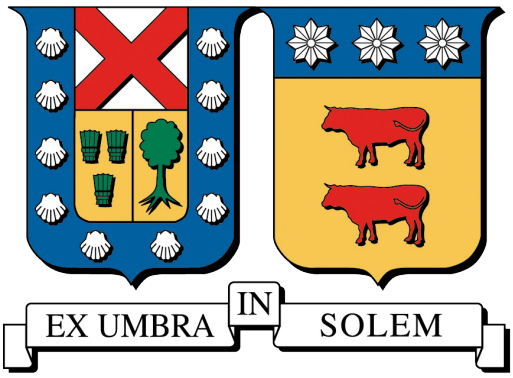
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INDUSTRIAS
VALPARAÍSO - CHILE



COMBINACIÓN DE UNA RED NEURONAL RECURRENTE LSTM Y UN MODELO GARCH PARA PREDECIR PRECIOS, VOLATILIDAD Y RENDIMIENTOS EN 100 ACCIONES DEL S&P 500

EDUARDO EZEQUIEL PEÑA CAMUS

TESIS PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INDUSTRIAL

PROFESOR GUÍA	:	WERNER KRISTJANPOLLER R.
PROFSOR CORREFERENTE	:	DIEGO CANESSA RICH

24 de junio de 2024

Dedicado a mi familia...

AGRADECIMIENTOS

Para mí, siempre fue un sueño formar parte de esta prestigiosa casa de estudios. Desde una temprana edad, veía este objetivo como un privilegio reservado solo para unos pocos, algo que parecía distante y fuera de mi alcance. Sin embargo, a través del constante esfuerzo, valentía y determinación, pude abrirme camino y finalmente ser Sansano de la Universidad Técnica Federico Santa María.

Es por esto por lo que quiero comenzar dando las gracias a las siguientes personas:

A Nadia, gracias por tu maravillosa compañía, por el cariño y por los buenos momentos, gracias por motivarme a seguir adelante para cumplir mis sueños. Tu apoyo y el de tus padres durante este proceso ha sido completamente invaluable para mí.

A mi hermano Matías, tu apoyo ha sido fundamental para alcanzar este objetivo. Gracias por ser mi compañero, amigo, confidente y por los buenos momentos junto a ti.

A mi madre Ada, gracias por confiar en mí y por enseñarme el valor del esfuerzo y perseverancia. Gracias por ser parte de mi vida y por siempre creer en mí, incluso en los momentos más difíciles.

A mi padre Eduardo, por su sabiduría y sus consejos, por enseñarme a enfrentar los desafíos con valentía y determinación. Agradezco tu constante motivación, por guiarme por el buen camino y estar siempre dispuesto a escucharme.

A todos ustedes, les dedico este trabajo con profundo agradecimiento y amor. Su apoyo y motivación han sido la fuerza que me ha impulsado a llegar hasta aquí.

También agradezco al resto de la familia y amigos que estuvieron presentes en el proceso y a mis profesores Diego Canessa Rich y Werner Kristjanpoller R., por su paciencia, empatía y dedicación. Su experiencia y profesionalismo me convencieron de elegirlos para este tema.

Con cariño,

Eduardo Ezequiel Peña Camus

[0]Ser Sansano significa pertenecer a una comunidad de excelencia académica y compromiso con el desarrollo de la sociedad. Es un símbolo de orgullo y responsabilidad que nos motiva a aplicar los conocimientos adquiridos para contribuir positivamente al progreso de nuestro país y del mundo.

RESUMEN EJECUTIVO

Sabemos que el mercado bursátil es un campo dinámico y complejo, donde los inversores buscan encontrar oportunidades para maximizar sus beneficios durante la operación de compra y venta de acciones. Sin embargo, tomar una decisión en este entorno siempre será desafiante debido a la gran cantidad de información disponible y la volatilidad que existe en el precio de los activos.

los estados financieros, el balance general, estado de resultado, estado de flujo de efectivo, entre otros informes e indicadores son de gran ayuda para poder analizar de manera fundamental una compañía. Esta información es importante porque se pueden diseñar indicadores claves para tener una referencia sobre el comportamiento de una industria y el valor de un sector determinado. Por otra parte sabemos que la industria financiera ha tenido un crecimiento importante en el uso de la inteligencia artificial que llamaremos de aquí en adelante como (IA)[1]. En base a lo anterior nos podemos hacer las siguientes preguntas: ¿La IA se puede considerar una herramienta clave en la toma de decisiones en el análisis cuantitativo de las acciones? ¿Es posible encontrar correlaciones con los modelos naturales de observación? Por esto veremos si estos modelos pueden ofrecer una vista alternativa para la toma de decisión de compra o venta de un valor bursátil en el mercado de capitales.

Estudios previos han demostrado que los modelos predictivos basados en IA pueden ofrecer beneficios bastante significativos para las instituciones o inversores que utilicen estas herramientas en el mercado bursátil. Por ejemplo, investigaciones académicas y experimentos en la industria financiera han revelado que los modelos de aprendizaje automático y las técnicas de IA pueden capturar patrones complejos en los datos históricos del mercado y generar predicciones precisas sobre los precios de los activos. A pesar de los avances en el uso de la IA en el mercado bursátil, aún existen desafíos y preguntas sin resolver que requieren una investigación más profunda. Por lo tanto, se intentará resolver preguntas sobre ¿Cómo se puede mejorar la toma de decisiones en el mercado bursátil mediante el uso de modelos predictivos basados en IA? ¿Las redes neuronales recurrentes son más precisas al tomar una decisión de inversión? ¿Un modelo GARCH[2] se puede vincular a un modelo RNN-LSTM[3]? ¿Cómo se pueden validar y evaluar estos modelos para garantizar su efectividad en diferentes escenarios del mercado? Esta investigación se centrará en analizar los resultados de RN-LSTM (Long – Short Term Memory Recurrent Network) determinando precio, volatilidad y rentabilidad del S&P 500 “Standard & Poor’s 500, y combinar los resultados con un modelo GARCH (Generalized Autoregressive Conditional Heteroskedasticity).

[1]Inteligencia artificial (IA) es un campo de la informática que se centra en crear sistemas capaces de realizar tareas que normalmente requieren inteligencia humana, como el reconocimiento de voz, la toma de decisiones y la traducción de idiomas

[2]El modelo GARCH (Generalized Autoregressive Conditional Heteroskedasticity) es utilizado en econometría y finanzas para modelar la volatilidad de series temporales, especialmente en mercados financieros.

[3]RNN-LSTM se refiere a una red neuronal recurrente (RNN) específicamente utilizando una unidad de memoria a largo plazo (LSTM, por sus siglas en inglés: Long Short-Term Memory). Es un tipo de arquitectura de red neuronal diseñada para capturar dependencias a largo plazo en datos secuenciales, como texto o series temporales.

ABSTRACT

We know that the stock market is a dynamic and complex field where investors seek to find opportunities to maximize their benefits during the buying and selling of stocks. However, making a decision in this environment will always be challenging due to the large amount of available information and the volatility that exists in asset prices.

Financial statements, balance sheets, income statements, cash flow statements, among other reports and indicators, are of great help in fundamentally analyzing a company. This information is important because key indicators can be designed to have a reference on the behavior of an industry and the value of a specific sector. On the other hand, we know that the financial industry has had significant growth in the use of artificial intelligence, which we will refer to henceforth as (AI)[4]. Based on the above, we can ask ourselves the following questions: Can AI be considered a key tool in decision-making in the quantitative analysis of stocks? Is it possible to find correlations with natural observation models? For this reason, we will see if these models can offer an alternative view for making a decision to buy or sell a stock in the capital market.

Previous studies have shown that predictive models based on AI can offer quite significant benefits for institutions or investors who use these tools in the stock market. For example, academic research and experiments in the financial industry have revealed that machine learning models and AI techniques can capture complex patterns in historical market data and generate accurate predictions about asset prices. Despite advances in the use of AI in the stock market, there are still unresolved challenges and questions that require deeper investigation. Therefore, we will attempt to answer questions such as: How can decision-making in the stock market be improved through the use of AI-based predictive models? Are recurrent neural networks more accurate when making an investment decision? Can a GARCH[5] model be linked to an RNN-LSTM[6] model? How can these models be validated and evaluated to ensure their effectiveness in different market scenarios? This research will focus on analyzing the results of RN-LSTM (Long – Short Term Memory Recurrent Network) determining price, volatility, and profitability of the S&P 500 "Standard & Poor's 500, and combining the results with a GARCH model (Generalized Autoregressive Conditional Heteroskedasticity).

[4]Artificial Intelligence (AI) is a field of computer science that focuses on creating systems capable of performing tasks that normally require human intelligence, such as speech recognition, decision-making, and language translation

[5]The GARCH (Generalized Autoregressive Conditional Heteroskedasticity) model is used to model and predict volatility in financial time series.

[6]RNN-LSTM refers to a recurrent neural network (RNN) specifically using a long short-term memory unit (LSTM). It is a type of neural network architecture designed to capture long-term dependencies in sequential data, such as text or time series.

RESULTADOS OBTENIDOS

Durante la investigación sobre los modelos RNN-LSTM y GARCH, se pudo determinar una serie de resultados interesantes y, a veces, desconcertantes. Al analizar el rendimiento del RNN-LSTM, pudimos ver una amplia gama de errores durante el análisis. pudimos ver que en el modelo RNN-LSTM la variable 180 mostro muy buen comportamiento en base a los hiperparámetros del programa, esta variable mostró un MAE de apenas 0.010288, mientras que se obtuvo un MAE de 0.388422 para la variable 55, esto se debe a que la cantidad de mercados que se estan analizando no todos convergen o tienen relación entre sí.

Esta variabilidad nos hace reflexionar sobre la naturaleza de los datos que están manejando. Es evidente que algunos patrones eran más fáciles de captar para el modelo, quizás porque tenían tendencias más claras o porque contaba con datos históricos de mejor calidad. Sin embargo, otras variables parecían ser un verdadero rompecabezas para el RNN-LSTM.

En general, el modelo RNN-LSTM entrego buenos resultados y tuvo un buen rendimiento. La mayoría de las variables mostraron un MAE por debajo de 0.2, lo cual es bastante decente. También nos deja tranquilos al ver que las diferentes métricas de error (MAE, MSE, RMSE) eran consistentes entre sí. Al menos el modelo está siendo coherente con las predicciones, aunque no siempre acertara.

Pero, como en toda investigación, siempre hay margen de mejora. Me di cuenta de que necesitaba encontrar una forma de manejar mejor los valores atípicos. Algunas variables tenían errores tan altos que me hicieron dudar si estaba pasando algo por alto. También pensé que quizás necesitaba más datos para algunas variables o tal vez un preprocesamiento más avanzado.

Cuando pasamos al modelo GARCH, obtuvimos impresiones similares. Para algunos símbolos, como BDX y CPB, los resultados obtenidos fueron muy buenos. Los valores bajos de AIC, BIC y Log-Likelihood indican que el modelo se ajusta muy bien a estos datos. Sin embargo, al ver los resultados para AMD y ADSK, pudimos observar que no tienen un buen desempeño. Los altos valores de RMSE, MSE y MAE nos indican que algo no estaba funcionando como espera.

Mientras se analizaban estos resultados, empecé a pensar en cómo podría combinar lo mejor de ambos modelos. ¿Y si se pudiera usar el Modelo GARCH para manejar la volatilidad y luego aplicar el Modelo RNN-LSTM a los datos ajustados? Esta propuesta suena bastante bien, pero sabemos que no es un proceso fácil de implementar.

Por estos motivos podemos determinar diferentes enfoques: quizás podría crear un modelo ensamblado, o tal vez se necesita entrar en una optimización de hiperparámetros más profunda. La idea de una validación cruzada más rigurosa también es una excelente opción.

Podemos ver que la combinación de modelos RNN-LSTM y GARCH no solo es factible, sino que nos entrega nuevas posibilidades para mejorar significativamente las capacidades predictivas de los modelos. La mezcla de las actuales tecnologías en aprendizaje profundo con métodos probados de modelado de volatilidad nos ha llevado a un terreno bastante interesante, donde la complejidad de los patrones secuenciales se enlaza con las sutilezas de la variabilidad de los datos.

El ajuste de los hiperparámetros no ha sido fácil, teniendo en cuenta que debemos analizar los efectos de memoria en nuestras secuencias y calibrando con precisión los parámetros del GARCH. Este proceso, aunque a veces frustrante, ha sido fundamental para asegurar que nuestro modelo capte fielmente la esencia de los datos que estamos analizando.

Por otra parte, uno de los aspectos más críticos que hemos descubierto es el preprocesamiento de datos, ya que cada técnica de normalización, cada decisión sobre cómo manejar valores atípicos, y cada estrategia para imputar datos faltantes ha tenido un impacto significativo en nuestros resultados. Este proceso de preparación de datos, aunque a menudo subestimado, es de mayor importancia para ambos modelos, esto mejora la precisión y también la capacidad para generalizar y mantener la estabilidad frente a nuevos datos.

Esta investigación nos ha llevado a un punto donde podemos afirmar que la integración de modelos RNN-LSTM y GARCH, respaldada por técnicas avanzadas de preprocesamiento y optimización, tiene el potencial de revolucionar la forma en que abordamos las predicciones de series temporales. No obstante, es fundamental reconocer que este enfoque no es una solución única. Requiere un diseño matemático e ingenieril meticuloso, una experimentación constante y una disposición para ajustar y reajustar cada componente del modelo.

RESULTS OBTAINED

During the research on RNN-LSTM and GARCH models, we were able to determine a series of interesting and sometimes perplexing results. When analyzing the performance of the RNN-LSTM, we could see a wide range of errors during the analysis. We observed that in the RNN-LSTM model, variable 180 showed very good behavior based on the hyperparameters of the program, with this variable showing a MAE of just 0.010288, while a MAE of 0.388422 was obtained for variable 55. This is due to the fact that not all of the markets being analyzed converge or have a relationship with each other.

This variability makes us reflect on the nature of the data being handled. It's evident that some patterns were easier for the model to capture, perhaps because they had clearer trends or because they had better quality historical data. However, other variables seemed to be a real puzzle for the RNN-LSTM.

Overall, the RNN-LSTM model delivered good results and performed well. Most variables showed a MAE below 0.2, which is quite decent. It's also reassuring to see that the different error metrics (MAE, MSE, RMSE) were consistent with each other. At least the model is being coherent with its predictions, even if it doesn't always get them right.

But, as in all research, there's always room for improvement. We realized that we needed to find a way to better handle outliers. Some variables had such high errors that it made us doubt if we were overlooking something. We also thought that perhaps we needed more data for some variables or maybe more advanced preprocessing.

When we moved to the GARCH model, we obtained similar impressions. For some symbols, like BDX and CPB, the results obtained were very good. The low values of AIC, BIC, and Log-Likelihood indicate that the model fits this data very well. However, when looking at the results for AMD and ADSK, we could observe that they don't perform well. The high values of RMSE, MSE, and MAE indicate that something wasn't working as expected.

While analyzing these results, we started thinking about how we could combine the best of both models. What if we could use the GARCH Model to handle volatility and then apply the RNN-LSTM Model to the adjusted data? This proposal sounds quite good, but we know it's not an easy process to implement.

For these reasons, we can determine different approaches: perhaps we could create an ensemble model, or maybe we need to enter into a deeper hyperparameter optimization. The idea of a more rigorous cross-validation is also an excellent option.

We can see that the combination of RNN-LSTM and GARCH models is not only feasible, but it

gives us new possibilities to significantly improve the predictive capabilities of the models. The mixture of current technologies in deep learning with proven volatility modeling methods has led us to a quite interesting terrain, where the complexity of sequential patterns intertwines with the subtleties of data variability.

The adjustment of hyperparameters has not been easy, considering that we must analyze the memory effects in our sequences and precisely calibrate the GARCH parameters. This process, although sometimes frustrating, has been fundamental to ensure that our model faithfully captures the essence of the data we are analyzing.

On the other hand, one of the most critical aspects we have discovered is data preprocessing, as each normalization technique, each decision on how to handle outliers, and each strategy to impute missing data has had a significant impact on our results. This data preparation process, although often underestimated, is of utmost importance for both models, as it improves accuracy and also the ability to generalize and maintain stability in the face of new data.

This research has led us to a point where we can affirm that the integration of RNN-LSTM and GARCH models, supported by advanced preprocessing and optimization techniques, has the potential to revolutionize the way we approach time series predictions. However, it is essential to recognize that this approach is not a one-size-fits-all solution. It requires meticulous mathematical and engineering design, constant experimentation, and a willingness to adjust and readjust each component of the model.

Índice de Contenidos

Índice de Tablas

Índice de Figuras

1 OBJETIVO GENERAL

Implementaremos y desarrollaremos un modelo matemático basado en IA utilizando redes neuronales recurrentes con LSTM, un modelo GARCH y MGARCH, estos serán programados en el entorno de desarrollo interactivo Live Code de Python, para predecir precios, volatilidad y rentabilidad de un conjunto de 100 acciones del S&P 500.

También una vez logrado el objetivo principal veremos si ambos modelos se vinculan en el análisis de manera efectiva, agregaremos recomendaciones de inversión precisas y oportunas a los inversionistas sobre el resultado de los modelos matemáticos a estudiar, con el objetivo de maximizar los resultados aprovechando la diversidad sectorial del S&P 500. Es por esto por lo que aplicaremos enfoques analíticos sofisticados, utilizando RN-LSTM para modelar relaciones temporales complejas y GARCH para analizar interacciones de volatilidad y rentabilidad entre las acciones del índice.

1.1 Objetivos Específicos

Modelaremos la volatilidad y rentabilidad aplicaremos una RNN-LSTM y un Modelo GARCH a un conjunto de 100 acciones que forman parte del índice S&P 500, este es reconocido como uno de los indicadores más prominentes y representativos del mercado de valores estadounidense.

Esta elección de una RNN-LSTM se justifica por su capacidad para capturar patrones temporales complejos inherentes a las series de tiempo, lo que resulta crucial en el contexto de la volatilidad y rentabilidad de instrumentos financieros. Esta red neuronal se configurará para comprender la dinámica temporal de las acciones, permitiendo una modelización precisa de sus comportamientos a lo largo del tiempo.

Adicionalmente, se implementará la metodología GARCH y posible vinculación de un Modelo DCC-MGARCH[7] para modelar la volatilidad condicional y la correlación dinámica en múltiples series financieras. Esta técnica de modelado de correlaciones dinámicas condicionales y covarianzas generalizadas autorregresivas proporciona una perspectiva integral sobre las interacciones complejas entre las distintas acciones que componen el índice.

[7]Dynamic Conditional Correlation Multivariate Generalized Autoregressive Conditional Heteroskedasticity (Heterocedasticidad Condicional Autorregresiva Generalizada Multivariada de Correlación Condicional Dinámica, en español). Este modelo es una extensión del modelo GARCH que aborda la correlación dinámica entre múltiples series de tiempo, como los retornos de varios activos financieros.

2 MARCO TEÓRICO

El presente estudio abordará los modelamientos y pronósticos de la volatilidad y rentabilidad de 100 acciones del S&P500, para conocer esto lo primero que debemos hacer es conocer cómo funciona matemáticamente el retorno financiero. Los retornos financieros suelen caracterizarse por su volatilidad y la presencia de eventos extremos que desafían los modelos tradicionales. Se centran en la comprensión y la modelización de la distribución de los retornos, especialmente en situaciones en las que los eventos extremos son más probables de lo que sugieren los modelos convencionales (McNeil et al., 2015) . Esto se plantea de la siguiente manera, sea r1, $\ldots\ldots,r_n$ una serie de retornos definida como la diferenciación logarítmica de una serie temporal S_t se refiere a calcular la tasa de cambio porcentual de los precios, índices o tipos de cambio en diferentes momentos. Esto se hace tomando el logaritmo natural de la relación entre dos valores consecutivos de la serie. La fórmula para la diferenciación logarítmica se expresa así:

$$r_t = \ln\left(\frac{S_t}{S_{t-1}}\right) \tag{1}$$

Donde:

r_t : Tasa de cambio porcentual logarítmica en el tiempo (Retorno logarítmico)

S_t : Valor en el tiempo t

S_{t-1} : Valor en el tiempo anterior a t

ln : Función logaritmo natural

Esta diferenciación logarítmica permite convertir los cambios absolutos en cambios relativos, lo que facilita la comparación de los movimientos en diferentes momentos y ayuda a estabilizar la varianza en series temporales financieras. Además, es útil en el análisis de series temporales financieras para evaluar la volatilidad y tendencias a lo largo del tiempo.

2.1 Ventajas en el Análisis de Series Temporales

1.Estabilización de la varianza: Cuando aplicamos la diferenciación logarítmica a la tasa de cambio porcentual, tendemos a estabilizar la varianza de la serie temporal. Esto hace que sea más fácil identificar patrones y tendencias.

2.Interpretación aditiva: Los retornos logarítmicos son aditivos con el tiempo. Esto significa que podemos sumar los cambios porcentuales de la evaluación para obtener el cambio total durante un período de tiempo determinado.

3.Reducción de la no estacionariedad: A menudo, la diferenciación logarítmica se utiliza para hacer que una serie temporal sea estacionaria. Esto simplifica el análisis y la modelización.

Entonces bajo el contexto financiero, podemos determinar que los retornos logarítmicos se utilizan ampliamente para modelar los precios de los activos, también medir riesgos y predecir movimientos futuros. Esta herramienta es fundamental en la teoría financiera y en la valorización de activos. En síntesis, la diferenciación logarítmica es solo una de las muchas técnicas disponibles en econometría y análisis de series temporales.

3 VOLATILIDAD EN MERCADOS FINANCIEROS

Según (Rossi, 2013) Simplificadamente, la volatilidad es un concepto que refiere a la inestabilidad o variabilidad de los precios. No implica necesariamente modificaciones en el nivel promedio, sino una mayor dispersión alrededor de ese promedio. De hecho, es posible que el nivel medio de precios experimente cambios sin que se modifique la volatilidad, como ocurriría toda vez que la oferta y la demanda sufran impulsos positivos o negativos que equilibren el mercado en un nuevo nivel. Contrariamente, podría con el tiempo verificarse una disminución o incremento en la variabilidad de los precios sin que haya cambios en su nivel medio.

Según (Streb, 2001)la teoría de la elección racional propone que los inversionistas toman decisiones racionales al evaluar los beneficios y costos de diferentes opciones de inversión. Además, se espera que consideren factores como el rendimiento esperado, el riesgo y las preferencias individuales al tomar decisiones financieras. Además, según (Mankiw & Espáriz, 2014) quien describe los principios de economía, menciona que el termino inversión es la actividad de usar ciertos bienes con el objetivo de obtener ingresos o rentas a lo largo del tiempo. Es por esto por lo que la inversión se refiere al acto de utilizar un capital en algún tipo de actividad económica o negocio, con el objetivo de ir incrementándolo con el tiempo.

La teoría de cartera de inversiones según (Conti et al., 2005) se enfoca en cómo los inversionistas pueden maximizar su rendimiento y minimizar el riesgo a través de la diversificación y la asignación adecuada de activos. Los modelos de cartera, como el Modelo de (Eismann, 2018) (Markowitz, 2011), ayudan a los inversionistas a encontrar la combinación óptima de activos que equilibre el riesgo y el rendimiento.

