Comparación de metodologías basadas en una red neuronal artificial y un modelo GARCH para el pronóstico de la volatilidad del precio de las acciones cotizados en la BVC

Walter David Lopera Hernández

Universidad Nacional de Colombia
Facultad de Ciencias, Escuela de Estadística - Maestría en Ciencias - Estadística
Medellín, Colombia
2023

Comparación de metodologías basadas en una red neuronal artificial y un modelo GARCH para el pronóstico de la volatilidad del precio de las acciones cotizados en la BVC

Walter David Lopera Hernández

Trabajo de grado presentado como requisito parcial para optar al título de:
Magister en Ciencias - Estadística

Director:
Ph.D. en Matemáticas César Augusto Gómez Vélez

Línea de Investigación:
Métodos estadísticos aplicados a finanzas.
Grupo de Investigación en Estadística de la Universidad Nacional de Colombia, sede Medellín.

Universidad Nacional de Colombia
Facultad de Ciencias, Escuela de Estadística - Maestría en Ciencias - Estadística
Medellín, Colombia
2023

Cada mañana nacemos de nuevo. Lo que más importa es lo que hacemos hoy.

Buda

Declaración

Me permito afirmar que he realizado la presente tesis de manera autónoma y con la única ayuda de los medios permitidos y no diferentes a los mencionados en la propia tesis. Todos los pasajes que se han tomado de manera textual o figurativa de textos publicados y no publicados, los he reconocido en el presente trabajo. Ninguna parte del presente trabajo se ha empleado en ningún otro tipo de tesis.

Medellín, 30.06.2023

Walter David Lopera Hernández

Agradecimientos

A todas esas personas que me acompañaron hasta el final de la realización de este sueño, gracias.

Resumen

Comparación de metodologías basadas en una red neuronal artificial y un modelo GARCH para el pronóstico de la volatilidad del precio de las acciones cotizados en la BVC.

La volatilidad y de manera más general la covarianza de los precios de las acciones es de gran interés para medir los posibles riesgos que pueda tener la compra y venta de éstas, además de proporcionar datos para determinar su rentabilidad. La escasa literatura sobre la implementación de modelos DCC-MGARCH (dynamic conditional correlation GARCH Multivariate) y RN-LSTM (long - short term memory recurrent network) en el mercado financiero colombiano para el pronóstico de la volatilidad de las acciones, llevó a la realización de este trabajo, en el que se comparan estos dos modelos utilizando los softwares R (R Core Team, 1990) y Python (Van Rossum & Drake Jr, 2009), para el pronóstico de la volatilidad del precio de cuatro acciones. Con los datos suministrados se estimaron las volatilidades y covarianzas para luego realizar pronósticos con ambas herramientas y realizar una comparación entre ambas. Del estudio se encontró que el desempeño de ambas metodologías tienen gran similitud, aunque algunas medidas del error de los pronósticos fueron levemente mejor con la RN-LSTM y otras con el modelo DCC-GARCH. Con el objetivo de profundizar más en los análisis de estas volatilidades es adecuado incrementar la cantidad de activos para conocer la incidencia que pueden llegar tener cada uno de éstas en los cálculos de las covarianzas.

Palabras clave: volatilidad, retorno, covarianza, red neuronal, matriz, retropropagación.

Abstract

Comparasion of methodologies based in an artificial neural network and a GARCH model for the forecast of the stock price volatility listed on the BVC.

The volatility and more generally the covariance of share prices is of great interest to measure the possible risks that the purchase and sale of these may have, in addition to providing data to determine their profitability. The scant literature on the implementation of DCC-MGARCH (dynamic conditional correlation GARCH Multivariate) and RN-LSTM (long - short term memory recurrent network) models in the Colombian financial market for stock volatility forecasting led to the realization of this work, in which these two models are compared using R (R Core Team, 1990) and Python (Van Rossum & Drake Jr, 2009) software for forecasting the price volatility of four shares. With a total of 488 data, the volatilities and covariances were estimated to then make a forecast with both tools and make a comparison between them. The study found that the performance of both methodologies is somewhat similar, although some forecast error measures were slightly better with the RN-LSTM and others with the DCC-GARCH model. It is appropriate to establish a broader range of dates to determine the influence of other factors and increase the amount of assets to know the incidence that each one of these has on the others.

Keywords: volatility, return, covariance, neural network, matrix, backpropagation

Contenido

Agradecimientos vi

Resumen vii

Lista de tablas xi

Lista de figuras xii

Lista de abreviaturas xiii

1. Introducción 1

2. Marco teórico 5
2.1. Volatilidad financiera 5
2.2. Modelos ARCH 6
2.3. Modelos GARCH 7
2.4. Modelos MGARCH 8
2.5. Máxima verosimilitud 11

3. Redes neuronales artificiales 13
3.1. Proceso computacional de una RN 18
3.2. Red neuronal recurrente 20
3.3. Red neuronal recurrente long short - term memory 22
3.4. Computación para los análisis 25

4. Enfoque metodológico 27
4.1. Base de datos 27
4.2. El modelo DCC-MGARCH 30
4.3. RN-LSTM con Python 31

5. Resultados 33

6. Conclusiones y recomendaciones 45
6.1. Conclusiones 45
6.2. Trabajo futuro 46

A. Anexo: Código requerido para obtener los resultados del modelo DCC-MGARCH en R. 47

B. Anexo: Código requerido para obtener los resultados del modelo de la RN-LSTM en Python. 60

Bibliografía 82

Lista de tablas

5-1. Estadísticas descriptivas de los retornos para los cuatro activos en el periodo (2018/03/01 - 2019/12/30). 33
5-2. Parámetros óptimos obtenidos con el modelo DCC-MGARCH para los cuatro activos. 35
5-3. VaR (Valor en Riesgo) - ES (Déficit esperado) para los retornos de los cuatro activos. 36
5-4. Máximos, mínimos y rangos para los pronósticos de las varianzas - covarianzas con ambas metodologías DCC-MGARCH en R (R Core Team, 1990) y RN-LSTM en Python (Van Rossum & Drake Jr, 2009). 37
5-5. Error medio porcentual absoluto (MAPE) de los pronósticos para los cuatro activos con ambas metodologías DCC-MGARCH en R (R Core Team, 1990) y RN-LSTM en Python (Van Rossum & Drake Jr, 2009). 38
5-6. Error cuadrático medio (RMSE) de los pronósticos para los cuatro activos con ambas metodologías DCC-MGARCH en R (R Core Team, 1990) y RN-LSTM en Python (Van Rossum & Drake Jr, 2009) 39
5-7. Desviación absoluta media (MAD) de los pronósticos para los cuatro activos con ambas metodologías DCC-MGARCH en R (R Core Team, 1990) y RN-LSTM en Python (Van Rossum y Drake Jr, 2009). 40

Lista de figuras

3-1. Neurona (elaboración propia). 14
3-2. Neurona con bias o unidad correctora de sesgo (elaboración propia). 15
3-3. Funciones de activación (elaboración propia en Python (Van Rossum & Drake Jr, 2009)). 17
3-4. Red neuronal o perceptrón de una sola capa. Fuente: Gareth, Witten, Hastie & Tibshirani, (2021). 18
3-5. Red neuronal multicapa. Fuente: Gareth, Witten, Hastie & Tibshirani, (2021). 19
3-6. Red neuronal recurrente. Fuente: James, Witten, Hastie & Tibshirani (2021). 21
3-7. RN-LSTM. Fuente: Ha Yung Kim & Chang Hyun Won, (2018). 22

4-1. Precio de las acciones acciones entre los periodos 2018/01/03 - 2019/12/30. . 29

5-1. Rendimientos de los retornos entre 2018/01/03 - 2019/12/30. 34
5-2. Gráficas de covarianzas para los cuatro activos con R (R Core Team, 1990) comprendidas entre las fechas 2018/01/02 - 2019/12/30. 40
5-3. Gráficas covarianzas con Python (Van Rossum & Drake Jr, 2009) comprendidas entre las fechas 2018/01/02 - 2019/12/30. 41
5-4. Gráficas de los errores de pronósticos de las covarianzas con el modelo DCC-MGARCH comprendidas entre las fechas 2019/08/05 - 2019/12/30 42
5-5. Gráficas de los errores de pronósticos de las covarianzas con el modelo RN-LSTM comprendidas entre las fechas 2019/08/05 - 2019/12/30 43

Lista de abreviaturas

Abreviatura	**Término**
BVC	Bolsa de valores de Colombia
$ARCH$	Autorregresivo condicional heterocedástico
$GARCH$	Autorregresivo condicional heterocedástico generalizado
RN	Red neuronal
$LSTM$	Memoria a corto y largo plazo
CFE	Suma acumulada de los errores de los pronósticos
$MAPE$	Porcentaje absoluto medio
MAD	Desviación absoluta media
$RMSE$	Desviación de la raíz cuadrada media
CCC	correlación condicional constante
DCC	correlación condicional dinámica
IA	Inteligencia artificial
RNN	Redes neuronales recurrentes

1. Introducción

En economía, las acciones forman parte del capital social de una empresa constituida, son susceptibles de comercializar y se constituyen como elementos de valor estratégico utilizado por las empresas para capitalizar y financiar sus proyectos. La venta y compra de estas acciones son realizadas por un intermediario conocido como bolsa de valores, que tiene como fin el intercambio de acciones de forma pública, donde el comprador o titular de dichas acciones se conoce como accionista, el cual adquiere la titularidad directa de éstas.

La bolsa de valores de Colombia (BVC) es el organismo intermediario en la venta y compra de mercados de acciones; estas transacciones según Morales Castro (2016), están sujetas a las dinámicas globales del mercado, el producto interno bruto -PIB- las monedas, las tasas de interés, la ley de oferta-demanda, entre otros, por lo tanto el valor monetario fluctúa continuamente.

La variación o cambio de tendencia del precio (alto/bajo) de las acciones en el mercado, se relaciona con el concepto de volatilidad que es de gran interés para los inversionistas; para Márquez Cebrián (2002), la volatilidad es una variable que mide la intensidad de la aleatoriedad o la fuerza de los cambios en los precios y/o en el rendimiento de los títulos financieros. En términos estadísticos la volatilidad es representada por medio de la varianza (segundo momento), el cual es un parámetro de los modelos de series temporales, y que puede ser utilizado de manera condicional o incondicional en una serie de tiempo.

El concepto de volatilidad se expresa en porcentaje, es un indicador para los inversionistas y, tienen como objetivo conocer y medir los posibles riesgos en las inversiones de las acciones tales como, el VaR (Value At Risk) que mide la máxima pérdida posible de una inversión en un determinado periodo de tiempo. Con la información del VaR se puede determinar si una inversión en determinadas acciones son idóneas para la compra, la venta y en efecto la obtención de mayores rentabilidades (Muñoz & Torres, 2014).

Dicho lo anterior, el cálculo de la volatilidad adquiere una gran relevancia para los inversionistas en acciones, pues es un factor clave para la toma de decisiones financieras. En consecuencia, el interés por parte de académicos y diversas instituciones se enfoca en el desarrollo de modelos estadísticos para el cálculo de la volatilidad de las acciones, entre los que se encuentran: ARCH y GARCH con algunas de sus extensiones como el MGARCH,

EGARCH, TGARCH entre otros; el desarrollo de estos modelos puede ser resuelto por medio de herramientas computacionales como R (R Core Team, 1990), Python (Van Rossum & Drake Jr, 2009) y otros.

Un modelo muy utilizado para el cálculo de la volatilidad en series financieras son los ARCH (autoregressive conditionally heteroscedastic) por sus siglas en inglés, propuestos por Engle (2014), y que se caracterizan por presentar varianzas condicionales no constantes y que dependen del cuadrado de las perturbaciones pasadas. También están los modelos GARCH (generalized autoregressive conditionally heteroscedastic) propuesto por Bollerslev (1986), cuya varianza condicional depende de los cuadrados de las perturbaciones (como en el modelo ARCH) y de las varianzas condicionales de periodos anteriores.

GARCH es una herramienta utilizada para modelar la volatilidad de una sola variable, la implementación de este modelo tiene como resultado el valor de la varianza de una serie financiera, pero cuando se desea modelar un conjunto de variables financieras se debe utilizar MGARCH (o GARCH multivariado). El desarrollo de este modelo estima simultáneamente una matriz de covarianzas y otra de correlaciones para las variables de interés y, para el desarrollo de estas matrices obtenidas en los modelos MGARCH se emplean los métodos BEKK, CCC, DCC, DVEC entre otros (McNeil, Frey & Embrechets, 2015).

Los modelos MGARCH han sido de gran interés en diversas investigaciones, por ejemplo, Nor Syahilla, Spyridon Vrontos & Haslifah (2019), evalúan los modelos MGARCH de varios activos para estrategias de optimización en la selección de carteras que involucra modelos DCC-GARCH y modelos basados en copula-GARCH. Dhaene & Wu (2020), proponen modelos MGARCH de tipo BEKK de frecuencia mixta para pronosticar la volatilidad de baja frecuencia basada en rendimientos intradiarios de alta frecuencia y en los rendimientos nocturnos.

Es por lo anterior que este trabajo tiene como objetivo pronosticar la volatilidad del precio de cuatro acciones cotizadas en la BVC por medio de un modelo multivariado DCC-GARCH y utilizando la herramienta computacional R (R Core Team, 1990). Así mismo, se aplicará en conjunto una red neuronal artificial (RN) utilizando la herramienta computacional Python (Van Rossum & Drake Jr, 2009) para pronosticar las volatilidades de dichas acciones. Con la información obtenida, se busca realizar una comparación entre ambas metodologías y con ello analizar los resultados más apropiados a los intereses particulares de los accionistas.

Para realizar el pronóstico de las volatilidades de las acciones seleccionadas por ambas metodologías se tomarán datos diarios de la semana laboral con la excepción de sábados, domingos, festivos y algunos días especialemente particulares como lo son navidad y año nuevo; los datos fueron descargados de la página BVC en el siguiente rango de fechas, 2 de enero de 2018

hasta 30 de diciembre de 2019 para un total de 488 datos. Se optó por este rango de fechas con el fin de evitar que los datos fueran afectados en lo posible por algunas variables externas, tal como sucedió en el contexto de la coyuntura sanitaria mundial del COVID -19, elecciones presidenciales entre otros. Con el total de datos se calcularon los rendimientos por lo que se pasó de 488 a 487 datos, con estos rendimientos se calcularon las varianzas-covarianzas para los cuatro activos, estos resultados se dividieron en dos grupos, uno de 387 datos para realizar los pronósticos y otro grupo con 100 datos como mecanismo de comparación con los pronósticos.

Se usaron las herramientas computacionales R (R Core Team, 1990) y Python (Van Rossum & Drake Jr, 2009) para realizar los pronósticos y las comparaciones de las volatilidades del modelo DCC-MGARCH con la red neuronal, sin embargo, se encontraron las siguientes limitaciones, en la implementación del modelo DCC-MGARCH la herramienta R (R Core Team, 1990) contaba con las librerías adecuadas para el cálculo de dicha modelación mas no para Python (Van Rossum & Drake Jr, 2009), por lo que se procedió a realizar los pronósticos con R (R Core Team, 1990). Con el fin de hacer la comparación del resultado DCC-MGARCH con la red neuronal se implementó el modelo LSTM (Long Short-Term Memory) para el pronóstico de las volatilidades. R (R Core Team, 1990) no contaba con las librerías para implementar esta red neuronal por lo que se procedió a implementarlo con Python (Van Rossum & Drake Jr, 2009), con esto se utilizaron dos herramientas computacionales para el cálculo y comparación de las volatilidades.

La metodología implementada se llevó a cabo de la siguiente manera, con los precios de cierre diario de las acciones se hallaron los rendimientos para un total de 487 datos, estos rendimientos se cargaron en R (R Core Team, 1990) y se les calcularon las matrices de covarianzas o volatilidades de estas acciones; este grupo de covarianzas se dividieron en dos grupos, el primero con un total de 387 datos comprendidos entre las fechas 03/01/2018 hasta 02/08/2019 con el objetivo de realizar los pronósticos y el segundo grupo con 100 datos en el rango de fechas 03/08/2019 hasta 30/12/2019 para realizar las comparaciones con los datos pronosticados; las 487 matrices de covarianzas iniciales calculadas con R (R Core Team, 1990) se descargaron en forma de vectores y se cargaron en Python (Van Rossum & Drake Jr, 2009) para realizar de nuevo los pronósticos de las covarianzas, allí de nuevo se dividió los datos en los dos mismos grupos, cada uno con las mismas cantidades de datos para realizar los pronósticos y las respectivas comparaciones.

Para evaluar los resultados de los pronósticos con ambas metodologías se utilizó el error porcentual absoluto medio (MAPE) que mide el tamaño del error absoluto medio en términos porcentuales, el error cuadrático medio (RMSE) que mide la cantidad de error que hubo entre los pronósticos y valores de comparación o reales y la desviación absoluta media (MAD) la cual calcula la media de las desviaciones absolutas; luego del cálculo de estos criterios de

evaluación con R (R Core Team, 1990) y Python (Van Rossum & Drake Jr, 2009) se les realiza una comparación con el objetivo de detallar que metodología calcula un rango de estos criterios más angosto o más preciso.

Finalmente, el análisis con la herramienta MGARCH aprovechado por practicantes del área de econometría financiera pretende aportar un estudio práctico, para el cálculo de medidas de riesgo y la gestión de riesgo de portafolios en general.

La implementación del modelo DCC en el modelo MGARCH tiene como objetivo un cálculo menor de parámetros, a diferencia de los modelos BEKK y DVEC; otro objetivo de estos modelos DCC, es el análisis de las correlaciones dinámicas de las variables en el tiempo, a diferencia de los modelos CCC que no tienen en cuenta el cambio de las correlaciones con el paso del tiempo.

Así mismo, este trabajo también busca ampliar el conocimiento y contribuir a la escasa literatura sobre la implementación de estos modelos MGARCH y las RN-LSTM en el mercado financiero colombiano para el pronóstico de los rendimientos de las acciones.

2. Marco teórico

El tema central de este trabajo tienen relación con el modelamiento y pronóstico de la volatilidad de activos financieros. Para entender lo que es la volatilidad financiera se debe conocer primero cómo se calcula el retorno financiero de un activo en una serie de tiempo. Según McNeil, Frey & Embrechts (2015) sea $X_1, ..., X_n$ una serie de retornos definida como la diferenciación logarítmica de una serie de precios, índices o tipos de cambio (S_t), entonces :

$$X_t = \ln\left(\frac{S_t}{S_{t-1}}\right), t = 1, ..., n \tag{2-1}$$

2.1. Volatilidad financiera

Según Quintero Valencia (2019), se puede concebir la volatilidad de un activo financiero como una medida de la fluctuación o amplitud del retorno de un activo con respecto a su media en un periodo de tiempo; esto tiene gran importancia en el mercado de acciones ya que logra medir el riesgo de la rentabilidad expuesto por los inversionistas, es decir, a mayor volatilidad de un activo mayor riesgo. Lo que hace la volatilidad es cuantificar la variabilidad o dispersión de un activo con respecto a su tendencia central. Para Yao, Zhao & Li (2022), la precisión de la medición de la volatilidad determina la precisión y credibilidad de las medidas de riesgo y fijación de los precios de los activos. Por lo tanto, el análisis y el pronóstico de la volatilidad de los activos se convierte en una actividad fundamental para la economía de un país, de una empresa y de los inversionistas, por ejemplo, Giraldo Picón (2022), "desarrolla un proceso de modelamiento de volatilidad para interpretar la volatilidad en tiempo real de activos financieros de renta variable del índice colombiano MSCI COLCAP"(p. 6).

Según McNeil, Frey & Embrechts (2015) , la volatilidad puede ser modelada como la desviación estándar condicional de los retornos financieros, esto quiere decir que la desviación estándar está sujeta a los cambios que ocurren en el tiempo; algunas veces ésta desviación estándar es alta por largos periodos de tiempo por lo que se puede decir que el activo tiene mucha volatilidad y cuando la desviación estándar es baja en otro periodo largo de tiempo,

se dice que tiene menor volatilidad, para Stock & Watson (2012), este comportamiento se puede representar en agrupamientos de la volatilidad. "La desviación estándar se define como la raíz cuadrada de la varianza de una población o de una variable aleatoria que la representa, se denota como σ y se expresa de la siguiente manera"(Ruiz Espejo, 2017, pp. 37-38) :

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2}, \tag{2-2}$$

donde y_i es la variable de interés y n es el tamaño de la población.

Según Stock & Watson (2012), existen modelos que estudian los agrupamientos de la volatilidad condicional llamados heterocedásticos, tales como el modelo de heterocedasticidad condicional autorregresiva (ARCH) y su extensión, el modelo ARCH generalizado (GARCH). Dada la importancia que han tenido estos modelos en la investigación de académicos, universidades y empresas, se revisaron diversos artículos como lo fueron Márquez Cebrían (2002), Nor Syahilla & Vrontos & Haslifah (2019), Wu & Dhaene (2020), Yao & Zhao & Li (2022), Cheong (2009), Rojas & Palacios (2004), Carrillo Ríoz (2017), Sánchez V. & Reyes M. (2006), Guo (2022), Zagst & Gollart & Escobar Anel (2022), que hacen mención a los resultados obtenidos en el uso de estos modelos para analizar y pronosticar la volatilidad de diversos activos.

2.2. Modelos ARCH

Un modelo autorregresivo de heterocedasticidad condicional de orden p, denotado como ARCH(p) (autoregressive conditionally heteroscedastic) presentado originalmente en Engle (1982) se especifica de la siguiente forma:

$$X_t = \sigma_t \epsilon_t, \tag{2-3}$$

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i X_{t-i}^2, \tag{2-4}$$

donde $\alpha_0 > 0$ y $\alpha_i \geq 0$, $i = 1, ..., p$; ϵ_t es una sucesión de variables i.i.d. con distribución normal $N(0, 1)$, α_0 y α_i son coeficientes desconocidos, los cuales deben ser positivos como

condición de estacionaridad. Numerosas investigaciones se han realizado en torno al uso de este modelo, una de ellas muy común es para el pronóstico del precio del crudo, ya que la volatilidad de este activo influye en otros indicadores macroeconómicos. Por ejemplo, Cheong (2009), investigó los agrupamientos de la volatilidad que tienen los mercados de petróleo crudo WTI y Brent, por medio de un modelo ARCH, y cómo algunas noticias y otros aspectos impactaron esta volatilidad.

Rojas & Palacios (2004), realizaron un análisis por medio de un modelo ARCH para pronosticar la volatilidad de las cotizaciones de las acciones de la empresa minera Atacocha, utilizando la serie de datos observados desde 1992 a 2003, concluyendo que los modelos ARCH capturan la heterocedasticidad presente en la serie de retornos financieros lo que permite estimar la volatilidad (riesgo) de estas series.

Carrillo Ríoz (2017), analizó la volatilidad de los precios de acciones del Banco Pichincha y Ecuindex que se negocian en la bolsa de valores de Quito en un periodo de 2005 a 2017 por medio de un modelo ARCH que generó resultados estadísticamente significativos en su coeficiente beta para modelar los retornos, concluyendo que la varianza condicional es importante para determinar el rendimiento esperado del precio de las acciones. Finalmente se concluyó que la serie dejó de ser heterocedástica para pasar a ser homocedástica por lo que no hay efecto ARCH debido al mercado emergente ecuatoriano.

2.3. Modelos GARCH

Los modelos ARCH descritos con frecuencia requieren de una estructura de errores considerablemente larga para la varianza condicional; con el fin de aliviar esta deficiencia, se concibió el modelo GARCH propuesto por Bollerslev (1986), en el cual su forma general de representarlo es GARCH(p, q), donde p indica las observaciones más recientes y q es la estimación más reciente de la varianza. Este modelo permite capturar la memoria larga y se caracteriza por tener una estructura de errores más flexible (Sanchez V. & Reyes M., 2006). La forma en la que se puede especificar un modelo GARCH se muestra a continuación:

$$X_t = \sigma_t \epsilon_t, \tag{2-5}$$

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i X_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2 \tag{2-6}$$

Al igual que en el modelo ARCH, ϵ_t en este modelo GARCH es una sucesión de variables i.i.d. con una distribución normal $N(0,1)$; $\alpha_0 > 0$, $\alpha_i \geq 0$, $i = 1, ..., p$ y $\beta_j \geq 0$, $j = 1, ..., q$. Los modelos GARCH propuestos por Bollerslev (1986), constituyen los modelos ARCH generalizados donde el cuadrado de la volatilidad en el instante t, dependen del cuadrado de las volatilidades en instantes anteriores $t - j$, al igual que de los valores previos al cuadrado X^2_{t-i}. El modelo más popular es el GARCH(1, 1) donde se indica que la tasa de variación se basa en la observación más reciente de la variable de mercado y la estimación más reciente de la varianza (Hull, 2012).

Entre las investigaciones que se han realizado con estos modelos GARCH para una sola variable se encuentra Guo (2022), quien investigó la gestión del riesgo cuantitativo de los futuros del Bitcoin mediante el uso de modelos GARCH normal y GARCH normal inversa, comparándolo con una estimación de VaR, concluyendo así que el modelo GARCH normal inversa podría generar estimaciones más precisas que VaR para la serie de rendimientos futuros del Bitcoin. Zagst, Gollart & Escobar Anel (2022), utilizó un modelo GARCH para optimizar una cartera de forma cerrada para un activo, basados en la metodología de Campbell & Viceira (1999), para obtener una solución aproximada y el modelo HN-GARCH para probar la convergencia.

