MODELO DE ESTIMACIÓN DEL PRECIO INTRADIARIO DE LA BOLSA DE ENERGÍA ELÉCTRICA EN COLOMBIA MEDIANTE REDES LSTM

JUAN CARLOS ANAYA BOHÓRQUEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE CIENCIAS
ESCUELA DE MATEMATICAS
BUCARAMANGA
2025

MODELO DE ESTIMACIÓN DEL PRECIO INTRADIARIO DE LA BOLSA DE ENERGÍA ELÉCTRICA EN COLOMBIA MEDIANTE REDES LSTM

JUAN CARLOS ANAYA BOHÓRQUEZ

Trabajo de grado presentado como requisito parcial para optar al título de Matemático

Director
Andrés Sebastian Ríos Gutiérrez
Magíster en Estadística

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE CIENCIAS
ESCUELA DE MATEMATICAS
BUCARAMANGA
2025

CONTENIDO

pág.

LISTA DE FIGURAS

pág.

[1] Wang, Yanwen; Khodadadzadeh, Mahdi y Zurita-Milla, Raúl. «Spatial+: A new cross-validation method to evaluate geospatial machine learning models». En: *Int. J. Appl. Earth Obs. Geoinf.* 121 (2023), pág. 103364. ISSN: 1569-8432. DOI: 10.1016/j.jag.2023.103364. URL: https://www.sciencedirect.com/science/article/pii/S1569843223001887.

LISTA DE TABLAS

pág.

DEDICATORIA

A la memoria de mis padres,
quienes con su amor, ejemplo y sacrificio
sembraron en mí los valores que hoy me permiten alcanzar este logro.
Mi padre, trabajador incansable de la Universidad Industrial de Santander,
que con esfuerzo y orgullo dedicó su vida al servicio de esta institución.
Mi madre, ama de casa ejemplar, bondadosa y constante en su enseñanza.
Ellos no pudieron ver culminado este sueño,
pero siento que su presencia y bendición me acompañaron en cada paso.

A mi esposa Ligia,
por su paciencia, apoyo y amor incondicional.
A mis hijos, Juan Nicolás y Juan Diego,
por ser mi mayor fuente de motivación
y por comprender los momentos en que el tiempo familiar debió sacrificarse
para hacer posible este sueño.

A mis hermanas,
por su compañía, aliento y apoyo constante
en los días de esfuerzo y perseverancia.

AGRADECIMIENTOS

Este logro es el resultado de un camino largo, lleno de ilusiones, pausas, tropiezos y aprendizajes.

A la **Universidad Industrial de Santander**, por abrir nuevamente sus puertas y brindar las condiciones para culminar mi formación académica como *Matemático*. Esta institución no solo fue el escenario de mi crecimiento profesional, sino también un lugar de profundas memorias familiares que me llenan de orgullo.

A mi **director de tesis, profesor Andrés Sebastián Ríos Gutiérrez**, por su orientación, paciencia y compromiso durante este proceso. Su guía permitió que una idea se transformara en un proyecto real, y su ejemplo como docente y persona constituye una inspiración.

A mis **padres**, que desde la eternidad acompañaron cada paso de este esfuerzo. Su dedicación en vida y el amor con que forjaron mi educación han sido el motor que impulsó este sueño.

A mi **esposa Ligia** y a mis hijos **Juan Nicolás** y **Juan Diego**, por su apoyo incondicional, por su comprensión ante mis ausencias y por su fe constante en mis capacidades. Este logro es tanto mío como de ustedes.

A mis **hermanas**, por su respaldo, sus palabras de ánimo y por representar ese vínculo familiar que nunca se quiebra.

Y finalmente, a todos aquellos amigos, profesores y compañeros que, en distintos momentos, me brindaron su ayuda, consejo o simplemente su compañía, les expreso mi gratitud sincera.

Este título no es únicamente el resultado de mi esfuerzo, sino el reflejo de todas las personas que, directa o indirectamente, creyeron en mí.

RESUMEN

TÍTULO: MODELO DE ESTIMACIÓN DEL PRECIO INTRADIARIO DE LA BOLSA DE ENERGÍA ELÉCTRICA EN COLOMBIA MEDIANTE REDES LSTM

AUTOR: JUAN CARLOS ANAYA BOHÓRQUEZ

PALABRAS CLAVE: Predicción de precios, energía eléctrica, mercado eléctrico, series de tiempo, redes neuronales LSTM, aprendizaje profundo.

DESCRIPCIÓN:

El mercado energético colombiano atraviesa una etapa de transformación, caracterizada por la volatilidad de los precios y la integración creciente de energías renovables. En este contexto dinámico, anticipar con precisión el precio intradiario de la energía eléctrica es fundamental para que generadores, comercializadores y consumidores tomen decisiones informadas, gestionen riesgos y optimicen sus estrategias comerciales.

Este trabajo de grado desarrolla un modelo de predicción basado en redes neuronales LSTM (Long Short-Term Memory), una técnica de aprendizaje profundo especialmente adecuada para el análisis de series de tiempo en mercados eléctricos. Su idoneidad radica en la capacidad para capturar relaciones a largo plazo, manejar patrones no lineales y adaptarse a la alta volatilidad de los precios en los mercados energéticos, que dependen de múltiples variables históricas y exógenas como: consumo de combustible (MBTU), contratos de energía, variables temporales y cíclicas (hora, día, mes), etc.[2][3].

A partir de datos históricos y variables exógenas relevantes, se diseñaron modelos LSTM univariados y multivariados, los primeros se basan únicamente en la serie temporal del precio de la energía para hacer sus predicciones, es decir, toman como entrada una sola

[2]Hochreiter, Sepp y Schmidhuber, Jürgen. «Long short-term memory». En: *Neural Comput.* 9.8 (1997), págs. 1735-1780.

[3]Rumelhart, David E.; Hinton, Geoffrey E. y Williams, Ronald J. «Learning representations by back-propagating errors». En: *Nature* 323.6088 (1986), págs. 533-536.

variable y generan una única respuesta; mientras que los modelos multivariados incorporan, además del historial de precios, otras variables exógenas (consumo de combustible (MBTU), contratos de energía, variables temporales y cíclicas, etc.). El desempeño obtenido por el modelo multivariado se comparó con el método tradicional ARIMA, así mismo con redes neuronales no recurrentes. Los resultados encontrados evidencian que los modelos LSTM presentan no solo una mayor precisión en la predicción de precios, sino también una mayor adaptabilidad. En este contexto, la adaptabilidad se refiere a la capacidad del modelo para ajustar eficazmente sus pronósticos frente a cambios abruptos o a nuevas dinámicas del mercado eléctrico, manteniendo un buen desempeño incluso bajo condiciones no vistas previamente o ante fluctuaciones inesperadas en las variables que afectan el precio. Esto destaca el potencial de las LSTM para responder a la naturaleza dinámica y compleja de los mercados eléctricos actuales.

El presente trabajo proporciona una herramienta valiosa para la gestión de riesgos y la toma de decisiones en el sector eléctrico colombiano porque permite anticipar las fluctuaciones del mercado, posibilitando a las empresas generadoras, comercializadoras y demás actores del sector ajustar sus estrategias de compra y venta, optimizar contratos y diseñar esquemas de cobertura que minimicen pérdidas ante escenarios adversos.

ABSTRACT

TITLE: INTRADAY ELECTRICITY PRICE ESTIMATION MODEL FOR THE COLOMBIAN ELECTRICITY EXCHANGE USING LSTM NETWORKS

AUTHOR: JUAN CARLOS ANAYA BOHÓRQUEZ

KEYWORDS: Price forecasting, electric energy, electricity market, time series, LSTM neural networks, deep learning.

DESCRIPTION:

The Colombian energy market is undergoing a period of transformation, characterized by price volatility and the increasing integration of renewable energy sources. In this dynamic context, accurately forecasting the intraday price of electricity is essential for generators, marketers, and consumers to make informed decisions, manage risks, and optimize their commercial strategies.

This thesis develops a prediction model based on LSTM (Long Short-Term Memory) neural networks, a deep learning technique particularly well-suited for analyzing time series in electricity markets. Its suitability lies in its ability to capture long-term relationships, handle nonlinear patterns, and adapt to the high price volatility typical of energy markets, which depend on multiple historical and exogenous variables such as fuel consumption (MBTU), energy contracts, and temporal or cyclical variables (hour, day, month), among others[4][5].

Based on historical data and relevant exogenous variables, both univariate and multivariate LSTM models were designed. Univariate models rely solely on the electricity price time series for their predictions, that is, they take a single variable as input and generate a single output; while multivariate models incorporate, in addition to price history, other exogenous variables (fuel consumption (MBTU), energy contracts, tem-

[4]Hochreiter y Schmidhuber, «Long short-term memory», óp.cit.

[5]Rumelhart; Hinton y Williams, «Learning representations by back-propagating errors», óp.cit.

poral and cyclical variables, etc.). The performance achieved by the multivariate model was compared with the traditional ARIMA method as well as with non-recurrent neural networks. The results demonstrate that LSTM models exhibit not only greater accuracy in price prediction but also greater adaptability. In this context, adaptability refers to the model's ability to effectively adjust its forecasts in response to sudden changes or new dynamics in the electricity market, maintaining strong performance even under previously unseen conditions or unexpected fluctuations in price-determining variables. This highlights the potential of LSTM models to address the dynamic and complex nature of today's electricity markets.

This work provides a valuable tool for risk management and decision-making in the Colombian electricity sector by enabling market participants—generators, marketers, and other stakeholders—to anticipate market fluctuations, adjust their buying and selling strategies, optimize contracts, and design hedging schemes that minimize losses in adverse scenarios.

INTRODUCCIÓN

La predicción del precio intradiario de la energía eléctrica representa en la actualidad uno de los mayores retos del mercado energético colombiano, un entorno dinámico, en constante transformación y caracterizado por una alta volatilidad. En este contexto, la estadística, el aprendizaje de máquina y la economía ofrecen herramientas que permiten anticipar el comportamiento de los precios a partir de covariables, lo que facilita la toma de decisiones estratégicas y la planificación en el sistema eléctrico.

En Colombia, el precio de la energía eléctrica se fija en un mercado mayorista regulado por el Estado, en el que entidades como el Ministerio de Minas y Energía, la Comisión de Regulación de Energía y Gas (CREG) y XM S.A. E.S.P. cumplen funciones clave de dirección, regulación y operación del sistema[6][7]. No obstante, pese a este marco institucional, los precios fluctúan de forma significativa a lo largo del día, afectados por la variabilidad de la demanda, la oferta de generación hidráulica y térmica, y otros factores externos. Esta complejidad convierte la predicción de precios en un desafío permanente.

Este trabajo de grado propone el desarrollo de un modelo basado en redes neuronales LSTM (Long Short-Term Memory) para estimar el precio intradiario de la bolsa de energía eléctrica en Colombia. El objetivo es ofrecer una herramienta más precisa, capaz de superar el desempeño de enfoques tradicionales. Se espera que el modelo contribuya a mejorar la toma de decisiones de los agentes del mercado, optimizando estrategias de generadores, comercializadores y grandes consumidores[8].

[6] Congreso de Colombia. *Ley 143 de 1994: Régimen para la generación, interconexión, transmisión, distribución y comercialización de electricidad*. https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=4631. Accedido el 24 de julio de 2025. 1994.

[7] Comisión de Regulación de Energía y Gas (CREG). *Resolución CREG 024 de 1995: Aspectos comerciales del mercado mayorista de energía en el SIN*. https://gestornormativo.creg.gov.co/gestor/entorno/docs/resolucion_creg_0024_1995.htm. Accedido el 24 de julio de 2025. 1995.

[8] Chen, Y. y col. «Deep learning integration optimization of electric energy load forecasting and market price based on the ANN–LSTM–transformer method». En: *Front. Energy Res.* 11 (2023), pág. 1292204. DOI: 10.3389/fenrg.2023.1292204. URL: https://www.frontiersin.org/journals/energy-research/articles/10.3389/fenrg.2023.1292204/full.

La relevancia de esta investigación radica en su potencial para fortalecer la eficiencia y la competitividad del mercado eléctrico colombiano. Contar con mejores estimaciones de precios puede ayudar a optimizar estrategias de compra y venta y a gestionar riesgos, aspectos cruciales en el marco de la transición energética y la descarbonización. Además, desde la perspectiva académica, la aplicación de técnicas avanzadas de aprendizaje profundo, como las redes LSTM, abre oportunidades para abordar con mayor solvencia la predicción de series temporales en mercados energéticos complejos.

Con este fin, se analizarán la estructura y el comportamiento de las series temporales de precios, recopilando y preprocesando datos históricos tanto de precios como de variables externas relevantes. Posteriormente, se implementarán y compararán modelos LSTM univariados y multivariados, evaluando su desempeño frente a métodos como ARIMA y redes neuronales no recurrentes. El proceso comprende la limpieza y transformación de datos, su división en conjuntos de entrenamiento y de prueba y la evaluación de la capacidad predictiva de los modelos desarrollados.

Este trabajo de grado está organizado de la siguiente manera: el primer capítulo introduce y contextualiza el problema; el segundo y el tercer capítulo detallan el planteamiento del problema y presentan los objetivos; el cuarto desarrolla el marco teórico sobre series de tiempo, aprendizaje automático y redes LSTM; el quinto explica la metodología utilizada; el sexto expone los resultados; y, finalmente, los capítulos séptimo y octavo presentan las conclusiones y las recomendaciones para investigaciones futuras.

1. PLANTEAMIENTO DEL PROBLEMA

Mediante la Ley 142 de 1994[1], que regula los servicios públicos domiciliarios, y la Ley 143 de 1994[2], denominada Ley Eléctrica, se estableció que el Estado colombiano se encargará de la dirección, la regulación y la vigilancia del sector energético. La dirección la ejerce el Ministerio de Minas y Energía (MME) junto con la Unidad de Planeación Minero–Energética (UPME); la regulación se realiza a través del ministerio mencionado y de la Comisión de Regulación de Energía y Gas (CREG). Esta última, además de regular los sectores eléctrico y de gas, debe asegurar el crecimiento de dichos mercados y garantizar la implementación de un sistema de intercambio que permita realizar transacciones de compra y venta en el mercado mayorista. La vigilancia está a cargo de la Superintendencia de Servicios Públicos; en materia de competencia, el control corresponde a la Superintendencia de Industria y Comercio.

La operación y la administración del mercado energético mayorista en Colombia están a cargo de XM, entidad responsable del Sistema Interconectado Nacional y de la gestión de los registros de actores y activos tanto en mercados mayoristas como minoristas, en cumplimiento de la normatividad vigente[3][4]. El mercado colombiano comprende cinco actividades principales: generación, interconexión, transmisión, distribución y comercialización. Existe competencia en generación y en comercialización, mientras que las actividades de transporte constituyen un monopolio natural[5]. Las transacciones en bolsa se realizan bajo un modelo de libre competencia: los generadores presentan ofertas de energía y precio para el día siguiente; XM ordena dichas ofertas de menor a mayor para cubrir la demanda estimada y el valor marginal se fija con el último recurso seleccionado, con lo cual se determina el precio de bolsa[6]. En cuanto a la matriz de generación, el

[1] Congreso de Colombia. *Ley 142 de 1994: Régimen de los servicios públicos domiciliarios*. `https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=2752`. Accedido el 24 de julio de 2025. 1994.

[2] Íd., *Ley 143 de 1994: Régimen para la generación, interconexión, transmisión, distribución y comercialización de electricidad*, óp.cit.

[3] Íd., *Ley 142 de 1994: Régimen de los servicios públicos domiciliarios*, óp.cit.

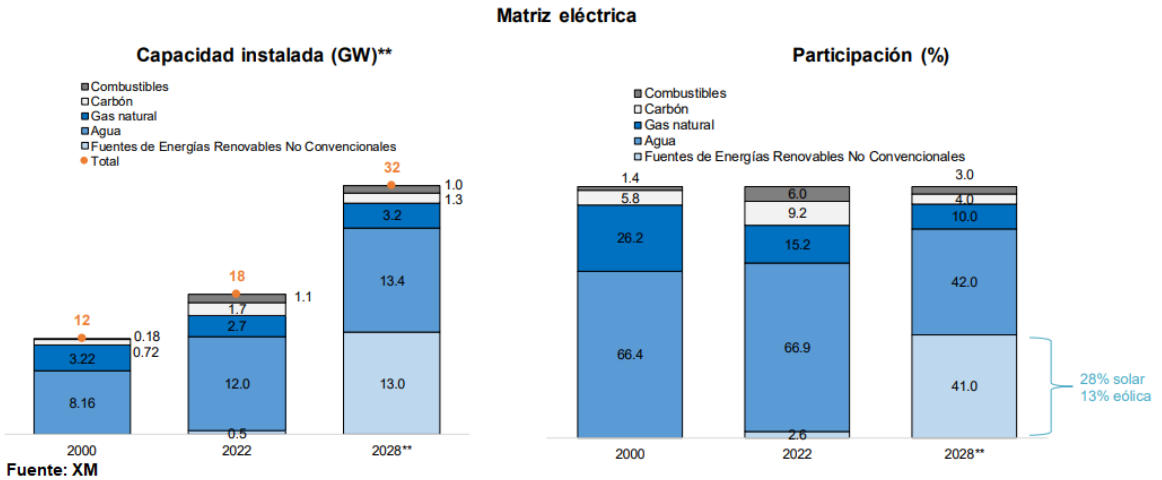
[4] Íd., *Ley 143 de 1994: Régimen para la generación, interconexión, transmisión, distribución y comercialización de electricidad*, óp.cit.

[5] Ibíd.

[6] Comisión de Regulación de Energía y Gas (CREG), *Resolución CREG 024 de 1995: Aspectos comerciales del mercado mayorista de energía en el SIN*, óp.cit.

66,9 % de la energía proviene de fuentes hidroeléctricas, el 30,4 % de plantas térmicas y el 2,7 % de fuentes solares, eólicas o cogeneradores, lo que refleja la diversificación de la oferta nacional[7].

Figura 1: La matriz eléctrica se concentra en aportes hídricos (66,9 %)



Fuente: XM[a]

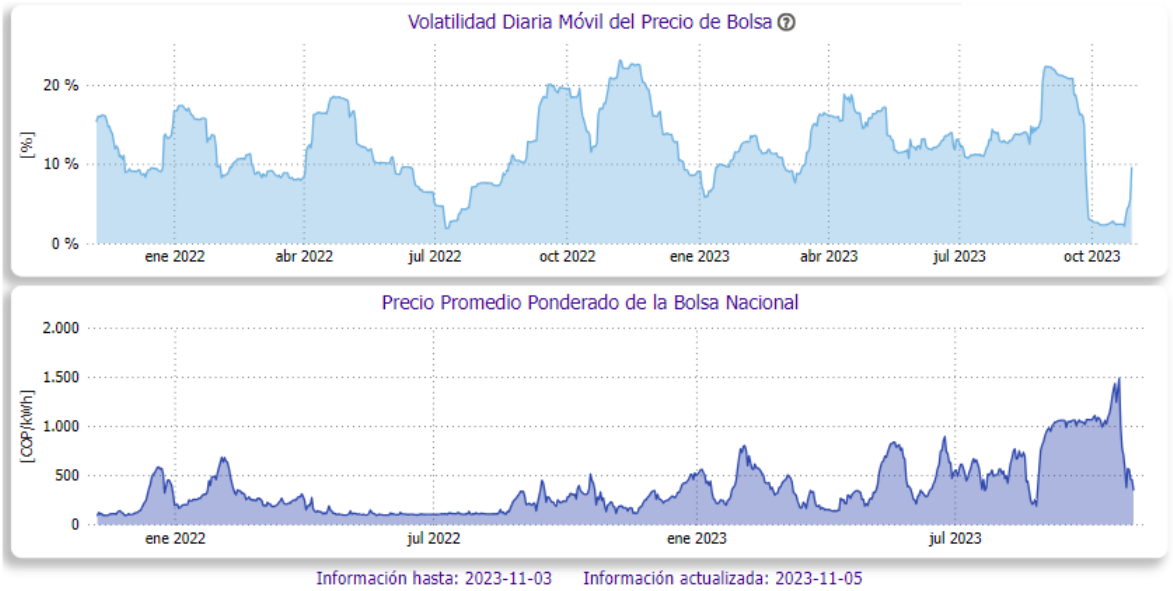
[a]Mejía, Luis Fernando. «Retos de la seguridad y transición energética». En: *Fedesarrollo, Centro de Investigación Económica y Social* (2023).

Los elementos anteriores permiten concluir que el mercado energético colombiano, en particular el precio *spot* de la energía eléctrica, depende en gran medida de las fuentes hídricas, cuya oferta aumenta o disminuye según la variabilidad climática. Además, dado que el precio *spot* es intradiario, se observan picos de consumo, especialmente en horas nocturnas, cuando se encienden las luces de los grandes centros poblados del país. Estos factores, junto con otras variables operativas y de mercado, contribuyen a que el precio *spot* en Colombia sea altamente volátil, como se aprecia en la Figura 2[8].

[7]Mejía, Luis Fernando. «Retos de la seguridad y transición energética». En: *Fedesarrollo, Centro de Investigación Económica y Social* (2023).

[8]Mejía, «Retos de la seguridad y transición energética», óp.cit.

Figura 2: Volatilidad del precio de bolsa



Fuente: XM[a]

[a]Mejía, Luis Fernando. «Retos de la seguridad y transición energética». En: *Fedesarrollo, Centro de Investigación Económica y Social* (2023).

La predicción precisa de los precios futuros de la electricidad resulta fundamental para optimizar decisiones operativas y estratégicas de todos los actores del mercado, pues permite a productores, distribuidores y grandes consumidores mejorar la rentabilidad de sus inversiones, implementar estrategias comerciales más efectivas y gestionar acuerdos bilaterales con mayor eficiencia. Sin embargo, la elevada volatilidad intradiaria, influida por factores climáticos, la variabilidad de la demanda y condiciones específicas del mercado, continúa representando un desafío, pese a los mecanismos regulatorios vigentes en Colombia. Los métodos estadísticos tradicionales, como ARIMA, muestran limitaciones para capturar patrones no lineales y dependencias de largo plazo presentes en las series de precios eléctricos; en consecuencia, surge la necesidad de modelos más avanzados. En este sentido, las redes neuronales LSTM (*Long Short-Term Memory*) ofrecen mayor capacidad para identificar dinámicas complejas y mejorar la previsibilidad del comportamiento intradiario del precio de la energía, por lo que se configuran

como una herramienta adecuada para atender las exigencias del sector[9].

A partir de lo expuesto, se plantea la siguiente pregunta de investigación: ¿De qué manera un modelo basado en redes neuronales LSTM puede mejorar la predicción del precio intradiario de la energía eléctrica en el mercado mayorista colombiano, considerando la alta volatilidad asociada a factores externos?

[9]Hyndman, Rob J. y Athanasopoulos, George. *Forecasting: Principles and Practice*. OTexts, 2018. URL: `https://otexts.com/fpp2/`.

2. OBJETIVOS

2.1. OBJETIVO GENERAL

Desarrollar un modelo para la predicción del precio intradiario en la bolsa de energía eléctrica en Colombia utilizando redes neuronales LSTM (Long Short-Term Memory).

2.2. OBJETIVOS ESPECÍFICOS

1. Realizar un análisis descriptivo de los factores que influyen en la formación de los precios intradiarios en la bolsa de energía eléctrica en Colombia.
2. Recopilar datos históricos relevantes sobre la bolsa de energía eléctrica en Colombia, incluyendo información detallada sobre precios, volúmenes de transacción y variables externas que puedan afectar los precios.
3. Implementar una red neuronal LSTM para la estimación del precio intradiario de la energía eléctrica en Colombia.
4. Evaluar el desempeño del modelo propuesto mediante métricas de precisión y compararlo con otros enfoques estadísticos y de aprendizaje automático.
5. Proponer recomendaciones basadas en los resultados obtenidos para mejorar la toma de decisiones en el mercado eléctrico colombiano.

3. MARCO TEÓRICO - FUNDAMENTOS DE REDES NEURONALES PARA SERIES DE TIEMPO

3.1. SERIES DE TIEMPO

Las series de tiempo son un componente fundamental en una variedad de disciplinas, desempeñando un papel crucial en la comprensión, modelado y predicción de fenómenos que varían a lo largo del tiempo. Su aplicación se extiende a campos como la economía, finanzas, meteorología, entre otros. En el campo de la economía y las finanzas, las series de tiempo son fundamentales para el análisis y pronóstico de variables económicas clave, como el producto interno bruto (PIB), la inflación, el desempleo y las tasas de interés. Por ejemplo[1], destaca cómo las series de tiempo son utilizadas en modelos macroeconómicos para entender la dinámica de la actividad económica y evaluar el impacto de políticas fiscales y monetarias, por otra parte, en las finanzas, las series de tiempo son esenciales para modelar la evolución de los precios de los activos financieros, tales como acciones, bonos, divisas y en particular commodities. Los modelos de series temporales son ampliamente utilizados en la predicción de precios y en la gestión de riesgos en el mercado financiero[2].

En el año de 1976 se definieron las series de tiempo como una secuencia de observaciones de una variable, tomadas en intervalos de tiempo sucesivos o discretos[3]. En el 2016 fue definida como: Una serie de tiempo es una secuencia de observaciones, usualmente ordenadas en el tiempo equiespaciado[4]. Por lo anterior podemos decir que las series de tiempo son conjuntos de datos secuenciales recolectados a lo largo del tiempo.

Las series de tiempo pueden considerarse de baja frecuencia si los datos están registrados en intervalos largos de tiempo (mensual, trimestral, anual, etc) son útiles para identificar tendencias y patrones de largo plazo en los datos, debido a la naturaleza de los intervalos

[1] Hamilton, James D. *Time Series Analysis*. Princeton University Press, 1994.
[2] Tsay, Ruey S. *Analysis of Financial Time Series*. Wiley, 2010.
[3] **box1976time**.
[4] Brockwell, Peter J. y Davis, Richard A. *Introduction to Time Series and Forecasting*. 3rd. Springer, 2016.

de tiempo más amplios, las series de baja frecuencia tienden a exhibir bajas volatilidades y pueden carecer del detalle temporal necesario para capturar fluctuaciones de corto plazo en los datos. En la Figura 3, se muestra la gráfica de una serie de tiempo de baja frecuencia con los datos anuales de la tasa de crecimiento porcentual anual del PIB en Colombia durante los últimos 20 años expresados en dólares, los datos fueron tomados del Banco Mundial[5].

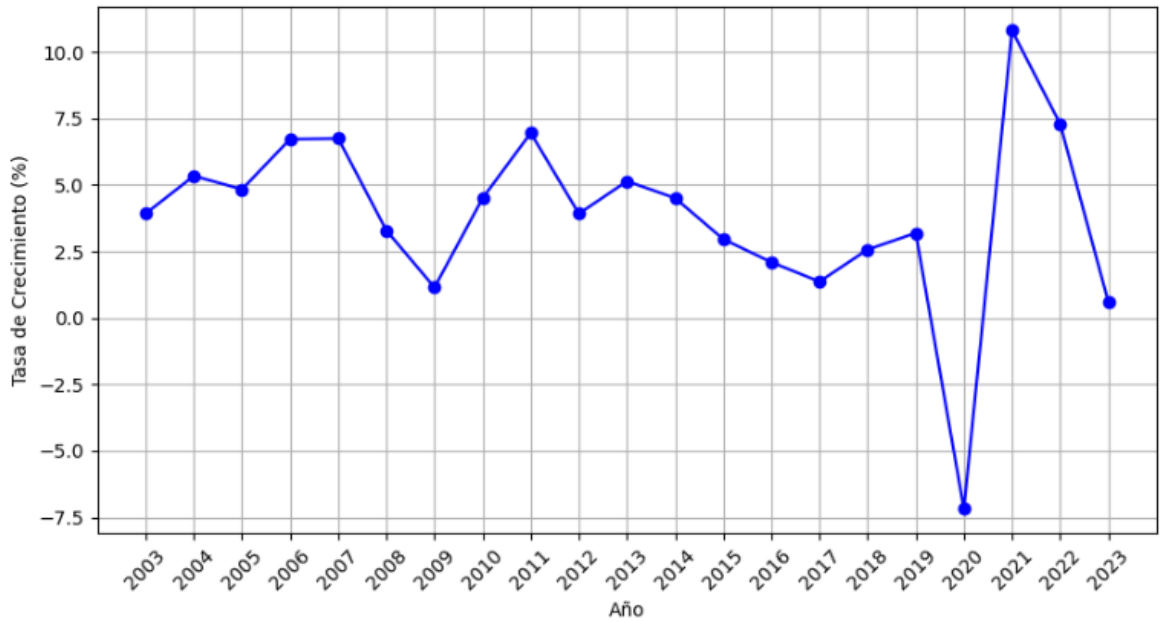


Figura 3: Tasa de crecimiento porcentual anual del PIB a precios de mercado en moneda local constante de Colombia desde el 2003 hasta el 2023

Por otra parte las principales características de una serie de alta frecuencia (diarios o intradiarios) son: poseer intervalos de tiempo cortos, lo que conlleva a contener una gran cantidad de datos en un período de tiempo dado, esto puede resultar en conjuntos de datos grandes y detallados que contienen información sobre la dinámica del proceso en una escala temporal fina. Mayor volatilidad como resultado de fluctuaciones de corto plazo que pueden ser más pronunciadas y reflejar cambios rápidos en las condiciones del mercado o del proceso en estudio. Las series de alta frecuencia son útiles para el

[5]Banco Mundial. *Tasa de crecimiento porcentual anual del PIB en Colombia (constante)*. Accedido el 13 de febrero de 2025. 2023. URL: `https://datos.bancomundial.org/indicador/NY.GDP.MKTP.KD.ZG?end=2023&locations=CO&start=2003`.

análisis de corto plazo y la identificación de patrones y tendencias a corto plazo en los datos. Esto puede ser especialmente relevante en áreas como las finanzas, donde se monitorean los precios de los activos en tiempo real. Por último se puede detectar en este tipo de series fluctuaciones aleatorias en los datos que ocurren a una escala temporal muy pequeña, estas fluctuaciones son rápidas y de corta duración, lo que significa que se producen en intervalos de tiempo muy cortos, como segundos, minutos o incluso fracciones de segundo, dependiendo de la frecuencia de muestreo de los datos, a esto se denomina ruido de alta frecuencia, lo que puede dificultar la identificación de señales importantes en los datos. El filtrado y la suavización de los datos pueden ser necesarios para extraer información relevante.

En la Figura 4, se puede apreciar una serie de alta frecuencia, determinada por los precios de escasez y precio máximo de Bolsa (pesos/kWh) diarios máximos del 01 de enero del 2024 hasta el 04 de enero del 2025 de la Bolsa de energía colombiana, en ella se aprecia la alta volatilidad de las serie de tiempo al igual que las demás características propias de este tipo de series nombradas anteriormente.

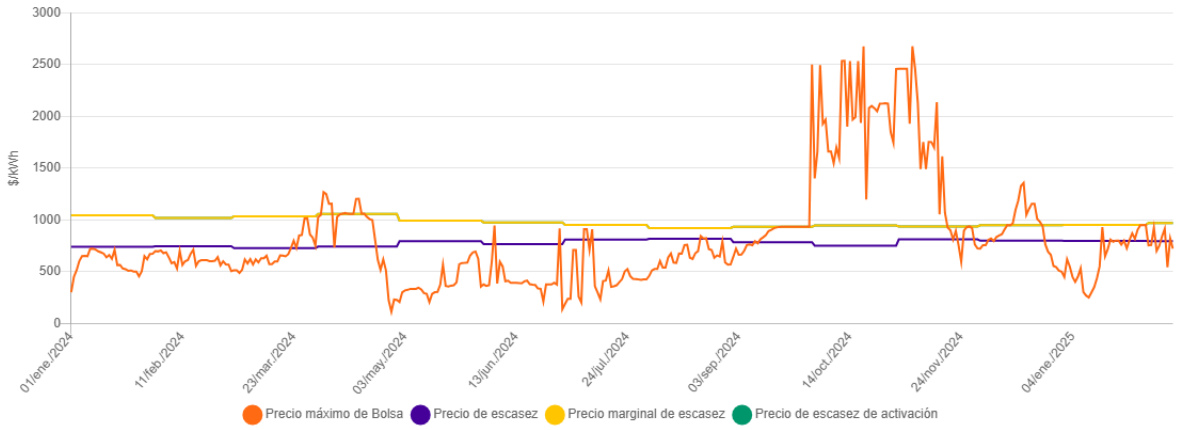


Figura 4: Precio de bolsa y precio de escasez en Colombia

Fuente: XM (2025)[6]

Los modelos de series de tiempo son herramientas fundamentales en el análisis y la predicción de datos secuenciales. Una clasificación importante de estos modelos se basa en la distinción entre modelos deterministas y estocásticos. Según Box y Jenkins[7].

[7] **box1976time**.

3.1.1. Modelos deterministas

Los modelos deterministas asumen que la serie temporal sigue un patrón predecible y regular sin ningún componente aleatorio. Estos modelos están basados en funciones matemáticas o reglas conocidas y no incluyen términos de error estocástico. Usualmente este término se modela como una distribución normal de parámetros media igual a cero y varianza igual a σ^2 en caso de que el modelo sea homocedástico.

En un modelo de regresión, se utilizan variables explicativas para predecir el valor de la serie temporal. La ecuación típica de un modelo de regresión puede expresarse como:

$$y_t = \beta_0 + \beta_1 X_{1t} + \beta_2 X_{2t} + \ldots + \beta_k X_{kt} + \varepsilon_t \tag{3.1}$$

donde $X_{1t}, X_{2t}, \ldots, X_{kt}$ son variables aleatorias explicativas en el tiempo t, $\beta_0, \beta_1, \beta_2, \ldots, \beta_k$ son los coeficientes respectivos de cada variable explicativa y ε_t es el término de error.

El modelo de estacionalidad es una forma de modelar patrones repetitivos y predecibles en una serie de tiempo. En este modelo, la estacionalidad se considera como una función determinista de tiempo. Matemáticamente, el modelo de estacionalidad determinista se puede expresar de la siguiente manera:

$$y_t = f(t) + \epsilon, \quad \epsilon \sim \text{Normal}(0, \sigma^2) \tag{3.2}$$

donde y_t es el componente de estacionalidad en el tiempo t y $f(t)$ es una función determinista que describe la variación estacional en función del tiempo t.

La función $f(t)$ puede tomar diferentes formas dependiendo de la naturaleza de la estacionalidad en los datos. Algunas formas comunes de funciones deterministas incluyen:

Funciones trigonométricas: Tales como senos y cosenos, que son útiles para modelar

patrones estacionales que siguen una forma sinusoidal.

$$f(t) = \beta_0 + \beta_1 \sin(2\pi\omega t + \phi) \tag{3.3}$$

donde β_0 y β_1 son coeficientes que determinan la amplitud y la fase de la estacionalidad, y ω es la frecuencia estacional.

Funciones polinomiales: Tales como polinomios de bajo grado (lineales, cuadráticos, etc.), que pueden ser útiles para modelar patrones estacionales que no siguen una forma sinusoidal.

$$f(t) = \beta_0 + \beta_1 t + \beta_2 t^2 + \ldots + \beta_n t^n + \epsilon, \quad \epsilon \sim \text{Normal}(0, \sigma^2) \tag{3.4}$$

donde $\beta_0, \beta_1, \ldots, \beta_n$ son coeficientes que determinan la forma de la función polinomial.

Además existen funciones de otros tipos, dependiendo del tipo de estacionalidad presente en los datos, también se pueden utilizar otras funciones deterministas para modelarla, como funciones exponenciales, logarítmicas, etc.

Es importante destacar que en el modelo de estacionalidad determinista, la estacionalidad se asume como completamente determinística y no se considera ningún componente de error aleatorio. Esto puede ser adecuado en casos donde la estacionalidad es muy regular y predecible. Sin embargo, en muchos casos, es posible que también sea necesario considerar componentes estacionales estocásticos para capturar la variabilidad aleatoria en la estacionalidad[8].