Según (Fama, 1970) menciona que la hipótesis de los mercados eficientes sugiere que los precios de los activos financieros reflejan toda la información disponible públicamente. Según esta teoría, los precios ya incorporan cualquier noticia relevante o cambio en las condiciones del mercado, esto dificulta la obtención de beneficios consistentes a través del análisis de información pasada o disponible públicamente.

Es crucial distinguir entre volatilidad histórica e implícita. La primera se deriva de los rendimientos pasados, mientras que la segunda se infiere de los precios de opciones financieras.

3.1 Factores Determinantes de la Volatilidad

1. Eventos Económicos y Políticos: Noticias o anuncios en los medios sobre economía, actuales decisiones gubernamentales sobre políticas monetarias de los diferentes países, elecciones sobre gobernantes, conflictos internacionales y otros sucesos pueden afectar y determinar la percepción de las inversiones y generar volatilidad o movimientos importantes en los precios de las acciones en los distintos mercados bursátiles.

2. Datos económicos: los indicadores económicos como el PIB, inflación, tasas de desempleo y otros pueden tener un gran impacto directamente en la volatilidad de los activos donde se comercialicen. Las reacciones de los mercados quedan determinadas por las noticias o datos económicos que generen una mayor o menor reacción.

3. Factores Geopolíticos: entre estos eventos se encuentran posibles tensiones comerciales, conflictos armados o cambios en las relaciones internacionales, pueden generar incertidumbre y aumentar la volatilidad en el mercado bursátil de un país o sector determinado.

4. Condición del mercado: La liquidez que tenga un mercado determinado, la velocidad de las transacciones y la participación de los inversores pueden influir en la volatilidad de un sector puntual. Mercados con baja liquidez tienden a ser más propensos a movimientos bruscos.

5. Sentimiento del mercado: Las percepciones y emociones de los inversores mayoritarios y minoritarios pueden tener un impacto significativo en la volatilidad de un sector en el que se invierte. La euforia, el miedo o la indecisión pueden llevar a movimientos demasiado volátiles en los precios.

6. Cambios en las tasas de interés: Las decisiones de los bancos centrales sobre las tasas de interés pueden afectar la volatilidad, ya que estos influyen en los costos de endeudamiento de un país y en las decisiones de inversión.

7. Condiciones climáticas y desastres naturales: La existencia de condiciones climáticas extremas pueden afectar la producción y la oferta de bienes y servicios de un sector determinado, esto tiene un impacto en la volatilidad de los mercados que estén relacionados.

8. Tecnología: cuando existen cambios tecnológicos en diferentes industrias, específicamente en el sector financiero, pueden afectar la velocidad y la forma en que se realizan las transacciones bursátiles, influyendo de manera activa en la volatilidad.

Los factores mencionados anteriormente pueden interactuar de manera compleja el sistema, debemos recordar que la volatilidad es inherente a los mercados financieros. Los inversores tienden a gestionar y aprovechar la volatilidad como parte de sus estrategias de inversión.

3.2 Gestión de Riesgos y Estrategia de Inversión

Se debe tener siempre presente que la volatilidad juega un papel crucial en la gestión de riesgos, hay que tener en cuenta que este riesgo y volatilidad también está presente en la elaboración de portafolios de inversión. Estrategias como el trading intradía y el uso de otras opciones para gestionar el riesgo ilustran su impacto en las decisiones de inversión. Entonces la gestión de riesgos vista desde el punto de las inversiones desempeña un papel bastante importante en la preservación del capital y la consecución de objetivos financieros a largo plazo. Este enfoque estratégico implica la identificación de los riesgos, posibles evaluaciones y mitigación de los riesgos asociados a las decisiones de inversión.

Una adecuada gestión y comprensión del riesgo es salvaguardar el capital invertido, este es el objetivo principal que se busca implementar, esta gestión se aplica tanto a persona, corporación, sociedad anónima, etc. El inversionista al comprender y cuantificar los riesgos inherentes a cada posición, pueden implementar medidas preventivas para mitigar y minimizar las perdidas potenciales de los clientes. Esta precaución se traduce en una mayor estabilidad financiera y en la capacidad de resistir periodos de volatilidad del mercado.

Se debe comprender que la gestión de riesgos no es más que la correcta diversificación de las inversiones en la cartera de inversión. Para lograr esto se debe realizar una meticulosa y correcta distribución de activos entre diferentes tipos de inversión y sectores reduce la exposición a riesgos específicos y contribuye a la estabilidad general de la cartera. Este enfoque estratégico también implica el establecimiento de límites de pérdida y la adopción de medidas para mantener la cartera alineada con los objetivos y tolerancias de riesgo del inversor.

La gestión de riesgos no solo se centra en la prevención de pérdidas, sino también en la mejora de la toma de decisiones. Un análisis exhaustivo de los riesgos asociados con cada inversión permite a los inversores tomar decisiones más informadas y conscientes de las posibles implicaciones. Este enfoque informado contribuye a la adaptabilidad del inversor ante cambios en el entorno económico, político y financiero. Se debe tener principal importancia que la gestión de riesgos se adapta al perfil de riesgo de cada inversor, cada individuo se comporta de manera distinta a las fluctuaciones y volatilidades que presenta el mercado, es importante aprender a controlar las emociones, esto se puede mejorar y gestionar con un correcto estudio de la psicología del trading.

Por otro lado, para tener éxito financiero se debe implementar una correcta estrategia de inversión y gestión de riesgo en el corto y mediano plazo. A continuación, se entregará una vista clave de una estrategia de inversión bien elaborada:

a) Objetivos Financieros: Primero se debe identificar claramente los objetivos del inversor, ya sean metas a corto, mediano y largo plazo. Al establecer metas específicas se proporciona una dirección clara para la toma de decisión del inversor.

b) Diversificación: Una estrategia bien diseñada incluye la diversificación de la cartera. Distribuir los activos entre diferentes clases y geografías ayuda a reducir el riesgo y mejora la estabilidad de la cartera de inversión.

c) Horizonte Temporal: Un aspecto crítico de la estrategia es el reconocimiento del horizonte temporal del inversor. Las necesidades y tolerancias al riesgo varían según el tiempo disponible para alcanzar los objetivos. Una estrategia bien diseñada tiene en cuenta este factor, permitiendo ajustes graduales a medida que cambian las circunstancias y evolucionan las metas.

d) Revisión y Ajuste Continuo: Las estrategias de inversión no son estáticas, estas deben revisarse y ajustarse periódicamente para adaptarse a cambios en los objetivos, condiciones del mercado y factores económicos.

e) Maximización del Rendimiento: La estrategia busca maximizar el rendimiento dentro de los límites de riesgo aceptables. A través de decisiones informadas, los inversores pueden perseguir rendimientos adecuados sin comprender sobre gestión de riesgos.

A pesar de su relevancia, la volatilidad presenta importantes desafíos, como la dificultad para prever con precisión los movimientos del mercado. La gestión de riesgos y la estrategia de inversión son esenciales para equilibrar el rendimiento y la protección del capital. Un portafolio eficiente, según Markowitz, es aquel que tiene un mínimo riesgo para un retorno dado (Markowitz, 1991), equivalentemente un portafolio con un máximo retorno para un nivel de riesgo dado permitiendo a los inversores navegar de manera más efectiva en los mercados financieros.

4 MODELO GARCH

El modelo GARCH, es un modelo econométrico ampliamente utilizado para modelar y pronosticar la volatilidad en series temporales financieras, el cual nos indica que la volatilidad no es constante en el tiempo y puede ir variando, este modelo entrega información sobre patrones de persistencia y cambios. Entre los modelos más conocidos se encuentran: MGARCH o GARCH multivariado propuesto por Bollerslev, Engle y Wooldridge en 1988 y ha sido ampliamente adoptado en la modelización de la volatilidad en los mercados financieros (Bollerslev et al., 1988).

El modelo GARCH extiende el modelo ARCH al incorporar términos autorregresivos en la varianza condicional, la ecuación general se define por:

4.1 Nivel de volatilidad condicional en el tiempo

t (σ_t^2) se define de la siguiente manera:

$$h_t = \alpha_0 + \sum_{i=1}^{p} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j h_{t-j} \tag{2}$$

Donde:

- h_t es la varianza condicional en el tiempo t.
- ϵ_{t-i} son los errores del pasado al cuadrado en el tiempo $t-i$.
- h_{t-j} son las estimaciones pasadas de la varianza condicional.
- α es el parámetro que mide el impacto de los errores pasados al cuadrado en la volatilidad actual.
- β es el parámetro que mide el impacto de la varianza condicional pasada en la volatilidad actual.

Esta ecuación describe cómo la varianza condicional en el tiempo t (σ_t^2) se forma a partir de una combinación de un término constante (α_0), el impacto de los errores al cuadrado pasados ($\alpha\epsilon_{t-1}^2$), y el impacto de la varianza condicional pasada ($\beta\sigma_{t-1}^2$).

El modelo GARCH utiliza los errores al cuadrado pasados y la varianza condicional pasada para modelar cómo cambia la volatilidad a lo largo del tiempo. Al estimar los parámetros α_0, α y β del modelo, se pueden realizar predicciones sobre la volatilidad futura en los mercados financieros. Debemos recordar que el modelo GARCH se estima con el método de máxima verosimilitud (MLE, por sus siglas en inglés). Esta estimación permite encontrar los valores de los parámetros del modelo que maximizan la posibilidad de observar la serie temporal dada. Supongamos que r_t son los retornos observados en el tiempo t y σ_t^2 es

la varianza condicional correspondiente. La función de verosimilitud para un conjunto de observaciones $r_1, r_2, \ldots, r_T$ es:

$$L(\theta) = \sum_{t=1}^{T} \ln\left(f(r_t \mid F_{t-1})\right) \tag{3}$$

En el Modelo GARCH vemos que la función de densidad condicional de los retornos financieros depende de la normalidad de los errores que se presenten y se expresa en términos de la varianza condicional σ_t^2. Utilizando métodos de optimización específicos, como el método de Newton-Raphson o el algoritmo BFGS, buscamos maximizar el logaritmo de la función de verosimilitud con respecto a los parámetros.

$$\theta = (\omega, \alpha, \beta) \tag{3.1}$$

$$\hat{\theta} = \arg\max \ln L(\theta) \tag{3.2}$$

Este proceso se puede realizar numéricamente con software especializado.

Las aplicaciones del modelo GARCH son fundamentales para modelar la volatilidad en series temporales financieras, abarcando áreas como los rendimientos de acciones, tasas de cambio y precios de commodities. Este modelo no solo provee herramientas analíticas importantes para realizar un análisis, sino que también brinda a analistas y gestores de carteras la capacidad de entender la naturaleza dinámica de la volatilidad. Esta comprensión resulta esencial para tomar decisiones de inversión informadas y gestionar de manera efectiva los riesgos asociados. En consecuencia, el modelo GARCH se presenta como un aliado valioso en el complejo mundo financiero, permitiendo a profesionales utilizar esta herramienta para adaptarse a las fluctuaciones del mercado con mayor perspicacia y confianza.

5 MODELO MGARCH MULTIVARIADO

El modelo GARCH Multivariado (MGARCH) es una extensión del modelo GARCH univariado que permite analizar la volatilidad conjunta entre múltiples activos o series temporales financieras. La aplicación más utilizada de los modelos MGARCH es el estudio y comprensión de las relaciones entre las volatilidades y co-volatilidades de varios mercados financieros (Kearney & Patton, 2000).

A diferencia del modelo GARCH univariado, que se enfoca en la volatilidad de un solo activo en un mercado determinado, el MGARCH es especialmente útil cuando se trabaja con carteras que contienen varios instrumentos financieros. ya que permite capturar la dinámica de la varianza condicional y la covarianza entre las series de tiempo. La idea principal detrás de los modelos GARCH es capturar la heterocedasticidad condicional, es decir, la variabilidad en la volatilidad a lo largo del tiempo.

La aplicación del MGARCH se da principalmente en la gestión de carteras, ya que permite a los gestores entender como la volatilidad de diferentes activos interactúan entre sí. Facilitando la construcción de carteras diversificadas que consideran las relaciones de volatilidad entre los activos.

Este modelo extiende el enfoque del modelo GARCH a situaciones donde se maneja múltiples activos o series temporales. La ecuación del modelo MGARCH utilizando las especificaciones VECH y DCC es la siguiente:

5.1 Modelo VECH

El modelo VECH (Vectorized Conditional Heteroskedasticity) tiene un enfoque econométrico utilizado para modelar la varianza y covarianza condicionales en series temporales multivariadas financieras. Es una extensión del modelo ARCH (Autoregressive Conditional Heteroskedasticity) para múltiples series, y se aplica en situaciones donde es importante capturar información sobre la dinámica de la covarianza entre varias series financieras, como los retornos de diferentes activos.

5.2 Ecuación de la Media

La ecuación de la media en el modelo VECH es simplemente:

$$r_t = u_t + \epsilon_t \tag{4}$$

Donde r_t es el vector de rendimiento en el tiempo t, u_t es el vector de medias condicionales, y ϵ_t es el vector de residuos o innovaciones.

5.3 Ecuación de la Varianza (Especificación VECH)

Aquí, H_t es la matriz de covarianza condicional en el tiempo t. El operador vech$(*)$ transforma una matriz simétrica en un vector apilando sus columnas.

$$\text{vech}(H_t) = C + A \cdot \text{vech}(\epsilon_{t-1}\epsilon'_{t-1}) + B \cdot \text{vech}(H_{t-1}) \tag{5}$$

Donde:

- r_t es el vector de rendimiento en el tiempo t
- u_t es el vector de medias condicionales en el tiempo t
- ϵ_t es el vector de residuos o innovaciones en el tiempo t
- H_t es la matriz de covarianza condicional en el tiempo t
- C, A y B son matrices de parámetros
- vech$(*)$ es el operador de apilamiento de columnas de una matriz simétrica

5.4 Operador VECH

El operador vech es fundamental en este modelo. Para una matriz simétrica S, vech(S) es un vector que contiene los elementos de S apilados por columnas, pero solo considera los elementos únicos debido a la simetría. Por ejemplo, para una matriz S de 3×3:

$$S = \begin{pmatrix} s_{11} & s_{12} & s_{13} \\ s_{12} & s_{22} & s_{23} \\ s_{13} & s_{23} & s_{33} \end{pmatrix}$$

El vector vech(S) sería:

$$\text{vech}(S) = \begin{pmatrix} s_{11} \\ s_{12} \\ s_{22} \\ s_{13} \\ s_{23} \\ s_{33} \end{pmatrix}$$

5.5 Limitaciones

A pesar de la forma en como es utilizado, el modelo VECH es computacionalmente intensivo debido al número creciente de parámetros que se tienen que estimar a medida que aumentaba la dimensión del vector de series temporales. Esto había hecho que el modelo fuera difícil de estimar y aplicar en la práctica para sistemas de alta dimensión.

En resumen, el modelo VECH ha sido una herramienta fundamental para el análisis de series temporales multivariadas, proporcionando un marco para capturar las dinámicas de la varianza y la covarianza de manera flexible y detallada.

6 MODELO DCC

El modelo DCC (Dynamic Conditional Correlation) es una herramienta estadística avanzada que es utilizada para analizar y modelar la evolución temporal de las correlaciones entre múltiples series temporales. Este modelo es desarrollado y utilizado para superar las limitaciones de los modelos de varianza condicional constantes en el tiempo, el modelo DCC permite que las correlaciones condicionales cambien dinámicamente durante el proceso, lo cual es crucial en muchas aplicaciones financieras y económicas actuales.

De acuerdo con (Engle, 2009) y (Aielli, 2013) la ecuación de la matriz de covarianza condicional que describe a este proceso está dada por :

6.1 Ecuación de la matriz de covarianza condicional

$$H_t = D_t R_t D_t \tag{6}$$

6.2 Ecuación de la matriz de correlaciones condicionales (DCC)

$$Q_t = (1 - \alpha - \beta)\bar{Q} + \alpha\psi_{t-1}\psi'_{t-1} + \beta(\epsilon_{t-1}\epsilon'_{t-1}) + BQ_{t-1} \tag{7}$$

$$R_t = (Q_t \circ I)^{-\frac{1}{2}} Q_t (Q_t \circ I)^{-\frac{1}{2}} \tag{8}$$

Donde:

- D_t es una matriz diagonal con las desviaciones estándar condicionales en la diagonal principal.
- R_t es la matriz de correlaciones condicionales en el tiempo t.
- Q_t es la matriz de cuasi-correlaciones en el tiempo t.

- $\bar{Q}$ es la matriz de correlaciones incondicionales.
- $\psi_t = D_t^{-1}\epsilon_t$ es el vector de residuos estandarizados.
- α y β son parámetros del modelo DCC.
- $\circ$ representa el producto de Hadamard (elemento por elemento).
- I es una matriz identidad.

El modelo MGARCH, mediante fórmulas como la especificación VECH y el modelo DCC, proporciona una herramienta bastante útil para capturar la dinámica de la volatilidad y la correlación en múltiples series de tiempo financieras. Estas fórmulas permiten modelar la evolución de la matriz de covarianza condicional, reflejando así la variabilidad y las interacciones entre diferentes activos. Este enfoque avanzado es esencial para la gestión de riesgos y la toma de decisiones en mercados financieros de alta complejidad, ya que permite una mejor comprensión y predicción de las fluctuaciones de precios y la diversificación de carteras.

7 REDES NEURONALES ARTIFICIALES

Las Redes Neuronales Artificiales (RNA)[8] han emergido con el pasar del tiempo como una herramienta poderosa en el campo de la IA y el aprendizaje automático. Estas estructuras computacionales están inspiradas por sus desarrolladores en la organización y funcionamiento del sistema nervioso del ser humano, estas son un subconjunto de machine learning y constituyen el eje de los algoritmos de Deep learning. Su nombre y estructura se inspiran en el cerebro humano, e imitan la forma en la que las neuronas biológicas se señalan entre sí.

Su aplicación está inserta en distintas disciplinas científicas, los encargados utilizan este modelo para determinar y resolver diferentes problemas matemáticos. Esta red esta compuestas por neuronas artificiales interconectadas que forman capas, estas capas trabajan juntas y son las que están encargadas de procesar información y realizar tareas específicas. la siguiente imagen "Figura 1" lo explica de forma más gráfica.

Desde la perspectiva y mirada de la ciencia computacional, las RNA se destacan como modelos matemáticos altamente sofisticados que de cierta manera intentan imitan la complejidad y la adaptabilidad del cerebro humano. Estas redes están compuestas por unidades fundamentales llamadas neuronas artificiales, Estas se encuentran conectadas entre sí mediante ponderaciones ajustables que permiten a la red aprender y adaptarse a patrones complejos en los datos y realizar tareas sofisticadas de procesamiento de información. Aunque han demostrado un gran éxito en muchas áreas, las RNA también enfrentan

[8]Las Redes Neuronales Artificiales (RNA) son modelos computacionales inspirados en la estructura y funcionamiento del cerebro humano, utilizadas para el aprendizaje automático y el procesamiento de datos complejos, como el reconocimiento de patrones y la predicción.

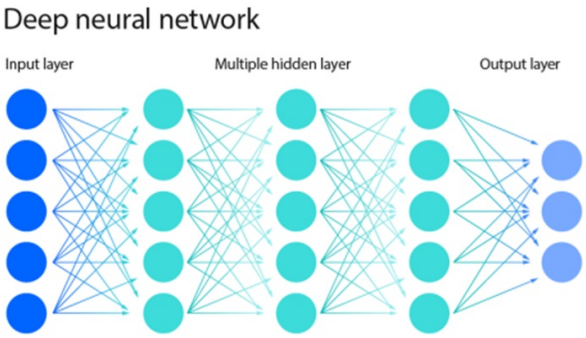


Figure 1: Deep Neural Network

desafíos como la interpretabilidad de los resultados, la necesidad de disponer de un conjunto grande de datos para el entrenamiento y la tendencia al sobreajuste en modelos demasiado complejos.

En conjunto, las RNA representan una herramienta poderosa en el campo de la inteligencia artificial, con un gran potencial para revolucionar diversas industrias y disciplinas científicas.

7.1 Fundamentos de las RNA

Las RNA están compuestas por unidades básicas llamadas neuronas, organizadas en capas. Cada conexión entre neuronas está asociada con un peso que se ajusta durante el proceso de entrenamiento. El aprendizaje se logra mediante la propagación hacia adelante y hacia atrás de la información a través de la red, ajustando los pesos para minimizar la diferencia entre las predicciones del modelo y los valores reales.

1. Neuronas y conexiones: • Las neuronas en una RNA están de una manera inspiradas en las células nerviosas biológicas. Cada una de las neuronas realiza operaciones en su entrada y transmite la salida a través de conexiones ponderadas a otras neuronas. (McCulloch & Pitts, 1943).

2. Aprendizaje y ponderación: • En un RNA el aprendizaje implica ajustar las ponderaciones de las conexiones entre neuronas para minimizar la diferencia entre las predicciones del modelo y los datos reales durante el entrenamiento. (Rumelhart et al., 1986).

Las Redes Neuronales Artificiales (RNA) se fundamentan en la emulación de procesos cerebrales para realizar tareas de procesamiento de información. Su base radica en el concepto de neuronas y conexiones ponderadas, donde las neuronas realizan operaciones en entradas y transmiten la información a través de conexiones ajustables. El aprendizaje en las RNA implica la adaptación de estas ponderaciones para optimizar la capacidad del modelo de hacer predicciones precisas.

7.2 Arquitectura de una RNA

La arquitectura de esta RNA consiste en una capa de entrada que recibe las secuencias temporales, luego está compuesta por una capa de unidades LSTM (Long Short-Term Memory), conocidas por su capacidad para aprender dependencias temporales a largo plazo. La capa de salida produce la predicción del valor futuro de la acción en cuestión.

7.3 Redes Neuronales Feedforward (FNN)

- Estructura: Capas de entrada, ocultas y de salida sin ciclos ni retroalimentación.
- Aplicación: Ampliamente utilizadas en problemas de clasificación y regresión.

Estas son ampliamente utilizadas para problemas de clasificación y regresión, y su estructura modular facilita el análisis matemático y teórico. Este modelo radica en la ausencia de ciclos y retroalimentación, permitiendo así un flujo unidireccional de la información a través de la red. Esta configuración modular ha demostrado su eficacia en la resolución de problemas complejos de clasificación y regresión, donde la simplicidad y eficiencia de las FNN se manifiestan de manera notable. El reconocido trabajo de Hornik, Stinchcombe y White respalda esta afirmación al demostrar que las redes feedforward multicapa poseen una propiedad única, es decir, actúan como aproximadores universales, capaces de representar cualquier función de manera arbitraria. Este hallazgo subraya la versatilidad inherente de las FNN y solidifica su aplicación generalizada en diversos campos de investigación, consolidándose como una herramienta esencial en el arsenal de las redes neuronales (Hornik et al., 1989).

7.4 Fundamentos de una RNA

Los fundamentos son modelos de aprendizaje profundo que consisten en capas de neuronas conectadas, organizadas bajo una estructura capa de entrada, capas ocultas y capa de salida. A continuación, se presenta una breve descripción interpretada matemáticamente de una red neuronal simple:

Sea $\mathbf{X}$ un vector de entrada de tamaño n representando características, y Y la salida deseada. La red neuronal realiza transformaciones mediante capas conectadas:

7.5 Capa de Entrada

$\mathbf{X}$ se introduce en la red como la capa de entrada: $\mathbf{X} = [x_1, x_2, \ldots, x_n]$.