2.4. Modelos MGARCH

Los modelos GARCH son utilizados también para múltiples variables, esta extensión del GARCH univariado fue presentada por Bollerslev (1986). "El modelo GARCH multivariado fue propuesto como una solución al problema de que muchas variables económicas y financieras reaccionan a la misma información y por lo tanto tienen covarianzas distintas de cero" (Sáchez V. & Reyes M, 2006, p. 153).

GARCH multivariado o MGARCH no sólo da información de la varianza como en GARCH, sino que también brinda información sobre las covarianzas y correlaciones de las variables de interés. Para plantear las ecuaciones de los modelos MGARCH se debe tener en cuenta la desomposición de Cholesky, el cual es una forma de descomponer una matriz simétrica definida positiva como el producto de una matriz triangular inferior y la transpuesta de esta matriz de la siguiente manera, $AA^\top = \Sigma$, donde A es la matriz triangular inferior y $A^\top$ es la transpuesta. Para el caso MGARCH, $\Sigma^{1/2}$ representa la desomposición de Cholesky de la matriz de covarianza. Igual que en los modelos ARCH y GARCH "sea $(\epsilon_t)_{t\in\mathbb{Z}}$ una distribución normal con vector de medias 0 y matriz de varianzas covarianzas la matriz identidad I_d, $(\mathbf{0}, \mathbf{I}_d)$, donde d es la dimensión de la diagonal de la matriz, entonces el proceso $(\mathbf{X}_t)_{t\in\mathbb{Z}}$ es un MGARCH si es estrictamente estacionario y satisface la ecuación de la siguiente forma: (McNeil, Frey & Embrechts, 2015, p. 545):

$$\mathbf{X}_t = (\Sigma)_t^{1/2} \epsilon_t, \tag{2-7}$$

donde $\mathbf{\Sigma_t^{1/2}} \in \mathbb{R}^{d\times d}$, siendo $\mathbf{\Sigma_t}$ la matriz condicional de los valores de σ^2 que corresponden a las volatilidades de las variables, o sea, los elementos de $\mathbf{\Sigma_t}$ son $\sigma_{t,ij}$ y deben ser positivos" McNeil, Frey & Embrechts (2015). $\mathbf{\Sigma_t}$ se puede reescribir de la siguiente manera, $\mathbf{\Sigma_t} = \Delta_t P_t \Delta_t$, donde:

$$\Delta_t = \Delta\left(\Sigma\right)_t = diag(\sigma_{t,1}, ..., \sigma_{t,d}), P_t = \wp\left(\Sigma\right)_t. \tag{2-8}$$

Según McNeil, Frey & Embrechts (2015), la matriz diagonal Δ_t se conoce como la matriz de volatilidad y $\wp(\mathbf{\Sigma}) := (\Delta(\mathbf{\Sigma}))^{-1}\mathbf{\Sigma}(\Delta(\mathbf{\Sigma}))^{-1}$ es utilizada para extraer la matriz de correlación P_t de la matriz de covarianzas. A continuación se presentan algunos modelos que permiten abordar las matrices antes mencionadas; uno de estos es el modelo de correlación condicional constante (CCC) propuesto por Bollerslev (1986); McNeil, Frey & Embrechts (2015) plantean la matriz de covarianzas de la siguiente forma $\mathbf{\Sigma_t} = \Delta_t P_c \Delta_t$, en la cual P_c es la matriz positiva definida de correlación y Δ_t es la matriz de volatilidad diagonal que satisface:

$$\sigma_{t,k}^2 = \alpha_{k0} + \sum_{i=1}^{pk} \alpha_{ki} X_{t-i,k}^2 + \sum_{j=1}^{qk} \beta_{kj} \sigma_{t-j,k}^2, k = 1, ..., d, \tag{2-9}$$

donde $\alpha_{k0} > 0$, $\alpha_{ki} \geq 0$, $i = 1, ..., pk$, $\beta_{kj} \geq 0$, $j = 1, ..., qk$, este modelo así definido asegura que Σ_t es positiva. Además es estacionario en covarianza si y solo si $\sum_{i=1}^{pk} \alpha_{ki} + \sum_{j=1}^{qk} \beta_{kj} < 1$ según McNeil, Frey & Embrechts (2015); lo anterior puede ser expresado de la siguiente manera:

$$\sum_t^{CCC} = \begin{bmatrix} \sigma_{t,11}^2 & \sigma_{t,12} & ... & \sigma_{t,1q} \\ \sigma_{t,21} & \sigma_{t,2}^2 & ... & \sigma_{t,2q} \\ \vdots & \vdots & ... & \vdots \\ \sigma_{t,p1} & \sigma_{t,p2} & ... & \sigma_{t,pq}^2 \end{bmatrix} = \begin{bmatrix} \sigma_{t,1} & 0 & ... & 0 \\ 0 & \sigma_{t,2} & ... & 0 \\ \vdots & \vdots & ... & \vdots \\ 0 & 0 & ... & \sigma_{t,p} \end{bmatrix} \times \begin{bmatrix} 1 & \rho_{12} & ... & \rho_{1q} \\ \rho_{21} & 1 & ... & \rho_{2q} \\ \vdots & \vdots & ... & \vdots \\ \rho_{p1} & \rho_{p2} & ... & 1 \end{bmatrix} \times$$

$$\begin{bmatrix} \sigma_{t,1} & 0 & ... & 0 \\ 0 & \sigma_{t,2} & ... & 0 \\ \vdots & \vdots & ... & \vdots \\ 0 & 0 & ... & \sigma_{t,q} \end{bmatrix}.$$

Otro de los modelos utilizados para resolver este tipo de situaciones son los modelos de correlación condicional dinámica (DCC) propuesto por Engle (1982), en el cual las correlaciones condicionales evolucionan dinámicamente en el tiempo y su matriz $(P_t)_{t\in\mathbb{Z}}$ satisface la siguiente ecuación:

$$P_t = \wp\left(\left(1 - \sum_{i=1}^{p}\alpha_i - \sum_{j=1}^{q}\beta_j\right)P_c + \sum_{i=1}^{p}\alpha_i \mathbf{Y}_{t-i}\mathbf{Y}'_{t-i} + \sum_{j=1}^{q}\beta_j P_{t-j}\right). \tag{2-10}$$

"P_c es una matriz de correlación positiva definida de largo plazo, $\wp$ es el operador definido anteriormente, $\mathbf{Y}_t = \Delta_t^{-1}\mathbf{X}_t$ es el proceso de desvolatilización y los coeficientes satisfacen las siguientes condiciones $\alpha_i \geq 0$, $\beta_j \geq 0$ y $\sum_{i=1}^{p}\alpha_i - \sum_{j=1}^{q}\beta_j < 1$" (McNeil, Frey & Embrechts, 2015, p. 549) . Iterando la ecuación (2-6) de un modelo GARCH univariado se puede llegar al siguiente resultado:

$$\sigma_t^2 = \left(1 - \sum_{i=1}^{p}\alpha_i - \sum_{j=1}^{q}\beta_j\right)\sigma^2 + \sum_{i=1}^{p}\alpha_i X_{t-i} + \sum_{j=1}^{q}\beta_j\sigma_{t-j}^2. \tag{2-11}$$

Este modelo DCC puede representarse de la siguiente forma matricial:

$$\sum_{t+1}^{DCC} = \begin{bmatrix} \sigma_{t,1}^2 & \sigma_{t,12} & ... & \sigma_{t,1p} \\ \sigma_{t,21} & \sigma_{t,2}^2 & ... & \sigma_{t,2q} \\ \vdots & \vdots & ... & \vdots \\ \sigma_{t,p1} & \sigma_{t,p2} & ... & \sigma_{t,pq}^2 \end{bmatrix} = \begin{bmatrix} \sigma_{t,1} & 0 & ... & 0 \\ 0 & \sigma_{t,2} & ... & 0 \\ \vdots & \vdots & ... & \vdots \\ 0 & 0 & ... & \sigma_{t,p} \end{bmatrix} \times \begin{bmatrix} 1 & \rho_{t+1,12} & ... & \rho_{t+1,1q} \\ \rho_{t+1,21} & 1 & ... & \rho_{t+1,2q} \\ \vdots & \vdots & ... & \vdots \\ \rho_{t+1,p1} & \rho_{t+1,p2} & ... & 1 \end{bmatrix} \times$$
$$\begin{bmatrix} \sigma_{t,1} & 0 & ... & 0 \\ 0 & \sigma_{t,2} & ... & 0 \\ \vdots & \vdots & ... & \vdots \\ 0 & 0 & ... & \sigma_{t,q} \end{bmatrix}.$$

A parte de los modelos antes mencionados, en la literatura se encuentran otros, como lo son el VEC el cual genera demasiados parámetros para fines prácticos, el modelo DVEC y BEKK.

Entre las aplicaciones de los modelos antes mencionados se encuentran Escobar-Anel, Rastegari & Sfentoft (2020), los cuales propusieron y analizaron en su artículo Affin multivariate

GARCH models, una clase de modelo GARCH multivariados construido como combinaciones lineales de modelos GARCH unidimensionales que permitieron cálculos eficientes de gestión del riesgo y fijación de precios de opciones de múltiples activos, este modelo fue aplicado a cinco activos demostrando que esta metodología es más rápida que la simulación de Monte Carlo.

Cruz Zúñiga & Ramírez Tapia (2021), analizan el efecto de la incertidumbre real y la inflación sobre el crecimiento económico en México y Brasil, estimando un modelo CCC-GARCH bivariado para capturar la correlación entre la inflación y el producto interno bruto para obtener así una estimación más precisa de la incertidumbre inflacionaria y del producto. López Villada (2020), utiliza el modelo DCC-GARCH para estimar las volatilidades e investigar los posibles efectos de contagio entre las variables, índice de precios y cotizaciones de la bolsa Mexicana de valores y la volatilidad del tipo de cambio, concluyendo que la varianza condicional no disminuye rápidamente. Melo Velandia, Becerra Camargo & el Banco de la República de Colombia (2006), utilizaron modelos VARX-GARCH multivariado para establecer la relación, en frecuencia diaria entre dos tasas de interés de corto plazo, concluyendo que identificaron algunas relaciones entre las variables.

2.5. Máxima verosimilitud

Los modelos ARCH y GARCH se estiman mediante el método de máxima verosimilitud y los estimadores de los coeficientes se distribuyen aproximadamente normal en muestras grandes (James & Marck, 2012). Para utilizar el método de máxima verosimilitud con los modelos GARCH se asumen que la probabilidad de distribución condicional de ϵ_i es normal con media cero y varianza σ_i^2 (varianza estimada para el i-ésimo día), por tanto la probabilidad de que generalmente ocurra n observaciones $(\epsilon_1, \epsilon_2, ..., \epsilon_n)$ en el orden observado es (Hull, 2012):

$$L\left(\sigma_1, \sigma_2, \ldots, \sigma_n | \epsilon_1^2, \epsilon_2^2, \ldots, \epsilon_n^2\right) = \prod_{i=1}^{n} \left(\frac{1}{\sqrt{2\pi}\sigma_i^2} \times \exp\left(\frac{-\epsilon_i^2}{2\sigma_i^2}\right)\right). \tag{2-12}$$

Para maximizar la expresión en (2-12), se puede aplicar logaritmos e ignorando constantes para obtener:

$$\sum_{i=1}^{n} \left(-\ln(\sigma_i^2) - \frac{\epsilon_i^2}{\sigma_i^2}\right). \tag{2-13}$$

Es necesario realizar un procedimiento iterativo para encontrar los parámetros en el modelo que maximicen la suma de la expresión (2-12) (Hull, 2012).

3. Redes neuronales artificiales

Avances del área de inteligencia artificial (IA) han proporcionado otros métodos para abordar el problema de pronósticos de series de tiempo, entre las que se encuentran el aprendizaje profundo o Deep Learning, que es un área muy activa en la IA. La piedra angular del aprendizaje profundo es la Red Neuronal Artificial (RNA) o en breve red neuronal (RN) (Garth James & Trevor Hastie, 2021).

La idea de las RN está basada en el intento de simular los mecanismos que poseen las neuronas biológicas para procesar información, en las que una gran cantidad de neuronas están unidas por dendritas o conexiones, por las cuales fluye la información. Para Regueiro, Barro, Sánchez & Fernández-Delgado (1995), la RN se presenta como un sistema de procesamiento de información compuesto por un gran número de Elementos de Procesamiento (EP), o neuronas conectadas entre sí a través de conexiones de comunicación, donde estos EP son especialmente utilizadas para derivar restricciones lógicas de forma explícita, como el reconocimiento de patrones y análisis predictivo. Según Charu C. (2018) las RN están conectadas entre sí por medio de pesos o parámetros y que tienen la tarea de calcular una función de entrada y propagar estos valores calculados desde la entrada de la red hasta la salida y el aprendizaje de una RN ocurre combinando los pesos que conectan las neuronas.

La estructura más simple de una RN llamada perceptrón o neurona debido al intento de simular las actividades de una neurona biológica (ver figura **3-1**) está constituida por una sola capa de la siguiente manera ($\mathbf{X}$, Y) donde $\mathbf{X} = (\mathrm{X}_1, \mathrm{X}_2, ..., \mathrm{X}_p)$ son los valores del conjunto de entradas o vector de entrenamiento con p variables que ingresan a la RN, la cual es la encargada de realizar las operaciones y cálculos matemáticos para generar el valor de salida $\mathbf{Y}$, para Charu C. (2018), el cálculo de estas operaciones en el caso de una regresión lineal se realiza mediante una variación generalizada de la siguiente función lineal, $\mathbf{W} \times \mathbf{X} = \sum_{i=1}^{p} w_i \times x_i$, donde $\mathbf{W} = (\mathrm{w}_1, \mathrm{w}_2, ..., \mathrm{w}_p)$ es el vector de valores o pesos que se le asignan a cada valor x_i del vector $\mathbf{X}$ que tienen como finalidad definir con que intensidad cada una de estas variables de entrada afecta a la neurona, estos pesos se ajustan para modificar positiva o negativamente el valor de la suma y son los parámetros del modelo.

Otro tipo de RN son las multicapas donde las neuronas están dispuestas en capas de entrada, ocultas y de salida, esta arquitectura, según Charu C. (2018), es denominada red de avance porque las capas sucesivas se alimentan entre sí en la dirección de avance desde la entrada

hasta la salida; la suma ponderada de la función lineal de una red neuronal multicapa que realiza cada neurona es como se muestra en la ecuación (3-1), donde $\hat{Y}_1$ es el valor de predicción :

$$\hat{Y}_1 = W_1X_1 + W_2X_2 + ... + W_pX_p. \tag{3-1}$$

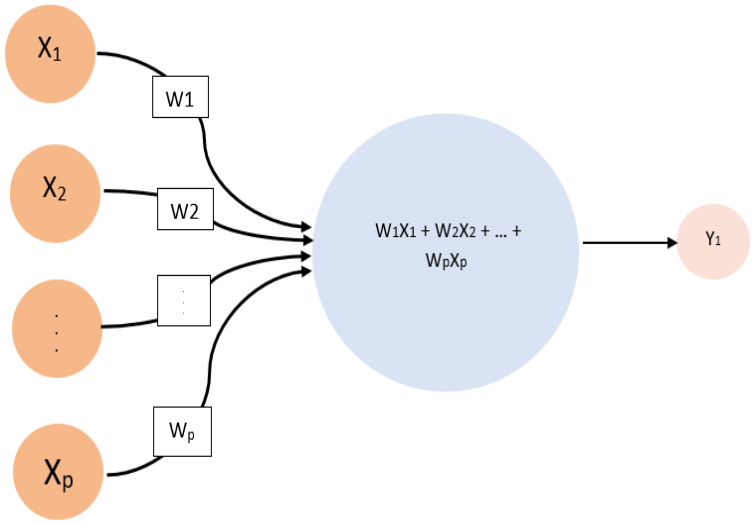

Figura 3-1.: Neurona (elaboración propia).

La ecuación (3-1) es llamada función de activación donde el resultado del valor de predicción genera un error y, que es calculada de la siguiente manera:

$$E(X) = Y - \hat{Y}_1, \tag{3-2}$$

donde Y es un valor de referencia y $\hat{Y}_1$ es el valor predicho, para poder obtener un valor de $E(X)$ cercano a cero se deben modificar los parámetros W_i. La ecuación (3-2) es denominada función de minimización o función de perdida la cual es definida sobre todos los datos de entrenamiento, el algoritmo de entrenamiento de las redes neuronales funciona alimentando cada instancia de datos de entrada $\mathbf{X}$ en la red uno por uno (o en lotes pequeños) para crear la predicción $\hat{Y}_1$; luego, los parámetros se actualizan en función del valor de error $E(X)$ como lo indica Charu C. (2018), de la siguiente manera: $\mathbf{W} + \alpha(Y - \hat{Y}_1)\mathbf{X} \Longrightarrow \mathbf{W}$, el parámetro α regula la tasa de aprendizaje de la neurona.

Muchas estructuras de las RN tienen una parte invariable de la predicción conocida como sesgo (b) que tiene como finalidad capturar la parte invariable de una predicción Charu C. (2018); b representada como otra conexión a la neurona en el que sus valores son aprendidos, es decir, se escogen durante el aprendizaje o entrenamiento de la neurona, dicho de otra manera, controla la salida para ayudar a la neurona a que ajuste de una mejor manera los datos iniciales. A partir de la ecuación (3-1) la función lineal para una neurona se reescribe como se muestra a continuación:

$$\hat{Y}_1 = W_1X_1 + W_2X_2 + ... + W_pX_p + b. \tag{3-3}$$

Gráficamente la ecuación (3-3) se puede representar como se muestra en la Figura **3-2**, esta ecuación es similar a una Regresión Lineal (RL) donde los pesos W_i representan los parámetros de la misma y b representa el intercepto.

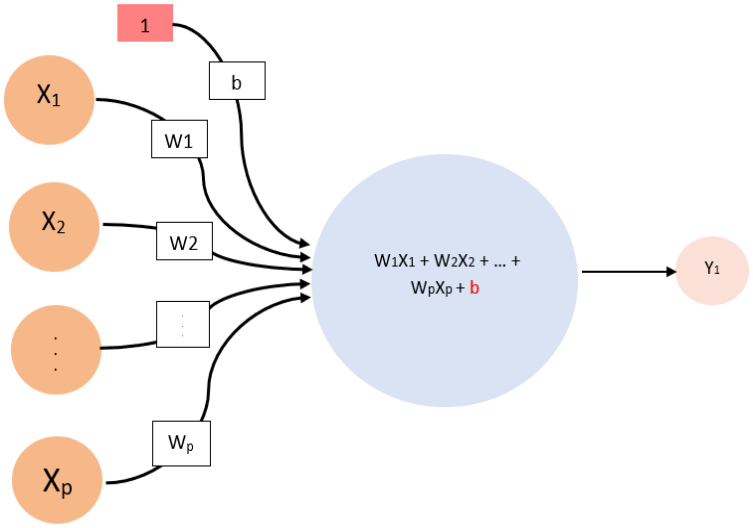

Figura 3-2.: Neurona con bias o unidad correctora de sesgo (elaboración propia).

Luego de calcular la ponderación de los valores de entrada, aparece otra operación interna que realiza cada una de las neuronas llamada función de activación, la cual según Regueiro, Barro, Sánchez & Fernández-Delgado (1995), tiene como misión establecer un valor de salida, básicamente lo que hace la función de activación es modificar el valor de salida (o la RL) de cada neurona para que finalmente la neurona pueda representar una relación no lineal entre los inputs y la variable de respuesta.

Según Alberto Ruiz, Basualdo & Matich (2001), los resultados del cálculo de una función de activación puede tomar valores desde -1 hasta 1, cuando la neurona toma el valor de -1 o 0 indica que la neurona esta totalmente inactiva y si toma el valor de 1 esta activa. Hay varias funciones de activación en las que se encuentra la escalonada cuyo rango es desde -1 a 1 como se muestra a continuación:

$$f_i\left(\sum_{j=1}^{n} w_{ij}x_j - \theta_i\right), \tag{3-4}$$

donde:

$$y_i = \begin{cases} 1 \; si \; \sum_{j=1}^{n} w_{ij}x_j \; \geq \; \theta_i \\ 0 \; si \; \sum_{j=1}^{n} w_{ij}x_j \; < \; \theta_i. \end{cases} \tag{3-5}$$

La función sigmoide con un rango de variación de 0 a 1 cuya ecuación se presenta a continuación:

$$f(x) = \frac{1}{1 - e^{-x}}. \tag{3-6}$$

La función tangente hiperbólica que tiene una forma similar a la sigmoide con la diferencia que el rango de variación es de -1 a 1:

$$f(x) = \frac{2}{1 - e^{-2x}} - 1, \tag{3-7}$$

y la función ReLu con rango de variación de 0 a 1 como se muestra en la siguiente ecuación (3-8):

$$g(z) = (z)_+ = \begin{cases} 0 \; si \; z \; < \; 0 \\ 1 \; otro \; caso. \end{cases} \tag{3-8}$$

En la Figura **3-3**, se muestran las funciones de activación, el uso de una de estas funciones depende del resultado final deseado.

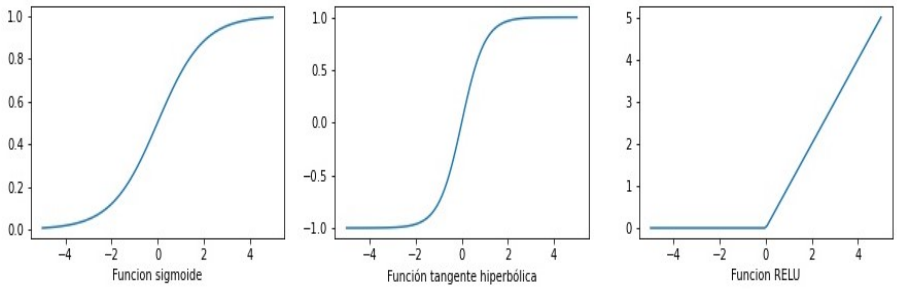

Figura 3-3.: Funciones de activación (elaboración propia en Python (Van Rossum & Drake Jr, 2009)).

Hasta acá se ha expuesto las neuronas y sus funciones, pero una RN está compuesta por muchas neuronas, la red o arquitectura más simple posee al menos tres capas, una de entrada en las que se ingresa un vector de variables $\mathbf{X}_p$, una o varias capas ocultas en las que se procesa la información como se describió en párrafos anteriores y las cuales son calculadas como funciones de la capa de entrada y, una capa de salida para predecir una respuesta, como se muestra en la Figura **3-4**. La capa oculta dentro de la RN puede tener varias capas (multicapas) y cada una de estas capas puede estar compuesta por una o varias neuronas como se muestra en la Figura **3-5**.

Una forma adecuada para describir las RN multicapa es en bloques de construcción. La función de pérdida en las RN multicapa es aun más compleja debido a que "la pérdida es una función de composición complicada de los pesos en capas anteriores, esta función de composición es calculada utilizando el algoritmo de *backpropagation*" (Charu C., 2018, p. 21), el procedimiento de este algoritmo es llevado a cabo por una serie de pasos:

- Primero, alimenta la RN con los datos $\mathbf{X}$ de entrenamiento, en el que los pesos $\mathbf{W}$ toman valores iniciales aleatorios.
- Segundo, al final de la red se calcula el MSE o Mean Square Error de los pronósticos de la siguiente manera: $MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$.
- Tercero, el MSE calculado en el paso anterior se multiplica por la derivada de la función de activación de cada neurona con el objetivo de calcular el error para todas las capas (salida y ocultas) de derecha a izquierda.

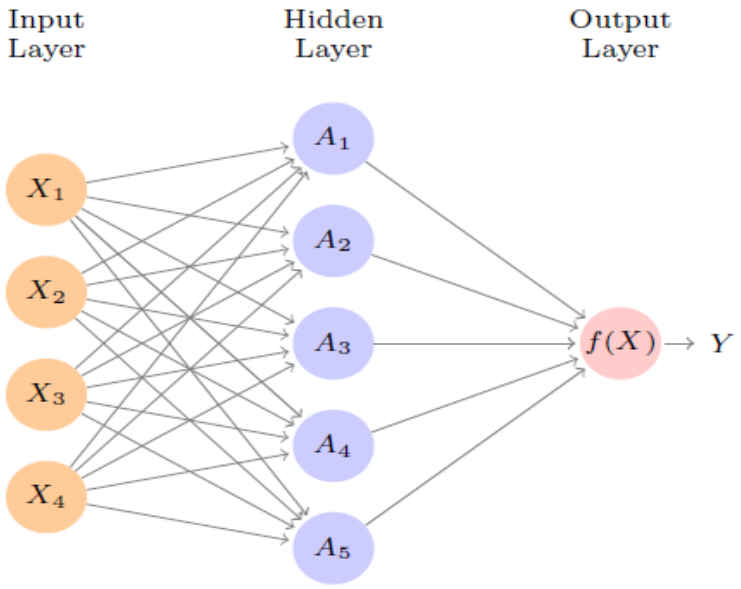

Figura 3-4.: Red neuronal o perceptrón de una sola capa. Fuente: Gareth, Witten, Hastie & Tibshirani, (2021).

