APLICACIÓN DE MODELOS DE DEEP LEARNING Y MACHINE LEARNING PARA EL PRONÓSTICO DE LA SERIE DE TIEMPO DEL PAR DE DIVISAS EURO - DÓLAR

CAMILO ANDRÉS CORREDOR ROA
CAMILO MONTOYA ZAPATA

Monografía para optar al título de Especialista en Analítica y Ciencia de Datos

JOHN JAIR QUIZA MONTEALEGRE
Magister en Ingeniería - Telecomunicaciones

Universidad de Antioquia
Facultad de Ingeniería
Especialización en Analítica y Ciencia de Datos
Medellín, Antioquia
2022

Cita		Corredor Roa y Montoya Zapata [1]
Referencia **Estilo IEEE (2020)**	[1]	C. Corredor Roa y C. Montoya Zapata, “APLICACIÓN DE MODELOS DE DEEP LEARNING Y MACHINE LEARNING PARA EL PRONÓSTICO DE LA SERIE DE TIEMPO DEL PAR DE DIVISAS EURO - DÓLAR, Trabajo de grado de especialización, Especialización en Analítica y Ciencia de Datos, Universidad de Antioquia, Medellín, Antioquia, Colombia, 2022

Especialización en Analítica y Ciencia de Datos, Cohorte III

Centro de Documentación CENDOI

Universidad de Antioquia - www.udea.edu.co
Rector: Jhon Jairo Arboleda Céspedes.
Decano/Director: Jesús Francisco Vargas Bonilla.
Jefe departamento: Diego José Luis Botia Valderrama.

1. RESUMEN EJECUTIVO 5
2. DESCRIPCIÓN DEL PROBLEMA 7
2.1 PROBLEMA DE NEGOCIO 7
2.2 APROXIMACIÓN DESDE LA ANALÍTICA DE DATOS 7
2.3 ORIGEN DE LOS DATOS 8
2.4 MÉTRICAS DE DESEMPEÑO 9
3. DATOS 10
3.1 DATOS ORIGINALES 10
3.2 DATASETS 11
3.3 DESCRIPTIVA 12
4. PROCESO DE ANALÍTICA 15
4.1 PIPELINE PRINCIPAL 15
4.2 PREPROCESAMIENTO 16
4.3 MODELOS 17
4.3.1 ARIMA 17
4.3.2 XGBOOST 17
4.3.3 REDES NEURONALES RECURRENTE - RNN 20
4.3.3.1 LONG SHORT TERM MEMORY - LSTM 25
4.3.3.2 GATED RECURRENT UNIT - GRU 27
4.5 MÉTRICAS 28
5. METODOLOGÍA 29
5.1 BASELINE 29
5.2 VALIDACIÓN 30
5.3 ITERACIONES y EVOLUCIÓN 31
5.3.1 XGBOOST 31
5.3.2 REDES NEURONALES RECURRENTES 32
5.3.3 LONG SHORT TERM MEMORY - LSTM 35
5.3.4 GATED RECURRENT UNIT - GRU 36
5.4 HERRAMIENTAS 38
6. RESULTADOS 40
6.1 MÉTRICAS 40
6.1.1 MODELOS ARIMA 40
6.1.2 REDES NEURONALES RECURRENTES - RNN 40
6.1.3 LONG SHORT TERM MEMORY - LSTM 42
6.1.4 GATED RECURRENT UNIT - GRU 43

6.1.2 PREDICCIONES DE VALORES FUTUROS 45
6.2 EVALUACIÓN CUALITATIVA 46
6.3 CONSIDERACIONES DE PRODUCCIÓN 47
7. CONCLUSIONES 48
8. REFERENCIAS 49

1. RESUMEN EJECUTIVO

Los países con grandes economías representan su poder en la fuerza de las monedas que emiten y que son usadas para el intercambio de bienes y servicios en el contexto nacional e internacional. Es así como el dólar estadounidense se ubica en el primer lugar como la moneda con mayor demanda a nivel mundial para las transacciones internacionales y la tasa de cambio representativa en general para las economías emergentes de América latina. Ahora bien, a pesar de la dominancia en el mercado por parte del dólar estadounidense, existen en el mundo monedas con alto poder adquisitivo como el Euro, el Yuan chino, la libra esterlina, que también son usadas en el mercado de intercambio de bienes y servicios.

La confluencia de diversas monedas en el mercado da lugar a la aparición de un nuevo actor, el mercado internacional de divisas Forex, cuya función general es la estimación comparativa de la dominancia de una moneda frente a otra. El mercado se desarrolla bajo una idea fundamental, realizar operaciones de intercambio de monedas que fluctúan en el tiempo para obtener ganancias, a esto se denomina *trading*. En este punto es necesario resaltar que, este mercado de divisas movía para 2019 diariamente cerca de seis trillones de dólares, según datos de [1] y su operación se extiende sobre todo el planeta.

La posibilidad de realizar operaciones de intercambio de monedas se ha expandido ampliamente, y hoy, cualquier persona con acceso a internet puede realizar estas operaciones de trading basados en análisis gráfico o fundamental y múltiples índices, con lo cual buscan generar ganancias a partir de la puesta en riesgo de determinado capital.

En relación con los aspectos técnicos económicos, la dominancia de una moneda frente a sus pares, se define por decisiones y características macroeconómicas inherentes de la economía, que finalmente quedan sintetizadas en las tasas de intercambio de las monedas analizadas. Por tal motivo, es un asunto de especial interés para los gobiernos, compañías de operación nacional y/o internacional, grandes y pequeños inversores, conocer de antemano el comportamiento de las monedas de intercambio.

Aunque, la cotización del euro frente al dólar es un tema transversal a la economía mundial y su comprensión involucra múltiples aspectos macroeconómicos como los cambios en la política económica, desastres naturales e incluso terrorismo, este trabajo está enfocado únicamente en el desarrollo de modelos basados en aprendizaje profundo y machine learning para el pronóstico de la tasa cambiaria de dos de las monedas con mayor dominancia en el mercado.

Bajo este contexto, en la web dispone de información de la tasa cambiaría que puede ser obtenida a través de *scraping* o por medio de la descarga de información de sitios web como *Investing.com*. En tal sentido, se ha definido un marco temporal para el análisis comprendido entre el 03 de enero de 2005 y hasta el 31 de enero de 2022, conjunto de datos que cuenta con 4457 observaciones.

El flujo de trabajo para la implementación de modelos deep learning y machine learning está comprendido así: a partir de las 4457 observaciones se desarrolla un preprocesamiento de la información, que significa el escalamiento de los datos, luego se realiza una segmentación de los datos en subconjuntos de entrenamiento, testeo en el caso de la aplicación del modelo *ARIMA* y *XGBOOST*. En el caso de las redes neuronales recurrentes LSTM, *Long Short Time Memory*, se genera un subconjunto de datos adicional que opera como validación en el proceso de entrenamiento. Finalmente, a partir de las métricas consideradas a lo largo de este documento se evalúa la precisión de los

pronósticos generados en cada uno de los modelos para determinar de esta forma, el más apropiado para la serie de tiempo analizada.

Una última etapa considera la predicción de los modelos en un periodo comprendido entre el primero de febrero de 2022 hasta el 28 de febrero del mismo año y comparado con los datos reales de la cotización euro - dólar, al igual que en el caso anterior, se evalúa la calidad de la predicción a partir de valor de las métricas consideradas.

Se evidencia a lo largo de este documento el desempeño de cada uno de los modelos analizados por medio de métricas en comparación con los datos reales de la cotización y gráficas que evidencian la distancia entre el pronóstico obtenido y la serie de tiempo.

Particularmente, los modelos implementados y presentados a lo largo de este documento tienen métricas de error que no exceden en ningún caso el 2%; sin embargo, estos pronósticos no recogen la volatilidad del comportamiento de la serie de tiempo y su aplicabilidad en casos reales no tiene una incidencia positiva. En contraste, las redes neuronales recurrentes LSTM aunque tienen una métrica de error superior al valor planteado, la visualización de este pronóstico evidencia particularidades que serán comentadas en la penúltima sección de este documento y que merecen por supuesto ampliar su análisis desde el enfoque de la construcción de la arquitectura de la red neuronal en futuros trabajos.

2. DESCRIPCIÓN DEL PROBLEMA

Al considerar la dimensión económica del mercado FOREX, según [1], su alta volatilidad debido a factores como políticas macroeconómicas, y la permeabilidad de este mercado a acontecimientos sociales, entre otros, pone de manifiesto la necesidad de uso de algoritmos que permitan pronosticar el valor de la tasa de cambio o simplemente la tendencia de esta, con el objetivo fundamental de generar riqueza aprovechando las fluctuaciones del mercado.

Las cotizaciones del mercado FOREX terminan siendo de vital importancia no solo para grandes o pequeños inversionistas de capital privado; también los estados nacionales realizan operaciones de intercambio con metas como el sostenimiento en los primeros lugares de sus economías y la dominancia en el mercado de sus monedas. Uno de los ejemplos con mayor divulgación para el caso de Colombia son los fondos privados de pensión que, con ocasión del Decreto 2955 de 2010, le otorgan facultades para la implementación de esta práctica para el crecimiento económico no solo del fondo de pensiones sino también del monto de la cotización pensional de cada individuo. Cabe resaltar que, las inversiones de los fondos de pensiones incluyen el mercado accionario que no dista en gran medida del mercado FOREX pero sale de los límites propuestos en este documento. Evidentemente, la volatilidad del mercado pone de manifiesto los riesgos a los que se enfrenta un usuario al realizar operaciones de *trading.*

2.1 PROBLEMA DE NEGOCIO

La operación en el mercado por parte de los inversionistas está supeditada evidentemente a la volatilidad del mercado, y este factor logra poner en aprietos a esta amplia comunidad alrededor del mundo. Por ello, múltiples herramientas están disponibles en las plataformas digitales de *trading* para contrarrestar el efecto de la volatilidad frente a las posiciones que toman los inversionistas en el mercado. No obstante, aunque en una operación de trading el usuario haga uso de múltiples herramientas técnicas para proteger su inversión y asegurar el éxito de su operación, la volatilidad del mercado siempre será un factor a considerar y el riesgo siempre estará latente, por ello, lo que se busca, es disminuir la incertidumbre del próximo movimiento de la tasa cambiaria, es decir, que el problema de negocio al que se enfrentan los inversionistas es fundamentalmente el siguiente: estando en el tiempo ***t,*** debe obtener la predicción del valor de la tasa cambiaria en los tiempo ***t+1, t+2, t+3 … t+n,*** y con ello lograr las mejores condiciones del mercado para la implementación de su estrategia de trading y de esta forma maximizar sus ganancias.

2.2 APROXIMACIÓN DESDE LA ANALÍTICA DE DATOS

La conceptualización matemática de las herramientas y algoritmos de *machine learning* y *deep learning* usadas para el análisis de datos, en muchos casos, sus diseños fueron elaborados en el siglo pasado y solo hasta hoy, gracias al boom tecnológico que impacta nuestra sociedad, se tiene la oportunidad de hacer uso de ellas. Puntualmente, en el caso de las redes neuronales, su génesis tuvo lugar a mediados del siglo X y sus primeras implementaciones se desarrollaron para la consideración de funciones linealmente separables.

Esta aproximación somera al perceptrón otorga hoy la posibilidad de construir intrincadas arquitecturas neuronales y la facultad de resolver problemas más complejos como la predicción del valor de la tasa de intercambio para dos activos monetarios.

Desde la analítica de datos, se busca la construcción de modelos predictivos basados en aprendizaje de máquina y aprendizaje profundo, específicamente con redes neuronales recurrentes, que predigan el comportamiento futuro del mercado. Su validación, aunque proviene de datos reales de la tasa cambiaría del par de divisas euro y dólar, no puede estar supeditada únicamente al periodo que comprende el subconjunto de datos de entrenamiento y testeo, por el contrario, se debe ampliar sobre un conjunto de datos posterior a la serie de tiempo considerada, para realizar el contraste entre el pronóstico y los datos reales en el periodo mencionado anteriormente y verificar su precisión.

En consecuencia, los algoritmos que se discuten a continuación están encaminados a dotar de herramientas técnicas al inversionista para la operación que pretende desarrollar y que, a partir de ello, logre obtener y maximizar beneficios económicos. No obstante, el marco de referencia para el comparativo de las predicciones será detallado en secciones posteriores.

Seguramente, someter los algoritmos propuestos en este documento para realizar operaciones en el mercado FOREX, mostrarán resultados que exigen de la implementación de arquitecturas neuronales artificiales más complejas que requieren un mayor recurso computacional; adicionalmente, la alta volatilidad inherente a tipo de mercados pone un límite natural a la precisión de la predicción.

2.3 ORIGEN DE LOS DATOS

Los datos utilizados en este proyecto representan el cierre de cotización diaria del par de divisas euro y dólar, estos son públicos y para obtenerlos, el usuario debe registrarse en el sitio web *Investing.com*, posteriormente debe dirigirse al siguiente enlace Histórica cotización diaria EUR/USD. Ya en el sitio web, el usuario determina la fecha de inicio y final para el respectivo análisis y con ello el aplicativo permite una descarga de un fichero en formato csv que contiene seis columnas: fecha, último, apertura, máximo, mínimo y variación porcentual.

La columna “último”, corresponde con el cierre de la cotización del día anterior, de acuerdo con una revisión preliminar de la base de datos, esta puede ser igual al valor de la columna “apertura” para del día siguiente, sin embargo, esto no se cumple en todos los casos. Las columnas “máximo” y “mínimo” corresponden con la cotización del par euro - dólar diariamente y por último la variación porcentual está relacionada con el cambio presente entre la cotización del periodo ***t*** frente al periodo ***t-1,*** que finalmente evidencia la presencia de alta volatilidad mencionada anteriormente.

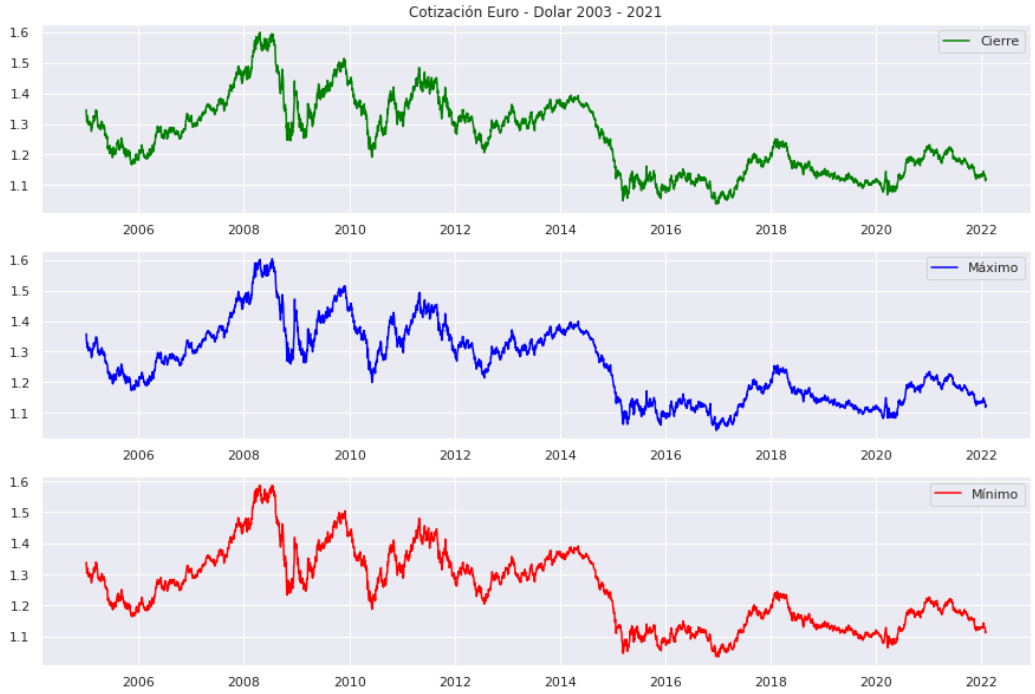

Figure 1: Vista general de los componentes de la base de datos. Fuente: Elaboración propia

Sobre el marco de referencia planteado para el análisis, se descargaron las cotizaciones diarias comprendidas del par euro - dólar entre el 3 de enero del 2005 hasta el 31 de enero del 2022. Respecto a la base de datos mencionada, la columna elegida para ingresar a los algoritmos que se proponen a continuación es: "último". Una visión general del proceso estocástico con base en la Fig. 1, evidencia una alta correlación entre las 3 columnas restantes, por lo tanto, ejecutar los algoritmos sobre las demás columnas tendrá resultados similares, exceptuando la columna de variación porcentual.

2.4 MÉTRICAS DE DESEMPEÑO

El monitoreo de los resultados de los modelos propuestos se realiza a través del error porcentual absoluto medio - MAPE, dado por la eq. (1), que representa la diferencia obtenida entre el valor predicho frente al valor presente en la serie de tiempo en términos porcentuales. Posteriormente al contrastar frente al valor real, se tendrá una visión amplia y suficiente del estado del pronóstico, adicionalmente, es de resaltar que, el análisis de las predicciones frente a esta medida lo hace intuitivo y de fácil interpretación para problemas de series de tiempo. MAPE está dado por la siguiente expresión:

$$MAPE = \frac{1}{m} \sum \left| \frac{x_i - \hat{f}_i}{X_i} \right| \% \tag{1}$$

De otra parte, se propone el error medio cuadrático para evaluar el modelo en términos de la tasa de cambio entre el par de divisas euro y dólar. Aunque no resulta tan intuitiva como MAPE, de acuerdo con la eq. (2), el MSE otorga una visión generalizada de la respuesta del modelo frente a la serie de tiempo de las cotizaciones diarias del par de divisas y está dada por la siguiente ecuación: Como se en la eq. (2)

$$MSE = \sum \frac{(X_i - \hat{f}_i)^2}{m} \quad (2)$$

Para las dos métricas de evaluación, las variables son:

Xi: Es el valor real para el tiempo ***t*** en la serie de tiempo
$\hat{f}\imath$: Es el valor pronosticado en el tiempo t por el modelo propuesto
m: Número de observaciones evaluadas en el periodo de tiempo

Un valor de MSE alto indica que los valores estimados por medio del modelo están alejados de la serie de tiempo real, por el contrario, un valor pequeño del MSE indica una cercanía en términos de la escala de la serie de tiempo entre los valores reales y los valores estimados. La construcción de los modelos y la evaluación de las métricas dependen fundamentalmente de las reglas del negocio; en el caso de la cotización del par euro -dólar, el valor MSE debe tender a cero, ya que de esta forma el usuario del algoritmo podrá definir una estrategia de trading acertada. Sin embargo, está condición puede resultar utópica considerando la alta volatilidad del mercado y las afectaciones que, aunque se ven reflejadas en el comportamiento provienen de factores externos.