7.6 Capas Ocultas

Cada neurona en las capas ocultas realiza una combinación lineal de las entradas seguida de una función de activación no lineal.

$$z_j^{(l)} = \sum_{i=1}^{n} w_{ij}^{(l)} x_i + b_j^{(l)} \tag{9}$$

$$a_j^{(l)} = \Phi(z_j^{(l)}) \tag{10}$$

Donde:

- $w_{ij}^{(l)}$ es el peso de la conexión entre la entrada x_i y la neurona j
- $b_j^{(l)}$ es el sesgo
- Φ es la función activación
- $a_j^{(l)}$ es la salida de la neurona j en la capa l
- $z_j^{(l)}$ es el valor de activación ponderado y sesgado de la neurona j en la capa l

Podemos observar que, en una capa oculta de una red neuronal, cada neurona realiza una combinación lineal de las entradas ponderadas y sesgadas, seguida de una función de activación no lineal, lo que permite que la red aprenda y modele relaciones complejas en los datos.

7.7 Capa de Salida

Esta capa combina las activaciones de las capas ocultas

$$z_k^{(L)} = \sum_{j=1}^{m} w_{kj}^{(L)} a_j^{(L-1)} + b_k^{(L)} \tag{11}$$

$$a_k^{(L)} = \Phi(z_k^{(L)}) \tag{12}$$

Donde:

- $z_k^{(L)}$ es el valor de activación ponderado y sesgado de la neurona k en la capa de salida L
- $w_{kj}^{(L)}$ es el peso de la conexión entre la activación $a_j^{(L-1)}$ de la neurona j en la capa oculta anterior y la neurona k en la capa de salida L
- $a_j^{(L-1)}$ son las activaciones de la neurona en la capa anterior
- $b_k^{(L)}$ es el sesgo de la neurona k en la capa de salida L

- $a_k^{(L)}$ es la salida de la neurona k en la capa de salida L después de haber aplicado la función de activación Φ al valor de activación $z_k^{(L)}$
- Φ es la función de activación aplicada a $z_k^{(L)}$, esta función puede ser cualquier función no lineal, como la función sigmoide, la función ReLU, la función tangente hiperbólica, etc.

7.8 Función de Pérdida

Definimos una función de pérdida $L(Y, \hat{Y})$ que mide la discrepancia entre la salida predicha $\hat{Y}$ y la salida real Y en un problema de aprendizaje supervisado. Una función de pérdida comúnmente utilizada es la pérdida cuadrática (Mean Squared Error - MSE), que se define como la diferencia al cuadrado entre las predicciones y las salidas reales, y luego se toma el promedio de estas diferencias. La expresión matemática del MSE para un conjunto de m muestras sería:

$$L(Y, \hat{Y}) = \frac{1}{m} \sum_{i=1}^{m} (\hat{Y} - Y)^2 \tag{13}$$

Donde:

- $L(Y, \hat{Y})$ representa la función de pérdida, donde Y son las salidas reales (valores verdaderos o etiquetas) y $\hat{Y}$ son las predicciones realizadas por el modelo.
- m es el número total de muestras en el conjunto de datos, este término se utiliza para normalizar la pérdida.
- $\hat{Y}_i$ es la predicción del modelo para la muestra i en el conjunto de datos, es el valor que la red neuronal predice para la variable de interés.
- Y_i es el valor real, la salida deseada o verdadera, para la muestra i en el conjunto de datos.
- $\sum$ representa la suma de las diferencias al cuadrado entre las predicciones y las salidas reales para todas las muestras en el conjunto de datos.
- $(\hat{Y} - Y)^2$ es la diferencia al cuadrado entre la predicción y el valor real para cada muestra en el conjunto de datos.

La función de pérdida cuadrática mide el promedio de las diferencias al cuadrado entre las predicciones del modelo y los valores reales para todas las muestras. Es una forma de cuantificar lo bien que está funcionando el modelo en términos de la discrepancia entre sus predicciones y las salidas reales. Esta interpretación matemática nos describe cómo una RNA transforma las entradas a través de capas ocultas para producir una salida, aprendiendo los patrones en los datos mediante la adaptación de los pesos y sesgos durante el entrenamiento. Es por esto que son adecuadas para procesar datos de entrada independientes, como imágenes, donde la relación entre los datos no depende del orden temporal.

8 REDES NEURONALES RECURRENTES

La arquitectura propuesta consiste en una capa de entrada que recibe secuencias temporales, seguida por una capa de unidades LSTM (Long Short-Term Memory), conocidas por su capacidad para aprender dependencias temporales a largo plazo. La capa de salida produce la predicción del valor futuro de la acción en cuestión.

8.1 Modelado Secuencial

Los modelos de aprendizaje automático o profundo que tienen como entrada o salida una secuencia se conocen como modelos secuenciales. Ejemplos de datos secuenciales incluyen texto, audio, video, series temporales y otros tipos de sucesiones. Este tipo de modelos están diseñados específicamente para manejar datos que tienen un orden inherente y dependencias temporales entre las diferentes partes de la secuencia. Al utilizar técnicas como las redes neuronales recurrentes (RNN) [9].

las redes neuronales convolucionales 1D (CNN 1D) y las redes neuronales Transformers, los modelos secuenciales pueden capturar eficazmente la estructura y las relaciones complejas presentes en datos secuenciales, permitiendo tareas como la traducción automática, la generación de texto, el reconocimiento de voz, la clasificación de series temporales y muchas otras aplicaciones relacionadas con el procesamiento de secuencias de datos.

8.2 Redes Neuronales Recurrentes una vista más amplia

Los estudios IA han generado gran variedad de trabajos de investigación sobre las redes neuronales, la aplicación de técnicas de aprendizaje profundo en la predicción de índices bursátiles ha ganado interés significativo en el ámbito financiero. Este estudio se centra en la implementación de Redes Neuronales Recurrentes (RNN) para modelar y predecir el comportamiento del índice S&P 500, aprovechando la capacidad de las RNN para capturar patrones temporales complejos en series financieras.

Las RNN son una clase especial de arquitecturas de redes neuronales que incorporan conexiones retroalimentadas, permitiendo que la información persista a lo largo del tiempo y creando una especie de memoria interna. Este diseño las hace excepcionalmente aptas para manejar datos secuenciales y tareas que involucran dependencias temporales, como el procesamiento del lenguaje natural, la predicción de series temporales y la música generativa.

A continuación, se mostrará matemáticamente cómo funciona este modelo.

[9] Una Red Neuronal Recurrente (RNN) es un tipo de red neuronal que es capaz de manejar secuencias de datos de cualquier tamaño mediante la utilización de ciclos en la red, permitiendo que la información persista.

8.3 Arquitectura de una RNN

Dada una secuencia de entrada $\mathbf{X} = \{x_1, x_2, \ldots, x_T\}$ de longitud T, la propagación hacia adelante en una RNN se realiza mediante las siguientes ecuaciones en cada paso de tiempo t.

8.4 Cálculo del Estado Oculto

$$h_t = \Phi(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \tag{14}$$

Donde:

- h_t es el estado oculto en el tiempo t
- Φ es una función de activación aplicada elemento a elemento
- W_{hh} es la matriz de pesos que rige la transición del estado oculto anterior h_{t-1} a h_t
- W_{xh} es la matriz de pesos que transforma la entrada x_t al estado oculto h_t
- b_h es el sesgo asociado

El cálculo del estado oculto en una Red Neuronal Recurrente (RNN) es fundamental para entender cómo la red procesa y almacena información a lo largo del tiempo. El estado oculto en una RNN se refiere a la representación interna de la red en un momento dado, que encapsula la información relevante de las entradas anteriores y puede influir en las predicciones futuras.

8.5 Cálculo de la Salida

$$y_t = \sigma(W_{hy}h_t + b_y) \tag{15}$$

Donde:

- y_t es la salida en el tiempo t
- σ es una función de activación aplicada a la salida
- W_{hy} es la matriz de pesos que conecta el estado oculto h_t con la salida y_t
- b_y es el sesgo asociado

El resultado del cálculo ponderado h_t se convierte en el nuevo estado oculto en el tiempo t. Este proceso se repite en cada paso de tiempo, permitiendo que la red capture dependencias temporales y mantenga una representación interna de la información relevante.

Figure 2: Red Neuronal Recurrente

A continuación, tenemos una vista gráfica de una RNN y cómo se comporta en los procesos de aprendizaje:

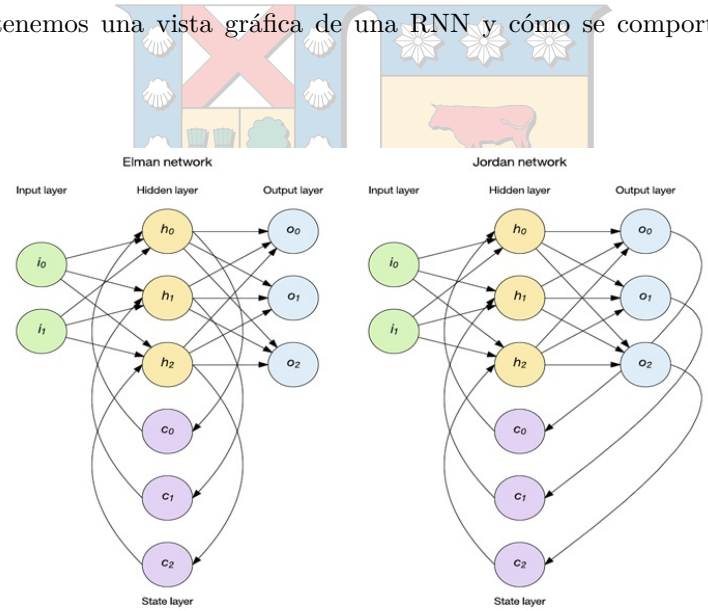


Figure 3: RNN

9 RNN-LSTM

Una RNN-LSTM es una extensión de una RNN tradicional, Las RNN fueron diseñadas para manejar datos secuenciales al conservar información de pasos previos en el tiempo, pero tienen una limitación significativa: el "problema del desvanecimiento o explosión del gradiente". Este problema dificulta que las RNN aprendan dependencias a largo plazo en secuencias, porque las actualizaciones de peso se vuelven demasiado pequeñas (o grandes) cuando se propagan hacia atrás en una secuencia larga.

En una capa LSTM viene a solucionar este problema, en lugar de tener una simple neurona recurrente que procesa la información y la transmite al siguiente paso temporal, existe una estructura de "celdas de memoria" y "puertas" (gates) que controlan el flujo de información. Esto permite que las LSTM "recuerden" y "olviden" información en los momentos adecuados, reteniendo mejor la información relevante a largo plazo.

10 Funcionamiento de una Celda LSTM

Una celda LSTM realiza los siguientes pasos en cada paso temporal t:

Supongamos que la entrada de datos en el tiempo t es x_t, y el estado oculto de la celda en el tiempo anterior es h_{t-1}.

10.1 Estado de la celda

La celda LSTM también tiene un estado de celda C_t, que es el "estado de memoria" que se lleva a través del tiempo y se actualiza en cada paso temporal.

10.2 Estructura de las LSTM

Las LSTM tienen tres puertas principales:

- Puerta de Olvido (f_t)
- Puerta de Entrada (i_t)
- Puerta de Salida (o_t)

Estas puertas están diseñadas para controlar qué información se mantiene o se olvida en cada paso.

10.3 1. Puerta de Olvido f_t

La puerta de olvido determina qué parte de la memoria de la celda anterior (C_{t-1}) se mantiene y cuál se olvida. Matemáticamente, esto se expresa como:

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

donde:

- W_f son los pesos de la puerta de olvido.
- b_f es el sesgo de la puerta de olvido.
- σ es la función sigmoide, que restringe la salida a valores entre 0 y 1.
- $[h_{t-1}, x_t]$ representa la concatenación del estado oculto anterior y la entrada actual.

La salida de la puerta de olvido, f_t, es un vector con valores entre 0 y 1. Un valor cercano a 1 significa que la celda mantendrá información previa, mientras que un valor cercano a 0 significa que olvidará esa información.

10.4 2. Puerta de Entrada i_t

La puerta de entrada controla cuánto de la nueva información $\tilde{C}_t$ (candidata a ser almacenada en la celda) se añadirá al estado de la celda. La puerta de entrada se calcula como:

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

donde:

- W_i son los pesos de la puerta de entrada.
- b_i es el sesgo de la puerta de entrada.

La información candidata $\tilde{C}_t$, que se suma al estado de la celda, se calcula mediante una función tanh:

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

Finalmente, el nuevo estado de la celda C_t se actualiza usando la combinación de la información que se retiene ($f_t \cdot C_{t-1}$) y la nueva información ($i_t \cdot \tilde{C}_t$):

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

10.5 3. Puerta de Salida o_t

La puerta de salida controla qué parte del estado de la celda se usará como la salida de la celda LSTM. Esta salida también se usa como el nuevo estado oculto h_t. La puerta de salida se calcula como:

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

La salida final h_t se determina aplicando la función tanh al estado de la celda actualizado C_t y luego multiplicándolo por o_t:

$$h_t = o_t \cdot \tanh(C_t)$$

11 Resumen de las Ecuaciones

Para cada paso temporal t, las ecuaciones de una celda LSTM son las siguientes:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad \text{(Puerta de Olvido)}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad \text{(Puerta de Entrada)}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad \text{(Candidato a nuevo estado de la celda)}$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \quad \text{(Estado de la celda actualizado)}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad \text{(Puerta de Salida)}$$

$$h_t = o_t \cdot \tanh(C_t) \quad \text{(Salida de la celda y nuevo estado oculto)}$$

11.1 Explicación de cada parámetro

- W_f, W_i, W_C, W_o: Son las matrices de pesos de cada una de las puertas y del estado de celda.
- b_f, b_i, b_C, b_o: Son los términos de sesgo de cada puerta y del estado de celda.
- f_t: Puerta de olvido que decide qué información del estado de celda previo debe "olvidarse".
- i_t: Puerta de entrada que decide qué nueva información debe añadirse al estado de la celda.
- $\tilde{C}_t$: Estado candidato a ser añadido a la celda, creado a partir de la entrada actual y el estado oculto anterior.
- C_t: Estado de celda actualizado que lleva la información de la memoria en cada paso temporal.
- o_t: Puerta de salida que decide qué parte del estado de la celda se usará como salida.
- h_t: Estado oculto (salida) que transporta la información secuencial al siguiente paso temporal.

En cada paso temporal, una celda LSTM procesa la información de forma dinámica mediante una estructura que le permite controlar cuidadosamente qué información debe conservar o descartar a lo largo del tiempo. Esto se logra a través de tres componentes fundamentales denominados puertas: la puerta de olvido, la puerta de entrada y la puerta de salida.

Cada puerta aplica funciones de activación específicas, como la sigmoide y la tangente hiperbólica, que limitan los valores a rangos que facilitan la regulación de la memoria en cada instante. De esta manera, el LSTM no solo almacena y actualiza información relevante para la secuencia, sino que también puede adaptarse a la variabilidad temporal, preservando la continuidad de datos importantes y descartando aquellos menos relevantes a lo largo de la serie temporal.

12 RETROPROPAGACIÓN A TRAVÉS DEL TIEMPO (BACK-PROPAGATION THROUGH TIME - BPTT)

La Retropropagación a Través del Tiempo (BPTT) es una técnica central en el correcto adiestramiento de modelos de redes neuronales recurrentes (RNN), desempeña un papel crucial en la capacitación de estos sistemas en la comprensión de patrones en datos secuenciales, tales como series temporales. La BPTT se caracteriza por desplegar la red a lo largo del tiempo, determinando la secuencia y permitiendo así la retroalimentación de los gradientes de pérdida a lo largo de las distintas instancias temporales.

Este enfoque encuentra particular aplicación en modelos LSTM (Long Short-Term Memory), los cuales se han destacado por su capacidad para modelar dependencias a largo plazo en secuencias temporales, superando las limitaciones de las redes neuronales recurrentes tradicionales. La BPTT actúa como un mecanismo clave para el ajuste de los parámetros de los LSTM a lo largo del tiempo, facilitando la captura eficiente de dependencias temporales complejas.

Por otro lado, se destaca que, aunque los modelos GARCH (Generalized Autoregressive Conditional Heteroskedasticity) son predominantemente utilizados en contextos financieros para modelar la volatilidad condicional, no se adscriben de manera directa al paradigma de la BPTT, dado que su enfoque se centra en la modelización de la varianza condicional basada en el estudio de observaciones pasadas.

En última instancia, aunque los modelos LSTM y GARCH se desenvuelven en ámbitos diferentes, ambos pueden beneficiarse de enfoques de optimización y técnicas de entrenamiento adaptadas a sus respectivos contextos y características particulares. La BPTT, al ser una herramienta versátil, ofrece un método efectivo para el adiestramiento de modelos que operan en dominios temporales, aportando a la mejora del rendimiento y la capacidad predictiva de dichos modelos.

Para entrenar la RNN, se utiliza el algoritmo de retropropagación a través del tiempo (BPTT) para ajustar los pesos y sesgos. El objetivo es minimizar la función de pérdida L que mide la discrepancia entre las predicciones y los valores reales a lo largo de la secuencia temporal.

$$L = \frac{1}{T} \sum_{t=1}^{T} L_t(y_t, \hat{y}_t) \tag{16}$$

Donde:

- L_t es la función de pérdida en el tiempo t
- $\hat{y}_t$ es la predicción de la red en el tiempo t

Entonces podemos calcular el error entre la salida real Y y la salida de la red $\hat{Y}$. Esto puede hacerse utilizando la función de pérdida $L(Y, \hat{Y}) = \frac{1}{m}\sum_{i=1}^{m}(\hat{Y} - Y)^2$:

$$E_t = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_{t,i} - y_{t,i})^2 \tag{17}$$

Ahora necesitamos retropropagar el error a través del tiempo para actualizar los pesos de la red. Esto implica calcular las derivadas parciales del error con respecto a los pesos en cada paso del tiempo.

Comenzamos calculando las derivadas parciales del error con respecto a la salida de la red en el tiempo t, $\frac{\partial E_t}{\partial \hat{Y}_t}$, y luego aplicar la regla de la cadena para obtener las derivadas parciales con respecto a los pesos de la capa oculta W_{hh} y W_{xh} y de entrada W_{hy}.

12.0.1 Derivadas parciales del error con respecto a los pesos de la capa oculta W_{hh} y W_{xh}

1. Respecto a W_{hh}:

$$\frac{\partial E_t}{\partial W_{hh}} = \frac{\partial E_t}{\partial \hat{Y}_t} \cdot \frac{\partial \hat{Y}_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial W_{hh}} \tag{18}$$

2. Respecto a W_{xh}:

$$\frac{\partial E_t}{\partial W_{xh}} = \frac{\partial E_t}{\partial \hat{Y}_t} \cdot \frac{\partial \hat{Y}_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial W_{xh}} \tag{19}$$

12.0.2 Derivada parcial del error con respecto a W_{hy}

$$\frac{\partial E_t}{\partial W_{hy}} = \frac{\partial E_t}{\partial \hat{Y}_t} \cdot \frac{\partial \hat{Y}_t}{\partial W_{hy}} \tag{20}$$

En cada una de estas ecuaciones, las derivadas parciales son calculadas mediante la regla de la cadena, donde $\frac{\partial E_t}{\partial \hat{Y}_t}$ es el gradiente del error con respecto a la salida de la red, $\frac{\partial \hat{Y}_t}{\partial h_t}$ es la derivada de la salida de la red con respecto al estado oculto, y $\frac{\partial h_t}{\partial W}$ es la derivada del estado oculto con respecto a los pesos correspondientes.

Finalmente utilizamos los pesos utilizando un algoritmo de optimización como el descenso del gradiente estocástico (SGD) o el algoritmo de Adam. Este algoritmo combina ideas del descenso del gradiente con momentos (como el descenso del gradiente estocástico con impulso) y la adaptación del aprendizaje por cada parámetro individual (como Adagrad) para calcular los cambios de los pesos durante el entrenamiento. Aquí mostramos cómo sería la actualización de los pesos utilizando el algoritmo Adam:

12.0.3 Actualización de los pesos de la capa oculta W_{hh} y W_{xh}

1. Para W_{hh}:

$$mW_{hh} = \beta_1 mW_{hh} + (1+\beta_1)\frac{\partial E_t}{\partial W_{hh}} \tag{21}$$

$$vW_{hh} = \beta_2 vW_{hh} + (1-\beta_2)\left(\frac{\partial E_t}{\partial W_{hh}}\right)^2 \tag{22}$$

$$m\hat{W}_{hh} = \frac{mW_{hh}}{1-\beta_1^t} \tag{23}$$

$$v\hat{W}_{hh} = \frac{vW_{hh}}{1-\beta_2^t} \tag{24}$$

$$W_{hh\_new} = W_{hh\_old} - \alpha\frac{m\hat{W}_{hh}}{\sqrt{v\hat{W}_{hh}}+\epsilon} \tag{25}$$

2. Para W_{xh}:

$$mW_{hy} = \beta_1 mW_{hy} + (1+\beta_1)\frac{\partial E_t}{\partial W_{hy}} \tag{26}$$

$$vW_{hy} = \beta_2 vW_{hy} + (1-\beta_2)\left(\frac{\partial E_t}{\partial W_{hy}}\right)^2 \tag{27}$$

$$m\hat{W}_{hy} = \frac{mW_{hy}}{1-\beta_1^t} \tag{28}$$

$$v\hat{W}_{hy} = \frac{vW_{hy}}{1-\beta_2^t} \tag{29}$$

$$W_{hy\_new} = W_{hy\_old} - \alpha\frac{m\hat{W}_{hy}}{\sqrt{v\hat{W}_{hy}}+\epsilon} \tag{30}$$

Donde:

- mW_{hh} y mW_{hy} son los momentos del primer orden para W_{hh} y W_{hy}, respectivamente.
- vW_{hh} y vW_{hy} son los momentos de segundo orden para W_{hh} y W_{hy}, respectivamente.
- $m\hat{W}_{hh}$ y $m\hat{W}_{hy}$ son las versiones corregidas de los momentos de primer orden.
- α es la tasa de aprendizaje.
- β_1 y β_2 son los parámetros de decaimiento de los momentos.
- t es el contador de iteraciones.
- ϵ es un término pequeño para evitar la división por cero.

Estas actualizaciones se aplican de forma iterativa durante el entrenamiento de la red neuronal recurrente para ajustar los pesos y minimizar el error en la salida. El algoritmo Adam adapta la tasa de aprendizaje para cada parámetro individual y acumula un historial de gradientes para ajustar los cambios

de los pesos. Esto puede ayudar a mitigar los problemas de desvanecimiento o explosión del gradiente al ajustar dinámicamente la magnitud de las actualizaciones de los pesos. La combinación de BPTT y Adam proporciona una herramienta poderosa para abordar los desafíos del entrenamiento de RNNs y mejorar su capacidad para capturar dependencias a largo plazo en datos secuenciales.