- Cuarto, se aplica la regla de aprendizaje Δ, que sirve para calcular los pesos de entrada utilizando un método de gradiente y que arroja información del valor de cambio del coeficiente $\mathbf{W}$, de la siguiente manera: $\Delta\mathbf{W} = \alpha(Y - \hat{Y}_1)\mathbf{X}$.
- Quinto, se aplica el valor de aprendizaje Δ calculado en el paso anterior $\mathbf{W} = \mathbf{W} + \Delta\mathbf{W}$ para corregir los pesos.

Los pasos descritos anteriormente tienen dos fases, una fase denominada de avance hacia adelante (de izquierda a derecha) que es la que alimenta la RN con los datos de entrenamiento y utilizando un conjunto de pesos iniciales para calcular la función de pérdida y una fase de retroceso (de derecha a izquierda) que es cuando se modifican los pesos hasta alcanzar un error cercano a cero, ésta es la manera en la que la RN se va ajustando al conjunto de datos.

3.1. Proceso computacional de una RN

El diseño inicial de una RN (perceptrón multicapa) tiene una dinámica donde la información fluye desde la capa de entrada hacia la capa de salida o, de izquierda a derecha descrita como conexiones hacia adelante, pero diseños posteriores incluyeron otra dinámica descrita como conexiones hacia atrás en las que la información de la capa de salida o resultado se compara con un dato real y la diferencia entre los dos valores arroja un error, el cual se devuelve hacia la capa de entrada con el fin de modificar los pesos W_i para disminuir este error, esta

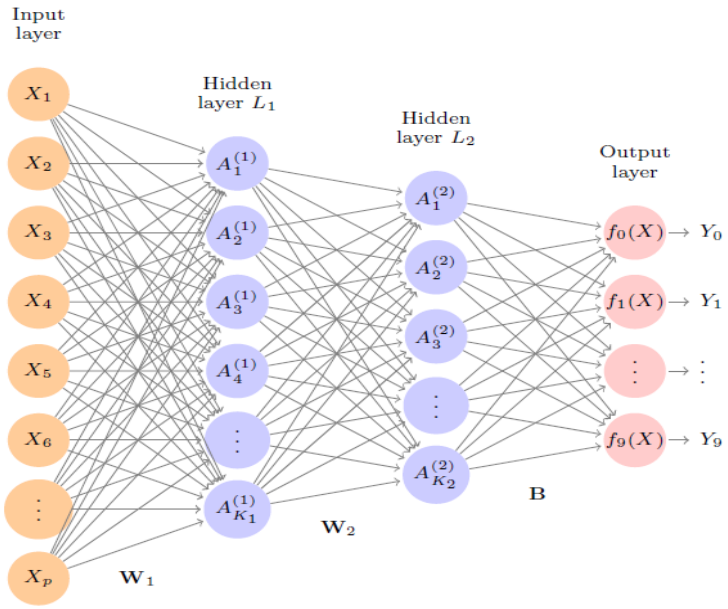

Figura 3-5.: Red neuronal multicapa. Fuente: Gareth, Witten, Hastie & Tibshirani, (2021).

propagación del error hacia atrás se conoce como *backpropagation*; Hilera & Matínez (1995), explican como se lleva a cabo el procedimiento de *backpropagation*, el cual se realiza en dos fases, en la primera se ingresan los datos que fluyen desde la capa de entrada, pasando por la capa oculta donde se hace el procedimiento matemático hasta llegar a la capa de salida, para obtener un resultado, el cual se compara con un resultado deseado, calculando con esto un valor de error para cada neurona de salida, en la segunda fase estos errores se trasmiten hacia atrás por todas las capas con el fin de que todas las neuronas hayan recibido una parte del error que describan su aportación relativa al error total.

El ajuste de los pesos por medio del algoritmo de *backpropagation* constituye el mecanismo con el que una RN aprende; como lo comenta Hilera & Matínez (1995), los cambios que tiene luego la estructura o arquitectura de la red durante el proceso de aprendizaje se traducen en la destrucción, modificación y creación de conexiones entre las neuronas; se pueden distinguir dos tipos de aprendizaje, el supervisado y el no supervisado; según Galarza Hernández (2017), en el aprendizaje supervisado se conoce previamente el resultado de la variable respuesta y, en el aprendizaje no supervisado hacen uso de una base de datos sobre la que no hay variable respuesta específica, el objetivo en este caso consiste en explorar y descubrir características o subgrupos dentro de la misma base de datos.

La modelación de un conjunto de datos por medio de una RN se lleva a cabo en dos fases, antes de iniciarlas, la cantidad de datos preliminares se debe dividir en dos grupos, uno de entrenamiento y otro de prueba. El desarrollo de la primera fase inicial con el grupo de en-

trenamiento, donde se usa este conjunto de datos para determinar los pesos W_i que definen el modelo de la RN, esto se calcula de manera iterativa hasta minimizar el error como se describió en párrafos anteriores, en esta primera fase puede haber sobreajuste que compromete la capacidad del modelo de generalizar, es decir, de pronosticar de manera adecuada la variable respuesta en observaciones con las que no ha sido entrenada la RN, para controlar esto se sigue con la segunda fase donde se utiliza el segundo grupo de datos de prueba o validación que permite controlar el proceso y evitar sobreajustes.

3.2. Red neuronal recurrente

Los actuales avances en Inteligencia Artificial (IA) han generado numerosas investigaciones sobre las RN y diversos tipos de éstas, las cuales dependen del objetivo del investigador. Si los datos de análisis tienen dependencias secuenciales como en una serie de tiempo, la mejor RN para llevar a cabo estos análisis son las Redes Neuronales Recurrentes (RNR), según Charu C. (2018), las capas de esta red tienen una correspondencia secuencial y temporal donde cada variable que ingresa a esta red, lo hace a una sola capa en un momento específico del tiempo, y éstas capas pueden interactuar con capas de un paso anterior, utilizando el mismo conjunto de parámetros en cada momento para garantizar que el modelo es similar durante el transcurso del tiempo; en otras palabras, la arquitectura por capas se repite en el tiempo por lo que la red se denomina recurrente.

"Las RNR en particular tienen un bucle de retroalimentación en sus capas de tal modo que puede almacenar información en memoria" (Bernaldo de Quiróz, 2020, p. 6), en la parte izquierda de la Figura **3-6** se observa el autobucle A_l que hará que el estado oculto de la RNR cambie después de la entrada de cada valor de X_l; en la parte derecha de la Figura **3-6**, se muestra un esquema de una RNR desenrollado; James, Witten, Hastie & Tibshirani (2021) detallan como opera esta RNR en la cual la entrada a esta red es una secuencia de vectores $\{\mathbf{X}_l\}_1^L$ por lo que $\mathbf{X} = (\mathrm{X}_1, \mathrm{X}_2, \ldots, \mathrm{X}_L)$ donde cada X_l representa un dato que alimenta una capa oculta secuencial $\{\mathbf{A}_l\}_1^L$, $\mathbf{A} = (\mathrm{A}_1, \mathrm{A}_2, \ldots, \mathrm{A}_L)$ que también tiene como entrada el vector de activación A_{l-1} del anterior paso de la secuencia, acumulando una historia de lo que ha sucedido antes, con el fin de alimentar la capa de salida y producir una predicción O_l para la salida Y; las letras $\mathbf{W}$, $\mathbf{B}$ y $\mathbf{U}$ son las matrices de pesos con valores ω, β y μ respectivamente.

La formula general de una RNR es planteada de la siguiente manera:

$$A_{lk} = g\left(\omega_{k0} + \sum_{j=1}^{p} \omega_{kj} X_{lj} + \sum_{s=1}^{k} \mu_{ks} A_{l-1,s}\right). \tag{3-9}$$

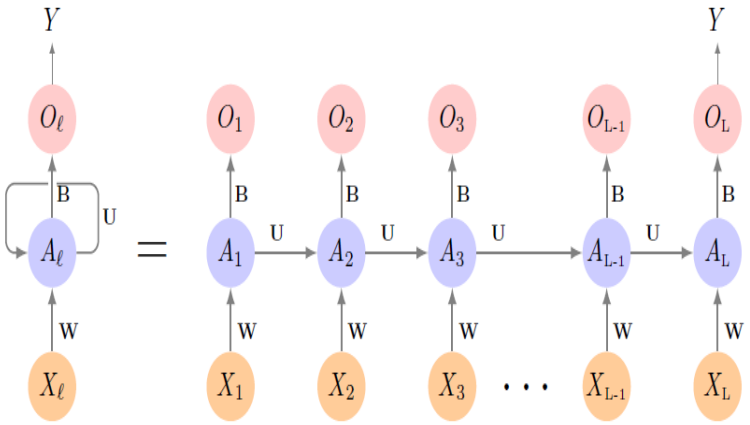

Figura 3-6.: Red neuronal recurrente. Fuente: James, Witten, Hastie & Tibshirani (2021).

> "La ecuación (3-9) considera un vector para la capa de entrada X_l con p componentes $X_l^\top = (X_{l1}, X_{l2}, ..., X_{lp})$, para la capa oculta $A_l^\top$ con k unidades $A_l^\top = (A_{l1}, A_{l2}, ..., A_{lk})$, una función de activación $g(\cdot)$ y tres matrices una $\mathbf{W}$ de $K \times (p+1)$ con una serie de pesos w_{kj} para la capa de entrada, una matriz $\mathbf{U}$ de $K \times K$ con pesos μ_{ks} para las capas ocultas y otra matriz $\mathbf{B}$ con pesos β_k para la capa de salida" (James, Witten, Hastie & Tibshirani, 2021, p. 423).

La ecuación (3-9) se plantea como una función en el tiempo t y el vector oculto en el tiempo $t-1$. Por lo tanto James, Witten, Hastie & Tibshirani (2021) menciona que aunque el estado oculto evolucione con el tiempo, los pesos y la función subyacente permanecen fijos en todas las marcas de tiempo. La salida O_l se calcula como se muestra a continuación:

$$O_l = \beta_0 + \sum_{k=1}^{k} \beta_k A_{lk}. \tag{3-10}$$

Para problemas de regresión, la función de pérdida para una observación (X, Y) es:

$$(Y - O_l)^2, \tag{3-11}$$

> "esta ecuación (3-11) solo hace referencia a la salida final $O_l = \beta_0 + \sum_{k=1}^{K} \beta_k A_{lk}$, las demás $O_1, O_2, ..., O_{l-1}$ no se utilizan. Cada que el modelo se ajusta se contribuye indirectamente al aprendizaje de la matriz de pesos $\mathbf{W}$, $\mathbf{B}$ y $\mathbf{U}$. Con n pares de secuencia de entrada/respuesta $(x_i,\ y_i)$ los pesos son encontrados minimizando la suma de cuadrados como se muestra a continuación (James, Witten, Hastie & Tibshirani, 2021, p. 424):

$$\sum_{i=1}^{n}(y_i - o_{il})^2 = \sum_{i=1}^{n}\left(y_i - \left(\beta_0 + \sum_{k=1}^{K}\beta_k g\left(\omega_{k0} + \sum_{j=1}^{p}\omega_{kj}x_{ilj} + \sum_{s=1}^{K}\mu_{ks}a_{i,l-1,s}\right)\right)\right)^2. \quad (3\text{-}12)$$

3.3. Red neuronal recurrente long short - term memory

Como una extensión de las RNR se desarrolló otra red neuronal llamada Long Short-Term Memory (RN-LSTM) por sus siglas en inglés, básicamente como lo dice Bernaldo Quiróz (2020), lo que busca esta red es ampliar su memoria para aprender de sucesos pasados, dicho de otra manera, recuerda información durante periodos largos y cortos de tiempo, pretendiendo almacenar la función de activación por un tiempo más largo.

La Figura **3-7** muestra la estructura de una RN-LSTM, la cual está compuesta básicamente por una celda de memoria (c_t) y tres puertas, una de entrada (i_t), una de olvido (g_t) y una de salida (o_t).

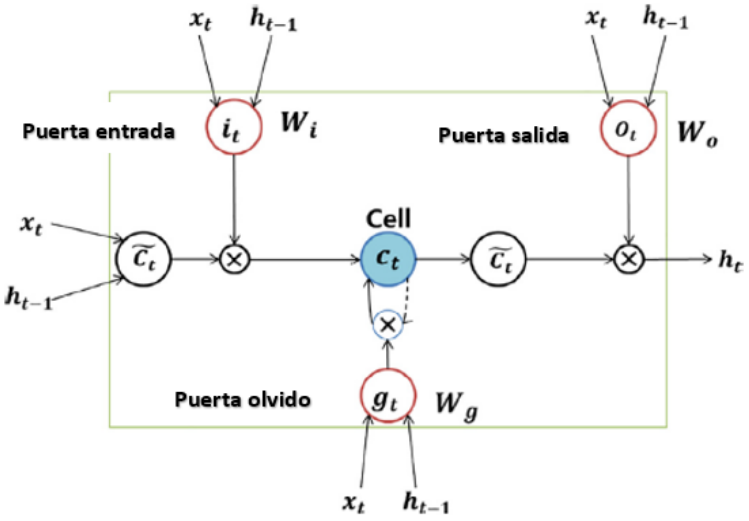

Figura 3-7.: RN-LSTM. Fuente: Ha Yung Kim & Chang Hyun Won, (2018).

En el momento t, x_t representa la entrada y h_t el estado oculto. La celda de estado c_t, el estado oculto h_t y las tres puertas g_t, i_t y o_t se calculan de la siguiente manera:

La puerta de olvido g_t, generada por una suma ponderada de x_t, h_{t-1} y b_f, con un valor de 0 a 1 a través de σ:

$$g_t = \sigma(U_g x_t + W_g h_{t-1} + b_f), \tag{3-13}$$

donde b_f es el sesgo sumado en una capa donde se cálculo un valor de salida por medio de la función de activación f; g_t depende de h_{t-1} ya que los valores se propagan hacia atrás o dicho de otra manera depende del estado de la celda anterior como entrada de la capa actual. $\mathbf{U}$ y $\mathbf{W}$ son las matrices de peso generadas en la puerta de olvido g_t. Un valor de 0 de g_t significa que no pasa información a través de la puerta y un valor de 1 significa que toda la información de entrada pasa a través de la puerta, por lo tanto g_t controla la información en el estado de la celda en un momento anterior c_{t-1}.

Puerta de entrada i_t, determina cuanta información nueva se almacena en el estado de la celda c_t:

$$i_t = \sigma(\mathbf{U}_i x_t + \mathbf{W}_i h_{t-1} + b_i). \tag{3-14}$$

Celda de estado c_t, se calcula por medio de las ecuaciones (3-13) y (3-14) de la siguiente manera:

$$\tilde{c}_t = tanh(\mathbf{U}_c x_t + \mathbf{W}_c h_{t-1} + b_c), \tag{3-15}$$

calcula la nueva información en el tiempo t y, su salida a través de la función tanh tiene un valor entre -1 y 1.

$$c_t = g_t \times c_{t-1} + i_t \times \tilde{c}_t, \tag{3-16}$$

donde c_t depende de la información del estado pasado c_{t-1} y la nueva $\tilde{c}_t$ en el tiempo t.

Estado oculto h_t, actualiza el estado de la celda c_{t-1} separando la información innecesaria:

$$h_t = o_t \times tanh(c_t), \tag{3-17}$$

Puerta de salida o_t, actualiza el estado del sistema:

$$o_t = \sigma(\mathbf{U}_o x_t + \mathbf{W}_o h_{t-1} + b_o). \tag{3-18}$$

Para las ecuaciones anteriores $\sigma(\cdot)$ es una función sigmoidea (Young Kim & Hyun Won, 2018, pp. 29-30).

Guamán Pachacama & Muñoz Génesis (2020), aplicaron pruebas de técnicas de aprendizaje automático supervisado mediante una RN-LSTM para pronosticar el comportamiento del consumo energético de una vivienda familiar, tomando datos de aproximadamente cuatro años y cuatro arquitecturas de la RN-LSTM. Para evaluar los resultados consideraron las siguientes métricas RMSE, MAPE, MAE para seleccionar el mejor modelo de la RN-LSTM, siendo la mejor estructura la RN-LSTM simple con salida vectorial. Mei-Li, Cheng-Feng, Hsiou-Hsiang, Po-Yin & Cheng-Hong (2021), desarrollaron un método efectivo de pronóstico del comercio exterior que se basa en una RN-LSTM, analizando los datos comerciales de diez países para extraer cambios temporales de los datos comerciales y proporcionar pronósticos comerciales efectivos, los resultados indicaron que el método puede modelar adecuadamente la información temporal sobre las tendencias de incertidumbre en los datos de comercio exterior.

Algunas de estas redes neuronales han sido mezcladas con métodos tradicionales no computacionales para mejorar el rendimiento y compararlas con el fin de verificar que las RN son

una buena metodología para los pronósticos, por ejemplo, Quintero Valencia (2019), propuso un modelo híbrido de una RN-LSTM-GARCH para el pronóstico de la volatilidad de la tasa representativa del mercado TRM, para lo cual incluyó como variables explicativas los coeficientes de modelos de series de tiempo GARCH, EGARCH y EWMA para la TRM, se evidenció que la inclusión de los coeficientes de los modelos GARCH y EGARCH mejora la precisión de las predicciones del modelo híbrido de la RN-LSTM comparado con un modelo de RN-LSTM estándar.

3.4. Computación para los análisis

Para el diseño de las redes neuronales ha sido muy utilizado un lenguaje de programación de alto nivel llamado Python (Van Rossum & Drake Jr, 2009) con licencia de código abierto, para el desarrollo de aplicaciones de todo tipo como ciencia de datos y el Machine Learning (ML). Python (Van Rossum & Drake Jr, 2009) utiliza librerías para todas sus aplicaciones, es decir, un conjunto de funcionalidades o implementaciones codificadas en un lenguaje de programación que permiten realizar nuevas tareas. Algunas de las librerías más utilizadas para el desarrollo de redes neuronales son:

- Matplotlib: para generar gráficos de calidad en series de tiempo, histogramas, diagramas de errores entre otros.
- Seaborn: es una librería basada en matplotlib para la visualización de datos estadísticos.
- NumPy: para el análisis de datos entre distintos algoritmos; la estructura de datos que implementa es por medio de vectores y matrices.
- SciPy: opera en la misma estructura de datos proporcionados por NumPy.
- Pandas: la estructura de datos que maneja son Series para datos en una dimensión y DataFrame para datos en dos dimensiones.
- Scikit-learn: está basada en NumPy, SciPy y Matplotlib para el Machine Learning y análisis de datos, la principal ventaja es para el aprendizaje automático, supervisado y no supervisado.
- TensorFlow: desarrollado por Google para realizar cálculo numéricos mediante diagramas de flujo de datos.
- Keras: interfaz de alto nivel para trabajar con redes neuronales.
- PyTorch: desarrollada por Faceebook para el cálculo numérico eficiente en CPU y CPUs.

- Anaconda: distribución de Python (Van Rossum & Drake Jr, 2009) para cálculo numérico, análisis de datos y Machine Learning.

Otra de las herramientas tecnológica muy utilizada para el análisis estadístico es R como lenguaje de programación de software libre muy utilizada en ciencia de datos, matemáticas financieras, aprendizaje automático, entre otras. R (R Core Team, 1990) utiliza un entorno de desarrollo integrado (IDE) como editor de código fuente a RStudio; al igual que Python (Van Rossum & Drake Jr, 2009), RStudio maneja librerías para el análisis estadístico, a modo de mencionar unas pocas se tiene a:

- tidyverse: para la ciencia de datos.
- ggplot2: para la visualización de datos.
- stringr: para manipular cadenas de caracteres (variables categóricas).
- lubridate: para convertir caracteres en formato fecha.
- tidyr: para manipular datos en forma de tablas; convertir filas en columnas y viceversa, dividir y juntar columnas.
- dplyr: para filtrar seleccionar, agrupar, crear nuevas variables a partir de las existentes, para una serie de datos. También realiza cálculos estadísticos como la media la varianza de cada una de las variables.

4. Enfoque metodológico

Uno de los objetivos de este trabajo consiste en comparar la metodología DCC-MGARCH y una metodología menos convencional, como la red neuronal long short-term memory (RN-LSTM) clásica, como modelos para pronosticar las matrices de covarianzas de los rendimientos de cuatro activos: Bancolombia, Grupo Sura, Corficol e Isa cotizados en la BVC. La covarianza es un factor importante para determinar la relación que pueda haber entre estas variables. La matriz de covarianzas es cuadrada de tamaño $n \times n$ con la varianza de cada variable en cada uno de los elementos de la diagonal y las entradas por fuera de esta correspondiéndose con las covarianzas entre cada par de variables; la covarianza entre dos variables es estimada por medio de la siguiente manera:

$$S_{jk} = \frac{\sum_{i=1}^{n}(X_{ij} - \bar{X}_j)(X_{ik} - \bar{X}_k)}{n-1},\ para\ j,k = 1,...,p. \tag{4-1}$$

Este capítulo se divide en tres secciones, en la primera sección se describe la base de datos utilizada para los respectivos análisis; en la segunda sección se describe el procedimiento para el cálculo de las covarianzas con la metodología DCC-MGARCH utilizando el software R (R Core Team, 1990) y en la tercera sección se explica la metodología para el cálculo de los pronósticos con la RN-LSTM utilizando el software Python (Van Rossum & Drake Jr, 2009).

4.1. Base de datos

Para el desarrollo de este trabajo y los respectivos análisis, se utilizaron datos publicados por BVC del índice bursátil COLEQTY, el cual a la fecha de selección de los datos estaba compuesto por 40 acciones, la participación de cada una de estas acciones es determinada por el flotante el cual es el porcentaje de las acciones que pueden ser adquiridas por un accionista (BVC, 2020), de estas acciones se seleccionaron cuatro, las cuales fueron Bancolombia, Grupo Sura, Corficol e Isa con una participación al índice de 10.11 %, 8.45 %, 3.14 % y 9.17 % respectivamente.

Los datos corresponden al periodo entre 2018/01/02 y 2019/12/30, con datos diarios, exceptuando fines de semana (sábado y domingo), días festivos y algunos días de fechas especiales

como lo fueron navidad, año nuevo, semana santa y otros, para un total de 488 datos; se seleccionó este conjunto de datos con el objetivo de evitar en lo posible el efecto que pudiese ejercer la pandemia COVID-19 o elecciones presidenciales sobre las variables seleccionadas; los datos descargados de la página contenía la siguiente información para cada activo: cantidad, volumen, precio de cierre, precio mayor, precio medio, precio menor, variación porcentual, variación absoluta y fecha de cotización, para los respectivos análisis en este trabajo se consideró el precio de cierre al final de cada día.

Bancolombia con presencia en varios países como Colombia, Panamá, Guatemala y El Salvador, realizan actividades bancarias que comprende cuentas, depósitos, servicios transaccionales, créditos de consumo, comerciales, de vivienda, microcréditos y nuestras plataformas a la mano y nequi, leasing, renting, corretaje de bolsa, fiducia, banca de inversión y soluciones de mercado de capitales. Las acciones se encuentran listadas en la Bolsa de Valores de Colombia desde 1981 y en la Bolsa de Valores de Nueva York (NYSE) desde 1995. Sus accionistas a diciembre del 2022 son Suramericana de inversiones y filiales, programa ADR, fondos de pensiones colombianos privados, otros accionistas internacionales y otros accionistas locales con unas participaciones del 24.5 %, 12.7 %, 25.2 %, 18.8 % y 18.7 % respectivamente. El número de acciones ordinarias en circulación 509,704,584 y preferenciales de 452,122,416 (Bancolombia, 2023).

Grupo Sura es un gestor de inversiones, como compañía holding del conglomerado financiero SURA-Bancolombia, presente en 11 países de América Latina. Cotiza en la Bolsa de Valores de Colombia (BVC) y estan inscritos en el programa ADR -Nivel I, en Estados Unidos. La asignación de capital del Grupo Sura tiene como foco prioritario tres inversiones que son Suramericana, Sura Asset Management y Bancolombia, igualmente, son los principales accionistas (no controlantes) en dos inversiones industriales con las que comparten vínculos patrimoniales y una filosofía en la forma de hacer empresa las cuales son Grupo Argos y Grupo Nutresa. Desde 2011 Grupo Sura hace parte el Índice Global de Sostenibilidad Dow Jones (DJSI), que reconoce a las empresas con mejores prácticas económicas, sociales, ambientales y de gobierno corporativo en el mundo (Grupo Sura, 2023).

Isa es una empresa de servicios públicos mixta, lo que significa que cuentan con inversionistas estatales, públicos y privados. Isa pertenece al grupo Ecopetrol que opera en los negocios de energía eléctrica, vías, telecomunicaciones y TIC. Con 28.464 accionistas a un valor de $32.80 el valor nominal de la acción y en el cual Ecopetrol es el accionista mayoritario con una participación del 51.41 % del capital social; a diciembre del 2022 sus acciones en circulación fueron de 1.107.677.894 de los cuales 10 % son inversionistas extranjeros y 90 % son inversionistas nacionales. Isa participa con sus empresas en Colombia, Perú, Chile, Brasil, Bolivia, Argentina, Bermudas, Centro América y Estados Unidos. Las acciones y bonos de Isa se transan en la Bolsa de Valores de Colombia y cuenta con American Depositary Receipt

(ADR) Nivel I que se negocian en el mercado Over the Counter (OTC) de Estados Unidos; actualmente las acciones en circulación son ordinarias, nominativas y desmaterializadas, no existiendo restricciones estatutarias a su transferibilidad (Isa, 2022).

