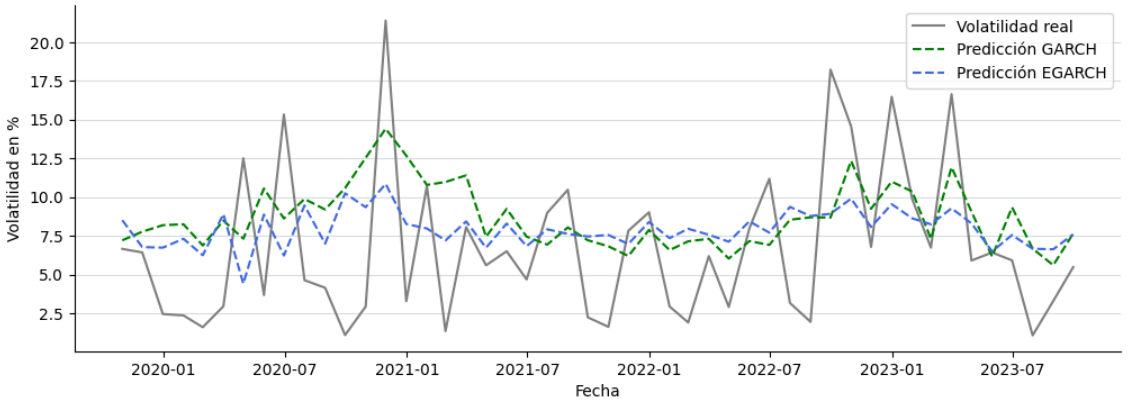


Figura A. 41. *Predicción de la volatilidad mensual de Apple (GARCH vs EGARCH)*