Definición 3.1.1 (Ruido blanco)**.** El ruido blanco es un proceso estocástico con variables aleatorias de media cero, varianza constante y covarianzas nulas. Es decir, es a_t, $t = 0, \pm 1, \pm 2, \ldots$.

$$E(a_t) = 0, \quad \forall t$$

[8]Brockwell y Davis, *Introduction to Time Series and Forecasting*, óp.cit.

$$V(a_t) = \sigma^2, \quad \forall t$$

$$\text{Cov}(a_t, a_s) = 0, \quad \forall t \neq s$$

con $t = 0, 1, \ldots$

Así, un proceso de ruido blanco, $a_t \sim RB(0, \sigma^2)$, es estacionario si la varianza σ^2 es finita con función de autocovarianzas (FACV):

$$\gamma_k = \sigma^2, \quad k = 0 \qquad \gamma_k = 0, \quad k > 0$$

y función de autocorrelación (FAC):

$$\rho_k = 1, \quad k = 0 \qquad \rho_k = 0, \quad k > 0$$

Para una introducción a los preliminares de la teoría de probabilidad, como definición de probabilidad, variable aleatoria, proceso estocástico y estacionariedad, el lector puede remitirse a Modelización del covid-19 en Santander mediante series temporales. Tesis de pregrado[9].

3.1.2. Modelos estocásticos

3.1.2.1. Modelo de media móvil (MA)

En un modelo de media móvil, se supone que la serie temporal es una combinación lineal de errores aleatorios en períodos de tiempo anteriores. La ecuación típica de un modelo MA puede expresarse como:

$$y_t = c + \varepsilon_t + \beta_1 \varepsilon_{t-1} + \beta_2 \varepsilon_{t-2} + \ldots + \beta_q \varepsilon_{t-q} \tag{3.5}$$

[9]Díaz Garcés, Cristian Julián. *Modelización del covid-19 en Santander mediante series temporales.* Tesis de pregrado, Universidad Industrial de Santander. 2023. URL: `https://noesis.uis.edu.co/handle/20.500.14071/14839`.

donde y_t es la variable de interés en el tiempo t, c es una constante, $\varepsilon_t \sim RB(0, \sigma^2)$ es el término de error en el tiempo t, y $\beta_1, \beta_2, \ldots, \beta_q$ son los coeficientes de los errores pasados.

3.1.2.2. Modelo Autorregresivo ((AR)

En un modelo autorregresivo, se supone que la variable dependiente en el tiempo actual es una función de sus valores anteriores, junto con un término de error estocástico. La ecuación típica de un modelo AR puede expresarse como:

$$y_t = \beta_0 + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \ldots + \beta_p y_{t-p} + \varepsilon_t \tag{3.6}$$

donde y_t es la variable de interés en el tiempo t, $\beta_0, \beta_1, \ldots, \beta_p$ son los coeficientes autorregresivos y $\varepsilon_t \sim RB(0, \sigma^2)$ es el término de error.

3.1.2.3. Modelo (ARIMA) (Autorregresivo Integrado de Media móvil)

El modelo ARIMA combina componentes autorregresivos (AR) y de medias móviles (MA), junto con una etapa de diferenciación, con el objetivo de hacer estacionaria la serie temporal. La ecuación típica del modelo ARIMA se expresa como:

$$\Phi(B)(1 - B)^d y_t = \Theta(B)\varepsilon_t \tag{3.7}$$

En esta formulación, los polinomios $\Phi(B)$ y $\Theta(B)$ se construyen sobre el operador de rezagos B, mientras que el parámetro d representa el grado de diferenciación aplicado a la serie y ε_t indica el término de error. Para evitar ambigüedades, conviene señalar que el operador de rezagos puede escribirse como $B^s X_t = X_{t-s}$, lo cual facilita la

representación de valores anteriores dentro de la serie temporal[10].

Los procesos autorregresivos y de medias móviles, cuando se emplean en su versión clásica, suelen estar asociados a series estacionarias, es decir, aquellas cuyas características estadísticas, como la media y la varianza, permanecen constantes a lo largo del tiempo[11]. En la práctica, esto significa que el comportamiento de la serie es estable y no presenta tendencias persistentes ni cambios en la dispersión. Por el contrario, el modelo ARIMA se utiliza precisamente en contextos donde se detecta no estacionaridad, permitiendo que, mediante la diferenciación, se ajusten los datos para analizar únicamente componentes estacionarios.

Por ello, antes de aplicar un modelo ARIMA, se recomienda revisar si la serie evidencia una tendencia sistemática o patrones de no estacionaridad. De no ser así, podrían ser más apropiados los modelos AR o MA. La selección del modelo debe justificarse en función del comportamiento observado y del objetivo del análisis de series temporales[12].

3.1.3. Componentes de una serie de tiempo

Las componentes de una serie de tiempo sirven fundamental para analizar y modelar correctamente el comportamiento de los datos a lo largo del tiempo[13]. Se dice que una serie de tiempo puede descomponerse en cuatro componentes (cinco si se considera una constante llamada nivel) que no son directamente observables, de los cuales únicamente se pueden obtener estimaciones. Estas son las cuatro componentes[14]:

- **Tendencia:** La tendencia se refiere a la dirección general en la que se mueven los

[10]Brockwell y Davis, *Introduction to Time Series and Forecasting*, óp.cit.

[11]Box, George E. P. y col. *Time Series Analysis: Forecasting and Control*. 5th. John Wiley & Sons, 2015.

[12]Shumway, Robert H. y Stoffer, David S. *Time Series Analysis and Its Applications: With R Examples*. 4th. Springer, 2017.

[13]Hyndman y Athanasopoulos, *Forecasting: Principles and Practice*, óp.cit.

[14]Ríos, Gonzalo y Hurtado, Carlos. «Series de tiempo». En: *Universidad de Chile. Facultad de Ciencias Físicas y Matemáticas* 52 (2008).

datos a lo largo del tiempo. Puede ser ascendente, descendente o seguir un patrón más complejo. La tendencia puede ser determinística o estocástica, dependiendo de si sigue un patrón predecible o es aleatoria[15].

- **Estacionalidad:** La estacionalidad se refiere a patrones recurrentes o cíclicos en los datos que se repiten a intervalos regulares. Por ejemplo, las ventas minoristas suelen mostrar estacionalidad debido a los picos durante las temporadas de vacaciones. La estacionalidad puede tener una duración fija o variable y puede ser aditiva o multiplicativa[16].

- **Ciclos:** Los ciclos representan fluctuaciones periódicas que no están necesariamente relacionadas con estacionalidad. A menudo tienen una duración más larga que los patrones estacionales y pueden ser el resultado de factores económicos, sociales o políticos. Los ciclos pueden ser difíciles de identificar y modelar, ya que su naturaleza es más compleja que la estacionalidad.

- **Componente Irregular:** El componente irregular, también conocido como componente aleatorio o residual, captura las variaciones aleatorias o no sistemáticas en los datos que no pueden ser explicadas por la tendencia, la estacionalidad o los ciclos. Este componente puede deberse a factores externos impredecibles o errores en la medición.

Un modelo clásico de series de tiempo, supone que la serie $y_1, ..., y_t$ puede ser expresada como suma o producto de sus componentes[17].

3.1.3.1. Descomposición en Modelos Aditivos

La descomposición en modelos aditivos para series de tiempo es una técnica que divide una serie en cuatro componentes principales:

[15]Brockwell y Davis, *Introduction to Time Series and Forecasting*, óp.cit.
[16]Shumway y Stoffer, *Time Series Analysis and Its Applications: With R Examples*, óp.cit.
[17]Ríos y Hurtado, «Series de tiempo», óp.cit.

- **Tendencia** (T_t): Captura la dirección general de la serie temporal a largo plazo.
- **Estacionalidad** (S_t): Representa los patrones repetitivos y predecibles en intervalos fijos de tiempo.
- **Ciclo** (C_t): Muestra fluctuaciones más largas que la estacionalidad, pero sin una periodicidad fija.
- **Componente irregular** (E_t): Contiene las variaciones aleatorias, errores de medición y variabilidad no explicada por los otros componentes. Este componente cumple el papel de término de error y suele modelarse como ruido blanco, es decir, una secuencia de variables independientes con media cero y varianza constante cuando se asume homocedasticidad[18][19].

El modelo aditivo se expresa como:

$$y_t = T_t + S_t + C_t + E_t \tag{3.8}$$

En este contexto, el componente irregular E_t representa el término de error tradicional en modelos estadísticos y permite capturar la variabilidad aleatoria residual en la serie temporal. Si el modelo se asume homocedástico, la varianza de E_t permanece constante en el tiempo y es frecuente suponer que sigue una distribución normal. De esta manera el modelo consigue reflejar tanto la estructura determinista como la aleatoria de la serie.

Este modelo es más adecuado cuando la variabilidad estacional y la tendencia tienen una relación aditiva, es decir, cuando la amplitud de la estacionalidad se mantiene constante a lo largo del tiempo. Los modelos aditivos resultan especialmente útiles cuando la magnitud de los cambios en la serie temporal no depende de su nivel promedio[20].

[18] Brockwell y Davis, *Introduction to Time Series and Forecasting*, óp.cit.
[19] Shumway y Stoffer, *Time Series Analysis and Its Applications: With R Examples*, óp.cit.
[20] Hyndman y Athanasopoulos, *Forecasting: Principles and Practice*, óp.cit.

3.1.3.2. Descomposición en Modelos Multiplicativos

Por otro lado, cuando la variabilidad de la componente estacional y del ciclo aumenta proporcionalmente con la tendencia, el modelo aditivo puede no resultar adecuado. En estos casos, se emplea el modelo multiplicativo, definido como:

$$y_t = T_t \times S_t \times C_t \times E_t \tag{3.9}$$

En este modelo, el componente irregular E_t sigue interpretándose como el término de error, pero ahora afecta proporcionalmente al nivel de la serie. Cuando se quiere analizar este modelo bajo técnicas aditivas, es común aplicar una transformación logarítmica, la cual convierte la relación multiplicativa en aditiva.

3.1.3.3. Transformación Logarítmica

La transformación logarítmica se utiliza para convertir un modelo multiplicativo en uno aditivo y facilitar la aplicación de métodos lineales y homocedásticos. Matemáticamente, esto se representa como:

$$\log(y_t) = \log(T_t) + \log(S_t) + \log(C_t) + \log(E_t) \tag{3.10}$$

El término $\log(E_t)$ en la nueva escala cumple la función de capturar el componente de error y se asume que sigue las propiedades de homocedasticidad y distribución normal para mantener la interpretación estadística. Aplicar la transformación logarítmica ayuda a estabilizar la varianza de la serie temporal y facilita el análisis bajo supuestos clásicos

de los modelos aditivos[21][22][23][24].

Este enfoque permite revertir la transformación para interpretar los resultados en la escala original y es especialmente útil cuando la magnitud de la estacionalidad varía proporcionalmente con el nivel de la serie, como ocurre en datos económicos o de crecimiento exponencial.

3.2. CONCEPTOS BÁSICOS DEL APRENDIZAJE DE MÁQUINA

El aprendizaje de máquina constituye una rama de la inteligencia artificial orientada al desarrollo de técnicas que permiten que los sistemas informáticos aprendan y mejoren su rendimiento de manera autónoma, utilizando datos históricos sin requerir la programación explícita de cada tarea individual[25]. Esta disciplina integra principios de estadística, matemáticas e informática para diseñar algoritmos y modelos capaces de descubrir patrones y tomar decisiones de forma automática, adaptándose de manera dinámica conforme disponen de nuevos datos.

Las técnicas del aprendizaje de máquina permiten reconocer patrones, extraer conocimiento, descubrir información y hacer predicciones[26].

3.2.1. Elementos en el aprendizaje de máquinas

El aprendizaje de máquinas incluye tres tipos principales de enfoques, pero antes de detallarlos resulta esencial considerar algunos conceptos fundamentales, como los datos

[21] Box y col., *Time Series Analysis: Forecasting and Control*, óp.cit.
[22] Brockwell y Davis, *Introduction to Time Series and Forecasting*, óp.cit.
[23] Shumway y Stoffer, *Time Series Analysis and Its Applications: With R Examples*, óp.cit.
[24] Hyndman y Athanasopoulos, *Forecasting: Principles and Practice*, óp.cit.
[25] Alpaydin, Ethem. *Introduction to Machine Learning*. 4th. MIT Press, 2020.
[26] Rojas, Esperanza Manrique. «Machine Learning: análisis de lenguajes de programación y herramientas para desarrollo». En: *Revista Ibérica de Sistemas e Tecnologias de Informaç ao* E28 (2020), págs. 586-599.

de entrenamiento, datos de validación, función de pérdida y la validación cruzada, utilizada para evaluar la capacidad de generalización del algoritmo sobre nuevos datos.

Los datos de entrenamiento consisten en datos de entrada y salida que se utilizan para entrenar un modelo de aprendizaje de máquina. Estos ejemplos pueden ser imágenes, textos, señales, datos tabulares u otros tipos de información, dependiendo del problema que se esté abordando.

Durante el entrenamiento, los datos de entrenamiento sirven para estimar los parámetros del algoritmo de aprendizaje[27].

Datos de validación: Son un conjunto de datos utilizado durante el entrenamiento de modelos de aprendizaje de máquinas para ajustar hiperparámetros y verificar la capacidad de generalización del modelo, sin formar parte ni del conjunto de entrenamiento ni del de prueba. Su función principal es servir como una muestra representativa e independiente sobre la cual el modelo no ha sido entrenado directamente y, por tanto, refleja cómo se desempeñaría con información nueva o no vista previamente[28].

La función de pérdida desempeña un papel crucial en el proceso de entrenamiento, ya que guía al modelo hacia una configuración de parámetros que minimiza el error en las predicciones. Diferentes problemas pueden requerir diferentes funciones de pérdida, y elegir la función adecuada es fundamental para obtener buenos resultados en el ML. Además, el diseño de la función de pérdida puede influir en la capacidad del modelo para generalizar a nuevos datos y evitar el sobre ajuste o bajo ajuste[29].

La validación cruzada Es una técnica utilizada en aprendizaje automático y estadística para estimar la precisión de un modelo predictivo al aplicarlo sobre datos independientes, separados de aquellos usados durante el entrenamiento del modelo. Consiste

[27]Goodfellow, Ian; Bengio, Yoshua y Courville, Aaron. *Deep Learning*. MIT Press, 2016. URL: `https://www.deeplearningbook.org/`.

[28]Cebollero, Carmen Mayora. «Deep Learning from a Mathematical Point of View». Trabajo de Fin de Máster. Universidad de Zaragoza, jun. de 2021.

[29]Hastie, Trevor; Tibshirani, Robert y Friedman, Jerome H. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd. Springer, 2009.

en el proceso iterativo de particionar el conjunto de datos en subconjuntos de entrenamiento y prueba, obteniendo como resultado la media de las métricas de evaluación en cada partición, lo que genera una mejor estimación de la capacidad generalizadora del modelo y reduce el sesgo producido por la selección puntual de los subconjuntos de datos.

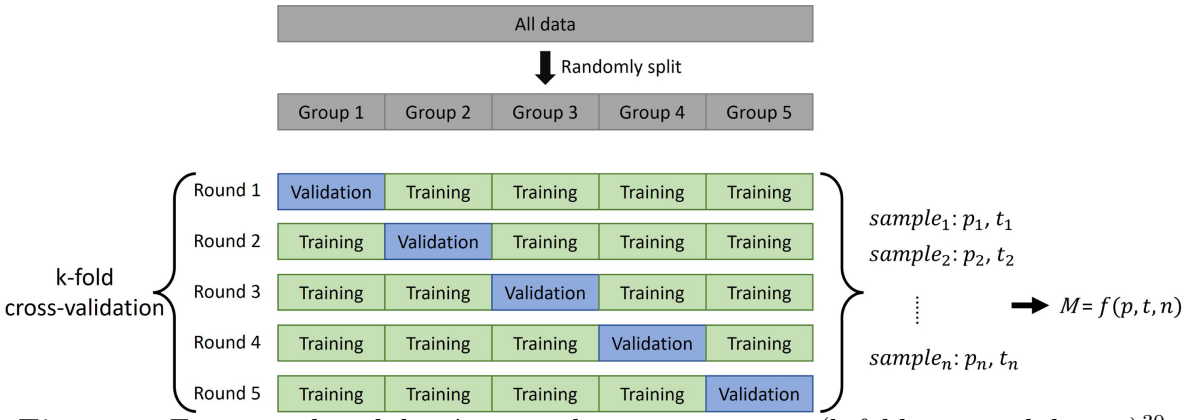


Figura 5: Esquema de validación cruzada por iteraciones (k-fold cross-validation)[30]

La Figura 5 ilustra el flujo de trabajo del método de validación cruzada k-fold para la evaluación de modelos de aprendizaje automático[31]. En este esquema, el conjunto completo de datos se divide aleatoriamente en k grupos del mismo tamaño. Cada ronda utiliza uno de los grupos como conjunto de validación y los restantes como entrenamiento, permitiendo obtener una métrica de desempeño específica para cada pliegue. Los resultados de cada ronda se representan como (p_i, t_i), donde p_i es el conjunto de parámetros ajustados y t_i la métrica obtenida para el pliegue i. Posteriormente, la figura muestra que todas las métricas y conjuntos de parámetros pueden combinarse, típicamente mediante un promedio o función agregada representada por $M = f(p, t, n)$, donde M es el desempeño general, p resume los parámetros ajustados en todos los pliegues, t agrupa las métricas obtenidas y n representa el número total de pliegues. Este procedimiento se repite k veces hasta que todos los subconjuntos han sido empleados como validación, estimando así el rendimiento promedio y la capacidad de generalización

[31] Wang, Yanwen; Khodadadzadeh, Mahdi y Zurita-Milla, Raúl. «Spatial+: A new cross-validation method to evaluate geospatial machine learning models». En: *Int. J. Appl. Earth Obs. Geoinf.* 121 (2023), pág. 103364. ISSN: 1569-8432. DOI: 10.1016/j.jag.2023.103364. URL: https://www.sciencedirect.com/science/article/pii/S1569843223001887.

del modelo.

3.2.2. Tipos de aprendizaje

Los principales tipos de aprendizaje de máquinas son:

- **Aprendizaje supervisado:** Se define como el proceso en el que un algoritmo de aprendizaje busca aprender una función que mapee las entradas a las salidas correspondientes, utilizando ejemplos etiquetados de pares entrada-salida[32]. Durante este proceso, se busca inferir una función a partir de datos de entrenamiento que consisten en ejemplos de entrenamiento, cada uno de los cuales comprende un objeto de entrada y un valor de salida deseado, también conocido como señal de supervisión. El objetivo principal del aprendizaje supervisado es desarrollar un modelo que pueda generalizar de manera efectiva a partir de los datos de entrenamiento, permitiendo que el algoritmo pueda asignar correctamente las etiquetas de clase incluso para instancias invisibles.

- **Aprendizaje no supervisado:** Es una técnica dentro del campo del aprendizaje de máquinas que difiere del aprendizaje supervisado en que no requiere la supervisión directa de los usuarios para entrenar el modelo. En lugar de eso, permite que el modelo opere de forma autónoma para descubrir patrones e información que podrían pasar desapercibidos de otra manera[33]. Esta técnica se centra principalmente en el análisis de datos no etiquetados, lo que significa que el modelo debe aprender a partir de la estructura inherente de los datos sin la guía explícita de etiquetas de clase o categorías previamente definidas.

- **Aprendizaje por refuerzo:** Es un enfoque que se centra en cómo los agentes aprenden a través de la interacción con un entorno para lograr objetivos específi-

[32]Hastie; Tibshirani y Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, óp.cit.

[33]Russell, Stuart y Norvig, Peter. *Artificial Intelligence: A Modern Approach*. 3rd. Prentice Hall, 2010.

cos[34]. A diferencia del aprendizaje supervisado, donde se proporcionan ejemplos etiquetados para entrenar modelos, y del aprendizaje no supervisado, donde el modelo debe encontrar patrones en datos no etiquetados, el aprendizaje por refuerzo se basa en la idea de que los agentes pueden aprender a través de la retroalimentación obtenida de las acciones que realizan.

En el aprendizaje por refuerzo, un agente toma acciones en un entorno y recibe una señal de refuerzo, que puede ser una recompensa o una penalización, en función de la acción tomada y el estado resultante del entorno[35]. El objetivo del agente es aprender una política, es decir, una estrategia de toma de decisiones, que maximice la suma total de recompensas a largo plazo.

Este enfoque se ha aplicado con éxito en una variedad de áreas, incluyendo robótica, juegos, control de procesos y sistemas de recomendación[36]. Al permitir que los agentes aprendan de la experiencia y ajusten su comportamiento en función de las recompensas recibidas, el aprendizaje por refuerzo ofrece una forma poderosa de lograr comportamientos inteligentes en sistemas autónomos.

3.2.3. Construcción de un modelo de aprendizaje de máquina

Podemos resaltar los siguientes pasos a la hora de crear un modelo:

- **Recopilación y Preprocesamiento de Datos:** El primer paso en la construcción de un modelo de Machine Learning es recopilar los datos relevantes para el problema en cuestión. Estos datos pueden provenir de diversas fuentes, como bases de datos, APIs (Interfaces de Programación de Aplicaciones), o incluso la genera-

[34]Sutton, Richard S. y Barto, Andrew G. *Reinforcement Learning: An Introduction*. 2nd. MIT Press, 2018.
[35]Ibíd.
[36]Kaelbling, Leslie Pack; Littman, Michael L. y Moore, Andrew W. «Reinforcement Learning: A Survey». En: *J. Artif. Intell. Res.* 4 (1996), págs. 237-285.

ción de datos sintéticos. Una vez recopilados, los datos deben ser preprocesados para limpiarlos, normalizarlos y prepararlos para su uso en el modelo.

- **Selección y Extracción de Características:** Después de preprocesar los datos, es importante seleccionar las características más relevantes para el problema en cuestión. Esto puede implicar la extracción de características mediante técnicas como el análisis de componentes principales (PCA) o la selección de características basada en la importancia.

- **Elección del Modelo y Entrenamiento:** Una vez seleccionadas las características, se elige un modelo de Machine Learning apropiado para el problema en cuestión. La elección del modelo dependerá de diversos factores, como el tipo de datos, el tamaño del conjunto de datos y la naturaleza del problema. Una vez seleccionado el modelo, se entrena utilizando los datos de entrenamiento.

- **Evaluación del Modelo:** Después de entrenar el modelo, es crucial evaluar su rendimiento utilizando datos de prueba que no se hayan utilizado durante el entrenamiento. Esto permite determinar la capacidad predictiva del modelo y su capacidad para generalizar a nuevos datos.

- **Ajuste y Optimización del Modelo:** Si el rendimiento del modelo no es satisfactorio, es posible que sea necesario ajustar los hiperparámetros del modelo o probar diferentes algoritmos de Machine Learning. Esto puede implicar la búsqueda de hiperparámetros, la validación cruzada y otras técnicas de optimización.

- **Despliegue y Monitoreo:** Una vez que se ha construido y optimizado el modelo, se puede desplegar en un entorno de producción. Es importante monitorear el rendimiento del modelo en producción y realizar ajustes según sea necesario para mantener su precisión y eficacia a lo largo del tiempo.

3.2.4. Python: Librerías y aspectos relevantes en la implementación

En el desarrollo de los modelos presentados en el presente trabajo de grado, **Python** fue el lenguaje principal debido a su robusto ecosistema de librerías especializadas para el

análisis de datos, modelado de series temporales y aprendizaje profundo. A continuación se describen las principales librerías utilizadas y su función específica en los códigos de los apéndices:

3.2.4.1. Principales librerías empleadas

- **NumPy:** Proporciona estructuras eficientes para el manejo de arrays y operaciones matemáticas de alto rendimiento, fundamentales para el procesamiento numérico y la manipulación de grandes volúmenes de datos en series temporales.
- **Pandas:** Permite la manipulación y análisis de datos tabulares mediante estructuras como `DataFrame` y `Series`. Es esencial para la carga, limpieza, transformación y exploración de los datos históricos de precios y variables exógenas.
- **Matplotlib:** Utilizada para la visualización de datos y resultados, permitiendo la creación de gráficos de líneas, boxplots, histogramas y comparaciones entre valores reales y predichos.
- **Seaborn:** Complementa a Matplotlib, facilitando la generación de gráficos estadísticos más atractivos y simplificando la visualización de distribuciones y relaciones entre variables.
- **Scikit-learn:** Proporciona herramientas para el preprocesamiento de datos (escalado, normalización), división de conjuntos de entrenamiento y prueba, y cálculo de métricas de evaluación. Además, se emplea para comparar modelos tradicionales como ARIMA y redes neuronales no recurrentes.
- **TensorFlow y Keras:** Constituyen el núcleo para la construcción, entrenamiento y evaluación de redes neuronales profundas, especialmente las arquitecturas LSTM. Keras, como interfaz de alto nivel, facilita la definición de modelos secuenciales y funcionales, la configuración de capas LSTM, Dense, Dropout y mecanismos de atención.

- **Statsmodels:** Utilizada para el análisis estadístico de series de tiempo, pruebas de estacionariedad y modelado ARIMA, así como para la generación de gráficos de autocorrelación y autocorrelación parcial de residuos.

3.2.4.2. Aspectos relevantes de la implementación

- **Carga y preprocesamiento de datos:** Se emplean Pandas y NumPy para importar archivos Excel, transformar columnas de fechas, crear variables cíclicas (seno y coseno para hora, día y mes), y generar variables derivadas como promedios móviles y diferencias.
- **Visualización y análisis exploratorio:** Matplotlib y Seaborn permiten identificar valores atípicos, tendencias y patrones estacionales en los datos, así como comparar visualmente las predicciones del modelo con los valores reales.
- **Modelado y entrenamiento:** TensorFlow/Keras se utiliza para definir arquitecturas LSTM tanto univariadas como multivariadas, incluyendo capas convolucionales, bidireccionales y mecanismos de atención. Se configuran hiperparámetros como el número de unidades, capas, funciones de activación y regularización.
- **Evaluación y validación:** Scikit-learn y Statsmodels facilitan la evaluación del desempeño mediante métricas como MAE, RMSE y MAPE, así como el análisis de residuos y la validación cruzada.
- **Comparación de modelos:** Se implementan y comparan modelos ARIMA, redes neuronales no recurrentes (ANN) y LSTM, utilizando las mismas herramientas de preprocesamiento y evaluación para garantizar una comparación objetiva.

Estas librerías, ampliamente reconocidas en la comunidad científica y de ingeniería, aseguran la reproducibilidad, eficiencia y escalabilidad de los experimentos realizados en la tesis, y constituyen la base tecnológica para el desarrollo de modelos avanzados de predicción en series temporales energéticas[37].

[37] Goodfellow; Bengio y Courville, *Deep Learning*, óp.cit.

3.2.5. Optimizadores en el entrenamiento de redes neuronales

El entrenamiento de redes neuronales implica la minimización de una función de pérdida mediante la actualización iterativa de los parámetros del modelo (pesos y sesgos). Para ello, se emplean algoritmos de optimización conocidos como **optimizadores**, que determinan cómo se ajustan los parámetros en cada iteración a partir de los gradientes calculados sobre la función de pérdida[38].

Entre los optimizadores más utilizados en aprendizaje profundo se encuentran:

- **Descenso de Gradiente Estocástico (SGD):** Actualiza los parámetros utilizando el gradiente de la función de pérdida respecto a un subconjunto aleatorio (mini-batch) de los datos de entrenamiento. Es simple y eficiente, pero puede presentar problemas de convergencia lenta o quedar atrapado en mínimos locales.
- **Momentum:** Introduce una fracción del gradiente anterior en la actualización actual, acelerando la convergencia y ayudando a superar mínimos locales.
- **RMSProp:** Ajusta la tasa de aprendizaje de cada parámetro de forma adaptativa, dividiendo el gradiente por una media móvil de las magnitudes recientes de los gradientes.
- **Adam (Adaptive Moment Estimation):** Combina las ventajas de Momentum y RMSProp, calculando tasas de aprendizaje adaptativas para cada parámetro a partir de estimaciones de los primeros y segundos momentos de los gradientes[39].

Adam es actualmente uno de los optimizadores más populares en el entrenamiento de redes neuronales profundas, debido a su eficiencia computacional, robustez y capacidad para manejar gradientes ruidosos o dispersos. En esta tesis, Adam fue utilizado para entrenar los modelos LSTM, permitiendo una convergencia rápida y estable.

[38] Íd., *Deep Learning*, óp.cit.; Kingma, Diederik P. y Ba, Jimmy. «Adam: A Method for Stochastic Optimization». En: *Proc. Int. Conf. Learn. Represent.* (2015). URL: `https://arxiv.org/abs/1412.6980`.

[39] Íd., «Adam: A Method for Stochastic Optimization», óp.cit.

Las ecuaciones de actualización de Adam son las siguientes:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{3.11}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{3.12}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{3.13}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{3.14}$$

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{3.15}$$

donde g_t es el gradiente en el paso t, α es la tasa de aprendizaje, β_1 y β_2 son los coeficientes de decaimiento exponencial para los momentos, y ϵ es un pequeño valor para evitar divisiones por cero[40].

Adam es especialmente adecuado para problemas con grandes volúmenes de datos y parámetros, y es ampliamente recomendado en la literatura para el entrenamiento de modelos LSTM y otras arquitecturas profundas[41][42].

3.3. REDES NEURONALES PERCEPTRÓN

Las redes neuronales perceptrón, introducidas por Frank Rosenblatt en 1957, constituyen el modelo matemático más simple de una neurona artificial y son la base de las redes neuronales artificiales modernas. Su estructura y funcionamiento imitan el procesamiento de señales en el cerebro humano, permitiendo la clasificación binaria de patrones y la resolución de problemas linealmente separables.

Definición 3.3.1 (Perceptrón)**.** Sea $\mathbf{x} = (x_1, x_2, \ldots, x_n)^\top \in \mathbb{R}^n$ un vector de entradas

[40] Ibíd.

[41] Goodfellow; Bengio y Courville, *Deep Learning*, óp.cit.

[42] Chollet, François. *Deep Learning with Python*. Manning Publications, 2018. ISBN: 9781617294433. URL: `https://www.manning.com/books/deep-learning-with-python`.

y $\mathbf{w} = (w_1, w_2, \ldots, w_n)^\top \in \mathbb{R}^n$ un vector de pesos asociados. El perceptrón se define como la función:

$$Y = \varphi(z) = \varphi\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

donde:

- $\mathbf{x}$ es el vector de entrada.
- $\mathbf{w}$ es el vector de pesos.
- b es el sesgo.
- $\varphi(z)$ es la función de activación.

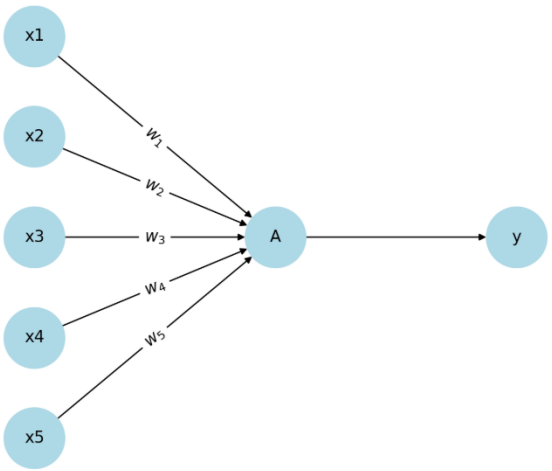


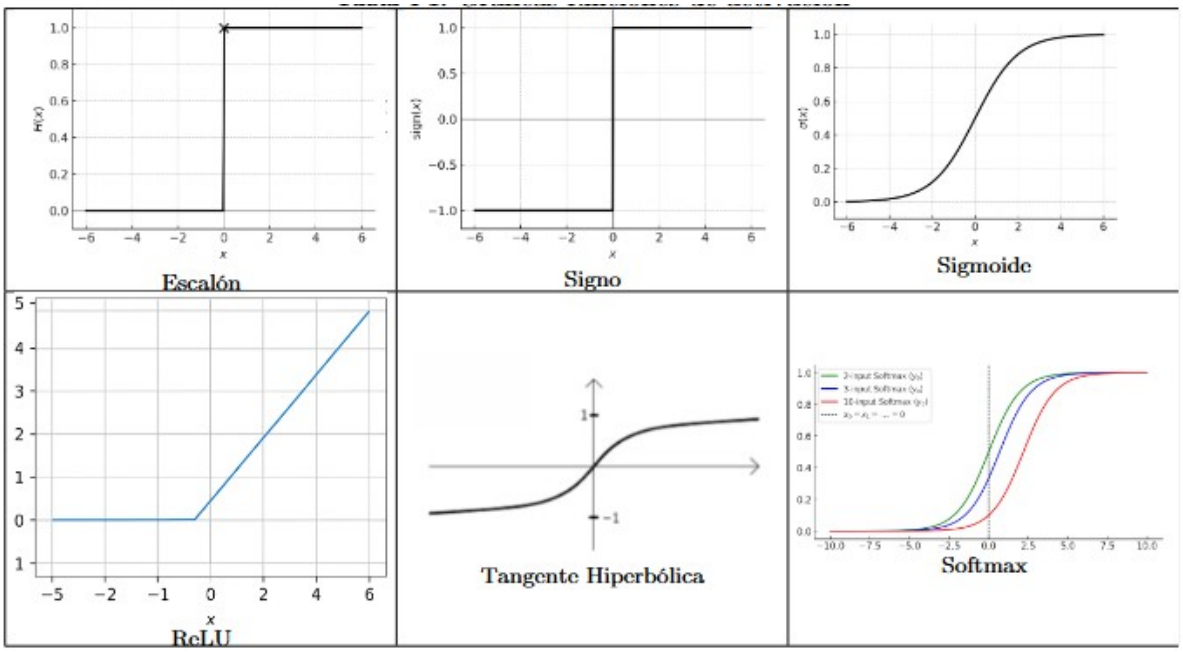
Figura 6: Representación gráfica de un perceptrón con 5 entradas

Alternativamente, si se utiliza el vector aumentado $\tilde{\mathbf{x}} = (1, x_1, \ldots, x_n)^\top$ y $\tilde{\mathbf{w}} = (b, w_1, \ldots, w_n)^\top$, la expresión se simplifica a:

$$y = \varphi(\tilde{\mathbf{w}}^\top \tilde{\mathbf{x}})$$

Geométricamente el perceptrón define un hiperplano en el espacio de las entradas, dado por la ecuación $\mathbf{w}^\top \mathbf{x} + b = 0$, que separa las dos clases. Si los datos son linealmente separables, existe un vector de pesos $\mathbf{w}$ y un sesgo b tal que el perceptrón puede clasificarlos correctamente[43].

Tabla 1: Gráficas de funciones de activación



El perceptrón produce una salida $y = \varphi(z)$. En problemas binarios, las funciones escalón y signo resultan adecuadas por su simplicidad para implementar umbrales y separar dos clases, aunque son no diferenciables y, por tanto, menos idóneas cuando se entrena mediante descenso de gradiente; para modelos más complejos y redes profundas, se prefieren activaciones diferenciables como sigmoide y tanh (a menudo en capas internas), ReLU y variantes (por su eficiencia y mejor propagación del gradiente), y softmax en la capa de salida multiclase para obtener probabilidades normalizadas; las formas gráficas de estas funciones se ilustran en la Tabla 1 y sus fórmulas y usos típicos se resumen en la Tabla 2, mientras que el ajuste de los pesos se realiza en un esquema de aprendizaje supervisado minimizando una función de pérdida, por ejemplo mediante descenso de

[43]Minsky, Marvin y Papert, Seymour. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, 1969; Rosenblatt, Frank. «The perceptron: a probabilistic model for information storage and organization in the brain». En: *Psychol. Rev.* 65.6 (1958), págs. 386-408.

gradiente[44].