Corficol o Corficolombiana es una inversionista líder en la estructuración, gestión y administración de empresas en Colombia; sus inversiones en los sectores más dinámicos son la agroindustria, hotelería, financiero, energía e infraestructura, sus áreas de negocio son los portafolios de inversión, tesorería, banca de inversión y banca comercial. La composición accionaria de Corficol esta comprendida por Grupo Aval y Adminegocios, Fondo de Pensiones Colombiana, inversionistas internacionales y otros, con unas participaciones del 64.3 %, 9.2 %, 1.9 % y 24.6 % respectivamente (Corficol, 2023).

En la figura **4-1** se muestra el comportamiento del precio de las cuatro acciones entre los periodos 2018/01/02 y 2019/12/30, en estas gráficas se puede apreciar que no hay estacionalidad en ninguna de éstas acciones ya que en algunos periodos de tiempo sus precios tienden al alza y en otros a la baja, por lo que se puede considerar que el comportamiento de los precios de los cuatro acciones tienen componentes con tendencias crecientes y decrecientes en diversos periodos de tiempo.

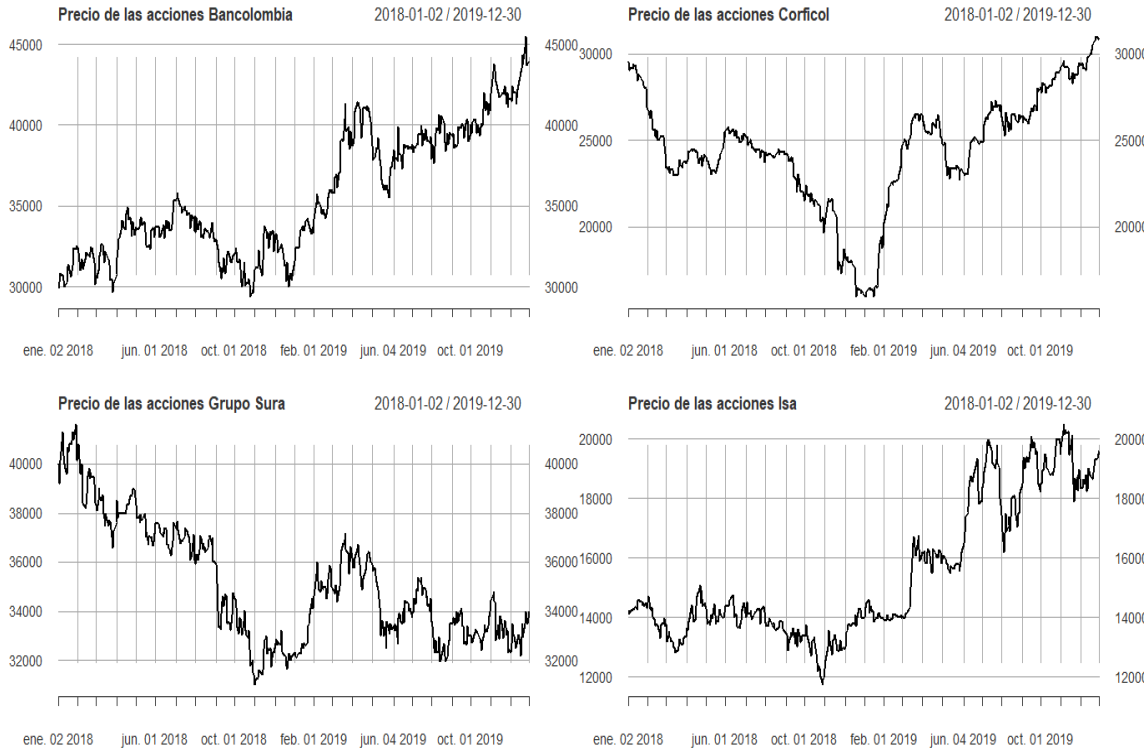

Figura 4-1.: Precio de las acciones acciones entre los periodos 2018/01/03 - 2019/12/30.

4.2. El modelo DCC-MGARCH

Para realizar el análisis con la metodología se procede en el siguiente orden:

1. Se inicia con la lectura de estos (precio de cierre) de cada uno de los cuatro activos seleccionados. Con el valor del precio de cierre se calculan los retornos de cada uno de los activos con la ecuación (2-1). Dado que los datos están distribuidos en una escala de tiempo se procede a transformarlos a una clase de series de tiempo (ts).

2. Luego de cargar los datos a R (R Core Team, 1990) se procede a realizar el análisis con el modelo DCC-MGARCH, primero se debe fijar para cada activo un modelo GARCH univariado, enseguida se toman los retornos de los cuatro activos considerados, se les ajusta el modelo con la librería *dccspec* de R (R Core Team, 1990); el cálculo de los parámetros se llevó a cabo con la distribución t-Student y con varios órdenes para el modelo, que correspondieron al GARCH(1, 1), GARCH(2, 1), GARCH(1, 2) y GARCH(2, 2), al final se seleccionó el modelo GARCH(1, 1) por su menor valor de Akaike de 13.775, el orden de este modelo también es uno de los más utilizados en series financieras, según Kim & Won (2018), por ser particularmente fácil de manejar y superior a otros modelos financieros en series de temporales.

3. Una vez ajustado el modelo DCC-MGARCH a los cuatro activos se puede calcular las matrices de covarianzas por medio de la función `dcc.fit@mfit\$H`; calculadas estas matrices, se dividen en los siguientes dos grupos:

   - Grupo 1: comprende las fechas 2018/01/03 hasta 2018/01/17 para un total de 387 datos para el cálculo de los pronósticos.

   - Grupo 2: comprende las fechas 2019/08/05 hasta 2019/12/30 para un total de 100 datos con el objetivo de comparar y evaluar el desempeño del pronóstico.

4. Se procede a realizar los pronósticos con el grupo 1, con el objetivo de realizar las comparaciones con las matrices del grupo 2. La validación del desempeño de los pronósticos se realizó por medio del error porcentual absoluto medio (MAPE), que mide el tamaño del error absoluto medio en términos porcentuales, el error cuadrático medio (RMSE) que mide la cantidad de error que hubo entre los pronósticos y valores de comparación o reales y la desviación absoluta media (MAD) la cual calcula la media de las desviaciones absolutas. Para cada una de estas medidas se calcularon los intervalos de confianza del 95 %.

4.3. RN-LSTM con Python

Para poder aplicar las RN en el software Python (Van Rossum & Drake Jr, 2009) y realizar los pronósticos, las matrices de covarianzas calculadas con R (R Core Team, 1990) se transformaron a vectores obteniendo así, seis vectores de covarianzas que fueron: Bancolombia - Corficol, Bancolombia - Grupo Sura, Bancolombia - Isa, Corficol - Grupo Sura, Corficol - Isa y Grupo Sura - Isa, y cuatro vectores de varianzas que fueron, Bancolombia, Grupo Sura, Corficol e Isa.

El procedimiento que se describirá a continuación para la RN-LSTM en Python (Van Rossum & Drake Jr, 2009) se realizó de igual manera para cada uno de los seis vectores de covarianzas y los cuatro vectores de varianzas mencionados anteriormente, calculado con los 487 datos.

1. Se normalizaron los datos con la función `MinMaxScaler` que tiene como objetivo escalar o normalizar los datos en el rango 0 - 1, ya que la RN-LSTM es sensible a los datos de entrada cuando se va a utilizar la función de activación.

2. Luego de normalizar los datos (para cada uno de los diez vectores) se procede a dividir en dos grupos cada vector, uno de entrenamiento y otro de prueba; el grupo de entrenamiento está compuesto por el 80 % de los 487 datos y el restante 20 % como datos de prueba correspondientes al final del periodo observado. Los dos grupos, el de pronósticos y prueba quedarán de tamaño similar a los grupos establecidos para la prueba del modelo DCC-MGARCH llevado a cabo en R (R Core Team, 1990).

3. Luego de dividir los datos se debe especificar el número de pasos de tiempo previos que se utilizan como variables de entrada para predecir el siguiente periodo de tiempo. Para este caso fue un paso atrás en cada uno de los dos grupos (entrenamiento y prueba), esto con el fin de imitar el mecanismo del modelo DCC-MGARCH(1, 1) que se empleó para realizar los pronósticos en R (R Core Team, 1990); en este paso se crearán dos vectores de datos, uno donde están las covarianzas en un momento del tiempo t y otro vector con las covarianzas en el momento $t + 1$.

4. Hasta este punto los datos tienen el siguiente formato bidimensional: [muestras, características], para que la RN-LSTM lea e interprete la serie de tiempo, los datos deben tener la forma tridimensional de la siguiente manera: [muestra, pasos de tiempo, características], la primera dimensión es el número de registros o filas, la segunda dimensión es el número de pasos de tiempo, que para este caso fue de uno y, la tercera dimensión

es el número de indicadores, el cual es uno ya que solo se está evaluando el precio de cierre del activo, para obtener este formato se utiliza la función `numpy.reshape()`. Con la estructura de datos indicada se procede a implementar la RN-LSTM.

5. El modelo de RN-LSTM que se creará, será con dos capas secuenciales LSTM, esta secuencia se hace por medio de la función `keras.models()` de Python (Van Rossum & Drake Jr, 2009) el cual se especifica al inicio de la red; luego se genera la primera capa LSTM donde se especifican la cantidad de neuronas en esta capa, la cual fueron de cuatro y el número de pasos de tiempo que fue uno, luego se genera la segunda capa LSTM con las mismas especificaciones que la anterior, posterior a esto se genera la capa de salida con una sola neurona, por último se compila la red creada antes del entrenamiento con los datos, utilizando la función de pérdida del error cuadrático medio. Para reducir la pérdida o para optimizarla se utiliza el algoritmo *Adam* (*Adaptive moment estimation*), que es común para el aprendizaje profundo y que tiene como objetivo el descenso del gradiente estocástico calculando una combinación lineal entre el gradiente y el incremento anterior y considera los gradientes recientemente aparecidos en las actualizaciones para mantener diferentes tasas de aprendizaje por variable para así converger más rápido a un valor mínimo, que en este caso es el error. *Adam* es la combinación de otros dos algoritmos, *AdaGrad* (*Adaptive Gradient Algorithm*) el cual escala y adapta los pesos o parámetros utilizando diferentes tasas de aprendizaje para las variables teniendo en cuenta el gradiente acumulado en cada iteración para cada una de ellas manteniendo así un factor de entrenamiento específico, y el algoritmo *RMSprop* (*Root Mean Square Propagation*) que es una variación de *AdaGrad* donde en lugar de mantener los gradientes acumulados utiliza los gradientes más recientes.

6. Luego de ajustado el modelo de a RN-LSTM se procede a entrenar la red con los datos de entrenamiento (387 datos) con la función `model.fit()`, con el objetivo de estimar el rendimiento del modelo.

7. Con la red entrenada y ajustada se pueden realizar las predicciones con los datos de prueba con la función `model.predict()`.

8. Teniendo en cuenta que en el paso uno los datos se transformaron con la función `MinMaxScaler`, ahora se debe invertir esta transformación con la misma función. Realizados los pronósticos se procede a evaluar estos valores por medio de MAPE, RMSE, MAD y el cálculo del IC del 95 % para las predicciones igual que con R (R Core Team, 1990) .

5. Resultados

En la Tabla **5-1** se muestra las estadísticas descriptivas de los retornos, en estos resultados se puede apreciar que el valor de la media está muy cercana a cero para los retornos de los cuatro activos, Grupo Sura, Bancolombia (BColombia), Isa, y Corficol. La desviación estándar para el retorno Grupo Sura es casi uno, su valor de la curtosis es muy cercana a cero lo que tiende a ser mesocúrtica, la asimetría negativa indica más valores a la izquierda de la media y con el resultado del p-valor obtenido mediante del test de Shapiro-Wilks menor al 5 % se puede inferir que los retornos de Grupo Sura no se distribuye aproximadamente normal; las desviaciones estándar de los demás retornos, BColombia, Isa y Corficol son un poco mayores que uno y muy similares entre sí, las curtosis de los retornos BColombia e Isa son menores que uno e interpretadas como mesocúrticas y sus p-valores mediante el test de Shapiro-Wilks menores al 5 % indican que estos retornos no se distribuye aproximadamente normal, la curtosis del retorno Corficol es mayor que cero que indica una leptocurtosis en el que los datos están muy cercanos a la media y con una curva muy apuntada y su p-valor dado por el test de Shapiro-Wilks menores al 5 % indican que estos retornos no se distribuye aproximadamente normal; la asimetría de los retornos BColombia, Isa y Corficol son cercanas a cero por lo que se puede deducir que los valores tienden a estar agrupados en la media.

Tabla 5-1.: Estadísticas descriptivas de los retornos para los cuatro activos en el periodo (2018/03/01 - 2019/12/30).

	BColombia	Corficol	Grupo Sura	Isa
Mínimo	-4.8155	-9.0247	-4.6437	-5.8496
Q_1	-0.7918	-0.6852	-0.6561	-0.7563
Media	0.0791	0.00886	-0.0333	0.0671
Mediana	2.5001	0.0	-0.3722	2.3996
Desviación estandar	1.6010	1.5165	1.1480	1.6897
Q_3	1.0079	0.6537	0.6695	0.9343
Asimetría	-0.0499	-0.0999	-0.2305	0.0630
Curtosis	0.6177	4.3474	0.5589	0.8997
p-valor (Test de Shapiro Wilks)	0.0012	0.0000	0.0033	0.0001

La representación gráfica de los retornos de los cuatro activos calculados con la ecuación (2-

1), se muestra en la figura **5-1**, en éstas se puede apreciar la variabilidad de los rendimientos en el tiempo.

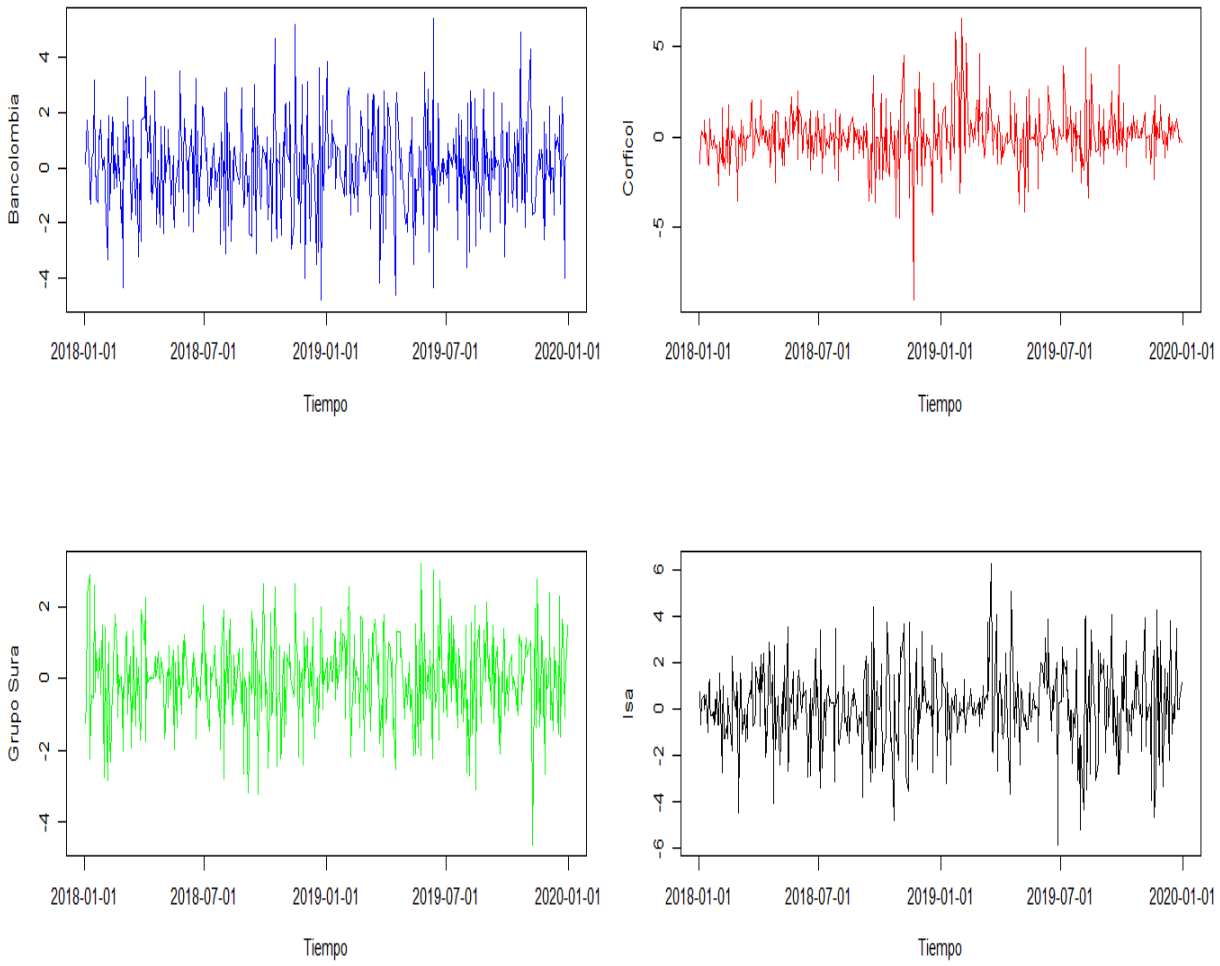

Figura 5-1.: Rendimientos de los retornos entre 2018/01/03 - 2019/12/30.

La Tabla **5-2** muestra los resultados paramétricos del modelo DCC-MGARCH de los datos completos, observando los resultados de los p-valores se aprecia que son estadísticamente significativos al 5 % sólo para los parámetros β_1 o varianzas al cuadrado de todos los activos y el α_1 de Corficol e Isa, los demás parámetros (μ, ω, α_1) de los cuatro activos son no significativos. Los valores del rendimiento promedio diario (μ) a largo plazo es mayor para BColombia con un valor de 0.0927 y menor para Grupo Sura con un valor de -0.0070; la varianza a largo plazo de retorno (ω) es mayor para Corficol con un valor de 0.5206 de volatilidad y menor para Bcolombia con un valor de 0.0232; α_1 que es el impacto que tienen la varianza al cuadrado rezagada en el retorno de hoy es mayor para Corficol con un valor de 0.3145 y menor para BColombia con un valor de 0.0, estos valores explican la volatilidad de un día atrás; β_1 que es mayor para el Bcolombia con un valor del 0.9915 explica el impacto

que tienen los residuos cuadrados rezagados por la varianza ajustada del día anterior y es menor para Corficol con un valor del 0.5809.

Tabla 5-2.: Parámetros óptimos obtenidos con el modelo DCC-MGARCH para los cuatro activos.

Activo	**Parámetro**	**estimado**	**Std. Error**	**t valor**	**Pr(>|t|)**
BColombia	μ	0.0927	0.0682	1.3594	0.1740
	ω	0.0232	0.0234	0.9921	0.3211
	α_1	0.0	0.0088	0.0	1.0
	β_1	0.9915	0.0006	1608.2469	0.0
Corficol	μ	0.0088	0.0461	0.1915	0.8482
	ω	0.5206	0.2925	1.7795	0.0752
	α_1	0.3145	0.1461	2.1522	0.0314
	β_1	0.5809	0.1672	3.4735	0.0005
Grupo Sura	μ	-0.0070	0.0504	-0.1379	0.8903
	ω	0.0742	0.1101	0.6741	0.5002
	α_1	0.0484	0.0408	1.1852	0.2359
	β_1	0.8956	0.1145	7.8224	0.0
Isa	μ	0.06642	0.0565	1.1757	0.2397
	ω	0.1409	0.1150	1.2247	0.2207
	α_1	0.2249	0.0591	3.8060	0.0001
	β_1	0.7741	0.0578	13.3885	0.0
Parámetros del modelo	dcca1	0.0	0.0	0.2337	0.8152
	dccb1	0.8951	0.0977	9.1657	0.0
	mshape	7.8080	1.2125	6.4393	0.0

Para justificar el modelo DCC-MGARCH los parámetros adicionales $dcca1$ y $dccb1$ deben ser mayores que cero y su suma menor que 1, de los resultados vemos que $dcca1 = 0$ lo que pudiera indicar que el modelo adecuado es un CCC-MGARCH (correlacional condicional constante GARCH Multivariate) pero el valor de $dccb1 = 0.8951 > 0$ y $dcca1 + dccb1 = 0.8951 < 1$ pueden justificar el uso del modelo DCC-MGARCH. El último valor de la tabla es el resultado del parámetro $mshape = 7.8080$ que especifica los grados de libertad de la distribución t-Student, este valor indica una distribución con cola gruesa para el término de error.

En el Tabla **5-3** se presentan las medidas de VaR (Valor en Riesgo) y ES (déficit esperado o Expected Shortfall por sus siglas en inglés) como una medición del riesgo de los activos. El VaR es un cuantil de la distribución de pérdidas y calcula la máxima pérdida esperada

como se indica a continuación:

> Dado un nivel de confianza de $\alpha \in (0,1)$, el VaR de una cartera con pérdida L en el nivel de confianza α viene dado por el número más pequeño l tal que la probabilidad de que la pérdida L supere a l no es mayor que $1-\alpha$ como se muestra a continuación: $VaR = VaR_\alpha(L) = inf\{l \in R : P(L > l) \leq 1-\alpha\} = inf\{l \in R : F_L(l) \geq \alpha\}$, donde $F_L(l) = P(L \leq l)$ es el diferencial de la distribución de pérdidas; los valores más comunes de α son $\alpha = 0.95$ y $\alpha = 0.99$ (McNeil, Frey & Embrechts, 2015, p. 64).

El ES (Expected Shortfall) es la media o el valor esperado de la distribución del VaR dicho de otra manera. "La ES a un nivel de confianza $\alpha \in (0,1)$, para una pérdida L con $E(|L|) < \infty$ esta definida como, $ES_\alpha = \frac{1}{1-\alpha}\int_\alpha^1 VaR_u(L)du$" (McNeil, Frey & Embrechts, 2015, p. 69). Estas medidas se realizaron para cada uno de los retornos con un horizonte de un día o periodo y con un nivel de confianza del 99 % y 95 % ($\alpha = 1\,\%, 5\,\%$), con esto, los resultados muestran que la máxima pérdida posible con un nivel del 99 % de confianza al final de un periodo es mayor para el retorno Isa con un valor del 4.3620 % de pérdida para un capital invertido y menor para el retorno Grupo Sura con un valor de 2.7930 % de pérdida y, con un nivel de confianza del 95 % la máxima pérdida fue para el retorno Isa con un valor del 2.8269 % y menor para Grupo Sura con un valor de 2.1262 %.

Tabla 5-3.: VaR (Valor en Riesgo) - ES (Déficit esperado) para los retornos de los cuatro activos.

Activo	**VaR** 95 %, $\alpha = 5\,\%$	**VaR** 99 %, $\alpha = 1\,\%$	**ES**
BColombia	2.6690	4.0598	3.4285
Corficol	2.4520	4.1267	3.5558
Grupo Sura	2.1262	2.7930	3.7210
Isa	2.8269	4.3620	3.7210

En la Tabla **5-4**, se muestran los valores máximos, mínimos y el rango en el que se movieron los pronósticos de las varianzas y covarianzas con los modelos DCC-MGARCH en R (R Core Team, 1990) y la RN-LSTM en Python (Van Rossum & Drake Jr, 2009), los resultados de los pronósticos de la volatilidad (varianza) con la metodología DCC-MGARCH fue más acotado para los activos Corficol e Isa y con la metodología RN-LSTM para los activos BColombia y Grupo Sura; el rango de los pronósticos de las covarianzas fue más acotado con la metodología DCC-MGARCH con excepción del rango de BColombia - Grupo Sura que fue menor con la metodología RN-LSTM.

Tabla 5-4.: Máximos, mínimos y rangos para los pronósticos de las varianzas - covarianzas con ambas metodologías DCC-MGARCH en R (R Core Team, 1990) y RN-LSTM en Python (Van Rossum & Drake Jr, 2009).