3. DATOS

3.1 DATOS ORIGINALES

Surtido el proceso de descarga de la información, el usuario obtendrá un fichero en formato csv, cuyo espacio en disco no supera los 500 kb. Este archivo contiene seis columnas: fecha, último, apertura, máximo, mínimo y variación porcentual. Donde:

Fecha: Corresponde al día de la cotización del par de divisas euro - dólar.
Último: Corresponde a la cotización de cierre del par de divisas
Apertura: Corresponde a la cotización con la que abre el mercado de divisas
Máximo: Corresponde a la cotización máxima del par de divisas
Mínimo: Corresponde a la cotización mínima del par de divisas
Variación porcentual: Variación de la cotización de cierre del par de divisas frente a la cotización del día anterior. Esta es modificada como parte del procesamiento de la data, dado que, se pretende identificar la tendencia alcista o bajista como parte de un análisis preliminar.

Una inspección ocular básica de la Fig. 1 generada a partir de las columnas Último, Máximo y Mínimo evidencia un comportamiento similar de las series de tiempo, la escala de esta variabilidad es pequeña, es decir, los valores en el tiempo tienen cambios del orden de $1x10^{-2}$ aproximadamente, en consecuencia, esta situación eleva la dificultad en el proceso de modelamiento de los datos.

Por lo expuesto anteriormente y dadas las características inherentes de la data y de los algoritmos para su modelamiento, este proceso se desarrolla con base en la columna “Último”, y consecuentemente las métricas de testeo y validación frente a los datos reales son realizadas en contraste con dicha columna.

Hay que reiterar que, los datos correspondientes a la serie de tiempo que representa la tasa de cambio entre el par de divisas euro - dólar no tiene restricciones de acceso y/o uso. En tal sentido, para su análisis y modelamiento no se ha requerido en ningún caso elevar solicitud a ninguna entidad del orden nacional ni internacional.

3.2 DATASETS

En consideración del dataset y la condición de serie de tiempo que presenta los datos, la creación de los subconjuntos de datos debe realizarse sin alterar el orden secuencial de las cotizaciones euro - dólar en el tiempo, dado que, de esta forma los algoritmos como las redes neuronales recurrentes usadas para el modelamiento de los datos, logran reconocer los patrones secuenciales existentes y definir de esta forma un pronóstico cercano a la cotización del tiempo ***t+1.***

Ahora bien, en cuanto a los algoritmos, las redes neuronales recurrentes requieren como insumo 3 subconjuntos de datos: 1. Para el entrenamiento se ha determinado como parámetro inicial el 70% de los datos, es decir que, los entrenamientos de los modelos realizados durante la etapa de experimentación tienen un total de 3119 observaciones que van desde el 3 de enero del 2005 hasta el 31 de enero del 2022. 2. Para el testeo se precisa el 40% de la data subsiguiente en el tiempo al subconjunto definido para el entrenamiento, con un total de 1984 observaciones. 3. Para la validación, se han elegido de manera arbitraria 589 observaciones.

Figure 2: Subconjuntos de datos para entrenamiento y validación. Fuente: Elaboración propia

Respecto al algoritmo XGBOOST de machine learning, implementado, es necesario indicar que requiere un preprocesamiento particular en el que solo se incluyen los datos de entrenamiento y testeo. En razón a ello, se construye un *dataframe* con base en una observación en el tiempo ***t*** y a partir de procesos explicados en secciones posteriores se generan veintiséis (26) periodos rezagados para cada observación, es decir, cada cotización de cierre (variable a predecir) contiene las cotizaciones de cierre de los últimos veintiún días (variables predictoras), tal como se observa en la Fig. 3.

	Cierre	t-1	t-2	t-3	t-4	t-5	...	t-22	t-23	t-24	t-25	t-26
0	1.2769	1.2759	1.2872	1.2973	1.3036	1.3042	...	1.3054	1.3171	1.3262	1.3272	1.3463
1	1.2805	1.2769	1.2759	1.2872	1.2973	1.3036	...	1.3084	1.3054	1.3171	1.3262	1.3272
2	1.2876	1.2805	1.2769	1.2759	1.2872	1.2973	...	1.3114	1.3084	1.3054	1.3171	1.3262
3	1.2866	1.2876	1.2805	1.2769	1.2759	1.2872	...	1.3265	1.3114	1.3084	1.3054	1.3171
4	1.2963	1.2866	1.2876	1.2805	1.2769	1.2759	...	1.3218	1.3265	1.3114	1.3084	1.3054

Figure 3: Estructura de datos de entrada a modelo XGBOOST. Fuente: Elaboración propia

3.3 DESCRIPTIVA

De acuerdo con la Fig. 4, la serie de tiempo objeto de estudio, representa en el tiempo la cantidad de euros que son necesarios para intercambiar por un dólar estadounidense. Dada la condición de serie de tiempo que tienen los datos analizados, resulta innecesario realizar graficas para la construcción de un análisis exploratorio de los datos, dado que, no aporta información relevante al análisis de la serie de tiempo. En su lugar, se presentan a continuación pruebas estadísticas para determinar la factibilidad del modelamiento del proceso estocástico a través de un modelo autorregresivo ARIMA. Cabe destacar que, algunas de estas pruebas estadísticas pueden ser utilizadas para el proceso de experimentación en la búsqueda de los hiper parámetros en las redes neuronales artificiales.

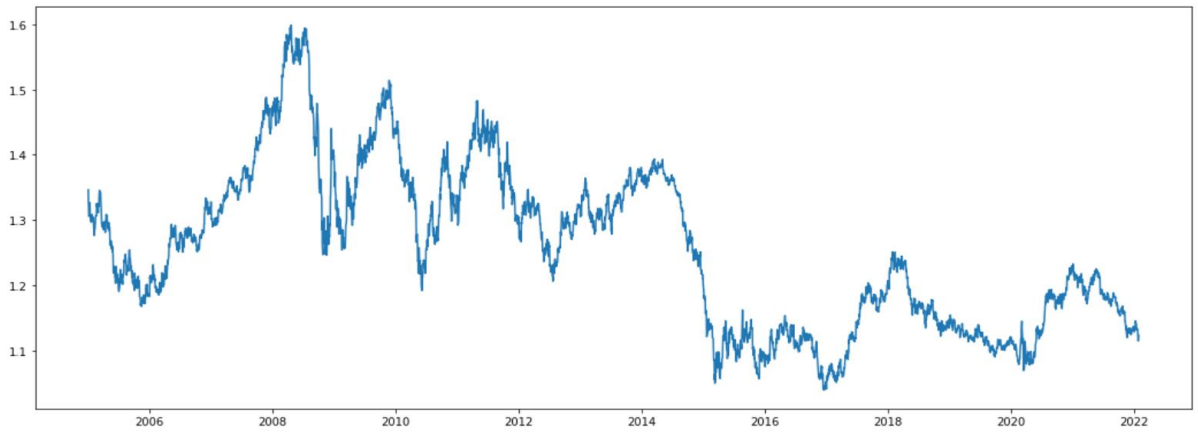

Figure 4: Cotización de cierre Euro-Dólar. Fuente: Elaboración propia

La condición de estacionariedad de una serie de tiempo define si esta presenta media y varianza constante a través del periodo de tiempo analizado. La condición de estacionariedad define si dicha serie de tiempo puedo o no considerar un modelo autorregresivo para predecir su comportamiento. Para ello, la estacionariedad es evaluada con la prueba de hipótesis de Dicky - Fuller en un nivel de confianza del 95%, donde:

$$Ho: La\ serie\ de\ tiempo\ no\ es\ estacionaria$$
$$Hi: La\ serie\ de\ tiempo\ es\ estacionaria$$

Este contraste de hipótesis arrojó un *p-valor* de 0.334875, siendo este mayor al nivel de confianza del 95% establecido para la prueba de hipótesis. En consecuencia, la evidencia estadística señala el rechazo de la hipótesis nula, es decir su media y varianza si cambian en el tiempo y la serie de tiempo no es estacionaria.

Existen transformaciones de los datos que permiten convertir en estacionaria una serie de tiempo que no lo es, puntualmente se utiliza la primera diferencia de valores que está definida como la resta entre el periodo ***t*** y el ***t-1***. En la Fig. 5 se muestra gráficamente cómo queda determinada la nueva serie de tiempo.

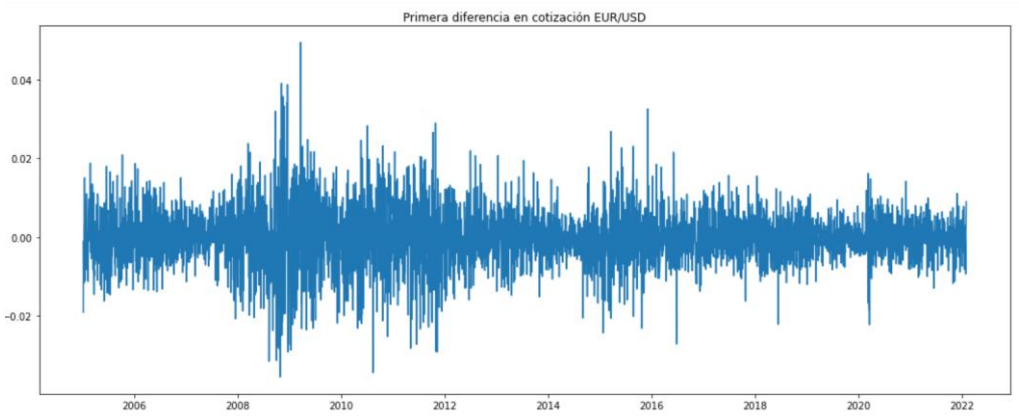

Figure 5: Primera diferencia aplicada a la serie de tiempo. Fuente: Elaboración propia

Al aplicar nuevamente la prueba Dicky - Fuller a la serie de tiempo con la primera diferencia se encuentra que el valor-p es cercano a 0, rechazando la hipótesis nula.

Uno de los aspectos más importantes para la definición de los parámetros del modelo autorregresivo, es la estimación de la autocorrelación de la serie de tiempo que determinan sus rezagos en el tiempo. Esta medida es útil como punto de referencia para predeterminar en los modelos de deep learning los retrasos en el tiempo.

De acuerdo con la Fig. 6, la autocorrelación es cercana a 1 hasta más allá de 50 periodos, con una leve tendencia bajista no significativa. Extendiendo esta medida, se observa que hacía el rezago 65, la autocorrelación disminuye sin cruzar la barrera de 0.8.

Respecto a los valores de significancia, el gráfico sugiere que estos son significativos estadísticamente hasta el número de retraso propuesto preliminarmente. Esta alta autocorrelación encontrada hasta el retraso 65, provee un valor preliminar al hiper parámetro a considerar durante la experimentación con las redes neuronales artificiales.

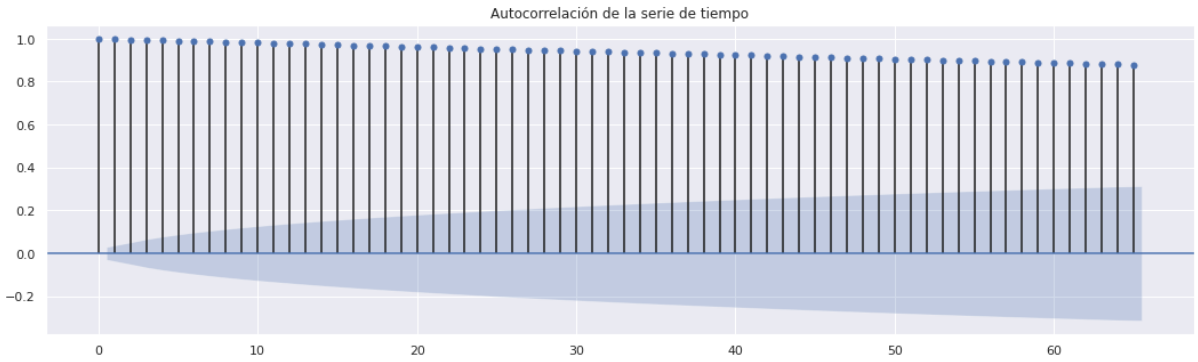

Figure 6: Autocorrelación de la serie de tiempo con 65 rezagos. Fuente: Elaboración propia.

Realizada la descomposición de la serie de tiempo bajo el supuesto que el valor observado en el tiempo ***t*** es la suma de la tendencia, el efecto estacional y el residual, se evalúan los supuestos para la implementación del modelo autorregresivo ARIMA en donde se busca establecer si los residuos responden a dos hechos fundamentales: 1. independencia y 2. normalidad. Ante esta situación, se evalúan los residuos del modelo con base en las pruebas de hipótesis de box-Ljung y Shapiro - Wilk

El contraste de Box–Ljung determina la independencia de los residuos con base en el p-valor y el nivel de confianza de la prueba, que, para el caso particular, se usa un nivel de confianza del 95% y 50 rezagos atrás. El contraste por tanto establece la hipótesis nula y la hipótesis alternativa en los siguientes términos:

$$Ho: Los\ residuos\ son\ independientes$$
$$Hi: Los\ residuos\ no\ son\ independientes$$

El p-valor obtenido para la prueba es 0, con esto existe suficiente evidencia estadística para rechazar la hipótesis nula y definir la no independencia en el componente de los residuos de la serie de tiempo.

Ahora bien, para definir la normalidad en los residuos, se aplica el contraste de shapiro - wilk con un nivel de significancia del 95% y bajo el siguiente supuesto:

$$Ho: Los\ residuos\ se\ distribuyen\ normalmente$$
$$Hi: Los\ residuos\ no\ se\ distribuyen\ normalmente$$

Para este caso, el p-valor obtenido es de 3.0953×10^{-31}, valor que evidentemente es inferior al nivel de significancia y por tanto se acepta la hipótesis alternativa.

Teniendo en cuenta los elementos estadísticos presentados, el modelo autorregresivo ARIMA no es eficiente para modelar la serie de tiempo del par de divisas euro y dólar, dado que, dicho proceso no es lineal y las observaciones provienen de un proceso estocástico.

4. PROCESO DE ANALÍTICA

4.1 PIPELINE PRINCIPAL

En la Fig. 7, se observa el flujo de trabajo de este proyecto, que se define en tres etapas fundamentales: 1. Adquisición de los datos, preprocesamiento de los datos, implementación de arquitecturas y modelos de deep learning y machine learning, por último, la evaluación de resultados.

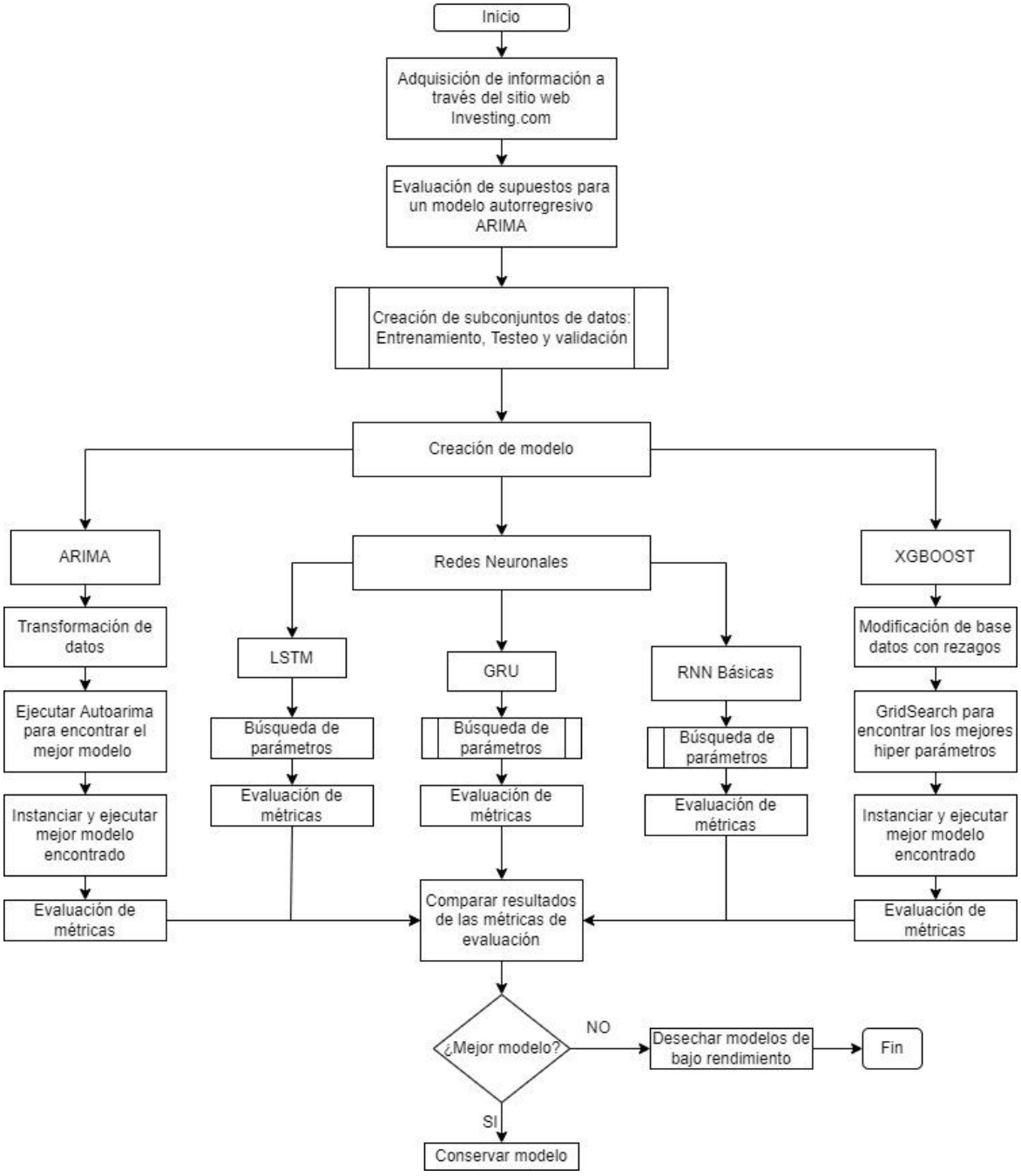

Figure 7: Diagrama de flujo. Fuente: Elaboración propia

4.2 PREPROCESAMIENTO

El escalado y normalización de los datos, es un proceso que debe surtirse en todas las etapas de los procesos de implementación de determinado algoritmo de machine learning y deep learning. Este proceso se surte generalmente, cuando la realidad a modelar comprende varias variables y la razón fundamental está orientada a eliminar los sesgos ocasionados debido a las distribuciones a priori que describen los datos y los valores definidos por la varianza y la media, y escalar todas las variables sobre un mismo rango. En ese sentido, los modelos implementados como estrategia para la predicción en el tiempo de la variable no realizan supuestos sobre la distribución de los datos, lo que conlleva a sesgos en el pronóstico de la serie.