13 APLICACIÓN DE LOS MODELOS PARA PREDICCIÓN DEL S&P 500

Dentro del mundo de las inversiones el S&P 500 es un indicador bastante clave dentro de la Bolsa de valores de EE. UU. Se ha ganado el mérito y consolidado en el tiempo que lleva en operación. Este indicador refleja la evolución en el tiempo de las 500 empresas líderes más líquidas y relevantes de este mercado. Este índice fue creado por Standard & Poor's en 1957 y desde entonces ha sido uno de los principales indicadores del mercado de valores estadounidense. Comprender la predicción del comportamiento de este indicador es fundamental para inversores y analistas de todo el mundo.

La historia de este índice se extiende en el pasado por más de seis décadas, el cual proporciona a inversores y analistas una amplia gama de datos históricos para analizar el comportamiento del mercado de valores a lo largo del tiempo. En cuanto a su impacto en la economía actual, el S&P 500 es ampliamente seguido como un potente indicador del rendimiento general del mercado de valores de Estados Unidos y, por extensión, se considera a menudo como un indicador del estado de la economía en general. Un aumento en el S&P 500 puede ser interpretado como un signo de confianza en la economía, mientras que una disminución puede indicar preocupaciones sobre el crecimiento económico futuro o la estabilidad financiera.

Sin embargo, es importante tener en cuenta que el S&P 500 es solo un indicador y no representa todos los aspectos de la economía local y global. Además, los movimientos en el mercado de valores pueden estar influenciados por una variedad de factores internos y externos, incluyendo cambios en las tasas de interés, políticas gubernamentales, eventos geopolíticos, y desarrollos económicos tanto nacionales como internacionales.

En este estudio nos centraremos en utilizar un conjunto de datos históricos de 100 acciones del S&P 500, trabajaremos con el precio ajustado (precio de cierre) y volumen negociado dentro del horizonte de evaluación, estos datos fueron procesados y normalizados desde una API desde Yahoo Finance, donde cada entrada de la red contiene información sobre los valores pasados de las 100 empresas elegidas para este estudio del S&P 500, el horizonte de datos es desde el año 2015 a 2023. La metodología aplicada en la programación del modelo matemático Red Neuronal Recurrente (RNN), específicamente una variante de RNN que es conocida como Long Short-Term Memory (LSTM) y se explicará en el siguiente capitulo.

14 DESARROLLO DE RNN-LSTM EN PYTHON

14.1 Importación de Bibliotecas

Comenzamos con la instalación de bibliotecas necesarias que se utilizan en el programa desarrollado para este experimento. Estas se instalan y se ejecutan en el código en un entorno Jupyter Notebook o en cualquier entorno de Python que soporte comandos de Shell.

```
# Importamos las librerias que usaremos para el correcto funcionamiento del modelo LSTM y G
import yfinance as yf
import numpy as np
import pandas as pd
from matplotlib import font_manager
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Reshape
from sklearn.metrics import mean_absolute_error
from arch import arch_model
from itertools import groupby
from tensorflow.keras.losses import Huber
import tensorflow as tf
import math
import warnings
from matplotlib import font_manager
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

Figure 4: Bibliotecas

Luego se importaron las bibliotecas que necesitamos para ejecutar correctamente el programa. El detalle de cada una es el siguiente:

a) **yfinance:**

   - **Propósito:** Descargar datos históricos del mercado financiero directamente desde Yahoo Finance.

b) **NumPy:** Proporciona soporte para arrays y matrices multidimensionales, junto con una colección de funciones matemáticas.

c) **Pandas:** Ofrece estructuras de datos y herramientas de análisis para manipular datos numéricos y series temporales.

d) **Matplotlib:** Biblioteca para crear visualizaciones estáticas, animadas e interactivas en Python.

e) **sklearn.preprocessing:** Proporciona herramientas para el preprocesamiento de datos, modelos de aprendizaje automático, y evaluación de modelos.

f) **sklearn.model_selection:** Ofrece herramientas para dividir conjuntos de datos y validación cruzada.

g) **Tensorflow.keras:** API de alto nivel para la creación y entrenamiento de modelos de aprendizaje profundo.

h) **sklearn.metrics:** Incluye funciones para evaluar el rendimiento de modelos de machine learning.

i) **arch:** Implementa modelos ARCH (Autoregressive Conditional Heteroskedasticity) para el análisis de volatilidad.

j) **itertools:** Proporciona funciones para crear iteradores eficientes.

k) **warnings:** Permite controlar advertencias en Python.

l) **math:** Proporciona funciones matemáticas.

14.2 Definición de Símbolos

Definimos una lista de símbolos que representan acciones de empresas del S&P 500. Esta lista sirve como punto de partida para la adquisición de datos históricos y el análisis posterior.

```python
# Lista de símbolos de 100 Acciones de diferentes empresas del S&P 500
symbols = ['ABT', 'ADM', 'LRCX', 'AMAT', 'DHR', 'DE', 'DAL', 'ECL', 'CPB', 'BDX', 'BBY', 'BIIB', 'GILD', 'ACN', 'AMGN', 'AAPL', 'MSFT', 'GOOGL', 'AMZN',

# Verificamos si existen símbolos repetidos en el codigo anterior
repeated_symbols = set()
unique_symbols = set()

for symbol in symbols:
    if symbol in unique_symbols:
        repeated_symbols.add(symbol)
    else:
        unique_symbols.add(symbol)

if repeated_symbols:
    print("Estos son los símbolos repetidos en la lista:")
    for symbol in repeated_symbols:
        print(symbol)
else:
    print("No hay símbolos repetidos en la lista analizada.")

# Separamos los símbolos por la primera letra de abecedario para dar orden
symbols_by_first_letter = {k: list(g) for k, g in groupby(sorted(symbols), key=lambda x: x[0])}

# Imprimir los símbolos en columnas por la primera letra del abecedario

print("Acciones únicas por letra del abecedario que no se repiten:")
for letter, symbols_list in symbols_by_first_letter.items():
    symbols_str = ', '.join(symbols_list)
    print(f"{letter}: {symbols_str}")
```

Figure 5: Simbolos 100 acciones

14.3 Verificación de Simbolos Repetidos en la Lista Utilizando Conjuntos

Esto asegura la integridad de los datos y evita redundancias que puedan surgir durante el análisis.

```
No hay símbolos repetidos en la lista analizada.
Acciones únicas por letra del abecedario que no se repiten:
A: AAPL, ABBV, ABT, ACN, ADBE, ADM, ADP, ADSK, ALL, AMAT, AMD, AMGN, AMZN, AVGO, AXP
B: BA, BAC, BBY, BDX, BIIB, BK, BMY
C: C, CAT, CL, CMCSA, CNC, COF, COP, COST, CPB, CRM, CSCO, CVS, CVX
D: DAL, DD, DE, DHR, DIS, DUK
E: EA, ECL, EIX, EMR, EOG, ETN
F: F, FAST, FDX, FIS
G: GE, GILD, GM, GOOGL, GS
H: HAL, HOG, HON
I: INTC, INTU
J: JCI, JNJ, JPM
K: KMB, KO
L: LMT, LNC, LOW, LRCX
M: MA, MCD, MDLZ, MDT, MMM, MRK, MS, MSFT
N: NEE, NFLX, NKE, NVDA
O: ORCL
P: PEP, PFE, PG, PGR, PRU
Q: QCOM
R: RSG, RTX
S: SCHW, SLB, SNY, SSNLF
T: TGT, TSLA
U: UPS
V: VZ
X: XOM
```

Figure 6: Verificación de simbolos

Como podemos observar realizamos el ordenamiento de símbolos por letra o un diccionario donde las claves son las primeras letras de cada símbolo, y los valores son listas de símbolos que comienzan con esa letra, todo esto es para tener un orden de lo que se está midiendo.

14.4 Descarga de Datos Históricos

Descargamos datos históricos de las acciones utilizando la biblioteca yfinance para el rango de fechas desde 01-01-2015 hasta el 31-12-2023.

```python
# Descargar datos históricos de las 100 acciones de empresas del indice S&P 500 desde el añ
data_frames = []
download_count = 1

for symbol in symbols:
    # Descargar datos históricos de las 100 acciones del S&P 500
    stock_data = yf.download(symbol, start='2015-01-01', end='2023-12-31')

    # Añadimos a la lista solo las columnas que deseamos ver, precio cierre y volumen
    if ('Close' in stock_data.columns) and ('Volume' in stock_data.columns):
        data_frames.append(stock_data[['Close', 'Volume']])
        print(f"Descarga {download_count}: Datos históricos de {symbol} descargados.")
        download_count += 1
    else:
        print(f"No se encontraron las columnas 'Close' y 'Volume' para {symbol}.")
```

Figure 7: Datos 100 Acciones

El resultado es un DataFrame con una jerarquía de columnas donde el primer nivel es el símbolo de la acción y el segundo nivel es el tipo de dato (’Close’ o ’Volume’).

14.5 Filtrado y Preparación de Datos

Se filtraron los datos descargados para quedarnos solo con las columnas de cierre y volumen.

- **Propósito:** Combinar los datos históricos de cierre y volumen de todas las acciones en un solo DataFrame.

```
# Concatenamos los datos
data = pd.concat(data_frames, axis=1, keys=symbols, names=['Symbol', 'Data'])
```

Figure 8: Datos Concatenados

14.6 Reemplazamos los Valores Infinitos por nan en el Programa

```
# Reemplazamos posibles datos infinitos por nan
data = data.replace([np.inf, -np.inf], np.nan)
```

Figure 9: Reemplazo de Datos Infinitos por nan

Luego, un ordenamiento de filtros para las filas con valores NaN en alguna de las columnas de los datos.

```
# Filtrar las filas con valores NaN en alguna columna
filas_con_nan = data[data.isna().any(axis=1)]

print(filas_con_nan)
```

Figure 10: Filtro de Datos

14.7 Normalizar Datos

Luego normalizamos los datos de las acciones utilizando MinMaxScaler. Este paso es crucial antes de aplicar el algoritmo de aprendizaje, especialmente las redes neuronales.

```
# Normalizar los datos
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(data)
```

Figure 11: Normalizado de Datos

Realizamos la normalización de los datos para tener una consistencia en las escalas, mejorar el rendimiento, o más bien mejorar la convergencia y la precisión del modelo, evitando que ciertas características dominen otras debido a sus magnitudes.

14.8 Ajustar el Índice y las Columnas

Primero, se ajusta el índice y las columnas del DataFrame para asegurar que los datos normalizados estén en un formato estructurado y legible.

```
# Ajustar el índice y las columnas
data = pd.DataFrame(scaled_data, index=data.index, columns=data.columns)
```

Figure 12: Ajuste

Se convierten los datos escalados (normalizados) de nuevo en un DataFrame de pandas con los mismos índices y nombres de columnas que el original. También definimos los parámetros que utilizaremos:

- **SEQ_LEN (5):** Número de días de historia que se usarán como entrada para el modelo LSTM.
- **PRED_LEN (2):** Número de días en el futuro que se desea predecir.

```
# Parámetros
SEQ_LEN = 5  # Días de historia
PRED_LEN = 2  # Días a predecir en el futuro
```

Figure 13: Parámetros - Dias de Historia a Predecir al Futuro

14.9 Preprocesamiento de Datos para el Modelo LSTM

Implementamos una función para el procesamiento de datos para el modelo LSTM dividiéndolos en secuencias de entrada y sus correspondientes valores objetivo. Además, aplicamos squeeze para eliminar dimensiones de tamaño 1.

```python
# Función para preprocesar los datos de la serie temporal para prepararlos para su uso en u

#Entradas en Array Numpy
def preprocess_data(data, seq_len, pred_len):

# salidas
    X, y = [], []

    for i in range(len(data) - seq_len - pred_len):
        seq = data[i:i + seq_len]
        x = seq.values.reshape(1, seq_len, data.shape[1])
        X.append(x)

        pred = data.iloc[i + seq_len:i + seq_len + pred_len].values
        y.append(pred)

    return np.array(X), np.array(y)
```

Figure 14: Procesar Datos de Serie Temporal

Llamamos a la función `preprocess_data` para transformar los datos normalizados en secuencias de entrada y salida.

```python
# Preprocesar los datos
X, y = preprocess_data(pd.DataFrame(scaled_data), SEQ_LEN, PRED_LEN)
```

Figure 15: Preprocesar datos

14.10 Eliminar dimensión de tamaño 1

El propósito de esto es convertir los datos normalizados en secuencias de entrada (X) y sus salidas correspondientes (y) para el modelo LSTM.

```python
# Aplicar squeeze para eliminar la dimensión de tamaño 1
X = np.squeeze(X, axis=1)
```

Figure 16: Squeeze para Eliminar Dimensión de Tamaño 1

- Funcionamiento: La función `preprocess_data` toma los datos normalizados y crea secuencias de longitud `SEQ_LEN` y predicciones de longitud `PRED_LEN`.
- Propósito: Eliminar la dimensión de tamaño 1 de los datos de entrada.

Luego nos aseguramos de que no existan valores NaN o infinitos en los datos de entrada y salida.

```
# Verificar la presencia de NaN o infinitos en los conjuntos de entrenamiento y prueba
print(np.isnan(X).any(), np.isnan(y).any())
print(np.isinf(X).any(), np.isinf(y).any())
```

Figure 17: Verificar la Presencia de nan

Funcionamiento:

- `np.isnan(X).any()`: Verifica si hay algún valor NaN en X.
- `np.isnan(y).any()`: Verifica si hay algún valor NaN en y.
- `np.isinf(X).any()`: Verifica si hay algún valor infinito en X.
- `np.isinf(y).any()`: Verifica si hay algún valor infinito en y.

Si cualquiera de estas verificaciones retorna `True`, significaría que hay valores no válidos en los datos, lo cual necesita ser corregido antes de proceder.

División de datos en entrenamiento y prueba: Dividimos los datos en conjuntos de entrenamiento y prueba utilizando train_test_split (técnica de división aleatoria). Esta etapa es esencial para evaluar el rendimiento del modelo en datos no vistos durante el entrenamiento.

```
# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False, random_state=42)
```


Figure 18: Entrenamiento y Prueba

Funcionamiento

- `train_test_split` de `sklearn.model_selection` se usa para dividir los datos.
- X y y son divididos en X_train, X_test, y_train, y_test.
- `test_size=0.2`: Significa que el 20% de los datos se utilizan para el conjunto de prueba y el 80% para el conjunto de entrenamiento.
- `shuffle=False`: Indica que los datos no deben ser mezclados antes de la división. Esto es importante para series temporales para mantener la continuidad temporal.
- `random_state=42`: Asegura que la división de los datos sea reproducible.

14.11 Semilla

Garantizamos que los números aleatorios generados sean reproducibles. Al fijar una semilla tanto en Numpy como en TensorFlow, aseguramos que cualquier experimento o análisis que involucre números aleatorios produzca resultados consistentes y predecibles.

```
# Fijamos una semilla en Numpy
seed = 5
np.random.seed(seed)
tf.random.set_seed(seed)

# Generar números aleatorios con numpy
np_random_numbers = np.random.rand(5)
print("Números aleatorios de numpy:", np_random_numbers)

# Generar números aleatorios con TensorFlow
tf_random_numbers = tf.random.uniform([5])
print("Números aleatorios de TensorFlow:", tf_random_numbers)
```

Figure 19: Seed

14.12 Configuración del Modelo LSTM

Luego se programa un modelo utilizando la arquitectura LSTM, utilizando la API secuencial de la biblioteca Keras de TensorFlow, que consiste en una capa LSTM y una capa densa seguida de una capa de redimensionamiento. Compilamos el modelo utilizando el optimizador Adam y la pérdida basada en el error de Huber.

```
# Configuración del modelo LSTM
model = Sequential()
model.add(LSTM(units=500, activation='relu', input_shape=(SEQ_LEN, data.shape[1])))
model.add(Dense(PRED_LEN * data.shape[1], activation='linear'))
model.add(Reshape((PRED_LEN, data.shape[1])))
model.compile(optimizer='adam', loss=Huber())

# Entrenamiento del modelo
model.fit(X_train, y_train, epochs=200, batch_size=32)

# Evaluación del modelo
loss = model.evaluate(X_test, y_test)
print(f'Loss en conjunto de prueba: {loss}')

# Realizar predicciones
predictions = model.predict(X_test)
```

Figure 20: Configuración - Entrenamiento - Evaluación - Predicción

Propósito: Inicializar un modelo secuencial. Este tipo de modelo es apropiado para una pila simple de capas, donde cada capa tiene exactamente una entrada y una salida.

LSTM:

- `units=500`: El número de unidades de la celda LSTM, que corresponde al número de neuronas en esta capa. Esto determina la dimensionalidad del espacio de salida.
- `activation='relu'`: La función de activación ReLU (Rectified Linear Unit) que se aplicará a las salidas de las celdas LSTM.

- `input_shape=(SEQ_LEN, data.shape[1])`: La forma de los datos de entrada. `SEQ_LEN` es el número de pasos de tiempo en la secuencia de entrada, y `data.shape[1]` es el número de características en cada paso de tiempo.

Dense:

- `PRED_LEN * data.shape[1]`: El número de neuronas en esta capa densa. Multiplicamos el número de características por el número de pasos de tiempo que queremos predecir (`PRED_LEN`).
- `activation='linear'`: La función de activación lineal, adecuada para problemas de regresión donde necesitamos predecir valores continuos.

Reshape:

- `(PRED_LEN, data.shape[1])`: Cambia la forma de la salida de la capa densa a la forma deseada para la predicción final, que es (`PRED_LEN`, `data.shape[1]`). Esto significa que las salidas serán una secuencia de longitud `PRED_LEN` con `data.shape[1]` características en cada paso de tiempo.

Compilación:

- `optimizer='adam'`: El optimizador Adam es utilizado para actualizar los pesos del modelo durante el entrenamiento. Es un optimizador eficiente y ampliamente utilizado para modelos de aprendizaje profundo.
- `loss=Huber()`: La función de pérdida Huber se utiliza para calcular el error entre las predicciones del modelo y los valores reales. La pérdida Huber es menos sensible a valores atípicos en comparación con el error cuadrático medio (MSE), lo que la hace adecuada para problemas de regresión.

14.13 Entrenamiento del Modelo

Procedemos a entrenar el modelo LSTM utilizando los datos de entrenamiento. Este proceso implica la optimización de los parámetros del modelo a través de múltiples iteraciones utilizando el conjunto de datos proporcionado.

El objetivo del entrenamiento es minimizar la función de pérdida (Huber) ajustando los pesos del modelo. A medida que el modelo se entrena durante múltiples epochs, se espera que el error de predicción disminuya, mejorando la capacidad del modelo para generalizar y hacer predicciones precisas en datos no vistos (datos de prueba).

En resumen, este bloque de código entrena el modelo LSTM utilizando los datos de entrenamiento X_train y y_train durante 200 epochs, con un tamaño de lote de 32, ajustando los pesos del modelo para minimizar el error de predicción.

14.14 Evaluación del modelo

Luego se evaluó el rendimiento del modelo entrenado utilizando los datos de prueba. Este paso es crucial para determinar la capacidad predictiva del modelo en datos no observados durante el proceso de entrenamiento.

Evaluación en Datos de Prueba:

- La evaluación en datos de prueba es crucial para medir la capacidad del modelo para generalizar a nuevos datos no vistos durante el entrenamiento.
- Aquí, se utiliza el conjunto X_test (datos de entrada) e y_test (etiquetas verdaderas) para evaluar el modelo.

Pérdida (Loss)

- La función de pérdida utilizada durante la evaluación es la misma que se utilizó durante el entrenamiento, en este caso, la pérdida de Huber.
- El valor de la pérdida indica qué tan bien o mal está funcionando el modelo en el conjunto de prueba. Un valor de pérdida más bajo indica un mejor rendimiento del modelo.

Proceso de Evaluación:

1. Forward Pass:
   - El modelo calcula las predicciones para las muestras de entrada en X_test.
2. Cálculo de la Pérdida:
   - Compara las predicciones del modelo con las etiquetas verdaderas en y_test utilizando la función de pérdida de Huber y calcula el error.
3. Resultado:
   - Devuelve el valor de la pérdida calculada, que se almacena en la variable `loss`.

Objetivo de la Evaluación:

- El objetivo de la evaluación es medir el rendimiento del modelo en un conjunto de datos separado (datos de prueba) que no se utilizaron durante el entrenamiento. Esto proporciona una estimación de la capacidad del modelo para generalizar a datos nuevos y no vistos.

14.15 Predicciones y evaluación adicional

Se realizaron predicciones utilizando el modelo entrenado y se calculó el error absoluto medio (MAE) para cada variable en el conjunto de prueba. Esta evaluación adicional proporcionó información sobre la precisión del modelo en la predicción de valores futuros.

14.16 Suavizado exponencial en Plot

```python
# Función para aplicar suavizado exponencial
def exponential_smoothing(data, alpha):
    # Convertir a DataFrame de Pandas
    data_df = pd.DataFrame(data)
    smoothed_data = data_df.ewm(alpha=alpha, adjust=False).mean()
    return smoothed_data.values.flatten()  # Convertir de nuevo a array NumPy después del suavizado

# Definir el número de filas y columnas para los subplots
num_cols = 2  # número de columnas
num_rows = 3  # número de filas
plots_per_figure = num_cols * num_rows

# Colores para las predicciones
prediction_colors = ['blue', 'green', 'red', 'purple', 'brown'] * ((len(symbols) // 5) + 1)

# Tamaños de fuente y marcadores
font_size_title = 11
font_size_labels = 11
font_size_legend = 11
marker_size = 0.5

# Graficar predicciones y valores reales para cada empresa
for fig_num in range((len(symbols) // plots_per_figure) + 1):
    fig, axs = plt.subplots(num_rows, num_cols, figsize=(20, 15))
    axs = axs.flatten()

    for i in range(plots_per_figure):
        symbol_index = fig_num * plots_per_figure + i
        if symbol_index >= len(symbols):
```

Figure 21: Suavizado exponencial en plot

La función `exponential_smoothing(data, alpha)` aplica suavizado exponencial a los datos utilizando Pandas para manejar los datos como un DataFrame, luego calcula la media exponencial ponderada con un parámetro de suavizado α y devuelve los datos suavizados como un array NumPy.

Se configuran variables para organizar los gráficos en una cuadrícula de subplots, especificando el número de columnas (`num_cols`) y filas (`num_rows`). Esto permite visualizar múltiples conjuntos de datos simultáneamente.

Los colores (`prediction_colors`) se definen para las líneas de los gráficos basándose en una lista predefinida de colores, ajustada para el número de símbolos de interés.

Además, se establecen tamaños de fuente (`font_size_title`, `font_size_labels`, `font_size_legend`) y el tamaño de los marcadores (`marker_size`) para asegurar que los gráficos sean legibles y estéticamente agradables.

Finalmente, se itera sobre los símbolos disponibles para generar gráficos individuales para cada uno, utilizando Matplotlib para crear subplots dentro de una figura grande.