	Resultados DCC-MGARCH			Resultados RN-LSTM		
Varianzas	**Máximo**	**Mínimo**	**Rango**	**Máximo**	**Mínimo**	**Rango**
BColombia	2.9014	2.8511	0.0503	2.6361	2.6276	0.0085
Corficol	2.6497	1.9396	0.7101	4.3623	1.0907	3.2716
Grupo Sura	1.2760	1.2189	0.0572	1.3674	1.3597	0.0078
Isa	6.3153	3.2251	3.0893	7.0590	1.8138	5.2451
Covarianzas	**Máximo**	**Mínimo**	**Rango**	**Máximo**	**Mínimo**	**Rango**
BColombia/Corficol	0.2678	0.227	0.0407	0.3977	0.2867	0.1110
BColombia/Grupo Sura	0.7398	0.7280	0.0118	0.7635	0.7634	0.0001
BColombia/Isa	1.1227	0.8069	0.3158	1.1100	0.5943	0.5157
Corficol/Grupo Sura	0.4727	0.4138	0.0589	0.7833	0.5218	0.2615
Corficol/Isa	0.8069	0.6434	0.1635	1.9862	0.7343	1.2519
Grupo Sura/Isa	0.9141	0.6385	0.2756	1.1338	0.8020	0.3318

Todas las covarianzas calculadas fueron positivas, por lo que se cumple una de las condiciones de los supuestos del modelo DCC-MGARCH que dice que las covarianzas deben ser positivas. En la figura **5-2** se muestran las gráficas de las covarianzas calculadas en R (R Core Team, 1990) de los valores reales (línea negra) y las predicciones (línea roja) para todas las variables. Los valores medios de las covarianzas para los activos BColombia - Corficol, BColombia - Grupo Sura, BColombia - Isa, Corficol - Grupo Sura, Corficol - Isa y Grupo Sura - Isa fueron de 0.2953, 0.75788, 0.73929, 0.4528, 0.6226 y 0.6319 respectivamente, si se comparan estos resultados con la tendencia de los pronósticos (línea roja en las gráficas) se puede ver que los pronósticos tienden a estabilizarse aproximadamente en estos valores medios; se observa un caso particular entre las covarianzas de las variables BColombia - Grupo Sura en el que las covarianzas aumentan de una manera casi lineal positiva constante.

En la Figura **5-3** se muestran las gráficas de comparación de las covarianzas reales o iniciales con los pronósticos de los datos de entrenamiento y de prueba calculadas con Python (Van Rossum & Drake Jr, 2009); se puede apreciar que las predicciones de los datos de entrenamiento y prueba tienden a solaparse con los datos reales, lo que puede indicar una buena metodología de predicción.

Los resultados del criterio de desempeño del error medio porcentual absoluto (MAPE) se muestran en la Tabla **5-5**, en ésta se puede apreciar que los menores resultados de las volati-

lidades (varianzas) y las covarianzas se obtuvieron con la metodología de la DCC-MGARCH, lo que indica buena exactitud del método para la construcción ajustada de los valores. El rango del IC al 95 % tanto para las varianzas como para las covarianzas fue menor con la RN-LSTM, lo que puede indicar mayor exigencia en los cálculos realizados a la metodología implementada pero, esto puede ocasionar mayor error de predicción.

Tabla 5-5.: Error medio porcentual absoluto (MAPE) de los pronósticos para los cuatro activos con ambas metodologías DCC-MGARCH en R (R Core Team, 1990) y RN-LSTM en Python (Van Rossum & Drake Jr, 2009).

	Resultados DCC-MGARCH			Resultados RN-LSTM		
Varianzas	**MAPE**	**IC 95 %**	**Rango IC**	**MAPE**	**IC 95 %**	**Rango IC**
BColombia	0.0566	0.0539/0.0593	0.0054	1.6876	1.6864/1.6888	0.0024
Corficol	0.4655	0.1551/0.7759	0.6208	0.6154	0.4581/0.7727	0.3146
Grupo Sura	0.1977	0.1346/0.2607	0.1261	0.4793	0.4785/0.4800	0.0015
Isa	0.3607	-0.1352/0.8565	0.7213	0.4713	0.2050/0.7376	0.5326
Covarianzas	**MAPE**	**IC 95 %**	**Rango IC**	**MAPE**	**IC 95 %**	**Rango IC**
BColombia/Corficol	0.1480	0.1317/0.1642	0.0325	0.2468	0.2361/0.2576	0.0214
BColombia/Grupo Sura	0.1047	0.0872/0.1222	0.035	0.0121	0.0120/0.0122	0.0002
BColombia/Isa	0.1903	0.1435/0.2370	0.0935	0.2202	0.1925/0.2478	0.0553
Corficol/Grupo Sura	0.2105	0.1781/0.2409	0.0628	0.2455	0.2291/0.2619	0.0328
Corficol/Isa	0.3155	0.2367/0.3942	0.1575	0.4373	0.3945/0.4802	0.0857
Grupo Sura/Isa	0.2557	0.2010/0.3104	0.1094	0.2196	0.1960/0.2432	0.0472

El error cuadrático medio (RMSE) se muestra en la Tabla **5-6**, donde se puede apreciar que la metodología de la RN-LSTM tuvo menores valores de este criterio para las volatilidades o varianzas y las covarianzas, con lo que se puede inferir una menor cantidad de error que hubo entre los datos pronosticados y reales, dicho de otra manera, una mejor precisión o mejor ajuste absoluto del modelo a los datos. El rango del IC al 95 % tanto para las varianzas como para las covarianzas fue menor con la RN-LSTM.

La Tabla **5-7** muestra los resultados de la desviación absoluta media (MAD), con la metodología DCC-MGARCH en R (R Core Team, 1990) se obtuvieron menores valores para las volatilidades o varianzas para los activo Corficol, Grupo Sura e Isa y en Python (Van Rossum & Drake Jr, 2009) con la metodología RN-LSTM estos valores del MAD fueron más pequeños para el activo BColombia lo que indica que los puntos pronosticados y reales están más cerca o dicho de otra manera hay más precisión; los menores valores de la covarianzas con la metodología DCC-MGARCH fueron para BColombia-Corficol, BColombia-Isa, Corficol-Isa y con la metodología RN-LSTM fue para BColombia-Grupo Sura. El rango del IC al 95 % para las varianzas calculadas con la metodología DCC-MGARCH fue menor para los acti-

Tabla 5-6.: Error cuadrático medio (RMSE) de los pronósticos para los cuatro activos con ambas metodologías DCC-MGARCH en R (R Core Team, 1990) y RN-LSTM en Python (Van Rossum & Drake Jr, 2009)

	Resultados DCC-MGARCH			Resultados RN-LSTM		
Varianzas	**RMSE**	**IC 95 %**	**Rango IC**	**RMSE**	**IC 95 %**	**Rango IC**
BColombia	0.1547	0.1520/0.1574	0.0054	0.0509	0.0497/0.0521	0.0024
Corficol	1.5790	1.2686/1.8894	0.6208	0.4054	0.2475/0.5632	0.3157
Grupo Sura	0.3690	0.3059/0.4321	0.1262	0.0315	0.0307/0.0322	0.0015
Isa	2.8684	2.3725/3.3642	0.9917	0.9147	0.6512/1.1782	0.5270
Covarianzas	**RMSE**	**IC 95 %**	**Rango IC**	**RMSE**	**IC 95 %**	**Rango IC**
BColombia/Corficol	0.0909	0.0746/0.1071	0.0325	0.0232	0.0130/0.0333	0.0203
BColombia/Grupo Sura	0.1032	0.0857/0.1207	0.0350	0.0161	0.0157/0.0165	0.0008
BColombia/Isa	0.2655	0.2187/0.3122	0.0935	0.1159	0.0893/0.1425	0.0532
Corficol/Grupo Sura	0.3223	0.1297/0.1926	0.0629	0.0356	0.0200/0.0511	0.0311
Corficol/Isa	0.4153	0.3365/0.4941	0.1576	0.1176	0.0781/0.1570	0.0789
Grupo Sura/Isa	0.6532	0.2719/0.3813	0.1094	0.1003	0.0776/0.1230	0.0454

vos Corficol e Isa y fue menor con la metodología RN-LSTM para los activos BColombia y Grupo Sura; el rango del IC al 95 % para las covarianzas fue menor con la metodología DCC-MGARCH con excepción de la covarianza BColombia-Grupo Sura la cual fue menor con la metodología RN-LSTM.

La suma acumulada de los errores de pronóstico (CFE por sus siglas en inglés), para los retornos de los activos BColombia - Corficol, BColombia - Grupo Sura, BColombia - Isa, Corficol - Grupo Sura, Corficol - Isa y Grupo Sura - Isa fueron de 0.4652, 4.0081, 1.1374, -5.4615, -2.7083 y 6.1126 respectivamente, el CFE para los retornos de Grupo Sura - Isa y BColombia - Grupo Sura fueron más grandes que para los demás retornos, lo que indica que la mayoría de los pronósticos están por debajo de los valores reales, aunque este valor no es grande puede llegar a indicar un leve error sistemático o sesgo en el cálculo de los pronósticos a causa de una deficiencia en el diseño de los análisis; la Figura **5-4** muestra el comportamiento de los errores de los pronósticos con el modelo DCC-MGARCH.

Al igual que con el modelo DCC-MGARCH, los datos con el modelo RN-LSTM se dividieron en dos grupos, uno de entrenamiento y otro de prueba, a cada uno de estos dos grupos se les realizó las transformaciones correspondientes con las respectivas librerías, luego de las transformaciones se procede a entrenar la red neuronal, que está compuesta por dos capas LSTM, se ensayaron diversas cantidad de capas LSTM desde uno hasta cinco pero se obtuvieron mejores resultados con dos capas LSTM, dentro de estas dos capas se pusieron cuatro neuronas, se realizaron ensayos con cantidades de neuronas diferentes dentro de cada capa

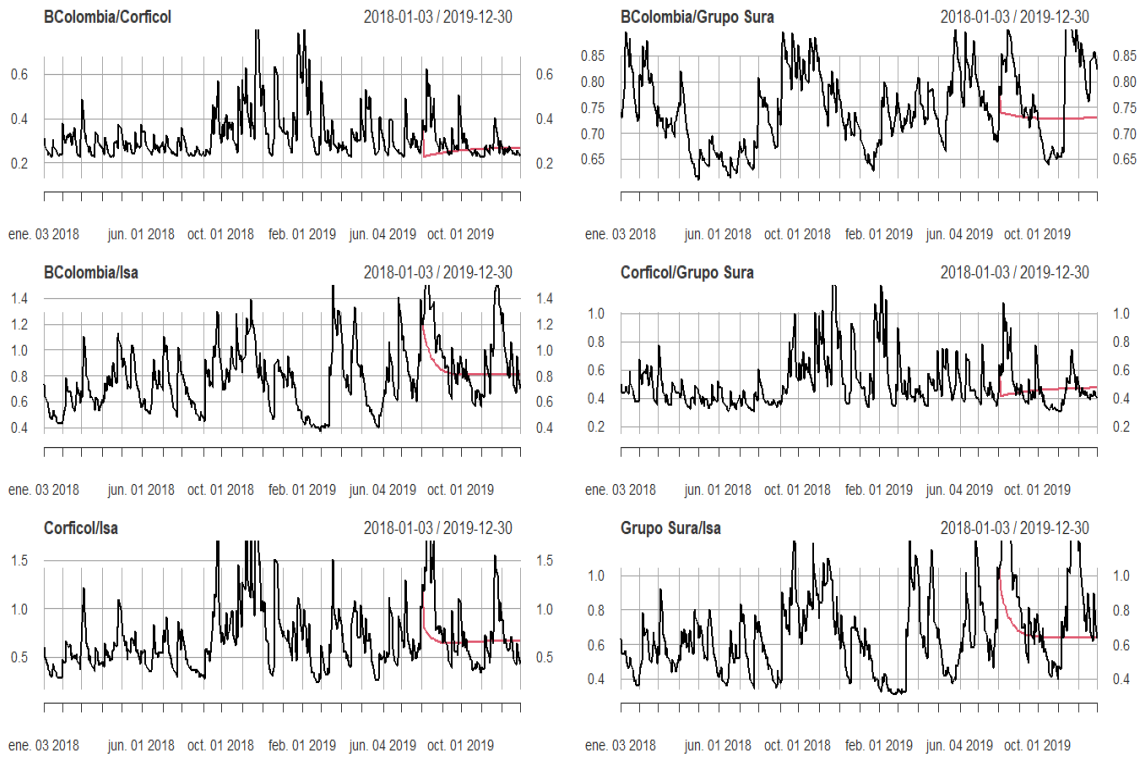

Figura 5-2.: Gráficas de covarianzas para los cuatro activos con R (R Core Team, 1990) comprendidas entre las fechas 2018/01/02 - 2019/12/30.

Tabla 5-7.: Desviación absoluta media (MAD) de los pronósticos para los cuatro activos con ambas metodologías DCC-MGARCH en R (R Core Team, 1990) y RN-LSTM en Python (Van Rossum y Drake Jr, 2009).

	Resultados DCC-MGARCH			Resultados RN-LSTM		
Varianzas	**MAD**	**IC 95 %**	**Rango IC**	**MAD**	**IC 95 %**	**Rango IC**
BColombia	0.0188	0.0159/0.0217	0.0058	0.0019	0.0007/0.0031	0.0024
Corficol	0.1787	0.1398/0.2175	0.0778	0.7555	0.5977/0.9134	0.3157
Grupo Sura	0.0081	0.0052/0.0111	0.0059	0.0126	0.0114/0.0138	0.0024
Isa	0.0111	-0.1086/0.1308	0.0222	0.9902	0.8817/1.0988	0.2171
Covarianzas	**MAD**	**IC 95 %**	**Rango IC**	**MAD**	**IC 95 %**	**Rango IC**
BColombia/Corficol	0.0101	0.0079/0.0123	0.0044	0.0473	0.0371/0.0575	0.0204
BColombia/Grupo Sura	0.0010	0.0004/0.0015	0.0011	0.0004	0.0000/0.0008	0.0008
BColombia/Isa	0.0025	-0.0102/0.0151	0.0049	0.1533	0.1267/0.1799	0.0532
Corficol/Grupo Sura	0.0147	0.0115/0.0179	0.0064	0.0715	0.0560/0.0871	0.0311
Corficol/Isa	0.0140	0.0084/0.0197	0.0113	0.1722	0.1382/0.2240	0.0857
Grupo Sura/Isa	0.0033	-0.0079/0.0146	0.0067	0.1311	0.1328/0.2117	0.0789

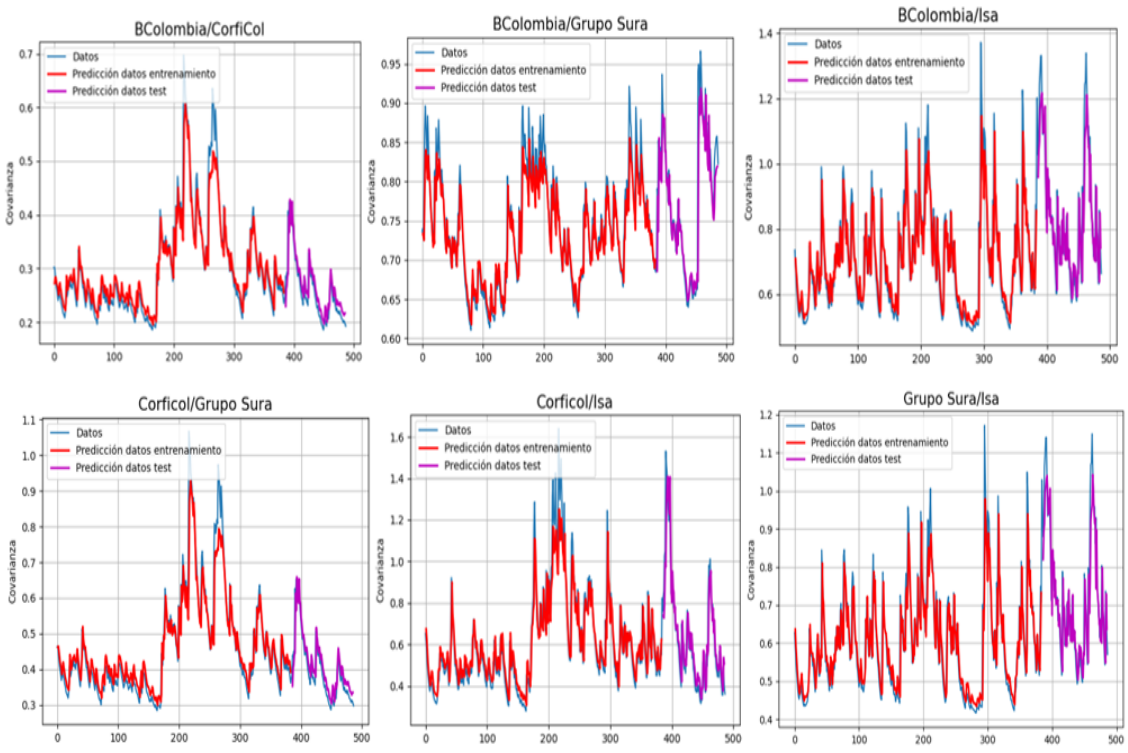

Figura 5-3.: Gráficas covarianzas con Python (Van Rossum & Drake Jr, 2009) comprendidas entre las fechas 2018/01/02 - 2019/12/30.

LSTM: 1, 2, 3, 4, 5, 10, 50 y la de mejor resultados fue con cuatro neuronas, posterior a entrenar la red neuronal se procede a realizar los pronósticos, tanto para los datos de entrenamiento como para los datos de prueba.

El CFE calculado con el modelo RN-LSTM, para las covarianzas de los activos BColombia - Corficol, BColombia - Grupo Sura, BColombia - Isa, Corficol - Grupo Sura, Corficol - Isa y Grupo Sura - Isa fueron de -12.4569, 18.2907, -40.0095, -26.6837, -36.4461 y -0.28072 respectivamente, estos valores están más alejados del cero en comparación con los calculados con el modelo DCC-MGARCH. El CFE de mayor valor encontrado con la RN-LSTM fue para BColombia - Grupo Sura lo que indica que la mayoría de los pronósticos están por debajo de los valores reales, los demás valores del CFE son negativos lo que indica que la mayoría de los pronósticos para estas covarianzas son superiores a sus valores reales. La Figura **5-5** muestra el comportamiento de los errores de los pronósticos con el modelo RN-LSTM.

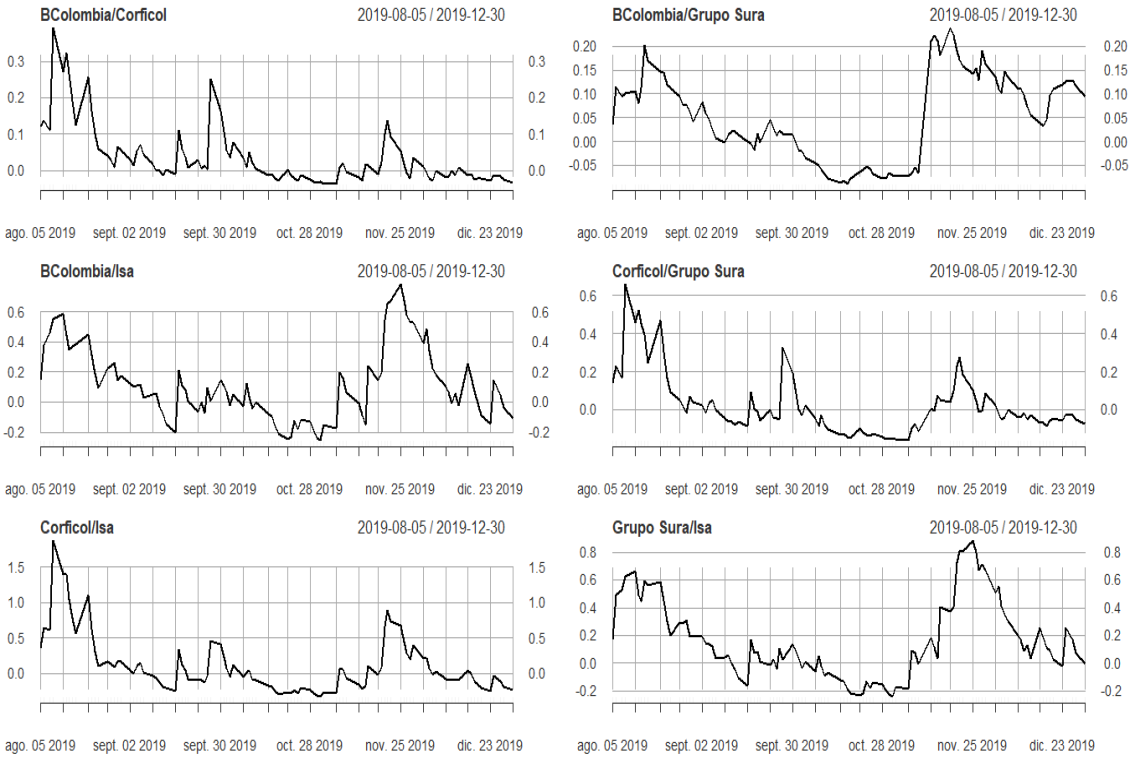

Figura 5-4.: Gráficas de los errores de pronósticos de las covarianzas con el modelo DCC-MGARCH comprendidas entre las fechas 2019/08/05 - 2019/12/30

Al comparar las gráficas **5-4** y **5-5** se puede apreciar que los pronósticos para los errores de las covarianzas fueron levemente mayores con la metodología DCC-MGARCH en comparación con los encontrados con la metodología de la RN-LSTM para los cuales sus pronósticos estuvieron por debajo de los valores reales. El rango de los cálculos de estos errores fue

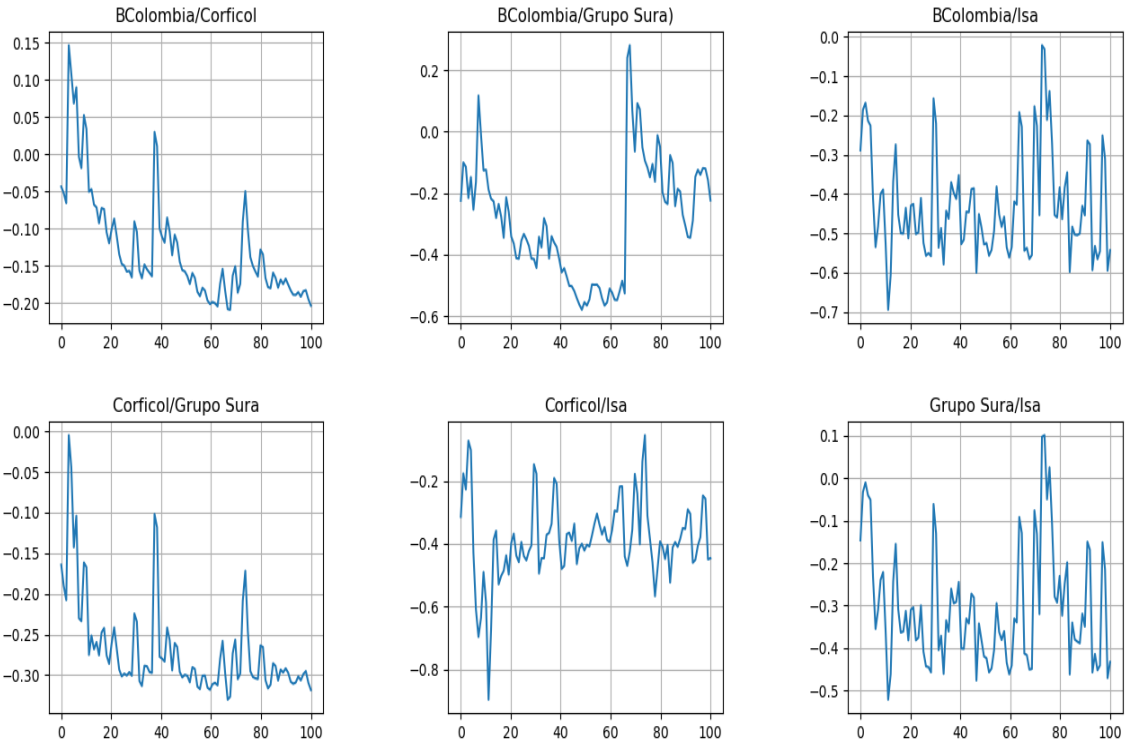

Figura 5-5.: Gráficas de los errores de pronósticos de las covarianzas con el modelo RN-LSTM comprendidas entre las fechas 2019/08/05 - 2019/12/30

menor con la metodología DCC-MGARCH para los activos BColombia - Grupo Sura y con la metodología de la RN-LSTM fue menor para los activos BColombia - Corficol, BColombia - Isa, Corficol - Grupo Sura, Corficol - Isa y Grupo Sura - Isa, esto puede llegar a indicar mejores pronósticos con la metodología RN-LSTM desarrollada con Python (Van Rossum & Drake Jr, 2009).

6. Conclusiones y recomendaciones

6.1. Conclusiones

En cualquier actividad económica hay algún grado de incertidumbre que se traduce en riesgos para un inversor. Dentro del periodo de análisis realizado en este trabajo fue el activo Isa (Interconexión eléctrica s.a.) el que tuvo el mayor riesgo posible de pérdida en su inversión, por lo que se puede deducir que para este activo pudo haber circunstancias especiales, hechos inesperados y/o contingentes ya sean políticos, económicos, sociales u otro, que hicieron de una posición de este activo más riesgosa en su inversión, cabe resaltar que en el periodo de este análisis ocurrió la contingencia de hidroituango (represa para la producción de energía) lo que pudo haber incrementado la incertidumbre a nivel nacional e internacional de la producción de energía en Colombia; fue el activo Grupo Sura el de menor riesgo por lo que no tuvo tantos factores influyendo en su inversión, dejando este activo como una buena opción de inversión.

Los resultados del criterio MAPE fueron mejores con el método DCC-MGARCH en R (R Core Team, 1990), solo un activo tuvo menor MAPE con el método de la RN-LSTM en Python (Van Rossum & Drake Jr, 2009), lo que puede permitir establecer que el desempeño de los pronósticos con el método DCC-MGARCH es un poco mejor desde el punto de vista de este criterio de desempeño .