Para la serie de tiempo objeto de estudio, se pretende modelar una única variable, es decir, estamos frente a un proceso estocástico y no lineal del que se busca encontrar una predicción en el tiempo ***t+1*** por medio de los algoritmos de *machine learning y deep learning***.** En tal sentido, los sesgos por las razones expuestas previamente no encuentran lugar en el análisis. Realizado el proceso de experimentación, teniendo como objetivo evidenciar similitudes y diferencias a partir del escalado de los datos; por tanto, no se muestran cambios significativos en el desempeño de las redes neuronales implementadas. Esta particularidad está definida por la condición de la serie de tiempo, aunque existe una alta variabilidad, los valores alcanzados no superan en ningún caso un valor superior a 2. No obstante, la implementación de este tipo de algoritmos de deep learning para conjuntos de datos con alta variabilidad y un rango de valores más amplio, si puede ser afectado por la no implementación del escalado o normalización de los datos y por ende, las predicciones del algoritmo serán distantes de los datos en los periodos ***t+1, t+2,t+3 … t+n.***

Por los motivos expuestos anteriormente, se realiza el escalado de la serie de tiempo analizada, escalado que está comprendido entre [0, 1]. Obtenido este resultado, se ejecutan las redes neuronales para posteriormente, realizar la transformación inversa de las predicciones y lograr la comparación frente a los datos reales.

Además del escalado y normalización de los datos, para la implementación del algoritmo de *machine learning*, se realizó una transformación a la base de datos con el fin de darle mayor poder predictivo al modelo. La transformación como se había descrito anteriormente consta de rezagar los valores ***t*** periodos y crear columnas con los datos rezagados. En la Fig. 8 se muestra el comportamiento de los rezagos, allí se puede ver como en la fila 5 son los valores transpuestos de los anteriores periodos, esto se realiza con la función de pandas llamada “shift”.

	Cierre	t-1	t-2	t-3	t-4	t-5
0	1.3131	1.3186	1.3237	1.3243	1.3203	1.3213
1	1.3110	1.3131	1.3186	1.3237	1.3243	1.3203
2	1.3241	1.3110	1.3131	1.3186	1.3237	1.3243
3	1.3209	1.3241	1.3110	1.3131	1.3186	1.3237
4	1.3344	1.3209	1.3241	1.3110	1.3131	1.3186
5	1.3388	1.3344	1.3209	1.3241	1.3110	1.3131

Figure 8: Preprocesamiento de datos modelo XGBoost. Fuente: Elaboración propia

4.3 MODELOS

4.3.1 ARIMA

La palabra ARIMA es el acrónimo de Modelos Autorregresivos Integrados de Medias Móviles, es una técnica creada por Box y Jenkins en los años 70 del siglo pasado, está es ampliamente utilizada para predecir series de tiempo.

Está metodología está basada en teorías estadísticas y uno de sus principales conceptos son los procesos estocásticos que están definidos como una sucesión aleatoria que están ordenadas cronológicamente. Esta técnica está compuesta por tres componentes, modelos regresivos (AR), medias móviles (MA) y un componente integrador (I).

Se define un modelo autorregresivo (AR) cuando una variable en un periodo ***t*** es explicada por sus periodos predecesores, esto se define en [2] matemáticamente esto se expresa con la eq. (4).

$$Y_t = \phi_0 + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_n Y_{t-n} + \epsilon_t \tag{4}$$

A su vez el componente de medias móviles (MA) está definido por la eq. (5) como una regresión que explica el valor de una variable en el periodo ***t*** en función de un término independiente y una sucesión de errores de periodos independientes [2].

$$Y_t = \mu + \alpha_t + \theta_1 \alpha_{t-1} + \theta_2 \alpha_{t-2} + \ldots + \theta_n \alpha_{t-n} \tag{5}$$

Así el modelo completo ARIMA se determina normalmente por tres componentes, *(p,d,q)* como se muestra en la eq. (6), donde la *p* determina los parámetros de la parte autorregresiva, *d* determina las diferencias para convertir la serie original en estacionaria y el término de integración y *q* los parámetros de la parte de medias móviles en el modelo.

$$Y_t = (\triangle^d Y_t - Y_t) + \phi_0 + \sum_{i=1}^{p} \phi_i \triangle^d Y_{t-i} - \sum_{i=1}^{q} \theta_i \epsilon t - i + \epsilon_t \tag{6}$$

4.3.2 XGBOOST

El nombre XGBOOST es el acrónimo de “Extreme Gradient Boosting” y es considerado el estado del arte del modelo de árboles de decisiones. Esta ha sido una evolución desde el Bagging, pasando por Random Forest y Boosting, hasta mejorar la aplicación de Gradient Boosting, la evolución se evidencia en la Fig. 9.

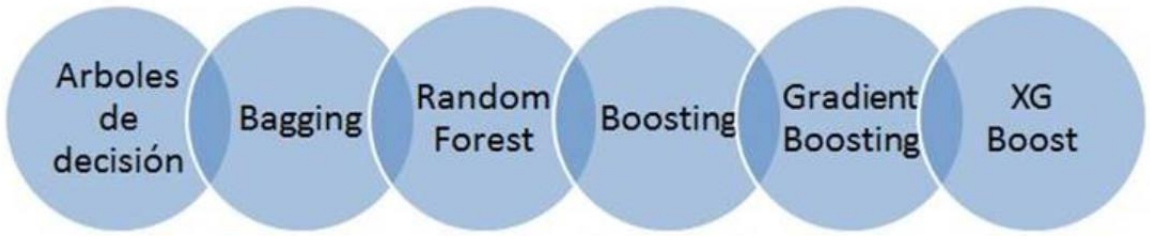

Figure 9: Evolución de los árboles de decisión. Fuente: [3]

Los modelos de árboles de decisión parten de la lógica simple de ir construyendo secuencias lógicas o series de condiciones que dividan el conjunto de datos con el objetivo de llegar a una clasificación que represente adecuadamente la distribución de los datos. Estos modelos suelen representarse gráficamente por un conjunto de nodos y arcos que son llamados raíz, hojas y ramas. A continuación, en la Fig. 10 se muestra la lógica seguida para estos modelos, partiendo desde el nodo raíz se elige un atributo o característica de los datos y se crea la primera condición, dependiendo de esta toma uno u otro camino. Como se evidencia en la imagen el siguiente nodo puede ser un "nodo hijo" o un "nodo hoja", la principal diferencia es que el primero de ellos continúa creando condiciones para mejorar la división de los datos y el segundo es el estado final de la clasificación.

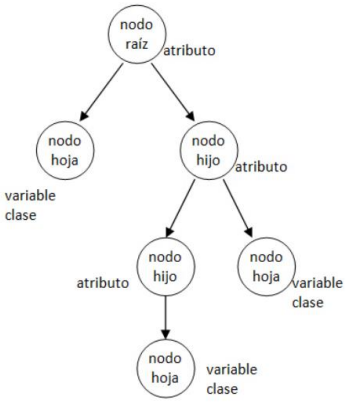

Figure 10: Ejemplo de Árbol de decisiones. Fuente: [4]

Al pasar el tiempo los modelos de árboles de decisiones fueron evolucionando a modelos más complejos como el Random Forest, el cual es la creación de muchos árboles de decisión, pero sus características son elegidas aleatoriamente, según se observa en la Fig. 11, obligando así a crear múltiples árboles diferentes, esto se da principalmente a que las primeras aplicaciones de árboles de decisiones usualmente proponen siempre el mismo árbol de decisión y el sesgo no se mejora. Con esta técnica se da mayor versatilidad al modelo y a su vez mayor poder predictivo.

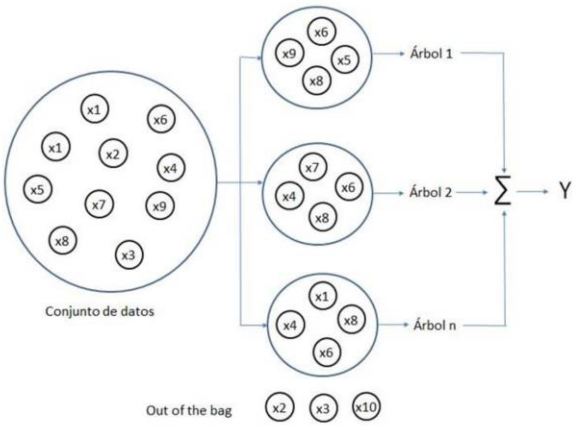

Figure 11: Ejemplo Random Forest. Fuente: [3]

Dentro de la metodología de Random Forest está incluida la técnica Bagging, la cual se puede definir como la votación de cada modelo por una clase y la clase con mayor votación es la determinada como predicción. En la Fig. 12 se puede evidenciar en un caso hipotético donde un modelo de Random Forest construye tres árboles diferentes y al final cada uno de estos determina una clase según su configuración. Allí se puede ver como el primero determina "Class A" y los otros dos "Class B", como la mayoría en este caso es "Class B" la determinación final es está, a este último paso es al cual llamamos Bagging.

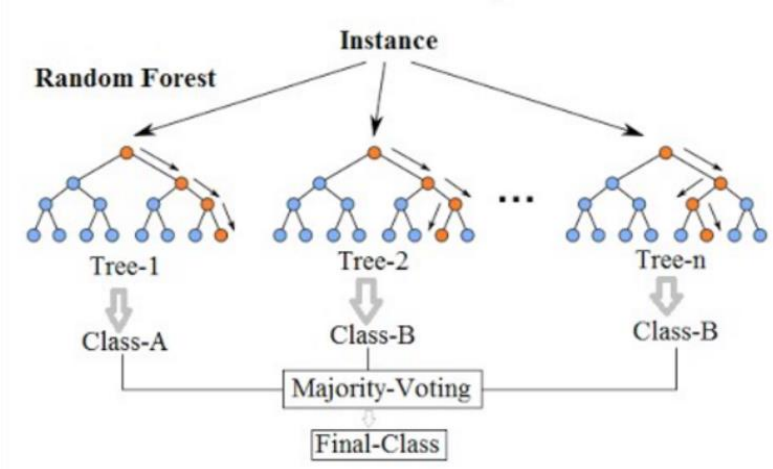

Figure 12: Ejemplo Bagging. Fuente: differencebetween.net

La evolución de estos modelos continuó, llegando a *Boosting*, el cual se define como una secuencia de árboles, donde cada nuevo árbol mejora la predicción de los datos mal clasificados o condicionados del árbol anterior. Como se muestra en la Fig. 13 la principal diferencia entre *Bagging* y *Boosting* está en que el segundo modelo se preocupa por mejorar las predicciones del anterior, en contraste, *Bagging* únicamente se preocupa por lograr mayor diversidad, pero no necesariamente con calidad.

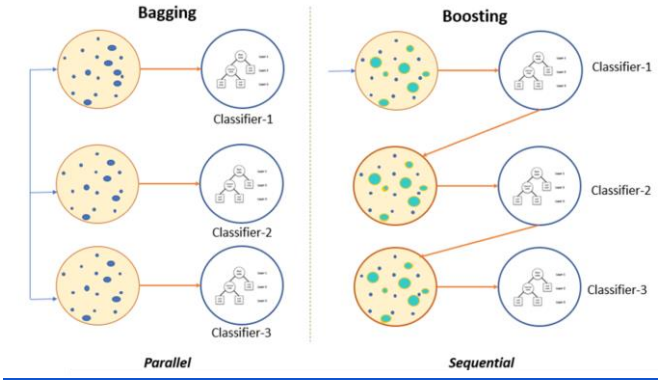

Figure 13: Diferencia entre Bagging y Boosting. Fuente: medium.com

El método XGBOOST es la utilización de árboles de decisión, Random Forest, Bagging y Gradient Boosting, al emplear y ensamblar todas estas técnicas, este se convierte en un modelo altamente potente y con gran velocidad de ejecución. Este es utilizado en problemas supervisados tanto en clasificación como en regresión, también funciona muy bien con grandes volúmenes de datos. El modelo matemático se explica en [5].

4.3.3 REDES NEURONALES RECURRENTE - RNN

La particularidad del conjunto de datos analizados en este documento direcciona su modelamiento hacía las redes neuronales recurrentes. Este tipo de redes neuronales son capaces de adaptarse con relativa facilidad a las secuencias de datos o series de tiempo. Adicionalmente, componentes como el aprendizaje recurrente de cada una de las neuronas que componen la arquitectura de la red a través del backpropagation dan lugar a una reducción sustancial en las métricas de error evaluadas.

La flexibilidad de las redes neuronales recurrentes inunda de posibilidades la construcción de arquitecturas, que incluyen la implementación de LSTM -Long Short Term Memory-, GRU -Gated Recurrent Unit, redes convolucionales entre muchas otras. Evidentemente dichas arquitecturas dependen de las reglas del negocio y la experticia del analista para la comprensión del problema para la predicción del comportamiento en el tiempo del par de divisas euro dólar. Ante tantas posibilidades inherentes a las redes neuronales recurrentes, destacar que los conceptos explicados a continuación son aplicables a todos los tipos de redes artificiales mencionadas con anterioridad.

El componente mínimo de una red neuronal artificial es el perceptrón o neurona, al igual que en el cerebro humano, múltiples neuronas están interconectadas para la resolución de tareas específicas [6], los millones de neuronas y las múltiples arquitecturas existentes en el cerebro humano, le permiten desarrollar múltiples tareas complejas como el razonamiento o el movimiento, a partir de una gran arquitectura flexible. En contraste, desde el enfoque del aprendizaje profundo, sólo contamos con arquitecturas que logran desarrollar tareas específicas que distan ampliamente de lograr razonamientos complejos como lo hacen los seres humanos, sin embargo, es posible encomendar tareas a este tipo de redes que a través de la aplicación del análisis estadístico y la recurrencia logran generalizar complejos procesos estocásticos que se desarrollan en el tiempo.

Dependiendo de la arquitectura de la red neuronal y los parámetros predeterminados, estas son capaces de resolver problemas de regresión para realizar pronósticos de valores reales de una variable que se desarrolla en el tiempo o realizar tareas de clasificación, todo lo anterior, dependiendo de los datos

ingresados y las reglas del negocio. En ese sentido, es necesario indicar que, al interior de cada perceptrón, se desarrollan procedimientos matemáticos de optimización de pesos, similar a una regresión lineal, en la que se emplean los mínimos cuadrados ordinarios para el ajuste de dichos pesos. Ahora bien, estas operaciones son reiterativas en los enésimos perceptrones que componen la red neuronal, cuyos resultados o predicciones son condensados por medio de funciones de activación y neuronas de salida.

En el caso de la redes neuronales recurrentes -RNN-, el perceptrón recibe además del vector definido en el tamaño del lote, el cálculo preliminar de los pesos determinados por un perceptrón en un estado anterior, es decir, las RNN tienen una retroalimentación a cada neurona para la optimización de los pronósticos tal como se observa en la Fig. 14**.** Cabe resaltar que, esta retroalimentación queda definida desde la segunda capa de perceptrones de la red neuronal, dado que, la primera iteración sólo está en capacidad de recibir una inicialización [7]. La bibliografía disponible en relación con la inicialización de la red neuronal propone funciones ligadas a distribuciones normales, uniformes, sus variaciones, entre muchas otras, esto con el fin de optimizar el potencial de la red neuronal para la generalización del modelo, promover la velocidad para la convergencia y por supuesto la obtención de resultados acordes con la realidad modelada.

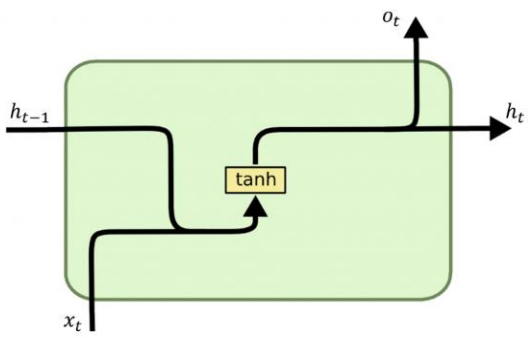

Figure 14: Esquema básico de una neurona en una red neuronal recurrente. Tomado de: [8]

Se mencionó previamente que, la neurona recibe dos entradas, el vector correspondiente a la data y el estado anterior de los pesos calculados. En consecuencia, la salida de esta neurona está dada por un conjunto de pesos que se convierten en el tiempo ***t+1*** en el estado previo de la red neuronal y un vector de longitud 1 o mayor a uno que se constituyen en el pronóstico a la serie de tiempo [8]. En este punto, es necesario indicar que, la arquitectura conserva una matriz de pesos calculados en cada iteración de la red neuronal y una matriz de pesos que se consideran el estado anterior de la red Ver eq. (7); finalmente esta recurrencia facilita el cómputo de matrices, deriva en resoluciones de pronósticos óptimos y una convergencia rápida del algoritmo

$$y_t = \phi(W_x^T X_t + W_y^T X_{t-1} + b) \tag{7}$$

Replicando el ejercicio y dada la flexibilidad de las redes neuronales, es posible conformar capas, que no es más que la agrupación de neuronas que ejecutan tareas bajo parámetros predeterminados. Entonces, esta primera capa, estaría encargada del ajuste primario de los pesos considerando la respectiva inicialización, que en el flujo de trabajo mutan a un estado previo y que para el tiempo ***t+1,*** dicho estado previo se ejecuta en la segunda capa para mejorar la optimización de los pesos [9]. Este flujo de trabajo se repite en relación con la cantidad de capas que componen la arquitectura propuesta.