Tabla 2: Principales Funciones de Activación

Función	Fórmula	Cuándo se usa
Lineal	$f(x) = x$	Capa de salida en problemas de regresión cuando se requieren valores continuos no acotados.
Escalón	$f(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$	Perceptrón clásico y reglas de decisión binarias con umbral; usada sobre todo en modelos simples por no ser diferenciable.
Lineal a Tramos	$f(x) = \begin{cases} 0 & x < 0 \\ x & 0 \leq x \leq 1 \\ 1 & x > 1 \end{cases}$	Aproximaciones "duras" de sigmoide/tanh (p. ej., HardSigmoid/HardTanh) y activaciones tipo rampa con saturación cuando se prioriza simplicidad o cuantización.
Sigmoidea	$f(x) = \frac{1}{1+e^{-x}}$	Salida de clasificación binaria para probabilidades en $[0, 1]$; también en compuertas de RNN/LSTM junto con tanh.
Gaussiana	$f(x) = e^{-x^2}$	Neuronas en redes de base radial (RBF) donde se requieren activaciones locales centradas en prototipos/centros.
Signo	$f(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases}$	Decisión discreta con etiquetas $\{-1, 0, 1\}$ o $\{-1, 1\}$; rara en entrenamiento por no ser diferenciable.
ReLU	$f(x) = \text{máx}(0, x)$	Activación por defecto en capas ocultas de redes profundas por eficiencia y mejor flujo de gradiente; muy usada en visión y audio.
Tangente Hiperbólica	$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	Capas ocultas cuando se desea salida centrada en 0 y en RNN/LSTM para estados internos junto con compuertas sigmoides.
Softmax	$f_i(\mathbf{x}) = \frac{e^{x_i}}{\sum_j e^{x_j}}$	Capa de salida en clasificación multiclase para obtener probabilidades normalizadas que suman 1.

Debido a la limitación estructural del perceptrón simple, incapaz de resolver problemas no linealmente separables como el de XOR, se introducen arquitecturas con capas ocultas y activaciones no lineales, dando lugar al perceptrón multicapa. El enunciado formal del problema de XOR se encuentra especificado en el Anexo A1 (anexo teórico).

[44]Rumelhart; Hinton y Williams, «Learning representations by back-propagating errors», óp.cit.

3.4. PERCEPTRÓN MULTICAPA

Un perceptrón multicapa es una red neuronal compuesta por una capa de entrada, una o más capas ocultas y una capa de salida, cuyas neuronas aplican activaciones no lineales para modelar relaciones complejas entre variables. En su forma clásica es completamente conectada: cada neurona de una capa se une a todas las neuronas de la siguiente mediante pesos sinápticos, y la información circula solo hacia adelante sin bucles ni retroalimentación. La topología se organiza en capas sucesivas sin conexiones recurrentes, y se considera "profundaçuando incorpora múltiples capas ocultas por su mayor capacidad de representación. La Figura 7 muestra un ejemplo básico con un vector de entrada $X = [x_1, x_2]$ y un vector de salida con 1 elemento $y = [y]$, donde $X \in \mathbb{R}^2$ y $Y \in \mathbb{R}$, por tanto podemos imaginar esta red en particular como una función f tal que $f : \mathbb{R}^2 \longrightarrow \mathbb{R}$.

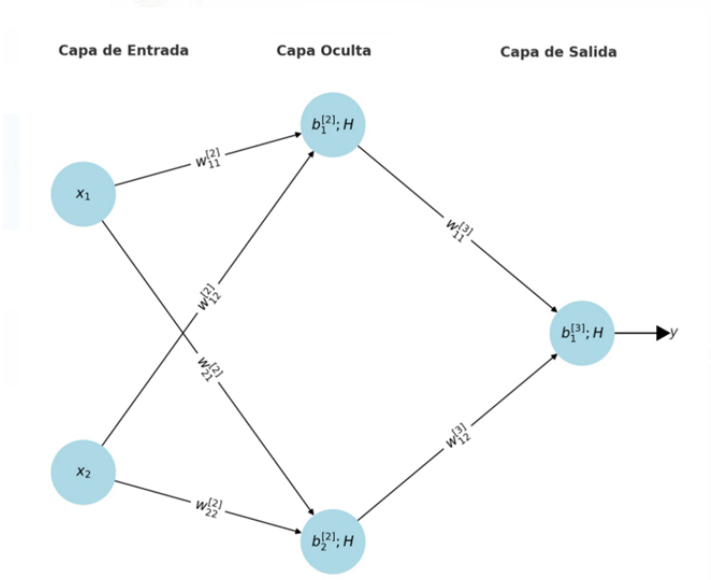


Figura 7: Perceptrón multicapa

Definición 3.4.1 (Perceptrón Multicapa)**.** Sea $L \in \mathbb{N}$ el número total de capas en la red, donde la capa 0 corresponde a la capa de entrada, las capas $1, 2, \ldots, L-1$ son capas ocultas, y la capa L es la capa de salida. Para cada capa $l = 1, 2, \ldots, L$, sea n_l el número de neuronas en dicha capa.

Sea $\mathbf{x} = (x_1, x_2, \ldots, x_{n_0})^\top \in \mathbb{R}^{n_0}$ el vector de entrada, y definamos recursivamente para cada capa l el vector de activaciones $\mathbf{a}^{[l]} \in \mathbb{R}^{n_l}$ y el vector de entradas netas $\mathbf{z}^{[l]} \in \mathbb{R}^{n_l}$ como:

$$\mathbf{z}^{[l]} = W^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]}, \quad l = 1, 2, \ldots, L,$$

$$\mathbf{a}^{[l]} = \sigma^{[l]}(\mathbf{z}^{[l]}),$$

donde:

- $W^{[l]} \in \mathbb{R}^{n_l \times n_{l-1}}$ es la matriz de pesos que conecta la capa $l-1$ con la capa l, cuyos elementos son $w_{ji}^{[l]}$ con $j = 1, \ldots, n_l$ y $i = 1, \ldots, n_{l-1}$.
- $\mathbf{b}^{[l]} = (b_1^{[l]}, b_2^{[l]}, \ldots, b_{n_l}^{[l]})^\top \in \mathbb{R}^{n_l}$ es el vector de sesgos de la capa l.
- $\varphi^{[l]} : \mathbb{R} \to \mathbb{R}$ es la función de activación aplicada elemento a elemento en la capa l.
- Por convención, la activación de la capa de entrada es $\mathbf{a}^{[0]} = \mathbf{x}$.

La salida de la red neuronal es entonces:

$$\mathbf{y} = \mathbf{a}^{[L]} = \varphi^{[L]}\left(W^{[L]}\mathbf{a}^{[L-1]} + \mathbf{b}^{[L]}\right).$$

En el caso particular de una red con una sola neurona en la capa de salida (es decir, $n_L = 1$), la salida es un escalar $y \in \mathbb{R}$.

El Teorema de Aproximación Universal es un resultado fundamental en la teoría del aprendizaje automático y las redes neuronales artificiales, estableciendo que una red

neuronal de una sola capa oculta con una función de activación no lineal adecuada puede aproximar cualquier función continua definida en un subconjunto compacto de $\mathbb{R}^n$ con una precisión arbitraria de $\epsilon > 0$[45][46], este teorema no solo fundamenta el desarrollo de las redes neuronales artificiales, sino que también ha motivado el estudio de arquitecturas más profundas y eficientes, como las redes convolucionales y los modelos de atención, que extienden sus principios a dominios más complejos y de mayor dimensionalidad. Su detalle se puede consultar en el Anexo A2 (anexo teórico).

3.5. DEEP LEARNING

El aprendizaje profundo es una parte del aprendizaje automático y se basa en redes neuronales con múltiples capas que aprenden representaciones jerárquicas mediante composiciones sucesivas de transformaciones no lineales, lo que permite extraer características de bajo, medio y alto nivel de forma automática a gran escala. Esta capacidad ha impulsado avances en reconocimiento de imágenes, audio y lenguaje, favorecida por grandes conjuntos de datos, mejoras algorítmicas como la retropropagación y el aumento de la capacidad computacional (GPU/TPU) que habilitan el entrenamiento de modelos profundos eficientes[47][48].

Históricamente, los cimientos teóricos se remontan al modelo de neurona de McCulloch y Pitts (1943) y al perceptrón de Rosenblatt (1958), cuyo alcance quedó limitado por los resultados críticos de Minsky y Papert (1969) sobre funciones no linealmente separables. El resurgimiento llegó con la retropropagación de Rumelhart, Hinton y Williams (1986), seguido por arquitecturas influyentes como LeNet-5 para reconocimiento de documentos (1998) y las LSTM para dependencias de largo plazo (1997), culminando en la era moderna con AlexNet en ImageNet (2012) y los Transformers de Vaswani et al. (2017),

[45] Cybenko, George. «Approximation by superpositions of a sigmoidal function». En: *Math. Control Signals Syst.* 2.4 (1989), págs. 303-314.

[46] Hornik, Kurt; Stinchcombe, Maxwell y White, Halbert. «Multilayer feedforward networks are universal approximators». En: *Neural Netw.* 2.5 (1989), págs. 359-366.

[47] LeCun, Yann; Bengio, Yoshua e Hinton, Geoffrey E. «Deep learning». En: *Nature* 521.7553 (2015), págs. 436-444.

[48] Schmidhuber, Jürgen. «Deep Learning in Neural Networks: An Overview». En: *Neural Netw.* 61 (2015), págs. 85-117.

que redefinieron el procesamiento secuencial y multimodal en inteligencia artificial[49].

3.5.1. Arquitectura de Redes Neuronales en Deep Learning

En Deep Learning, existen varias arquitecturas de redes neuronales diseñadas para diferentes tipos de tareas. A continuación se muestran las principales:

3.5.1.1. Redes Neuronales Recurrentes RNN

Las redes neuronales recurrentes (RNN) son modelos para datos secuenciales en los que la salida en cada paso depende de la entrada actual y de un estado oculto que resume información previa, incorporando "memoria"mediante conexiones recurrentes y entrenándose típicamente con retropropagación a través del tiempo para capturar dependencias temporales. A diferencia de las redes feed-forward —donde la información fluye solo de la entrada a la salida—, las RNN aprovechan el contexto histórico para predecir elementos futuros o completar secuencias, lo que resulta crucial cuando el significado o la dinámica depende del orden y de lo ocurrido antes. Gracias a ello, se emplean de forma destacada en procesamiento del lenguaje natural (p. ej., modelado de lenguaje y traducción), reconocimiento de voz y predicción de series temporales (incluida la meteorología), así como en generación secuencial de música y texto, ámbitos en los que su manejo del contexto les otorga ventajas frente a arquitecturas sin memoria explícita[50].

Definición 3.5.1 (Red Neuronal Recurrente (RNN))**.** Sea $t \in \mathbb{Z}_{\geq 0}$. Sean $n_0, n_h \in \mathbb{N}$. Sea $\mathbf{x}_t \in \mathbb{R}^{n_0}$ la entrada en el tiempo t, $\mathbf{h}_t \in \mathbb{R}^{n_h}$ el estado oculto, $W_h \in \mathbb{R}^{n_h \times n_h}$ y $W_x \in \mathbb{R}^{n_h \times n_0}$ matrices de pesos, y $\mathbf{b}_h \in \mathbb{R}^{n_h}$ un vector de sesgo. Sea $\varphi : \mathbb{R} \to \mathbb{R}$ una función de activación aplicada elemento a elemento. Dado un estado inicial $\mathbf{h}_0$, se obtiene:

[49] Rumelhart; Hinton y Williams, «Learning representations by back-propagating errors», óp.cit.
[50] Cebollero, «Deep Learning from a Mathematical Point of View», óp.cit.

$$\mathbf{h}_t = \varphi\Big(W_h\,\mathbf{h}_{t-1} + W_x\,\mathbf{x}_t + \mathbf{b}_h\Big).$$

Sea $n_L \in \mathbb{N}$. Sean $W_y \in \mathbb{R}^{n_L \times n_h}$ y $\mathbf{b}_y \in \mathbb{R}^{n_L}$. La salida en el tiempo t se define como

$$\mathbf{y}_t = \varphi_{\text{out}}\Big(W_y\,\mathbf{h}_t + \mathbf{b}_y\Big),$$

donde φ_{out} es una activación de salida acorde a la tarea (identidad en regresión, sigmoide en binaria, softmax en multiclase).

Dos resultado importantes en el marco de las RNN son: el lema de expansión recursiva del estado oculto que permite entender cómo la información de estados pasados y de las entradas sucesivas incide en el estado presente; de forma complementaria, el teorema de condición de gradiente evanescente precisa cuándo los productos de jacobianos $\prod_{k=s+1}^{t} J_k$, con $J_k = \text{Diag}\Big(\varphi'(\mathbf{a}_k)\Big)\,W_h$, provocan que el gradiente decaiga (o explote) exponencialmente con la distancia temporal, orientando decisiones de inicialización (p. ej., ortogonal/unitaria), elección de activaciones y diseño arquitectónico (celdas LSTM, normalización y *clipping*); ambos resultados son fundamentales para este trabajo de grado centrado en la estimación intradiaria del precio de la Bolsa de Energía en Colombia mediante redes LSTM porque guían la selección del horizonte de dependencias, la incorporación de covariables exógenas y la preservación de la señal de aprendizaje a largo plazo con estabilidad numérica en el entrenamiento. Los enunciados formales y las demostraciones completas del lema y del teorema se presentan en los Anexos A3 y A4 (anexo teórico).

3.5.1.2. Redes Neuronales Convolucionales (CNN)

Las redes neuronales convolucionales, desarrolladas en los años ochenta, han demostrado su eficacia en el procesamiento de información bidimensional, tridimensional y temporal mediante operaciones de convolución que emplean kernels especializados para

extraer características distintivas de los datos de entrada. Su mecanismo fundamental de reutilización de pesos representa una de las innovaciones más significativas en el campo del aprendizaje profundo[51].

El **kernel** constituye una matriz de parámetros que actúa como detector de patrones automático, diseñado para identificar características específicas dentro de los datos temporales[5253]. Representa una ventana de análisis de tamaño reducido (respecto a los datos temporales) que se desliza sistemáticamente a lo largo de la serie temporal completa de entrada, realizando en cada posición un producto punto que combina multiplicación y suma para extraer información relevante[54]. La razón fundamental de su definición radica en la necesidad de aprovechar la estructura inherente de los datos temporales, donde elementos adyacentes mantienen relaciones significativas (tienden a estar correlacionados entre sí) que deben ser preservadas y analizadas. Esta eficiencia se refleja en que las CNNs 1D alcanzan una complejidad computacional de orden $O(NK)$ donde N representa la longitud de la serie temporal y K el tamaño del kernel significativamente menor que la complejidad $O(N^2K^2)$ de sus contrapartes bidimensionales[55].

El propósito esencial del kernel consiste en reutilizar un conjunto fijo de parámetros permitiendo que detecte patrones similares independientemente de su ubicación temporal en la serie[5657]. Esta propiedad resulta particularmente valiosa para el análisis de series temporales, procesamiento de audio y análisis de texto, donde los patrones relevantes pueden aparecer en diferentes momentos pero mantener su significado intrínseco[58]. La construcción del kernel se realiza mediante un proceso de aprendizaje automático donde

[51]Hoogen, Jurgen van den. «Time Series Analysis Using Convolutional Neural Networks». Tesis doct. Tilburg, The Netherlands: Tilburg University, ene. de 2025. ISBN: 978-94-6506-648-6. DOI: 10.26116/tshd.30141293.

[52]Rohrer, Brandon. *Convolution in one dimension for neural networks*. Accessed: September 25, 2025. 2024. URL: https://brandonrohrer.com/convolution_one_d.html.

[53]Kiranyaz, Serkan y col. «1D convolutional neural networks and applications: A survey». En: *Mech. Syst. Signal Process.* 151 (2021), pág. 107398. DOI: 10.1016/j.ymssp.2020.107398.

[54]Rohrer, *Convolution in one dimension for neural networks*, óp.cit.

[55]Kiranyaz y col., «1D convolutional neural networks and applications: A survey», óp.cit.

[56]Jain, Abhishek. *Understanding the 1D Convolutional Layer in Deep Learning*. Medium. Accessed: September 25, 2025. Ene. de 2025. URL: https://medium.com/@abhishekjainindore24/understanding-the-1d-convolutional-layer-in-deep-learning-7a4cb994c981.

[57]Hoogen, «Time Series Analysis Using Convolutional Neural Networks», óp.cit.

[58]Jain, *Understanding the 1D Convolutional Layer in Deep Learning*, óp.cit.

los pesos se inicializan aleatoriamente y posteriormente se ajustan a través del algoritmo de retropropagación durante el entrenamiento[5960].

Las redes neuronales convolucionales extraen características jerárquicas mediante la operación de convolución; en el caso temporal, es decir, la red convolucional de una dimensión (Conv1D), un kernel se desliza a lo largo del eje temporal para detectar patrones locales y formar mapas de características que nutren módulos posteriores como capas recurrentes o densas. En una Conv1D, cada kernel es una matriz de pesos aprendibles que cubre una vecindad de longitud k y todos los C_{in} canales; se ajusta por retropropagación para especializarse en motivos temporales de los datos; en la posición t combina linealmente los k pasos vecinos de cada canal, y tras la activación (generalmente ReLU) actúa como extractor de rasgos locales[616263]. El conjunto de kernels es comunmente denominado banco de kernels (o banco de filtros) $W \in \mathbb{R}^{k \times C_{\text{in}} \times C_{\text{out}}}$, donde cada kernel procesa simultáneamente todos los canales de entrada para generar características de salida especializadas. Los hiperparámetros del *padding* p controlan la cobertura de bordes y longitud de salida, mientras el *stride* s regula el nivel de detalle de la secuencia de salida y el costo computacional[646566].

La implementación práctica de la correlación cruzada discreta aplicada a series temporales energéticas se ilustra mediante un ejemplo detallado en el Apéndice A5, donde se demuestra paso a paso cómo un kernel de tamaño 3×3 extrae características de una serie temporal de precios energéticos.

Definición 3.5.2 (Definición formal del kernel en CNN)**.** En una red neuronal convolucional, el *kernel* (también denominado *filtro*) es una matriz (o tensor en el caso multidimensional) de parámetros entrenables, de tamaño reducido en comparación con

[59]Rohrer, *Convolution in one dimension for neural networks*, óp.cit.

[60]Hoogen, «Time Series Analysis Using Convolutional Neural Networks», óp.cit.

[61]McCarter, Daniel. «Mathematical Analysis of Convolutional Neural Networks». Tesis de mtría. University of South Dakota, 2023.

[62]Borja-Robalino, Rodrigo; Monleón-Getino, Antoni y Rodellar, Josep. «Matemática oculta bajo el proceso de aprendizaje en redes neuronales convolucionales». En: *Ciencia Latina* (2022).

[63]Beinat, Natalia Casado. «Redes Neuronales Convolucionales y Aplicaciones». Tesis doct. Universidad Complutense de Madrid, 2022.

[64]Kiranyaz y col., «1D convolutional neural networks and applications: A survey», óp.cit.

[65]Jain, *Understanding the 1D Convolutional Layer in Deep Learning*, óp.cit.

[66]McCarter, «Mathematical Analysis of Convolutional Neural Networks», óp.cit.

los datos de entrada, cuya función es extraer patrones locales relevantes. Formalmente, dado un vector o matriz de entrada $\mathbf{X}$, el kernel $\mathbf{K} \in \mathbb{R}^m$ (en el caso unidimensional) o $\mathbf{K} \in \mathbb{R}^{m \times n}$ (en el caso bidimensional), actúa mediante la *operación de convolución discreta*:

$$(\mathbf{X} * \mathbf{K})(t) = \sum_{\tau=0}^{m-1} X(t+\tau)K(\tau),$$

en el caso unidimensional, o más generalmente,

$$(\mathbf{X} * \mathbf{K})(i,j) = \sum_{u=0}^{m-1} \sum_{v=0}^{n-1} X(i+u, j+v)K(u,v),$$

en el caso bidimensional.

Este operador se *desplaza sistemáticamente* sobre la entrada, realizando en cada posición una combinación lineal (producto y suma) entre los valores locales de los datos y los parámetros del kernel. El resultado constituye un *mapa de características* (feature map) que resalta patrones estructurales en la entrada (p. ej., correlaciones temporales en series de tiempo o bordes en imágenes).

El kernel es *entrenado automáticamente* mediante optimización (normalmente backpropagation) y se reutiliza a lo largo de toda la entrada, lo que permite detectar patrones similares en diferentes ubicaciones, reduciendo significativamente el número de parámetros y la complejidad computacional frente a arquitecturas densas.

Definición 3.5.3 (Capa convolucional 1D (correlación cruzada discreta))**.** Sea $X \in \mathbb{R}^{T \times C_{\text{in}}}$ la ventana de entrada, $W \in \mathbb{R}^{k \times C_{\text{in}} \times C_{\text{out}}}$ un banco de filtros 1D, $\mathbf{b} \in \mathbb{R}^{C_{\text{out}}}$ un sesgo por canal de salida, y $\varphi : \mathbb{R} \to \mathbb{R}$ una activación aplicada elemento a elemento[67][68].

[67] CS231n Course Staff. *Convolutional Neural Networks (CNNs / ConvNets).* `https://cs231n.github.io/convolutional-networks/`. Accedido: 2025-09-10. 2016.

[68] Dumoulin, Vincent y Visin, Francesco. «A guide to convolution arithmetic for deep learning». En: *arXiv preprint arXiv:1603.07285* (2016). DOI: `10.48550/arXiv.1603.07285`. URL: `https://arxiv.`

Dados el *padding* $p \in \mathbb{Z}_{\geq 0}$ y el *stride* $s \in \mathbb{Z}_{\geq 1}$, la preactivación y activación de salida se definen por

$$Z_{t,c} = \sum_{u=0}^{k-1} \sum_{r=0}^{C_{\text{in}}-1} X_{t+u-p,r} W_{u,r,c} + b_c, \qquad Y_{t,c} = \varphi\Big(Z_{t,c}\Big),$$

para $t = 0, \ldots, T-1$ y $c = 0, \ldots, C_{\text{out}} - 1$; en práctica de aprendizaje profundo esta operación se denomina çonvolución.ªunque corresponde a la correlación cruzada discreta al no invertirse el kernel[69][70].

Valores concretos del modelo. Con `kernel_size=` $k = 3$, `filters=` $C_{\text{out}} = 128$, `padding='same'` y `stride=` $s = 1$, sobre ventanas de longitud $T = 48$ y $C_{\text{in}} = 18$ canales de entrada, se tiene $p = \lfloor k/2 \rfloor = 1$ y la operación queda

$$Z_{t,c} = \sum_{u=0}^{2} \sum_{r=0}^{17} X_{t+u-1,r} W_{u,r,c} + b_c, \qquad Y_{t,c} = \varphi\Big(Z_{t,c}\Big), \quad \varphi(x) = \text{máx}(0, x),$$

con $t = 0, \ldots, 47$ y $c = 0, \ldots, 127$, preservando la longitud temporal de la secuencia ($T_{\text{out}} = T$) por el esquema `same`+stride 1[71].

3.6. REDES LSTM

Las redes neuronales LSTM (Long Short-Term Memory) son una extensión de las redes neuronales recurrentes que amplían su capacidad para aprender y recordar patrones a largo plazo. A diferencia de las redes neuronales recurrentes tradicionales, las LSTM

`org/abs/1603.07285`.

[69] CS231n Course Staff, *Convolutional Neural Networks (CNNs / ConvNets)*, óp.cit.

[70] Dumoulin y Visin, «A guide to convolution arithmetic for deep learning», óp.cit.

[71] Keras Team. *Conv1D layer*. `https://keras.io/api/layers/convolution_layers/convolution1d/`. Accedido: 2025-09-10. 2025.

tienen la capacidad de mantener y utilizar información durante largos periodos de tiempo, lo que las hace especialmente efectivas para tareas que implican secuencias de datos, como el procesamiento del lenguaje natural y la generación de texto.

Las redes neuronales LSTM fueron introducidas por primera vez por Sepp Hochreiter y Jürgen Schmidhuber en 1997. Estas redes han demostrado ser muy efectivas en el procesamiento de secuencias de datos debido a su capacidad para manejar dependencias a largo plazo, lo que las hace ideales para tareas como el análisis, la traducción automática y la generación de texto[72].

Matemáticamente, una red LSTM se caracteriza por su capacidad para almacenar información a lo largo del tiempo mediante el uso de puertas (gates) que regulan el flujo de información. El componente clave en una LSTM es su *celda de memoria*, que es capaz de mantener información durante largos períodos sin sufrir los efectos de desvanecimiento del gradiente.

3.6.1. Celda LSTM

Definición 3.6.1 (Celda LSTM)**.** Sea $x_t \in \mathbb{R}^d$ el vector de entrada en el tiempo t, $h_{t-1} \in \mathbb{R}^n$ el estado oculto anterior y $c_{t-1} \in \mathbb{R}^n$ el estado de la celda en el tiempo $t-1$. Una **celda LSTM** es una unidad de memoria recurrente definida por el siguiente sistema de ecuaciones:

[72]Rumelhart; Hinton y Williams, «Learning representations by back-propagating errors», óp.cit.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{3.16}$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{3.17}$$

$$\tilde{c}_t = \tanh\left(W_c \cdot [h_{t-1}, x_t] + b_c\right) \tag{3.18}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{3.19}$$

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \tag{3.20}$$

$$h_t = o_t \odot \tanh(c_t) \tag{3.21}$$

donde:

- $f_t \in \mathbb{R}^n$ es la **puerta de olvido**, que controla qué información del estado anterior se olvida.
- $i_t \in \mathbb{R}^n$ es la **puerta de entrada**, que regula qué nueva información se almacena en la celda.
- $\tilde{c}_t \in \mathbb{R}^n$ es el **vector candidato** para actualizar el estado de la celda.
- $c_t \in \mathbb{R}^n$ es el **estado de la celda** en el tiempo t.
- $o_t \in \mathbb{R}^n$ es la **puerta de salida**, que decide qué parte del estado de la celda se utiliza para la salida.
- $h_t \in \mathbb{R}^n$ es la **salida de la celda** (estado oculto) en el tiempo t.
- W_f, W_i, W_c, W_o son matrices de pesos; b_f, b_i, b_c, b_o son vectores de sesgo.
- $\sigma(\cdot)$ es la función sigmoide y $\tanh(\cdot)$ la función tangente hiperbólica, ambas aplicadas elemento a elemento.
- $\odot$ denota el producto elemento a elemento (Hadamard).

Esta formulación formaliza el funcionamiento interno de la celda LSTM, donde las puertas de entrada, olvido y salida regulan el flujo de información, permitiendo a la red

aprender dependencias a largo plazo y mitigar el problema del gradiente desvanecido en secuencias largas[73].

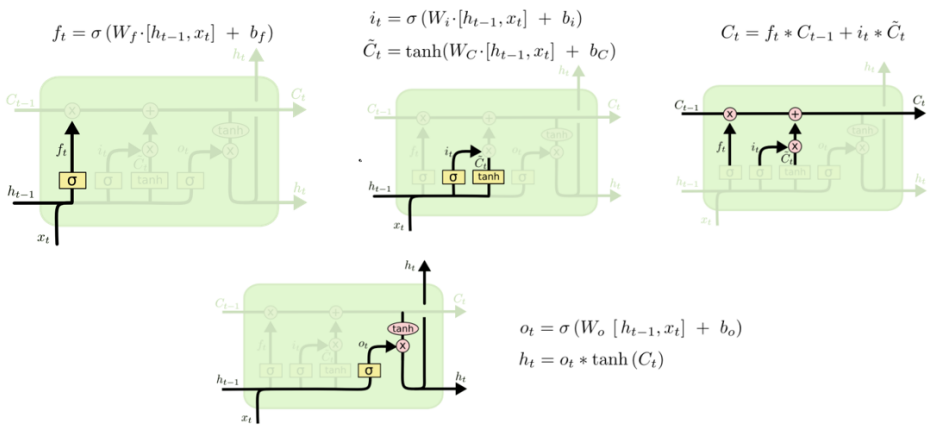


Figura 8: Recorrido paso a paso por LSTM, tomado de Olah, Christopher. *Understanding LSTM Networks*. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`. Accedido: 2025-09-11. 2015; en la figura, el carácter '$*$' representa el producto de Hadamard $\odot$ (producto elemento a elemento), en concordancia con la notación estándar y con Hochreiter, Sepp y Schmidhuber, Jürgen. «Long short-term memory». En: *Neural Comput.* 9.8 (1997), págs. 1735-1780

Las redes LSTM han sido ampliamente utilizadas en una variedad de aplicaciones de modelado de series de tiempo, incluyendo: Predicción de valores futuros en series temporales financieras, como precios de acciones y tipos de cambio, modelado y predicción de demanda en industrias como la logística y el transporte, diagnóstico y predicción de enfermedades en datos médicos de series temporales, entre otras. La capacidad de las redes LSTM para capturar dependencias temporales complejas las hace especialmente adecuadas para estas aplicaciones, donde la precisión y la capacidad de generalización son cruciales.

Algunas de las ventajas clave de las redes LSTM en el contexto de modelos de series de tiempo son: capacidad para capturar, resistencia al problema de desvanecimiento del gradiente, lo que permite el entrenamiento efectivo de redes profundas, flexibilidad para modelar secuencias de diferentes longitudes y frecuencias y adaptabilidad a diferentes

[73]Hochreiter y Schmidhuber, «Long short-term memory», óp.cit.

tipos de datos o patrones temporales.

En este trabajo, el modelo LSTM se ha implementado bajo un enfoque multivariado, integrando un total de 18 covariables adicionales, además de la serie objetivo principal. Específicamente, el vector de entrada x_t en cada instante se constituye por 6 variables cíclicas temporales (hora, día de la semana y mes, cada una codificada como seno y coseno), 3 variables físicas del sistema energético (contratos de energía, generación KW y consumo de combustible MBTU), 3 derivadas estadísticas (media móvil, diferencia entre periodos e interacción entre consumo y generación), y 6 rezagos (lags) de la variable objetivo que representan memoria de corto y largo plazo.

La integración de todas estas covariables se realiza en la **capa de entrada** de la red neuronal, donde se concatenan junto a la variable principal para formar el tensor de entrada $(x_0, x_1, \ldots, x_T)$, de dimensión $T \times d$, siendo $d = 19$ el número total de características. Este tensor multivariado alimenta directamente la primera capa LSTM, permitiendo que la red procese de forma secuencial, para cada paso temporal, la información contemporánea de todas las covariables y la variable objetivo.

La inclusión de covariables cíclicas, físicas, derivadas y de memoria busca maximizar la capacidad de aprendizaje de la arquitectura, permitiéndole capturar patrones estacionales, relaciones físicas y dependencias temporales complejas, lo que resulta en mayor precisión predictiva y robustez frente a variaciones en el sistema energético.

3.6.2. Variables cíclicas y derivadas en el modelado de series temporales

En el análisis y modelado de series temporales, especialmente en contextos como la predicción de precios de la energía eléctrica, es fundamental incorporar variables que permitan capturar tanto la naturaleza periódica de los datos como las relaciones complejas entre diferentes factores. Las variables cíclicas y derivadas enriquecen la representación de la información y mejoran la capacidad predictiva de los modelos de aprendizaje automático y profundo.

3.6.2.1. Variables cíclicas

Las variables cíclicas se emplean para representar fenómenos periódicos (como hora del día, día de la semana y mes) mediante pares seno/coseno que sitúan cada valor sobre el círculo unitario, evitando saltos artificiales entre extremos adyacentes (p. ej., 23 y 0) y facilitando que el modelo identifique patrones recurrentes de forma continua y estable[74][75]. En la práctica, cada atributo con periodo P se codifica como $x_{\sin} = \sin(2\pi\, t/P)$ y $x_{\cos} = \cos(2\pi\, t/P)$ (p. ej., $P = 24$ para la hora, $P = 7$ para el día de la semana y $P = 12$ para el mes), lo que distingue unívocamente posiciones dentro del ciclo y supera las limitaciones de tratarlos como categorías ordinales o con codificación one-hot[76][77]. En precios intradiarios de electricidad, donde se observan picos vespertinos y descensos nocturnos asociados a hábitos de consumo y esquemas de tarifas por periodo, esta representación mejora la captación de regularidades temporales frente a codificaciones lineales. Además de preservar la topología circular y la vecindad natural entre tiempos, la codificación seno/coseno suele favorecer el aprendizaje en modelos secuenciales y redes profundas al ofrecer señales temporales suaves y derivables[78][79].

Estas variables permiten a los modelos identificar y aprender patrones estacionales y cíclicos, mejorando la precisión de las predicciones en contextos donde la periodicidad es relevante.

[74] scikit-learn developers. *Time-related feature engineering*. `https://scikit-learn.org/stable/auto_examples/applications/plot_cyclical_feature_engineering.html`. Accedido: 2025-09-11. 2024.

[75] Skforecast. *Cyclical features in time series forecasting*. `https://skforecast.org/0.8.1/faq/cyclical-features-time-series`. Accedido: 2025-09-11. 2022.

[76] scikit-learn developers, *Time-related feature engineering*, óp.cit.

[77] Data (Feature-engine), Train in. *CyclicalFeatures*. `https://feature-engine.trainindata.com/en/1.8.x/user_guide/creation/CyclicalFeatures.html`. Accedido: 2025-09-11. 2024.

[78] NVIDIA. *Three Approaches to Encoding Time Information as Features for ML Models*. `https://developer.nvidia.com/blog/three-approaches-to-encoding-time-information-as-features-for-ml-models/`. Accedido: 2025-09-11. 2022.

[79] Skforecast, *Cyclical features in time series forecasting*, óp.cit.

3.6.2.2. Variables derivadas

Las variables derivadas se obtienen a partir de transformaciones matemáticas o combinaciones de las variables originales para capturar tendencias, cambios y relaciones no lineales en los datos, y en este proyecto se construyeron antes del escalamiento y de la creación de ventanas para la LSTM. A continuación se describen y se indica su implementación exacta en el código:

- **Promedios móviles:** Suavizan las fluctuaciones a corto plazo e iluminan tendencias locales; para una variable X_t y ventana k,

$$\mathrm{MA}_k(t) = \frac{1}{k} \sum_{i=0}^{k-1} X_{t-i}.$$

- **Diferencias:** Resaltan cambios entre periodos consecutivos; para x_t,

$$\Delta x_t = X_t - X_{t-1}.$$

- **Interacciones:** Capturan efectos conjuntos al multiplicar variables relevantes; por ejemplo,

$$\text{Interacción}(t) = x_t \cdot y_t.$$

- **Lags o retardos:** Incorporan memoria explícita con valores pasados X_{t-k} (p. ej., $k \in \{1, 2, 3, 24, 48, 168\}$ para corto plazo, diario y semanal aproximado):

$$X_{t-k}.$$

Con el fin de reforzar la señal informativa antes del escalamiento y de la construcción de secuencias para la LSTM, en el código se incorporaron variables derivadas que: (i) suavizan ruido de corto plazo (promedios móviles), (ii) resaltan cambios entre periodos consecutivos (diferencias), (iii) capturan efectos conjuntos (interacciones) y (iv) añaden memoria explícita en múltiples escalas temporales (retardos). La inclusión de variables

cíclicas y derivadas en los modelos de series temporales permite capturar tanto la estacionalidad como las relaciones complejas entre las variables del sistema, mejorando la capacidad de generalización y la precisión de las predicciones. Estas transformaciones son ampliamente recomendadas en la literatura para el análisis de series temporales en mercados energéticos y financieros, ya que permiten a los modelos aprender patrones subyacentes que no son evidentes en las variables originales[80].