El modelo de la RN-LSTM con Python (Van Rossum & Drake Jr, 2009) proporcionó unos resultados más pequeños del error cuadrático medio -RMSE-, indicando un mejor ajuste absoluto del modelo RN-LSTM a los datos, con esto se puede inferir que los datos pronosticados están más cerca de los datos reales tanto para las varianzas como las covarianzas, obteniendo mejor precisión en los pronósticos con la RN-LSTM.

Los valores más pequeños de la desviación absoluta de la media -MAD- ocurrieron para el modelo DCC-MGARCH lo que indica un mejor ajuste del modelo, con excepción de la varianza para activo BColombia y la covarianza para los activos BColombia - Grupo Sura el cual fue menor con el método de la RN-LSTM.

En los intervalos de confianza al 95 %, calculados con las dos metodologías se evidenció mejor precisión con la RN-LSTM en Python (Van Rossum & Drake Jr, 2009), ya que el rango

resultó más estrecho sobre los resultados, así este modelo ofrece una estimación de los pronósticos más precisa.

Las gráficas de los errores de las covarianzas muestran rangos mas reducidos para la mayoría de los activos lo que indica mejores resultados para la RN-LSTM desarrollada en Python (Van Rossum & Drake Jr, 2009).

6.2. Trabajo futuro

Un aspecto importante que puede fortalecer este trabajo realizado es ampliar el rango de fechas del cálculo de los retornos de los cuatro activos con el objetivo de verificar la influencia que pueden llegar a tener otros factores como lo fueron la pandemia COVID-19, elecciones presidenciales u otro.

Ampliar la cantidad de activos en este trabajo puede mejorar la perspectiva sobre la influencia que tienen cada uno de los activos sobre los demás para el cálculo de las covarianzas.

Como objetivo de un próximo trabajo relacionado con la comparación de metodologías, se recomienda utilizar otro tipo de redes neuronales con el fin de tener otro enfoque en las comparaciones de dichas metodologías para el cálculo de los pronósticos.

A. Anexo: Código requerido para obtener los resultados del modelo DCC-MGARCH en R.

- Librerías requeridas.

```
library(quantmod)
library(tseries)
library(fImport)
library(rugarch)
library(rmgarch)
library(gtools)
library(dplyr)
library(readxl)
library(xts)
library(zoo)
library(MLmetrics)
library(forecast)
library(BSDA)
library(modeest)
library(fImport)
library(PerformanceAnalytics)
library(highcharter)
library(Risk)
```

- Lecturas de los datos y cálculo de los rendimientos.

```
Datos <- read_xls(ruta_archivo)
BBCOLOMBIA <- Datos[,1:2]
BBCOLOMBIA$fecha <- as.Date(BBCOLOMBIA$fecha)
BBCOLOMBIA <- xts(BBCOLOMBIA$PC_BBCOLOMBIA, BBCOLOMBIA$fecha)
CORFICOL = Datos[c("fecha", "PC_CORFICOL")]
```

```
CORFICOL$fecha <- as.Date(CORFICOL$fecha)
CORFICOL <- xts(CORFICOL$PC_CORFICOL, CORFICOL$fecha)
GRUPOSURA = Datos[c("fecha", "PC_GRUPOSURA")]
GRUPOSURA$fecha <- as.Date(GRUPOSURA$fecha)
GRUPOSURA <- xts(GRUPOSURA$PC_GRUPOSURA, GRUPOSURA$fecha)
ISA = Datos[c("fecha", "PC_ISA")]
ISA$fecha <- as.Date(ISA$fecha)
ISA <- xts(ISA$PC_ISA, ISA$fecha)
Activos <- merge(BBCOLOMBIA, CORFICOL, GRUPOSURA, ISA)
rtos = na.omit(diff(log(Activos))*100)
```

- Gráficas de los precios de las acciones.

```
PA = matrix(c(1:4), nrow=2, byrow=TRUE)
lay.1 = layout(PA)
layout.show(lay.1)
plot(Activos$BBCOLOMBIA, main = "Precio de las acciones Bancolombia", xlab =
"Tiempo")
plot(Activos$CORFICOL, main = "Precio de las acciones Corficol", xlab = "Tiem-
po")
plot(Activos$GRUPOSURA, main = "Precio de las acciones Grupo Sura", xlab =
"Tiempo")
plot(Activos$ISA, main = "Precio de las acciones Isa", xlab = "Tiempo")
```

- Test de Shapiro-Wilks.

```
SW_BColombia = shapiro.test(as.vector(rtos$BBCOLOMBIA))
SW_Corficol = shapiro.test(as.vector(rtos$CORFICOL))
SW_GS = shapiro.test(as.vector(rtos$GRUPOSURA))
SW_Isa = shapiro.test(as.vector(rtos$ISA))
```

- Gráficas de los rendimientos de los datos completos:

```
DC <- as.timeSeries(rtos)
plot(DC, main = Rendimientos", xlab = "Tiempo")
```

- Fijar y ajustar el modelo DCC-MGARCH.

```
garch11 <- ugarchspec(mean.model = list(armaOrder = c(0, 0)), variance.model =
list(garchOrder = c(1, 1),model = "sGARCH"), distribution="std")
DCC_Garch11 = dccspec(uspec = multispec(replicate(4, garch11)), dccOrder = c(1,
1), distribution = "mvt")
dcc.fit_DC <- dccfit(DCC_Garch11, data = DC, solver.control=list(trace=0))
```

- Cálculo de las matrices de varianzas covarianzas en forma de vector. Para las próximas lineas V1, V2, V3 y V4 representan a Bancolombia, Corficol, Grupo Sura y Isa respectivamente.

```
dcc.fit_DC@mfit$H
CovV1_V2 <-(dcc.fit_DC@mfit$H[1:1,2:2,])
CovV1_V3 <-(dcc.fit_DC@mfit$H[1:1,3:3,])
CovV1_V4 <-(dcc.fit_DC@mfit$H[1:1,4:4,])
CovV2_V3 <-(dcc.fit_DC@mfit$H[2:2,3:3,])
CovV2_V4 <-(dcc.fit_DC@mfit$H[2:2,4:4,])
CovV3_V4 <-(dcc.fit_DC@mfit$H[3:3,4:4,])
Varianza_BBC <- (dcc.fit_DC@mfit$H[1:1,1:1,])
Varianza_Corficol <- (dcc.fit_DC@mfit$H[2:2,2:2,])
Varianza_GS <- (dcc.fit_DC@mfit$H[3:3,3:3,])
Varianza_Isa <- (dcc.fit_DC@mfit$H[4:4,4:4,])
```

- Dividir los vectores anteriores en los dos grupos, prueba y ensayo.

```
Grupo 1, 387 datos:
Primeras_Cov <- dcc.fit_DC@mfit$H[„1:387]
Prim_v1v2 = Primeras_Cov[1:1,2:2,]
Prim_v1v3 = Primeras_Cov[1:1,3:3,]
Prim_v1v4 = Primeras_Cov[1:1,4:4,]
Prim_v2v3 = Primeras_Cov[2:2,3:3,]
Prim_v2v4 = Primeras_Cov[2:2,4:4,]
Prim_v3v4 = Primeras_Cov[3:3,4:4,]
Prim_Var1 = Primeras_Cov[1:1,1:1,]
Prim_Var2 = Primeras_Cov[2:2,2:2,]
Prim_Var3 = Primeras_Cov[3:3,3:3,]
```

```
Prim_Var4 = Primeras_Cov[4:4,4:4,]


Grupo 2, 100 datos:
Ultimas_Cov <- dcc.fit_DC@mfit$H[,,388:487]
Ulti_v1v2 = Ultimas_Cov[1:1,2:2,]
Ulti_v1v3 = Ultimas_Cov[1:1,3:3,]
Ulti_v1v4 = Ultimas_Cov[1:1,4:4,]
Ulti_v2v3 = Ultimas_Cov[2:2,3:3,]
Ulti_v2v4 = Ultimas_Cov[2:2,4:4,]
Ulti_v3v4 = Ultimas_Cov[3:3,4:4,]
Ulti_Var1 = Ultimas_Cov[1:1,1:1,]
Ulti_Var2 = Ultimas_Cov[2:2,2:2,]
Ulti_Var3 = Ultimas_Cov[3:3,3:3,]
Ulti_Var4 = Ultimas_Cov[4:4,4:4,]
```

- Predicciones con los datos del grupo 1.

```
dcc.fit_G1@mfit$H
Pred <-(dccforecast(dcc.fit_G1, n.ahead=100))
Pred_Cov <-Pred@mforecast$H
summary(dcc.fit_G1@mfit$H)
```

- Cálculo del VaR y ES:

```
VaR_BBC = VaR(RtoBBC, p = c(0.95, 0.99), method = "historical")
VaR_Corf = VaR(RtoCorf, p = c(0.95, 0.99), method = "historical")
VaR_GS = VaR(RtoGS, p = c(0.95, 0.99), method = "historical")
VaR_ISA = VaR(RtoISA, p = c(0.95, 0.99), method = "historical")


ES_BBC = ES(RtoBBC, method = "historical")
ES_Corf = ES(RtoCorf, method = "historical")
ES_GS = ES(RtoGS, method = "historical")
ES_ISA = ES(RtoISA, method = "historical")
```

- Separar las matrices pronosticadas en vectores. Luego de realizar los pronósticos como se mostró en líneas anteriores, estos fueron descargados al pc con el fin de organizar el

archivo Excel.

```
PronV1_V2 <- read_xlsx(rut_arch)
PronV1_V2 <- as.numeric(unlist(PronV1_V2))
PronV1_V3 <- read_xlsx(rut_arch1)
PronV1_V3 <- as.numeric(unlist(PronV1_V3))
PronV1_V4 <- read_xlsx(rut_arch2)
PronV1_V4 <- as.numeric(unlist(PronV1_V4))
PronV2_V3 <- read_xlsx(rut_arch3)
PronV2_V3 <- as.numeric(unlist(PronV2_V3))
PronV2_V4 <- read_xlsx(rut_arch4)
PronV2_V4 <- as.numeric(unlist(PronV2_V4))
PronV3_V4 <- read_xlsx(rut_arch5)
PronV3_V4 <- as.numeric(unlist(PronV3_V4))
```

```
PronVar1 <- read_xlsx(rut_arch6)
PronVar1 <- as.numeric(unlist(PronVar1))
PronVar2 <- read_xlsx(rut_arch7)
PronVar2 <- as.numeric(unlist(PronVar2))
PronVar3 <- read_xlsx(rut_arch8)
PronVar3 <- as.numeric(unlist(PronVar3))
PronVar4 <- read_xlsx(rut_arch9)
PronVar4 <- as.numeric(unlist(PronVar4))
```

- Gráficas de las varianzas covarianzas.

```
Conf3x2 = matrix(c(1:6), nrow=3, byrow=TRUE)
lay.1 = layout(Conf3x2)
layout.show(lay.1)
```

```
CovV1_V2
PriCovV1V2
PronV1_V2
DatosV1_V2 <- c(PriCovV1V2, PronV1_V2)
Fecha = Datos[c("fecha")]
Fecha = Fecha[-1, ]
Graf_DC <- cbind(Fecha, CovV1_V2)
Graf_DC$fecha <- as.Date(Graf_DC$fecha)
```

```
Graf_DC <- xts(Graf_DC$CovV1_V2, Graf_DC$fecha)
CompV1V2 <- cbind(Fecha, DatosV1_V2)
CompV1V2$fecha <- as.Date(CompV1V2$fecha)
CompV1V2 <- xts(CompV1V2$DatosV1_V2, CompV1V2$fecha)
todoV1V2 <- cbind(Graf_DC, CompV1V2)
plot(todoV1V2, ylim = c(0.1, 0.8), main = "BColombia/Corficol")


CovV1_V3
PriCovV1V3
PronV1_V3
DatosV1_V3 <- c(PriCovV1V3, PronV1_V3)
Graf_DCV1_V3 <- cbind(Fecha, CovV1_V3)
Graf_DCV1_V3$fecha <- as.Date(Graf_DCV1_V3$fecha)
Graf_DCV1_V3 <- xts(Graf_DCV1_V3$CovV1_V3, Graf_DCV1_V3$fecha)
CompV1V3 <- cbind(Fecha, DatosV1_V3)
CompV1V3$fecha <- as.Date(CompV1V3$fecha)
CompV1V3 <- xts(CompV1V3$DatosV1_V3, CompV1V3$fecha)
todoV1V3 <- cbind(Graf_DCV1_V3, CompV1V3)
plot(todoV1V3, ylim = c(0.6, 0.9), main = "BColombia/Grupo Sura")


CovV1_V4
PriCovV1V4
PronV1_V4
DatosV1_V4 <- c(PriCovV1V4, PronV1_V4)
Graf_DCV1_V4 <- cbind(Fecha, CovV1_V4)
Graf_DCV1_V4$fecha <- as.Date(Graf_DCV1_V4$fecha)
Graf_DCV1_V4 <- xts(Graf_DCV1_V4$CovV1_V4, Graf_DCV1_V4$fecha)
CompV1V4 <- cbind(Fecha, DatosV1_V4)
CompV1V4$fecha <- as.Date(CompV1V4$fecha)
CompV1V4 <- xts(CompV1V4$DatosV1_V4, CompV1V4$fecha)
todoV1V4 <- cbind(Graf_DCV1_V4, CompV1V4)
plot(todoV1V4, ylim = c(0.3, 1.5), main = "BColombia/Isa")


CovV2_V3
PriCovV2V3
PronV2_V3
DatosV2_V3 <- c(PriCovV2V3, PronV2_V3)
Graf_DCV2_V3 <- cbind(Fecha, CovV2_V3)
Graf_DCV2_V3$fecha <- as.Date(Graf_DCV2_V3$fecha)
```

```
Graf_DCV2_V3 <- xts(Graf_DCV2_V3$CovV2_V3, Graf_DCV2_V3$fecha)
CompV2V3 <- cbind(Fecha, DatosV2_V3)
CompV2V3$fecha <- as.Date(CompV2V3$fecha)
CompV2V3 <- xts(CompV2V3$DatosV2_V3, CompV2V3$fecha)
todoV2V3 <- cbind(Graf_DCV2_V3, CompV2V3)
plot(todoV2V3, ylim = c(0.1, 1.2), main = Çorficol/Grupo Sura")


CovV2_V4
PriCovV2V4
PronV2_V4
DatosV2_V4 <- c(PriCovV2V4, PronV2_V4)
Graf_DCV2_V4 <- cbind(Fecha, CovV2_V4)
Graf_DCV2_V4$fecha <- as.Date(Graf_DCV2_V4$fecha)
Graf_DCV2_V4 <- xts(Graf_DCV2_V4$CovV2_V4, Graf_DCV2_V4$fecha)
CompV2V4 <- cbind(Fecha, DatosV2_V4)
CompV2V4$fecha <- as.Date(CompV2V4$fecha)
CompV2V4 <- xts(CompV2V4$DatosV2_V4, CompV2V4$fecha)
todoV2V4 <- cbind(Graf_DCV2_V4, CompV2V4)
plot(todoV2V4, ylim = c(0.1, 1.7), main = Çorficol/Isa")


CovV3_V4
PriCovV3V4
PronV3_V4
DatosV3_V4 <- c(PriCovV3V4, PronV3_V4)
Graf_DCV3_V4 <- cbind(Fecha, CovV3_V4)
Graf_DCV3_V4$fecha <- as.Date(Graf_DCV3_V4$fecha)
Graf_DCV3_V4 <- xts(Graf_DCV3_V4$CovV3_V4, Graf_DCV3_V4$fecha)
CompV3V4 <- cbind(Fecha, DatosV3_V4)
CompV3V4$fecha <- as.Date(CompV3V4$fecha)
CompV3V4 <- xts(CompV3V4$DatosV3_V4, CompV3V4$fecha)
todoV3V4 <- cbind(Graf_DCV3_V4, CompV3V4)
plot(todoV3V4, ylim = c(0.3, 1.2), main = "Grupo Sura/Isa")


legend("topright", legend = c("Datos reales","Pronósticos"), lty = c(1, 1), col = c(1,
2), lwd = 2)
```

- Índice de confianza (IC) para los pronósticos.

```
t.test(PronV1_V2, conf.level=0.95)$conf.int
t.test(PronV1_V3, conf.level=0.95)$conf.int
t.test(PronV1_V4, conf.level=0.95)$conf.int
t.test(PronV2_V3, conf.level=0.95)$conf.int
t.test(PronV2_V4, conf.level=0.95)$conf.int
t.test(PronV3_V4, conf.level=0.95)$conf.int
```

- Validación de los pronósticos.

  - MAPE.

```
n = 100
(Mape_V1_V2 <- MAPE(PronV1_V2, G2CovV1_V2))
DEV1_V2 <- sd((G2CovV1_V2 - PronV1_V2))
zsum.test(Mape_V1_V2, DEV1_V2, n, conf.level=0.95)

(Mape_V1_V3 <- MAPE(PronV1_V3, G2CovV1_V3))
DEV1_V3 <- sd((G2CovV1_V3 - PronV1_V3))
zsum.test(Mape_V1_V3, DEV1_V3, n, conf.level=0.95)

Mape_V1_V4 <- MAPE(PronV1_V4, G2CovV1_V4)
DEV1_V4 <- sd((G2CovV1_V4 - PronV1_V4))
zsum.test(Mape_V1_V4, DEV1_V4, n, conf.level=0.95)

(Mape_V2_V3 <- MAPE(PronV2_V3, G2CovV2_V3))
DEV2_V3 <- sd((G2CovV2_V3 - PronV2_V3))
zsum.test(Mape_V2_V3, DEV2_V3, n, conf.level=0.95)

(Mape_V2_V4 <- MAPE(PronV2_V4, G2CovV2_V4))
DEV2_V4 <- sd((G2CovV2_V4 - PronV2_V4))
zsum.test(Mape_V2_V4, DEV2_V4, n, conf.level=0.95)

(Mape_V3_V4 <- MAPE(PronV3_V4, G2CovV3_V4))
DEV3_V4 <- sd((G2CovV3_V4 - PronV3_V4))
zsum.test(Mape_V3_V4, DEV3_V4, n, conf.level=0.95)
```

```
(MapeVarBBC <- MAPE(PronVar1, Ulti_Var1))
DEVarBBC <- sd((Ulti_Var1 - PronVar1))
zsum.test(MapeVarBBC, DEVarBBC, n, conf.level=0.95)

(MapeVarCorf <- MAPE(PronVar2, Ulti_Var2))
DEVarCorf <- sd((Ulti_Var2 - PronVar2))
zsum.test(MapeVarCorf, DEVarCorf, n, conf.level=0.95)

(MapeVarGS <- MAPE(PronVar3, Ulti_Var3))
DEVarGS <- sd((Ulti_Var3 - PronVar3))
zsum.test(MapeVarGS, DEVarGS, n, conf.level=0.95)

(MapeVarIsa <- MAPE(PronVar4, Ulti_Var4))
DEVarIsa <- sd((Ulti_Var4 - PronVar4))
zsum.test(MapeVarIsa, DEVarIsa, n, conf.level=0.95)
```

- RMSE.

```
(sqrt(mean((G2CovV1_V2 - PronV1_V2)^2)))
(rmseV1_V2 <- RMSE(G2CovV1_V2, PronV1_V2))
DErmseV1_V2 <- sd(G2CovV1_V2 - PronV1_V2)
zsum.test(rmseV1_V2, DErmseV1_V2, n, conf.level=0.95)

(sqrt(mean((G2CovV1_V3 - PronV1_V3)^2)))
(rmseV1_V3 <- RMSE(G2CovV1_V3, PronV1_V3))
DErmseV1_V3 <- sd(G2CovV1_V3 - PronV1_V3)
zsum.test(rmseV1_V3, DErmseV1_V3, n, conf.level=0.95)

(sqrt(mean((G2CovV1_V4 - PronV1_V4)^2)))
(rmseV1_V4 <- RMSE(G2CovV1_V4, PronV1_V4))
DErmseV1_V4 <- sd(G2CovV1_V4 - PronV1_V4)
zsum.test(rmseV1_V4, DErmseV1_V4, n, conf.level=0.95)

(sqrt(mean((G2CovV2_V3 - PronV2_V3)^2)))
(rmseV2_V3 <- RMSE(G2CovV2_V3, PronV2_V3))
DErmseV2_V3 <- sd(G2CovV2_V3 - PronV2_V3)
zsum.test(rmseV2_V3, DErmseV2_V3, n, conf.level=0.95)
```

```
(sqrt(mean((G2CovV2_V4 - PronV2_V4)^2)))
(rmseV2_V4 <- RMSE(G2CovV2_V4, PronV2_V4))
DErmseV2_V4 <- sd(G2CovV2_V4 - PronV2_V4)
zsum.test(rmseV2_V4, DErmseV2_V4, n, conf.level=0.95)


(sqrt(mean((G2CovV3_V4 - PronV3_V4)^2)))
(rmseV3_V4 <- RMSE(G2CovV3_V4, PronV3_V4))
DErmseV3_V4 <- sd(G2CovV3_V4 - PronV3_V4)
zsum.test(rmseV3_V4, DErmseV3_V4, n, conf.level=0.95)


(sqrt(mean((Ulti_Var1 - PronVar1)^2)))
(rmseVarBBC <- RMSE(Ulti_Var1, PronVar1))
DErmseVarBBC <- sd(Ulti_Var1 - PronVar1)
zsum.test(rmseVarBBC, DErmseVarBBC, n, conf.level=0.95)


(sqrt(mean((Ulti_Var2 - PronVar2)^2)))
(rmseVarCorf <- RMSE(Ulti_Var2, PronVar2))
DErmseVarCorf <- sd(Ulti_Var2 - PronVar2)
zsum.test(rmseVarCorf, DErmseVarCorf, n, conf.level=0.95)


(sqrt(mean((Ulti_Var3 - PronVar3)^2)))
(rmseVarGS <- RMSE(Ulti_Var3, PronVar3))
DErmseVarGS <- sd(Ulti_Var3 - PronVar3)
zsum.test(rmseVarGS, DErmseVarGS, n, conf.level=0.95)


(sqrt(mean((Ulti_Var4 - PronVar4)^2)))
(rmseVarIsa <- RMSE(Ulti_Var4, PronVar4))
DErmseVarIsa <- sd(Ulti_Var4 - PronVar4)
zsum.test(rmseVarIsa, DErmseVarIsa, n, conf.level=0.95)
```

- MAD.

```
(madV1_V2 <- mad(PronV1_V2))
mediaV1_V2 <- mean(PronV1_V2)
DEmadV1_V2 <- sd(PronV1_V2 - mediaV1_V2)
zsum.test(madV1_V2, DEmadV1_V2, n, conf.level=0.95)
```

```
(madV1_V3 <- mad(PronV1_V3))
mediaV1_V3 <- mean(PronV1_V3)
DEmadV1_V3 <- sd(PronV1_V3 - mediaV1_V3)
zsum.test(madV1_V3, DEmadV1_V3, n, conf.level=0.95)

(madV1_V4 <- mad(PronV1_V4))
mediaV1_V4 <- mean(PronV1_V4)
DEmadV1_V4 <- sd(PronV1_V4 - mediaV1_V4)
zsum.test(madV1_V4, DEmadV1_V4, n, conf.level=0.95)

(madV2_V3 <- mad(PronV2_V3))
mediaV2_V3 <- mean(PronV2_V3)
DEmadV2_V3 <- sd(PronV2_V3 - mediaV2_V3)
zsum.test(madV2_V3, DEmadV2_V3, n, conf.level=0.95)

(madV2_V4 <- mad(PronV2_V4))
mediaV2_V4 <- mean(PronV2_V4)
DEmadV2_V4 <- sd(PronV2_V4 - mediaV2_V4)
zsum.test(madV2_V4, DEmadV2_V4, n, conf.level=0.95)

(madV3_V4 <- mad(PronV3_V4))
mediaV3_V4 <- mean(PronV3_V4)
DEmadV3_V4 <- sd(PronV3_V4 - mediaV3_V4)
zsum.test(madV3_V4, DEmadV3_V4, n, conf.level=0.95)

(madVarBBC <- mad(PronVar1))
mediaVarBBC <- mean(PronVar1)
DEmadVarBBC <- sd(PronVar1 - mediaVarBBC)
zsum.test(madVarBBC, DEmadVarBBC, n, conf.level=0.95)

(madVarCorf <- mad(PronVar2))
mediaVarCorf <- mean(PronVar2)
DEmadVarCorf <- sd(PronVar2 - mediaVarCorf)
zsum.test(madVarCorf, DEmadVarCorf, n, conf.level=0.95)

(madVarGS <- mad(PronVar3))
mediaVarGS <- mean(PronVar3)
DEmadVarGS <- sd(PronVar3 - mediaVarGS)
```

```
zsum.test(madVarGS, DEmadVarGS, n, conf.level=0.95)


(madVarIsa <- mad(PronVar4))
mediaVarIsa <- mean(PronVar4)
DEmadVarIsa <- sd(PronVar4 - mediaVarIsa)
zsum.test(madVarIsa, DEmadVarIsa, n, conf.level=0.95)
```