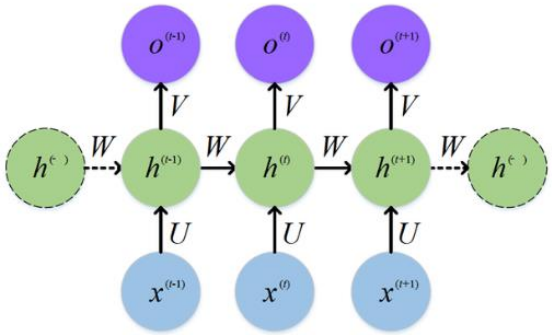

Figure 15: Esquema de red neuronal recurrente con neuronales en secuencia. Modificado de: [10].

La matriz de pesos del estado previo ***t+1*** y del estado presente ***t*** dotan a la red neuronal recurrente de una memoria cortoplacista que define el pronóstico de la serie de tiempo en un tiempo ***t+10*** [11]**,** es decir que, una secuencia de entrada puede bajo este supuesto, estimar uno o hasta 10 valores, teniendo en cuenta que la precisión del pronóstico decae en la medida que avanza el tiempo, la arquitectura definida para la red neuronal recurrente; evidencia de ello, es el aumento de las métricas de error evaluadas en el proceso. No obstante, esta definición a priori de 10 valores atrás para definir 10 pronósticos en el comportamiento futuro de la variable queda supeditado a las reglas de negocio y a la arquitectura de la RNN.

En este punto, si se implementa un algoritmo con las condiciones explicadas hasta aquí, probablemente, no obtendría resultados óptimos en sus pronósticos y las funciones de optimización podrían ser erráticas en la búsqueda de los parámetros para la optimización de la función de costo. En consecuencia, la ejecución del algoritmo podría tardar mucho más tiempo de lo previsto exigiendo mayor recurso computacional o simplemente no encontrando la convergencia del algoritmo.

Por ello, se implementa una condición adicional que se ejecuta paralelamente y que permite actualizar el ajuste de los pesos de cada neurona en cada una de las capas, a este procedimiento se le denomina *backpropagation*. Esta herramienta adicional es un proceso que se desarrolla en dos vías. La primera ya ha sido descrita como la optimización de los pesos, la configuración de los estados previos y su paso a la siguiente capa, además de la recurrencia de las neuronas para la optimización de la generalización en la serie de tiempo. Ahora bien, al tratarse de un aprendizaje supervisado, se define un error relativo al pronóstico y es allí donde el algoritmo de *backpropagation* identifica el aporte a dicho error por cada una de las capas y neuronas [12]. Bajo este contexto, se establece para cada componente de la red neuronal un error ponderado y su objetivo, a partir de ese momento es disminuir este valor en las siguientes iteraciones. Realizando un símil, suponga una vía que va de norte a sur y que cuenta con dos carriles, un carril tiene dirección norte sur y el contiguo tendrá por tanto dirección sur - norte. En el carril norte - sur, el algoritmo optimiza los pesos que se ajusten mejor a los datos, retroalimenta la siguiente capa con el estado anterior y establece por tanto una predicción. En el carril de sur a norte, el algoritmo establece un error con base en los datos de entrenamiento y validación, fija el aporte del error para cada capa y neurona, buscando que en la próxima iteración la disminución del error relativo.

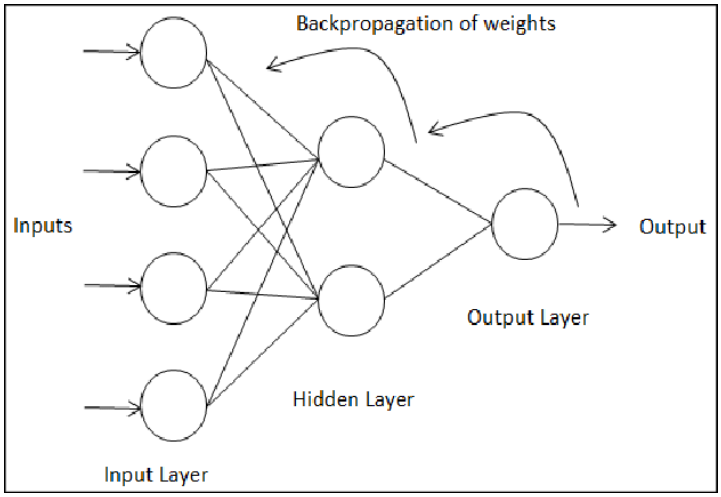

Figure 16: Esquema básico del funcionamiento del algoritmo de backpropagation. Modificado de: [12]

Uno de los aspectos más importantes para la convergencia del algoritmo es la optimización de la función de coste, previamente en un símil con la regresión lineal se indicó que, dicha optimización se desarrollaba con base en los mínimos cuadrados ordinarios, una forma similar para ver desde otra perspectiva esta situación es a través de la función derivada que corresponde con la recta tangente a una función de una única variable, igualar ese resultado a 0 y determinar de esta forma los mínimos absolutos de dicha función. Para la resolución de problemas más complejos, recurrimos a la misma solución, sólo que, para este caso, la función de coste está definida por múltiples variables y en tal sentido, el cálculo vectorial proporciona una herramienta poderosa que permite encontrar los mínimos o máximos de una función, la función gradiente, que, desde el enfoque del aprendizaje profundo, el algoritmo se denomina el gradiente descendente.

Considerando este contexto, el algoritmo calcula a través de las derivadas parciales la dirección en la cual la función podría tener un mínimo, sin embargo, en este punto se desconoce si este es mínimo local o global. Ahora bien, encontrado este nuevo punto, se calcula un nuevo gradiente en la dirección de un posible valor mínimo y se repite el algoritmo hasta encontrar el valor mínimo global de la función [13].

En términos de la función del gradiente descendente, es necesario señalar que, encontrada la dirección en la cual se debe mover, la pregunta que surge es ¿cuánta distancia se debe mover? Por ello, se asigna una tasa de aprendizaje para que el algoritmo defina con base en los cálculos matemáticos previstos, las condiciones mínimas del vector, que conlleva a desplazarse desde el punto A hasta el punto B, en repetidas ocasiones.

La definición de esta tasa de aprendizaje es fundamental, dado que, una asignación errónea podría derivar en la no convergencia del algoritmo. Por ejemplo, tasas de aprendizaje relativamente grandes, provocan grandes desplazamientos sobre la función y en consecuencia no se optimiza la función de coste. En el caso contrario, una definición de la tasa de aprendizaje muy pequeña deriva en un coste computacional demasiado alto, adicionalmente, el vector de búsqueda para la optimización de la función podría quedar atrapado en un mínimo local.

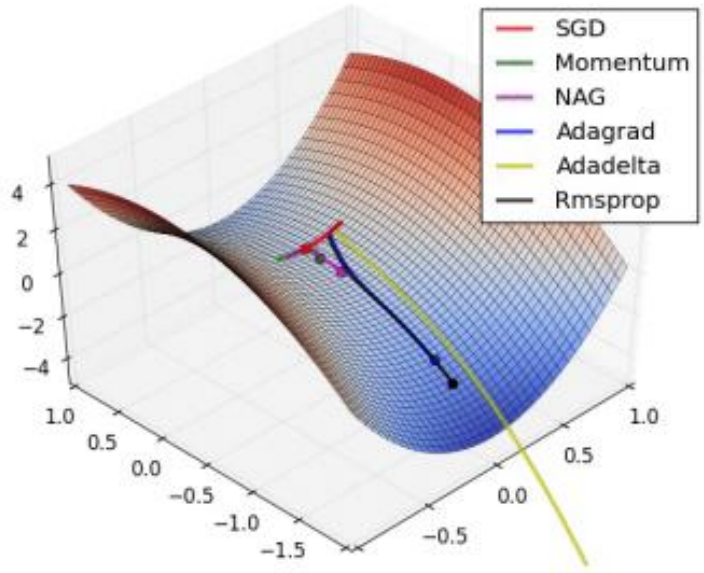

Figure 17: Esquema básico del algoritmo del gradiente descendiente. Tomado de: [13]

Por tal motivo, el algoritmo del gradiente descendiente es objeto de profundos análisis que equiparen las condiciones para la optimización; por una parte, la utilización mínima de recursos computacionales y de otro lado la convergencia rápida y efectiva del algoritmo sobre la función de coste.

Por lo expuesto, el algoritmo de gradiente descendente ADAM [14], -Adaptive moment estimation-, proporciona una herramienta potente para la optimización de la función de coste, fundamentada en el cómputo y actualización de la tasa de aprendizaje sobre la ejecución del algoritmo, minimizando los tiempos y recursos computacionales.

Con la implementación de ADAM, se recogen ventajas propuestas de algoritmos como RMSProp y AdaGrad, donde funciones objetivo con gradientes irrelevantes o series de tiempo provenientes de procesos estocásticos son optimizadas eficientemente. En el caso particular de RMSProp, la variación de la tasa de aprendizaje acelera el proceso de optimización, sin embargo, su uso tiene restricciones respecto al cómo cambia la tasa de aprendizaje durante la ejecución y como los parámetros β_1 y β_2 que controlan su variación en el tiempo permiten la convergencia del algoritmo en menos iteraciones. En consideración de los datos, RMSProp no tiene un rendimiento aceptable con grandes conjuntos de datos y/o donde el gradiente de la función a optimizar no es relevante por lo que el parámetro β_2 debe ser cercano a uno, la implementación en este tipo de casos conlleva a ejecutar múltiples iteraciones y disminuir la probabilidad de convergencia del algoritmo [14].

Respecto a AdaGrad, este algoritmo de optimización también tiene una actualización de la tasa de aprendizaje durante la ejecución del algoritmo, en la cual incorpora el conocimiento adquirido en previas iteraciones sobre la geometría de los datos y la función que se pretende optimizar, en tal sentido, las primeras iteraciones tienen una ejecución lenta y a medida que avanza el proceso, las iteraciones cada vez son más veloces. Al igual que RMSProp, las desventajas de AdaGrad también están relacionadas con la funciones de optimización cuyo gradiente no es relevante, en estos casos AdaGrad tiende a reducir prematura y excesivamente la tasa de aprendizaje [14], haciendo que la convergencia del algoritmo requiera de muchos más recursos computacionales

Revisadas las ventajas y desventajas de dos de los algoritmos para la optimización de funciones a través del gradiente estocástico descendente, se observa que los mayores inconvenientes de este tipo de algoritmos están relacionados con funciones cuyo gradiente sea escaso y consecuentemente las tasas de aprendizaje no ayudan eficientemente a la convergencia del algoritmo.

Entonces, el algoritmo de Adam proporciona soluciones a dichas situaciones con cambios puntuales en su implementación. El cambio fundamental está determinado por llevar a cabo la actualización de los parámetros tal como se desarrolla en RMSProp y AdaGrad, a través de medias móviles exponenciales que son calculadas y controladas por la media y la varianza del gradiente y que realiza correcciones al sesgo resultante en cada iteración. Esto evita la reducción, estancamiento o minimización de la tasa de aprendizaje, acelerando la convergencia del algoritmo y disminuyendo las iteraciones en la etapa de aprendizaje [14].

4.3.3.1 LONG SHORT TERM MEMORY - LSTM

Hay que recordar que, las neuronas al interior de una arquitectura de red neuronal recurrente reciben como entrada una vector de una variable aleatoria que se desarrolla en el tiempo y adicionalmente, requieren conocer el estado de la iteración anterior, esto otorga a la red, la facultad de memorizar en el corto plazo el comportamiento de determinada variable.

Sin embargo, los grandes conjuntos de datos temporales provenientes de procesos estocásticos requieren para la obtención de un pronóstico idóneo, recordar valores con ocurrencia mucho más atrás en el tiempo, es decir, que para el tiempo t, el pronóstico está dado por los tiempos ***t-1, t-2, t-3 … t-n.*** Para ello la arquitectura mínima de la neurona cambia en relación con la presentada para las redes neuronales recurrentes, otorgando plenamente la capacidad de recordar los valores de la serie de tiempo en el pasado.

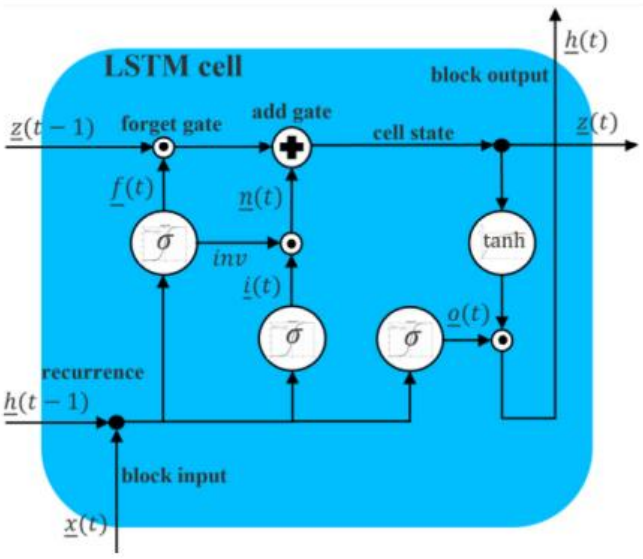

Figure 18: Esquema básico de una neurona en una red neuronal LSTM. Fuente: [15]

La condición de recurrencia le permite a una neurona retroalimentar su proceso y establecer correctamente los parámetros de los datos que ingresan en cada iteración; ahora bien, como se mencionó anteriormente, la estructura para LSTM cambia desde la perspectiva de la unidad fundamental, teniendo para este caso cuatro neuronas que cumplen distintos roles dentro del funcionamiento.

Considere cuatro neuronas dispuestas en paralelo, que reciben al mismo tiempo un vector como subconjunto de los datos, ya sea en la etapa de entrenamiento, testeo, validación o para la predicción, y un estado anterior de los pesos que pretenden ser ajustados [8] y [15]. Con base en estos insumos, los roles de cada elemento se definen así: de acuerdo con la eq. (8) la función ***f*** en el tiempo ***t*** define que parte del comportamiento de la serie de tiempo guardada en la memoria de la neurona debe ser eliminada.

$$f(t) = \sigma(W_{xi}^{T} X(t) + W_{hi}^{T} h((t-1) + b_f) \tag{8}$$

Respecto con la eq. (9), la función ***c*** en el tiempo ***t*** calcula los parámetros para la función de optimización, tal como se realiza en el modelo básico de redes neuronales recurrentes.

$$c(t) = f(t) \otimes c(t-1) + i(t) \otimes g(t) \tag{9}$$

Considerando la eq. (10), la función ***i*** en el tiempo ***t*** establece que porción del comportamiento de la serie de tiempo se alojará en la memoria para dicho periodo.

$$i(t) = \sigma(W_{xo}^{t} X(t) + W_{hi}^{T} h(t-1) - b_i) \tag{10}$$

Por su parte, la función ***O*** en el tiempo ***t,*** recibe al igual que en los casos anteriores los datos y el estado anterior, y realiza el ajuste de los pesos correspondientes de forma similar a lo expuesto en la eq. (10). De esta manera, los resultados preliminares pasan por la función de activación, como se muestra en la eq. (11), para ser usado en la siguiente iteración.

$$g(t) = tanh(W_{xg}^{T} X(t) + W_{hg}^{T} h(t-1) b_g) \tag{11}$$

Cabe destacar que, los resultados de la función ***f*** se eliminan a través de la "puerta de olvido" y por la "puerta de salida" que terminan por transferirse al estado anterior, como se observa en la Fig. 18. En cuanto al vector de salida, está dado por el producto de tensores calculados en eq. (10) y la eq. (11), que se sintetiza en la eq. (12)

$$y(t) = g(t) \otimes i(t) \tag{12}$$

En este punto, se ha intentado desagregar los elementos y los procesos que se desarrollan al interior de la unidad mínima de esta arquitectura, sin embargo, todo lo mencionado anteriormente ocurre simultáneamente y suele generar confusión y presentarse como un proceso de caja negra [16].

4.3.3.2 GATED RECURRENT UNIT - GRU

Propuestas en 2014 por [17], las redes neuronales GRU son una simplificación de las redes neuronales LSTM detalladas previamente. Nuevamente la idea fundamental está basada en entregarle a cada neurona el estado previo de los parámetros encontrados en la iteración y a partir de ellos, iniciar un nuevo proceso de cálculo que evidentemente está acotado por las repeticiones previas.