3.6.3. Evaluación gráfica del desempeño del modelo

La evaluación del modelo de predicción intradiaria se sustentó en un conjunto integrado de diagnósticos visuales que permiten contrastar precisión, robustez y validez de supuestos: (i) el histograma de residuos, para verificar errores centrados en cero y, de ser posible, con distribución aproximadamente normal —asimetrías o colas pesadas sugieren especificación insuficiente o valores atípicos—; (ii) el diagrama de dispersión .$^{\text{er}}$rores vs. predicciones", que debe exhibir aleatoriedad alrededor de cero y homocedasticidad —patrones o abanicos indican sesgo u omisión de variabilidad—; (iii) la serie residuos vs. tiempo", útil para descartar remanencias de tendencia, estacionalidad o cambios estructurales no capturados por la arquitectura; (iv) las funciones de autocorrelación (ACF) y autocorrelación parcial (PACF) de los residuos, cuya ausencia de picos significativos respalda que la dinámica temporal relevante fue absorbida por el modelo; y (v) la superposición "predicciones vs. valores reales.$^{\text{en}}$ el conjunto de prueba, que permite juzgar la capacidad para seguir niveles, giros y amplitudes de la serie —discrepancias sistemáticas señalan áreas de mejora—; en conjunto, estas visualizaciones complementan las métricas cuantitativas, aportan evidencia sobre sesgos, heterocedasticidad y dependencias residuales, y orientan iteraciones informadas de ingeniería de variables, regularización e hiperparámetros para afianzar la generalización del modelo en series energéticas.

[80]Hyndman y Athanasopoulos, *Forecasting: Principles and Practice*, óp.cit.

4. METODOLOG'IA

En este capítulo se describe detalladamente la metodología empleada para desarrollar el modelo de estimación del precio intradiario de la bolsa de energía eléctrica en Colombia mediante redes neuronales LSTM (Long Short-Term Memory). La metodología es un componente crucial de cualquier investigación, ya que establece el marco y las herramientas necesarias para abordar el problema de investigación de manera sistemática y rigurosa, asegurando la validez y confiabilidad de los resultados obtenidos.

La predicción precisa de los precios intradiarios de la energía eléctrica es esencial para los actores del mercado energético colombiano, ya que la alta volatilidad en estos precios puede afectar tanto las decisiones comerciales como operativas. En este contexto, se optó por utilizar redes neuronales LSTM debido a su capacidad superior para manejar y predecir series temporales complejas, como los precios de la energía eléctrica, que presentan patrones no lineales y dependencias a largo plazo, características que los métodos tradicionales de predicción no logran capturar de manera eficiente[1].

Las redes LSTM han demostrado ser más efectivas que los modelos estadísticos tradicionales, como el ARIMA, en la captura de relaciones no lineales y de largo plazo en datos altamente volátiles. Esto se debe a su arquitectura especializada que permite a las redes mantener "memoria"de eventos pasados a través de celdas de memoria, lo que las hace particularmente útiles para series temporales con fluctuaciones complejas[2].

Este capítulo se estructura en varias secciones. Inicialmente, se detallan los procedimientos de recolección y preprocesamiento de los datos históricos de precios y otras variables relevantes del mercado energético colombiano. Posteriormente, se presenta la implementación del modelo LSTM, explicando la arquitectura de la red, la selección de hiperparámetros y las técnicas de optimización utilizadas para mejorar el rendimiento del modelo. En la última sección, se discuten los métodos empleados para evaluar el desempeño del modelo, incluyendo métricas clave como el error cuadrático medio (RMSE), el error absoluto medio (MAE) y el error porcentual absoluto medio (MAPE).

[1] Hochreiter y Schmidhuber, «Long short-term memory», óp.cit.
[2] Ibíd.

Además, se analiza la comparación con otros enfoques de predicción, como modelos estadísticos tradicionales.

Finalmente, se abordan las limitaciones de la metodología utilizada, considerando aspectos como la calidad de los datos y las restricciones inherentes al modelo LSTM, y se presentan las conclusiones derivadas del análisis metodológico, destacando las recomendaciones para futuros trabajos y aplicaciones prácticas.

4.1. FUENTES DE DATOS Y RECOPILACIÓN

4.1.1. Datos históricos de precios de energía

Para el desarrollo del modelo de estimación del precio intradiario de la bolsa de energía eléctrica en Colombia, se utilizaron datos históricos de precios por hora desde el año 2015 hasta 2024. Los datos fueron obtenidos de la plataforma **SINERGIAS** de **XM**, disponible en `https://sinergox.xm.com.co/trpr/Paginas/Informes/PrecioBolsaNacional.aspx`, que proporciona información detallada sobre los precios de la energía eléctrica en la bolsa nacional para cada hora del día.

XM (Centro de Expansión del Mercado), es una entidad responsable de administrar y operar el mercado eléctrico mayorista en Colombia. Su plataforma **SINERGIAS** es una herramienta clave que permite a los participantes del mercado acceder a datos históricos y actuales sobre los precios de la energía, garantizando transparencia y eficiencia en el sector energético.

Los datos de precios de la energía eléctrica están expresados en **pesos colombianos por kilovatio-hora (COP/kWh)** y se encuentran distribuidos por hora (de Hora 0 a Hora 23) para cada día. La base de datos comprende más de 85,000 registros horarios, cubriendo el período desde el 1 de enero de 2015 hasta el 31 de diciembre de 2024. Esta estructura permite un análisis detallado de las fluctuaciones horarias dentro de los días y facilita la identificación de patrones de comportamiento en los precios de la energía a

lo largo del tiempo.

4.1.1.1. Partición temporal del conjunto de datos LSTM multivariado

Tras la construcción de secuencias de ventanas deslizantes con *time_steps* = 48 (horizonte de dos días) aplicadas sobre las características multivariadas escaladas, se realizó una división temporal del conjunto resultante mediante la función `train_test_split` de `sklearn` con los siguientes parámetros:

- **Conjunto de entrenamiento:** 80 % de las secuencias (aproximadamente las primeras observaciones en orden cronológico, desde inicios de 2015 hasta aproximadamente mediados de 2023), utilizado para el ajuste de los parámetros del modelo (pesos y sesgos de las capas convolucionales, LSTM y densas).
- **Conjunto de validación/prueba:** 20 % de las secuencias restantes (aproximadamente desde mediados de 2023 hasta finales de 2024), empleado *simultáneamente* como conjunto de validación durante el entrenamiento (argumento `validation_data` en `model.fit`) para monitorear la pérdida y activar la detención temprana (*EarlyStopping*), y posteriormente como conjunto de prueba para la evaluación final de métricas (MAPE, RMSE, MAE).

La partición se realizó con `shuffle=False` para preservar el orden temporal de las observaciones, evitando la introducción de información futura en el entrenamiento y garantizando que la evaluación refleje un escenario de pronóstico realista. Es importante señalar que, en este esquema, **no se utilizó un conjunto de prueba independiente del conjunto de validación**; el mismo 20 % final cumplió ambas funciones: guiar el entrenamiento mediante la señal de `val_loss` y medir el desempeño final del modelo entrenado.

4.1.2. Variables externas

En el desarrollo del modelo LSTM para la predicción del precio intradiario de la bolsa de energía eléctrica en Colombia, se incorporaron diversas variables externas (exógenas) que influyen significativamente en la formación de los precios. Estas variables permiten al modelo capturar la complejidad y la dinámica real del mercado eléctrico, mejorando la precisión de las predicciones frente a modelos que solo consideran el precio histórico.

A continuación se detallan las variables exógenas integradas en el modelo, extraídas de los datos históricos y de las bases de datos del mercado energético colombiano:

- **Demanda de energía:** Refleja el consumo total de energía eléctrica en el sistema, una de las variables más determinantes en la formación del precio.
- **Generación de energía (GENERATION (kW)):** Cantidad total de energía generada, que puede desglosarse por tipo de fuente (hidráulica, térmica, renovable).
- **Contratos de energía (CONTRATOS DE ENERGIA):** Volumen de energía comprometida mediante contratos, que afecta la oferta disponible en el mercado spot.
- **Consumo de combustible (CONSUMO_COMBUSTIBLE_MBTU):** Medida del consumo de combustibles fósiles (expresado en MBTU), relevante para la generación térmica y su impacto en los costos de producción.
- **Variables temporales y cíclicas:** Hora del día (`hour_sin`, `hour_cos`), día de la semana (`dayofweek_sin`, `dayofweek_cos`), mes del año (`month_sin`, `month_cos`). Estas variables permiten capturar patrones estacionales y cíclicos en la demanda y los precios.
- **Variables derivadas:** Promedios móviles y diferencias de consumo y generación (`consumo_comb_mavg3`, `consumo_comb_diff`, `consumo_gen_interaction`), lags o retardos de la variable objetivo (`value_lag_1`, `value_lag_2`, etc.), que ayudan a modelar la dependencia temporal de los precios.

Estas variables se integran al modelo LSTM como entradas adicionales junto con la serie histórica del precio de la energía. El preprocesamiento incluyó la normalización de todas las variables y la creación de secuencias temporales multivariadas, permitiendo que la red neuronal aprenda tanto de la evolución pasada del precio como de la interacción con los factores exógenos.

La inclusión de variables como la demanda, la generación, los contratos y el consumo de combustible permite al modelo anticipar cambios abruptos en el precio asociados a variaciones en la oferta, la demanda y los costos de producción. Las variables cíclicas y derivadas mejoran la capacidad del modelo para capturar estacionalidades y tendencias recurrentes en el mercado eléctrico.

Tabla 3: Resumen de variables externas empleadas en el modelo

Variable	**Descripción**
Demanda de energía	Consumo total horario del sistema
Generación (GENERATION (kW))	Energía generada por hora
Contratos de energía	Volumen de energía comprometida en contratos
Consumo de combustible (MBTU)	Consumo horario de combustibles fósiles
Hora, día, mes (seno/coseno)	Variables cíclicas para patrones estacionales
Promedios móviles y diferencias	Variables derivadas de consumo y generación
Lags de precio	Retardos de la variable objetivo (precio)

Estas variables fueron seleccionadas tras un análisis de correlación y relevancia, asegurando que cada una aporte información significativa para la predicción del precio intradiario de la energía eléctrica en Colombia.

4.2. MODELO LSTM UNIVARIADO

4.2.1. Preprocesamiento de datos

4.2.1.1. Limpieza y transformación de datos

La limpieza y transformación de datos es un paso esencial en cualquier trabajo de análisis de series temporales, especialmente cuando se estudian los precios intradiarios de la energía eléctrica. Este proceso abarca desde la verificación de datos faltantes, la identificación y tratamiento de valores atípicos, hasta la obtención de resúmenes estadísticos que ayuden a comprender la distribución y volatilidad del conjunto de datos analizados. Todas las etapas detalladas y el código utilizado para este procedimiento pueden consultarse en el repositorio de GitHub[3].

Resumen estadístico Una de las primeras tareas fue la revisión estadística de las columnas numéricas, permitiendo identificar tendencias centrales y la dispersión de los precios por hora. En la Figura 9 se puede observar una impresión parcial de estos estadísticos, lo que facilita el análisis exploratorio inicial para detectar posibles anomalías y determinar la conveniencia de técnicas adicionales de limpieza.

[3] `https://github.com/juank2572/codigo-LSTM-MultivariadoJuan-Carlos-Anaya-/blob/main/PREPROCESAMIENTO_DATOS_UNIVARIADO_JUAN_ANAYA.ipynb`

```
#resumen estadístico de las columnas
df.describe()
```

1 to 8 of 8 entries Filter

index	Hora 0	Hora 1	Hora 2	Hora 3	Hora 4	Hora 5
count	3652.0	3652.0	3652.0	3652.0	3652.0	3652.0
mean	267.99039179574663	261.5137891122855	254.33429458873678	250.15997914514423	254.34936057997447	266.65958785270175
std	288.66689779012313	273.9683092470074	265.6704118906679	258.85938297913805	259.0338688514001	268.30085920187236
min	51.649498	51.649498	46.791501	51.116576	51.649498	51.116576
25%	99.9471810153	99.579583	96.81889600000001	96.46110250000001	99.654521	104.30417375
50%	154.9352925	153.0524625	148.64732650000002	146.851085	152.33614	160.841096
75%	304.7654795011	300.46312350000005	292.46506722780003	286.452249	290.053969	308.07814725000003
max	2674.6989315546	2674.6989315546	2498.777619221	2498.777619221	2498.777619221	2498.777619221

Show 25 per page

Figura 9: Impresión parcial de estadísticos generados

Estadísticas de volatilidad El cálculo de la desviación estándar por cada hora del día dejó en evidencia la alta volatilidad de los precios a lo largo de la jornada. Se organiza a continuación una tabla con la desviación estándar en COP/kWh para cada franja horaria, permitiendo observar las horas con mayor variabilidad y, por tanto, mayor incertidumbre para la predicción.

Tabla 4: Desviación estándar de los precios de energía eléctrica por hora

Hora	**Desviación estándar (COP/kWh)**
Hora 0	288.67
Hora 1	273.97
Hora 2	265.67
Hora 3	258.86
Hora 4	259.03
Hora 5	268.30
Hora 6	268.29
Hora 7	269.71
Hora 8	277.32
Hora 9	280.45
Hora 10	287.36
Hora 11	296.00
Hora 12	295.41
Hora 13	298.93
Hora 14	304.90
Hora 15	307.79
Hora 16	311.68
Hora 17	317.64
Hora 18	330.78
Hora 19	328.83

Identificación de valores atípicos La visualización mediante boxplots para las distintas horas del día permitió identificar fácilmente valores atípicos, aquellos puntos que se alejan significativamente del rango intercuartílico. En el análisis, la mayoría de los precios se concentran en un rango estrecho, pero existen valores extremos que pueden estar asociados a eventos inusuales en el mercado o a irregularidades en la medición. La Figura 10 presenta un ejemplo representativo de estos boxplots, permitiendo apreciar la distribución y la presencia de estos valores extremos.

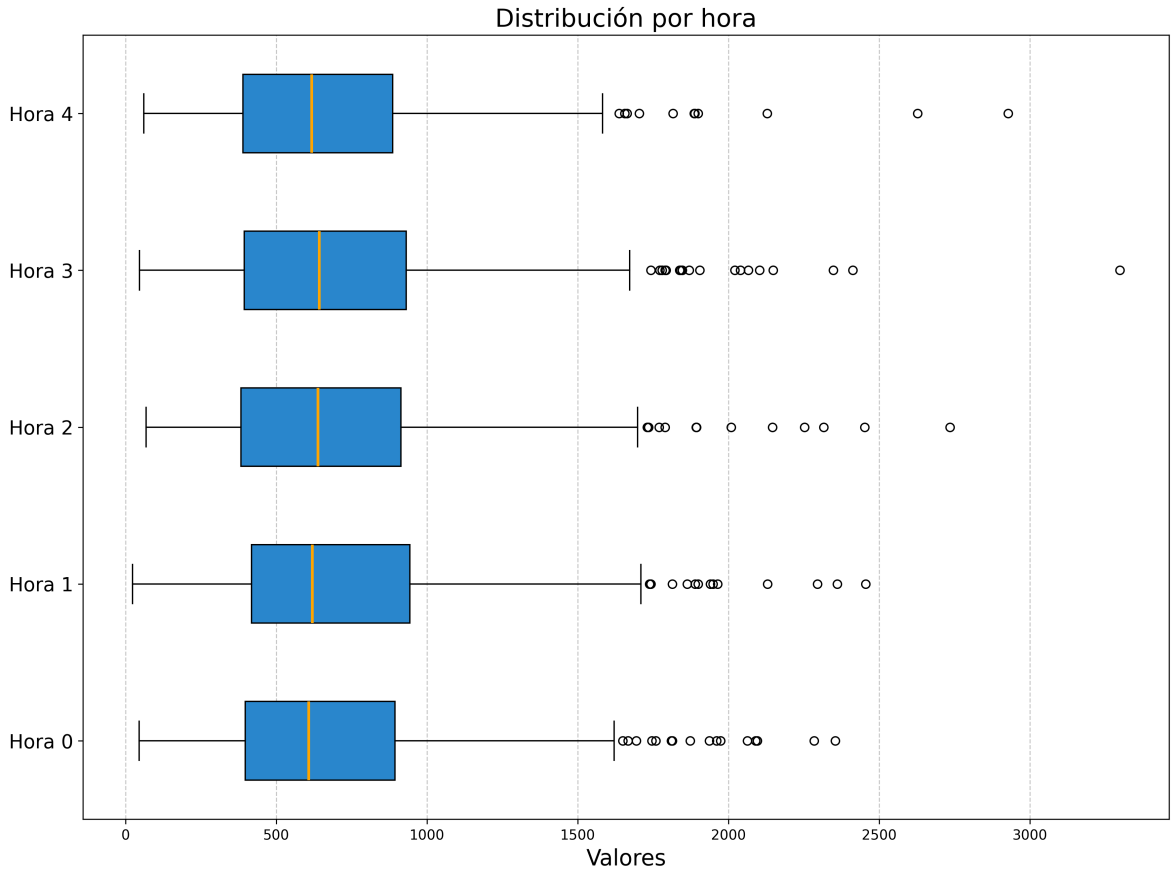


Figura 10: Algunos boxplot de los precios de energía eléctrica por hora generados

Limpieza y duplicidad No se encontraron celdas vacías ni filas duplicadas. Este hecho asegura una base de datos robusta para la modelización subsecuente. El archivo depurado fue guardado en formato Excel para garantizar la reproducibilidad de los resultados y su reutilización en las siguientes fases del trabajo de grado.

4.2.2. Diseño del modelo LSTM univariado

A continuación se realiza una descripción del proceso computacional implementado en el cuaderno de Python alojado en el repositorio de GitHub[4]. El objetivo de dicho código es el desarrollo, entrenamiento y evaluación de un modelo de red neuronal de memoria a corto y largo plazo (LSTM) para la predicción de una serie temporal univariada, correspondiente al precio de la energía eléctrica. Se detalla el flujo de trabajo desde el preprocesamiento de los datos hasta la evaluación del rendimiento del modelo, manteniendo una perspectiva formal y en tercera persona.

4.2.2.1. Preparación y preprocesamiento de datos

El primer paso del procedimiento consiste en la carga de los datos desde un archivo en formato Excel. Para esta tarea, se emplea la biblioteca `pandas`, una herramienta estándar en el ecosistema de Python para la manipulación y el análisis de datos. El conjunto de datos inicial se presenta en un formato tabular donde cada fila representa un día y las columnas corresponden a los precios de la energía para cada una de las 24 horas.

Posteriormente, se realiza una transformación estructural de los datos. El formato ancho original se convierte a un formato largo, generando una única serie temporal. Esta nueva estructura consta de dos columnas principales: un índice de tiempo (`Datetime`), que se construye combinando la fecha y la hora, y una columna de valor (`value`), que contiene el precio de la energía. Este formato es indispensable para el análisis secuencial que requieren los modelos de series de tiempo.

Con la serie temporal ya estructurada, se procede a la normalización de los datos. Se aplica una transformación de escalado Mín-Máx (`MinMaxScaler`) a la columna de precios. Esta técnica reescala los valores de la serie al intervalo $[0, 1]$, lo cual es una

[4] `https://github.com/juank2572/codigo-LSTM-MultivariadoJuan-Carlos-Anaya-/blob/main/MODELO_LSTM_UNIVARIADO_JUAN_ANAYA.ipynb`

práctica fundamental en el entrenamiento de redes neuronales. La normalización asegura que todas las entradas tengan una escala comparable, lo que previene problemas de convergencia durante la optimización y mejora la estabilidad del proceso de aprendizaje por descenso de gradiente.

Finalmente, la serie temporal normalizada se transforma en un conjunto de datos supervisado, adecuado para el modelo LSTM. Se implementa una estrategia de ventana deslizante para generar secuencias de entrada y sus correspondientes etiquetas de salida. Se define una longitud de secuencia (por ejemplo, 30 pasos temporales), de modo que cada muestra de entrada X_i consiste en una secuencia de 30 observaciones pasadas, y su etiqueta y_i es la observación inmediatamente posterior. Este proceso convierte el problema de predicción de series temporales en un problema de regresión supervisada.

4.3. ARQUITECTURA Y COMPILACIÓN DEL MODELO

Para la construcción del modelo predictivo se utiliza la API `Sequential` de la biblioteca `Keras`, la cual permite definir una pila lineal de capas de red neuronal. La arquitectura del modelo LSTM univariado se compone de las siguientes capas:

- **Primera capa LSTM:** Una capa LSTM con 50 unidades de memoria. Se configura con el parámetro `return_sequences=True`, lo que indica que la capa debe devolver la secuencia completa de salidas ocultas en cada paso de tiempo, en lugar de solo la salida final. Esto es necesario para poder apilar una segunda capa recurrente sobre la primera.
- **Segunda capa LSTM:** Una segunda capa LSTM, también con 50 unidades. En esta capa, el parámetro `return_sequences` se omite (asumiendo su valor por defecto, `False`), de modo que solo se propaga la salida del último paso temporal. Esta salida constituye un vector que resume la información relevante de la secuencia de entrada.
- **Capa de salida:** Una capa densa (`Dense`) con una única neurona y una función

de activación lineal implícita. Esta capa final toma el vector de resumen de la capa LSTM anterior y produce una única salida escalar, que corresponde a la predicción del precio de la energía para el siguiente paso temporal.

Una vez definida la arquitectura, el modelo se compila. En este paso se configura el proceso de aprendizaje, especificando el optimizador y la función de pérdida. Se selecciona el optimizador **Adam** (Adaptive Moment Estimation), un algoritmo de optimización estocástica eficiente y ampliamente utilizado. Como función de pérdida, se establece el **error cuadrático medio** (`mean_squared_error`), una métrica estándar para problemas de regresión que mide el promedio de los errores al cuadrado entre los valores predichos y los valores reales.

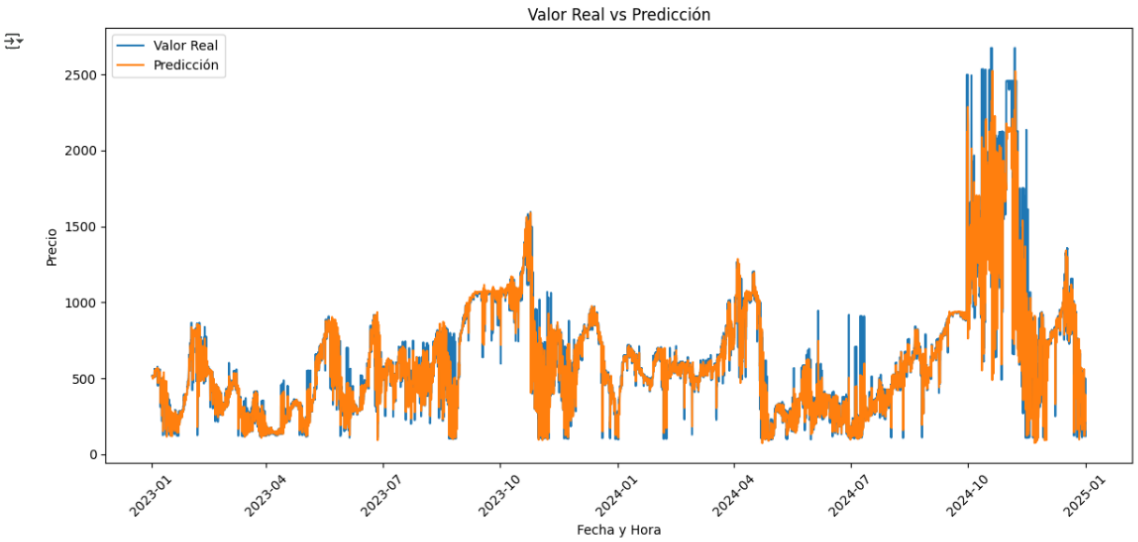
Entrenamiento, evaluación y pronóstico El conjunto de datos secuenciado se divide en dos subconjuntos: uno para entrenamiento y otro para prueba. Para las series temporales, esta división no puede ser aleatoria; debe ser cronológica para evitar la fuga de datos del futuro al pasado. Se asigna un 80 % de los datos iniciales para el entrenamiento del modelo y el 20 % final para su evaluación.

El entrenamiento se lleva a cabo mediante el método `fit`, iterando sobre el conjunto de entrenamiento durante un número predefinido de épocas (por ejemplo, 50) y utilizando un tamaño de lote (`batch_size`) de 32. Durante este proceso, el modelo ajusta sus pesos internos para minimizar la función de pérdida (error cuadrático medio) sobre los datos de entrenamiento. Simultáneamente, se monitorea el rendimiento en el conjunto de validación (los datos de prueba) al final de cada época para observar la capacidad de generalización del modelo y detectar un posible sobreajuste.

Finalizado el entrenamiento, se procede a la evaluación cuantitativa y cualitativa. Se generan predicciones sobre el conjunto de prueba, las cuales están en la escala normalizada $[0, 1]$. Para hacerlas interpretables, estas predicciones se revierten a su escala original de precios utilizando la transformación inversa del `MinMaxScaler`.

El rendimiento del modelo se evalúa de varias maneras:

Análisis detallado de errores y visualización tabular Para una comprensión más profunda del rendimiento del modelo, es útil no solo visualizar las predicciones, sino también calcular y analizar las métricas de error, y presentar los resultados en un formato tabular para una inspección detallada.



	Fecha y Hora	Valor Real	Predicción	Error
0	2023-01-01 20:00:00	518.40	513.081055	5.318945
1	2023-01-01 21:00:00	518.40	514.466431	3.933569
2	2023-01-01 22:00:00	510.90	515.090576	4.190576
3	2023-01-01 23:00:00	510.90	509.913940	0.986060
4	2023-01-02 00:00:00	509.35	508.039917	1.310083

Figura 11: Gráfico de valor real vs. predicción con tabla de resultados detallados

- Las primeras líneas del código (hasta `errores = np.abs(...)`) repiten los pasos de predicción y desnormalización, y añaden el cálculo del **error absoluto** entre las predicciones y los valores reales. Esto proporciona una medida directa de la magnitud de la desviación.
- `resultados = pd.DataFrame(...)`: Se crea un `DataFrame` de Pandas que consolida los valores reales, las predicciones y los errores, junto con las correspondientes fechas y horas del conjunto de prueba. Las fechas se obtienen del índice del `DataFrame` original (`df.index`) tomando las últimas `len(y_prueba)` entradas.
- La sección de graficación (repetida del punto 8.2 pero usando las fechas del

`DataFrame`) genera un gráfico de línea que superpone los valores reales y predichos a lo largo del tiempo, con fechas en el eje X.

- `plt.xticks(rotation=45)` y `plt.tight_layout()`: Mejoran la legibilidad de las etiquetas de fecha en el eje X.

- `resultados.head()`: Muestra las primeras filas de este nuevo `DataFrame`, permitiendo una inspección numérica directa de las predicciones, los valores reales y los errores asociados para los primeros puntos del conjunto de prueba. Como se observa en la Figura 11, la tabla presenta `Fecha y Hora`, `Valor Real`, `Predicción` y `Error`. Por ejemplo, para el 2023-01-01 20:00:00, el valor real fue 518.40 y la predicción 513.08, con un error de 5.31. Esta tabla es crucial para un análisis cuantitativo punto por punto de la precisión del modelo.

Este análisis multifacético del rendimiento del modelo, tanto visual como cuantitativo, es esencial para comprender sus fortalezas y debilidades, especialmente en la predicción de series de tiempo complejas como los precios de la energía.

Para una evaluación exhaustiva de la capacidad predictiva del modelo, es esencial calcular métricas estadísticas clave y analizar la distribución y el comportamiento de los errores (residuos). Esto proporciona una visión cuantitativa de la precisión del modelo y ayuda a identificar patrones no capturados.

Métricas de regresión Se calculan métricas estándar como la Raíz del Error Cuadrático Medio (RMSE), el Error Absoluto Medio (MAE) y el Coeficiente de Determinación (R^2). Estas métricas proporcionan una medida cuantitativa de la precisión del modelo.

- `from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score`: Importa las funciones necesarias del módulo `sklearn.metrics` para calcular el Error Cuadrático Medio, el Error Absoluto Medio y el Coeficiente de Determinación (R^2).

- `mse = mean_squared_error(y_prueba_desnormalizado, predicciones_desnormalizada`
  Calcula el Error Cuadrático Medio (MSE). El MSE penaliza fuertemente los errores grandes, ya que eleva al cuadrado las diferencias entre los valores reales (`y_prueba_desnormalizado`) y las predicciones (`predicciones_desnormalizadas`). Su unidad es el cuadrado de la unidad de la variable objetivo.

- `rmse = np.sqrt(mse)`: Calcula la Raíz del Error Cuadrático Medio (RMSE). El RMSE es la raíz cuadrada del MSE y es una métrica de error muy utilizada. Tiene la misma unidad que la variable objetivo (en este caso, el precio), lo que la hace más interpretable que el MSE. Un RMSE más bajo indica un mejor ajuste del modelo. Las predicciones del modelo se desvían de los valores reales en aproximadamente 106.85 unidades de precio. Considerando la escala de los precios (como se ve en la Figura 11, que van desde 0 hasta más de 2500), este valor es una desviación razonable, pero sugiere espacio para la mejora en la precisión de las predicciones.

- `mae = mean_absolute_error(y_prueba_desnormalizado, predicciones_desnormalizad`
  Calcula el Error Absoluto Medio (MAE). El MAE es el promedio de las diferencias absolutas entre los valores reales y las predicciones. Es menos sensible a los valores atípicos (outliers) que el RMSE y es más robusto a los errores grandes. También tiene la misma unidad que la variable objetivo. El error absoluto medio obtenido es de 43.77 significa que la magnitud promedio del error en las predicciones es de aproximadamente 43.77 unidades de precio. Este valor es menor que el RMSE, lo que es esperable ya que el MAE no penaliza los errores grandes tan severamente como el RMSE.

- `r2 = r2_score(y_prueba_desnormalizado, predicciones_desnormalizadas)`:
  Calcula el Coeficiente de Determinación (R^2). El R^2 mide la proporción de la varianza en la variable dependiente que es predecible a partir de la(s) variable(s) independiente(s). Un valor de R^2 de 1 indica que el modelo explica perfectamente la varianza, mientras que un valor de 0 indica que el modelo no explica ninguna varianza. Un valor negativo indica que el modelo es peor que un modelo que simplemente predice la media. El modelo arroja un R^2 de 0.928 (o 92.8 %) es un R^2 muy alto para un modelo de series de tiempo de predicción de precios. Esto sugiere que el modelo explica aproximadamente el 92.8 % de la variabilidad en

los precios de la energía. Este es un indicador fuerte de que el modelo tiene una buena capacidad para capturar la tendencia general y la mayoría de los patrones en los datos.

Análisis de residuos Se analizan los errores (residuos) del modelo. Se genera un histograma para verificar si su distribución se aproxima a una normal centrada en cero, y un gráfico de dispersión de los residuos contra los valores predichos para detectar patrones como la heterocedasticidad.

El análisis de los residuos (la diferencia entre los valores reales y las predicciones) es fundamental para diagnosticar posibles problemas en el modelo, como sesgos o patrones no capturados.

```
[ ] # Calcular los errores
    errores = y_prueba_desnormalizado - predicciones_desnormalizadas

    # Graficar el histograma de los errores
    plt.figure(figsize=(10, 6))
    plt.hist(errores, bins=50, alpha=0.7, color='blue')
    plt.title('Distribución de los Errores')
    plt.xlabel('Error (Valor Real - Predicción)')
    plt.ylabel('Frecuencia')
    plt.show()
```

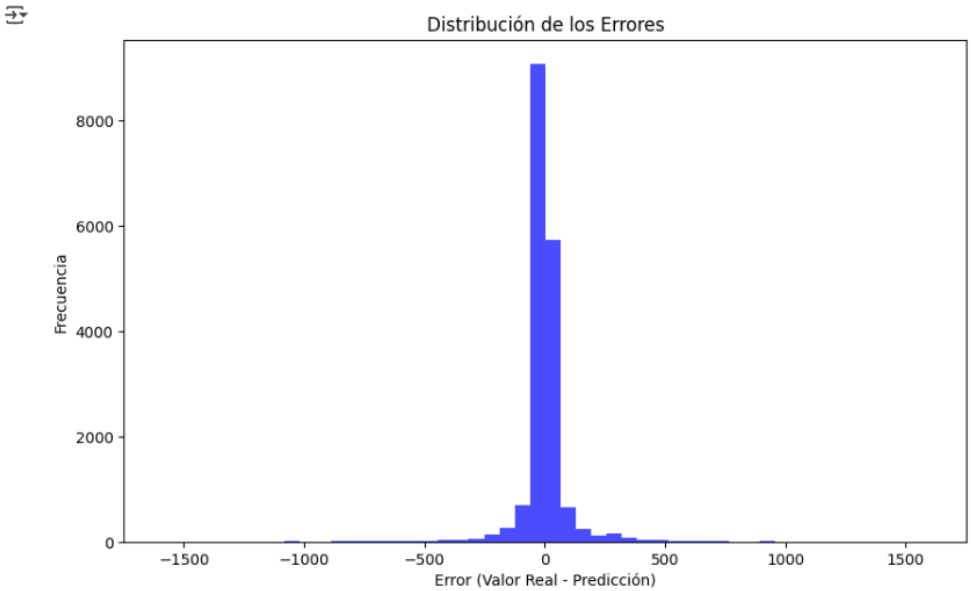


Figura 12: Distribución de los errores (histograma de residuos)

Histograma de residuos

- `errores = y_prueba_desnormalizado - predicciones_desnormalizadas`: Calcula los residuos o errores, que son la diferencia entre los valores reales y los valores predichos.
- `plt.hist(errores, bins=50, alpha=0.7, color='blue')`: Genera un histograma de los errores con 50 "bins"(barras), lo que muestra la distribución de estos errores.
- En el histograma de residuos presentado en la Figura 12 se observa que la distribución de los errores es más apuntada y concentrada en torno a cero que una distribución normal estándar. Adicionalmente, las colas exhiben menor peso en comparación con las colas de la normal, lo que indica una baja frecuencia de errores extremos. Este comportamiento sugiere que no es apropiado asumir normalidad en la distribución de los residuos obtenidos por el modelo, ya que la forma observada difiere significativamente tanto en razón de su curtosis como en la dispersión de los valores atípicos. Por lo tanto, resulta recomendable emplear herramientas de diagnóstico y métodos robustos que no requieran del supuesto explícito de normalidad en el análisis de los errores.

```
[ ] # Gráfico de dispersión de errores vs predicciones
    plt.figure(figsize=(10, 6))
    plt.scatter(predicciones_desnormalizadas, errores, color='blue', alpha=0.5)
    plt.title('Errores vs Predicciones')
    plt.xlabel('Predicción')
    plt.ylabel('Error (Valor Real - Predicción)')
    plt.show()
```

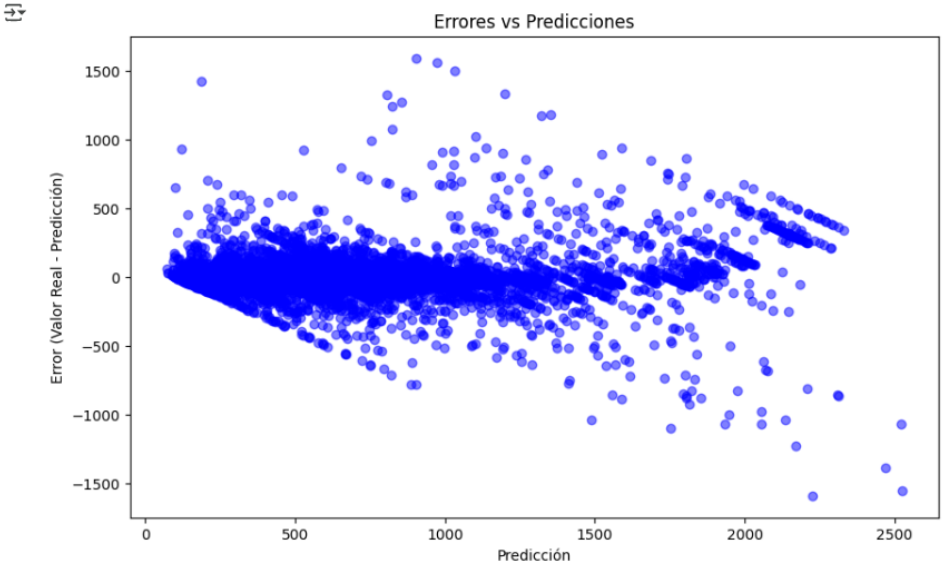


Figura 13: Gráfico de dispersión de errores vs. predicciones

Errores vs. predicciones (gráfico de dispersión)

- `plt.scatter(predicciones_desnormalizadas, errores, color='blue', alpha=0.5)`: Genera un gráfico de dispersión (scatter plot) donde el eje X representa las predicciones del modelo y el eje Y representa los residuos (errores).
- La Figura 13 muestra los Errores vs. Predicciones. En un modelo ideal, los errores deberían estar distribuidos aleatoriamente alrededor de cero, sin mostrar ningún patrón en relación con los valores predichos. Se observa una **forma de cono o abanico invertido**. Los errores son más pequeños (más cercanos a cero) para predicciones de valores bajos y se vuelven **más grandes (mayor dispersión)** a medida que los valores predichos aumentan.
- Esto sugiere **heterocedasticidad**, es decir, que la varianza de los errores no es constante. Para precios de energía más altos, el modelo tiene una mayor incer-

tidumbre o una mayor magnitud de error. Esto es un patrón que el modelo no ha capturado completamente y es común en series de tiempo donde la volatilidad cambia con la magnitud.