```
            break
        symbol = symbols[symbol_index]
        ax = axs[i]

        # Aplicar suavizado exponencial a las predicciones
        smoothed_predictions = exponential_smoothing(predictions[:, :, symbol_index].reshape(-1), alpha=0.2)

        # Graficar predicciones suavizadas con colores diferentes
        ax.plot(smoothed_predictions, label=f'Predicciones {symbol} (Suavizadas)', color=prediction_colors[symbol_index], marker='o', linestyle='--', mar

        # Graficar valores reales con color constante (naranja)
        ax.plot(y_test[:, :, symbol_index].reshape(-1), label=f'Valores Reales {symbol}', color='orange', marker='x', linestyle='-', markersize=marker_si

        # Configurar título, etiquetas y leyenda
        ax.set_title(f'Predicciones vs Valores Reales para {symbol}', fontsize=font_size_title)
        ax.set_xlabel('Tiempo', fontsize=font_size_labels)
        ax.set_ylabel('Valor', fontsize=font_size_labels)
        ax.legend(fontsize=font_size_legend)

    plt.tight_layout()
    plt.show()
```

Figure 22: Suavizado exponencial en plot

14.17 MAE, MSE y RMSE

Mean Absolute Error (MAE):

El Mean Absolute Error (MAE) calcula el error absoluto medio entre las predicciones y los valores reales para cada variable en el conjunto de prueba.

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

```
# Calcular el MAE
mae_test = mean_absolute_error(y_test.reshape(-1, data.shape[1]), predictions.reshape(-1, data.shape[1]), multioutput='raw_values')
print("MAE para cada variable en conjunto de prueba:", mae_test)

# Calcular el MSE
mse_test = mean_squared_error(y_test.reshape(-1, data.shape[1]), predictions.reshape(-1, data.shape[1]), multioutput='raw_values')
print("MSE para cada variable en conjunto de prueba:", mse_test)

# Calcular el RMSE
rmse_test = np.sqrt(mse_test)
print("RMSE para cada variable en conjunto de prueba:", rmse_test)
```

Figure 23: MAE MSE y RMSE

Mean Squared Error (MSE):

El Mean Squared Error (MSE) calcula el error cuadrático medio entre las predicciones y los valores reales para cada variable en el conjunto de prueba.

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

Root Mean Squared Error (RMSE):

El Root Mean Squared Error (RMSE) calcula la raíz cuadrada del MSE, proporcionando una medida del

error en la misma unidad que los datos originales.

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

Donde:

- n es el número de muestras.
- y_i son los valores reales.
- $\hat{y}_i$ son las predicciones del modelo.

MAE, MSE y RMSE son herramientas fundamentales en la evaluación y ajuste en el modelo LSTM y DCC-GARCH utilizados en el tema de investigación, estos proporcionan medidas cuantitativas de la precisión y el error de las predicciones en diferentes contextos de modelado y análisis de series temporales.

modelos con un menor valor de RMSE, diremos que tienen mejor desempeño.

14.18 Cálculo de Rentabilidad y Volatilidad

Se implementó una función para calcular la rentabilidad y la volatilidad para cada símbolo utilizando datos históricos de Yahoo Finance. Este análisis proporcionó información sobre el rendimiento financiero de cada acción a lo largo del tiempo.

```python
# Lista de símbolos de 100 Acciones de diferentes empresas del S&P 500
symbols = ['ABT', 'ADM', 'LRCX', 'AMAT', 'DHR', 'DE', 'DAL', 'ECL', 'CPB', 'BDX', 'BBY', 'BIIB', 'GILD', 'ACN', 'AMGN

# Función para obtener datos históricos de una acción y calcular rentabilidad y volatilidad
def calcular_rentabilidad_y_volatilidad(symbol):
    # Obtener datos históricos desde el año 2015 hasta 2023
    data = yf.download(symbol, start="2015-01-01", end="2023-12-31")

    # Calcular la rentabilidad
    data['Rentabilidad'] = (data['Close'] / data['Close'].shift(1) - 1) * 100

    # Calcular la volatilidad
    data['Volatilidad'] = data['Rentabilidad'].rolling(window=252).std() * (252 ** 0.5)

    # Agrupar por año y calcular la rentabilidad media y la volatilidad media
    rentabilidad_anual = data['Rentabilidad'].resample('Y').mean()
    volatilidad_anual = data['Volatilidad'].resample('Y').mean()

    # Retornar los resultados por año
    return rentabilidad_anual, volatilidad_anual

# Crear un DataFrame vacío para almacenar los resultados
resultados = pd.DataFrame(columns=['Símbolo', 'Año', 'Rentabilidad', 'Volatilidad'])

# Calcular rentabilidad y volatilidad para cada símbolo
for symbol in symbols:
    rentabilidad_anual, volatilidad_anual = calcular_rentabilidad_y_volatilidad(symbol)
    for year in rentabilidad_anual.index:
        resultados.loc[len(resultados)] = [symbol, year.year, rentabilidad_anual[year], volatilidad_anual[year]]

# Imprimir los resultados
print(resultados)
```

Figure 24: Rentabilidad y Volatilidad

Esta sección del código realiza los siguientes pasos:

1. Lista de Símbolos

- Utilizamos la lista de los 100 símbolos de las acciones de diferentes empresas del S&P500.

2. Rentabilidad y Volatilidad

- **calcular_rentabilidad_y_volatilidad(symbol)**: Obtiene los datos históricos de precios de cierre de una acción desde 2015 hasta 2023 usando yfinance. Luego, calcula la rentabilidad diaria y la volatilidad anualizada de la acción.

3. Datos Históricos

- **Rentabilidad**: Se calcula como el porcentaje de cambio diario del precio de cierre.
- **Volatilidad**: Se calcula como la desviación estándar de la rentabilidad diaria anualizada con una ventana de 252 días (aproximadamente el número de días hábiles en un año).

4. Almacenamiento de los Resultados

- Se creó un DataFrame vacío llamado **resultados**, para almacenar la rentabilidad y volatilidad anual para cada símbolo.
- Luego, al calcular la rentabilidad y volatilidad anual por símbolo, se imprime el DataFrame **resultados**.

14.19 Convertir valores nan a cero “0”

Implementar esta función permite mantener la integridad y continuidad del análisis de datos, asegurando que no haya interrupciones o errores debido a la presencia de valores nan.

```
# Convertir NaN a 0 en las columnas 'Rentabilidad' y 'Volatilidad'
resultados['Rentabilidad'] = resultados['Rentabilidad'].fillna(0)
resultados['Volatilidad'] = resultados['Volatilidad'].fillna(0)
```

Figure 25: Convertir nan a Cero en las Columnas Rentabilidad y Volatilidad

Esta sección del código realiza los siguientes pasos:

1. Lista de símbolos

- Utilizamos la lista de los 100 símbolos de las acciones de diferentes empresas del S&P500.

2. Rentabilidad y volatilidad

- **calcular_rentabilidad_y_volatilidad(symbol)**: Obtiene los datos históricos de precios de cierre de una acción desde 2015 hasta 2023 usando yfinance. Luego, calcula la rentabilidad diaria y la volatilidad anualizada de la acción.

3. Datos históricos

- **Rentabilidad**: Se calcula como el porcentaje de cambio diario del precio de cierre.
- **Volatilidad**: Se calcula como la desviación estándar de la rentabilidad diaria anualizada con una ventana de 252 días (aproximadamente el número de días hábiles en un año).

4. Almacenamiento de los resultados

- Se creó un DataFrame vacío llamado **resultados**, para almacenar la rentabilidad y volatilidad anual para cada símbolo.
- Luego, al calcular la rentabilidad y volatilidad anual por símbolo, se imprime el DataFrame **resultados**.

14.20 Imprimir Cálculo de Rentabilidad y Volatilidad

Ordenamos alfabéticamente los símbolos de las acciones y luego iteramos sobre cada símbolo para graficar su rentabilidad y volatilidad anual. Para cada símbolo, se generan dos gráficos de barras:

- **Rentabilidad por año**: Se muestra la rentabilidad anual de la acción. Las barras son de color naranja si la rentabilidad es positiva y roja si es negativa.
- **Volatilidad por año**: Este gráfico muestra la volatilidad anual de la acción. Las barras son de color azul si la volatilidad es positiva y roja si es negativa.

```python
import matplotlib.pyplot as plt

# Ordenar los símbolos alfabéticamente
symbols_ordenados = sorted(resultados['Símbolo'].unique())

# Lista para almacenar los símbolos con error
simbolos_con_error = []

# Contadores para rentabilidad y volatilidad
rentabilidad_graficos = 0
volatilidad_graficos = 0

# Definir el número de filas y columnas para los subplots
num_cols = 2  # número de columnas
num_rows = 3  # número de filas
plots_per_figure = num_cols * num_rows

# Función para graficar un conjunto de subplots
def plot_subplots(symbols, tipo):
    fig, axs = plt.subplots(num_rows, num_cols, figsize=(20, 15))
    axs = axs.flatten()

    for i, symbol in enumerate(symbols):
        if i >= len(axs):
            break
        ax = axs[i]
        try:
            # Obtener los datos de la empresa actual
            grupo = resultados[resultados['Símbolo'] == symbol]

            if tipo == 'Rentabilidad':
                # Graficar la rentabilidad por año (gráfico de líneas)
                rentabilidad_por_ano = grupo.pivot(index='Año', columns='Símbolo', values='Rentabilidad')[symbol]
                rentabilidad_por_ano.plot(ax=ax, kind='line', color='orange', label='Rentabilidad')
```

Figure 26: Parte 1 del gráfico

```
            # Configurar etiquetas y título del gráfico de rentabilidad
            ax.set_title(f'Rentabilidad por Año - {symbol}')
            ax.set_xlabel('Año')
            ax.set_ylabel('Rentabilidad')
            ax.legend(loc='upper left')

            # Incrementar el contador de gráficos de rentabilidad
            global rentabilidad_graficos
            rentabilidad_graficos += 1

        elif tipo == 'Volatilidad':
            # Graficar la volatilidad por año (gráfico de líneas)
            volatilidad_por_ano = grupo.pivot(index='Año', columns='Símbolo', values='Volatilidad')[symbol]
            volatilidad_por_ano.plot(ax=ax, kind='line', color='blue', label='Volatilidad')

            # Configurar etiquetas y título del gráfico de volatilidad
            ax.set_title(f'Volatilidad por Año - {symbol}')
            ax.set_xlabel('Año')
            ax.set_ylabel('Volatilidad')
            ax.legend(loc='upper left')

            # Incrementar el contador de gráficos de volatilidad
            global volatilidad_graficos
            volatilidad_graficos += 1

    except Exception as e:
        # Agregar el símbolo a la lista de errores
        simbolos_con_error.append(symbol)
        print(f"Error al graficar {symbol}: {e}")

# Ajustar el espaciado alrededor de los gráficos
plt.tight_layout()
```

Figure 27: Parte 2 del grafico

```
# Ajustar el espaciado alrededor de los gráficos
plt.tight_layout()

# Mostrar los gráficos
plt.show()

# Graficar rentabilidad
for fig_num in range((len(symbols_ordenados) // plots_per_figure) + 1):
    plot_subplots(symbols_ordenados[fig_num * plots_per_figure:(fig_num + 1) * plots_per_figure], 'Rentabilidad')

# Graficar volatilidad
for fig_num in range((len(symbols_ordenados) // plots_per_figure) + 1):
    plot_subplots(symbols_ordenados[fig_num * plots_per_figure:(fig_num + 1) * plots_per_figure], 'Volatilidad')

# Imprimir los símbolos que arrojaron error
if simbolos_con_error:
    print("Símbolos con error al graficar:")
    for simbolo in simbolos_con_error:
        print(simbolo)

# Imprimir el número de gráficos generados de rentabilidad y volatilidad
print(f"Número de gráficos de Rentabilidad: {rentabilidad_graficos}")
print(f"Número de gráficos de Volatilidad: {volatilidad_graficos}")
```

Figure 28: Parte 3 del grafico

los resultados de los valores de volatilidad y rentabilidad en forma de gráficos, los análisis de los resultados se indican en la sección resultados experimentales.

15 DESARROLLO DE MODELO GARCH EN PYTHON

15.1 Importación de Bibliotecas

Comenzamos con la instalación de bibliotecas necesarias que se utilizan para instalar las para ejecutar el código en un entorno Jupyter Notebook o en cualquier entorno de Python que soporte comandos de shell.

```python
# Lista de símbolos de 100 Acciones de diferentes empresas del S&P 500
symbols = ['ABT', 'ADM', 'LRCX', 'AMAT', 'DHR', 'DE', 'DAL', 'ECL', 'CPB', 'BDX', 'BBY', 'BIIB', 'GILD', 'ACN', 'AMGN',
           'AAPL', 'MSFT', 'GOOGL', 'AMZN', 'NVDA', 'UPS', 'ADSK', 'ADP', 'AXP', 'AMD', 'ADBE', 'MMM', 'VZ', 'BAC', 'BK',
           'AVGO', 'BA', 'BMY', 'COF', 'CAT', 'CVX', 'CSCO', 'C', 'KO', 'CL', 'CMCSA', 'ABBV', 'CNC', 'SCHW', 'COP', 'CVS',
           'NEE', 'DIS', 'MDT', 'DUK', 'ETN', 'DD', 'XOM', 'GE', 'FDX', 'F', 'GM', 'GS', 'HAL', 'HOG', 'HON', 'INTC', 'INTU',
           'JPM', 'JNJ', 'JCI', 'KMB', 'TSLA', 'LNC', 'LMT', 'LOW', 'MA', 'MCD', 'MRK', 'ORCL', 'MDLZ', 'COST', 'MS', 'NFLX',
           'NKE', 'PEP', 'PFE', 'PG', 'PGR', 'PRU', 'QCOM', 'RTX', 'RSG', 'CRM', 'SSNLF', 'SNY', 'SLB', 'ALL', 'FAST', 'FIS',
           'EA', 'EIX', 'EMR', 'EOG', 'TGT']

# DataFrame para almacenar las métricas de evaluación
metrics_df = pd.DataFrame(columns=['Symbol', 'AIC', 'BIC', 'Log-Likelihood', 'RMSE', 'MSE', 'MAE'])

# Definir el número de filas y columnas para los subplots
num_cols = 2
num_rows = 3

# Inicializar el índice de la tabla
table_index = 1

for i, symbol in enumerate(symbols):
    # Descarga de datos históricos
    data = yf.download(symbol, start='2015-01-01', end='2023-01-01')
    data['Return'] = data['Adj Close'].pct_change().dropna()

    # Preparación de los datos
    returns = data['Return'].dropna() * 100  # Convertir a porcentajes
```

Figure 29: Lista 100 Simbolos S&P 500

Para cada símbolo en la lista `symbols`, el código realiza los siguientes pasos para preparar los datos antes de aplicar el modelo GARCH:

1. **Descarga de datos históricos:**

   ```
   data = yf.download(symbol, start='2015-01-01', end='2023-01-01')
   ```

   Descarga los datos históricos desde Yahoo Finance para el símbolo especificado (`symbol`) desde el 1 de enero de 2015 hasta el 1 de enero de 2023.

2. **Cálculo de rendimientos diarios ajustados:**

   ```
   data['Return'] = data['Adj Close'].pct_change().dropna()
   ```

   Calcula los rendimientos diarios ajustados como el cambio porcentual del precio de cierre ajustado (`Adj Close`). El método `pct_change()` calcula el cambio porcentual respecto al período anterior, y `dropna()` elimina cualquier valor NaN resultante.

3. **Conversión a porcentaje:**

```
returns = data['Return'].dropna() * 100
```

Multiplica los rendimientos diarios por 100 para obtener los rendimientos diarios ajustados en porcentaje (`returns`).

16 Modelado GARCH y Predicción de Volatilidad

16.1 División de los Datos

El primer paso en nuestro análisis es la división de los datos en conjuntos de entrenamiento y prueba:

```
train_size = int(0.8 * len(returns))
train, test = returns[:train_size], returns[train_size:]
```

Explicación: Esta división es fundamental en el aprendizaje estadístico y la econometría financiera. Utilizamos el 80% de los datos para el entrenamiento y el 20% para las pruebas, una práctica común que equilibra la necesidad de un conjunto de entrenamiento robusto con la validación adecuada del modelo.

Importancia: Esta separación nos permite:

- Entrenar el modelo en un conjunto de datos sustancial.
- Evaluar su rendimiento en datos no vistos, simulando condiciones del mundo real.
- Evitar el sobreajuste, asegurando que nuestro modelo generalice bien a nuevos datos.

16.2 Ajuste del Modelo GARCH

A continuación, procedemos a ajustar un modelo GARCH(1,1) a nuestro conjunto de entrenamiento:

```
garch_model = arch_model(train, vol='Garch', p=1, q=1)
garch_fit = garch_model.fit(disp="off")
```

Explicación técnica: El modelo GARCH(1,1) se define como:

$$h_t = \alpha_0 + \sum_{i=1}^{p} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j h_{t-j} \tag{31}$$

donde:

- h_t es la varianza condicional en el tiempo t.
- α_0 es el coeficiente constante.
- α_i son los coeficientes para los términos de error al cuadrado anteriores (ε_{t-i}^2).
- β_j son los coeficientes para los términos de varianza condicional anterior (h_{t-j}).
- p y q son los órdenes de los términos GARCH y ARCH, respectivamente.

Importancia: El modelo GARCH(1,1) captura dos características cruciales de las series financieras:

- Agrupamiento de volatilidad: períodos de alta volatilidad tienden a ser seguidos por más alta volatilidad.
- Reversión a la media: la volatilidad tiende a volver a su nivel promedio a largo plazo.

A continuación una vista del código en Python entorno Jupyter

```python
# División de los datos en conjuntos de entrenamiento (80%) y prueba (20%)
train_size = int(0.8 * len(returns))
train, test = returns[:train_size], returns[train_size:]

# Ajuste del modelo GARCH(1,1) al conjunto de entrenamiento
garch_model = arch_model(train, vol='Garch', p=1, q=1)
garch_fit = garch_model.fit(disp="off")

# Evaluación del modelo GARCH en el conjunto de entrenamiento
aic_garch = garch_fit.aic
bic_garch = garch_fit.bic
log_likelihood_garch = garch_fit.loglikelihood

# Predicciones paso a paso
predictions = []
for j in range(len(test)):
    model = arch_model(np.concatenate([train, test[:j]]), vol='Garch', p=1, q=1)
    model_fit = model.fit(disp="off")
    forecast = model_fit.forecast(horizon=1, reindex=False)
    predictions.append(forecast.variance.values[-1, 0])

predicted_volatility = np.sqrt(predictions)
```

Figure 30: Training and Test

16.3 Evaluación del Modelo

Evaluamos el modelo ajustado utilizando varias métricas estadísticas:

```
aic_garch = garch_fit.aic
bic_garch = garch_fit.bic
log_likelihood_garch = garch_fit.loglikelihood
```

Explicación técnica:

- **AIC (Criterio de Información de Akaike):**

$$AIC = 2k - 2\ln(\hat{L}) \tag{32}$$

  Donde k es el número de parámetros y $\hat{L}$ es el máximo valor de la función de verosimilitud.

- **BIC (Criterio de Información Bayesiano):**

$$BIC = \ln(n)k - 2\ln(\hat{L}) \tag{33}$$

  Donde n es el número de observaciones.

- **Log-verosimilitud:** La suma de los logaritmos de las probabilidades de las observaciones, dado el modelo.

Importancia: Estas métricas nos permiten:

- Comparar diferentes especificaciones de modelos.
- Evaluar el equilibrio entre la bondad del ajuste y la complejidad del modelo.
- Identificar el modelo más parsimonioso que explica adecuadamente los datos.

16.4 Detalle de la programación de las Predicciones

Realizamos predicciones sobre el conjunto de prueba utilizando un enfoque de ventana móvil:

```
for j in range(len(test)):
model = arch_model(np.concatenate([train, test[:j]]), vol='Garch', p=1, q=1)
model_fit = model.fit(disp="off")
forecast = model_fit.forecast(horizon=1, reindex=False)
predictions.append(forecast.variance.values[-1, 0])
```

Explicación técnica: Este método, conocido como "rolling forecast" o "pronóstico móvil", implica:

- Reajustar el modelo en cada paso, incorporando nueva información.
- Realizar una predicción de un paso adelante en cada iteración.
- Ampliar gradualmente el conjunto de entrenamiento con datos del conjunto de prueba.

Importancia: Este enfoque:

- Simula un escenario de predicción en tiempo real.
- Permite que el modelo se adapte a cambios en la dinámica de la volatilidad.
- Proporciona una evaluación más robusta del rendimiento predictivo del modelo.

16.5 Cálculo de la Volatilidad Predicha

Finalmente, calculamos la volatilidad predicha:

```
predicted_volatility = np.sqrt(predictions)
```

Explicación técnica: Tomamos la raíz cuadrada de las varianzas pronosticadas para obtener la desviación estándar. Esto se basa en la relación:

$$\sigma = \sqrt{\sigma^2} \tag{34}$$

Donde σ es la desviación estándar (volatilidad) y σ^2 es la varianza.

Importancia: La volatilidad predicha:

- Es una medida más interpretable de la dispersión de los rendimientos.
- Se expresa en las mismas unidades que los rendimientos originales.
- Es ampliamente utilizada en la gestión de riesgos y la valoración de opciones.

16.6 Evaluación del Modelo GARCH y Recopilación de Resultados

Este segmento de código se centra en dos aspectos cruciales: el cálculo de métricas de error para evaluar el rendimiento del modelo GARCH, y la recopilación organizada de estos resultados junto con otras métricas importantes.

Cálculo de Métricas de Error

```
rmse_garch = np.sqrt(mean_squared_error(test, predicted_volatility))
mse_garch = mean_squared_error(test, predicted_volatility)
mae_garch = mean_absolute_error(test, predicted_volatility)
```

Explicación: Aquí se calculan tres métricas de error ampliamente utilizadas en la evaluación de modelos predictivos:

1. **RMSE (Root Mean Squared Error):**

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{35}$$

Donde y_i son los valores reales y $\hat{y}_i$ son los valores predichos.

2. **MSE (Mean Squared Error):**

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{36}$$

3. **MAE (Mean Absolute Error):**

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{37}$$

Importancia:

- **RMSE:** Proporciona una medida del error promedio, penalizando más los errores grandes debido al término cuadrático.
- **MSE:** Similar al RMSE, pero en unidades cuadradas. Útil para comparaciones matemáticas.
- **MAE:** Ofrece una medida del error promedio en las mismas unidades que los datos originales, siendo menos sensible a valores atípicos que RMSE y MSE.

Estas métricas se calculan comparando los valores reales del conjunto de prueba (`test`) con los valores de volatilidad predichos (`predicted_volatility`).

16.7 Recopilación de Resultados

```
temp_df = pd.DataFrame([{
 'Symbol': symbol,
 'AIC': aic_garch,
 'BIC': bic_garch,
 'Log-Likelihood': log_likelihood_garch,
 'RMSE': rmse_garch,
 'MSE': mse_garch,
 'MAE': mae_garch
}])


metrics_df = pd.concat([metrics_df, temp_df], ignore_index=True)
```

Explicación: Este código crea un DataFrame temporal (`temp_df`) para almacenar los resultados del modelo actual, incluyendo:

- `Symbol`: El identificador del activo financiero analizado.
- `AIC` y `BIC`: Criterios de información para la selección del modelo.
- `Log-Likelihood`: La log-verosimilitud del modelo ajustado.
- `RMSE`, `MSE`, `MAE`: Las métricas de error calculadas anteriormente.

Luego, este DataFrame temporal se concatena con un DataFrame principal (`metrics_df`) que presumiblemente almacena los resultados de múltiples modelos o activos.

Importancia:

- **Organización de datos:** Este enfoque permite una recopilación estructurada y eficiente de los resultados.
- **Comparación fácil:** Facilita la comparación de rendimiento entre diferentes activos o especificaciones de modelos.
- **Análisis posterior:** Prepara los datos para análisis estadísticos adicionales o visualizaciones.

Este proceso de evaluación y recopilación de resultados es crucial para:

- Evaluar cuantitativamente el rendimiento del modelo GARCH.
- Comparar la eficacia del modelo entre diferentes activos financieros.
- Proporcionar una base sólida para conclusiones y recomendaciones en la investigación.

La combinación de métricas de ajuste del modelo (AIC, BIC, Log-Likelihood) con métricas de error predictivo (RMSE, MSE, MAE) ofrece una evaluación integral del modelo GARCH, abarcando tanto su capacidad de ajuste a los datos históricos como su poder predictivo en datos no vistos.