- Gráficas de los errores.

```
Conf2x3 = matrix(c(1:6), nrow=3, byrow=TRUE)
lay.2 = layout(Conf2x3)
layout.show(lay.2)


ErrorV1V2 <- (G2CovV1_V2 - PronV1_V2)
Graf_ErrorV1V2 <- cbind(Fechas, ErrorV1V2)
Graf_ErrorV1V2$fecha <- as.Date(Graf_ErrorV1V2$fecha)
Graf_ErrorV1V2 <- xts(Graf_ErrorV1V2$ErrorV1V2, Graf_ErrorV1V2$fecha)
plot(Graf_ErrorV1V2, main = "BColombia/Corficol")
sumaV1V2 = sum(ErrorV1V2)


(ErrorV1V3 <- (G2CovV1_V3 - PronV1_V3))
Graf_ErrorV1V3 <- cbind(Fechas, ErrorV1V3)
Graf_ErrorV1V3$fecha <- as.Date(Graf_ErrorV1V3$fecha)
Graf_ErrorV1V3 <- xts(Graf_ErrorV1V3$ErrorV1V3, Graf_ErrorV1V3$fecha)
plot(Graf_ErrorV1V3, main = "BColombia/Grupo Sura")
sumaV1V3 = sum(ErrorV1V3)


(ErrorV1V4 <- (G2CovV1_V4 - PronV1_V4))
Graf_ErrorV1V4 <- cbind(Fechas, ErrorV1V4)
Graf_ErrorV1V4$fecha <- as.Date(Graf_ErrorV1V4$fecha)
Graf_ErrorV1V4 <- xts(Graf_ErrorV1V4$ErrorV1V4, Graf_ErrorV1V4$fecha)
plot(Graf_ErrorV1V4, main = "BColombia/Isa")
sumaV1V4 = sum(ErrorV1V4)


(ErrorV2V3 <- (G2CovV2_V3 - PronV2_V3))
Graf_ErrorV2V3 <- cbind(Fechas, ErrorV2V3)
Graf_ErrorV2V3$fecha <- as.Date(Graf_ErrorV2V3$fecha)
Grafv_ErrorV2V3 <- xts(Graf_ErrorV2V3$ErrorV2V3, Graf_ErrorV2V3$fecha)
```

```
plot(Graf_ErrorV2V3, main = Çorficol/Grupo Sura")
sumaV2V3 = sum(ErrorV2V3)

(ErrorV2V4 <- (G2CovV2_V4 - PronV2_V4))
Graf_ErrorV2V4 <- cbind(Fechas, ErrorV2V4)
Graf_ErrorV2V4$fecha <- as.Date(Graf_ErrorV2V4$fecha)
Graf_ErrorV2V4 <- xts(Graf_ErrorV2V4$ErrorV2V4, Graf_ErrorV2V4$fecha)
plot(Graf_ErrorV2V4, main = Çorficol/Isa")
sumaV2V4 = sum(ErrorV2V4)

(ErrorV3V4 <- (G2CovV3_V4 - PronV3_V4))
Graf_ErrorV3V4 <- cbind(Fechas, ErrorV3V4)
Graf_ErrorV3V4$fecha <- as.Date(Graf_ErrorV3V4$fecha)
Graf_ErrorV3V4 <- xts(Graf_ErrorV3V4$ErrorV3V4, Graf_ErrorV3V4$fecha)
plot(Graf_ErrorV3V4, main = "Grupo Sura/Isa")
sumaV3V4 = sum(ErrorV3V4)
```

B. Anexo: Código requerido para obtener los resultados del modelo de la RN-LSTM en Python.

- Librerías requeridas.

```
import numpy as np
import matplotlib.pyplot as plt
from pandas import read_csv
import pandas as pd
import math
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
import scipy.stats as st
from statsmodels import robust
import os
```

- Función para generar los gráficos.

```
def create_dataset(dataset, look_back=1):
dataX, dataY = [], []
for i in range(len(dataset)-look_back-1):
a = dataset[i : (i + look_back), 0]
dataX.append(a)
dataY.append(dataset[i + look_back, 0])
return np.array(dataX), np.array(dataY)
np.random.seed(7)
```

- Lectura de los datos.

```
dataframeV1_V2 = pd.read_excel(dfV1_V2)
dataframeV1_V3 = pd.read_excel(dfV1_V3)
dataframeV1_V4 = pd.read_excel(dfV1_V4)
dataframeV2_V3 = pd.read_excel(dfV2_V3)
dataframeV2_V4 = pd.read_excel(dfV2_V4)
dataframeV3_V4 = pd.read_excel(dfV3_V4)

VarBBC = pd.read_excel(Varianza_BBC)
VarCorf = pd.read_excel(Varianza_Corficol)
VarGS = pd.read_excel(Varianza_GS)
VarIsa = pd.read_excel(Varianza_Isa)
```

- Asegurar que los datos sean valores numéricos.

```
datasetV1_V2 = dataframeV1_V2.values
datasetV1_V3 = dataframeV1_V3.values
datasetV1_V4 = dataframeV1_V4.values
datasetV2_V3 = dataframeV2_V3.values
datasetV2_V4 = dataframeV2_V4.values
datasetV3_V4 = dataframeV3_V4.values

datasetVarBBC = VarBBC.values
datasetVarCorf = VarCorf.values
datasetVarGS = VarGS.values
datasetVIsa = VarIsa.values
```

- Conversión de los datos a flotante con el objetivo de modelar la RN.

```
datasetV1_V2 = datasetV1_V2.astype('float32')
datasetV1_V3 = datasetV1_V3.astype('float32')
datasetV1_V4 = datasetV1_V4.astype('float32')
datasetV2_V3 = datasetV2_V3.astype('float32')
datasetV2_V4 = datasetV2_V4.astype('float32')
datasetV3_V4 = datasetV3_V4.astype('float32')
```

```
datasetVarBBC = datasetVarBBC.astype('float32')
datasetVarCorf = datasetVarCorf.astype('float32')
datasetVarGS = datasetVarGS.astype('float32')
datasetVarIsa = datasetVIsa.astype('float32')
```

- Normalización de los datos.

```
scalerV1_V2 = MinMaxScaler(feature_range=(0, 1))
datasetV1_V2 = scalerV1_V2.fit_transform(datasetV1_V2)
scalerV1_V3 = MinMaxScaler(feature_range=(0, 1))
datasetV1_V3 = scalerV1_V3.fit_transform(datasetV1_V3)
scalerV1_V4 = MinMaxScaler(feature_range=(0, 1))
datasetV1_V4 = scalerV1_V4.fit_transform(datasetV1_V4)
scalerV2_V3 = MinMaxScaler(feature_range=(0, 1))
datasetV2_V3 = scalerV2_V3.fit_transform(datasetV2_V3)
scalerV2_V4 = MinMaxScaler(feature_range=(0, 1))
datasetV2_V4 = scalerV2_V4.fit_transform(datasetV2_V4)
scalerV3_V4 = MinMaxScaler(feature_range=(0, 1))
datasetV3_V4 = scalerV3_V4.fit_transform(datasetV3_V4)

scalerVarBBC = MinMaxScaler(feature_range=(0, 1))
datasetVarBBC = scalerVarBBC.fit_transform(datasetVarBBC)
scalerVarCorf = MinMaxScaler(feature_range=(0, 1))
datasetVarCorf = scalerVarCorf.fit_transform(datasetVarCorf)
scalerVarGS = MinMaxScaler(feature_range=(0, 1))
datasetVarGS = scalerVarGS.fit_transform(datasetVarGS)
scalerVarIsa = MinMaxScaler(feature_range=(0, 1))
datasetVarIsa = scalerVarIsa.fit_transform(datasetVarIsa)
```

- División de los datos en dos grupos entrenamiento y test.

```
train_sizeV1_V2 = int(len(datasetV1_V2) * 0.791)
test_sizeV1_V2 = len(datasetV1_V2) - train_sizeV1_V2
trainV1_V2, testV1_V2 = datasetV1_V2[0:train_sizeV1_V2,:],
datasetV1_V2[train_sizeV1_V2:len(datasetV1_V2),:]

train_sizeV1_V3 = int(len(datasetV1_V3) * 0.791)
test_sizeV1_V3 = len(datasetV1_V3) - train_sizeV1_V3
```

```
trainV1_V3, testV1_V3 = datasetV1_V3[0:train_sizeV1_V3,:],
datasetV1_V3[trainv_sizeV1_V3:len(datasetV1_V3),:]

train_sizeV1_V4 = int(len(datasetV1_V4) * 0.791)
test_sizeV1_V4 = len(datasetV1_V4) - train_sizeV1_V4
trainV1_V4, testV1_V4 = datasetV1_V4[0:train_sizeV1_V4,:],
datasetV1_V4[train_sizeV1_V4:len(datasetV1_V4),:]

train_sizeV2_V3 = int(len(datasetV2_V3) * 0.791)
test_sizeV2_V3 = len(datasetV2_V3) - train_sizeV2_V3
trainV2_V3, testV2_V3 = datasetV2_V3[0:train_sizeV2_V3,:],
datasetV2_V3[train_sizeV2_V3:len(datasetV2_V3),:]

train_sizeV2_V4 = int(len(datasetV2_V4) * 0.791)
test_sizeV2_V4 = len(datasetV2_V4) - train_sizeV2_V4
trainV2_V4, testV2_V4 = datasetV2_V4[0:train_sizeV2_V4,:],
datasetV2_V4[train_sizeV2_V4:len(datasetV2_V4),:]

train_sizeV3_V4 = int(len(datasetV3_V4) * 0.791)
test_sizeV3_V4 = len(datasetV3_V4) - train_sizeV3_V4
trainV3_V4, testV3_V4 = datasetV3_V4[0:train_sizeV3_V4,:],
datasetV3_V4[train_sizeV3_V4:len(datasetV3_V4),:]

train_sizeVarBBC = int(len(datasetVarBBC) * 0.791)
test_sizeVarBBC = len(datasetVarBBC) - train_sizeVarBBC
trainVarBBC, testVarBBC = datasetVarBBC[0:train_sizeVarBBC,:],
datasetVarBBC[train_sizeVarBBC:len(datasetVarBBC),:]

train_sizeVarCorf = int(len(datasetVarCorf) * 0.791)
test_sizeVarCorf = len(datasetVarCorf) - train_sizeVarCorf
trainVarCorf, testVarCorf = datasetVarCorf[0:train_sizeVarCorf,:],
datasetVarCorf[train_sizeVarCorf:len(datasetVarCorf),:]

train_sizeVarGS = int(len(datasetVarGS) * 0.791)
test_sizeVarGS = len(datasetVarGS) - train_sizeVarGS
trainVarGS, testVarGS = datasetVarGS[0:train_sizeVarGS,:],
```

```
datasetVarGS[train_sizeVarGS:len(datasetVarGS),:]


train_sizeVarIsa = int(len(datasetVarIsa) * 0.791)
test_sizeVarIsa = len(datasetVarIsa) - train_sizeVarIsa
trainVarIsa, testVarIsa = datasetVarIsa[0:train_sizeVarIsa,:],
datasetVarIsa[train_sizeVarIsa:len(datasetVarIsa),:]
```

- Creación de dos conjuntos $x = t$ y $y = t + 1$ con el fin de guardar las covarianzas en un momento del tiempo t y otro conjunto con las covarianzas en el momento $t + 1$, $look\_back = 1$ es con el objetivo de especificar los pasos hacia atrás el cual fue de uno.

```
look_back = 1
trainXV1_V2, trainYV1_V2 = create_dataset(trainV1_V2, look_back)
testXV1_V2, testYV1_V2 = create_dataset(testV1_V2, look_back)

trainXV1_V3, trainYV1_V3 = create_dataset(trainV1_V3, look_back)
testXV1_V3, testYV1_V3 = create_dataset(testV1_V3, look_back)

trainXV1_V4, trainYV1_V4 = create_dataset(trainV1_V4, look_back)
testXV1_V4, testYV1_V4 = create_dataset(testV1_V4, look_back)

trainXV2_V3, trainYV2_V3 = create_dataset(trainV2_V3, look_back)
testXV2_V3, testYV2_V3 = create_dataset(testV2_V3, look_back)

trainXV2_V4, trainYV2_V4 = create_dataset(trainV2_V4, look_back)
testXV2_V4, testYV2_V4 = create_dataset(testV2_V4, look_back)

trainXV3_V4, trainYV3_V4 = create_dataset(trainV3_V4, look_back)
testXV3_V4, testYV3_V4 = create_dataset(testV3_V4, look_back)

trainXVarBBC, trainYVarBBC = create_dataset(trainVarBBC, look_back)
testXVarBBC, testYVarBBC = create_dataset(testVarBBC, look_back)

trainXVarCorf, trainYVarCorf = create_dataset(trainVarCorf, look_back)
testXVarCorf, testYVarCorf = create_dataset(testVarCorf, look_back)
```

```
trainXVarGS, trainYVarGS = create_dataset(trainVarGS, look_back)
testXVarGS, testYVarGS = create_dataset(testVarGS, look_back)

trainXVarIsa, trainYVarIsa = create_dataset(trainVarIsa, look_back)
testXVarIsa, testYVarIsa = create_dataset(testVarIsa, look_back)
```

- Remodelación de los datos de la forma: [muestras, pasos de tiempo, características].

```
trainXV1_V2 = np.reshape(trainXV1_V2, (trainXV1_V2.shape[0],
trainXV1_V2.shape[1], 1))
testXV1_V2 = np.reshape(testXV1_V2, (testXV1_V2.shape[0],
testXV1_V2.shape[1], 1))

trainXV1_V3 = np.reshape(trainXV1_V3, (trainXV1_V3.shape[0],
trainXV1_V3.shape[1], 1))
testXV1_V3 = np.reshape(testXV1_V3, (testXV1_V3.shape[0],
testXV1_V3.shape[1], 1))

trainXV1_V4 = np.reshape(trainXV1_V4, (trainXV1_V4.shape[0],
trainXV1_V4.shape[1], 1))
testXV1_V4 = np.reshape(testXV1_V4, (testXV1_V4.shape[0],
testXV1_V4.shape[1], 1))

trainXV2_V3 = np.reshape(trainXV2_V3, (trainXV2_V3.shape[0],
trainXV2_V3.shape[1], 1))
testXV2_V3 = np.reshape(testXV2_V3, (testXV2_V3.shape[0],
testXV2_V3.shape[1], 1))

trainXV2_V4 = np.reshape(trainXV2_V4, (trainXV2_V4.shape[0],
trainXV2_V4.shape[1], 1))
testXV2_V4 = np.reshape(testXV2_V4, (testXV2_V4.shape[0],
testXV2_V4.shape[1], 1))

trainXV3_V4 = np.reshape(trainXV3_V4, (trainXV3_V4.shape[0],
trainXV3_V4.shape[1], 1))
testXV3_V4 = np.reshape(testXV3_V4, (testXV3_V4.shape[0],
testXV3_V4.shape[1], 1))
```

```
trainXVarBBC = np.reshape(trainXVarBBC, (trainXVarBBC.shape[0],
trainXVarBBC.shape[1], 1))
testXVarBBC = np.reshape(testXVarBBC, (testXVarBBC.shape[0],
testXVarBBC.shape[1], 1))

trainXVarCorf = np.reshape(trainXVarCorf, (trainXVarCorf.shape[0],
trainXVarCorf.shape[1], 1))
testXVarCorf = np.reshape(testXVarCorf, (testXVarCorf.shape[0],
testXVarCorf.shape[1], 1))


trainXVarGS = np.reshape(trainXVarGS, (trainXVarGS.shape[0],
trainXVarGS.shape[1], 1))
testXVarGS = np.reshape(testXVarGS, (testXVarGS.shape[0],
testXVarGS.shape[1], 1))


trainXVarIsa = np.reshape(trainXVarIsa, (trainXVarIsa.shape[0],
trainXVarIsa.shape[1], 1))
testXVarIsa = np.reshape(testXVarIsa, (testXVarIsa.shape[0],
testXVarIsa.shape[1], 1))
```

- Creación de la LSTM.

```
batch_size = 1
model = Sequential()
model.add(LSTM(4, batch_input_shape=(batch_size, look_back, 1),
stateful=True, return_sequences=True))
model.add(LSTM(4, batch_input_shape=(batch_size, look_back, 1),
stateful=True))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
```

- Entrenamiento de la red.

```
for i in range(10):
model.fit(trainXV1_V2, trainYV1_V2, epochs=10,
batch_size=batch_size, verbose=2, shuffle=False)
model.reset_states()
```

```
for i in range(10):
model.fit(trainXV1_V3, trainYV1_V3, epochs=10,
batch_size=batch_size, verbose=2, shuffle=False)
model.reset_states()

for i in range(10):
model.fit(trainXV1_V4, trainYV1_V4,
epochs=10
batch_size=batch_size, verbose=2, shuffle=False)
model.reset_states()

for i in range(10):
model.fit(trainXV2_V3, trainYV2_V3, epochs=10,
batch_size=batch_size, verbose=2, shuffle=False)
model.reset_states()

for i in range(10):
model.fit(trainXV2_V4, trainYV2_V4, epochs=10,
batch_size=batch_size, verbose=2, shuffle=False)
model.reset_states()

for i in range(10):
model.fit(trainXV3_V4, trainYV3_V4, epochs=10,
batch_size=batch_size, verbose=2, shuffle=False)
model.reset_states()

for i in range(10):
model.fit(trainXVarBBC, trainYVarBBC, epochs=10,
batch_size=batch_size, verbose=2, shuffle=False)
model.reset_states()

for i in range(10):
model.fit(trainXVarCorf, trainYVarCorf, epochs=10,
batch_size=batch_size, verbose=2, shuffle=False)
model.reset_states()

for i in range(10):
model.fit(trainXVarGS, trainYVarGS, epochs=10,
```

```
batch_size=batch_size, verbose=2, shuffle=False)
model.reset_states()

for i in range(10):
model.fit(trainXVarIsa, trainYVarIsa, epochs=10,
batch_size=batch_size, verbose=2, shuffle=False)
model.reset_states()
```

- Predicciones.

```
trainPredictV1_V2 = model.predict(trainXV1_V2, batch_size=batch_size)
model.reset_states()
testPredictV1_V2 = model.predict(testXV1_V2, batch_size=batch_size)

trainPredictV1_V3 = model.predict(trainXV1_V3, batch_size=batch_size)
model.reset_states()
testPredictV1_V3 = model.predict(testXV1_V3, batch_size=batch_size)

trainPredictV1_V4 = model.predict(trainXV1_V4, batch_size=batch_size)
model.reset_states()
testPredictV1_V4 = model.predict(testXV1_V4, batch_size=batch_size)

trainPredictV2_V3 = model.predict(trainXV2_V3, batch_size=batch_size)
model.reset_states()
testPredictV2_V3 = model.predict(testXV2_V3, batch_size=batch_size)

trainPredictV2_V4 = model.predict(trainXV2_V4, batch_size=batch_size)
model.reset_states()
testPredictV2_V4 = model.predict(testXV2_V4, batch_size=batch_size)

trainPredictV3_V4 = model.predict(trainXV3_V4, batch_size=batch_size)
model.reset_states()
testPredictV3_V4 = model.predict(testXV3_V4, batch_size=batch_size)

trainPredictVarBBC = model.predict(trainXVarBBC, batch_size=batch_size)
model.reset_states()
```

```
testPredictVarBBC = model.predict(testXVarBBC, batch_size=batch_size)

trainPredictVarCorf = model.predict(trainXVarCorf, batch_size=batch_size)
model.reset_states()
testPredictVarCorf = model.predict(testXVarCorf, batch_size=batch_size)

trainPredictVarGS = model.predict(trainXVarGS, batch_size=batch_size)
model.reset_states()
testPredictVarGS = model.predict(testXVarGS, batch_size=batch_size)

trainPredictVarIsa = model.predict(trainXVarIsa, batch_size=batch_size)
model.reset_states()
testPredictVarIsa = model.predict(testXVarIsa, batch_size=batch_size)
```

- Se invierten las predicciones.

```
trainPredictV1_V2 = scalerV1_V2.inverse_transform(trainPredictV1_V2)
trainYV1_V2 = scalerV1_V2.inverse_transform([trainYV1_V2])
testPredictV1_V2 = scalerV1_V2.inverse_transform(testPredictV1_V2)
testYV1_V2 = scalerV1_V2.inverse_transform([testYV1_V2])
maV1_V2 = max(testPredictV1_V2)
miV1_V2 = min(testPredictV1_V2)
rangoV1_V2 = maV1_V2 - miV1_V2

trainPredictV1_V3 = scalerV1_V3.inverse_transform(trainPredictV1_V3)
trainYV1_V3 = scalerV1_V3.inverse_transform([trainYV1_V3])
testPredictV1_V3 = scalerV1_V3.inverse_transform(testPredictV1_V3)
testYV1_V3 = scalerV1_V3.inverse_transform([testYV1_V3])
maV1_V3 = max(testPredictV1_V3)
miV1_V3 = min(testPredictV1_V3)
rangoV1_V3 = maV1_V3 - miV1_V3

trainPredictV1_V4 = scalerV1_V4.inverse_transform(trainPredictV1_V4)
trainYV1_V4 = scalerV1_V4.inverse_transform([trainYV1_V4])
testPredictV1_V4 = scalerV1_V4.inverse_transform(testPredictV1_V4)
testYV1_V4 = scalerV1_V4.inverse_transform([testYV1_V4])
maV1_V4 = max(testPredictV1_V4)
miV1_V4 = min(testPredictV1_V4)
```

```
rangoV1_V4 = maV1_V4 - miV1_V4


trainPredictV2_V3 = scalerV2_V3.inverse_transform(trainPredictV2_V3)
trainYV2_V3 = scalerV2_V3.inverse_transform([trainYV2_V3])
testPredictV2_V3 = scalerV2_V3.inverse_transform(testPredictV2_V3)
testYV2_V3 = scalerV2_V3.inverse_transform([testYV2_V3])
maV2_V3 = max(testPredictV2_V3)
miV2_V3 = min(testPredictV2_V3)
rangoV2_V3 = maV2_V3 - miV2_V3


trainPredictV2_V4 = scalerV2_V4.inverse_transform(trainPredictV2_V4)
trainYV2_V4 = scalerV2_V4.inverse_transform([trainYV2_V4])
testPredictV2_V4 = scalerV2_V4.inverse_transform(testPredictV2_V4)
testYV2_V4 = scalerV2_V4.inverse_transform([testYV2_V4])
maV2_V4 = max(testPredictV2_V4)
miV2_V4 = min(testPredictV2_V4)
rangoV2_V4 = maV2_V4 - miV2_V4


trainPredictV3_V4 = scalerV3_V4.inverse_transform(trainPredictV3_V4)
trainYV3_V4 = scalerV3_V4.inverse_transform([trainYV3_V4])
testPredictV3_V4 = scalerV3_V4.inverse_transform(testPredictV3_V4)
testYV3_V4 = scalerV3_V4.inverse_transform([testYV3_V4])
maV3_V4 = max(testPredictV3_V4)
miV3_V4 = min(testPredictV3_V4)
rangoV3_V4 = maV3_V4 - miV3_V4


trainPredictVarBBC = scalerVarBBC.inverse_transform(trainPredictVarBBC)
trainYVarBBC = scalerVarBBC.inverse_transform([trainYVarBBC])
testPredictVarBBC = scalerVarBBC.inverse_transform(testPredictVarBBC)
testYVarBBC = scalerVarBBC.inverse_transform([testYVarBBC])
maBC = max(testPredictVarBBC)
miBC = min(testPredictVarBBC)
rangoBC = maBC - miBC



trainPredictVarCorf = scalerVarCorf.inverse_transform(trainPredictVarCorf)
trainYVarCorf = scalerVarCorf.inverse_transform([trainYVarCorf])
testPredictVarCorf = scalerVarCorf.inverse_transform(testPredictVarCorf)
testYVarCorf = scalerVarCorf.inverse_transform([testYVarCorf])
```

```
maCor = max(testPredictVarCorf)
miCor = min(testPredictVarCorf)
rangoCor = maCor - miCor

trainPredictVarGS = scalerVarGS.inverse_transform(trainPredictVarGS)
trainYVarGS = scalerVarGS.inverse_transform([trainYVarGS])
testPredictVarGS = scalerVarGS.inverse_transform(testPredictVarGS)
testYVarGS = scalerVarGS.inverse_transform([testYVarGS])
maGS = max(testPredictVarGS)
miGS = min(testPredictVarGS)
rangoGS = maGS - miGS

trainPredictVarIsa = scalerVarIsa.inverse_transform(trainPredictVarIsa)
trainYVarIsa = scalerVarIsa.inverse_transform([trainYVarIsa])
testPredictVarIsa = scalerVarIsa.inverse_transform(testPredictVarIsa)
testYVarIsa = scalerVarIsa.inverse_transform([testYVarIsa])
maIsa = max(testPredictVarIsa)
miIsa = min(testPredictVarIsa)
rangoIsa = maIsa - miIsa
```

- Validación de los pronósticos.