- La asimetría positiva observada en el histograma también se refleja aquí, con una mayor dispersión de puntos hacia arriba (errores positivos) en los rangos de predicción más altos.
- Se observa que los residuales muestran un comportamiento independiente respecto al valor ajustado, sin evidencias claras de heterocedasticidad o patrones sistemáticos asociados a la magnitud de las predicciones. Esta independencia y la constancia de la varianza de los errores sugieren que resulta adecuado modelar los residuales como un proceso de ruido blanco, lo que implica que los supuestos de homocedasticidad y no autocorrelación se cumplen en forma razonable dentro del contexto del análisis estadístico realizado.

Finalmente, el código demuestra la capacidad del modelo para generar pronósticos a futuro. Se implementa una estrategia de predicción iterativa, donde la última secuencia de datos conocida se utiliza para predecir el siguiente paso. Esta predicción se añade a la secuencia de entrada para predecir el paso subsiguiente, y el proceso se repite para generar un pronóstico a lo largo de un horizonte de tiempo extendido (por ejemplo, 365 días).

Conclusiones del análisis de métricas y residuos El modelo muestra un rendimiento global prometedor con un R^2 alto, lo que indica que explica una gran parte de la variabilidad en los precios. Sin embargo, el análisis de residuos revela áreas de mejora:

- La presencia de sobreajuste, evidenciada por la divergencia de la pérdida de validación durante el entrenamiento.
- La asimetría en la distribución de errores, con una tendencia a subestimar los picos de precios.

- La heterocedasticidad, indicando que el modelo tiene más dificultad en predecir precios altos con la misma precisión que los precios bajos.

Estas observaciones son críticas para futuras iteraciones del modelo, sugiriendo la necesidad de explorar:

- Técnicas avanzadas de regularización o detención temprana (Early Stopping) para combatir el sobreajuste.
- Modelos que puedan manejar mejor la heterocedasticidad o transformar la serie para estabilizar la varianza.
- Arquitecturas más sofisticadas o la inclusión de características exógenas adicionales que capturen mejor los eventos que causan los picos de precio.

El análisis de residuos es una herramienta indispensable para ir más allá de una simple métrica de error y obtener una comprensión profunda de las fortalezas y debilidades de un modelo predictivo en el contexto de la serie de tiempo.

4.4. MODELO LSTM MULTIVARIADO

4.4.1. Preprocesamiento de datos

A continuación se describe el proceso de preprocesamiento aplicado al conjunto de datos multivariados de precios de la energía eléctrica. El objetivo de esta etapa consistió en transformar los datos brutos en un formato estructurado y normalizado, adecuado para el entrenamiento de un modelo predictivo basado en una red neuronal LSTM.

La implementación se desarrolló en Python dentro de un entorno de Google Colab, utilizando Google Drive para la gestión de los datos. Se emplearon bibliotecas especializadas como `Pandas` para la manipulación de datos, `Numpy` para operaciones numéricas,

y `Matplotlib` junto con `Seaborn` para la generación de visualizaciones. El código fuente completo, que incluye comentarios detallados y las representaciones gráficas de cada paso, se encuentra documentado en el cuaderno disponible en el siguiente repositorio de GitHub[5].

El proceso comienza con la configuración del entorno, la carga del archivo de datos y una inspección preliminar para entender su estructura.

Las primeras filas del `DataFrame` muestran las columnas que contienen la siguiente información relevante para el análisis:

- **Fecha**: Contiene la fecha en que se registran los datos (por ejemplo, `2015-01-01`), está en formato `YYYY-MM-DD`, por ejemplo, `2015-01-01`. La variable de fecha es esencial para los modelos de predicción como LSTM, que requieren una secuencia temporal de datos. La fecha ayuda a capturar la tendencia de precios a lo largo del tiempo.
- **Hora**: Esta columna contiene la hora exacta de cada registro, la hora está en formato `HH:MM:SS`, por ejemplo, `00:00:00`, es relevante porque puede haber variaciones significativas en los precios de la energía dependiendo de la hora del día, lo que el modelo LSTM debe aprender a predecir.
- **Precio Bolsa ($/kWh)**: Representa el precio de la energía en el mercado mayorista, expresado en dólares por kilovatio hora (kWh). Es un valor numérico de tipo `float`, por ejemplo, `184.210107`, cuyo objetivo principal del modelo de predicción. LSTM debe ser entrenado para predecir los precios de la energía, basándose en factores previos como la cantidad de energía generada, contratos, y consumo de combustible.
- **Contratos de Energía**: Indica la cantidad de energía contratada en el mercado, Es un valor numérico de tipo textttfloat, por ejemplo, `6042690.71`. Los contratos de energía reflejan la demanda futura en el mercado y afectan directamente el

[5] `https://github.com/juank2572/codigo-LSTM-MultivariadoJuan-Carlos-Anaya-/blob/main/PREPROCESAMIENTO_DATOS_MULTIVARIADO.ipynb`

precio de la energía. Un mayor volumen de contratos indica una mayor demanda, lo que puede incrementar el precio.

- **Generación (kWh)**: Es un valor numérico de tipo `float`, por ejemplo, `57263.056`, expresado en kilovatios hora (kWh). La generación de energía es un factor clave en la predicción del precio. Si la generación es alta y la demanda es baja, el precio de la energía suele ser bajo. Un modelo LSTM necesita aprender estas relaciones de oferta y demanda.

- **Consumo Combustible MBTU**: .Esta columna contiene el consumo de combustible utilizado para la generación de energía, expresado en unidades de MBTU (Million British Thermal Units). Es un valor numérico de tipo `float`, por ejemplo, `31966.280596`. El consumo de combustible afecta los costos de generación de energía. Si la generación depende de fuentes fósiles, un aumento en el consumo de combustible puede elevar los costos operativos y, por ende, los precios de la energía.

La inspección inicial sugiere que los datos están organizados por fecha y hora, y que se requieren transformaciones para crear una marca de tiempo unificada y para seleccionar las características más adecuadas para el modelado.

4.4.1.1. Selección y Filtrado de Columnas Relevantes

```
#IMPRIMIR LAS 28 PRIMERAS FILAS
df_final.head(28)
```

	Datetime	value	CONTRATOS DE ENERGIA	GENERACION (KW)	CONSUMO COMBUSTIBLE MBTU
0	2015-01-01 00:00:00	184.210107	6042690.71	57263.056792	31966.280596
1	2015-01-01 01:00:00	178.210107	5845780.88	57129.884020	31966.280596
2	2015-01-01 02:00:00	178.210107	5671809.30	52588.497925	31966.280596
3	2015-01-01 03:00:00	178.210107	5591974.57	51389.254952	31966.280596
4	2015-01-01 04:00:00	178.210107	5587617.59	50084.790190	31966.280596
5	2015-01-01 05:00:00	178.210107	5602378.31	49229.619615	31966.280596
6	2015-01-01 06:00:00	178.210107	5496181.47	44903.599904	31966.280596
7	2015-01-01 07:00:00	111.699107	5416719.95	45472.389320	31966.280596
8	2015-01-01 08:00:00	178.210107	5878466.88	47117.387404	31966.280596
9	2015-01-01 09:00:00	178.210107	6101920.94	50522.413069	31966.280596
10	2015-01-01 10:00:00	178.210107	6281922.07	51608.781845	31966.280596
11	2015-01-01 11:00:00	178.210107	6448100.01	53487.979515	31966.280596
12	2015-01-01 12:00:00	178.210107	6484363.60	54963.815922	31966.280596
13	2015-01-01 13:00:00	178.210107	6425635.22	54104.377596	31966.280596
14	2015-01-01 14:00:00	178.210107	6331164.86	53386.362308	31966.280596
15	2015-01-01 15:00:00	178.210107	6267066.36	52708.249423	31966.280596
16	2015-01-01 16:00:00	178.210107	6253820.52	52186.418846	31966.280596
17	2015-01-01 17:00:00	178.210107	6357525.00	51073.066111	31966.280596
18	2015-01-01 18:00:00	202.210107	7373023.42	60489.843670	31966.280596
19	2015-01-01 19:00:00	212.210107	7686921.22	64409.809444	31966.280596
20	2015-01-01 20:00:00	184.210107	7557908.96	64428.096792	31966.280596
21	2015-01-01 21:00:00	184.210107	7184647.78	63077.599712	31966.280596

Figura 14: Información del DataFrame Filtrado.

En esta etapa del preprocesamiento de datos, se seleccionan y filtran las columnas más relevantes para el modelo de predicción LSTM. Para ello se siguen los siguientes pasos: El primer paso consiste en eliminar los espacios extra en los nombres de las columnas, esto es esencial para evitar errores al manipular los datos más adelante. Después de limpiar los nombres de las columnas, se procede a unir las columnas `FECHA` y `HORA` en

una nueva columna llamada `Datetime`, que representará la marca de tiempo. Para esto, primero se convierte `FECHA` y `HORA` a tipo string, y luego se concatenan. Posteriormente, se convertir la concatenación de estas columnas en un formato de fecha y hora. En caso de que haya errores en la conversión, se manejan con `errors='coerce'`, lo que garantiza que los errores se manejen sin interrumpir el proceso.

A continuación, se renombra la columna de precios, que originalmente se llama `PRECIO BOLSA ($/kWh)`, a `value`. Este cambio de nombre facilita la comprensión de la columna y su uso posterior en el modelo.

Finalmente, se crea un `DataFrame` nuevo, denominado `df_final`, que incluye la columna `Datetime` y las demás columnas relevantes para el modelo de predicción: `value`, `CONTRATOS DE ENERGIA`, `GENERACION (kW)` y `CONSUMO COMBUSTIBLE MBTU`. Este paso asegura que solo las columnas necesarias se mantengan en el conjunto de datos final.

Al finalizar este proceso, se imprimen las primeras filas del `DataFrame df_final` para verificar que la selección y filtrado de columnas se haya realizado correctamente como se muestra en la figura 14.

4.4.1.2. Estimación de valores faltantes mediante interpolación lineal

En primer lugar, se efectuó una revisión de valores nulos por columna para cuantificar el problema y localizarlo, identificándose exactamente seis ausencias en la serie GENERACION (KW) mientras el resto de variables no presentaron faltantes, lo que permitió acotar la intervención a una sola columna y preservar la integridad del conjunto multivariado. Con base en este diagnóstico, se estimaron los valores a reemplazar de GENERACION (KW) mediante interpolación lineal sobre el eje temporal, técnica que estima cada valor faltante como punto en la recta que conecta las observaciones válidas inmediatamente anterior y posterior, manteniendo continuidad y suavidad local de la señal acorde con la definición y el comportamiento por defecto del método en pandas. La elección de la interpolación lineal es adecuada en series con muestreo regular porque preserva la estructura secuencial. Tras la estimación, se verificó nuevamente la ausen-

cia de valores nulos en todas las columnas, asegurando que la serie quedó lista para el escalado y la transformación supervisada sin introducir vacíos residuales que afecten el entrenamiento del modelo.

4.4.1.3. Exploración de valores únicos y ceros en GENERACION (KW)

La exploración se centró en GENERACION (KW) porque fue la única variable con evidencias de problemas de integridad en el diagnóstico inicial: el conteo de nulos arrojó exactamente 6 ausencias en esa columna mientras las demás no presentaron faltantes, y además se detectó la presencia de ceros que podían ser no informativos. Para ello se realiza una comprobación exploratoria de la columna GENERACION (KW) para detectar valores atípicos y, en particular, ceros potencialmente no informativos. Primero se extrae el conjunto de valores únicos presentes en la serie y se imprime; listar los únicos es una técnica rápida para reconocer rangos, codificaciones inesperadas y la presencia de ceros o NaN representados con otros códigos, apoyándose en la operación vectorizada de pandas para valores distintos. Luego, se filtran todas las filas cuyo valor en GENERACION (KW) es exactamente cero y se reporta cuántos registros cumplen esa condición; este conteo permite dimensionar si los ceros son casos aislados o un patrón sistemático que requiera corrección o documentación en el preprocesamiento. Finalmente, se muestran las primeras filas de ese subconjunto con ceros a modo de vista preliminar, lo que facilita inspeccionar fechas, otras variables asociadas y posibles causas (p. ej., cortes, fallos de sensor o cargas ausentes) antes de decidir el tratamiento de dichos registros en etapas posteriores; en ese contexto, la revisión de valores únicos y el filtrado de ceros se aplicaron donde había señales de calidad de datos a corregir, evitando transformaciones innecesarias sobre covariables sanas y manteniendo coherencia multivariada antes del modelado LSTM.

4.4.1.4. Tratamiento de ceros y líneas duplicadas

Este bloque identifica los registros con valor cero en GENERACION (KW), los marca explícitamente como valores faltantes sustituýendolos por NaN y reconstruye los huecos resultantes mediante interpolación lineal para preservar la continuidad de la serie temporal; esta estrategia es apropiada cuando los ceros reflejan fallos de medición o cortes operativos más que valores físicos reales, y permite obtener una trayectoria suave compatible con la construcción de ventanas para modelos secuenciales. Tras el rellenado, se verifica que no queden nulos en el DataFrame y se revisan posibles filas duplicadas, encontrando cero número de filas en esa condición.

4.4.1.5. Verificación de valores nulos y ceros

Se efectuó un diagnóstico sistemático de calidad de datos para identificar tanto valores nulos como ceros por columna, seguido de una estrategia de saneamiento que reinterpretó los ceros como faltantes (NaN) y los reconstruyó mediante interpolación lineal solo en las columnas donde existían huecos; el recuento inicial mostró ausencia de NaN en todas las variables, pero evidenció 1826 ceros en CONTRATOS DE ENERGIA, mientras que GENERACION (KW), value, Datetime y CONSUMO COMBUSTIBLE MBTU no presentaron ceros, lo que acotó el problema a una única columna y evitó intervenciones innecesarias en el resto del conjunto multivariado. Tras reemplazar ceros por NaN para habilitar técnicas estándar de completado y aplicar una pasada de interpolación lineal columna a columna donde hubiera faltantes, la verificación final arrojó conteos de nulos iguales a cero en todas las variables, indicando que la reconstrucción fue completa y dejó el DataFrame consistente para las etapas siguientes.

4.4.1.6. Estadísticas básicas del DataFrame

En este paso del preprocesamiento de datos, se realiza un análisis estadístico descriptivo del conjunto de datos. En la Figura 14 se muestran los resultados obtenidos de este

análisis estadístico. A continuación, se describe cada columna de manera detallada:

Datetime: Esta columna tiene un rango de fechas que abarca desde 2015-01-01 00:00:00 hasta 2024-12-31 23:00:00.

value: Los valores en esta columna tienen una media de 298.08, con un valor mínimo de 46.79 y un valor máximo de 2821.52. La desviación estándar de 598.69 muestra que los datos tienen una alta dispersión. Esta variabilidad puede ser resultado de fluctuaciones en los valores de la columna, lo que podría reflejar cambios significativos en el tiempo o diferentes condiciones de operación.

CONTRATOS DE ENERGIA: La media de esta columna es de 9.09e+06, con un valor mínimo de 4.97e+06 y un máximo de 1.44e+07. La desviación estándar es 1.65e+06, lo que sugiere una cierta variabilidad en el número de contratos de energía registrados, aunque el rango de los valores es relativamente constante. La diferencia entre el valor mínimo y el máximo podría reflejar las fluctuaciones en el número de contratos durante el periodo observado.

GENERACION (kW): La media de la generación de energía es 48,319.69 kW, con un mínimo de 20,925.68 kW y un máximo de 571,855.34 kW. La desviación estándar de 148,282.09 kW muestra que los valores de la generación son altamente variables. Este patrón es esperado en la generación de energía, ya que puede haber picos y valles dependiendo de la demanda de energía y de las condiciones de generación.

CONSUMO COMBUSTIBLE MBTU: El consumo promedio de combustible es 14,695.75 MBTU, con un valor mínimo de 3,097.85 MBTU y un máximo de 449,000.00 MBTU. La desviación estándar de 8,177.09 MBTU indica una variabilidad significativa en el consumo de combustible a lo largo del tiempo, lo que es típico en los procesos de generación de energía, donde el consumo de combustible puede depender de múltiples factores como la eficiencia de los equipos y la demanda energética.

Los resultados de estas estadísticas básicas indican que los datos de GENERACION (kW) y CONSUMO COMBUSTIBLE MBTU tienen una dispersión considerable, lo

cual es importante tener en cuenta al realizar el análisis o el modelado. La columna CONTRATOS DE ENERGIA muestra un rango más estrecho, lo que sugiere que el número de contratos de energía varía dentro de un rango más predecible.

```
[ ]  # Estadísticas descriptivas del DataFrame
     print(df_final.describe())
```

```
                           Datetime         value  CONTRATOS DE ENERGIA  \
count                         87648  87648.000000          8.764800e+04
mean   2019-12-09 04:48:35.881708800    298.078735          9.093263e+06
min              1900-01-01 00:00:00     46.791501          4.907331e+06
25%              2017-07-01 23:45:00    118.000537          7.848719e+06
50%              2019-12-31 23:30:00    189.694172          8.976849e+06
75%              2022-07-01 23:15:00    356.250000          1.013888e+07
max              2024-12-31 23:00:00   2821.520551          1.424540e+07
std                              NaN    294.341108          1.652232e+06

       GENERACION (KW)  CONSUMO  COMBUSTIBLE MBTU
count     87648.000000              87648.000000
mean      48319.692723              14695.574530
min       20925.682547               3079.845254
25%       39522.863258               7327.522756
50%       47688.282785              11898.793708
75%       55408.610473              21994.480181
max      571855.349289              44900.157041
std       14822.087278               8177.093862
```

Figura 15: Estadísticas descriptivas del conjunto de datos

La Figura 15 proporciona una visión clara de la distribución de los datos, permitiendo una comprensión más profunda de los patrones subyacentes.

4.4.1.7. Visualización de boxplots para la identificación de valores atípicos

El siguiente paso en el análisis de datos consiste en la visualización de boxplots para cada columna del conjunto de datos, con el fin de identificar posibles valores atípicos y comprender la distribución de los datos.

En la Figura 16, se muestran los boxplots para las columnas relevantes del DataFrame, que incluyen value, CONTRATOS DE ENERGIA, GENERACION (KW) y CONSUMO COMBUSTIBLE MBTU.

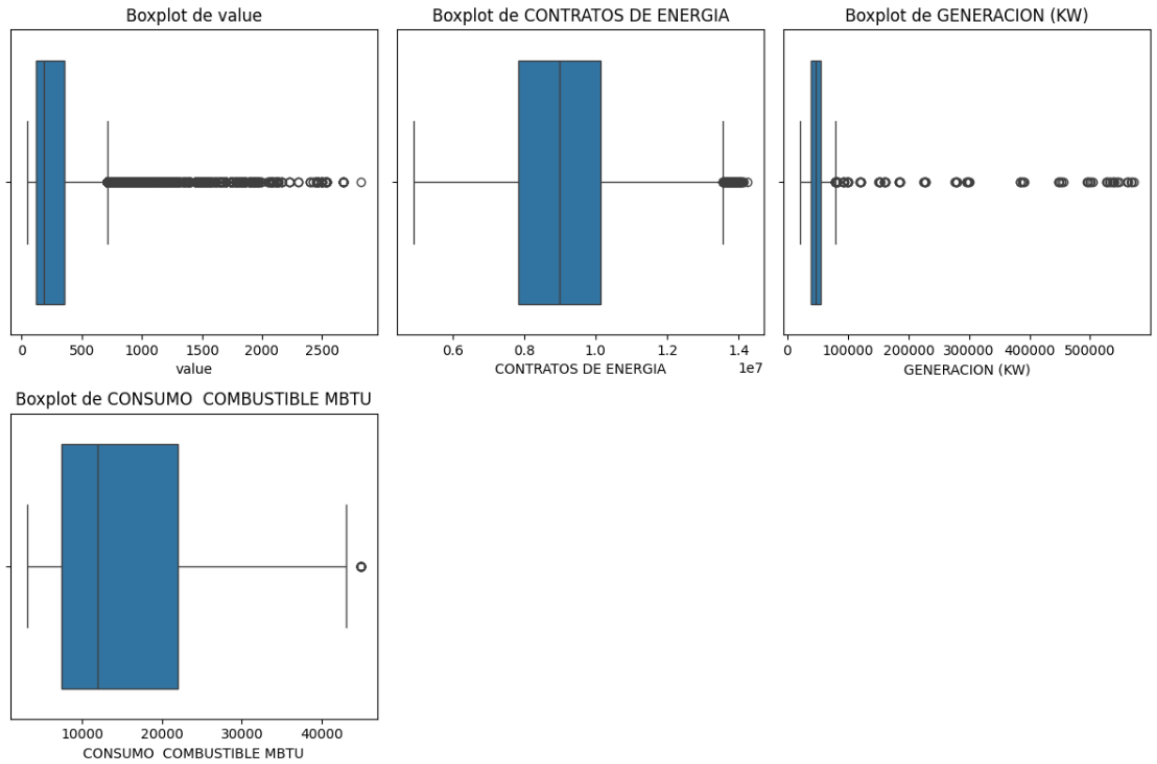


Figura 16: Boxplots de las columnas `value`, `CONTRATOS DE ENERGIA`, `GENERACION (kW)` y `CONSUMO COMBUSTIBLE MBTU`

En los boxplots observamos lo siguiente:

value: La distribución de esta columna muestra que la mayoría de los valores se agrupan en el rango inferior, con algunos valores atípicos hacia el extremo superior. Estos valores atípicos podrían ser causados por fluctuaciones extremas o eventos inesperados que deben ser investigados.

CONTRATOS DE ENERGIA: El boxplot de esta columna muestra una distribución bastante concentrada, lo que indica que la mayoría de los contratos de energía están dentro de un rango estrecho. Sin embargo, también hay algunos valores atípicos que se extienden por encima del valor máximo, lo cual podría señalar datos que merecen una revisión adicional.

GENERACION (KW): Esta columna presenta una gran dispersión, con muchos valores atípicos dispersos más allá de los bigotes del boxplot. Esto es común en los datos de generación de energía, donde se pueden observar picos de generación en ciertos períodos (por ejemplo, debido a una alta demanda).

CONSUMO COMBUSTIBLE MBTU: El boxplot de esta columna muestra una distribución más centrada, con algunos valores atípicos que indican una variabilidad significativa en el consumo de combustible. Al igual que con la generación de energía, estos valores atípicos pueden ser indicativos de fluctuaciones normales.

El análisis visual de los boxplots en la Figura 16 es esencial para la comprensión de la distribución de los datos y el tratamiento adecuado de los valores atípicos antes de la modelización.

4.4.1.8. Visualización de series temporales

Se busca visualizar las series temporales de varias columnas del conjunto de datos en un solo gráfico. Esto es útil para observar la evolución de los valores de distintas variables a lo largo del tiempo, facilitando la comparación entre ellas.

La Figura 17 muestra las series temporales de las columnas `value`, `CONTRATOS DE ENERGIA`, `GENERACION (KW)` y `CONSUMO COMBUSTIBLE MBTU` a lo largo del tiempo. Estos gráficos proporcionan una visualización clara de cómo cada una de estas variables evoluciona en el tiempo, permitiendo identificar tendencias generales, fluctuaciones y posibles valores atípicos.

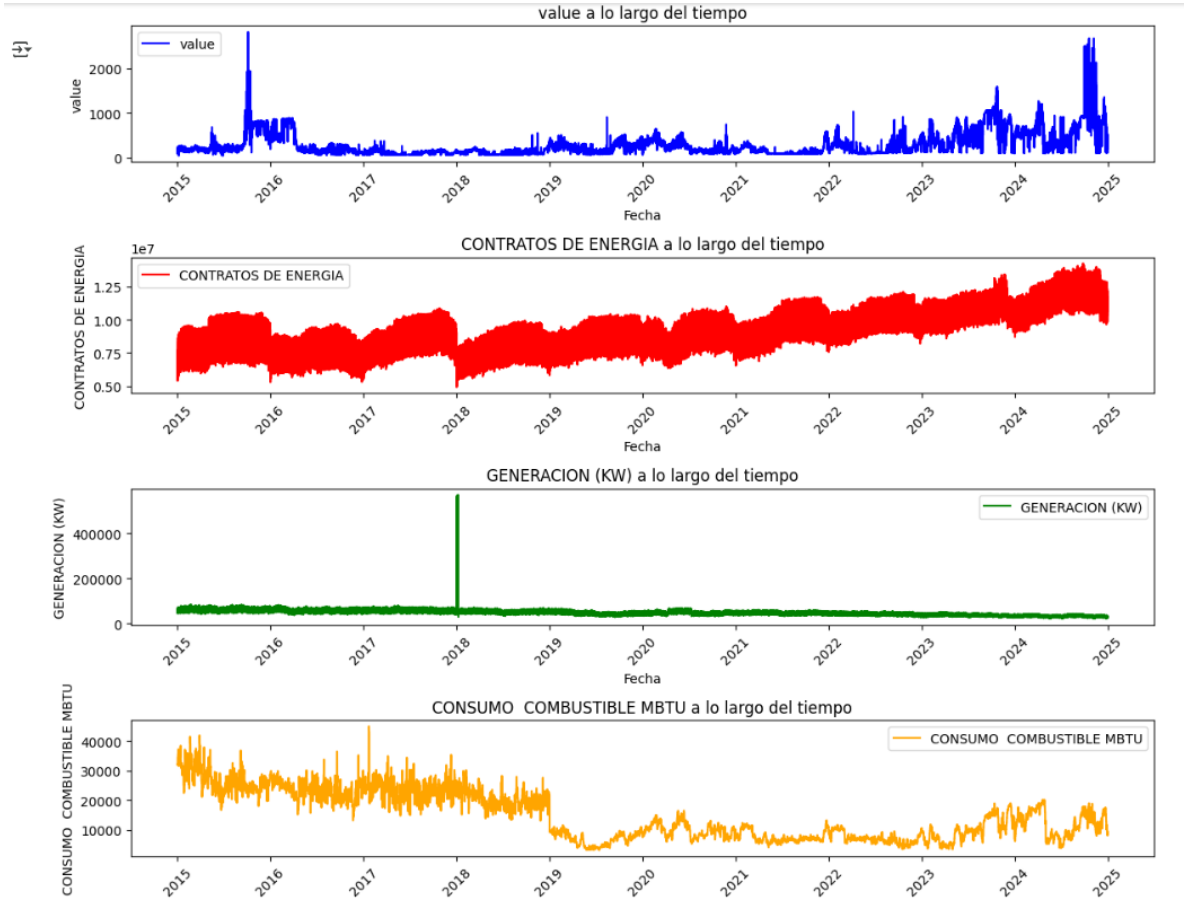


Figura 17: Visualización de las series temporales de las columnas `value`, `CONTRATOS DE ENERGIA`, `GENERACION (KW)` y `CONSUMO COMBUSTIBLE MBTU`

Los resultados obtenidos de los gráficos permiten hacer las siguientes observaciones:

- **value:** La serie temporal de `value` muestra una gran volatilidad, especialmente en los años 2015 y 2024. Los picos pronunciados indican fluctuaciones extremas, lo que podría estar asociado a eventos extraordinarios. La tendencia a estabilizarse en ciertos períodos sugiere que los datos son altamente sensibles a condiciones externas.
- **CONTRATOS DE ENERGIA:** Esta serie presenta una tendencia ascendente a lo largo de los años, desde 2015 hasta 2024, con ciertas fluctuaciones, lo que

sugiere un aumento constante en los contratos de energía. Las caídas breves podrían estar relacionadas con variaciones estacionales o cambios en la demanda de energía.

- **GENERACION (KW):** La columna `GENERACION (KW)` muestra un pico extremo en 2018, que podría estar relacionado con una alta demanda de energía o una anomalía en la generación de energía. Aparte de este pico, la serie muestra una tendencia más constante y estable, lo que sugiere que los datos de generación no presentan cambios drásticos, salvo en eventos específicos.
- **CONSUMO COMBUSTIBLE MBTU:** La serie de `CONSUMO COMBUSTIBLE MBTU` refleja una tendencia descendente, lo que podría indicar mejoras en la eficiencia energética o una disminución en el consumo de combustible a lo largo del tiempo. A pesar de la disminución general, se pueden observar aumentos ocasionales, que podrían correlacionarse con los picos en la generación de energía o la demanda variable de combustible.

Los boxplots y las gráficas de líneas permiten observar cómo las diferentes variables se comportan a lo largo del tiempo. Las gráficas revelan que, aunque existe una tendencia general en cada serie, hay fluctuaciones y valores atípicos que pueden afectar la estabilidad de las series. Esto es importante para el proceso de preprocesamiento, ya que los valores atípicos y las fluctuaciones extremas deben ser analizados y tratados antes de que los datos sean utilizados para entrenar modelos de predicción.

Los resultados obtenidos de la Figura 16 ofrecen una visualización integral de las series temporales, facilitando la identificación de tendencias, fluctuaciones y valores atípicos que pueden ser tratados en el siguiente paso del proceso de modelización. El recuento de entradas no nulas arrojó exactamente 87,648 observaciones válidas en todas las columnas analizadas, y la comprobación de unicidad de esos conteos confirmó que cada serie posee el mismo número de datos; en conjunto, esto indica que el conjunto está completamente alineado entre variables, sin desbalances por faltantes o recortes desiguales, condición favorable para generar ventanas temporales y entrenar modelos que requieran secuencias completas como LSTM.

4.4.2. Descripción código LSTM multivariado

A continuación, se presenta una descripción general del procedimiento seguido para la implementación de un modelo LSTM multivariado con fines de predicción en series temporales. El flujo de trabajo se organizó en diferentes etapas, desde la importación de librerías hasta la evaluación del rendimiento del modelo final.

Primero, se realizó la carga e integración de los datos dentro de un entorno reproducible, montando Google Drive en Google Colab y leyendo el archivo maestro mediante `pandas`. Este procedimiento aseguró un acceso controlado a la fuente de información y garantizó que a lo largo del experimento se trabajara siempre con una única "versión de verdad"del conjunto de datos. Con ello se preservó la trazabilidad, aspecto clave en investigaciones aplicadas a series temporales multivariadas.

Posteriormente, se llevó a cabo una revisión cuidadosa de nombres y tipos de datos. Cada etiqueta de columna fue inspeccionada para detectar la presencia de caracteres ocultos o inconsistencias. Además, se tipificó de manera explícita la columna de sello temporal al formato `datetime`, con el fin de evitar errores silenciosos en etapas posteriores como el filtrado, el escalado y la generación de ventanas temporales. Este paso resulta fundamental, pues en los modelos LSTM la correcta alineación de columnas y el manejo riguroso de la semántica temporal son determinantes para obtener un buen desempeño.

En cuanto a las herramientas empleadas, se importaron las librerías esenciales para el desarrollo del pipeline: `pandas` y `numpy` para la manipulación vectorizada de datos; `scikit-learn` para aplicar el escalado *MinMax*, realizar particiones de entrenamiento y prueba, y calcular métricas de error como MAE, RMSE y MAPE; `matplotlib` para la verificación gráfica de tendencias y predicciones; y finalmente `Keras/TensorFlow` para la construcción del modelo LSTM multivariado. Este último incluyó capas específicas como `Input`, `LSTM` (en su versión estándar y bidireccional), `Dense`, `Dropout` y normalización por lotes. Asimismo, se implementaron `callbacks` tales como `EarlyStopping` y `ReduceLROnPlateau`, orientados a optimizar el proceso de entrenamiento y evitar sobreajustes.

Los pormenores operativos, incluyendo rutas de archivo específicas, bloques de inspección de nombres y el conjunto completo de importaciones, se encuentran documentados en el cuaderno del repositorio de GitHub[6].

4.4.2.1. Variables temporales y cíclicas:

En el contexto del **modelo de estimación del precio intradía de la bolsa de energía eléctrica en Colombia mediante redes LSTM**, la creación de variables temporales y cíclicas es una práctica clave. El modelo busca capturar patrones temporales en los precios de la energía, que están sujetos a variaciones horarias, diarias y mensuales. A continuación, se profundiza en cada parte del proceso y por qué es crucial utilizar funciones seno y coseno.

Hora del día: Los precios de la energía eléctrica muestran variaciones a lo largo del día debido a factores como la demanda, que típicamente es más alta en ciertas horas pico (por ejemplo, en las mañanas y las noches). El modelo debe ser capaz de identificar estas variaciones para hacer predicciones precisas.

Día de la semana: El precio de la energía eléctrica también varía según el día de la semana. Por ejemplo, los precios suelen ser más bajos durante los fines de semana cuando la demanda es más baja en comparación con los días laborales.

Mes del año: Existen patrones estacionales que se repiten a lo largo de los meses debido a factores climáticos, cambios en la demanda energética y comportamientos estacionales en la industria. Estos patrones pueden ser cruciales para prever variaciones en los precios de la energía a lo largo del año.

Las funciones **seno** y **coseno** son particularmente útiles para representar ciclos de forma eficiente. La idea es transformar las variables temporales (hora, día, mes) en una representación que sea útil para el modelo de aprendizaje automático (LSTM en este

[6] `https://github.com/juank2572/codigo-LSTM-MultivariadoJuan-Carlos-Anaya-/blob/main/MODELO_MULTIVARIADO_LSTM_JUAN_CARLOS_ANAYA.ipynb`

caso). Los valores temporales como la hora, el día de la semana y el mes tienen una periodicidad, lo que significa que los valores se repiten de manera cíclica, y las funciones seno y coseno permiten representar estos ciclos de manera matemática.

Ciclicidad de las variables: La hora del día, el día de la semana y el mes del año no son variables lineales, sino que tienen una naturaleza cíclica. Por ejemplo, la hora 23 es muy cercana a la hora 0, lo que indica que el ciclo se repite. Lo mismo ocurre con los días de la semana (lunes y domingo) y los meses (diciembre y enero).

Ventajas de usar seno y coseno: Usar las funciones seno y coseno garantiza que la representación sea continua. En otras palabras, el valor del ciclo no cambiará abruptamente cuando pasemos de un valor alto a un valor bajo (por ejemplo, de las 23:00 a las 00:00). Las funciones seno y coseno proporcionan una simetría matemática que ayuda a captar patrones estacionales y cíclicos de manera eficiente. Dado que los valores generados por las funciones seno y coseno están limitados entre -1 y 1, los valores generados están bien escalados y listos para ser usados en modelos de aprendizaje automático, sin necesidad de normalización adicional.

El uso de funciones seno y coseno para representar variables cíclicas está respaldado por la literatura en análisis de series temporales y aprendizaje automático[7].

4.4.2.2. Variables derivadas

Para el presente trabajo, las *variables derivadas* son aquellas que se calculan a partir de otras variables del conjunto de datos mediante transformaciones matemáticas, agregaciones o combinaciones de características existentes para obtener información adicional útil para el modelo.