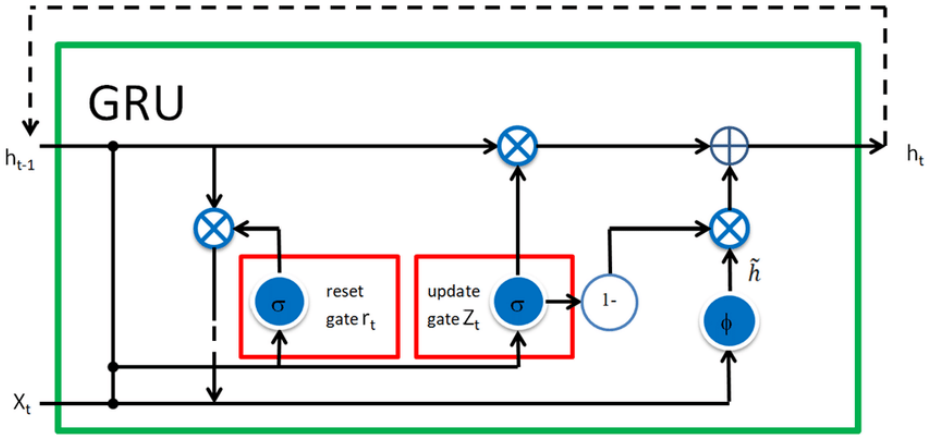

Figure 19: Esquema básico de una neurona en una red neuronal GRU Fuente: (Yang et al. 2020)

La arquitectura interna de la unidad mínima de la red neuronal cambia, considerando que, disminuyen la cantidad de elementos básicos visto en las redes neuronales LSTM y en consecuencia decrecen los cálculos realizados en cada iteración. De izquierda a derecha, el primero componente tiene como entrada el estado previo h_{t-1} de los pesos calculados y el vector o secuencia en el tiempo ***t,*** información con la cual define qué información queda o se borra definitivamente de la memoria de la neurona. En igual sentido, las redes neuronales GRU, ejecutan un procedimiento para la estimación de los pesos igual al ya descrito por la eq. (8). No obstante, radica una diferencia sutil, enmarcada en el manejo de la memoria para guardar determinadas secuencias o eliminarlas según corresponda. Al comparar la Fig. 18, se observa cómo en cada iteración resultaba para el primer caso un h_t correspondiente al estado anterior de los parámetros y adicionalmente un estado c_t que se puede catalogar como el residuo de cada iteración, esto promoviendo una memoria mucho más grande para las redes neuronales LSTM. El cambio a esta situación desde el enfoque de las redes neuronales GRU, está dado por la eliminación de esta condición, y solo considerando una salida binaria en la "puerta de olvido", con lo cual se borra o mantiene toda la información definida por la Ecuación eq. (8)**.**

La última neurona que compone la célula, al igual que las anteriores, también recoge información de entrada representada por el vector de la secuencia y el estado previo del proceso, y allí determina el vector de salida, que está supeditado al estado del proceso en el momento *t*, determinado por la función de activación en la eq. (11) *y* la función que contiene la memoria de la célula neuronal dada por la eq. (10)

Hay que recordar que, la recurrencia y memoria de este tipo de redes neuronales artificiales no deja a un lado en ningún caso, la implementación en paralelo de los algoritmos de *backpropagation* y del gradiente estocástico descendiente descritos en secciones previas. Es por ello, que este tipo de herramientas son utilizadas ampliamente para la resolución de problemas no lineales como las series de

tiempo, la predicción del tráfico [18], el procesamiento natural de lenguaje, reconocimiento de anomalías [15] y por supuesto para el pronóstico del comportamiento de variables económicas [16], entre muchos otros.

Adicionalmente hay que indicar que, las redes neuronales artificiales expuestas debido a su condición de flexibilidad permiten un amplio margen de maniobrabilidad para la definición de la arquitectura a implementar para el modelamiento de los datos objeto de estudio. Puntualmente, la serie de tiempo correspondiente al par de divisas euro - dólar, es modelado haciendo uso de la API de Keras, enfocada fundamentalmente en capas densas conectadas profundamente. En secciones posteriores se presentarán las arquitecturas de redes neuronales artificiales definidas durante el proceso de experimentación.

4.5 MÉTRICAS

Como se mencionó anteriormente, la arquitectura de las redes neuronales que se proponen, se establece a partir de capas altamente conectadas, disponibles en la API de Keras. La implementación de este algoritmo requiere de la definición preliminar de la función de pérdida, calculada en cada una de las iteraciones durante la experimentación y cuya minimización representa la convergencia del algoritmo. Considerando las reglas del negocio y buscando una generalización óptima de la serie de tiempo, se busca que la métrica establecida se minimice tanto como sea posible, sin afectar por supuesto la generalización que se tiene por objetivo. La API de Keras dispone de las siguientes métricas como opciones: error cuadrático medio -MSE-, error medio absoluto -MAE y error medio porcentual absoluto -MAPE.

Cabe destacar que, la definición de la función de pérdida y la función de evaluación del rendimiento del modelo puede ser mejorada en relación con su implementación. Por defecto, las funciones propuestas evalúan en su totalidad los pronósticos frente a los datos reales, lo que conlleva a un tiempo de ejecución mayor para lograr la convergencia del algoritmo. Las mejoras en la implementación de estas funciones pueden considerarse desde la personalización de dichas métricas de evaluación, en la cual sólo se evalúa un único pronóstico frente a un único valor real, que por las condiciones del algoritmo de *backpropagation* y del gradiente descendiente estocástico logran una convergencia más efectiva, en menor tiempo y con menor recurso computacional.

De otra parte, aunque no es posible considerar el número de iteraciones y el número de neuronas por capa como métricas de rendimiento, estos elementos sí definen el costo computacional y tiempo de ejecución, por tal motivo, durante la etapa de experimentación se buscan los mejores hiper parámetros para la ejecución del algoritmo, con el objetivo principal de la convergencia efectiva del algoritmo y de la minimización de las métricas de error.

5. METODOLOGÍA

5.1 BASELINE

Los modelos ARIMA han sido uno de los modelos más utilizados para la predicción de series de tiempo, estos han funcionado adecuadamente por décadas en diferentes escenarios y con distintos datos. Es por esta razón que nuestro modelo base es el ARIMA y por el cual pretendemos iniciar la predicción de la serie de tiempo del par de divisa EUR/USD. Es bueno aclarar que el ARIMA por ser un modelo autorregresivo utiliza sus propios datos históricos como predictores y no utiliza variables exógenas para realizar sus predicciones.

Como se ha expuesto anteriormente la serie de tiempo del par de divisas euro y dólar no es estacionaria y por tal motivo se recomienda realizar transformaciones a los datos para convertirla y así poder aplicar teóricamente de forma adecuada el ARIMA. Las transformaciones utilizadas en este proyecto son la primera y la segunda diferencia en la serie de datos.

Mediante la aplicación de la función "auto_arima", la cual se encarga de ejecutar automáticamente diferentes configuraciones de modelos ARIMA, se encuentra la mejor combinación de parámetros la cual minimiza el error. Para la primera ejecución se utiliza la primera diferencia como transformación de los datos. los mejores parámetros encontrados en la primera iteración fueron ARIMA(0,0,0)(0,1,2)[12], dando un error MSE de $6.4 * 10^{-05}$ y un MAPE de 0.5906%. A continuación, se agrega la Fig. 20 donde se visualiza las predicciones respecto a los datos reales.

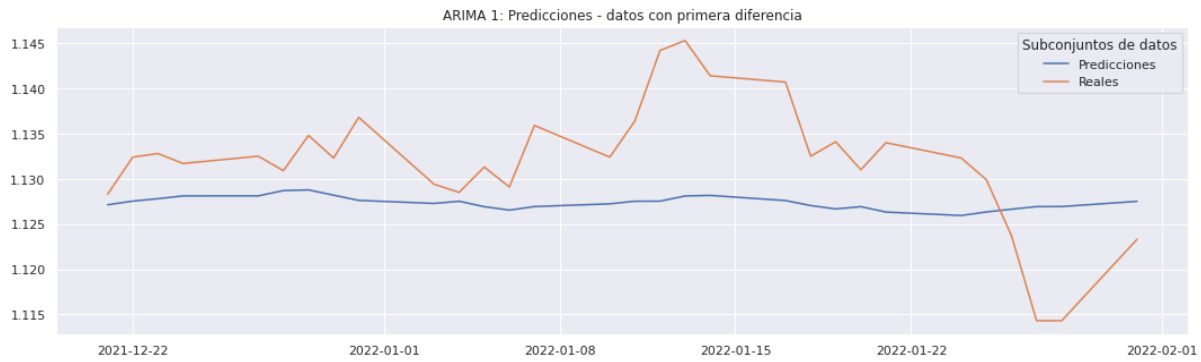

Figure 20: Modelo ARIMA 1, Predicciones y datos con primera diferencia. Fuente: Elaboración propia

Bajo la premisa de exploración para mejorar los resultados se aplica nuevamente la función "*auto_arima*" pero esta vez con la segunda diferencia como transformación de los datos. En este caso el mejor modelo fue ARIMA(2,0,1)(2,2,0)[7] donde el MSE tuvo un resultado de 0.001814 y el MAPE de 2.9577%. A continuación, en la Fig. 21 se muestran los resultados:

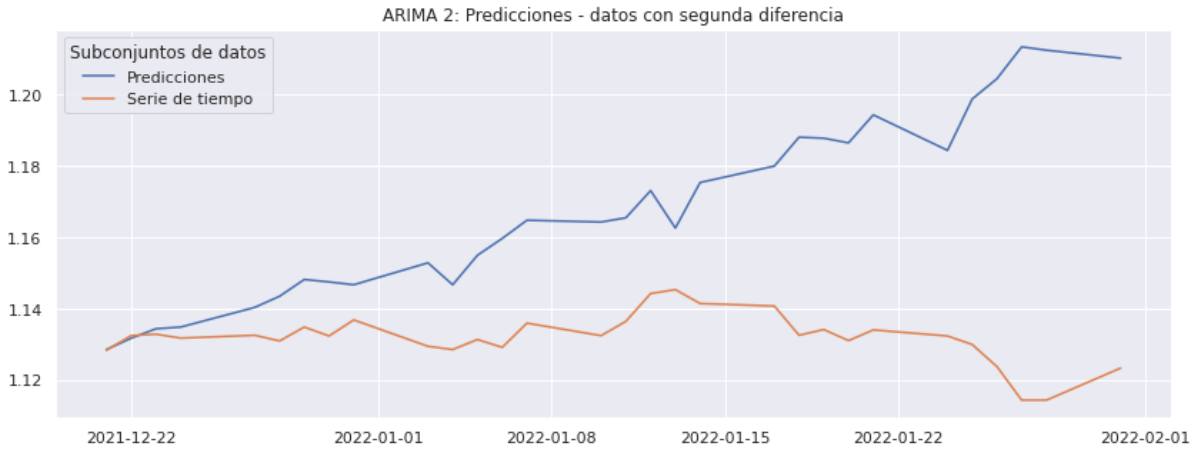

Figure 21: Modelo ARIMA 2 - Predicciones - datos con segunda diferencia. Fuente: Elaboración propia

Como último recurso se ejecutó la función de "auto_arima" sin transformar los datos, es decir, sin calcular ninguna diferencia. El modelo resultante fue ARIMA(3,0,1)(2,2,0)[7], para esta configuración el error MSE fue de $6.93 * 10^{-05}$ y el MAPE de 0.5206% . Ver la Fig. 22 para evidenciar los resultados

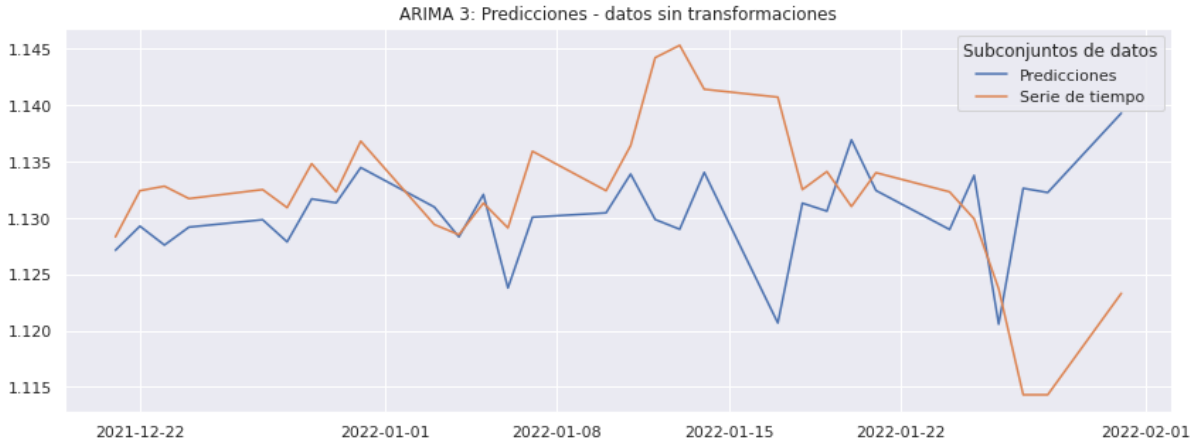

Figure 22: Modelo ARIMA 3 - Predicciones - datos sin transformación. Fuente: Elaboración propia

5.2 VALIDACIÓN

De acuerdo con la Fig. 2, para la implementación de las redes neuronales se realizó una subdivisión de los datos, comprendida por los datos de entrenamiento, testeo y evaluación. En consideración de la condición de serie de tiempo, está elección no se realiza de forma aleatoria, dado que, se perdería la continuidad del proceso estocástico, en su lugar, los tres segmentos son contiguos y dan continuidad sobre la serie de tiempo.

Por lo expuesto, la validación de los pronósticos se realiza evaluando los tres segmentos de acuerdo con el modelo de red neuronal implementado. Evidentemente, las métricas para la etapa de entrenamiento serán inferiores a las que se obtienen para el testeo y la validación, estas métricas serán presentadas en secciones posteriores.

Respecto a los modelos ARIMA no aplica la partición del dataset en entrenamiento, validación y prueba. Por tal motivo se utilizan todos los datos para calcular el mejor modelo.

En el modelo de XGBOOST se utiliza la técnica de *cross-validation*, la cual funciona generando ***n*** particiones a los datos de entrenamiento y luego los utiliza para validar su funcionamiento y corregir la configuración con el objetivo de mejorar la capacidad predictiva del modelo. Esta técnica es supremamente valiosa para evitar sobre entrenamiento del modelo. Para el modelo XGBOOST se definieron 5 folds que ayudarán a mejorar el rendimiento.

5.3 ITERACIONES y EVOLUCIÓN

5.3.1 XGBOOST

Recientemente uno de los métodos de machine learning más utilizados en las competencias de Kaggle son los gradientes *boosting* del que sobresale el XGBOOST, en el siguiente apartado se expondrá el desarrollo que se realizó con esta técnica y algunas funciones utilizadas para encontrar la mejor configuración de hiper parametros.

La primera iteración que se realiza con este modelo es con una base de datos de 14 rezagos que se ha elegido arbitrariamente y al mismo tiempo se cuenta con los parámetros por defecto. Lo anterior se realiza para tener un punto inicial de comparación luego de optimizar los hiper parámetros. Los resultados obtenidos se muestran en la Fig. 23, el MSE resultante fue $1.9413 * 10^{-05}$ y el MAPE de 0.3089%

El número de rezagos en la base de datos se vuelve una transformación importante para la predicción, por tal motivo se crea un automatismo donde ejecute el modelo XGBoost sin modificación de hiper parámetros en el intervalo 3 a 30 rezagos. a partir de allí se organizan de menor a mayor MSE encontrando que la mejor configuración de base de datos es con 26 rezagos. Se muestran los errores resultantes en la Table 1.

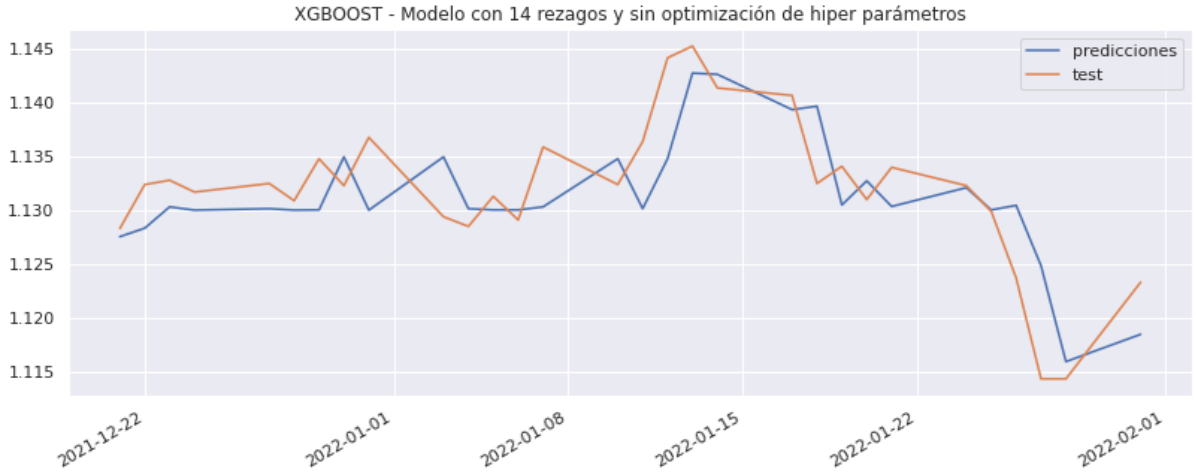

Figure 23: XGBoost - Modelo sin optimización de hiper parámetros. Fuente: Elaboración Propia

	Rezagos	MSE	MAPE
1	lags: 26	0.000018	0.3069%
2	lags: 28	0.000019	0.3030%
3	lags: 6	0.000019	0.3093%
4	lags: 17	0.000019	0.3100%
5	lags: 27	0.000019	0.3093%
6	lags: 18	0.000019	0.3122%

Table 1: Errores MSE y MAPE con diferentes rezagos. Fuente: Elaboración propia

El siguiente paso fue utilizar un Pipeline para encontrar la mejor configuración de hiper parámetros para el modelo, el primer paso del pipeline es escalar los datos con la función *Standard Scaler*, luego se crea una instancia del modelo *XGBoost*. Con ayuda de la función *GridSearchCV* se pretende evaluar miles de configuraciones con la combinación de tres hiper parámetros (*subsample, max_depth, colsample_bytree*) y a su vez se configura con cinco (5) folds para la validación cruzada. La mejor configuración del modelo se determina por subsample = 0.65, max_depth = 3 y colsample_bytree = 1. El MSE resultante fue $2.0806 * 10^{-05}$ y el MAPE de 0.3206%.