```python
# Cálculo de métricas en el conjunto de prueba
rmse_garch = np.sqrt(mean_squared_error(test, predicted_volatility))
mse_garch = mean_squared_error(test, predicted_volatility)
mae_garch = mean_absolute_error(test, predicted_volatility)

# Crear un DataFrame temporal para la fila actual
temp_df = pd.DataFrame([{
    'Symbol': symbol,
    'AIC': aic_garch,
    'BIC': bic_garch,
    'Log-Likelihood': log_likelihood_garch,
    'RMSE': rmse_garch,
    'MSE': mse_garch,
    'MAE': mae_garch
}])

# Concatenar el DataFrame temporal con el DataFrame principal
metrics_df = pd.concat([metrics_df, temp_df], ignore_index=True)
```

Figure 31: Métricas Conjunto de Prueba

16.8 Visualización de Resultados y Presentación de Métricas

Este segmento de código se encarga de la visualización gráfica de los resultados del modelo GARCH y la presentación organizada de las métricas de evaluación.

16.9 Visualización Gráfica

```
if i % (num_cols * num_rows) == 0:
if i > 0:
plt.tight_layout()
plt.show()
fig, ax = plt.subplots(num_rows, num_cols, figsize=(15, 10))
ax = ax.flatten()
ax[i % (num_cols * num_rows)].plot(returns, label='Returns')
ax[i % (num_cols * num_rows)].plot(garch_fit.conditional_volatility,
label='GARCH Volatility (Train)', color='blue')
ax[i % (num_cols * num_rows)].plot(test.index, predicted_volatility,
label='Predicted Volatility (Test)', color='red')
ax[i % (num_cols * num_rows)].set_title(f'{symbol} GARCH(1,1) Conditional Volatility')
ax[i % (num_cols * num_rows)].legend()
```

Explicación:

- Crea una cuadrícula de subgráficos para múltiples símbolos.
- Grafica los retornos, la volatilidad estimada en el conjunto de entrenamiento y la volatilidad predicha en el conjunto de prueba para cada símbolo.
- Utiliza un sistema de indexación modular para colocar cada gráfico en la posición correcta de la cuadrícula.

16.10 Presentación de Métricas

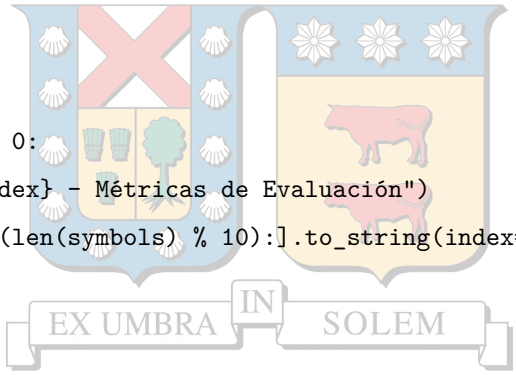
```
if (i + 1) % 10 == 0:
print(f"Tabla {table_index} - Métricas de Evaluación")
print(metrics_df.iloc[-10:].to_string(index=False))
table_index += 1
```

Explicación:

- Imprime una tabla de métricas cada 10 símbolos procesados.
- Utiliza un contador (`table_index`) para numerar las tablas.
- Muestra las últimas 10 filas del DataFrame de métricas.

16.11 Finalización

```
plt.tight_layout()
plt.show()
if len(symbols) % 10 != 0:
print(f"Tabla {table_index} - Métricas de Evaluación")
print(metrics_df.iloc[-(len(symbols) % 10):].to_string(index=False))
```



Explicación:

- Ajusta y muestra la última figura.
- Si el número total de símbolos no es múltiplo de 10, imprime las métricas restantes.

Este código combina visualización gráfica y presentación tabular de resultados, facilitando la interpretación y comparación de los modelos GARCH para múltiples símbolos financieros.

```
    # Graficar los retornos y la volatilidad estimada por el modelo GARCH
    if i % (num_cols * num_rows) == 0:
        if i > 0:
            plt.tight_layout()
            plt.show()
        fig, ax = plt.subplots(num_rows, num_cols, figsize=(15, 10))
        ax = ax.flatten()

    ax[i % (num_cols * num_rows)].plot(returns, label='Returns')
    ax[i % (num_cols * num_rows)].plot(garch_fit.conditional_volatility, label='GARCH Volatility (Train)', color='blue')
    ax[i % (num_cols * num_rows)].plot(test.index, predicted_volatility, label='Predicted Volatility (Test)', color='red')
    ax[i % (num_cols * num_rows)].set_title(f'{symbol} GARCH(1,1) Conditional Volatility')
    ax[i % (num_cols * num_rows)].legend()

    # Imprimir métricas cada 10 símbolos
    if (i + 1) % 10 == 0:
        print(f"Tabla {table_index} - Métricas de Evaluación")
        print(metrics_df.iloc[-10:].to_string(index=False))
        table_index += 1

plt.tight_layout()
plt.show()