Se definen las siguientes variables simbólicas para simplificar la notación:

[7]Hyndman y Athanasopoulos, *Forecasting: Principles and Practice*, óp.cit.

- $C(t)$ = consumo de combustible en el instante t (MBTU)
- $G(t)$ = generación de energía en el instante t (KW)
- Índice t = instante de tiempo (día, hora o período según corresponda)

Bajo esta notación, se definen las siguientes variables derivadas:

1. Promedio móvil de 3 períodos ($M_C(t)$): El promedio móvil es una técnica estadística que suaviza fluctuaciones a corto plazo. Se calcula como:

$$M_C(t) = \frac{1}{3} \sum_{i=t-2}^{t} C(i)$$

Esta variable captura tendencias a mediano plazo y elimina ruido de fluctuaciones diarias, facilitando la identificación de patrones subyacentes en el consumo de combustible.

2. Diferencia primera o cambio diario ($\Delta C(t)$): Mide el cambio en el consumo de combustible entre períodos consecutivos:

$$\Delta C(t) = C(t) - C(t-1)$$

Este cálculo permite captar las fluctuaciones en el consumo, lo cual es crucial para entender variaciones en los precios de la energía.

3. Interacción multiplicativa ($I(t)$): Refleja la influencia conjunta del consumo de combustible y la generación de energía en los costos de producción:

$$I(t) = C(t) \times G(t)$$

Esta variable es importante porque captura relaciones no lineales entre los factores de producción y sus efectos en los precios.

Estas variables derivadas permiten que el modelo LSTM capture dependencias más complejas y relaciones no lineales entre las variables. Las transformaciones matemáticas proporcionan características adicionales que mejoran la capacidad del modelo para aprender patrones subyacentes en los datos. El uso de variables derivadas es una práctica común en el análisis de series temporales y modelos predictivos, especialmente cuando se trabaja con datos financieros o de energía, donde las relaciones entre variables no son siempre lineales o directas[8].

4.4.2.3. Características de retraso (lags)

Las características de *retraso* o *lags* corresponden a variables que almacenan valores pasados de una serie temporal y resultan esenciales para capturar dependencias temporales en datos secuenciales. En el modelo desarrollado para la predicción del precio intradía de la energía eléctrica, estas características cumplen un papel central, ya que permiten que la red LSTM disponga de información histórica explícita para identificar patrones y regularidades en la dinámica del mercado. Al incorporar valores previos de las variables, el modelo puede reconocer de qué manera los precios recientes influyen en la formación de los precios futuros, ajustándose a la naturaleza altamente dependiente del tiempo que caracteriza al comportamiento energético.

En este estudio, las características de retraso fueron creadas mediante desplazamientos en las series originales, siguiendo lo descrito en el cuaderno de GitHub[9]. Dicho procedimiento, implementado a través del método `shift()` de pandas, generó nuevas columnas que contienen valores de cada covariable en momentos previos, lo que permitió al modelo construir una memoria temporal. Esta memoria se convierte en un insumo decisivo, pues los precios de la energía no solo dependen de factores inmediatos como la demanda

[8] Ibíd.

[9] `https://github.com/juank2572/codigo-LSTM-MultivariadoJuan-Carlos-Anaya-/blob/main/MODELO_MULTIVARIADO_LSTM_JUAN_CARLOS_ANAYA.ipynb`

actual o la oferta instantánea, sino también de patrones recurrentes que se manifiestan en horizontes temporales más amplios.

La importancia de los lags en este modelo se evidencia en la capacidad de la LSTM para capturar tanto dependencias de corto plazo, como las variaciones horarias derivadas del consumo durante el día, como dependencias de mayor escala relacionadas con la estacionalidad semanal. De esta forma, al considerar retardos como el de 24 horas o el de 168 horas, el modelo logra representar adecuadamente la influencia diferenciada de los días laborales frente a los fines de semana, así como los cambios en la demanda entre horas pico y horas valle. En consecuencia, las características de retraso no solo enriquecen la representación de los datos, sino que además fortalecen la capacidad predictiva de la arquitectura LSTM, al permitir que el aprendizaje profundo se apoye en la estructura temporal real del mercado eléctrico intradía[10].

4.4.2.4. Escalado de características y objetivo

El escalado de características y objetivos es una parte crucial en el preprocesamiento de los datos, especialmente cuando se trabaja con modelos de aprendizaje automático como las redes neuronales (en este caso, LSTM). La normalización o escalado de los datos asegura que todas las características (y el objetivo) estén dentro de un rango similar, lo que puede mejorar la eficiencia y efectividad del entrenamiento del modelo. En este caso, se ha utilizado el **escalado MinMax** para las características y el objetivo.

El **MinMaxScaler** es un método de escalado que transforma todas las variables, incluidas las características y la variable objetivo, a un rango predefinido, comúnmente entre 0 y 1. En el modelo implementado para la predicción del precio intradía de la energía eléctrica, documentado en el cuaderno de GitHub[11], la elección de este método frente a otros enfoques como la normalización Z-score o el escalado robusto se justifica porque favorece la convergencia eficiente de redes neuronales basadas en gradientes,

[10] Hyndman y Athanasopoulos, *Forecasting: Principles and Practice*, óp.cit.

[11] `https://github.com/juank2572/codigo-LSTM-MultivariadoJuan-Carlos-Anaya-/blob/main/MODELO_MULTIVARIADO_LSTM_JUAN_CARLOS_ANAYA.ipynb`

como la LSTM, al situar todas las variables dentro de un rango comparable y evitar que aquellas con magnitudes mayores dominen el proceso de entrenamiento. Además, el MinMaxScaler preserva la relación relativa entre los valores al ajustarlos según sus mínimos y máximos, sin alterar la forma fundamental de la distribución original, lo que permite mantener la coherencia en la representación de los datos. Otro aspecto relevante es que no requiere asumir una distribución particular de los datos, a diferencia de la normalización Z-score que presupone normalidad, lo cual lo hace más flexible para aplicaciones en contextos energéticos donde las series suelen ser asimétricas o contener valores extremos.

El **MinMaxScaler** reescala cada valor de una característica a un rango de $[0, 1]$, utilizando la siguiente fórmula matemática:

$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

Donde X es el valor original de la característica, X_{min} es el valor mínimo de la característica en el conjunto de datos, X_{max} es el valor máximo de la característica en el conjunto de datos, y X_{scaled} es el valor escalado de la característica, que estará dentro del rango $[0, 1]$.

El uso del **MinMaxScaler** en este modelo ayuda a que las características y el objetivo estén en un rango común (entre 0 y 1), lo que mejora la eficiencia del entrenamiento de redes neuronales como LSTM. Además, al no hacer suposiciones sobre la distribución de los datos, es más flexible y adecuado para una variedad de situaciones. Este tipo de escalado es particularmente útil cuando se utilizan funciones de activación como la **sigmoide** o **tanh**, que requieren que los datos estén dentro de un rango específico para mejorar la convergencia y el rendimiento del modelo[12].

[12]Hyndman y Athanasopoulos, *Forecasting: Principles and Practice*, óp.cit.

4.4.2.5. Creación de secuencias para el modelo LSTM

Una de las etapas fundamentales en la implementación de redes LSTM consiste en la construcción de secuencias de entrada que permitan al modelo aprovechar las dependencias temporales propias de las series. En este trabajo, siguiendo lo documentado en el repositorio de GitHub[13], se definieron ventanas deslizantes de tamaño fijo, denominadas *time steps*, cuyo propósito es agrupar los valores pasados de las variables predictoras para que actúen como contexto en la estimación del valor futuro de la variable objetivo.

Específicamente, se estableció un horizonte de 48 pasos temporales, lo que significa que para predecir el valor en un instante dado, el modelo recibe como entrada la información de las 48 observaciones inmediatamente anteriores. De esta manera, cada secuencia de entrada X está conformada por un bloque de datos que representa dos días completos de registros horarios, mientras que la salida y corresponde al valor de la variable objetivo en el paso siguiente. Este enfoque garantiza que la red LSTM no trabaje con observaciones aisladas, sino con fragmentos de la serie que contienen la estructura secuencial necesaria para identificar tendencias, patrones recurrentes o cambios de régimen.

Posteriormente, las secuencias generadas se dividieron en conjuntos de entrenamiento y prueba sin aplicar mezclado, con el fin de preservar la naturaleza cronológica de los datos. Esta decisión es clave en el análisis de series temporales, pues asegura que el modelo se entrene sobre información pasada y se evalúe sobre observaciones futuras, replicando adecuadamente el escenario real de predicción.

4.4.2.6. Incorporación de un mecanismo de atención

En la arquitectura desarrollada, se incluyó un bloque de atención unidimensional como complemento a las capas LSTM. La motivación para introducir este mecanismo radica en la necesidad de permitir que el modelo asigne distintos niveles de importancia a

[13] `https://github.com/juank2572/codigo-LSTM-MultivariadoJuan-Carlos-Anaya-/blob/main/MODELO_MULTIVARIADO_LSTM_JUAN_CARLOS_ANAYA.ipynb`

los diferentes pasos temporales de la secuencia de entrada. De esta forma, en lugar de tratar todos los valores históricos de manera uniforme, la red neuronal puede concentrar su capacidad de aprendizaje en aquellos instantes pasados que resultan más relevantes para estimar el valor futuro.

El mecanismo de atención se diseñó para operar sobre la dimensión temporal de las secuencias. Inicialmente, se reorganizan las dimensiones de la entrada con el fin de aplicar un proceso de ponderación mediante una capa densa acompañada de una función de activación *softmax*. Este procedimiento genera un conjunto de pesos de probabilidad que reflejan la importancia relativa de cada paso de tiempo. Posteriormente, dichos pesos son reordenados a su formato original y utilizados para multiplicar la secuencia inicial, con lo cual se obtiene una representación en la que cada observación ha sido ajustada en función de su relevancia predictiva.

La ventaja de este enfoque radica en que el mecanismo de atención actúa como un filtro adaptativo que refuerza la memoria de corto plazo o de largo plazo según la dinámica de los datos, permitiendo que el modelo capte con mayor eficiencia patrones situacionales, picos de demanda o caídas abruptas de oferta en el mercado eléctrico. Así, la integración de la atención unidimensional no solo mejora la capacidad de la red para discernir qué información pasada es significativa, sino que también aumenta su interpretabilidad al mostrar en qué momentos específicos concentra su análisis.

4.4.2.7. Entrenamiento del modelo LSTM multivariado

El entrenamiento del modelo CNN-LSTM bidireccional con mecanismo de atención se diseñó para equilibrar eficiencia computacional, estabilidad numérica y generalización en escenario prospectivo. Se empleó validación temporal (entrenamiento en bloque anterior y validación en bloque posterior) con control explícito del sobreajuste mediante detención temprana y reducción adaptativa de la tasa de aprendizaje sobre meseta.

La detención temprana interrumpe el ajuste cuando la pérdida de validación deja de mejorar durante un umbral de épocas y restaura los mejores pesos; la reducción sobre

meseta disminuye la tasa de aprendizaje al estancarse la métrica, permitiendo refinar la convergencia sin oscilar alrededor de mínimos poco profundos. Se fijaron número máximo de épocas y tamaño de lote constantes, se preservó la ordenación temporal de los datos en la evaluación y se registraron históricos de pérdida y métricas para trazabilidad y reproducibilidad de resultados.

A continuación se detallan los hiperparámetros y mecanismos de control utilizados durante el entrenamiento.

Hiperparámetros de optimización

Tabla 5: Hiperparámetros de optimización y entrenamiento

Hiperparámetro	**Valor**	**Justificación**
Optimizador	Adam	Adaptativo, eficiente en gradientes heterogéneos
Tasa de aprendizaje inicial	5×10^{-4}	Balance entre convergencia y estabilidad
Función de pérdida	MSE	Estándar en regresión continua
Epochs máximas	150	Límite superior; permite detención temprana
Batch size	32	Balance entre memoria y estimación de gradiente
Test size (partición)	0.2 (20 %)	80 % entrenamiento, 20 % val./prueba
Shuffle en partición	False	Preserva orden temporal

Mecanismos de control automático del entrenamiento

Se implementaron dos callbacks avanzados para mejorar convergencia y generalización:

Tabla 6: Callbacks para control del entrenamiento

Callback	**Configuración**	**Efecto**
EarlyStopping	Monitor: val_loss, patience: 30, restore best weights	Detiene si no mejora en 30 épocas y restaura mejores pesos
ReduceLROnPlateau	Monitor: val_loss, factor: 0.2, patience: 20, min_lr: 10^{-7}	Reduce LR ($\times$0,2) si estanca 20 épocas

Características multivariadas de entrada

El modelo recibe 19 características escaladas en cada paso temporal, organizadas en los siguientes grupos:

- **Variables exógenas originales (3):** CONTRATOS DE ENERGIA, GENERACION (KW), CONSUMO COMBUSTIBLE MBTU.
- **Características derivadas (3):** Media móvil de 3 periodos del consumo, diferencia primera del consumo, producto de consumo y generación.
- **Características cíclicas codificadas (6):** Seno y coseno de la hora del día (periodicidad diaria), seno y coseno del día de la semana (periodicidad semanal), seno y coseno del mes (periodicidad anual). Estas codificaciones trigonométricas evitan discontinuidades artificiales en las fronteras de ciclos.
- **Rezagos de la variable objetivo (7):** Rezagos de 1, 2 y 3 horas (inmediatos), rezagos de 24 y 48 horas (diarios), rezago de 168 horas (semanal). Estos lags capturan patrones autoregresivos en múltiples escalas temporales.

El tensor de entrada al modelo posee dimensiones $(N_{\text{muestras}}, 48, 19)$, donde N_{muestras} es el número de secuencias de entrenamiento, 48 es el horizonte temporal (dos días), y 19 es el número total de características multivariadas.

Proceso iterativo de entrenamiento

El modelo se entrena sobre el conjunto de datos de entrenamiento (80 % de las secuencias) utilizando optimización por descenso de gradiente adaptativo (Adam) con una función de pérdida MSE. Durante cada época:

1. Se itera sobre lotes de 32 muestras (secuencias de 48 pasos temporales con 19 características).
2. Se calcula el MSE entre predicciones y valores reales escalados.
3. Se propaga hacia atrás el gradiente a través del tiempo (BPTT) a través de las capas convolucionales, LSTM, de atención y densas.
4. Se actualizan los pesos mediante el optimizador Adam con tasa de aprendizaje inicial 5×10^{-4}.
5. Al final de cada época, se evalúa el modelo sobre el conjunto de validación/prueba (20 %), y se calcula la val_loss (MSE en datos no vistos durante entrenamiento).
6. Si val_loss no mejora durante 30 épocas consecutivas, se detiene el entrenamiento y se restauran los pesos que alcanzaron el mínimo de validación (EarlyStopping).
7. Si val_loss se estanca durante 20 épocas, se reduce la tasa de aprendizaje por un factor de 0.2 para explorar mínimos más finos (ReduceLROnPlateau).

Este procedimiento busca maximizar la capacidad generalizadora del modelo sobre datos no vistos, evitando sobreajuste mediante regularización temprana y ajuste dinámico de la tasa de aprendizaje.

Resultados del proceso de entrenamiento

El modelo se entrenó durante un total de **47 épocas** (de un máximo de 150 configurado), alcanzando su mejor desempeño en la **época 12**, donde se registró un val_loss

mínimo de **0.0025** (MSE en escala normalizada). Los pesos correspondientes a esta época fueron restaurados automáticamente por el callback EarlyStopping, garantizando que el modelo final preserve la configuración de parámetros que minimizó el error de validación.

La pérdida final en el conjunto de prueba fue de **0.002467**, muy cercana al mínimo de validación, lo cual confirma que el modelo no experimentó sobreajuste significativo y que su capacidad de generalización se preservó adecuadamente. Adicionalmente, se observó que el callback ReduceLROnPlateau se activó en la época 38, reduciendo la tasa de aprendizaje de 5×10^{-4} a 1×10^{-4}, facilitando la exploración de mínimos más finos en el espacio de parámetros durante las épocas finales del entrenamiento.

5. RESULTADOS

5.1. ANÁLISIS DE LOS RESULTADOS

A continuación, se presenta un análisis detallado de los resultados obtenidos durante la ejecución del modelo de estimación de precios intradiarios en la bolsa de energía eléctrica en Colombia mediante redes neuronales LSTM. Los resultados se presentan en varios gráficos que ilustran la relación entre los valores reales y las predicciones, así como el comportamiento de los residuos, fundamentales para la evaluación de la precisión del modelo.

5.1.1. Residuos a lo largo del tiempo

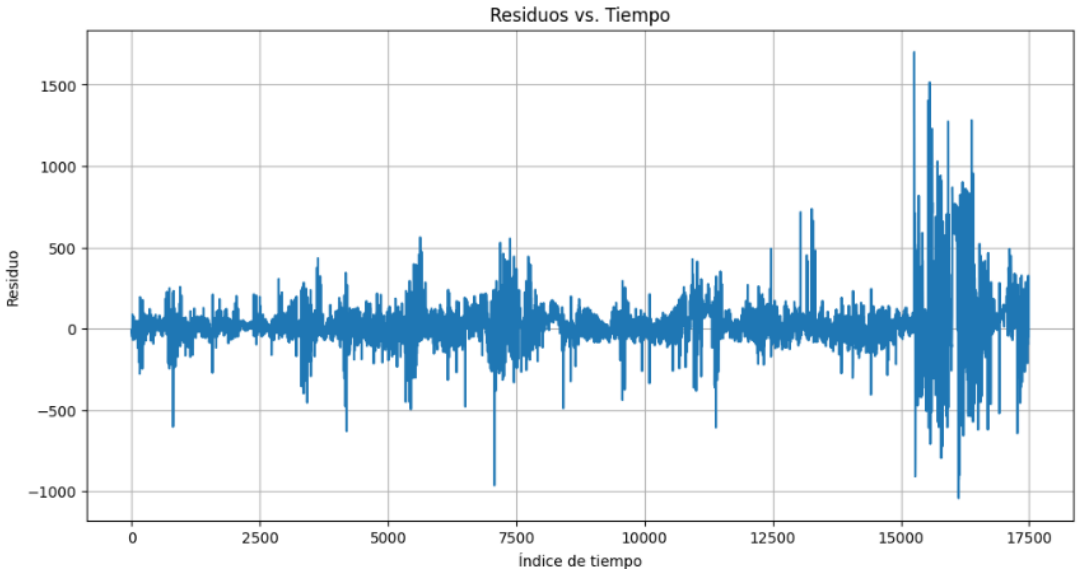


Figura 18: Residuos vs. tiempo

La Figura 18 muestra la evolución de los residuos en el dominio temporal, donde se aprecia un régimen de varianza relativamente estable en la mayor parte del histórico y, hacia el tramo final, un incremento marcado en la amplitud de las oscilaciones, con

episodios de mayor dispersión y presencia de valores extremos; este patrón sugiere heterocedasticidad episódica compatible con cambios de régimen o shocks de mercado, lo que es habitual en precios intradiarios con transiciones entre horarios valle y pico. En términos de diagnóstico, la expansión de la banda residual al final del periodo indica que el error crece bajo determinadas condiciones del sistema (por ejemplo, combinaciones de demanda, oferta o estacionalidad semanal) y motiva extensiones como modelado explícito de varianza condicional, ventanas adaptativas o ajuste de la pérdida ponderando tramos de alta volatilidad.

5.1.2. Residuos frente a predicciones

La Figura 19 grafica los residuos en función de las predicciones desescaladas y revela una estructura en abanico: a medida que crece el nivel predicho, aumenta la dispersión del error y aparecen colas más pesadas en negativo y positivo; este comportamiento es consistente con una varianza del error dependiente del nivel, típica en procesos de precios donde la magnitud del valor condiciona la escala de las fluctuaciones. Desde la práctica de pronóstico, esta evidencia respalda el uso de estrategias de estabilización de la varianza (por ejemplo, transformaciones monotónicas, pérdidas ponderadas por nivel o modelos que predicen tanto media como escala) y sugiere segmentar la evaluación por cuantiles del nivel para identificar zonas de desempeño diferencial.

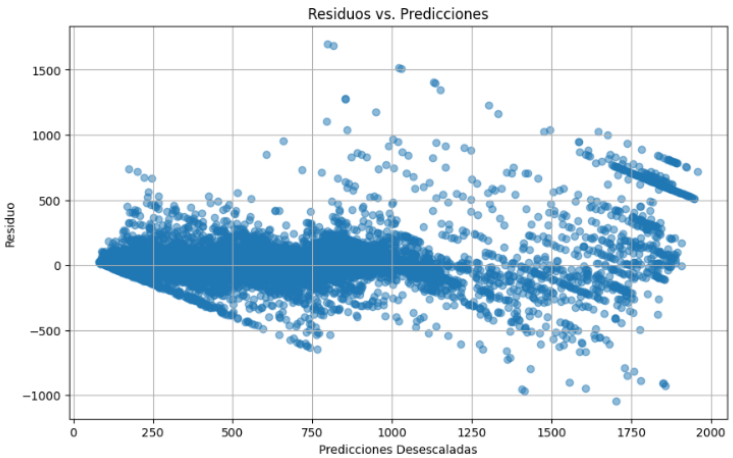


Figura 19: Residuos vs. predicciones

5.1.3. Distribución de los residuos

El histograma de la Figura 20 presenta una masa central aguda alrededor de cero con asimetrías y colas visibles, en consonancia con la estadística descriptiva reportada en la figura (media cercana a cero, desviación estándar moderada y valores extremos apreciables); la curtosis empírica sugiere colas más pesadas que una normal gaussiana, lo cual explica la sensibilidad del RMSE a picos y justifica complementar su lectura con el MAE y el MAPE en la evaluación. Para tareas operativas, estas colas implican riesgo de errores grandes en escenarios puntuales; en consecuencia, resulta pertinente considerar métricas robustas adicionales, análisis por percentiles del error e, incluso, pérdidas asimétricas si el costo de sobreestimar y subestimar difiere.

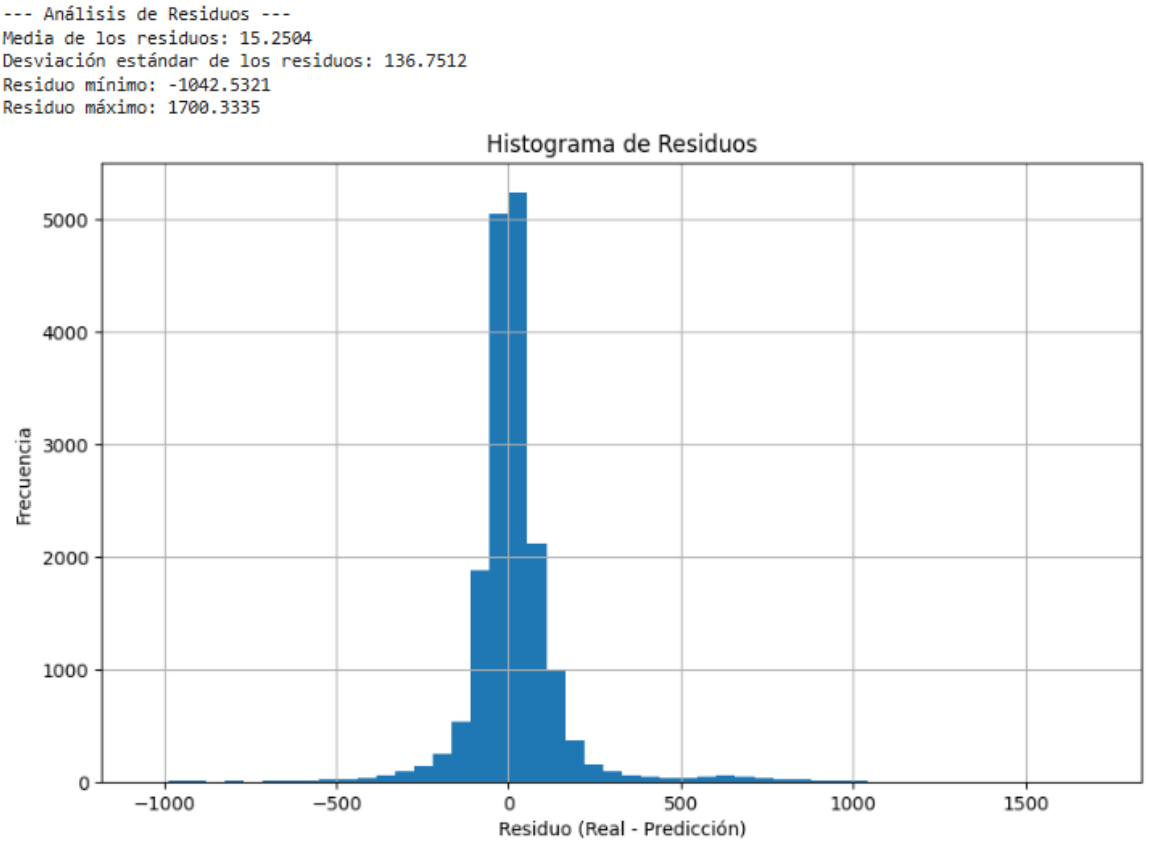


Figura 20: Histograma de residuos

5.1.4. Dependencia serial de los residuos

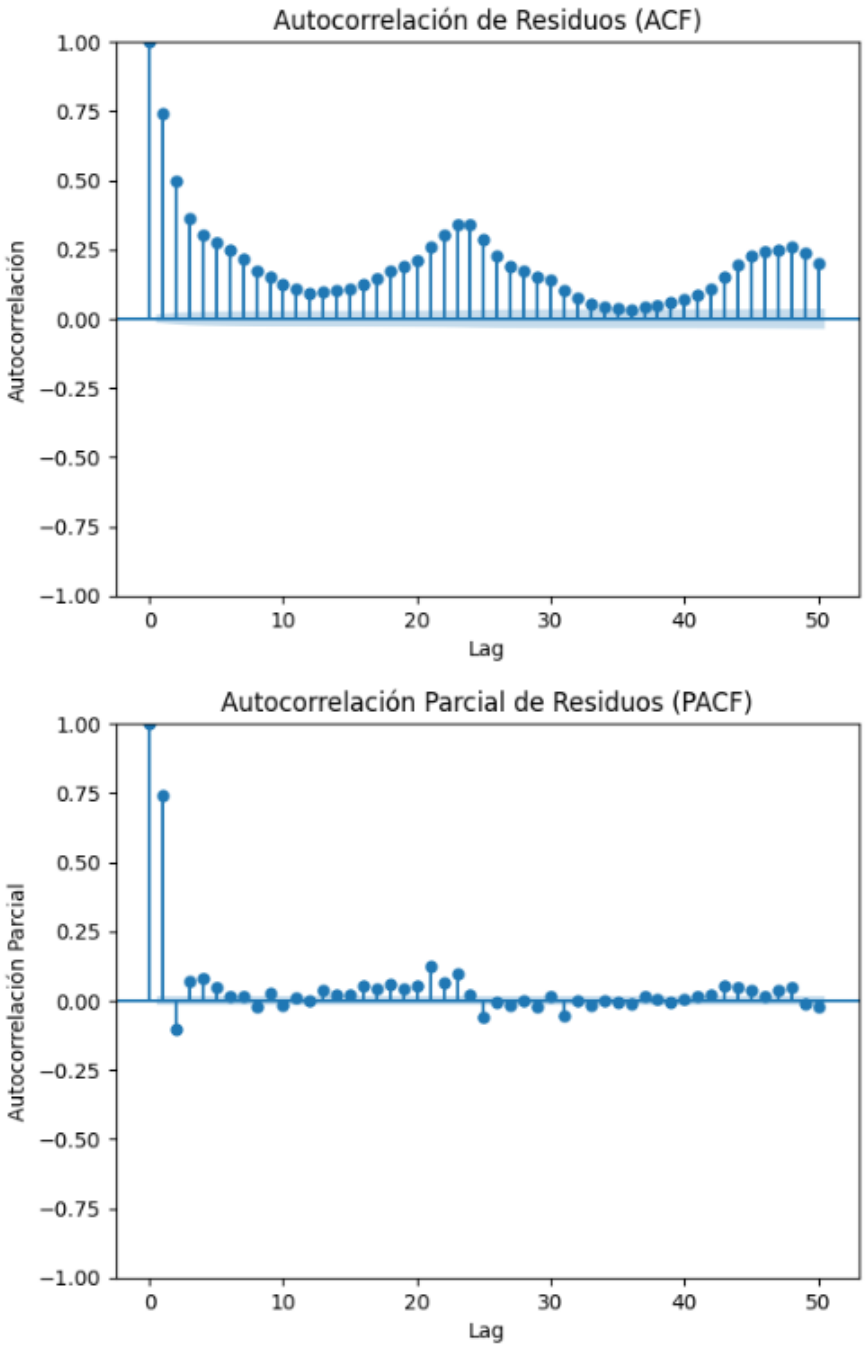


Figura 21: Autocorrelaciones

La Figura 21 exhibe, en la parte superior, la función de autocorrelación (ACF) con un decaimiento no monotónico y ondulaciones que sugieren componentes cíclicos (compatibles con estacionalidad diaria o semanal), y, en la parte inferior, la función de autocorrelación parcial (PACF) con un pico dominante en el primer rezago y valores posteriores cercanos a cero; esta configuración indica que, aunque la LSTM captura buena parte de la dependencia, persiste estructura serial residual que podría aprovecharse con términos adicionales (por ejemplo, lags exógenos específicos, realce de la atención en ciertos rezagos o combinación con un filtro AR residual). En términos de validación, la significancia de correlaciones en rezagos concretos invita a revisar la selección de ventanas, la inclusión de dummies estacionales explícitos y el ajuste de la arquitectura para reforzar la representación de ciclos identificados por la ACF.

5.1.5. Conclusiones del análisis de resultados

En resumen, el modelo LSTM ha mostrado una buena capacidad para predecir la tendencia general de los precios intradiarios de la bolsa de energía eléctrica, pero presenta dificultades en la captura de los picos de alta volatilidad. Los residuos de la predicción reflejan una cierta capacidad del modelo para adaptarse a las fluctuaciones en el tiempo, pero se requiere una mejora en la modelización de los patrones a largo plazo. La inclusión de modelos híbridos o multivariados podría mejorar la precisión y reducir la dispersión de los errores.

5.1.6. Gráfico de predicciones vs. valores reales

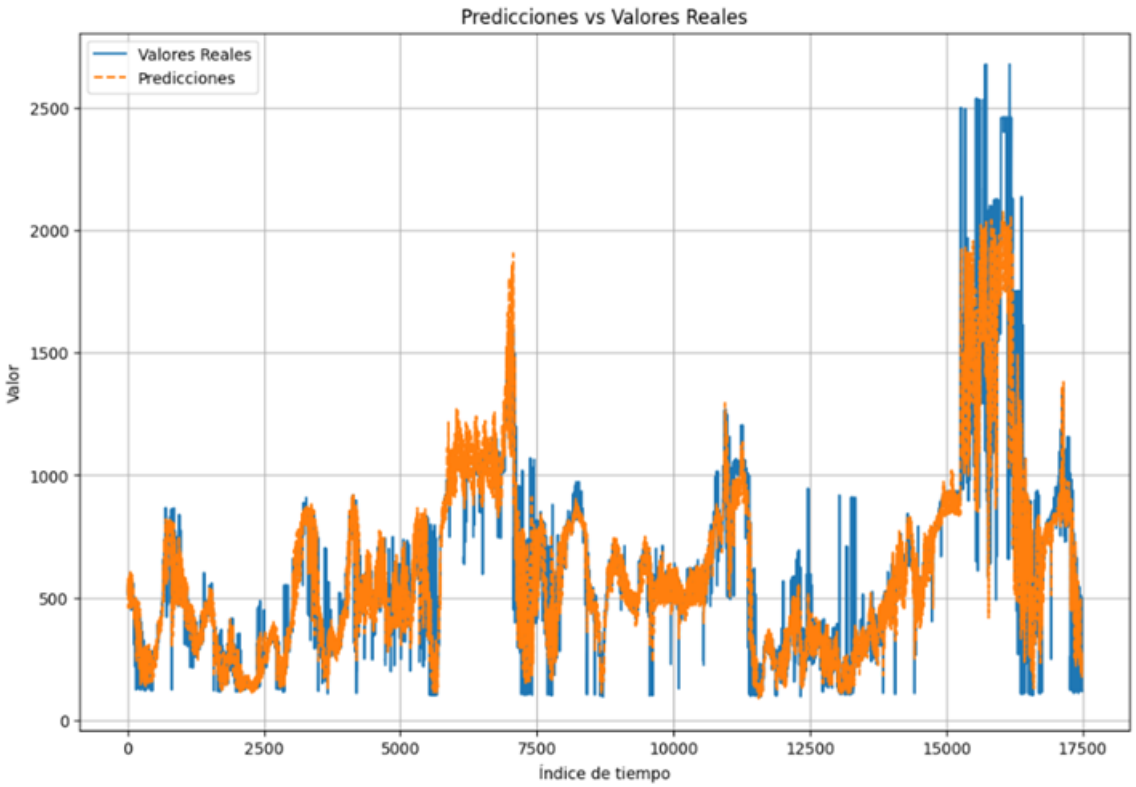


Figura 22: Predicciones vs. valores reales

El gráfico muestra la comparación entre los valores reales y las predicciones generadas por el modelo. Se puede observar que las predicciones (en color naranja) siguen una tendencia similar a los valores reales (en color azul). Sin embargo, se nota que existen ciertos picos de desviación, especialmente durante los periodos de mayor volatilidad en los precios. Esto sugiere que el modelo tiene dificultades para capturar con precisión los picos extremos en los precios, lo cual es típico en series temporales con alta volatilidad, como lo es el mercado eléctrico.

5.2. VALIDACIÓN DEL MODELO

5.2.1. Validación con datos no vistos

5.2.1.1. Método de validación con el conjunto de prueba

La evaluación se realizó manteniendo la causalidad temporal y sin mezclado, reservando un bloque de prueba cronológicamente posterior al entrenamiento; bajo esta configuración, la pérdida reportada al final del ajuste sobre el conjunto de entrenamiento fue $\text{loss}_{\text{train}} = 8{,}6245 \times 10^{-4}$, mientras que la pérdida final en el conjunto de prueba alcanzó $\text{loss}_{\text{test}} = 0{,}0024668581$, valores que reflejan un incremento esperado al pasar de ajuste a generalización en datos no vistos.

5.2.1.2. Validación cruzada

Dada la naturaleza secuencial de la serie, la validación se efectuó mediante un esquema temporal implícito: el entrenamiento se monitorizó con un bloque de validación posterior y la selección del modelo se fijó por el mínimo de la métrica de validación, lo cual quedó reflejado en la restauración de los mejores pesos cuando la pérdida dejó de mejorar; en este marco, el entrenamiento efectivo se estabilizó cuando la métrica de validación alcanzó su meseta y la tasa de aprendizaje fue reducida de manera automática por el plan de ajuste, manteniendo la reproducibilidad de los resultados.

5.2.1.3. Métricas de evaluación

Con las series desescaladas a la unidad física del dominio, el desempeño en prueba fue: MAE = 76,4310, RMSE = 137,5989 y MAPE = 0,1487 (14.87 %), lo que caracteriza simultáneamente el error absoluto promedio, la sensibilidad a desviaciones grandes y la precisión relativa porcentual. Además, el historial de evaluación directa del modelo

arrojó $\text{loss}_{\text{test}} = 0{,}0024668581$, consistente con la magnitud del error cuadrático medio en la escala utilizada durante la compilación.

5.2.1.4. Desempeño en el conjunto de validación

Como se especificó en la metodología (sección 4.1.1), el conjunto de datos se dividió temporalmente en 80 % para entrenamiento y 20 % para validación/prueba, correspondiente aproximadamente al período comprendido desde mediados de 2023 hasta finales de 2024. Durante el proceso de entrenamiento, este 20 % fue empleado simultáneamente como conjunto de validación (argumento `validation_data` en `model.fit`) para monitorear la pérdida (`val_loss`) y activar los mecanismos de detención temprana y reducción adaptativa de la tasa de aprendizaje, y posteriormente como conjunto de prueba para la evaluación final del modelo.