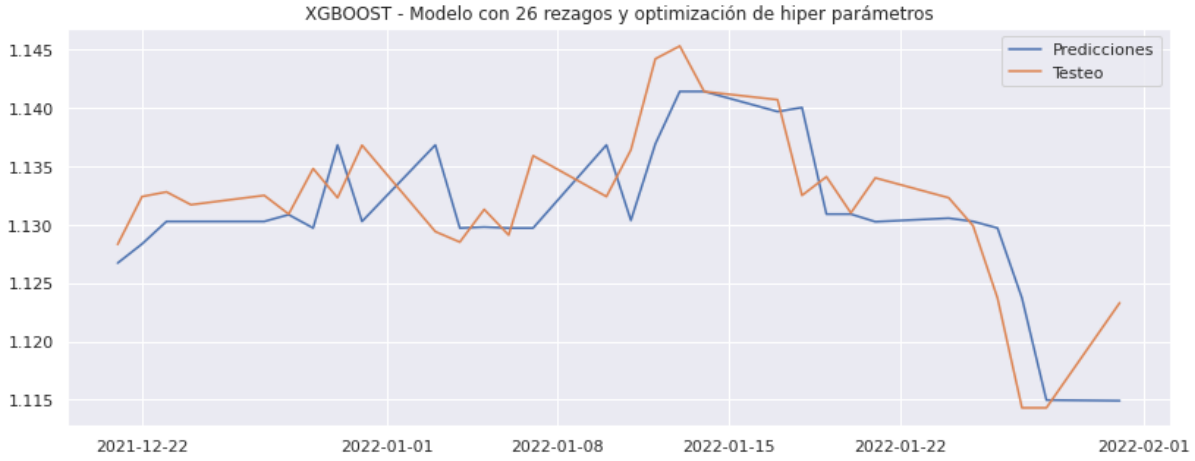

Figure 24: XGBoost - Modelo con optimización de hiper parámetros Fuente: Elaboración propia

5.3.2 REDES NEURONALES RECURRENTES

A partir de la revisión bibliográfica, se ha evidenciado que, las redes neuronales recurrentes y especialmente, las LSTM y GRU son opciones reales para modelar series de tiempo provenientes de diversos campos de estudio. En relación con la cotización euro - dólar y su desarrollo en el tiempo, se ha demostrado su eficiencia y precisión en los pronósticos para la aplicabilidad en tiempo real [19]. Sin embargo, es objeto amplio de estudio, la definición de arquitecturas y sus hiper parámetros, situación que solo es posible analizar a través de la experimentación.

Por lo expuesto, a continuación, se proponen tres experimentos de redes neuronales, correspondiente cada uno a: una red neuronal LSTM, GRU y RNN simple. Bajo este contexto, se realiza la búsqueda de hiper parámetros para las arquitecturas definidas preliminarmente; sobre ellas se realizará la búsqueda de los siguientes hiper parámetros y condiciones que se relacionan a continuación:

1. Definición de la cantidad de capas densas en la arquitectura. Para este caso, se define un vector unidimensional entre 1 y 5
2. Establecer la cantidad de neuronas por cada capa, para ello se define un arreglo que va de 16 hasta 256 neuronas y cada iteración de prueba avanza 5 pasos sobre el arreglo definido.
3. Para la inicialización de la tasa de aprendizaje para el algoritmo de gradiente descendiente, Adam, se define un arreglo que contiene 3 valores: $1x10^{-2}, 1x10^{-3}$ y $1x10^{-4}$.

Con base en los rangos establecidos para los tres casos presentados, la búsqueda de hiper parámetros arrojó los siguientes resultados:

Se realiza la experimentación con las redes neuronales recurrentes, se propone una arquitectura basada en este tipo de redes, sobre la cual se realizará la búsqueda de capas e hiper parámetros en los mismos términos expresados para las redes neuronales LSTM y GRU

De acuerdo con la Fig. 25, la optimización de capas e hiper parámetros, queda definida en dos capas, la primera de ellas recibe los datos e inicia con 36 neuronas, a continuación dos capas con 56 neuronas, de las que es necesario indicar que, retornan sus respectivas secuencias. Se agrega una capa Flatten para la reducción de la dimensionalidad y finalmente una capa densa con una única neurona que recoge las conexiones de la capa inmediatamente anterior. Esta arquitectura queda establecida por 6049 parámetros.

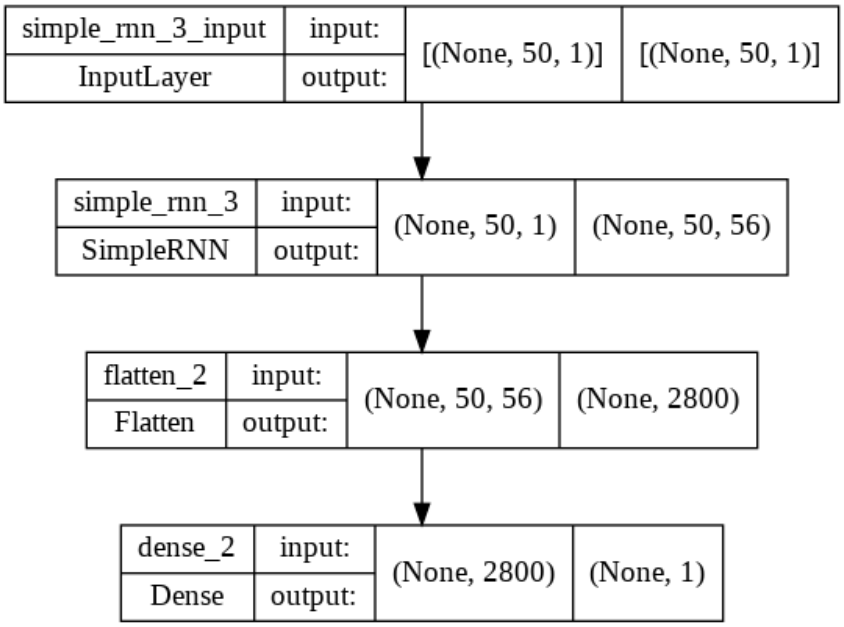

Figure 25: Arquitectura para la red neuronal recurrente simple. Fuente: Elaboración propia

Respecto con la ejecución sobre los datos de entrenamiento, testeo y validación, se observa en la Fig. 26, un descenso regular en las métricas de error para cada iteración, sin embargo, estas no alcanzan los valores obtenidos en los experimentos presentados previamente. Adicionalmente, no se observa

superposición entre las métricas consideradas para la validación y el entrenamiento, de hecho, es evidente como las métricas para el entrenamiento son superiores durante las 33 iteraciones que alcanza el algoritmo para su convergencia. Aunque no existe superposición de las métricas, no se configura una situación de *underfitting*, por lo que no se implementan capas adicionales de regularizadores o desconexiones neuronales.

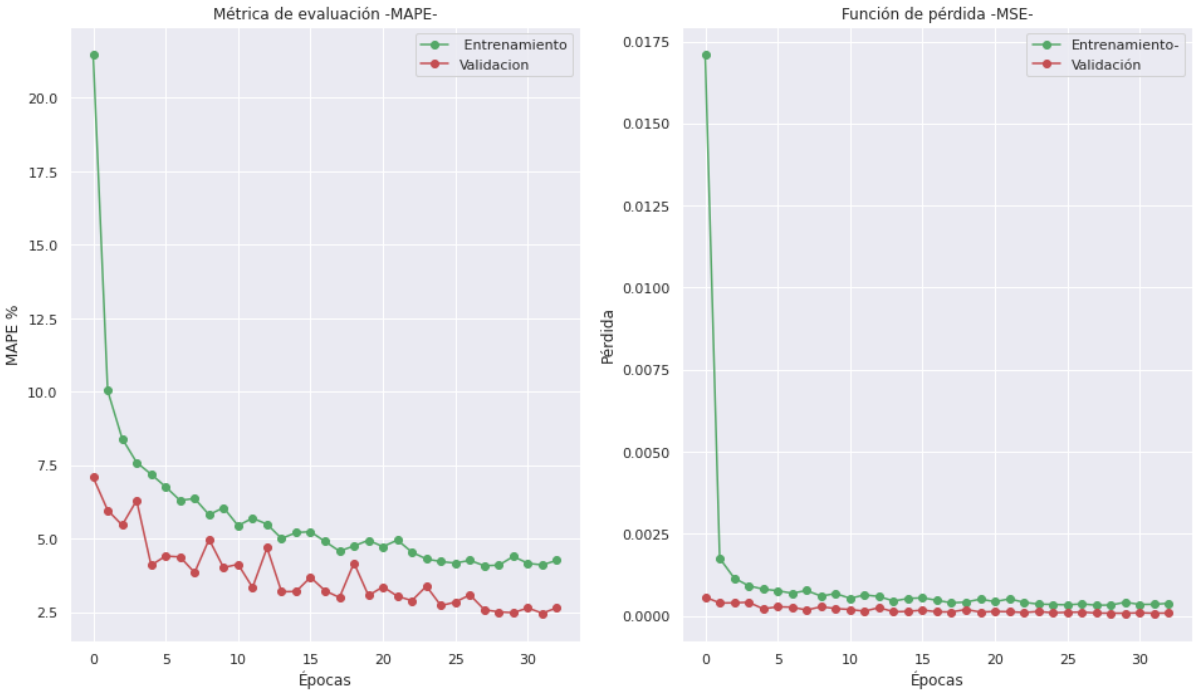

Figure 26: Métricas de evaluación del comportamiento de la arquitectura RNN simple. Fuente: Elaboración propia

Como se mencionó anteriormente, las iteraciones requeridas para la convergencia del algoritmo son 33 iteraciones para la ejecución; que significan para el tiempo de ejecución, 212 segundos aproximadamente para la finalización del ejercicio, situación que supone un mayor consumo del recurso computacional.

5.3.3 LONG SHORT TERM MEMORY - LSTM

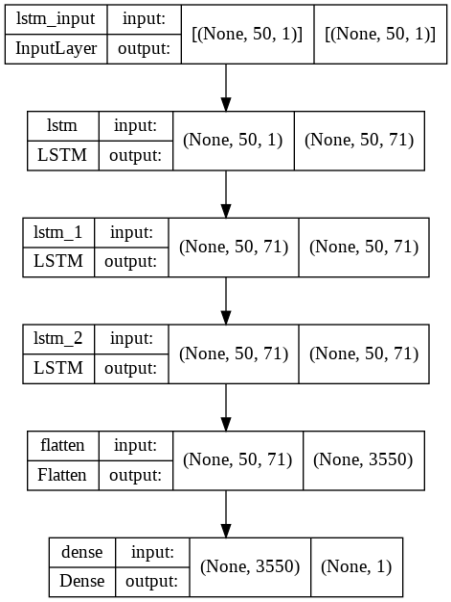

Figure 27: Arquitectura óptima para la red neuronal LSTM. Fuente: Elaboración propia

Como se observa en la Fig. 27, la arquitectura para las redes neuronales LSTM contiene 105507 parámetros que se disponen en cuatro capas densas, cada una de ellas contienen 71 neuronas, una capa *flatten* para reducir la dimensionalidad de la salida en la red neuronal y finalmente, una capa que contiene una única neurona que representa el pronóstico para el tiempo ***t+1.*** Para la etapa de predicción y bajo esta metodología, se realiza un procedimiento iterativo que genera las predicciones en los periodos ***t+1, t+2, t+3, …, t+n***.

La arquitectura presentada para LSTM, se ejecuta con una función de pérdida personalizada basada en el error medio cuadrático, que busca minimizar el resultado de la función y que está determinada por el valor calculado en el tiempo ***t*** frente al valor en el tiempo ***t-1***. Está situación es similar a la métrica de evaluación del modelo, destacando que, para este caso en particular, se hace uso del error absoluto medio porcentual y error medio cuadrático señalados en la eq. (1) y eq. (2).

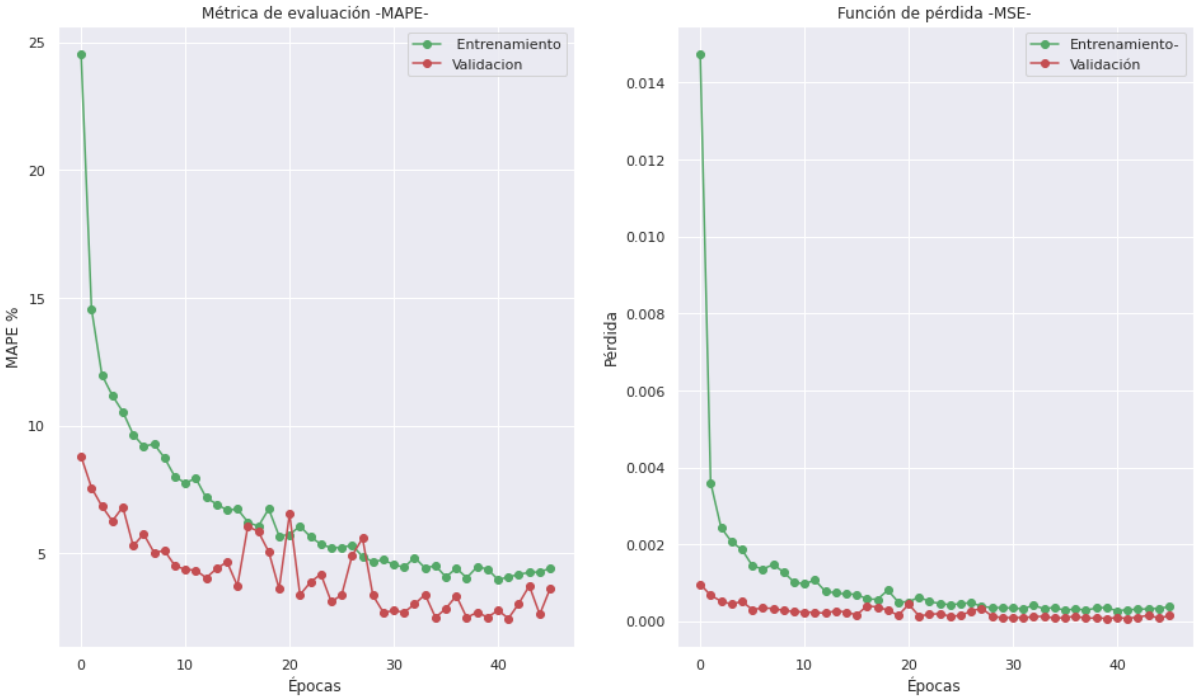

Figure 28: Métricas de evaluación del comportamiento de la arquitectura LSTM. Fuente: Elaboración propia

De acuerdo con la Fig. 28, se evidencia una rápida convergencia del algoritmo alrededor de las 31 iteraciones, tanto para los datos de entrenamiento como para los datos de validación Particularmente en la iteración 46, se evidencian saltos bruscos y anómalos al comportamiento regular entre la iteración 16 y 27, que no terminan por afectar significativamente la convergencia y minimización sobre la función de costo. Completar la ejecución no supera el minuto y por tanto, evidencia mejoras en relación con este ítem. Respecto a la función de pérdida, es evidente su rapidez para la convergencia, está requirió de tan solo 16 iteración para alcanzar valores cercanos a 0, lo anterior se debe en parte a la personalización realizada a la función de error MAE, explicada anteriormente. La configuración de la arquitectura y la rápida convergencia del algoritmo no están afectadas por sobre entrenamiento o desajuste del modelo, por lo que no fue necesario implementar elementos adicionales como regulizadores o desconexiones neuronales. Respecto a la función

5.3.4 GATED RECURRENT UNIT - GRU

Se replica el ejercicio para las redes neuronales GRU, en igual sentido, el objetivo principal es determinar la arquitectura óptima de la red, haciendo eficiente el recurso computacional y el tiempo de ejecución, por ello se evalúan los mismos parámetros analizados en el caso presentado previamente.

Se observa una red neuronal que se compone de 70169 parámetros distribuidos en cuatro capas densas, estas cuatro capas contienen 56 neuronas cada una y la primera de ellas recibe el vector de entrada. Luego se establece una capa *flatten* para disminuir la dimensionalidad del tensor y finalmente una capa densa con una neurona que recoge 2800 conexiones. Para el caso puntual, se estableció una tasa de aprendizaje de 0.001.

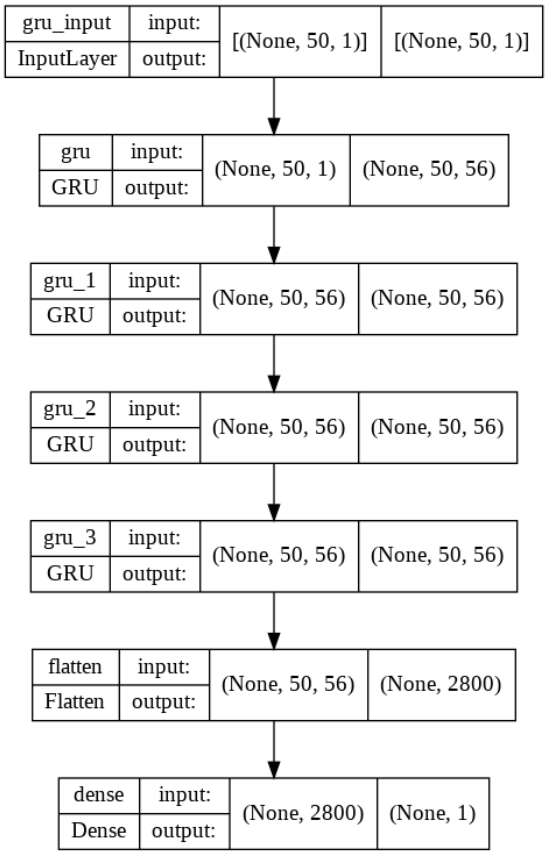

Figure 29: Arquitectura óptima para la red neuronal GRU. Fuente: Elaboración propia

Cabe destacar que, para la inclusión de capa *flatten* es necesario desactivar el retorno de la secuencia en la capa inmediatamente anterior, dado que, esto alteraría la dimensión del tensor resultante, teniendo para la predicción la condición de *secuence to secuence*, sobre lo que es necesario indicar que, la propuesta inicial para la predicción está acotada bajo la metodología *secuence to vector*.