# Imprimir las métricas restantes si hay menos de 10 símbolos al final
if len(symbols) % 10 != 0:
    print(f"Tabla {table_index} - Métricas de Evaluación")
    print(metrics_df.iloc[-(len(symbols) % 10):].to_string(index=False))
```

Figure 32: Gráfico de los Retornos y la Volatilidad Estimada por el Modelo GARCH

17 RESULTADOS OBTENIDOS

17.1 Viabilidad en el Rendimiento RNN-LSTM Tabla 1 - 5

- Se observa una considerable variación en el rendimiento de las predicciones entre las diferentes variables vistas en la tabla 1 a 5, evidenciada por las fluctuaciones en las métricas MAE, MSE y RMSE.
- **Rango de Errores**: El MAE varía desde un mínimo de 0.010288 (variable 180) hasta un máximo de 0.388422 (variable 55). El MSE varía desde un mínimo de 0.000175 (variable 180) hasta un máximo de 0.175895 (variable 55). El RMSE varía desde un mínimo de 0.013223 (variable 180) hasta un máximo de 0.419398 (variable 55).
- **Consistencia Entre Métricas**: En general, las variables que presentan un MAE alto también tienen MSE y RMSE altos. Esta consistencia indica que las diferentes métricas de error están alineadas, reflejando de manera coherente el rendimiento del modelo.
- **Outliers**: Algunas variables presentan errores notablemente más altos que el promedio. Por ejemplo, la variable 55 tiene los valores más altos en todas las métricas, sugiriendo que puede haber particularidades en los datos de esta variable que afectan negativamente las predicciones.
- **Rendimiento General**: La mayoría de las variables muestran un MAE inferior a 0.2, un MSE inferior a 0.05, y un RMSE inferior a 0.2. Esto sugiere que, en general, el modelo tiene un rendimiento aceptable, con predicciones razonablemente precisas para la mayoría de las variables.
- **Mejor Rendimiento**: La variable 180 muestra el mejor rendimiento con los valores más bajos en todas las métricas (MAE: 0.010288, MSE: 0.000175, RMSE: 0.013223), indicando una alta precisión en las predicciones para esta variable.
- **Peor Rendimiento**: La variable 55 muestra el peor rendimiento con los valores más altos en todas las métricas (MAE: 0.388422, MSE: 0.175895, RMSE: 0.419398), sugiriendo dificultades significativas en la predicción para esta variable.
- **Distribución de Errores**: La mayoría de las variables tienen un RMSE inferior a 0.2, lo que indica que las predicciones, en promedio, se desvían en menos de 0.2 unidades del valor real, demostrando una buena precisión en términos generales.
- **Variabilidad entre Variables**: La considerable variabilidad en el rendimiento entre las diferentes variables sugiere que algunas variables son más fáciles de predecir que otras. Esto podría deberse a factores específicos de cada variable, como patrones en los datos o la calidad de la información histórica disponible.

  Estas conclusiones proporcionan una visión clara del rendimiento del modelo, destacando tanto sus fortalezas como las áreas donde se podría mejorar.

17.2 Modelo GARCH - Model Fit (AIC, BIC, Log-Likelihood) Tabla 6

- **AIC y BIC**: Valores más bajos indican un mejor ajuste del modelo. BDX tiene los valores de AIC y BIC más bajos (5466.9913 y 5488.5272, respectivamente), lo que sugiere que el modelo se ajusta mejor a estos datos.
- **Log-Likelihood**: Valores más altos (menos negativos) indican un mejor ajuste del modelo. Nuevamente, BDX tiene el Log-Likelihood menos negativo (-2729.4956), lo que confirma un buen ajuste del modelo.

17.3 Errores de Predicción (RMSE, MSE, MAE) Tabla 6

- **RMSE**: Valores más bajos indican predicciones más precisas. BDX tiene el RMSE más bajo (1.9443), seguido de CPB (2.0146), lo que sugiere que las predicciones para estos símbolos son más precisas.
- **MSE**: Similar al RMSE, valores más bajos indican mejores predicciones. BDX y CPB también tienen los MSE más bajos (3.7803 y 4.0587, respectivamente).
- **MAE**: Valores más bajos indican errores absolutos más bajos en las predicciones. CPB tiene el MAE más bajo (1.5949), seguido de BDX (1.6010).

17.4 Comparación General Tabla 6

- Símbolos como LRCX y AMAT tienen los valores de AIC, BIC, RMSE, MSE y MAE más altos, lo que indica que los modelos aplicados a estos datos tienen peores ajustes y predicciones menos precisas en comparación con otros símbolos como BDX y CPB.
- Los valores de RMSE, MSE y MAE son consistentes entre sí en términos de indicar qué modelos tienen mejores predicciones. Los símbolos con RMSE más bajos tienden a tener también los valores de MSE y MAE más bajos.

17.5 Modelo GARCH - Model Fit (AIC, BIC, Log-Likelihood) - Tabla 7

- **AIC y BIC**: Entre los símbolos evaluados, ACN muestra los valores más bajos de AIC (5388.2010) y BIC (5409.7370), indicando un buen ajuste del modelo GARCH para sus datos.
- **Log-Likelihood**: Los valores de Log-Likelihood sugieren un ajuste razonable del modelo para ACN (-2690.1005) y otros como AMGN, GILD, y AAPL, con valores comparativamente altos y menos negativos.

17.6 Errores de Predicción (RMSE, MSE, MAE) - Tabla 7

- **RMSE**: ACN (2.5456), AMGN (1.9029), GILD (1.9776), y AAPL (2.7495) muestran niveles moderados de precisión en las predicciones del modelo GARCH, con RMSE relativamente bajos.
- **MSE**: AMGN (3.6210) y GILD (3.9107) presentan los MSE más bajos entre los símbolos evaluados, indicando predicciones más precisas en términos de error cuadrático medio.
- **MAE**: AMGN (1.5458) y ACN (1.9992) muestran los valores más bajos de MAE, sugiriendo errores absolutos menores en las predicciones del modelo.

17.7 Comparación General - Tabla 7

- ACN, AMGN, GILD, y AAPL muestran métricas favorables en términos de ajuste del modelo y precisión de predicción. Estos símbolos sugieren un buen ajuste del modelo GARCH y predicciones relativamente precisas en comparación con BIIB y NVDA, donde las métricas indican un ajuste menos óptimo y predicciones menos precisas.

17.8 Modelo GARCH - Model Fit (AIC, BIC, Log-Likelihood) Tabla 7

- **AIC y BIC**:
- **Log-Likelihood**:

17.9 Modelo GARCH - Model Fit (AIC, BIC, Log-Likelihood) Tabla 8

- **AIC y BIC UPS**: Tiene un AIC:5706.7713, BIC: 5728.3073, Log-Likelihood: -2849.3857. Estos valores sugieren que el modelo GARCH tiene un ajuste razonable para los datos de UPS, capturando adecuadamente la volatilidad en sus precios.

- **AIC y BIC ADSK**: Tiene un AIC: 7048.0987, BIC: 7069.6346, Log-Likelihood: -3520.0493. Los valores más altos de AIC y BIC indican que el modelo podría no ajustarse tan bien a los datos de ADSK. El Log-Likelihood también muestra un ajuste menos óptimo del modelo para este símbolo.

- **AIC y BIC AMD**: Tiene un AIC: 8879.1913, BIC: 8900.7273, Log-Likelihood: -4435.5957. Estos valores son los más altos entre los símbolos analizados, indicando que el modelo tiene más dificultades para ajustarse a los datos de AMD.

17.10 Errores de Predicción (RMSE, MSE, MAE) Tabla 8

- **UPS**: Presenta un RMSE: 2.6023, MSE: 6.7721, MAE: 2.0232. Estos valores sugieren que las predicciones del modelo para UPS tienen un nivel de precisión moderado, con errores que podrían mejorarse.

- **ADSK**:Presenta un RMSE: 3.7747, MSE: 14.2481, MAE: 3.0114. Los valores más altos de RMSE, MSE y MAE indican que las predicciones del modelo para ADSK tienen un nivel de error más significativo.

- **AMD**: Presenta un RMSE: 5.2063, MSE: 27.1052, MAE: 4.3365. Estos valores son los más altos entre los símbolos analizados, lo que sugiere que las predicciones del modelo para AMD tienen el mayor nivel de error absoluto.

17.11 Comparación General Tabla 8

- UPS, ADP y VZ muestran métricas más favorables en términos de AIC, BIC, Log-Likelihood, RMSE, MSE y MAE. Estos símbolos sugieren que el modelo GARCH aplicado tiene un mejor ajuste y produce predicciones más precisas en comparación con ADSK y AMD, donde las métricas indican un ajuste menos óptimo y predicciones menos precisas.

17.12 Modelo GARCH - Model Fit (AIC, BIC, Log-Likelihood) - Tabla 9

- **AIC y BIC**: Entre los símbolos evaluados, KO muestra los valores más bajos de AIC (4588.4626) y CL los más bajos de BIC (4852.9072), indicando buenos ajustes del modelo GARCH para estos datos.
- **Log-Likelihood**: Los valores de Log-Likelihood sugieren ajustes razonables del modelo para CL (-2422.4536) y otros como BMY, CVX, y CSCO, con valores comparativamente altos y menos negativos.

17.13 Errores de Predicción (RMSE, MSE, MAE) - Tabla 9

- **RMSE**: BMY (1.8719), KO (1.5274), y CL (1.6345) muestran niveles relativamente bajos de RMSE, indicando precisión moderada en las predicciones del modelo GARCH.
- **MSE**: BMY (3.5042), KO (2.3330), y CL (2.6714) presentan los MSE más bajos entre los símbolos evaluados, sugiriendo predicciones más precisas en términos de error cuadrático medio.
- **MAE**: BMY (1.6020), KO (1.2018), y CL (1.2952) muestran los valores más bajos de MAE, sugiriendo errores absolutos menores en las predicciones del modelo.

17.14 Comparación General - Tabla 9

- KO y CL muestran métricas favorables en términos de ajuste del modelo y precisión de predicción, con valores bajos de AIC, BIC, RMSE, MSE y MAE. Estos símbolos sugieren un buen ajuste del modelo GARCH y predicciones relativamente precisas en comparación con otros símbolos evaluados en esta tabla.

17.15 Modelo GARCH - Model Fit (AIC, BIC, Log-Likelihood) - Tabla 10

- **AIC y BIC**: NEE muestra los valores más bajos de AIC (5033.6603) y DUK los más bajos de BIC (4820.5589), indicando buenos ajustes del modelo GARCH para estos datos.
- **Log-Likelihood**: Los valores de Log-Likelihood sugieren ajustes razonables del modelo para DUK (-2406.2794) y otros como CVS, MDT, y NEE, con valores comparativamente altos y menos negativos.

17.16 Errores de Predicción (RMSE, MSE, MAE) - Tabla 10

- **RMSE**: DUK (1.7131), NEE (2.3028), y CVS (2.1593) muestran niveles relativamente bajos de RMSE, indicando precisión moderada en las predicciones del modelo GARCH.
- **MSE**: DUK (2.9348), NEE (5.3027), y CVS (4.6624) presentan los MSE más bajos entre los símbolos evaluados, sugiriendo predicciones más precisas en términos de error cuadrático medio.

- **MAE**: DUK (1.3296), NEE (1.8063), y CVS (1.7402) muestran los valores más bajos de MAE, sugiriendo errores absolutos menores en las predicciones del modelo.

17.17 Comparación General - Tabla 10

- NEE y DUK muestran métricas favorables en términos de ajuste del modelo y precisión de predicción, con valores bajos de AIC, BIC, RMSE, MSE y MAE. Estos símbolos sugieren un buen ajuste del modelo GARCH y predicciones relativamente precisas en comparación con otros símbolos evaluados en esta tabla.

17.18 Modelo GARCH - Model Fit (AIC, BIC, Log-Likelihood) - Tabla 11

- **AIC y BIC**: ETN presenta el valor más bajo de AIC (5832.0921), mientras que GS tiene el valor más bajo de BIC (6138.6958), lo que indica un buen ajuste del modelo GARCH para estos datos.
- **Log-Likelihood**: Los valores de Log-Likelihood sugieren ajustes razonables del modelo para ETN (-2912.0460) y otros como GS, XOM y DD, con valores comparativamente altos y menos negativos.

17.19 Errores de Predicción (RMSE, MSE, MAE) - Tabla 11

- **RMSE**: GS (2.4504), ETN (2.2503), y XOM (2.7532) muestran niveles relativamente bajos de RMSE, lo que indica una precisión moderada en las predicciones del modelo GARCH.
- **MSE**: GS (6.0047), ETN (5.0639), y XOM (7.5803) presentan los valores más bajos de MSE entre los símbolos evaluados, sugiriendo predicciones más precisas en términos de error cuadrático medio.
- **MAE**: GS (2.0409), ETN (1.8414), y XOM (2.2057) muestran los valores más bajos de MAE, sugiriendo errores absolutos menores en las predicciones del modelo.

17.20 Comparación General - Tabla 11

- ETN y GS muestran métricas favorables en términos de ajuste del modelo y precisión de predicción, con valores bajos de AIC, BIC, RMSE, MSE y MAE. Estos símbolos sugieren un buen ajuste del modelo GARCH y predicciones relativamente precisas en comparación con otros símbolos evaluados en esta tabla.

17.21 Modelo GARCH - Model Fit (AIC, BIC, Log-Likelihood) - Tabla 12

- **AIC y BIC**: JNJ presenta el valor más bajo de AIC (4710.9402), mientras que HON tiene el valor más bajo de BIC (5185.1179), lo que indica un buen ajuste del modelo GARCH para estos datos.
- **Log-Likelihood**: Los valores de Log-Likelihood sugieren ajustes razonables del modelo para JNJ (-2351.4701) y otros como HON, JPM y KMB, con valores comparativamente altos y menos negativos.

17.22 Errores de Predicción (RMSE, MSE, MAE) - Tabla 12

- **RMSE**: JNJ (1.4254), HON (1.9987), y KMB (1.7325) muestran niveles relativamente bajos de RMSE, lo que indica una precisión moderada en las predicciones del modelo GARCH.
- **MSE**: JNJ (2.0318), HON (3.9948), y KMB (3.0015) presentan los valores más bajos de MSE entre los símbolos evaluados, sugiriendo predicciones más precisas en términos de error cuadrático medio.
- **MAE**: JNJ (1.1769), HON (1.6212), y KMB (1.3711) muestran los valores más bajos de MAE, sugiriendo errores absolutos menores en las predicciones del modelo.

17.23 Comparación General - Tabla 12

- JNJ y HON muestran métricas favorables en términos de ajuste del modelo y precisión de predicción, con valores bajos de AIC, BIC, RMSE, MSE y MAE. Estos símbolos sugieren un buen ajuste del modelo GARCH y predicciones relativamente precisas en comparación con otros símbolos evaluados en esta tabla.

17.24 Modelo GARCH - Model Fit (AIC, BIC, Log-Likelihood) - Tabla 13

- **AIC y BIC**: MCD presenta el valor más bajo de AIC (4933.3051), mientras que MCD también tiene el valor más bajo de BIC (4954.8410), lo que indica un buen ajuste del modelo GARCH para estos datos.
- **Log-Likelihood**: Los valores de Log-Likelihood sugieren ajustes razonables del modelo para MCD (-2462.6525) y otros como MDLZ, MRK y COST, con valores comparativamente altos y menos negativos.

17.25 Errores de Predicción (RMSE, MSE, MAE) - Tabla 13

- **RMSE**: MCD (1.5691), MDLZ (1.6875), y MRK (1.9006) muestran niveles relativamente bajos de RMSE, lo que indica una precisión moderada en las predicciones del modelo GARCH.
- **MSE**: MCD (2.4620), MDLZ (2.8475), y MRK (3.6123) presentan los valores más bajos de MSE entre los símbolos evaluados, sugiriendo predicciones más precisas en términos de error cuadrático medio.
- **MAE**: MCD (1.2841), MDLZ (1.3520), y MRK (1.4919) muestran los valores más bajos de MAE, sugiriendo errores absolutos menores en las predicciones del modelo.

17.26 Comparación General - Tabla 13

- MCD y MDLZ muestran métricas favorables en términos de ajuste del modelo y precisión de predicción, con valores bajos de AIC, BIC, RMSE, MSE y MAE. Estos símbolos sugieren un buen ajuste del modelo GARCH y predicciones relativamente precisas en comparación con otros símbolos evaluados en esta tabla.

17.27 Modelo GARCH - Model Fit (AIC, BIC, Log-Likelihood) - Tabla 14

- **AIC y BIC**: PEP presenta el valor más bajo de AIC (4459.0115), mientras que PEP también tiene el valor más bajo de BIC (4480.5475), lo que indica un buen ajuste del modelo GARCH para estos datos.
- **Log-Likelihood**: Los valores de Log-Likelihood sugieren ajustes razonables del modelo para PEP (-2225.5058) y otros como PG, RSG y PGR, con valores comparativamente altos y menos negativos.

17.28 Errores de Predicción (RMSE, MSE, MAE) - Tabla 14

- **RMSE**: PEP (1.4903), PG (1.6516), y RSG (1.7523) muestran niveles relativamente bajos de RMSE, lo que indica una precisión moderada en las predicciones del modelo GARCH.
- **MSE**: PEP (2.2209), PG (2.7277), y RSG (3.0705) presentan los valores más bajos de MSE entre los símbolos evaluados, sugiriendo predicciones más precisas en términos de error cuadrático medio.
- **MAE**: PEP (1.1576), PG (1.2671), y RSG (1.3481) muestran los valores más bajos de MAE, sugiriendo errores absolutos menores en las predicciones del modelo.

17.29 Comparación General - Tabla 14

- PEP y PG muestran métricas favorables en términos de ajuste del modelo y precisión de predicción, con valores bajos de AIC, BIC, RMSE, MSE y MAE. Estos símbolos sugieren un buen ajuste del modelo GARCH y predicciones relativamente precisas en comparación con otros símbolos evaluados en esta tabla.

17.30 Modelo GARCH - Model Fit (AIC, BIC, Log-Likelihood) - Tabla 15

- **AIC y BIC**: ALL presenta el valor más bajo de AIC (5195.9802), mientras que ALL también tiene el valor más bajo de BIC (5217.5162), lo que indica un buen ajuste del modelo GARCH para estos datos.
- **Log-Likelihood**: Los valores de Log-Likelihood sugieren ajustes razonables del modelo para ALL (-2593.9901) y otros como SNY, FIS y EIX, con valores comparativamente altos y menos negativos.

17.31 Errores de Predicción (RMSE, MSE, MAE) - Tabla 15

- **RMSE**: SNY (2.1063), EIX (2.1256), y ALL (2.2644) muestran niveles relativamente bajos de RMSE, lo que indica una precisión moderada en las predicciones del modelo GARCH.
- **MSE**: SNY (4.4366), EIX (4.5180), y EMR (4.9948) presentan los valores más bajos de MSE entre los símbolos evaluados, sugiriendo predicciones más precisas en términos de error cuadrático medio.
- **MAE**: SNY (1.6723), EIX (1.7089), y ALL (1.7675) muestran los valores más bajos de MAE, sugiriendo errores absolutos menores en las predicciones del modelo.

17.32 Comparación General - Tabla 15

- ALL y SNY muestran métricas favorables en términos de ajuste del modelo y precisión de predicción, con valores bajos de AIC, BIC, RMSE, MSE y MAE. Estos símbolos sugieren un buen ajuste del modelo GARCH y predicciones relativamente precisas en comparación con otros símbolos evaluados en esta tabla.

18 CONCLUSIONES

Durante la investigación sobre los modelos RNN-LSTM y GARCH, se pudo determinar una serie de resultados interesantes y, a veces, desconcertantes. Al analizar el rendimiento del RNN-LSTM, pudimos ver una amplia gama de errores durante el análisis. pudimos ver que en el modelo RNN-LSTM la variable 180 (del análisis del MAE, MSE, RMSE) mostro muy buen comportamiento en base a los hiperparámetros del programa, esta variable mostró un MAE de apenas 0.010288, mientras que se obtuvo un MAE de 0.388422 para la variable 55, esto se debe a que la cantidad de mercados que se están analizando no todos convergen o tienen relación entre sí.