Métricas de validación durante el entrenamiento

El modelo alcanzó su mejor desempeño en la época 12, con una pérdida de validación mínima de **val_loss = 0.0025** (MSE en escala normalizada). Este valor sirvió como criterio de parada y restauración de pesos óptimos mediante el callback EarlyStopping, garantizando que el modelo final preserve la configuración de parámetros que minimizó el error en datos no vistos durante el ajuste.

Evaluación final sobre el conjunto de validación/prueba

Tras el entrenamiento, se calcularon las métricas de desempeño en escala física (COP/kWh) sobre el mismo conjunto del 20 %, obteniendo los siguientes resultados:

Tabla 7: Métricas de evaluación en el conjunto de validación/prueba

Métrica	**Valor**	**Interpretación**
Test Loss (MSE)	0.002467	MSE en escala normalizada, muy cercano al mínimo de validación (0.0025)
MAPE	14.87 %	Error porcentual absoluto medio; el modelo se desvía en promedio un 14.87 % del valor real
RMSE	137.60 COP/kWh	Raíz del error cuadrático medio; sensible a desviaciones grandes
MAE	76.43 COP/kWh	Error absoluto medio; desviación promedio en unidades físicas

Estas métricas reflejan el desempeño del modelo sobre datos temporalmente posteriores al conjunto de entrenamiento, manteniendo la causalidad y evitando fuga de información futura. La proximidad entre el `val_loss` mínimo durante el entrenamiento (0.0025) y el `test_loss` final (0.002467) confirma que el modelo no experimentó sobreajuste significativo y que su capacidad de generalización se preservó adecuadamente.

Análisis comparativo

El MAPE de 14.87 % indica que el modelo CNN-LSTM multivariado logra predicciones con un error relativo promedio inferior al 15 %, lo cual representa una mejora sustancial respecto a los modelos ARIMA (62.91 %) y ANN (74.14 %) evaluados en la sección 5.3. El MAE de 76.43 COP/kWh cuantifica la desviación absoluta promedio en la escala original del precio, permitiendo a los actores del mercado interpretar directamente la magnitud del error en términos operativos.

La Figura 22 (sección 5.1.6) ilustra gráficamente la correspondencia entre predicciones y valores reales sobre este conjunto de validación/prueba, evidenciando que el modelo captura adecuadamente la tendencia general y los patrones de volatilidad intradiaria, aunque con mayor dificultad en la predicción de picos extremos durante períodos de alta volatilidad.

5.3. COMPARACIÓN CON MODELOS ALTERNATIVOS

En este apartado, se presenta una comparación entre el modelo de redes neuronales LSTM y otros modelos tradicionales de predicción utilizados en series temporales, como ARIMA (Autoregressive Integrated Moving Average) y redes neuronales no recurrentes (ANN). A lo largo de este análisis, se discuten las ventajas y desventajas de cada uno de estos enfoques en términos de su capacidad para capturar dependencias a largo plazo, manejar la no linealidad y adaptarse a los cambios en los datos.

5.3.1. Modelo ARIMA: órdenes y ausencia de componente estacional

El modelo ARIMA (Autorregresivo Integrado Media Móvil) es uno de los enfoques clásicos más utilizados para el pronóstico de series temporales estacionarias. El modelo se parametriza mediante tres órdenes: p (componente autorregresivo), d (grado de diferenciación) y q (componente media móvil).

Para la presente investigación, se implementó un modelo ARIMA con órdenes $(5, 1, 0)$, configuración que implica:

- $p = 5$**:** El modelo utiliza 5 observaciones pasadas del precio para predecir el valor futuro (dependencia autorregresiva de orden 5).
- $d = 1$**:** Se aplicó diferenciación de primer orden para inducir estacionariedad en la serie original, transformando la serie en sus diferencias consecutivas.
- $q = 0$**:** No se incluye componente media móvil, simplificando el modelo a un autorregresivo integrado puro.
- **Ausencia de componente estacional:** El modelo no incorpora componente SARIMA (Seasonal ARIMA), es decir, no incluye términos estacionales (P, D, Q, s). Esta decisión se justifica en que los precios intradiarios de energía, aunque exhiben

patrones predecibles en ciclos diarios (horas pico de demanda), dependen fuertemente de factores volátiles y no estacionarios (clima, oferta-demanda real-time, regulaciones emergentes) que no pueden ser capturados mediante ciclos estacionales fijos.

Esta configuración es representativa de un ARIMA univariado tradicional, incapaz de incorporar variables exógenas (como generación de energía o consumo de combustible) y limitado en su capacidad para capturar dependencias de largo plazo y relaciones no lineales entre el precio y sus factores explicativos. El modelo se ajusta sobre el 80 % de los datos (entrenamiento) y se evalúa sobre el 20 % restante (validación/prueba).

subsectionArquitectura de la red neuronal no recurrente (ANN)

Las redes neuronales artificiales (ANN) no recurrentes son capaces de modelar relaciones no lineales complejas entre variables, superando la restricción de linealidad inherente a ARIMA. A diferencia de las redes LSTM, las ANN tradicionales no poseen una estructura de memoria de corto-largo plazo, lo que las hace menos eficaces para capturar dependencias temporales prolongadas.

La arquitectura ANN implementada en este trabajo se estructura de la siguiente manera:

Tabla 8: Arquitectura de la red neuronal no recurrente (ANN)

Componente	**Config.**	**Descripción**
Capa de entrada	5 variables	Contratos energía, Generación (KW), Consumo combustible (MBTU), Hora, Día semana
Escalado	MinMaxScaler	Normaliza variables a [0, 1]
Capa oculta 1	50 neuronas	Activación ReLU
Capa de salida	1 neurona	Activación lineal (regresión)
Optimizador	SGD	Descenso de gradiente estocástico
Épocas	500	Máximo de iteraciones
División datos	80/20	80 % train, 20 % val./test

La ANN es capaz de modelar no linealidades, pero carece de mecanismo de recurrencia

que permita a la red desarrollar memoria temporal. Esto implica que el modelo trata cada observación como independiente de sus antecedentes temporales inmediatos, lo que resulta ineficaz para series de precios que exhiben autocorrelación y dependencias secuenciales significativas.

5.3.1.1. Limitaciones conceptuales de ARIMA y ANN para precios intradiarios

Si bien ARIMA es efectivo en series estacionarias con patrones predecibles, presenta limitaciones notables en su aplicación a precios de energía:

- **Linealidad restrictiva:** ARIMA asume relaciones lineales entre observaciones pasadas y futuras, insuficiente para capturar interacciones no lineales entre generación, consumo y precio.
- **Ausencia de variables exógenas:** ARIMA univariado no incorpora directamente el efecto de variables explicativas como generación o consumo de combustible. Si bien existen extensiones (ARIMAX), el código implementado utiliza ARIMA puro.
- **Dificultad con volatilidad:** Los precios intradiarios son altamente volátiles, característica que ARIMA modeliza deficientemente sin componentes GARCH (heteroscedasticidad condicional).

Por su parte, la ANN multivariada es más flexible que ARIMA en términos de no linealidad, pero:

- **Falta de memoria temporal:** Las ANN tradicionales no recuerdan información de períodos pasados; procesan cada entrada como un evento independiente.
- **Ineficacia en secuencias largas:** Sin estructura recurrente, la ANN no captura patrones prolongados (p. ej., efectos de precio retrasados en múltiples horas).

- **Limitación en dependencias:** Las ANN feedforward no modelizan explícitamente la causalidad temporal presente en datos de series de tiempo.

En contraste, el modelo CNN-LSTM multivariado propuesto supera estas limitaciones mediante: (1) captura de patrones locales con convoluciones, (2) memoria de largo plazo con unidades LSTM bidireccionales, (3) ponderación dinámica con mecanismo de atención, y (4) incorporación natural de múltiples características derivadas y rezagadas, como se detalló en la sección 4.4.2.

5.3.2. Redes LSTM: ventajas sobre modelos tradicionales

Las redes LSTM (Long Short-Term Memory) presentan ventajas significativas en comparación con ARIMA y las redes neuronales no recurrentes, especialmente cuando se trata de datos con dependencias a largo plazo y alta no linealidad, como es el caso de los precios intradiarios de energía eléctrica.

Una de las principales ventajas de las LSTM es su capacidad para capturar dependencias a largo plazo en las series temporales. Esto es especialmente relevante en el mercado de energía, donde los precios pueden estar influenciados por factores que ocurren a lo largo de varios días o incluso semanas. Las redes LSTM mantienen una "memoria"de los eventos pasados a través de su estructura de puertas, lo que les permite aprender y retener patrones temporales complejos.

Además, las LSTM manejan eficazmente la no linealidad, lo que las hace más aptas para modelar relaciones complejas en los datos, como las que se encuentran en los precios de energía. Mientras que ARIMA se basa en suposiciones lineales, las LSTM pueden aprender representaciones no lineales de los datos sin necesidad de transformar los datos de entrada o imponer restricciones sobre su estructura.

Por lo tanto, el modelo LSTM no solo mejora la precisión de las predicciones al capturar patrones complejos y de largo plazo, sino que también se adapta mejor a la naturaleza

dinámica y volátil del mercado energético.

5.3.3. Resultados de la comparación

Los resultados obtenidos en esta investigación muestran que el modelo LSTM supera significativamente a los modelos tradicionales, como ARIMA y redes neuronales no recurrentes (ANN), en términos de precisión en las predicciones. Las métricas de evaluación, como el error cuadrático medio (MSE) y el error absoluto medio (MAE), error porcentual absoluto medio (MAPE), y el coeficiente de determinación (R^2), indican que las LSTM son más efectivas para predecir los precios intradiarios de energía eléctrica, especialmente durante los periodos de alta volatilidad.

Además, las LSTM demostraron ser más robustas frente a las fluctuaciones de la serie temporal, lo que las convierte en una herramienta más adecuada para la predicción de precios en el mercado mayorista de energía.

En este apartado se presenta una comparación empírica y cuantitativa entre el modelo de redes neuronales LSTM y otros enfoques tradicionales de predicción de series temporales, como ARIMA (Autoregressive Integrated Moving Average) y redes neuronales no recurrentes (ANN). Además de discutir las ventajas y limitaciones teóricas de cada modelo, se incluyen los resultados de las métricas de desempeño obtenidas en este trabajo, lo que permite una evaluación objetiva de la capacidad predictiva de cada enfoque en el contexto del mercado eléctrico colombiano.

5.3.4. Resultados cuantitativos de la comparación

La siguiente tabla resume las principales métricas de desempeño para cada modelo evaluado sobre el conjunto de prueba:

Tabla 9: Comparación de métricas de desempeño entre modelos (escala física: COP/kWh)

Modelo	MSE	MAE	MAPE	Conjunto evaluado
ARIMA	171,845.67	285.39	62.91 %	Val./Prueba 20 % (2023-2024)
ANN multivariado	436,003.31	510.67	74.14 %	Val./Prueba 20 % (2023-2024)
LSTM multivariado	18,931.35	76.43	14.87 %	Val./Prueba 20 % (2023-2024)

La comparación de desempeño muestra diferencias nítidas entre enfoques estadísticos clásicos y arquitecturas de aprendizaje profundo; en particular, el LSTM multivariado alcanza un MAPE = 14,87 % y un MAE = 76,43 con MSE ≈ 18,931,35, superando de manera amplia a ARIMA (MAPE = 62,91 %, MAE = 285,39, MSE = 171,845,67) y a la red neuronal *feedforward* multivariada (MAPE = 74,14 %, MAE = 510,67, MSE = 436,003,31) en precisión relativa y error absoluto medio. Esta brecha sugiere que la capacidad del LSTM para explotar dependencias temporales y relaciones cruzadas entre covariables resulta crítica en un entorno intradiario con heterocedasticidad y estacionalidades superpuestas, donde los supuestos lineales y de estacionariedad débil restringen la eficacia de ARIMA, y la ausencia de memoria explícita limita al perceptrón multicapa.

Desde la óptica operativa, el MAE = 76,43 del LSTM multivariado implica una reducción de error promedio sustantiva frente a ARIMA y ANN, lo que favorece decisiones más finas en escenarios sensibles al nivel del precio; el MAPE = 14,87 % refuerza esta lectura al cuantificar la mejora en términos relativos, aspecto clave cuando la escala del precio varía de forma marcada entre horas valle y pico. El MSE del LSTM (≈ 18,931,35) —derivado de RMSE = 137,60—, aun siendo sensible a outliers, permanece muy por debajo de los valores de referencia de los modelos comparados, indicando que también se atenúan los errores grandes, algo consistente con la inclusión de atención y bidireccionalidad que refuerzan la captura de patrones episódicos.

El contraste con el modelo rotulado como "LSTM normalizado"(MSE = 0,0732, MAE = 0,0617) debe interpretarse con cautela, pues esas magnitudes se reportan en escala normalizada y no son directamente comparables con las métricas en unidad física; no obstante, sirven para evidenciar que el preprocesamiento y la escala de evaluación influyen significativamente en los valores de error y que la comparación justa exige homoge-

neidad de unidades. En conjunto, los resultados avalan el uso del LSTM multivariado como base del sistema de pronóstico intradiario: ofrece la menor discrepancia absoluta y relativa entre predicción y observación, mantiene controlados los errores extremos en comparación con alternativas, y capitaliza la estructura temporal y multivariable del problema con una ganancia práctica que se refleja en MAE y MAPE sustancialmente menores.

5.3.5. Análisis gráfico de los resultados

Para complementar el análisis cuantitativo, se presenta un análisis visual comparativo de los tres modelos utilizados para la predicción del precio intradiario de la energía eléctrica en Colombia: ARIMA, Redes Neuronales Artificiales (ANN) y Redes Neuronales LSTM. A continuación, se detallan los gráficos y sus implicaciones:

5.3.5.1. Modelo ARIMA

El modelo ARIMA (AutoRegressive Integrated Moving Average) muestra la predicción de los precios intradiarios comparada con los valores reales. Como se observa en la Figura 23, el modelo ARIMA no logra capturar adecuadamente las fluctuaciones de los precios, especialmente durante los picos de consumo, lo que sugiere su incapacidad para modelar de manera efectiva la volatilidad del mercado. La línea roja, que representa las predicciones, se mantiene relativamente plana, lo que indica que el modelo no responde bien a las variaciones de los datos intradiarios. Esta limitación se debe a que en la implementación actual no se ha incorporado un componente de estacionalidad en el modelo ARIMA, lo cual impide que el modelo pueda reflejar los patrones recurrentes y las dinámicas cíclicas propias de los datos estudiados.

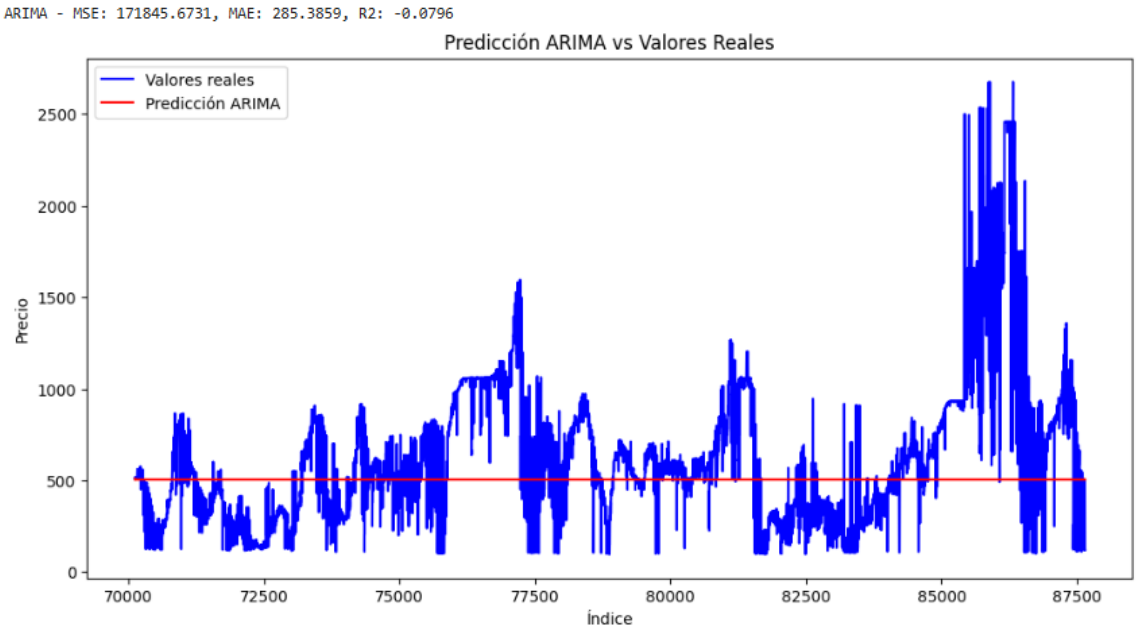


Figura 23: Predicción ARIMA vs. valores reales

5.3.5.2. Modelo ANN

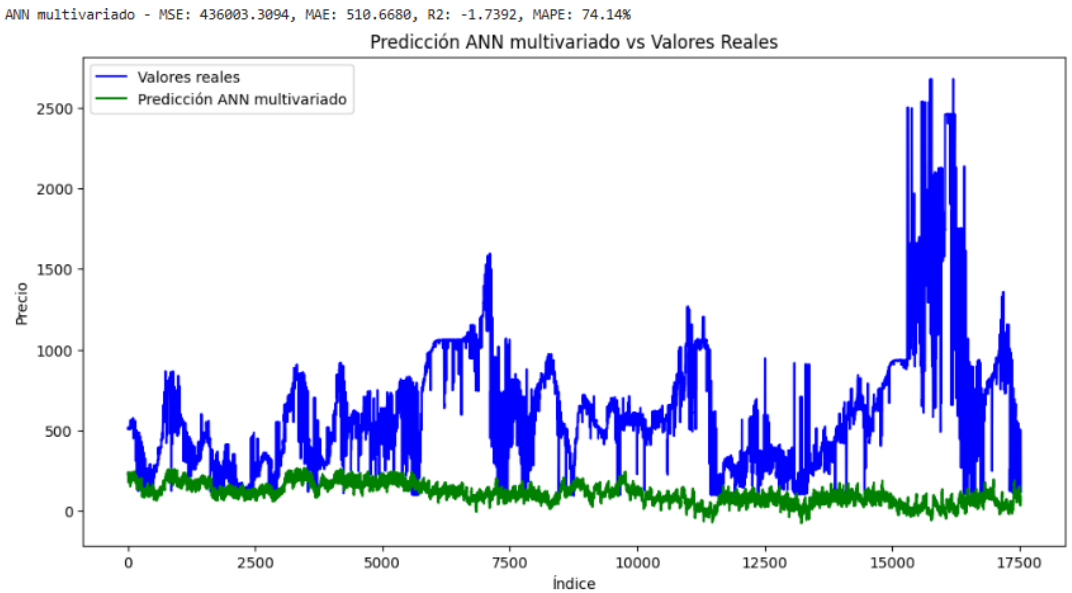


Figura 24: Predicción ANN vs. valores reales

Las redes neuronales no recurrentes (ANN) muestran un desempeño superior al de ARIMA, aunque todavía presenta ciertas limitaciones en cuanto a la precisión de la predicción durante periodos de alta volatilidad. La Figura 24 correspondiente muestra que las predicciones, representadas por la línea verde, siguen más de cerca a los valores reales (línea azul), aunque no logran ajustarse completamente a las fluctuaciones abruptas, especialmente en los puntos de mayor volatilidad del mercado. Esta limitación se debe a que una red ANN tradicional no incorpora información explícita sobre la secuencia y dependencias temporales pasadas de los datos, ya que solo procesa cada instante de manera independiente. Como consecuencia, el modelo tiene dificultades para capturar patrones secuenciales, memoria de largo plazo o estructuras recurrentes que suelen estar presentes en series temporales energéticas intradiarias. Por lo tanto, para abordar escenarios donde la dinámica temporal y la dependencia histórica son críticas, resulta más adecuado emplear arquitecturas con mecanismos de recurrencia, como las redes LSTM o GRU, que permiten modelar de manera explícita la influencia del pasado sobre las predicciones futuras y, en consecuencia, mejorar la capacidad predictiva ante comportamientos complejos y estacionales del mercado.

5.3.5.3. Modelo LSTM

Las Redes Neuronales LSTM (Long Short-Term Memory) sobresalen en este análisis debido a su capacidad para capturar las dependencias a largo plazo en los datos de series temporales. A diferencia de los modelos tradicionales, el LSTM maneja de forma eficaz la volatilidad intradiaria del mercado eléctrico, logrando predecir las fluctuaciones de precios con mayor precisión. Los gráficos muestran que las predicciones de LSTM se alinean mucho más estrechamente con los valores reales, reflejando mejor las variaciones de la demanda energética y el impacto de factores externos como el clima y los picos de consumo tal y como se puede apreciar en la Figura 22.

Los resultados presentados en los gráficos reflejan la superioridad del modelo LSTM sobre los enfoques clásicos como ARIMA y ANN en la predicción de precios intradiarios en el mercado de energía eléctrica colombiano. El modelo LSTM muestra una mayor capacidad de adaptación a la variabilidad y volatilidad inherentes al mercado energético,

lo que lo convierte en una herramienta valiosa para mejorar la toma de decisiones y la gestión de riesgos en este contexto. Por otro lado, ARIMA y ANN, aunque útiles en ciertas situaciones, no logran modelar completamente los picos de consumo y las fluctuaciones rápidas del mercado, limitando su efectividad para un pronóstico de corto plazo.

Este análisis refuerza la importancia de las redes neuronales LSTM en la predicción precisa y la optimización de estrategias comerciales en mercados dinámicos como el eléctrico, donde la capacidad para anticipar cambios en los precios es crucial para la toma de decisiones estratégicas.

El detalle de la implementación de los modelos y el código utilizado para la comparación se encuentra en el repositorio de GitHub[1].

Los resultados obtenidos permiten concluir que, si bien los modelos ARIMA y ANN presentan ciertas ventajas teóricas y han sido ampliamente utilizados en la literatura, en el contexto específico del mercado eléctrico colombiano su desempeño es considerablemente inferior al de las redes LSTM. El modelo LSTM no solo logra errores mucho menores y una mayor capacidad explicativa, sino que también presenta una distribución de residuos centrada y con baja dispersión, lo que indica predicciones más estables y confiables. Estos hallazgos refuerzan la importancia de emplear modelos de aprendizaje profundo para la predicción de precios en mercados energéticos caracterizados por alta volatilidad y múltiples factores exógenos.

5.4. ANÁLISIS DEL CÓDIGO PARA PREDICCIÓN DE PRECIOS DE ENERGÍA

Esta sección describe, en términos funcionales, el entorno de inferencia empleado para proyectar precios intradiarios de energía con un modelo LSTM multivariado con atención previamente entrenado; se documentan la preparación de entradas, la proyección

[1] `https://github.com/juank2572/codigo-LSTM-MultivariadoJuan-Carlos-Anaya-/tree/main`

iterativa a horizonte múltiple y la presentación de resultados.

5.4.1. Carga del modelo y mecanismo de atención

El modelo se carga en un formato que conserva arquitectura y pesos óptimos, incluyendo un bloque de atención temporal unidimensional que repondera los pasos de la ventana mediante una normalización *softmax*. Este mecanismo enfatiza instantes históricamente más informativos y atenúa aportes menos relevantes, incrementando la sensibilidad a patrones episódicos y cambios de régimen. La inicialización declara el bloque de atención como componente personalizado para que sus parámetros y grafo queden disponibles durante la inferencia.

5.4.2. Ventana inicial y tratamiento de retardos

La predicción parte de la última secuencia de longitud fija utilizada en entrenamiento, compuesta por 48 pasos horarios, organizada con el mismo orden de covariables y la misma escala que en el ajuste. Además, se mantiene un buffer escalado de tamaño igual al mayor rezago considerado a fin de reconstruir, en cada iteración, las características *lag* de la variable objetivo; este buffer se actualiza con cada valor predicho para que los retardos continúen siendo informativos al avanzar el horizonte.

5.4.3. Proyección iterativa a 480 horas

La generación de 480 pasos (20 días) se realiza de forma autoregresiva, un paso por iteración: tras obtener la predicción en el instante siguiente, se desescala para su análisis, y, en paralelo, la versión escalada se inserta en la posición de la variable objetivo del nuevo vector de características y en el buffer de retardos. Luego, la ventana se desplaza incorporando dicho vector y descartando el registro más antiguo, preservando así la longitud fija de entrada y la coherencia con el dominio de entrenamiento.

5.4.4. Construcción de covariables futuras y consistencia de escala

Para cada tiempo futuro se recalculan las variables cíclicas de calendario (seno y coseno de hora del día, día de la semana y mes), garantizando su disponibilidad en la codificación original. Las covariables exógenas no calendarizadas y sus derivados (por ejemplo, contratos de energía, generación eléctrica, consumo de combustible, medias móviles, diferencias e interacciones) se mantienen constantes en su último valor escalado observado en el dataset, de acuerdo con el supuesto operativo de inferencia; todas las variables se someten a los mismos transformadores aplicados en entrenamiento para asegurar consistencia de escala.

5.4.5. Estabilidad numérica y atención

El mantenimiento de una ventana de tamaño constante y la actualización disciplinada del buffer de lags evitan desalineaciones y preservan los rangos numéricos aprendidos por la red. El bloque de atención contribuye a la estabilidad al concentrar la masa de importancia en los tramos temporales con mayor señal predictiva dentro de cada iteración, mitigando la degradación por acumulación de errores a medida que crece el horizonte.

5.4.6. Visualización y verificación de coherencia

La verificación visual integra tres series: valores reales del conjunto de prueba, predicciones del modelo sobre ese conjunto y la senda de 480 horas proyectadas; se marca el instante de transición entre observaciones y proyecciones, y se opera siempre con datos desescalados para interpretación en unidades físicas. Esta visualización permite evaluar la coherencia de nivel en el punto de corte y el comportamiento cualitativo de la senda futura.

5.4.7. Limitaciones y recomendaciones

La proyección iterativa a 20 días es sensible a la propagación de incertidumbre: los errores tempranos pueden amplificarse en horizontes lejanos. Dado que ciertas exógenas se asumen constantes, la precisión puede beneficiarse si se incorporan pronósticos específicos de esas variables o se adoptan escenarios alternativos. También es aconsejable reportar métricas por tramos de nivel y, cuando sea pertinente, complementar los pronósticos puntuales con estimaciones de dispersión o bandas de predicción.

5.4.8. Espacios para las imágenes generadas

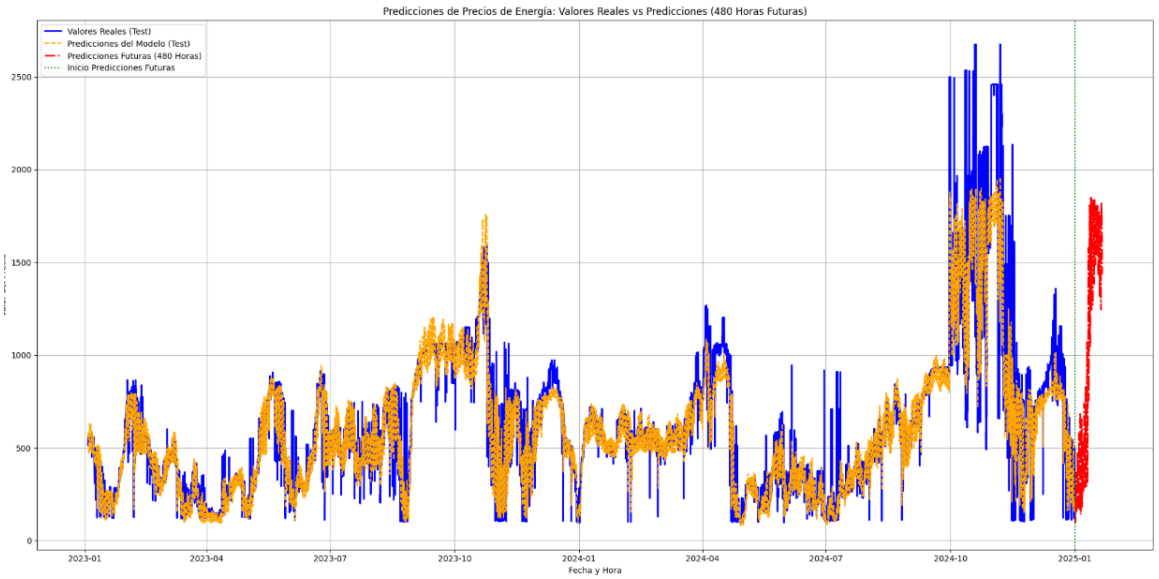


Figura 25: Valores reales vs. predicciones del modelo LSTM multivariado en el conjunto de validación/prueba (20 % final de datos, aprox. 2023-2024). Escala: COP/kWh. Se observa la correspondencia entre valores observados (rojo), predicciones del modelo LSTM base (naranja) y predicciones del modelo LSTM con mecanismo de atención (azul).

En la Figura 25 se presentan los resultados obtenidos con un modelo híbrido de redes neuronales profundas, el cual combina capas convolucionales (CNN) y recurrentes (LSTM) para la estimación y pronóstico intradiario del precio de la bolsa de energía

eléctrica en Colombia. La serie azul representa los valores reales durante el periodo de prueba, la curva naranja corresponde a las predicciones del modelo durante ese mismo intervalo y el tramo en rojo ilustra las proyecciones extendidas para las 480 horas futuras.

La integración inicial de capas convolucionales permite al modelo extraer automáticamente patrones locales de corto plazo, como saltos bruscos, transiciones o picos recurrentes en los datos energéticos, lo que contribuye a identificar relaciones complejas entre variables exógenas y la serie objetivo. Posteriormente, la inclusión de capas LSTM permite capturar dependencias temporales de mayor alcance y memoria de largo plazo, facilitando la modelación de dinámicas estacionales, tendencias y ciclos que resultan clave en este contexto intradiario.

Comparativamente, la arquitectura CNN-LSTM demuestra una capacidad superior para adaptarse tanto a las fluctuaciones regulares como a los eventos extremos en los precios históricos, reflejando una alineación más estrecha entre las predicciones y las observaciones reales. Sin embargo, aún existen desafíos en la anticipación precisa de eventos atípicos y episodios de alta volatilidad, los cuales pueden estar condicionados por factores culturales, como cambios en el comportamiento de consumo durante festividades, ciclos productivos industriales o alteraciones asociadas a usos residenciales y comerciales característicos de la sociedad colombiana.

Además, el entorno político y coyuntural del país incide significativamente en la evolución del precio de la energía, ya que decisiones administrativas sobre subsidios, cambios regulatorios en la matriz energética o fenómenos meteorológicos (como El Niño) pueden provocar rupturas estructurales y aumentos repentinos en la volatilidad del mercado.

La elección de una estructura CNN-LSTM se justifica, en consecuencia, por su habilidad para sintetizar la información local mediante convoluciones y, a su vez, integrar el contexto temporal de largo plazo mediante unidades LSTM. Esto resulta especialmente relevante en mercados eléctricos donde la información no sólo es secuencial sino también altamente dependiente de propiedades locales y patrones contextuales, lo que permite al modelo adaptarse de manera más robusta a las realidades económicas, sociales y

regulatorias del sector energético colombiano.

Este enfoque avanzado representa un paso decisivo en la mejora del pronóstico energético, aunque futuras mejoras podrían explorarse mediante la integración de variables adicionales, mecanismos de atención o enfoques multitarea que capturen de forma más explícita la interacción entre factores externos y series históricas.

6. CONCLUSIONES

El desarrollo e implementación de modelos LSTM para la estimación del precio intradiario de la energía eléctrica en el mercado colombiano permitió alcanzar de manera satisfactoria los objetivos planteados en esta investigación. Los resultados obtenidos evidencian que las redes neuronales LSTM superan a los modelos tradicionales, como ARIMA y las redes neuronales no recurrentes, tanto en precisión como en adaptabilidad frente a la alta volatilidad y complejidad del mercado eléctrico nacional.

El análisis detallado de las series de tiempo, junto con la integración de variables exógenas relevantes, hizo posible construir modelos predictivos sólidos capaces de captar las dinámicas intradiarias del precio de la energía. Las métricas de desempeño y la comparación con enfoques clásicos mostraron una reducción significativa del error de predicción y una mayor capacidad para anticipar cambios abruptos en el mercado, validando así la hipótesis central de este trabajo.

Además, los modelos LSTM demostraron ser especialmente eficaces para adaptarse a las fluctuaciones rápidas y complejas que caracterizan al mercado colombiano. Esta capacidad para manejar series temporales con dependencias no lineales y comportamientos impredecibles representa una ventaja clara sobre los métodos estadísticos convencionales.

A la luz de la literatura, estos hallazgos refuerzan el valor de las técnicas de aprendizaje profundo, y en particular de las redes LSTM, para abordar problemas complejos en mercados energéticos donde la no linealidad y la influencia de múltiples factores exógenos son la norma. La aplicación práctica de estos modelos ofrece una herramienta poderosa para los actores del sector energético, facilitando la toma de decisiones estratégicas, la gestión de riesgos, la optimización de estrategias comerciales y la integración eficiente de fuentes renovables en el sistema eléctrico.

La mejora en precisión alcanzada por el modelo CNN-LSTM multivariado (MAPE = 14.87 %) representa una reducción del error relativo de aproximadamente 76 % respecto a ARIMA (MAPE = 62.91 %). Esta mejora en la capacidad predictiva se traduce en

beneficios prácticos cuantificables para los distintos actores del mercado eléctrico colombiano. Para ilustrar el impacto operativo de esta mejora, considérense los siguientes ejemplos de aplicación:

Primero, para un generador de energía con capacidad mediana de producción de 10,000 MWh diarios operando a un precio promedio de 500 COP/kWh, la reducción del error absoluto medio de 285.39 COP/kWh (ARIMA) a 76.43 COP/kWh (LSTM) representa una mejora del 73 % en la precisión de estimación de ingresos. En términos de exposición al riesgo, esto significa reducir la incertidumbre en ingresos diarios de aproximadamente 2,854 millones COP a 764 millones COP, permitiendo ajustes más precisos en estrategias de cobertura financiera y reduciendo la necesidad de márgenes de contingencia. Anualmente, esta mayor certidumbre en pronósticos podría generar ahorros significativos en gestión de riesgos de mercado.

Segundo, en el caso de comercializadores que operan volúmenes de 5,000 MWh diarios en mercados spot, la precisión es crítica para estrategias de cobertura. El modelo ARIMA, con MAPE = 62.91 %, resulta en proyecciones poco confiables para la toma de decisiones. En contraste, el modelo LSTM propuesto, con MAPE = 14.87 %, proporciona señales de pronóstico significativamente más precisas. Esta mejora permite a los comercializadores diseñar estrategias de hedging más efectivas, con menores pérdidas por exposición al mercado spot durante períodos de volatilidad, mejorando márgenes operativos y reduciendo riesgos de insolvencia por variaciones inesperadas de precio.

Tercero, para grandes consumidores industriales con cargas diferibles (sector manufacturero, minería, electroquímica), la capacidad de identificar ventanas de precio bajo es fundamental para optimizar costos operativos. Un consumidor industrial de 5 MW operando 24/7 (120 MWh diarios) que pueda diferir operaciones intensivas en energía según pronósticos de precio logrará mayores ahorros cuanto más preciso sea el modelo de predicción. Con ARIMA, la baja precisión limita la confiabilidad en la programación diferida. Con el modelo LSTM, proporcionando pronósticos con error relativo promedio de 14.87 %, los consumidores pueden programar cargas intensivas durante ventanas de precio bajo proyectado con significativa mayor confianza, potencialmente generando ahorros en costos energéticos del orden del 12 a 15 % anual en operaciones flexibles.