Respecto a las condiciones de las métricas de evaluación, estas se mantienen iguales a las establecidas para las redes neuronales LSTM, esto con el fin de brindar condiciones de unanimidad al seguimiento y evaluación del desempeño.

El resultado del experimento, es similar al obtenido en el proceso presentado previamente, se observa en la Fig. 30, la convergencia del algoritmo en 26 iteraciones, que suponen un tiempo de ejecución de aproximadamente 37 segundos. El descenso regular para el subconjunto de datos de entrenamiento y validación, buscando la convergencia, presenta un pico particular en la iteración 9 que no termina por afectar las métricas evaluadas, de hecho, se observa sobre la última iteración un valor similar al obtenido con las redes neuronales LSTM. Los resultados en términos de métricas serán presentados en la siguiente sección.

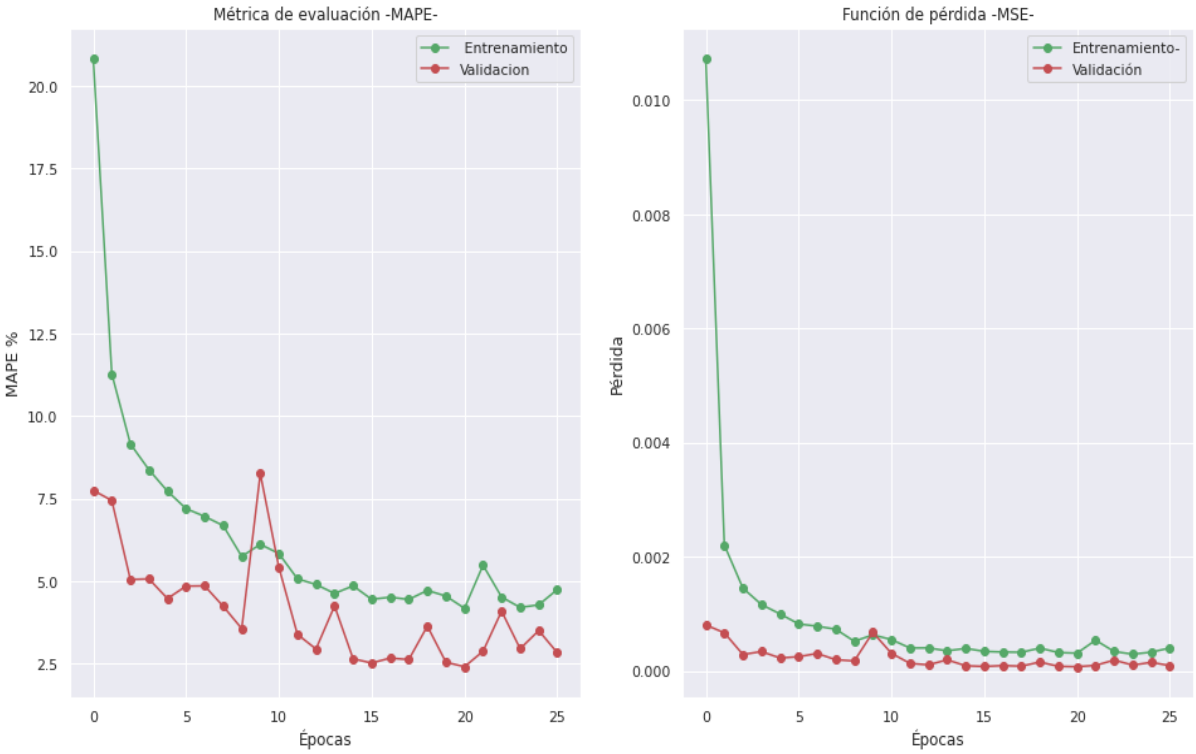

Figure 30: Métricas de evaluación del comportamiento de la arquitectura GRU. Fuente: Elaboración propia

5.4 HERRAMIENTAS

La principal herramienta utilizada en este proyecto es Google Collaboratory, que está basado en el proyecto de código abierto *Jupyter*. Al hacer uso de esta herramienta, se le proporciona al usuario una máquina virtual para la ejecución de código Python en la versión 3.7.13. Google Colab genera un ambiente de librerías preinstaladas en el espacio asignado, que el usuario solo debe importar bajo la sintaxis de Python. Google colab ofrece un servicio *PRO*, en el cual se eliminan restricciones relacionadas con la capacidad de cómputo que disminuyen sustancialmente los tiempos de ejecución.

Para la ejecución de los modelos ARIMA fueron supremamente importantes las librerías *Statsmodels* y *Pmdarima*, la primera de ella para la ejecución de las pruebas autocorrelación ACF y autocorrelación parcial PACF que, son las pruebas de hipótesis para determinar estacionariedad. La segunda librería es utilizada para automatizar el proceso de pruebas para la determinación de los parámetros óptimos para el mejor modelo ARIMA.

A su vez para la ejecución del modelo *XGBoost* es imperante la librería nativa del que lleva el mismo nombre, en igual importancia la implementación de la librería *GridSearch* que permite la automatización de ejecución de múltiples modelos con parámetros previamente definidos y al mismo tiempo permitir la validación cruzada. Luego de la definición del modelo en relación con sus parámetros la librería *Pipeline*, permite sintetizar todos los procesos por los cuales debe pasar los datos, desde el escalado hasta la obtención de resultados.

Respecto a las redes neuronales recurrentes, se hace uso de la API de Keras que hace parte de la librería de *tensorflow*, para la construcción y mejoramiento de la arquitectura de la red neuronal y de acuerdo con la definición del código fuente, se requiere de la versión mayor o igual a 2.0 para *tensorflow*. Acerca

del mejoramiento de la arquitectura, Google Colab no tiene precargado el paquete keras-tuner, por tanto, el usuario de la herramienta debe instalarlo e importarlo para su uso.

Ahora bien, luego de ser importada la API de Keras en el *notebook de* Google Colab, debe ser instanciada para la definición de las capas densas que a su vez comprenden las redes neuronales recurrentes simples, LSTM y GRU. Hay que indicar que, la API dispone de funciones para la implementación de las métricas de evaluación, que fueron personalizadas, tal como se expuso en secciones anteriores. Sobre las métricas, he de indicar que, en el proceso posterior al modelamiento de la serie de tiempo, se implementan métricas similares que están disponibles en la librería *sklearn*.

Respecto a la visualización en todas las etapas del procesamiento y en cada uno de los experimentos desarrollados, se implementó la librería *Matplotlib* y *Seaborn*.

6. RESULTADOS

6.1 MÉTRICAS

6.1.1 MODELOS ARIMA

En los modelos ARIMA se desarrollaron tres (3) modelos con diferentes configuraciones, los resultados obtenidos se muestran en la Fig 31, Allí se puede evidenciar que la primera ARIMA tiene el menor MSE pero su pronóstico tiende a ser una línea recta, en cambio la tercera ARIMA tiene un MSE muy similar pero su comportamiento trata de generar una estacionalidad muy marcada y que va creciendo con el tiempo.

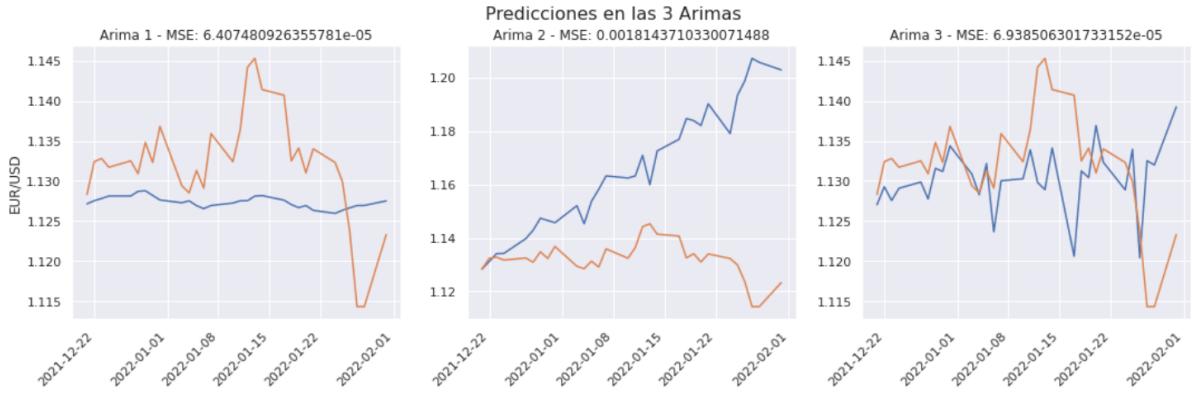

Figure 31: Resumen de predicción para los modelos ARIMA. Fuente: Elaboración propia

Por otro lado, en los modelos XGBoost se ejecutaron dos modelos, el primero construido con rezagos arbitrarios y el segundo con los rezagos de menor MSE y optimización de hiper parámetros. Los resultados se resumen en la Fig 32, allí se evidencia que la diferencia es poca y los errores semejantes.

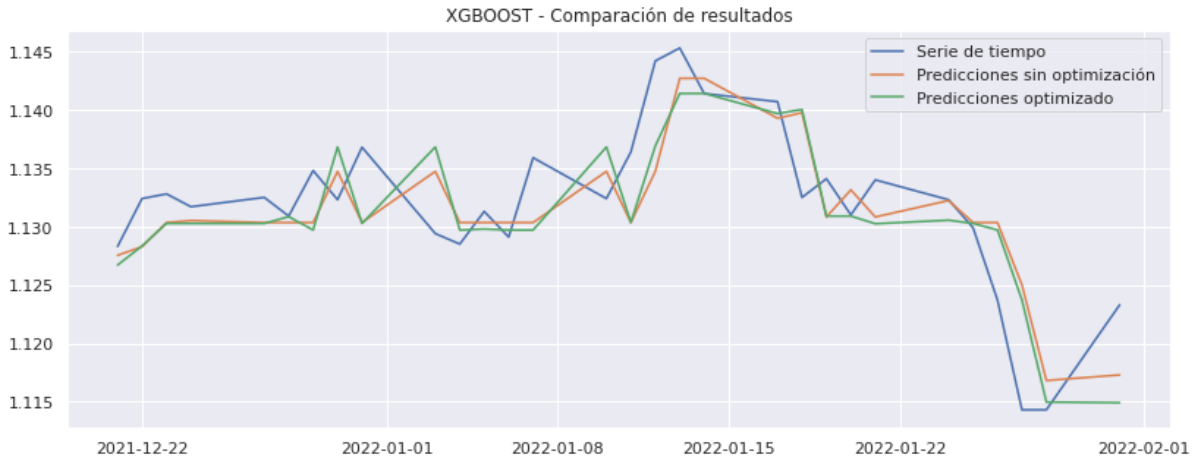

Figure 32: Comparación de resultados para XGBoost. Fuente: Elaboración propia

6.1.2 REDES NEURONALES RECURRENTES - RNN

Se abordan a continuación los resultados obtenidos a partir de la arquitectura propuesta en la sección 4.3.3. Las redes neuronales recurrentes, piedra angular para el análisis y modelamiento de procesos

estocásticos, de acuerdo con su estructura tienen una memoria cortoplacista que puede pasar por alto aspectos fundamentales y que impiden minimizar las métricas analizadas.

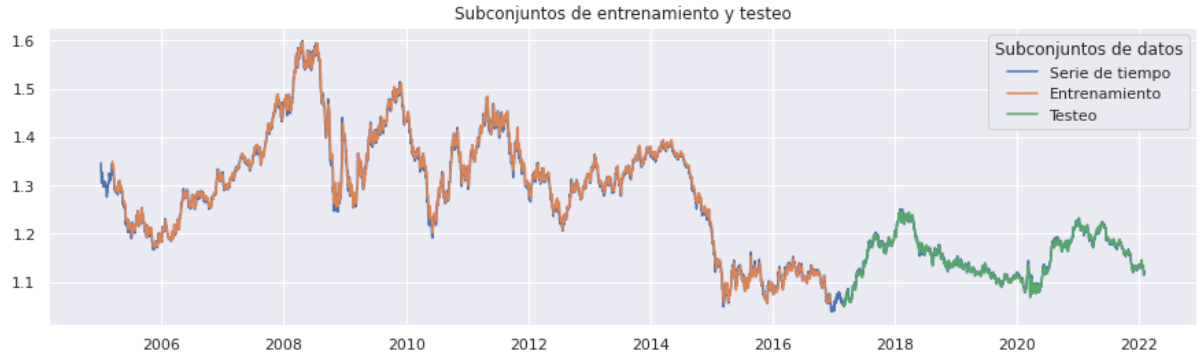

Figure 33: Pronóstico de la serie de tiempo para el entrenamiento y testeo. Fuente: Elaboración propia

Tanto en la etapa de entrenamiento como en la etapa de testeo, el pronóstico respectivo sigue la trayectoria de la serie de tiempo analizada. Desde la perspectiva presentada en la Fig. 33, el pronóstico recoge satisfactoria y parcialmente los cambios abruptos en los distintos periodos de la serie de tiempo. Esta predicción, no deja de ser imperfecta y se observan pequeños tramos, tanto en el entrenamiento como en el testeo, en los cuales no se superpone totalmente la predicción. Respecto a las métricas de evaluación, en la Table 2, estas no superan el 5% en el caso de MAPE para cualquiera de los subconjuntos evaluados.

DATOS - MSE	MAPE	MSE
Datos de entrenamiento	3.95%	0.0002
Datos de testeo	4.68%	0.0001
Datos de validación	2.52%	0.0002

Table 2: Métricas de evaluación para la arquitectura de la red neuronal recurrente. Fuente: Elaboración propia

Desde una escala mayor, se evidencia como la curva de pronóstico, en el caso del entrenamiento, es suave frente al proceso estocástico y persigue, de acuerdo con la Fig. 34, la tendencia a la baja o alza según corresponda. La prueba del modelo presentado sobre los datos de testeo también arroja una curva suave en comparación con los datos reales que de igual manera permite evidenciar y dar seguimiento a la tendencia, sin trazar los cambios más fuertes del proceso.

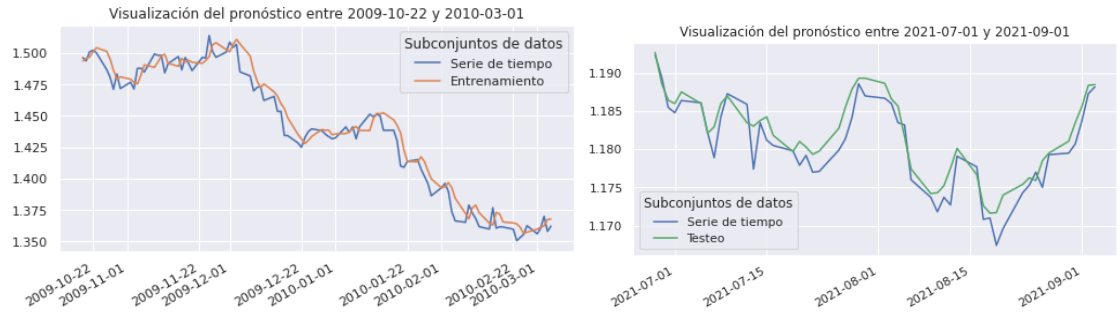

Figure 34: Izquierda: Pronóstico sobre los datos de entrenamiento. Derecha: Pronóstico sobre los datos de validación. Fuente: Elaboración propia

6.1.3 LONG SHORT TERM MEMORY - LSTM

La experimentación realizada y resumida en la Table 3, producto de la implementación de las redes neuronales LSTM, arroja resultados que no supera para ninguno de los subconjuntos de datos el 5%:

DATOS - MSE	MAPE	MSE
Datos de entrenamiento	3.65%	0.0002
Datos de testeo	4.57%	0.0001
Datos de validación	2.48%	0.0001

Table 3: Métricas de evaluación para la arquitectura de la red neuronal LSTM. Fuente: Elaboración propia

La comparación de los pronósticos para el entrenamiento y el testeo evidencia una aproximación generalizada a los datos reales, que no termina de recoger la totalidad de la información y por tanto no hay una superposición total de los pronósticos frente a la serie de tiempo analizada.

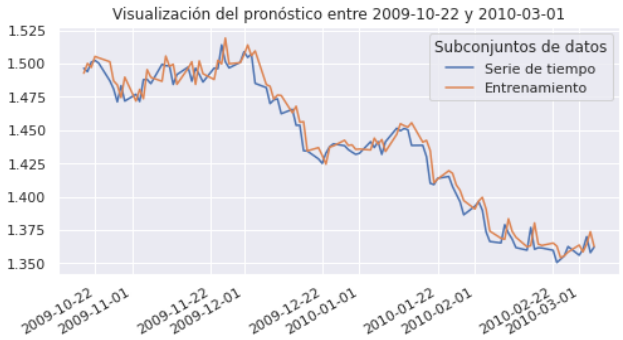

Figure 35: Izquierda: Pronóstico sobre los datos de entrenamiento. Derecha: Pronóstico sobre los datos de validación en LSTM. Fuente: Elaboración propia

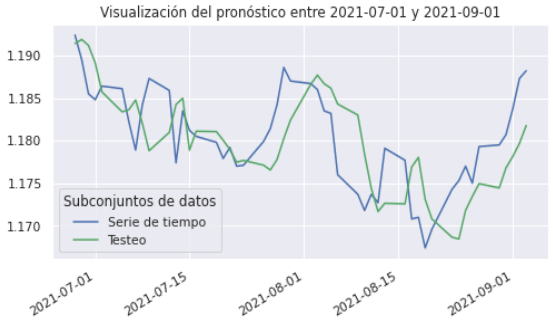

Figure 35: Izquierda: Pronóstico sobre los datos de entrenamiento. Derecha: Pronóstico sobre los datos de validación en LSTM. Fuente: Elaboración propia

Tal como se evidenció en el caso anterior, las curvas que indican el pronóstico son más suaves en contraste con el proceso estocástico, que por su condición presentan picos y rompen con la tendencia de periodos anteriores. Sin embargo, el pronóstico para el testeo, especialmente, muestra un comportamiento similar a los datos reales y en consecuencia hace seguimiento a la tendencia. Adicionalmente, se observa, un pequeño desplazamiento en ***t*** periodos hacía adelante, en el caso del pronóstico sobre los datos de testeo, del que se desconoce su causa.