  - MAPE.

```
testScoreV1_V2 = math.sqrt(mean_square_error(testYV1_V2[0],
testPredictV1_V2[:,0]))
print('Resultado del test: %.5f RMSE' % (testScoreV1_V2))

testScoreV1_V3 = math.sqrt(mean_squared_error(testYV1_V3[0],
testPredictV1_V3[:,0]))
print('Resultado del test: %.5f RMSE' % (testScoreV1_V3))

testScoreV1_V4 = math.sqrt(mean_squared_error(testYV1_V4[0],
testPredictV1_V4[:,0]))
print('Resultado del test: %.5f RMSE' % (testScoreV1_V4))

testScoreV2_V3 = math.sqrt(mean_squared_error(testYV2_V3[0],
testPredictV2_V3[:,0]))
```

```
print('Resultado del test: %.5f RMSE' % (testScoreV2_V3))


testScoreV2_V4 = math.sqrt(mean_squared_error(testYV2_V4[0],
testPredictV2_V4[:,0]))
print('Resultado del test: %.5f RMSE' % (testScoreV2_V4))

testScoreV3_V4 = math.sqrt(mean_squared_error(testYV3_V4[0],
testPredictV3_V4[:,0]))
print('Resultado del test: %.5f RMSE' % (testScoreV3_V4))

testScoreVarBBC = math.sqrt(mean_squared_error(testYVarBBC[0],
testPredictVarBBC[:,0]))
print('Resultado del test: %.5f RMSE' % (testScoreVarBBC))

testScoreVarCorf = math.sqrt(mean_squared_error(testYVarCorf[0],
testPredictVarCorf[:,0]))
print('Resultado del test: %.5f RMSE' % (testScoreVarCorf))

testScoreVarGS = math.sqrt(mean_squared_error(testYVarGS[0],
testPredictVarGS[:,0]))
print('Resultado del test: %.5f RMSE' % (testScoreVarGS))

testScoreVarIsa = math.sqrt(mean_squared_error(testYVarIsa[0],
testPredictVarIsa[:,0]))
print('Resultado del test: %.5f RMSE' % (testScoreVarIsa))
```

- Intervalos de confianza (IC) para el RMSE.

```
st.norm.interval(alpha = 0.95, loc = testScoreV1 _V2, scale = st.sem
(testPredictV1 _V2
st.norm.interval(alpha = 0.95, loc = testScoreV1 _V3, scale = st.
(testPredictV1 _V3))
st.norm.interval(alpha = 0.95, loc = testScoreV1 _V4, scale = st.sem
(testPredictV1 _V4))
st.norm.interval(alpha = 0.95, loc = testScoreV2 _V3, scale = st.sem
(testPredictV2 _V3))
st.norm.interval(alpha = 0.95, loc = testScoreV2 _V4, scale = st.sem
(testPredictV2 _V4))
```

st.norm.interval(alpha = 0.95, loc = testScoreV3 _V4, scale = st.sem (testPredictV3 _V4))

st.norm.interval(alpha = 0.95, loc = testScoreVarBBC, scale = st.sem (testPredictVarBBC))
st.norm.interval(alpha = 0.95, loc = testScoreVarCorf, scale = st.sem (testPredictVarCorf))
st.norm.interval(alpha = 0.95, loc = testScoreVarGS, scale = st.sem (testPredictVarGS))
st.norm.interval(alpha = 0.95, loc = testScoreVarIsa, scale = st.sem (testPredictVarIsa))

- MAPE.

def mape (testYV1_V2 , testPredictV1_V2):
testYV1_V2, testPredictV1_V2 = np.array (testYV1_V2), np.array (testPredictV1_V2)
return np.mean (np.abs ((testYV1_V2 - testPredictV1_V2) / testYV1_V2))
mape_testYV1_V2 = mape(testYV1_V2, testPredictV1_V2)
mape_testYV1_V2

def mape (testYV1_V3 , testPredictV1_V3):
testYV1_V3, testPredictV1_V3 = np.array (testYV1_V3), np.array (testPredictV1_V3)
return np.mean (np.abs ((testYV1_V3 - testPredictV1_V3) / testYV1_V3))
mape_testYV1_V3 = mape(testYV1_V3, testPredictV1_V3)
mape_testYV1_V3

def mape (testYV1_V4 , testPredictV1_V4):
testYV1_V4, testPredictV1_V4 = np.array (testYV1_V4), np.array (testPredictV1_V4)
return np.mean (np.abs ((testYV1_V4 - testPredictV1_V4) / testYV1_V4))
mape_testYV1_V4 = mape(testYV1_V4, testPredictV1_V4)
mape_testYV1_V4

```
def mape (testYV2_V3 , testPredictV2_V3):
testYV2_V3, testPredictV2_V3 = np.array (testYV2_V3), np.array
(testPredictV2_V3)
return np.mean (np.abs ((testYV2_V3 - testPredictV2_V3) /
testYV2_V3))
mape_testYV2_V3 = mape(testYV2_V3, testPredictV2_V3)
mape_testYV2_V3


def mape (testYV2_V4 , testPredictV2_V4):
testYV2_V4, testPredictV2_V4 = np.array (testYV2_V4), np.array
(testPredictV2_V4)
return np.mean (np.abs ((testYV2_V4 - testPredictV2_V4) /
testYV2_V4))
mape_testYV2_V4 = mape(testYV2_V4, testPredictV2_V4)
mape_testYV2_V4


def mape (testYV3_V4 , testPredictV3_V4):
testYV3_V4, testPredictV3_V4 = np.array (testYV3_V4), np.array
(testPredictV3_V4)
return np.mean (np.abs ((testYV3_V4 - testPredictV3_V4) /
testYV3_V4)) mape_testYV3_V4 = mape(testYV3_V4, testPredictV3_V4)
mape_testYV3_V4


def mape (testYVarBBC , testPredictVarBBC):
testYVarBBC, testPredictVarBBC = np.array (testYVarBBC), np.array
(testPredictVarBBC)
return np.mean (np.abs ((testYVarBBC - testPredictVarBBC) /
testYVarBBC))
mape_testYVarBBC = mape(testYVarBBC, testPredictVarBBC)
map_testYVarBBC


def mape (testYVarCorf , testPredictVarCorf):
testYVarCorf, testPredictVarCorf = np.array (testYVarCorf),
np.array (testPredictVarCorf)
return np.mean (np.abs ((testYVarCorf - testPredictVarCorf) /
testYVarCorf))
mape_testYVarCorf = mape(testYVarCorf, testPredictVarCorf)
mape_testYVarCorf
```

```
def mape (testYVarGS , testPredictVarGS):
testYVarGS, testPredictVarGS = np.array (testYVarGS), np.array
(testPredictVarGS)
return np.mean (np.abs ((testYVarGS - testPredictVarGS) /
testYVarGS))
mape_testYVarGS = mape(testYVarGS, testPredictVarGS)
mape_testYVarGS
```

```
def mape (testYVarIsa , testPredictVarIsa):
testYVarIsa, testPredictVarIsa = np.array (testYVarIsa), np.array
(testPredictVarIsa)
return np.mean (np.abs ((testYVarIsa - testPredictVarIsa) /
testYVarIsa))
mape_testYVarIsa = mape(testYVarIsa, testPredictVarIsa)
mape_testYVarIsa
```

- Intervalos de confianza (IC) para el MAPE.

```
st.norm.interval(alpha = 0.95, loc = mape_testYV1_V2, scale = st.sem (test-
PredictV1_V2))
st.norm.interval(alpha = 0.95, loc = mape_testYV1_V3, scale = st.sem (test-
PredictV1_V3))
st.norm.interval(alpha = 0.95, loc = mape_testYV1_V4, scale = st.sem (test-
PredictV1_V4))
st.norm.interval(alpha = 0.95, loc = mape_testYV2_V3, scale = st.sem (test-
PredictV2_V3))
st.norm.interval(alpha = 0.95, loc = mape_testYV2_V4, scale = st.sem (test-
PredictV2_V4))
st.norm.interval(alpha = 0.95, loc = mape_testYV3_V4, scale = st.sem (test-
PredictV3_V4))
```

```
st.norm.interval(alpha = 0.95, loc = mape_testYVarBBC, scale = st.sem (test-
PredictVarBBC))
st.norm.interval(alpha = 0.95, loc = mape_testYVarCorf, scale = st.sem (test-
PredictVarCorf))
st.norm.interval(alpha = 0.95, loc = mape_testYVarGS, scale = st.sem (testPre-
dictVarGS))
st.norm.interval(alpha = 0.95, loc = mape_testYVarIsa, scale = st.sem (testPre-
dictVarIsa))
```

- MAD.

```
robust.mad(trainPredictV1_V2)
mad_V1_V2 = robust.mad(testPredictV1_V2)
robust.mad(trainPredictV1_V3)
mad_V1_V3 = robust.mad(testPredictV1_V3)
robust.mad(trainPredictV1_V4)
mad_V1_V4 = robust.mad(testPredictV1_V4)
robust.mad(trainPredictV2_V3)
mad_V2_V3 = robust.mad(testPredictV2_V3)
robust.mad(trainPredictV2_V4)
mad_V2_V4 = robust.mad(testPredictV2_V4)
robust.mad(trainPredictV3_V4)
mad_V3_V4 = robust.mad(testPredictV3_V4)

robust.mad(trainPredictVarBBC)
madVarBBC = robust.mad(testPredictVarBBC)
robust.mad(trainPredictVarCorf)
madVarCorfi = robust.mad(testPredictVarCorf)
robust.mad(trainPredictVarGS)
madVarGS = robust.mad(trainPredictVarGS)
robust.mad(trainPredictVarIsa)
madVarIsa = robust.mad(trainPredictVarIsa)
```

- Intervalos de confianza (IC) para el MAD.

```
st.norm.interval(alpha = 0.95, loc = mad_V1_V2, scale = st.sem (testPredictV1_V2))
st.norm.interval(alpha = 0.95, loc = mad_V1_V3, scale = st.sem (testPredictV1_V3))
st.norm.interval(alpha = 0.95, loc = mad_V1_V4, scale = st.sem (testPredictV1_V4))
st.norm.interval(alpha = 0.95, loc = mad_V2_V3, scale = st.sem (testPredictV2_V3))
st.norm.interval(alpha = 0.95, loc = mad_V2_V4, scale = st.sem (testPredictV2_V4))
st.norm.interval(alpha = 0.95, loc = mad_V3_V4, scale = st.sem (testPredictV3_V4))

st.norm.interval(alpha = 0.95, loc = madVarBBC, scale = st.sem (testPredict-
VarBBC))
st.norm.interval(alpha = 0.95, loc = madVarCorfi, scale = st.sem (testPredict-
VarCorf))
```

st.norm.interval(alpha = 0.95, loc = madVarGS, scale = st.sem (trainPredictVarGS))
st.norm.interval(alpha = 0.95, loc = madVarIsa, scale = st.sem (trainPredictVarIsa))

- Gráficas de las varianzas covarianzas.

trainPredictV1_V2Plot = np.empty_like(datasetV1_V2)
trainPredictV1_V2Plot[:, :] = np.nan
trainPredictV1_V2Plot[look_back:len(trainPredictV1_V2)+look_back, :] = trainPredictV1_V2
testPredictV1_V2Plot = np.empty_like(datasetV1_V2)
testPredictV1_V2Plot[:, :] = np.nan
testPredictV1_V2Plot[len(trainPredictV1_V2)+(look_back*2)+1:len(datasetV1_V2)-1, :] = testPredictV1_V2
plt.plot(scalerV1_V2.inverse_transform(datasetV1_V2))
plt.plot(trainPredictV1_V2Plot,'r', linewidth = 2)
plt.plot(testPredictV1_V2Plot,'m', linewidth = 2)
plt.legend(('Datos', 'Predicción datos entrenamiento', 'Predicción datos test'), loc = 'upper left')
plt.grid(True)
plt.title("BColombia/CorfiCol", fontsize = 15)
plt.ylabel(Çovarianza", fontsize = 10)
plt.show()

trainPredictV1_V3Plot = np.empty_like(datasetV1_V3)
trainPredictV1_V3Plot[:, :] = np.nan
trainPredictV1_V3Plot[look_back:len(trainPredictV1_V3)+look_back, :] = trainPredictV1_V3
testPredictV1_V3Plot = np.empty_like(datasetV1_V3)
testPredictV1_V3Plot[:, :] = np.nan
testPredictV1_V3Plot[len(trainPredictV1_V3)+(look_back*2)+1:len(datasetV1_V3)-1, :] = testPredictV1_V3
plt.plot(scalerV1_V3.inverse_transform(datasetV1_V3))
plt.plot(trainPredictV1_V3Plot,'r', linewidth = 2)
plt.plot(testPredictV1_V3Plot,'m', linewidth = 2)
plt.legend(('Datos', 'Predicción datos entrenamiento', 'Predicción datos test'), loc = 'upper left')
plt.grid(True)

```
plt.title("BColombia/Grupo Sura", fontsize = 15)
plt.ylabel(Çovarianza", fontsize = 10)
plt.show()


trainPredictV1_V4Plot = np.empty_like(datasetV1_V4)
trainPredictV1_V4Plot[:, :] = np.nan
trainPredictV1_V4Plot[look_back:len(trainPredictV1_V4)+loo_back, :] = train-
PredictV1_V4
testPredictV1_V4Plot = np.empty_like(dataset_V4)
testPredictV1_V4Plot[:, :] = np.nan
testPredictV1_V4Plot[len(trainPredictV1_V4)+(look_back*2)+1:len(datasetV1_V4)-
1, :] = testPredictV1_V4 plt.plot(scalerV1_V4.inverse_transform(datasetV1_V4))
plt.plot(trainPredictV1_V4Plot,'r', linewidth = 2)
plt.plot(testPredictV1_V4Plot,'m', linewidth = 2)
plt.legend( ('Datos', 'Predicción datos entrenamiento', 'Predicción datos test'),
loc = 'upper left')
plt.grid(True)
plt.title("BColombia/Isa", fontsize = 15)
plt.ylabel(Çovarianza", fontsize = 10)
plt.show()


trainPredictV2_V3Plot = np.empty_like(datasetV2_V3)
trainPredictV2_V3Plot[:, :] = np.nan
trainPredictV2_V3Plot[look_back:len(trainPredictV2_V3)+look_back, :] = train-
PredictV2_V3
testPredictV2_V3Plot = np.empty_like(datasetV2_V3)
testPredictV2_V3Plot[:, :] = np.nan
testPredictV2_V3Plot[len(trainPredictV2_V3)+(look_back*2)+1:len(datasetV2_V3)-
1, :] = testPredictV2_V3
plt.plot(scalerV2_V3.inverse_transform(datasetV2_V3))
plt.plot(trainPredictV2_V3Plot,'r', linewidth = 2)
plt.plot(testPredictV2_V3Plot,'m', linewidth = 2)
plt.legend( ('Datos', 'Predicción datos entrenamiento', 'Predicción datos test'),
loc = 'upper left')
plt.grid(True)
plt.title(Çorficol/Grupo Sura", fontsize = 15)
plt.ylabel(Çovarianza", fontsize = 10)
plt.show()
```

```
trainPredictV2_V4Plot = np.empty_like(datasetV2_V4)
trainPredictV2_V4Plot[:, :] = np.nan
trainPredictV2_V4Plot[look_back:len(trainPredictV2_V4)+look_back, :] = train-
PredictV2_V4
testPredictV2_V4Plot = np.empty_like(datasetV2_V4)
testPredictV2_V4Plot[:, :] = np.nan
testPredictV2_V4Plot[len(trainPredictV2_V4)+(look_back*2)+1:len(datasetV2_V4)-
1, :] = testPredictV2_V4
plt.plot(scalerV2_V4.inverse_transform(datasetV2_V4))
plt.plot(trainPredictV2_V4Plot,'r', linewidth = 2)
plt.plot(testPredictV2_V4Plot,'m', linewidth = 2)
plt.legend( ('Datos', 'Predicción datos entrenamiento', 'Predicción datos test'),
loc = 'upper left')
plt.grid(True)
plt.title(Çorficol/Isa", fontsize = 15)
plt.ylabel(Çovarianza", fontsize = 10)
plt.show()


trainPredictV3_V4Plot = np.empty_like(datasetV3_V4)
trainPredictV3_V4Plot[:, :] = np.nan
trainPredictV3_V4Plot[look_back:len(trainPredictV3_V4)+look_back, :] = train-
PredictV3_V4
testPredictV3_V4Plot = np.empty_like(datasetV3_V4)
testPredictV3_V4Plot[:, :] = np.nan
testPredictV3_V4Plot[len(trainPredictV3_V4)+(look_back*2)+1:len(datasetV3_V4)-
1, :] = testPredictV3_V4
plt.plot(scalerV3_V4.inverse_transform(datasetV3_V4))
plt.plot(trainPredictV3_V4Plot,'r', linewidth = 2)
plt.plot(testPredictV3_V4Plot,'m', linewidth = 2)
plt.legend( ('Datos', 'Predicción datos entrenamiento', 'Predicción datos test'),
loc = 'upper left')
plt.grid(True)
plt.title("Grupo Sura/Isa", fontsize = 15)
plt.ylabel(Çovarianza", fontsize = 10)
plt.show()
```

- Gráficas de los errores.

```
from matplotlib import pyplot as plt
```

```
x = np.linspace(0, 100, 100)
fig = plt.figure()
fig.clf()
ax = fig.subplots(2,3)


ErroresV1V2 = testV1_V2 - testPredictV1_V2
yV1V2 = ErroresV1V2
ax[0,0].plot(x, yV1V2)
ax[0,0].grid(True)
ax[0,0].set_title('BColombia/Corficol')


ErroresV1V3 = testV1_V3 - testPredictV1_V3
yV1V3 = ErroresV1V3
ax[0,1].plot(x, yV1V3)
ax[0,1].grid(True)
ax[0,1].set_title('BColombia/Grupo Sura)')


ErroresV1V4 = testV1_V4 - testPredictV1_V4
yV1V4 = ErroresV1V4
ax[0,2].plot(x, yV1V4)
ax[0,2].grid(True)
ax[0,2].set_title('BColombia/Isa')


ErroresV2V3 = testV2_V3 - testPredictV2_V3
yV2V3 = ErroresV2V3
ax[1,0].plot(x, yV2V3)
ax[1,0].grid(True)
ax[1,0].set_title('Corficol/Grupo Sura')


ErroresV2V4 = testV2_V4 - testPredictV2_V4
yV2V4 = ErroresV2V4
ax[1,1].plot(x, yV2V4)
ax[1,1].grid(True)
ax[1,1].set_title('Corficol/Isa')


ErroresV3V4 = testV3_V4 - testPredictV3_V4
yV3V4 = ErroresV3V4
ax[1,2].plot(x, yV3V4)
```

```
ax[1,2].grid(True)
ax[1,2].set_title('Grupo Sura/Isa')

fig.tight_layout()
fig.show()
```

Bibliografía

[1] Alberto Ruiz, Carlos Basualdo, M. S. . M. D. J. *Redes neuronales: conceptos Básicos y aplicaciones.* 2001.

[2] Arana, C. Redes neuronales recurrentes:análisis de los modelos especializados en datos secuenciales. *Econstor Documento de trabajo Nro 797* (2021), 1–25.

[3] Bancolombia. https://www.bancolombia.com/, 2023.

[4] Bernaldo de Quirós, M. R. Análisis de líneas de costa con redes neuronales LSTM. Master's thesis, Universitat Obera de Catalunya, 2020.

[5] Bollerslev, T. Generalized autoregressive conditional heteroscedasticity. *Econometrics* (1986), 27–307.

[6] Campbell, J., and Viceira, L. Consumption and portfolio decisions when expected returns are time varying. *Q J Econ 114(2)* (1999), 433–95.

[7] Carrillo Ríoz, J. L. Análisis de volatilidad de precios de las acciones del banco del Pichincha utilizando el modelo arch. Master's thesis, Universidad Técnica de Ambato, 2017.

[8] Charu C., A. *Neural Networks and Deep Learning.* Springer, Yorktown Heights, NY, USA, 2018.

[9] Cheong, C. W. Modeling and forecasting crude oil markets using arch-type models. *Energy Policy 37* (2009), 2346–2355.

[10] Corficol. https://www.corficolombiana.com/, 2023.

[11] Cruz Zúñiga, M., and Ramírez Tapia, D. L. El efecto de la incertidumbre real, la inlfación y el crecimiento económico en Mexíco y Brasil: evidencia empírica con modelos garch bivariados con correlación condicional constante (1985-2019). Master's thesis, Universidad Autónoma del Estado de México, 2021.

[12] de Valores de Colombia, B. https://www.bvc.com.co/, Junio 2020.

[13] Dhaene, G., and Wu, J. Incorporating overnight and intraday returns into multivariate garch volatility models. *Journal of Econometrics 217* (2020), 471–495.

[14] Engle, R. F. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrics 20* (1982), 987–1008.

[15] Escobar-Anel, M., Gollart, M., and Zagst, R. Closed-form portfolio optimization under garch models. *Operations Research Perspectives 9* (2022), 100–216.

[16] Escobar-Anel, M., Rastegari, J., and Stentoft, L. Affine multivariate garch models. *Journal of Banking and Finance 118* (2020).

[17] Galarza Hernández, J. Reducción de dimensionalidad en machine learning. Master's thesis, Universidad Politécnica de Valencia, 2017.

[18] Giraldo Picón, E. L. Pronóstico de volatilidades a los rendimientos de activos financieros de renta variable en Colombia a través de modelos arch y garch. Master's thesis, Universidad Nacional de Colombia, 2022.

[19] Guamán Pachacama, J. A., and Noz Génesis Bridggitte, S. M. Red neuronal long short-term memory (lstm) aplicada a series temporales para pronosticar consumo energético en edificaciones. Master's thesis, Universidad de Guayaquil, 2020.

[20] Guo, Z.-Y. Risk management of bitcoin futures with garch models. *Finance Research Letters 45* (2022), 102–197.

[21] Hilera, J. R., and Matínez, V. J. *Redes neuronales artificiales. Fundamentos, modelos y aplicaciones.* RA-MA, Madrid, 1995.

[22] Hull, J. Options, futures, and other derivatives. *Prentice Hall* (2012).

[23] Isa. https://www.isa.co/es/, 2022.

[24] James, G., Witten, D., Hastie, T., and Tibshirani, R. *An Introduction to Statistical Learning With application in R, 2021.*

[25] James, H., and Marck, M. W. *Introducción a la econometría.* PEARSON, Madrid, 2012.

[26] Kim, H. Y., and Won, C. H. Forecasting the volatility of stock price index: A hybrid model integrating lstm with multiple garch-type models. *Elsevier 103* (2018), 25–37.

[27] López Villa, Jorge Sosa Castro, M. Volatilidad condicional y correlación dinámica entre los mercados cambiarios y de valores en México (2009-2019): una aproximación garch-dcc. *Azcapotzalco 10* (2020), 195–220.

[28] Márquez Cebrián, C. M. *Modelo SETAR aplicado a la volatilidad de la rentabilidad de las acciones: algoritmos para su identificación.* Tesis de doctorado, Universidad Politécnica de Cataluya, 2002.

[29] McNeil, A. F., Frey, R., and Embrechts, P. *Quantitative Risk Management.* PSF, New Jersey, 2015.

[30] Mei-Li, S., Cheng-Feng, L., Hsiou-Hsiang, L., Po-Yin, C., and Cheng-Hong, Y. Effective multinational trade forecasting using lstm recurrent neural network. *Expert Systems With Applications 182* (2021).

[31] Melo Velandia, Luis Fernando Becerra Camargo, O. R. Una aproximación a la dinámica de las tasas de interés de corto plazo en colombia a través de modelos garch multivariados. *Banco de la República* (2006).

[32] Morales Castro, J. A. Factores de influyen en las acciones sustentables de la bolsa mexicana de valores. *Escritos Contables y de Administración 7* (2016), 15–47.

[33] Muñoz, José Julián Torres, D. E. Construcción de un portafolio de inversión de renta variable y tes mediante modelos de volatilidad para un perfil de riesgo determinado. Master's thesis, Escuela de Economía y Finanzas, 2014.

[34] Nor Syahilla, A. A., Vrontos, S., and Haslifah, M. H. Evaluation of multivariate garch models in an optimal asset allocation framework. *North American Journal of Economics and Finance 47* (2019), 568–596.

[35] Quintero Valencia, D. E. Pronóstico de volatilidad de la trm mediante un modelo híbrido lstm-garch. Master's thesis, Universidad del Rosario, 2019.

[36] Regueiro, C. V., Barro, S., Sánchez, E., and Fernández-Delgado, M. Modelos básicos de redes neronales artificiales. *Universidad de Santiago de Compostela VII* (1995), 181–218.

[37] Rojas, A. E., and Palacios, Y. A. Modelos arch: una aplicación en el pronóstico de la volatilidad de acciones cotizadas en la bolsa de valores de Lima. *PESQUIMAT VII, Nro 1* (2004), 64–79.

[38] Ruiz Espejo, M. Estimación de la desviación estándar. *Estadística Española 59* (2017), 37–44.

[39] Sánchez V., A., and Reyes M., O. Regularidades probabilísticas de las series financieras y la familia de modelos garch. *Ciencia Ergo Sum 13* (2006), 149–156.

[40] Sura, G. https://www.gruposura.com/, 2023.

[41] Team, R. C. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 1993.

[42] Van Rossum, G., and Drake Jr, F. L. *Python 3 Reference Manual.* Python Software Foundation, Scotts Valley, CA, 2009.

[43] Yao, Y., Zhao, Y., and Li, Y. A volatility model based on adaptive expectation: An improvement on the rational expectation model. *International Review of Financial Analysis 82* (2022).