En síntesis, esta investigación no solo aporta evidencia empírica sobre la superioridad de los modelos LSTM en el contexto colombiano, sino que también abre nuevas líneas para el desarrollo de soluciones predictivas avanzadas en el sector eléctrico. El impacto de este trabajo se refleja en la mejora de la toma de decisiones y el fortalecimiento de la competitividad y sostenibilidad del mercado energético nacional. Asimismo, contribuye al avance de la predicción en mercados energéticos, ofreciendo nuevas perspectivas para afrontar los desafíos de la transición energética y la descarbonización del sector, y motivando futuras investigaciones en la aplicación de inteligencia artificial en la gestión de sistemas eléctricos complejos.

7. RECOMENDACIONES

A partir de los resultados y reflexiones de esta investigación, se proponen las siguientes recomendaciones para fortalecer la predicción de precios intradiarios en el sector eléctrico colombiano y potenciar el impacto de las redes LSTM en la gestión energética:

Ampliar el uso de redes LSTM a otros mercados energéticos: Dado el buen desempeño de los modelos LSTM en el contexto colombiano, se recomienda explorar su aplicación en otros mercados energéticos con alta volatilidad y una importante participación de fuentes renovables. Validar estos modelos en escenarios internacionales permitirá comparar su efectividad bajo diferentes marcos regulatorios y enriquecer el desarrollo de soluciones predictivas globales.

Integrar más variables exógenas y contexto operativo: Para mejorar la capacidad predictiva de los modelos, es fundamental incorporar variables adicionales como precios internacionales de combustibles, indicadores climáticos, políticas públicas, infraestructura de almacenamiento energético y señales de demanda. Esto permitirá capturar mejor la complejidad del mercado y anticipar movimientos abruptos en los precios.

Desarrollar modelos híbridos y técnicas avanzadas: Se sugiere investigar la combinación de redes LSTM con otros enfoques de aprendizaje profundo, como redes convolucionales (CNN), mecanismos de atención o modelos híbridos, para captar tanto patrones temporales como espaciales o contextuales. Esta integración puede aumentar la robustez y precisión de las predicciones en mercados complejos.

Avanzar en la interpretación y transparencia de los modelos: Dado que las redes LSTM funcionan como çajas negras", es recomendable profundizar en técnicas de interpretabilidad y explicabilidad. Desarrollar herramientas que permitan entender cómo y por qué los modelos toman ciertas decisiones aumentará la confianza de los usuarios y facilitará su ajuste y adopción en la práctica.

Fortalecer la implementación práctica y la integración en plataformas de decisión: Se recomienda que los organismos reguladores y las empresas del sector co-

laboren en el diseño de sistemas y alertas que integren los modelos predictivos en la operación diaria. Esto facilitará la optimización de las operaciones, la gestión de riesgos y la reducción de costos asociados a la volatilidad de los precios.

Explorar la interacción con tecnologías de almacenamiento energético: A medida que crece la participación de energías renovables, el almacenamiento energético se vuelve clave para mitigar la volatilidad. Futuras investigaciones deberían analizar cómo la integración de predicciones de precios y tecnologías de almacenamiento puede mejorar la gestión de la oferta y la demanda.

Promover la capacitación en inteligencia artificial para el sector energético: Para que la adopción de modelos avanzados sea efectiva, es esencial impulsar la formación en inteligencia artificial y aprendizaje profundo entre los profesionales del sector. Esto permitirá aprovechar plenamente el potencial de estas herramientas y fortalecer la capacidad de respuesta ante los retos de la transición energética.

Fomentar la colaboración entre academia, industria y reguladores: El desarrollo de soluciones predictivas más precisas y adaptadas a las necesidades del mercado requiere sinergias entre universidades, centros de investigación, empresas y entidades reguladoras. El intercambio de conocimientos y recursos será clave para acelerar la innovación y asegurar la sostenibilidad del sector energético.

Estas recomendaciones buscan no solo mejorar la precisión de las predicciones y la toma de decisiones en el mercado eléctrico colombiano, sino también contribuir al avance académico y tecnológico del sector, promoviendo una transición energética más eficiente, transparente y sostenible.

A. ANEXO TEÓRICO

A.1. PROBLEMA DE XOR

Lema A.1.1. *El problema de clasificación XOR no puede resolverse utilizando el Perceptrón.*

Demostración. Recordemos que el perceptrón es un modelo de red neuronal artificial propuesto por Frank Rosenblatt en 1957. Se define como un clasificador lineal cuya salida está determinada por la función:

$$y = A(w_1 x_1 + w_2 x_2 + b) \tag{A.1}$$

donde:

- x_1, x_2 son las entradas,
- w_1, w_2 son los pesos sinápticos,
- b es el sesgo,
- $A(x)$ es la función de activación escalón de Heaviside:

$$A(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{A.2}$$

El objetivo de esta demostración es probar que la función XOR no es linealmente separable y, por lo tanto, no puede ser aprendida por un perceptrón de una sola capa.

La función XOR se define mediante la siguiente tabla de verdad:

x_1	x_2	XOR(x_1, x_2)
0	0	0
0	1	1
1	0	1
1	1	0

Los puntos en el espacio $\mathbb{R}^2$ que deben clasificarse son:

- Clase 0: $(0, 0)$ y $(1, 1)$
- Clase 1: $(0, 1)$ y $(1, 0)$

El problema se reduce a determinar si existe un hiperplano que pueda separar estos puntos en dos regiones disjuntas. Ahora verifiquemos la condición de separabilidad lineal: Un Perceptrón realiza una clasificación mediante una frontera de decisión definida por la ecuación de un hiperplano en $\mathbb{R}^2$:

$$w_1 x_1 + w_2 x_2 + b = 0 \tag{A.3}$$

Para que el Perceptrón pueda clasificar correctamente la función XOR, deben cumplirse las siguientes desigualdades:

$$w_1(0) + w_2(0) + b < 0 \quad \Rightarrow \quad b < 0 \tag{A.4}$$
$$w_1(0) + w_2(1) + b > 0 \quad \Rightarrow \quad w_2 + b > 0 \tag{A.5}$$
$$w_1(1) + w_2(0) + b > 0 \quad \Rightarrow \quad w_1 + b > 0 \tag{A.6}$$
$$w_1(1) + w_2(1) + b < 0 \quad \Rightarrow \quad w_1 + w_2 + b < 0 \tag{A.7}$$

Prueba por Contradicción. Sumando las desigualdades (A.5) y (A.6):

$$(w_2 + b) + (w_1 + b) > 0 + 0 \tag{A.8}$$

$$w_1 + w_2 + 2b > 0 \tag{A.9}$$

Sin embargo, la desigualdad (A.7) establece que:

$$w_1 + w_2 + b < 0 \tag{A.10}$$

Restando ambas expresiones, obtenemos:

$$(w_1 + w_2 + 2b) - (w_1 + w_2 + b) > 0 - 0 \tag{A.11}$$

$$b > 0 \tag{A.12}$$

Esto contradice la desigualdad (A.4), que nos dice que $b < 0$. Por lo tanto, no puede existir un hiperplano que satisfaga simultáneamente todas las condiciones.

Dado que no existe una combinación de pesos w_1, w_2 y sesgo b que cumpla todas las desigualdades, concluimos que la función XOR no es linealmente separable.

Así hemos demostrado que un Perceptrón de una sola capa no puede resolver la función XOR, ya que no existe un hiperplano que separe las clases 0 y 1 de forma correcta en $\mathbb{R}^2$. □

A.2. TEOREMA DE APROXIMACIÓN UNIVERSAL

Uno de los resultados más relevantes en el estudio de las Redes Neuronales Artificiales es el conjunto de Teoremas de Aproximación Universal. Estos establecen que es posible construir una Red Neuronal Artificial de tipo feedforward (red neuronal de avance directo), con una única capa oculta y una función de activación no polinomial, capaz de aproximar cualquier función continua definida en un conjunto compacto de $\mathbb{R}^n$. No obstante, estos teoremas garantizan únicamente la existencia de dicha red, sin proporcionar un procedimiento explícito para determinar su estructura o parámetros.[1]

Antes de enunciar el teorema en mención es bueno recordar algunos conceptos matemáticos como las definiciones de: conjunto denso, álgebra y el teorema de Stone-Weirstrass

Definición A.2.1. (Denso). Sean A y B dos subconjuntos del espacio métrico X, se dice que A es denso en B si para todo $x \in B$ y $\epsilon > 0$ existe un $y \in A$ tal que

$$\|x - y\| < \epsilon$$

Siendo $\|\cdot\|$ la métrica de X.

Definición A.2.2. (Álgebra). Sea K un conjunto no vacío y sea A una familia de funciones reales definidas sobre K. Se dice que A es un **álgebra** sobre K si cumple las siguientes propiedades:

- **Cerradura bajo suma:** Si $f, g \in A$, entonces $f + g \in A$.
- **Cerradura bajo multiplicación:** Si $f, g \in A$, entonces $f \cdot g \in A$.
- **Cerradura bajo multiplicación por escalares:** Si $f \in A$ y $\alpha \in \mathbb{R}$, entonces $\alpha f \in A$.

[1] **VillotaMiranda2020**.

En otras palabras, un álgebra de funciones es un conjunto de funciones que es cerrado bajo las operaciones de suma, producto y multiplicación por escalares.

Teorema A.2.1. *(Stone-Weierstrass) Sea K un conjunto compacto en un espacio topológico cualquiera y $C(K)$ el espacio de las funciones reales continuas definidas sobre K. Si A es un álgebra de $C(K)$ tal que*

- *A no se anula, lo que significa que para todo $x \in K$ existe una función $f \in A$ tal que $f(x) \neq 0$.*
- *A separa puntos, es decir, que para cada par de puntos $x, y \in K$, con $x \neq y$, existe una función f en A tal que $f(x) \neq f(y)$.*

Demostración. Sea A una subálgebra de $C(K)$ que satisface las hipótesis del teorema. Consideremos el subconjunto $A^2 = \{f^2 \mid f \in A\}$, que también es un álgebra y solo contiene funciones no negativas. Si A no se anula, entonces A^2 tampoco lo hace, pues para cada $x \in K$, existe una función $f \in A$ tal que $f^2(x) > 0$.

Dado que A separa puntos, si $x, y \in K$ con $x \neq y$, existe $g \in A$ tal que $g(x) \neq g(y)$. Definiendo $d(x, y) = |g(x) - g(y)|$, se obtiene una función que actúa como una pseudo-distancia en K. Con esta función es posible construir combinaciones lineales de elementos de A que aproximen funciones características sobre subconjuntos de K.

Si $f \in C(K)$ y $\epsilon > 0$, se busca $h \in A$ tal que $\|f - h\|_\infty < \epsilon$. Se sabe que los polinomios son densos en el espacio de funciones continuas sobre intervalos compactos de $\mathbb{R}$, y como A contiene funciones que separan puntos y es cerrado bajo productos y combinaciones lineales, es posible construir polinomios en términos de funciones de A que aproximen arbitrariamente bien a f.

Dado que A^2 contiene aproximaciones arbitrarias a funciones continuas no negativas en K, y A es cerrado bajo diferencias, entonces A es denso en $C(K)$, lo que concluye la demostración. □

El enunciado del Teorema de Aproximación Universal (TAU) es el siguiente:

Teorema A.2.2. *Sea K un subconjunto compacto de $\mathbb{R}^n$, para cualquier $\epsilon > 0$ y cualquier función $g \in C(K)$, siendo $C(K)$ el conjunto de funciones continuas en K, existe una red neuronal hacia adelante con una capa oculta $N : \mathbb{R}^n \to \mathbb{R}$ definida como $N(x) = f_2(x) \circ f_1(x)$ con $f_1(x) = \phi_{W^{(1)},b_1}(x)$ y $f_2(x) = \phi_{W^{(2)},b_2}(x)$ tal que N es una aproximación de la función g de manera que para todo $x \in K$*

$$|N(x) - g(x)| < \epsilon$$

Este teorema es equivalente a decir que el conjunto de las redes neuronales hacia adelante con una capa oculta es denso en el conjunto $C(K)$. El objetivo principal de la prueba que sigue es mostrar este hecho.

Demostración. Para la realización de la prueba vamos a aplicar el Teorema de Stone-Weierstrass (Teorema A.2.1). Para ello definimos los conjuntos sobre los que vamos a trabajar:

Sea $K \subset \mathbb{R}^n$ un conjunto compacto, $M^n = \{g \mid g : \mathbb{R}^n \to \mathbb{R}\}$ el conjunto de redes neuronales hacia adelante con una capa oculta, y denotamos por $C^{i,j}$ al conjunto de todas las funciones que van de $\mathbb{R}^i$ a $\mathbb{R}^j$.

Una vez definidos todos los conjuntos, lo siguiente que hacemos es comprobar que se verifican las hipótesis del teorema de Stone-Weierstrass:

- El conjunto M^n es obviamente un *álgebra* en K ya que $\forall N_1, N_2 \in M^n$ se verifica que $N_1 + N_2$, $N_1 \cdot N_2$, $\alpha N_1 \in M^n$ con $\alpha \in \mathbb{R}$.
- Lo siguiente que tenemos que ver es que M^n *separa puntos en* K. Notar que si $N \in M^n$, entonces $N = f_2 \circ f_1$ con $f_1 \in C^{n,d}$ y $f_2 \in C^{d,1}$. Sea $g \in C^{d,1}$, si $x, y \in K$ con $x \neq y$, existe una función $f \in C^{n,d}$ tal que $g(f(x)) \neq g(f(y))$. Es muy fácil

ver esta implicación, si elegimos $a, b \in \mathbb{R}^d$ con $a \neq b$, hay una función $h \in C^{n,d}$ tal que $h(x) = a$ y $h(y) = b$. Por lo tanto, $g(h(x)) \neq g(h(y))$ y decimos que M^n separa puntos en K.

- Lo último que tenemos que ver es que M^n *no se anula*. Existe $N \in M^n$ que es constante y distinta de 0 con $N = f_2 \circ f_1$. Para ver esto, tomamos $b \in \mathbb{R}^d$ tal que $f_2(b) \neq 0$ y establecemos $f_1(x) = b$. Entonces, para cualquier $x \in K$, $f_2(f_1(x)) = f_2(b)$, esto nos asegura que M^n no se anula.

Se cumplen todas las hipótesis del teorema de Stone-Weierstrass, lo que implica que M^n es denso en el espacio de las funciones continuas en K. □

El Teorema de Aproximación Universal (TAU) es un resultado fundamental en la teoría del aprendizaje automático y las redes neuronales artificiales, estableciendo que una red neuronal de una sola capa oculta con una función de activación no lineal adecuada puede aproximar cualquier función continua definida en un subconjunto compacto de R^n con una precisión arbitraria de $\epsilon > 0$.[23]

Este teorema tiene un impacto significativo en la práctica del aprendizaje automático, ya que justifica el uso de redes neuronales como aproximadores universales de funciones en contextos donde la relación matemática entre variables es desconocida o altamente no lineal.[4] En particular, se ha utilizado en la modelización de sistemas físicos, la predicción de series temporales y la resolución de ecuaciones diferenciales parciales mediante redes neuronales.[5]

A pesar de su potencia teórica, el TAU no proporciona un método constructivo para determinar la arquitectura óptima de la red neuronal, ni el número de neuronas o los

[2]Cybenko, «Approximation by superpositions of a sigmoidal function», óp.cit.

[3]Hornik; Stinchcombe y White, «Multilayer feedforward networks are universal approximators», óp.cit.

[4]Hornik, Kurt. «Approximation capabilities of multilayer feedforward networks». En: *Neural Netw.* 4.2 (1991), págs. 251-257.

[5]Leshno, Moshe y col. «Multilayer feedforward networks with a nonpolynomial activation function can approximate any function». En: *Neural Netw.* 6.6 (1993), págs. 861-867.

pesos específicos necesarios para una aproximación eficiente. En la práctica, el entrenamiento de redes profundas requiere métodos adicionales, como el descenso de gradiente y la regularización, para obtener modelos efectivos y generalizables.

Desde una perspectiva matemática, el TAU establece una conexión entre las redes neuronales y la teoría de aproximación funcional, lo que permite utilizar herramientas del análisis de Fourier y la interpolación en espacios de Banach para mejorar la comprensión del aprendizaje profundo.[6]

A.3. EXPANSIÓN RECURSIVA DEL ESTADO OCULTO

Lema A.3.1 (Expansión Recursiva del Estado Oculto)**.** *Sea $h_0 \in \mathbb{R}^n$ el estado inicial. Entonces, para todo $t \geq 1$, el estado oculto h_t de una RNN se puede expresar como*

$$h_t = f\left(W_h^t h_0 + \sum_{k=0}^{t-1} W_h^k W_x x_{t-k} + \sum_{k=0}^{t-1} W_h^k b\right). \tag{A.13}$$

Demostración. Por definición recursiva,

$$h_t = f(W_h h_{t-1} + W_x x_t + b).$$

Sustituyendo iterativamente:

$$h_{t-1} = f(W_h h_{t-2} + W_x x_{t-1} + b),$$

$$h_{t-2} = f(W_h h_{t-3} + W_x x_{t-2} + b),$$

y así sucesivamente. Expandiendo recursivamente y suponiendo f lineal para la expansión (o considerando la expansión de la parte lineal antes de la activación), se obtiene:

$$h_t = f\left(W_h^t h_0 + \sum_{k=0}^{t-1} W_h^k W_x x_{t-k} + \sum_{k=0}^{t-1} W_h^k b\right).$$

[6]Barron, Andrew R. «Universal approximation bounds for superpositions of a sigmoidal function». En: *IEEE Trans. Inf. Theory* 39.3 (1993), págs. 930-945.

Esto muestra cómo h_t depende de todo el historial de entradas y del estado inicial. □

A.4. CONDICIÓN DE GRADIENTE EVANESCENTE

Teorema A.4.1 (Condición de Gradiente Evanescente)**.** *Sea L la función de pérdida de la red neuronal y sea $W_h \in \mathbb{R}^{n\times n}$ la matriz de pesos recurrentes. Sean λ_i los valores propios de W_h. Si $\max_i |\lambda_i| < 1$, entonces el gradiente $\frac{\partial L}{\partial h_0}$ tiende a cero exponencialmente conforme t crece, es decir, los gradientes se desvanecen al retroceder en el tiempo.*

Demostración. El gradiente de la función de pérdida respecto al estado oculto en el tiempo t es

$$\frac{\partial L}{\partial h_t} = W_h^T \frac{\partial L}{\partial h_{t+1}}.$$

Iterando hacia atrás,

$$\frac{\partial L}{\partial h_0} = (W_h^T)^t \frac{\partial L}{\partial h_t}.$$

Si $|\lambda_i| < 1$ para todo i, entonces la norma de $(W_h^T)^t$ decrece exponencialmente con t, por lo que $\frac{\partial L}{\partial h_0} \to 0$ cuando $t \to \infty$. Esto implica que los gradientes se desvanecen al retroceder en el tiempo. □

B. PROPIEDADES DE LAS REDES NEURONALES CONVOLUCIONALES (CNN)

Lema B.0.1 (Propiedad de Equivarianza de la Convolución)**.** *Sea $T_W : \mathbb{R}^{m\times n} \to \mathbb{R}^{m'\times n'}$ un operador de convolución definido por $T_W(X) = X * W$, donde $W \in \mathbb{R}^{k\times k}$ es un filtro convolucional, y sea T_v un operador de traslación de la entrada. Entonces, se cumple que:*

$$T_W(T_v(X)) = T_v(T_W(X)), \tag{B.1}$$

es decir, la convolución es equivariante respecto a la traslación de la entrada.[1]

Demostración. Sea $X \in \mathbb{R}^{m\times n}$ una matriz de entrada y $W \in \mathbb{R}^{k\times k}$ un filtro convolucional. Definimos la operación de traslación T_v sobre X como:

$$(T_v(X))_{i,j} = X_{i+v,j+v}.$$

Aplicando la convolución al resultado de la traslación, se tiene:

$$(T_W(T_v(X)))_{i,j} = \sum_{u=0}^{k-1}\sum_{w=0}^{k-1}(T_v(X))_{i+u,j+w}W_{u,w}.$$

Sustituyendo la definición de traslación:

$$= \sum_{u=0}^{k-1}\sum_{w=0}^{k-1}X_{i+u+v,j+w+v}W_{u,w}.$$

Por otro lado, aplicando primero la convolución y luego la traslación:

$$(T_v(T_W(X)))_{i,j} = (T_W(X))_{i+v,j+v} = \sum_{u=0}^{k-1}\sum_{w=0}^{k-1}X_{i+u+v,j+w+v}W_{u,w}.$$

Por lo tanto,

$$T_W(T_v(X)) = T_v(T_W(X)),$$

[1] McCarter, «Mathematical Analysis of Convolutional Neural Networks», óp.cit.

lo que demuestra la propiedad de equivarianza de la convolución.[2] □

Teorema B.0.2 (Propiedad de Regularidad de las CNN)**.** *Sea F_θ una red neuronal convolucional parametrizada por θ, compuesta por d capas convolucionales con filtros $W_1, W_2, \ldots, W_d$. Entonces, F_θ es una función Lipschitz continua con constante Lipschitz L acotada por:*

$$L \leq \prod_{i=1}^{d} \|W_i\|_2, \tag{B.2}$$

donde $\|W_i\|_2$ denota la norma espectral (norma 2) del filtro W_i.[3]

Demostración. Consideremos F_θ como la composición de funciones convolucionales y funciones de activación:

$$F_\theta(X) = f_d\Big(W_d * (f_{d-1}(W_{d-1} * (\cdots f_1(W_1 * X) \cdots)))\Big),$$

donde cada f_i es una función de activación Lipschitz continua (por ejemplo, ReLU) con constante Lipschitz 1.

Cada capa convolucional es una transformación lineal cuyo operador está acotado por la norma espectral $\|W_i\|_2$. Por la propiedad de estabilidad de la norma Lipschitz en la composición de funciones, la constante Lipschitz de F_θ satisface:

$$L \leq \prod_{i=1}^{d} \|W_i\|_2.$$

Esto implica que F_θ es Lipschitz continua con constante L, lo que garantiza la regularidad y estabilidad de la red neuronal convolucional. □

[2] Borja-Robalino; Monleón-Getino y Rodellar, «Matemática oculta bajo el proceso de aprendizaje en redes neuronales convolucionales», óp.cit.

[3] McCarter, «Mathematical Analysis of Convolutional Neural Networks», óp.cit.

C. PROPIEDADES DE LAS REDES NEURONALES CONVOLUCIONALES (CNN)

Lema C.0.1 (Propiedad de Equivarianza de la Convolución)**.** *Sea $T_W : \mathbb{R}^{m\times n} \to \mathbb{R}^{m'\times n'}$ un operador de convolución definido por $T_W(X) = X * W$, donde $W \in \mathbb{R}^{k\times k}$ es un filtro convolucional, y sea T_v un operador de traslación de la entrada. Entonces, se cumple que:*

$$T_W(T_v(X)) = T_v(T_W(X)), \tag{C.1}$$

es decir, la convolución es equivariante respecto a la traslación de la entrada.

Demostración. Sea $X \in \mathbb{R}^{m\times n}$ una matriz de entrada y $W \in \mathbb{R}^{k\times k}$ un filtro convolucional. Definimos la operación de traslación T_v sobre X como:

$$(T_v(X))_{i,j} = X_{i+v,j+v}.$$

Aplicando la convolución al resultado de la traslación, se tiene:

$$(T_W(T_v(X)))_{i,j} = \sum_{u=0}^{k-1}\sum_{w=0}^{k-1}(T_v(X))_{i+u,j+w}W_{u,w}.$$

Sustituyendo la definición de traslación:

$$= \sum_{u=0}^{k-1}\sum_{w=0}^{k-1}X_{i+u+v,j+w+v}W_{u,w}.$$

Por otro lado, aplicando primero la convolución y luego la traslación:

$$(T_v(T_W(X)))_{i,j} = (T_W(X))_{i+v,j+v} = \sum_{u=0}^{k-1}\sum_{w=0}^{k-1}X_{i+u+v,j+w+v}W_{u,w}.$$

Por lo tanto,

$$T_W(T_v(X)) = T_v(T_W(X)),$$

lo que demuestra la propiedad de equivarianza de la convolución.[1] □

[1] Borja-Robalino; Monleón-Getino y Rodellar, «Matemática oculta bajo el proceso de aprendizaje

Teorema C.0.2 (Propiedad de Regularidad de las CNN)**.** *Sea F_θ una red neuronal convolucional parametrizada por θ, compuesta por d capas convolucionales con filtros $W_1, W_2, \ldots, W_d$. Entonces, F_θ es una función Lipschitz continua con constante Lipschitz L acotada por:*

$$L \leq \prod_{i=1}^{d} \|W_i\|_2, \tag{C.2}$$

donde $\|W_i\|_2$ denota la norma espectral (norma 2) del filtro W_i.[2]

Demostración. Consideremos F_θ como la composición de funciones convolucionales y funciones de activación:

$$F_\theta(X) = f_d\Big(W_d * (f_{d-1}(W_{d-1} * (\cdots f_1(W_1 * X) \cdots)))\Big),$$

donde cada f_i es una función de activación Lipschitz continua (por ejemplo, ReLU) con constante Lipschitz 1.

Cada capa convolucional es una transformación lineal cuyo operador está acotado por la norma espectral $\|W_i\|_2$. Por la propiedad de estabilidad de la norma Lipschitz en la composición de funciones, la constante Lipschitz de F_θ satisface:

$$L \leq \prod_{i=1}^{d} \|W_i\|_2.$$

Esto implica que F_θ es Lipschitz continua con constante L, lo que garantiza la regularidad y estabilidad de la red neuronal convolucional. □

en redes neuronales convolucionales», óp.cit.

[2]McCarter, «Mathematical Analysis of Convolutional Neural Networks», óp.cit.

BIBLIOGRAFÍA

ALPAYDIN, Ethem. *Introduction to Machine Learning*. 4th. MIT Press, 2020.

BANCO MUNDIAL. *Tasa de crecimiento porcentual anual del PIB en Colombia (constante)*. Accedido el 13 de febrero de 2025. 2023. URL: https://datos.bancomundial.org/indicador/NY.GDP.MKTP.KD.ZG?end=2023&locations=CO&start=2003.

BARRON, Andrew R. «Universal approximation bounds for superpositions of a sigmoidal function». En: *IEEE Trans. Inf. Theory* 39.3 (1993), págs. 930-945.

BEINAT, Natalia Casado. «Redes Neuronales Convolucionales y Aplicaciones». Tesis doct. Universidad Complutense de Madrid, 2022.

BORJA-ROBALINO, Rodrigo; MONLEÓN-GETINO, Antoni y RODELLAR, Josep. «Matemática oculta bajo el proceso de aprendizaje en redes neuronales convolucionales». En: *Ciencia Latina* (2022).

BOX, George E. P.; JENKINS, Gwilym M.; REINSEL, Gregory C. y LJUNG, Greta M. *Time Series Analysis: Forecasting and Control*. 5th. John Wiley & Sons, 2015.

BROCKWELL, Peter J. y DAVIS, Richard A. *Introduction to Time Series and Forecasting*. 3rd. Springer, 2016.

CEBOLLERO, Carmen Mayora. «Deep Learning from a Mathematical Point of View». Trabajo de Fin de Máster. Universidad de Zaragoza, jun. de 2021.

CHEN, Y.; ZHANG, M.; WANG, X. y LI, L. «Deep learning integration optimization of electric energy load forecasting and market price based on the ANN–LSTM–transformer method». En: *Front. Energy Res.* 11 (2023), pág. 1292204. DOI: 10.3389/fenrg.2023.1292204. URL: https://www.frontiersin.org/journals/energy-research/articles/10.3389/fenrg.2023.1292204/full.

CHOLLET, François. *Deep Learning with Python*. Manning Publications, 2018. ISBN: 9781617294433. URL: https://www.manning.com/books/deep-learning-with-python.

COMISIÓN DE REGULACIÓN DE ENERGÍA Y GAS (CREG). *Resolución CREG 024 de 1995: Aspectos comerciales del mercado mayorista de energía en el SIN*. https://gestornormativo.creg.gov.co/gestor/entorno/docs/resolucion_creg_0024_1995.htm. Accedido el 24 de julio de 2025. 1995.

CONGRESO DE COLOMBIA. *Ley 142 de 1994: Régimen de los servicios públicos domiciliarios*. https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=2752. Accedido el 24 de julio de 2025. 1994.

— *Ley 143 de 1994: Régimen para la generación, interconexión, transmisión, distribución y comercialización de electricidad*. https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=4631. Accedido el 24 de julio de 2025. 1994.

CS231N COURSE STAFF. *Convolutional Neural Networks (CNNs / ConvNets)*. https://cs231n.github.io/convolutional-networks/. Accedido: 2025-09-10. 2016.

CYBENKO, George. «Approximation by superpositions of a sigmoidal function». En: *Math. Control Signals Syst.* 2.4 (1989), págs. 303-314.

DATA (FEATURE-ENGINE), Train in. *CyclicalFeatures*. https://feature-engine.trainindata.com/en/1.8.x/user_guide/creation/CyclicalFeatures.html. Accedido: 2025-09-11. 2024.

DÍAZ GARCÉS, Cristian Julián. *Modelización del covid-19 en Santander mediante series temporales*. Tesis de pregrado, Universidad Industrial de Santander. 2023. URL: https://noesis.uis.edu.co/handle/20.500.14071/14839.

DUMOULIN, Vincent y VISIN, Francesco. «A guide to convolution arithmetic for deep learning». En: *arXiv preprint arXiv:1603.07285* (2016). DOI: 10.48550/arXiv.1603.07285. URL: https://arxiv.org/abs/1603.07285.

GOODFELLOW, Ian; BENGIO, Yoshua y COURVILLE, Aaron. *Deep Learning*. MIT Press, 2016. URL: https://www.deeplearningbook.org/.

HAMILTON, James D. *Time Series Analysis*. Princeton University Press, 1994.

HASTIE, Trevor; TIBSHIRANI, Robert y FRIEDMAN, Jerome H. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd. Springer, 2009.

HOCHREITER, Sepp y SCHMIDHUBER, Jürgen. «Long short-term memory». En: *Neural Comput.* 9.8 (1997), págs. 1735-1780.

HOOGEN, Jurgen van den. «Time Series Analysis Using Convolutional Neural Networks». Tesis doct. Tilburg, The Netherlands: Tilburg University, ene. de 2025. ISBN: 978-94-6506-648-6. DOI: 10.26116/tshd.30141293.

HORNIK, Kurt. «Approximation capabilities of multilayer feedforward networks». En: *Neural Netw.* 4.2 (1991), págs. 251-257.

HORNIK, Kurt; STINCHCOMBE, Maxwell y WHITE, Halbert. «Multilayer feedforward networks are universal approximators». En: *Neural Netw.* 2.5 (1989), págs. 359-366.

HYNDMAN, Rob J. y ATHANASOPOULOS, George. *Forecasting: Principles and Practice*. OTexts, 2018. URL: https://otexts.com/fpp2/.

JAIN, Abhishek. *Understanding the 1D Convolutional Layer in Deep Learning*. Medium. Accessed: September 25, 2025. Ene. de 2025. URL: https://medium.com/@abhishekjainindore24/understanding-the-1d-convolutional-layer-in-deep-learning-7a4cb994c981.

KAELBLING, Leslie Pack; LITTMAN, Michael L. y MOORE, Andrew W. «Reinforcement Learning: A Survey». En: *J. Artif. Intell. Res.* 4 (1996), págs. 237-285.

KERAS TEAM. *Conv1D layer*. https://keras.io/api/layers/convolution_layers/convolution1d/. Accedido: 2025-09-10. 2025.

KINGMA, Diederik P. y BA, Jimmy. «Adam: A Method for Stochastic Optimization». En: *Proc. Int. Conf. Learn. Represent.* (2015). URL: https://arxiv.org/abs/1412.6980.

KIRANYAZ, Serkan; AVCI, Onur; ABDELJABER, Osama; INCE, Turker; GABBOUJ, Moncef e INMAN, Daniel J. «1D convolutional neural networks and applications: A survey». En: *Mech. Syst. Signal Process.* 151 (2021), pág. 107398. DOI: 10.1016/j.ymssp.2020.107398.

LECUN, Yann; BENGIO, Yoshua e HINTON, Geoffrey E. «Deep learning». En: *Nature* 521.7553 (2015), págs. 436-444.

LESHNO, Moshe; LIN, Vladimir Ya.; PINKUS, Allan y SCHOCKEN, Shimon. «Multilayer feedforward networks with a nonpolynomial activation function can approximate any function». En: *Neural Netw.* 6.6 (1993), págs. 861-867.

MCCARTER, Daniel. «Mathematical Analysis of Convolutional Neural Networks». Tesis de mtría. University of South Dakota, 2023.

MEJÍA, Luis Fernando. «Retos de la seguridad y transición energética». En: *Fedesarrollo, Centro de Investigación Económica y Social* (2023).

MINSKY, Marvin y PAPERT, Seymour. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.

NVIDIA. *Three Approaches to Encoding Time Information as Features for ML Models*. `https://developer.nvidia.com/blog/three-approaches-to-encoding-time-information-as-features-for-ml-models/`. Accedido: 2025-09-11. 2022.

OLAH, Christopher. *Understanding LSTM Networks*. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`. Accedido: 2025-09-11. 2015.

RÍOS, Gonzalo y HURTADO, Carlos. «Series de tiempo». En: *Universidad de Chile. Facultad de Ciencias Físicas y Matemáticas* 52 (2008).

ROHRER, Brandon. *Convolution in one dimension for neural networks*. Accessed: September 25, 2025. 2024. URL: `https://brandonrohrer.com/convolution_one_d.html`.

ROJAS, Esperanza Manrique. «Machine Learning: análisis de lenguajes de programación y herramientas para desarrollo». En: *Revista Ibérica de Sistemas e Tecnologias de Informaç ao* E28 (2020), págs. 586-599.

ROSENBLATT, Frank. «The perceptron: a probabilistic model for information storage and organization in the brain». En: *Psychol. Rev.* 65.6 (1958), págs. 386-408.

RUMELHART, David E.; HINTON, Geoffrey E. y WILLIAMS, Ronald J. «Learning representations by back-propagating errors». En: *Nature* 323.6088 (1986), págs. 533-536.

RUSSELL, Stuart y NORVIG, Peter. *Artificial Intelligence: A Modern Approach*. 3rd. Prentice Hall, 2010.

SCHMIDHUBER, Jürgen. «Deep Learning in Neural Networks: An Overview». En: *Neural Netw.* 61 (2015), págs. 85-117.

SCIKIT-LEARN DEVELOPERS. *Time-related feature engineering*. `https://scikit-learn.org/stable/auto_examples/applications/plot_cyclical_feature_engineering.html`. Accedido: 2025-09-11. 2024.

SHUMWAY, Robert H. y STOFFER, David S. *Time Series Analysis and Its Applications: With R Examples*. 4th. Springer, 2017.

SKFORECAST. *Cyclical features in time series forecasting*. `https://skforecast.org/0.8.1/faq/cyclical-features-time-series`. Accedido: 2025-09-11. 2022.

SUTTON, Richard S. y BARTO, Andrew G. *Reinforcement Learning: An Introduction*. 2nd. MIT Press, 2018.

TSAY, Ruey S. *Analysis of Financial Time Series*. Wiley, 2010.

WANG, Yanwen; KHODADADZADEH, Mahdi y ZURITA-MILLA, Raúl. «Spatial+: A new cross-validation method to evaluate geospatial machine learning models». En: *Int. J. Appl. Earth Obs. Geoinf.* 121 (2023), pág. 103364. ISSN: 1569-8432. DOI: 10.1016/j.jag.2023.103364. URL: https://www.sciencedirect.com/science/article/pii/S1569843223001887.

XM S.A. E.S.P. *Cargo por confiabilidad: Precio de bolsa y precio de escasez*. https://www.xm.com.co/transacciones/cargo-por-confiabilidad/precio-de-bolsa-y-escasez. Accedido el 6 de marzo de 2024. 2024.