6.1.4 GATED RECURRENT UNIT - GRU

Modelar a través de las redes neuronales implica generar tres subconjuntos a partir de los datos, tal como se expuso en la sección 3.2. Teniendo en cuenta este contexto, se evalúa a continuación los resultados obtenidos de acuerdo con la arquitectura presentada en la sección 4.3.3., en términos del MAPE y MSE.

DATOS - MSE	MAPE	MSE
Datos de entrenamiento	4.19%	0.0002
Datos de testeo	5.6%	0.0001
Datos de validación	2.67%	0.0002

Table 4: Métricas de evaluación sobre los subconjuntos de datos. Fuente: Elaboración propia

De acuerdo con la Table 4, los datos de testeo presentaron el mayor valor de MAPE frente a cifras similares obtenidas en el entrenamiento y testeo en contraste con las métricas obtenidas para las redes neuronales recurrentes y LSTM. No obstante, a partir de la Fig. 36, se evidencia que los datos pronosticados en las tres etapas siguen el comportamiento estocástico de la serie de tiempo original.

Figure 36: Pronóstico del entrenamiento y el testeo sobre la serie de tiempo. Fuente: Elaboración propia

Una visualización detallada de la serie de tiempo para los datos de entrenamiento y testeo, permite evidenciar las diferencias porcentuales expresadas en la Table 4. De acuerdo con la Fig. 37, el pronóstico se ajusta a la tendencia que tiene la serie de tiempo original, sin embargo, no se observa que el pronóstico del testeo recoja completamente todos los casos de picos altos o bajos del proceso estocástico, de hecho, se evidencia un aparente desplazamiento en ***t*** periodos hacía adelante, del que se desconoce la causa. Un análisis visual preliminar señala que el pronóstico encontrado no presenta cambios abruptos como los que se pueden evidenciar en la serie de tiempo original.

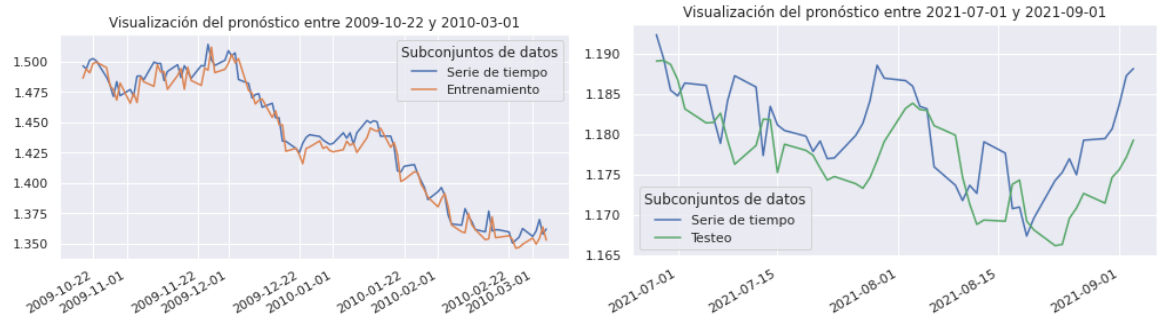

Figure 37: Comportamiento del pronóstico en el entrenamiento y testeo. Fuente: Elaboración propia

Se consolida el proceso en la de experimentación en la Table 5, en la que se describen las métricas de evaluación presentadas en las eq. (1) y eq. (2). Con base en esta información, se observa que el modelo ARIMA y XGBoost presentan métricas más pequeñas que las encontradas para los modelos de aprendizaje profundo implementadas para modelar las series de tiempo objeto de estudio.

Respecto a dichos modelos, hay que indicar que, las redes neuronales recurrentes LSTM presentan el error medio porcentual absoluto más pequeño, lo anterior, en un rango de error que no supera el 1.03%. En términos de las reglas del negocio, evidentemente métricas inferiores favorecen el uso de la arquitectura para ejecutar operaciones en el mercado FOREX.

	MODELO	MAPE	MSE
1	ARIMA	0.5206%	$6.9385 * 10^{-05}$
2	XGBOOST	0.3206%	$2.0806 * 10^{-05}$
3	GRU	5.6%	0.0001
4	LSTM	4.57%	0.0001
5	RNN	4.68%	0.0001

Table 5: Resumen de métricas de evaluación de los modelos implementados. Fuente: Elaboración propia

6.1.2 PREDICCIONES DE VALORES FUTUROS

Durante los experimentos presentados se obtuvieron pronósticos sobre los datos de test, presente la base de datos inicial, tal como se observó en la sección previa, los algoritmos implementados surtieron resultados favorables que seguían la trayectoria descrita por los datos reales. Ahora bien, se pretende a través de este ejercicio simular una circunstancia real de operación en el mercado FOREX, para poner a prueba los modelos descritos para la obtención de predicciones de los valores futuros. El ejercicio que se plantea es la predicción para 14 días a partir del último dato considerado en la base de datos que fue el 31 de enero del 2022.

De acuerdo con la Fig. 38, las predicciones de los modelos implementados a lo largo de este documento, no siguen la tendencia del comportamiento de la cotización del par de divisas euro - dólar, tampoco recogen los cambios que este proceso estocástico muestra para el periodo comprendido entre el 01-02-2022 y 17-02-2022. De hecho, las predicciones de los modelos RNN, GRU o XGBoost no presentan cambios significativos en el periodo mencionado.

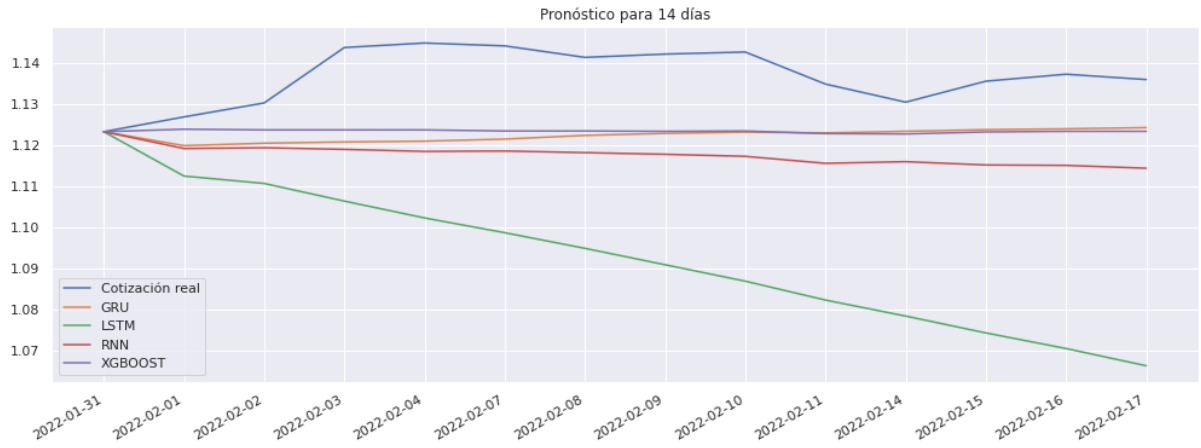

Figure 38: Pronósticos de los modelos frente a la cotización real. Fuente: Elaboración propia

Considerando un periodo de predicción inferior y en relación con los pronósticos obtenidos con base en el modelo de red neuronal LSTM, a partir de la Fig. 39 se observa un comportamiento que sigue la tendencia de la cotización real, sin embargo está se ve afectada por un comportamiento espejo.

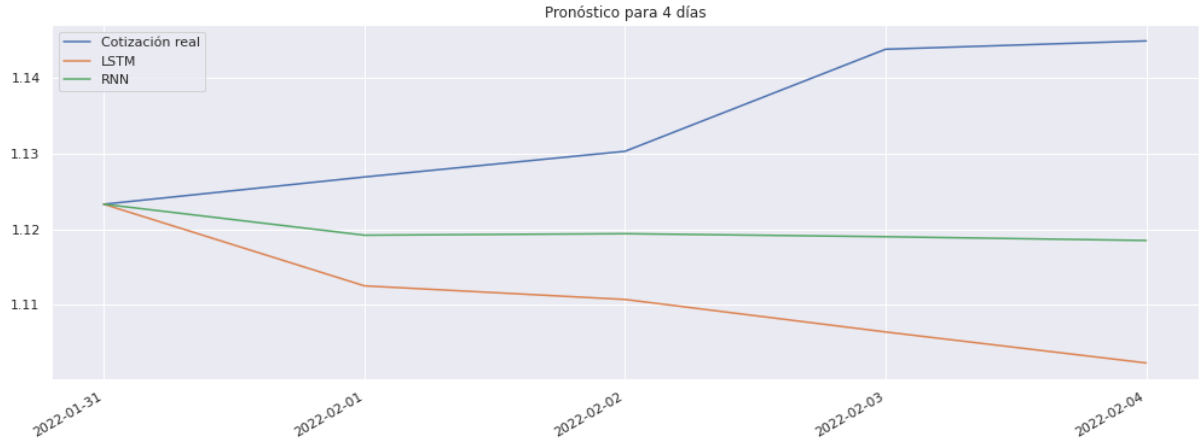

Figure 39: Pronóstico del modelo de red neuronal LSTM. Fuente: Elaboración propia

Suponga que el trazo que representa los pronósticos obtenidos a partir del modelo de red neuronal recurrente, esta opera como línea base de simetría, es decir si se proyectarán las diferencia entre el pronóstico de las redes neuronales recurrentes y LSTM, se obtendría un trazo que sigue la cotización real del par de divisas euro dólar, situación que, no deja de ser un supuesto y que obliga a enfatizar, analizar y proponer nuevas metodologías que verifiquen o descarten la situación planteada a través de la implementación de nuevas funciones y consideraciones para la predicción. Bajo este contexto, las predicciones realizadas solo son útiles hasta el tiempo ***t+3,*** posterior a este periodo los valores obtenidos se alejan de la cotización real.

En la Table 6 se muestran los errores MSE y MAPE calculados para las predicciones en 14 días. Desde el enfoque de las métricas propuestas es posible considerar que el modelo con las métricas más pequeñas es el elegido para utilizar como herramienta en operaciones que se desarrollen en el mercado FOREX. No obstante, desde el análisis gráfico expuesto anteriormente, esta situación no es válida.

	MODELO	**MAPE**	**MSE**
1	XGBoost	1.1845%	$2.15 * 10^{-04}$
2	GRU	1.2510%	$2.41 * 10^{-04}$
3	LSTM	4.4424%	$2.82 * 10^{-03}$
4	RNN	1.7639%	$4.33 * 10^{-04}$

Table 6: Resumen de error MSE y MAPE proyección 21 días

6.2 EVALUACIÓN CUALITATIVA

Se evidencia a partir de los resultados presentados anteriormente, que las métricas de error son inferiores al 5% considerando cada uno de los casos, sin embargo, las reglas del negocio exigen métricas inferiores que le permitan a un usuario del algoritmo realizar operaciones de trading seguras en el marco de las predicciones del algoritmo, se requiere al menos que el modelo describa con precisión la tendencia de la serie de tiempo. Aunque se observa que las métricas de error no superan el valor indicado previamente

de acuerdo con la Table 6, se concluye preliminarmente que el uso de esta herramienta tiene un alto riesgo para el desarrollo de operaciones de trading en el mercado

6.3 CONSIDERACIONES DE PRODUCCIÓN

Para una posible puesta en marcha en producción de un modelo de estos se debe considerar 3 principales etapas:

1. Extracción de datos en tiempo real: En los mercados financieros se debe contar siempre con información actualizada para tomar las decisiones, y como se evidenció en este documento, existe una alta correlación entre el valor en el tiempo ***t*** y periodos anteriores por lo cual es vital extraer la información de forma diaria y en constante actualización. Está información se debe extraer mediante la técnica de scraping o a través API's para la obtención de estos datos, sin embargo, será necesario contemplar un gasto económico adicional.
2. Ejecución de modelos: Aunque las arquitecturas de los modelos ya están entrenadas, se debe considerar el reentrenamiento constante y la posible actualización de dichas arquitecturas. Cabe resaltar que, se desconoce la periodicidad de estas actualizaciones
3. Conexión con plataformas de trading: El objetivo de los modelos predictivos es que a través de ellos se pueda generar riqueza en el mundo del trading. Lo ideal sería que, el modelo realizará predicciones cada cierto periodo de tiempo, y enviará una recomendación de compra o venta en el mercado financiero.

Los tres pasos anteriormente mencionados pueden ser establecidos o alojados en plataformas en la nube como AWS (Amazon Web Services) o GCP (Google Cloud Platform)

7. CONCLUSIONES

1. Por muchos años los modelos de ARIMA han sido utilizados para la predicción de series de tiempos, pero se puede apreciar que para un conjunto de datos con tendencias cambiantes, estacionalidades no muy marcadas y poca dependencia de valores futuros a sus valores pasados los resultados no son los mejores, puesto que métodos de machine learning como XGBoost obtiene mejores resultados en relación con la métricas de evaluación MSE y MAPE, incluso con su configuración por defecto.
2. Es altamente recomendable siempre optimizar los hiper parámetros de los modelos, pero para la serie de tiempo trabajada en este proyecto no se encontraron grandes diferencias en el modelo XGBoost con la configuración por defecto y la optimizada, por lo cual, si no te tiene buenos recursos computacionales o tiempo de ejecución, los hiper parámetros por defecto son una buena opción.
3. La implementación de redes neuronales recurrentes simples, GRU y el modelo XGBoost, no evidenciaron rendimientos aceptables en la etapa de validación con datos posteriores al periodo de tiempo analizado. Sus predicciones no demuestran la variabilidad de la serie de tiempo, a pesar de evidenciar métricas de evaluación inferiores al 2%
4. La arquitectura de las redes neuronales LSTM, aunque tienen métricas de error superiores a las obtenidas en los pronósticos de los demás modelos, estás evidencian un comportamiento similar a la serie de tiempo, sin embargo, las predicciones se encuentran reflejadas paralelamente al eje X. Esta situación podría deberse a la metodología implementada para la predicción que deberá ser analizada con detenimiento en trabajos posteriores.

Referencias

[1] B. f. I. Settlements, «Triennial Central Bank Survey - Foreign exchange turnover in April 2019,» 2019.

[2] A. Ariyo Adebiyi, A. Adewumi y C. Ayo, «Stock price prediction using the ARIMA model,» de *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, 2014.

[3] J. J. Espinosa-Zúñiga, «Aplicación de algoritmos Random Forest y XGBoost en una base de solicitudes de tarjetas de crédito,» *Ingeniería, investigación y tecnología,* vol. 21, nº 3, 2020.

[4] R. E. Barrientos Martínez, N. Cruz Ramírez, H. G. Acosta Mesa, I. Rabatte Suárez, M. d. C. Gogeascoechea Trejo, P. Pavón León y S. L. Blázquez Morales, «Árboles de decisión como herramienta en el diagnóstico médico,» 2009.

[5] J. H. Friedman, «Greedy function approximation: A gradient boosting machine.,» *The Annals of Statistics,* vol. 29, nº 5, pp. 1189-1232, 2001.

[6] L. Rodolfo, El Cerebro y el Mito del Yo, Norma, 2002.

[7] S. Krishna Kumar, «On weight initialization in deep neural networks,» *arXiv preprint arXiv:1704.08863,* 2017.

[8] A. Sherstinsky, «Fundamentals of Recurrent Neural Network (RNN),» *Physica D,* 2020.

[9] P. Agustín Granell, «Redes Neuronales Recurrentes: Una aplicación para los mercados bursátiles,» 2018.

[10] I. Bonet Cruz, S. Salazar Martinez, A. Rodriguez Abed, R. Grau Abalo y M. M. Garcia Lorenzo, «Redes neuronales recurrentes para el análisis de secuencias,» *REVISTA CUBANA DE CIENCIAS INFORMÁTICAS,* vol. 1, nº 4, pp. 48-57, 2007.

[11] A. Pajankar y A. Joshi , «Recurrent Neural Networks,» *Hands-on Machine Learning with Python,* pp. 285-305, 2022.

[12] H. Leung y S. Haykin, «The complex backpropagation algorithm,» *IEEE Transactions on Signal Processing,* vol. 39, nº 9, pp. 2101-2104, 1991.

[13] S. Ruder, «An overview of gradient descent optimization algorithms,» *arXiv: 1609.04747v2,* 2016.

[14] K. Diederik y B. Jimmy, «Adam: A Method for Stochastic Optimization,» *arXiv:1412.6980,* 2014.

[15] B. Lindemann, B. Maschler, N. Sahlab y M. Weyrich, «A survey on anomaly detection for technical systems using LSTM networks,» *Computers in Industry,* vol. 131, 2021.

[16] . T. Fischer y C. Krauss, «Deep learning with long short-term memory networks for financial market predictions,» *European Journal of Operational Research,* vol. 270, nº 2, pp. 654-669, 2018.

[17] K. Cho, B. v. Merriënboer, D. Bahdanau y Y. Bengio, «On the Properties of Neural Machine Translation: Encoder–Decoder Approaches,» *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation,* pp. 103-111, 2014.

[18] R. Fu, Z. Zhang y L. Li, «Using LSTM and GRU neural network methods for traffic flow prediction,» *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC),* 2016.

[19] P. Escudero, P. Paredes-Fierro y W. Alcocer, «Recurrent Neural Networks and ARIMA Models for Euro/Dollar Exchange Rate Forecasting,» *Applied Sciences,* 2021.

[20] A. García Figal y F. de la Oliva de Con, «Procedimiento para un pronóstico de la tasa de cambio euro-dólar,» *Economía UNAM,* vol. Vol 14, 2020.