Esta variabilidad nos hace reflexionar sobre la naturaleza de los datos que se están manejando. Es evidente que algunos patrones eran más fáciles de captar para el modelo, quizás porque tenían tendencias más claras o porque contaba con datos históricos de mejor calidad. Sin embargo, otras variables parecían ser un verdadero rompecabezas para el RNN-LSTM.

En general, el modelo RNN-LSTM entrego buenos resultados y tuvo un buen rendimiento. La mayoría de las variables mostraron un MAE por debajo de 0.2, lo cual es bastante decente. También nos deja tranquilos al ver que las diferentes métricas de error (MAE, MSE, RMSE) eran consistentes entre sí. Al menos el modelo está siendo coherente con las predicciones, aunque no siempre acertara.

Pero, como en toda investigación, siempre hay margen de mejora. Me di cuenta de que necesitaba encontrar una forma de manejar mejor los valores atípicos. Algunas variables tenían errores tan altos que me hicieron dudar si estaba pasando algo por alto. También pensé que quizás necesitaba más datos para algunas variables o tal vez un preprocesamiento más avanzado.

Cuando pasamos al modelo GARCH, obtuvimos impresiones similares. Para algunos símbolos, como BDX y CPB, los resultados obtenidos fueron muy buenos. Los valores bajos de AIC, BIC y Log-Likelihood indican que el modelo se ajusta muy bien a estos datos. Sin embargo, al ver los resultados para AMD y ADSK, pudimos observar que no tienen un buen desempeño. Los altos valores de RMSE, MSE y MAE nos indican que algo no estaba funcionando como espera.

Otros resultados del modelo GARCH como KO (Coca-Cola), PEP (Pepsi) y JNJ (Johnson & Johnson) muestran un rendimiento superior en términos de ajuste del modelo y precisión de predicción, según las métricas AIC, BIC, RMSE, MSE y MAE. En cambio, acciones como TSLA (Tesla) y SSNLF (Samsung) tienen métricas significativamente más altas, indicando que los modelos para estas acciones no son tan precisos o bien ajustados.

Es evidente que las acciones tecnológicas como NVDA (Nvidia) y AMD también tienen métricas relativamente altas, lo que puede sugerir una mayor volatilidad o dificultad para modelar sus comportamientos.

Mientras se analizaban estos resultados, empecé a pensar en cómo podría combinar lo mejor de ambos modelos. ¿Y si se pudiera usar el Modelo GARCH para manejar la volatilidad y luego aplicar el Modelo RNN-LSTM a los datos ajustados? Esta propuesta suena bastante bien, pero sabemos que no es un proceso fácil de implementar.

Por estos motivos podemos determinar diferentes enfoques: quizás podría crear un modelo ensamblado, o tal vez se necesita entrar en una optimización de hiperparámetros más profunda. La idea de una validación cruzada más rigurosa también es una excelente opción.

Podemos ver que la combinación de modelos RNN-LSTM y GARCH no solo es factible, sino que nos entrega nuevas posibilidades para mejorar significativamente las capacidades predictivas de los modelos. La mezcla de las actuales tecnologías en aprendizaje profundo con métodos probados de modelado de volatilidad nos ha llevado a un terreno bastante interesante, donde la complejidad de los patrones secuenciales se enlaza con las sutilezas de la variabilidad de los datos.

El ajuste de los hiperparámetros no ha sido para nada fácil de implementar, teniendo en cuenta que debemos analizar los efectos de memoria en nuestras secuencias y calibrando con precisión los parámetros del GARCH. Este proceso, aunque a veces frustrante, ha sido fundamental para asegurar que nuestro modelo capte fielmente la esencia de los datos que estamos analizando.

Por otra parte, uno de los aspectos más críticos que hemos descubierto es el preprocesamiento de datos, ya que cada técnica de normalización, cada decisión sobre cómo manejar valores atípicos, y cada estrategia para imputar datos faltantes ha tenido un impacto significativo en nuestros resultados. Este proceso de preparación de datos, aunque a menudo subestimado, es de mayor importancia para ambos modelos, esto mejora la precisión y también la capacidad para generalizar y mantener la estabilidad frente a nuevos datos.

Esta investigación nos ha llevado a un punto donde podemos afirmar que la integración de modelos RNN-LSTM y GARCH, respaldada por técnicas avanzadas de preprocesamiento y optimización, tiene el potencial de revolucionar la forma en que abordamos las predicciones de series temporales. No obstante, es fundamental reconocer que este enfoque no es una solución única ya que los modelos son más precisos para algunas acciones de consumo y farmacéuticas, mientras que tienen más dificultades con acciones tecnológicas y de alta volatilidad. Estos modelos requieren un diseño matemático e ingenieril meticuloso, una experimentación constante y una disposición para ajustar y reajustar cada componente del modelo.

19 ANEXOS

19.1 Resultados Tabulados RNN-LSTM

Table 1: Métricas de Evaluación por Variable RNN-LSTM1

Variable	MAE	MSE	RMSE
1	0.087864	0.010891	0.104361
2	0.045845	0.003284	0.057307
3	0.135171	0.023337	0.152765
4	0.125960	0.025891	0.160908
5	0.098917	0.016100	0.126885
6	0.039627	0.002457	0.049564
7	0.100505	0.013894	0.117872
8	0.029242	0.001150	0.033913
9	0.043209	0.002971	0.054504
10	0.028124	0.001243	0.035257
11	0.178782	0.040062	0.200154
12	0.064100	0.005347	0.073123
13	0.072376	0.008896	0.094320
14	0.087428	0.008728	0.093423
15	0.045685	0.003112	0.055785
16	0.030796	0.001458	0.038184
17	0.141498	0.029327	0.171253
18	0.042009	0.002756	0.052494
19	0.084237	0.010981	0.104789
20	0.058485	0.005025	0.070889

Table 2: Métricas de Evaluación por Variable RNN-LSTM2

Variable	MAE	MSE	RMSE
21	0.135109	0.022125	0.148745
22	0.046559	0.003364	0.057998
23	0.101323	0.015188	0.123240
24	0.034115	0.002994	0.054714
25	0.148562	0.028686	0.169369
26	0.147595	0.023985	0.154871
27	0.043693	0.003358	0.057950
28	0.082180	0.014211	0.119209
29	0.121254	0.023765	0.154158
30	0.059723	0.005718	0.075618
31	0.090003	0.013083	0.114380
32	0.104972	0.014053	0.118544
33	0.091689	0.013085	0.114391
34	0.073241	0.008688	0.093207
35	0.095253	0.013669	0.116915
36	0.078997	0.008764	0.093614
37	0.126607	0.022738	0.150792
38	0.077370	0.008288	0.091040
39	0.207235	0.073380	0.270887
40	0.056928	0.004891	0.069938

Table 3: Métricas de Evaluación por Variable RNN-LSTM3

Variable	MAE	MSE	RMSE
41	0.142549	0.028400	0.168524
42	0.049078	0.004845	0.069608
43	0.036173	0.002071	0.045509
44	0.053276	0.003945	0.062809
45	0.159895	0.031086	0.176312
46	0.034772	0.002123	0.046078
47	0.066459	0.006287	0.079292
48	0.047447	0.003147	0.056102
49	0.088299	0.011787	0.108566
50	0.125513	0.019360	0.139142
51	0.098472	0.013405	0.115780
52	0.061444	0.006980	0.083544
53	0.224921	0.066022	0.256948
54	0.040202	0.010834	0.104085
55	0.388422	0.175895	0.419398
56	0.066639	0.010029	0.100146
57	0.187832	0.044995	0.212121
58	0.091341	0.011107	0.105388
59	0.211233	0.051156	0.226176
60	0.095360	0.015931	0.126217

Table 4: Métricas de Evaluación por Variable RNN-LSTM4

Variable	MAE	MSE	RMSE
61	0.139586	0.037388	0.193360
62	0.042656	0.002622	0.051203
63	0.055834	0.005514	0.074256
64	0.099699	0.012304	0.110925
65	0.135040	0.023982	0.154862
66	0.023921	0.001057	0.032507
67	0.146710	0.026180	0.161802
68	0.084802	0.010802	0.103934
69	0.192900	0.050376	0.224445
70	0.105623	0.014005	0.118344
71	0.160062	0.030357	0.174234
72	0.301336	0.099892	0.316058
73	0.133528	0.023829	0.154367
74	0.118482	0.020187	0.142080
75	0.173300	0.038438	0.196056
76	0.209475	0.051836	0.227674
77	0.095343	0.012484	0.111732
78	0.085202	0.013752	0.117269
79	0.072456	0.008143	0.090244
80	0.036173	0.003013	0.054884

Table 5: Métricas de Evaluación por Variable RNN-LSTM5

81	0.047821	0.003844	0.062011
82	0.084031	0.010763	0.103743
83	0.112308	0.016200	0.127292
84	0.032738	0.002607	0.051057
85	0.043398	0.002791	0.052835
86	0.038178	0.003117	0.055846
87	0.035562	0.002446	0.049460
88	0.066202	0.009144	0.095602
89	0.064601	0.009157	0.095682
90	0.045716	0.003420	0.058465
91	0.035841	0.002373	0.048719
92	0.090159	0.012012	0.109599
93	0.060245	0.004688	0.068466
94	0.106499	0.017052	0.130659
95	0.097605	0.015337	0.123843
96	0.110737	0.018518	0.136074

19.2 Resultados Tabulados Modelo GARCH

Table 6: Métricas de Evaluación por Variable Model GARCH

Symbol	AIC	BIC	Log-Likelihood	RMSE	MSE	MAE
ABT	5600.657339	5622.193297	-2796.328670	2.159115	4.661776	1.712729
ADM	5787.654724	5809.190682	-2889.827362	2.317662	5.371559	1.798372
LRCX	7179.720781	7201.256739	-3585.860391	4.088285	16.714076	3.303766
AMAT	7156.016507	7177.552465	-3574.008254	3.944062	15.555629	3.184874
DHR	6585.566775	6607.102733	-3288.783388	2.989942	8.939752	2.443792
DE	6318.628950	6340.164908	-3155.314475	2.750071	7.562891	2.178628
DAL	6870.453747	6891.989705	-3431.226873	3.813355	14.541675	3.098507
ECL	5188.591154	5210.127112	-2590.295577	2.660377	7.077608	2.104958
CPB	5823.053128	5844.589086	-2907.526564	2.014626	4.058719	1.594924
BDX	5466.991286	5488.527244	-2729.495643	1.944293	3.780277	1.600999

Table 7: Métricas de Evaluación por Variable Model GARCH

Symbol	AIC	BIC	Log-Likelihood	RMSE	MSE	MAE
BBY	7255.685659	7277.221617	-3623.842830	3.619360	13.099765	2.884406
BIIB	7706.126741	7727.662699	-3849.063371	4.755238	22.612289	3.332893
GILD	6142.320852	6163.856809	-3067.160426	1.977562	3.910753	1.613657
ACN	5388.197680	5409.733638	-2690.098840	2.545608	6.480119	1.999217
AMGN	5922.289302	5943.825259	-2957.144651	1.902896	3.621012	1.545792
AAPL	6188.246951	6209.782909	-3090.123476	2.749492	7.559707	2.183792
MSFT	5888.562594	5910.098552	-2940.281297	2.672535	7.142442	2.103207
GOOGL	5985.326772	6006.862730	-2988.663386	2.914142	8.492225	2.335959
AMZN	6438.727431	6460.263388	-3215.363715	3.790719	14.369553	2.989196
NVDA	7759.368992	7780.904950	-3875.684496	4.919732	24.203762	4.024112

Table 8: Métricas de Evaluación por Variable Model GARCH

Symbol	AIC	BIC	Log-Likelihood	RMSE	MSE	MAE
UPS	5706.7713	5728.3073	-2849.3857	2.6023	6.7721	2.0232
ADSK	7048.0987	7069.6346	-3520.0493	3.7747	14.2481	3.0114
ADP	5475.8529	5497.3888	-2733.9264	2.1075	4.4414	1.6632
AXP	5894.2575	5915.7935	-2943.1287	2.9529	8.7199	2.4097
AMD	8879.1913	8900.7273	-4435.5957	5.2063	27.1052	4.3365
ADBE	6200.2883	6221.8243	-3096.1442	3.6369	13.2269	2.7628
MMM	5461.3941	5482.9300	-2726.6970	2.1694	4.7064	1.7434
VZ	4835.9526	4857.4886	-2413.9763	1.7350	3.0101	1.3614
BAC	6354.5838	6376.1197	-3173.2919	2.6567	7.0579	2.2125
BK	5976.3127	5997.8487	-2984.1564	2.5565	6.5356	2.0824

Table 9: Métricas de Evaluación por Variable Model GARCH

Symbol	AIC	BIC	Log-Likelihood	RMSE	MSE	MAE
AVGO	6921.6041	6943.1401	-3456.8021	2.8940	8.3751	2.3716
BA	6576.1719	6597.7079	-3284.0860	3.6901	13.6171	3.0117
BMY	6084.4084	6105.9444	-3038.2042	1.8719	3.5042	1.6020
COF	6483.3900	6504.9260	-3237.6950	3.4200	11.6967	2.7356
CAT	6394.1348	6415.6708	-3193.0674	2.6685	7.1211	2.1972
CVX	5902.2923	5923.8282	-2947.1461	2.4924	6.2118	1.9658
CSCO	5764.1713	5785.7072	-2878.0856	2.3291	5.4248	1.8421
C	6303.2922	6324.8281	-3147.6461	2.7248	7.4244	2.2413
KO	4588.4626	4609.9986	-2290.2313	1.5274	2.3330	1.2018
CL	4852.9072	4874.4432	-2422.4536	1.6345	2.6714	1.2952

Table 10: Métricas de Evaluación por Variable Model GARCH

Symbol	AIC	BIC	Log-Likelihood	RMSE	MSE	MAE
CMCSA	5657.1271	5678.6630	-2824.5635	2.4929	6.2147	1.9281
ABBV	6311.5895	6333.1254	-3151.7947	1.9834	3.9337	1.5893
CNC	7036.3874	7057.9234	-3514.1937	2.8256	7.9839	2.3415
SCHW	6708.9706	6730.5065	-3350.4853	2.9330	8.6023	2.3989
COP	7050.6444	7072.1804	-3521.3222	3.3366	11.1326	2.6970
CVS	5987.3520	6008.8880	-2989.6760	2.1593	4.6624	1.7402
NEE	5033.6603	5055.1962	-2512.8301	2.3028	5.3027	1.8063
DIS	5764.0825	5785.6185	-2878.0413	2.9507	8.7068	2.3953
MDT	5320.3161	5341.8521	-2656.1581	2.1688	4.7038	1.7341
DUK	4820.5589	4842.0948	-2406.2794	1.7131	2.9348	1.3296

Table 11: Métricas de Evaluación por Variable Model GARCH

Symbol	AIC	BIC	Log-Likelihood	RMSE	MSE	MAE
ETN	5832.0921	5853.6280	-2912.0460	2.2503	5.0639	1.8414
DD	6336.1732	6357.7092	-3164.0866	2.7688	7.6663	2.2714
XOM	5551.6960	5573.2320	-2771.8480	2.7532	7.5803	2.2057
GE	6610.3271	6631.8630	-3301.1635	3.0247	9.1490	2.4883
FDX	6293.6807	6315.2166	-3142.8403	3.3451	11.1897	2.6245
F	6446.9005	6468.4365	-3219.4503	4.0018	16.0144	3.2889
GM	6580.0047	6601.5407	-3286.0024	3.6776	13.5250	3.0155
GS	6117.1598	6138.6958	-3054.5799	2.4504	6.0047	2.0409
HAL	7382.2132	7403.7491	-3687.1066	4.0938	16.7595	3.3098
HOG	7165.9804	7187.5164	-3578.9902	3.8217	14.6055	3.1314

Table 12: Métricas de Evaluación por Variable Model GARCH

Symbol	AIC	BIC	Log-Likelihood	RMSE	MSE	MAE
HON	5163.5819	5185.1179	-2577.7910	1.9987	3.9948	1.6212
INTC	6485.0049	6506.5409	-3238.5025	3.0970	9.5917	2.5009
INTU	6002.5413	6024.0773	-2997.2707	3.6837	13.5696	2.8611
JPM	5746.4031	5767.9391	-2869.2016	2.3671	5.6030	1.9329
JNJ	4710.9402	4732.4761	-2351.4701	1.4254	2.0318	1.1769
JCI	5891.7410	5913.2770	-2941.8705	2.4596	6.0498	1.9788
KMB	5169.6432	5191.1792	-2580.8216	1.7325	3.0015	1.3711
TSLA	8303.8441	8325.3801	-4147.9221	5.3438	28.5565	4.2621
LNC	6998.7548	7020.2907	-3495.3774	4.4870	20.1331	3.3230
LMT	5126.9833	5148.5192	-2559.4916	2.0149	4.0600	1.5346

Table 13: Métricas de Evaluación por Variable Model GARCH

Symbol	AIC	BIC	Log-Likelihood	RMSE	MSE	MAE
LOW	6085.0424	6106.5783	-3038.5212	2.6253	6.8920	2.0620
MA	5690.6318	5712.1677	-2841.3159	2.7480	7.5517	2.2087
MCD	4933.3051	4954.8410	-2462.6525	1.5691	2.4620	1.2841
MRK	5346.6269	5368.1629	-2669.3135	1.9006	3.6123	1.4919
ORCL	5599.2605	5620.7965	-2795.6303	2.6237	6.8837	2.0208
MDLZ	5335.5849	5357.1209	-2663.7925	1.6875	2.8475	1.3520
COST	5151.1584	5172.6944	-2571.5792	2.3284	5.4215	1.7495
MS	6404.9263	6426.4623	-3198.4632	2.6471	7.0073	2.1504
NFLX	7692.9811	7714.5171	-3842.4906	4.9095	24.1033	3.7095
NKE	6021.5285	6043.0645	-3006.7643	3.1588	9.9783	2.4679

Table 14: Métricas de Evaluación por Variable Model GARCH

Symbol	AIC	BIC	Log-Likelihood	RMSE	MSE	MAE
PEP	4459.0115	4480.5475	-2225.5058	1.4903	2.2209	1.1576
PFE	5097.0790	5118.6150	-2544.5395	2.4620	6.0615	1.9846
PG	4603.4688	4625.0048	-2297.7344	1.6516	2.7277	1.2671
PGR	5400.8382	5422.3742	-2696.4191	2.0055	4.0220	1.6223
PRU	6338.4354	6359.9714	-3165.2177	2.4882	6.1912	2.0479
QCOM	6986.4662	7008.0022	-3489.2331	3.6604	13.3983	2.9473
RTX	5524.4560	5545.9919	-2758.2280	2.1105	4.4544	1.7448
RSG	4549.7277	4571.2637	-2270.8639	1.7523	3.0705	1.3481
CRM	6572.3126	6593.8486	-3282.1563	3.8598	14.8984	3.0644
SSNLF	14801.3443	14822.8803	-7396.6722	29.4431	866.8946	25.2120

Table 15: Métricas de Evaluación por Variable Model GARCH

Symbol	AIC	BIC	Log-Likelihood	RMSE	MSE	MAE
SNY	5537.5453	5559.0812	-2764.7726	2.1063	4.4366	1.6723
SLB	6703.6840	6725.2199	-3347.8420	3.9986	15.9891	3.2742
ALL	5195.9802	5217.5162	-2593.9901	2.2644	5.1276	1.7675
FAST	6239.4187	6260.9547	-3115.7093	2.2914	5.2505	1.8488
FIS	5496.2679	5517.8039	-2744.1340	3.5492	12.5970	2.6494
EA	6607.7200	6629.2559	-3299.8600	2.4376	5.9420	2.0134
EIX	5875.5758	5897.1118	-2933.7879	2.1256	4.5180	1.7089
EMR	5907.4836	5929.0196	-2949.7418	2.2349	4.9948	1.8160
EOG	7205.7877	7227.3237	-3598.8939	3.6842	13.5736	3.0375
TGT	6405.3177	6426.8537	-3198.6589	3.5007	12.2552	2.5223

19.3 Gráficos de Conclusiones RNN-LSTM

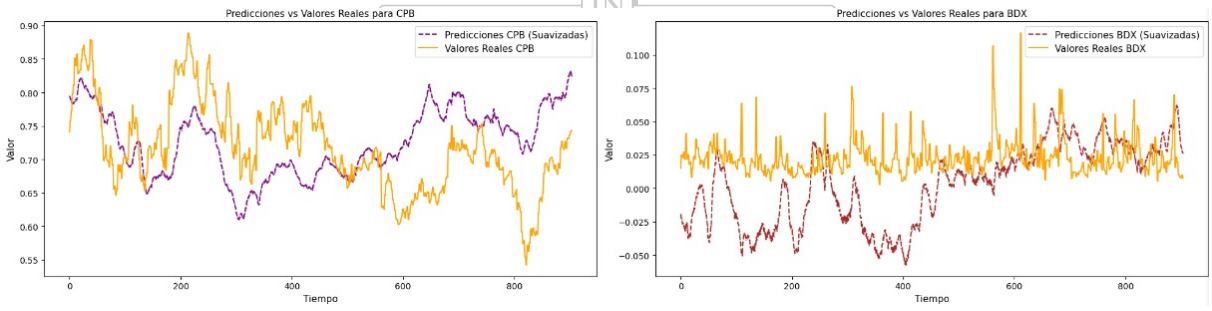


Figure 33: BDX y CPB

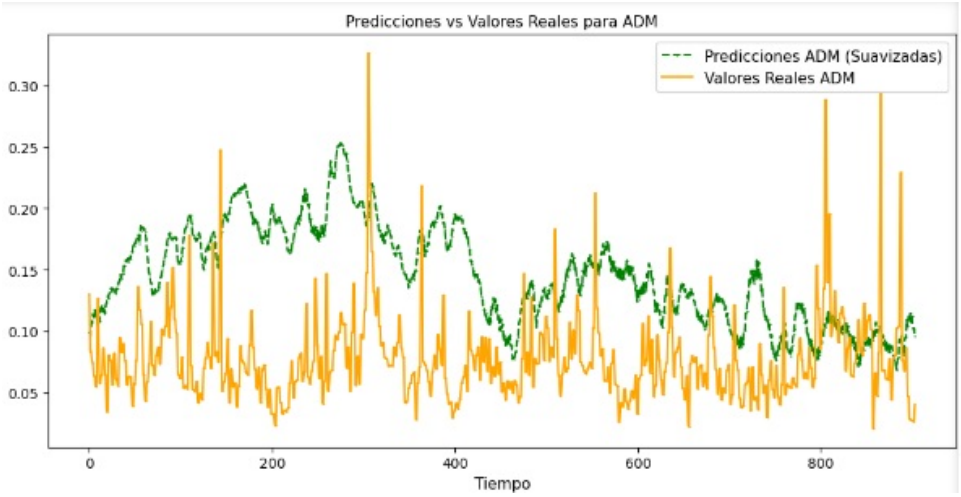


Figure 34: AMD

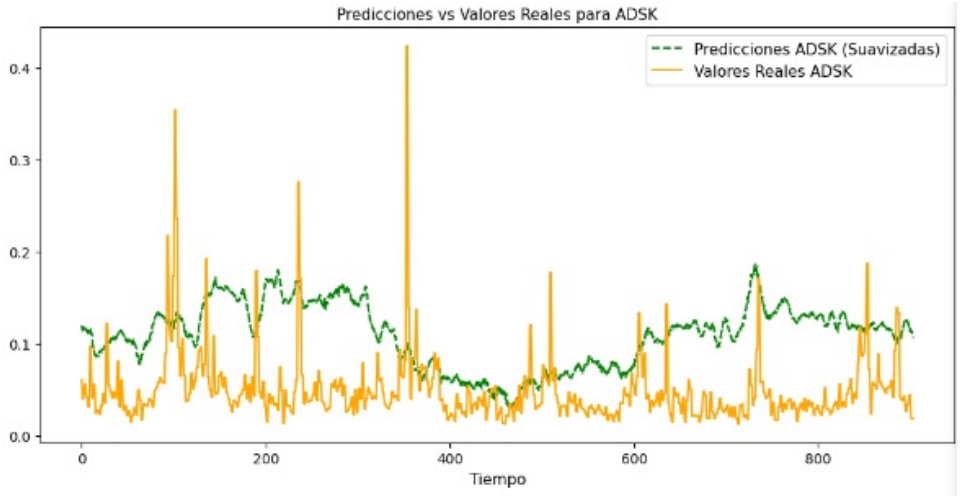


Figure 35: ADSK

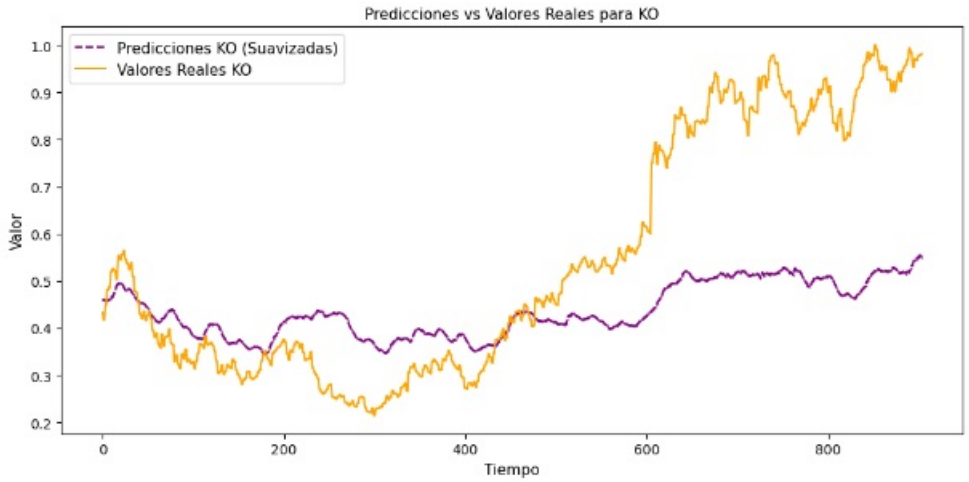


Figure 36: KO

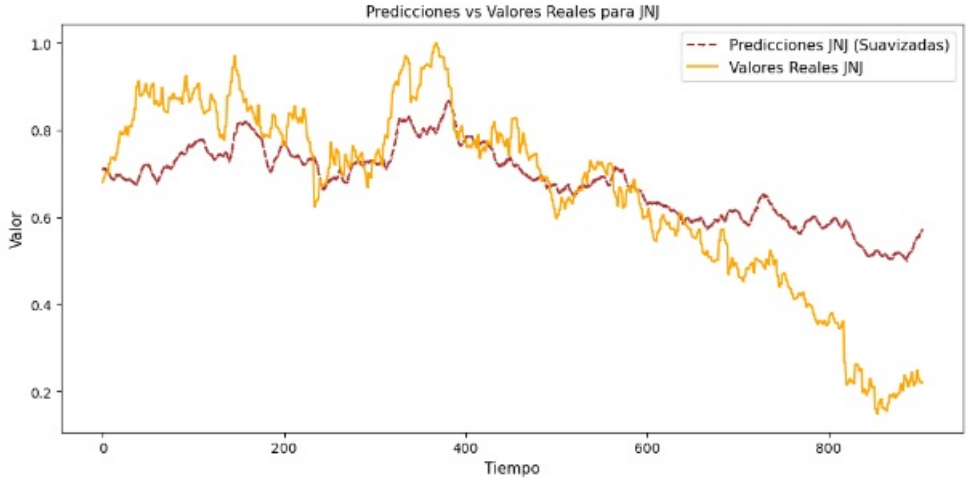


Figure 37: JNJ

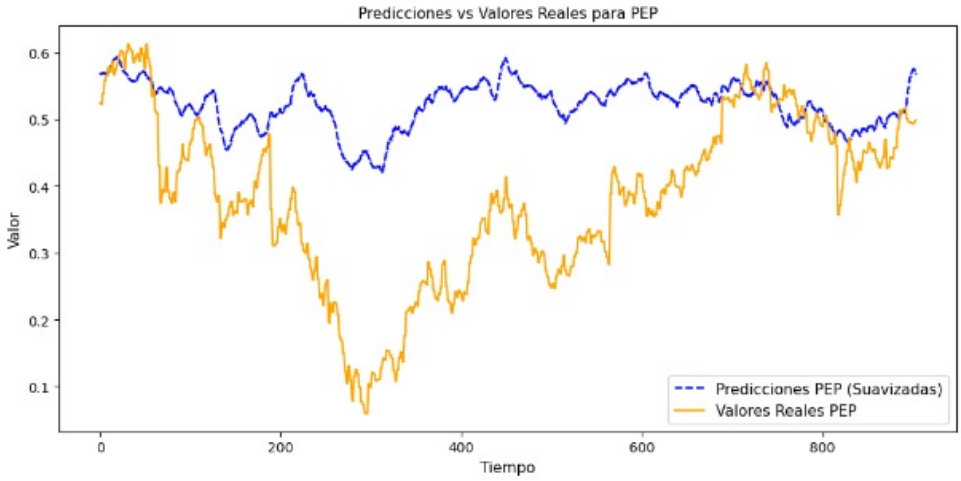


Figure 38: JNJ

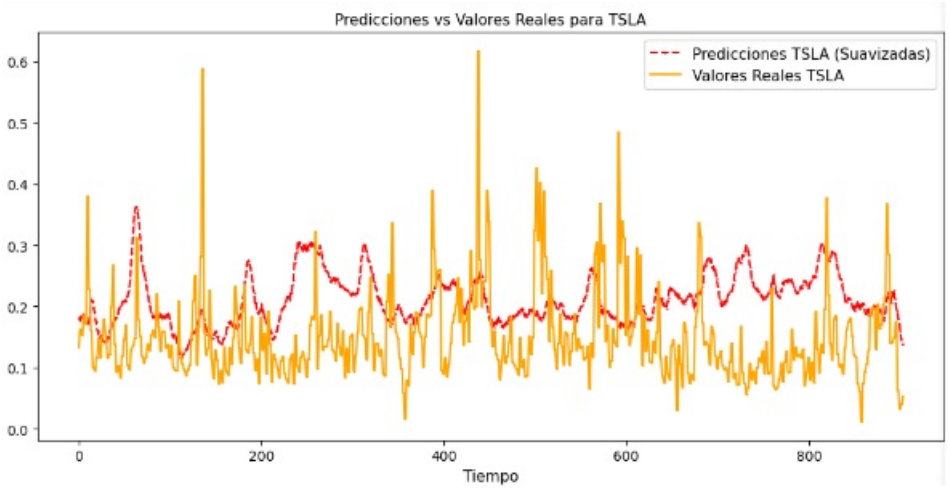


Figure 39: TSLA

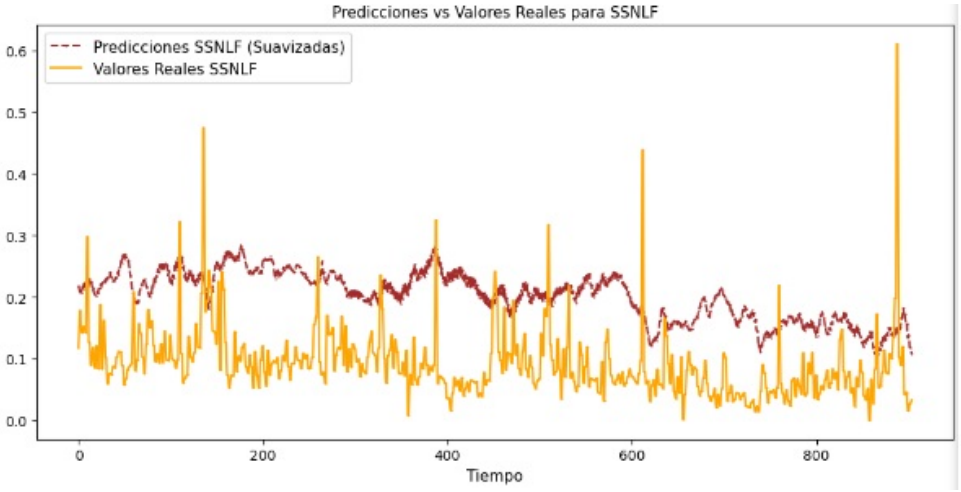


Figure 40: SSNLF

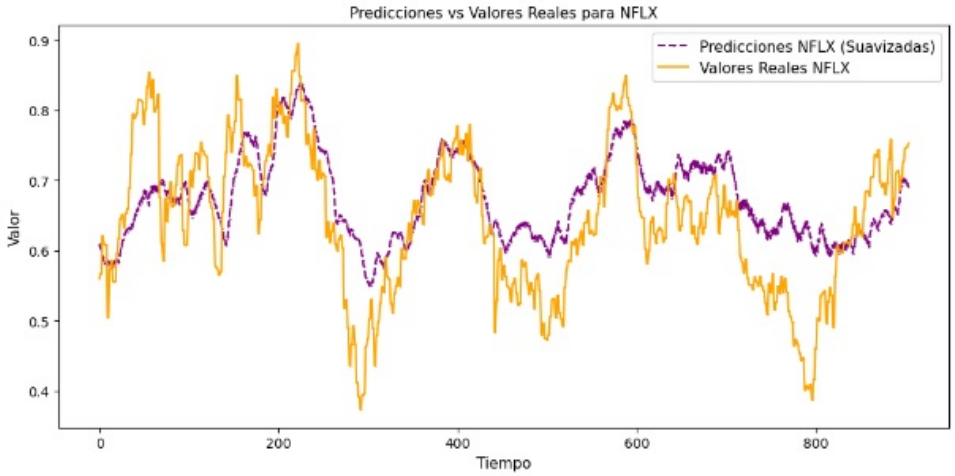


Figure 41: NFLX

19.4 Gráfico de conclusiones Model GARCH

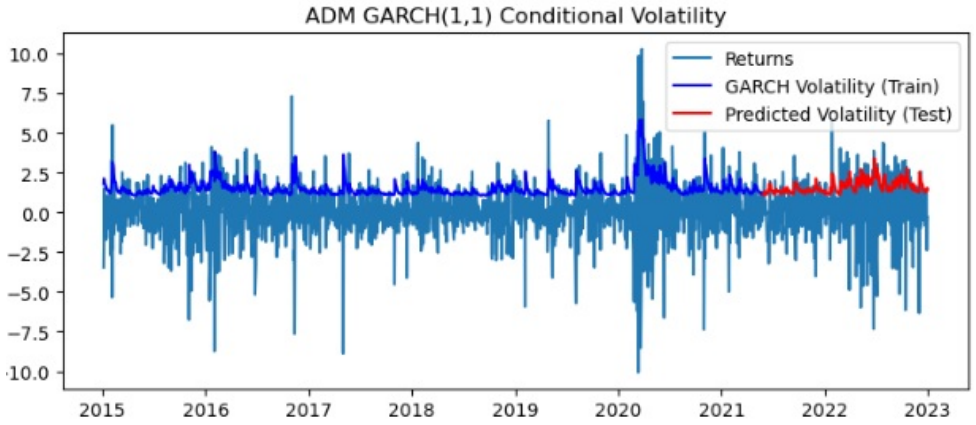


Figure 42: AMD

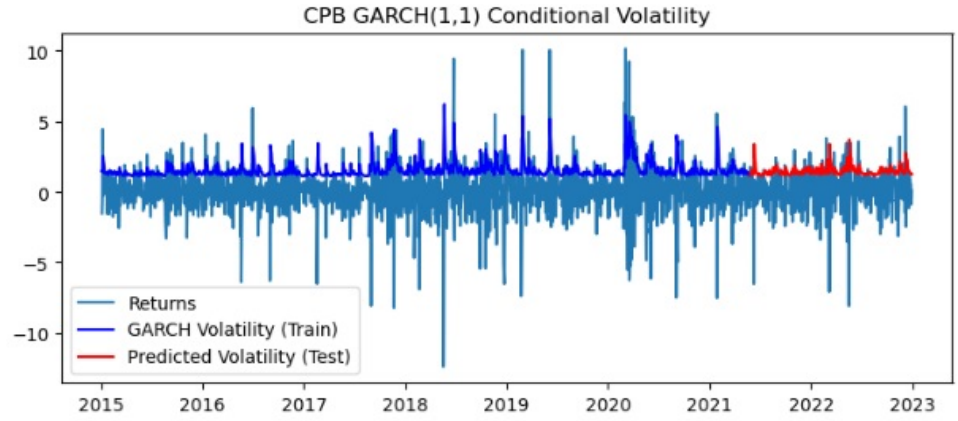


Figure 43: CPB

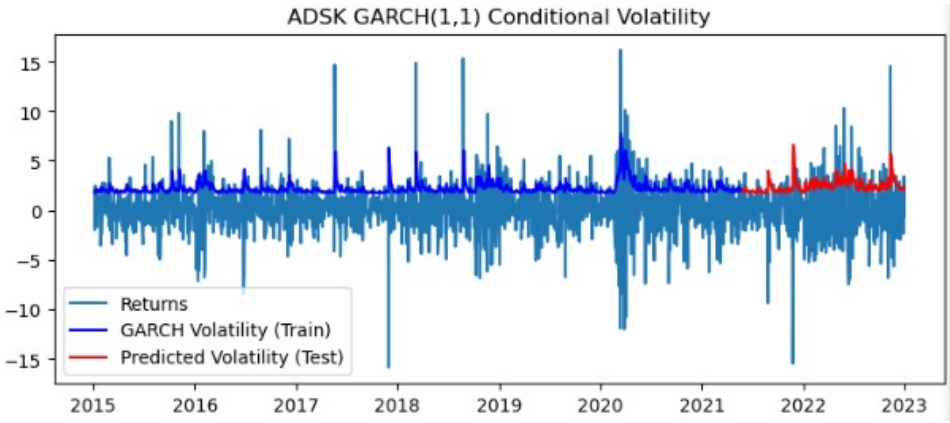


Figure 44: ADSK

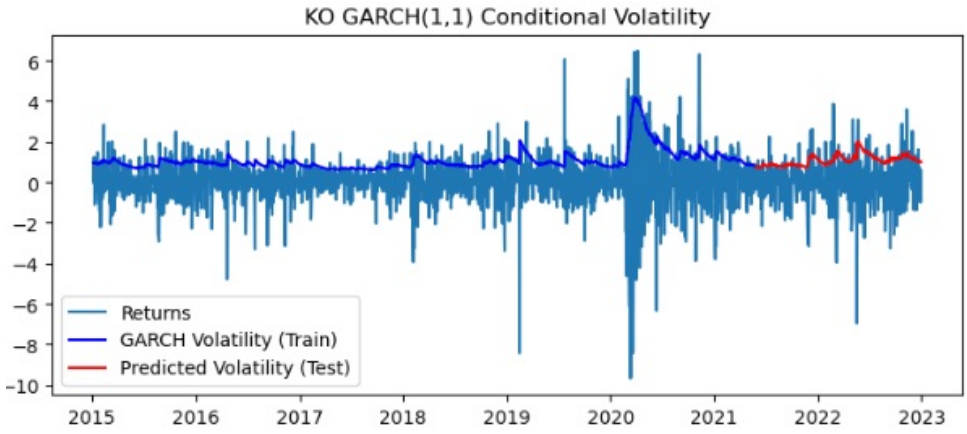


Figure 45: KO

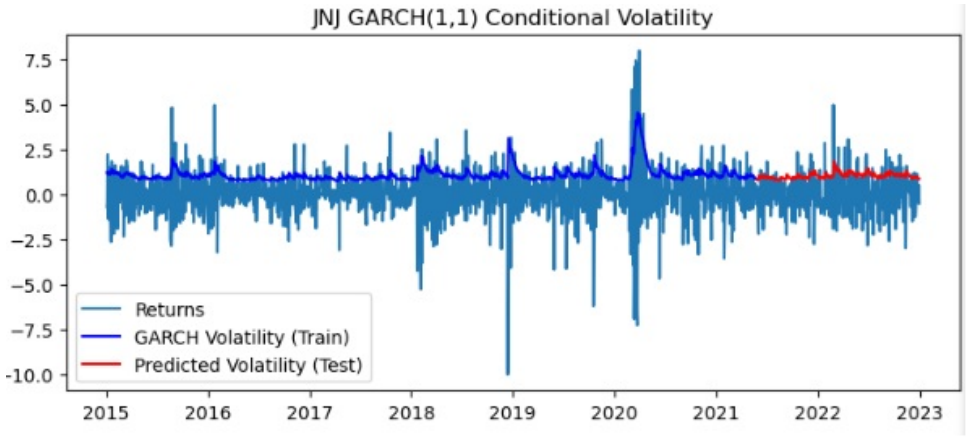


Figure 46: JNJ

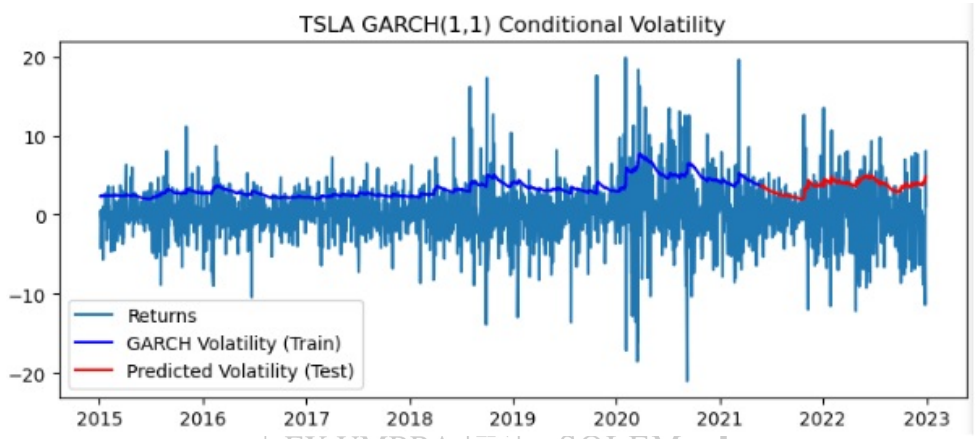


Figure 47: TSLA

20 REFERENCIAS

References

[]McNeil, A. J., Frey, R., & Embrechts, P. (2015). *Quantitative risk management: Concepts, techniques and tools-revised edition*. Princeton university press.

[]Rossi, G. D. (2013). La volatilidad en mercados financieros y de commodities: Un repaso de sus causas y la evidencia reciente. *Invenio*, *16*(30), 59–74.

[]Streb, J. M. (2001). Political uncertainty and economic underdevelopment.

[]Mankiw, N., & Espáriz, E. (2014). *Macroeconomía, 8ª ed.* Antoni Bosch. https://books.google.cl/books?id=78zUAgAAQBAJ

[]Conti, D., Simó, C., & Rodríguez, A. (2005). Teoría de carteras de inversión para la diversificación del riesgo: Enfoque clásico y uso de redes neuronales artificiales (rna). *Ciencia e Ingeniería*, *26*(1), 35–42.

[]Eismann, E. (2018). Markowitz vs black–litterman: A comparison of two portfolio optimisation models.

[]Fama, E. F. (1970). Session topic: Stock market price behavior session chairman: Burton g. malkiel efficient capital markets: A review of theory and empirical work. *The journal of Finance*, *25*(2), 383–417.

[]Markowitz, H. M. (1991). Foundations of portfolio theory. *The journal of finance*, *46*(2), 469–477.

[]Bollerslev, T., Engle, R. F., & Wooldridge, J. M. (1988). A capital asset pricing model with time-varying covariances. *Journal of political Economy*, *96*(1), 116–131.

[]Kearney, C., & Patton, A. J. (2000). Multivariate garch modeling of exchange rate volatility transmission in the european monetary system. *Financial Review*, *35*(1), 29–48.

[]Engle, R. F. (2009). High dimension dynamic correlations. *The Methodology and Practice of Econometrics: Festscrift for David Hendry', Oxford University Press, USA*, 122–148.

[]Aielli, G. P. (2013). Dynamic conditional correlation: On properties and estimation. *Journal of Business & Economic Statistics*, *31*(3), 282–299.

[]McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*, 115–133.

[]Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536.

[]Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, *2*(5), 359–366.