# Universidad Nacional de Ingeniería

## Facultad de Ingeniería Mecánica

TRABAJO DE SUFICIENCIA PROFESIONAL

# DESARROLLO DE UN MODELO DE PRONÓSTICO DE ENERGÍA A CORTO PLAZO, BASADO EN REDES NEURONALES RECURRENTES, PARA MEJORAR LA PROGRAMACIÓN DE GENERACIÓN DE ENERGÍA EN CENTRALES EÓLICAS

Para obtener el título de: Ingeniero Mecatrónico

Elaborado por

Nils Ericsson Sánchez Huayana

0009-0000-0065-0351

Asesor

MSc. Alcides Guillermo Joo Aguayo

0000-0002-8459-8489

LIMA – PERÚ

2023

**Dedicatoria**

A mis queridos padres, luz y guía permanente en mi camino, cuyo amor y apoyo inagotable han sido mi refugio y fuerza. A mi tía Gardenia, que me ha arropado con una bondad en los albores de mi educación. A mis abuelos, cuyo legado trasciende el cielo, siendo faros que iluminan cada palabra y hallazgo aquí presentes. A ustedes, con todo mi corazón, dedico cada página y victoria alcanzada en este viaje de conocimiento.

**Agradecimiento**

Este trabajo es el resultado de innumerables esfuerzos y apoyos recibidos. Debo un especial reconocimiento a mi asesor, Guillermo Joo, cuya disponibilidad y apoyo constante han sido cruciales en mi recorrido investigativo. A mi compañera de vida, Mary Claudia, le agradezco por ser el recordatorio viviente de nuestras metas y sueños compartidos. A mis padres y abuelos, les debo mi gratitud eterna por su amor y soporte incondicional. No puedo dejar de mencionar a la Srta. Letizia, cuya ayuda administrativa en la facultad de Mecánica fue indispensable. A todos ellos, mi más sincero agradecimiento por hacer este trabajo posible.

# Resumen

El objetivo de este estudio de investigación es mejorar la precisión del pronóstico de energía eólica a corto plazo mediante la propuesta de un modelo de pronóstico basado en la descomposición en modo variacional y la memoria a corto plazo a largo plazo (VMD-LSTM). Los datos utilizados para este modelo se obtuvieron de un parque eólico ubicado en Talara, al norte de Perú.

Inicialmente, se adoptó el método de descomposición en modo variacional para descomponer los datos de energía eólica en tres modos constituyentes: el componente a largo plazo, el componente de fluctuación y el componente aleatorio. Posteriormente, se empleó la red neuronal recurrente (RNN), específicamente la variante de memoria a corto plazo (LSTM), para analizar en profundidad las características de estos tres modos constituyentes. Gracias a su estructura única de puerta de olvido y puerta de memoria, la LSTM aprende la asociación con series de tiempo a largo plazo para construir un modelo de pronóstico de varios pasos.

El algoritmo se desarrolló utilizando el lenguaje de programación Python en el entorno de desarrollo de Google Colab, con el objetivo de aprovechar su capacidad computacional y su interfaz amigable.

La precisión del modelo VMD-LSTM se evaluó y los resultados experimentales indican que el modelo propuesto proporciona un rendimiento superior en el pronóstico de varios pasos.

Palabras clave: Pronóstico de energía eólica; Descomposición modal variacional; Red neuronal de memoria a corto plazo; Optimización; Red neuronal recurrente.

# Abstract

The aim of this research study is to enhance the accuracy of short-term wind energy forecasting by proposing a forecasting model based on Variational Mode Decomposition and Long Short-Term Memory (VMD-LSTM). The data used for this model were sourced from a wind farm located in Talara, in the north of Peru.

Initially, the Variational Mode Decomposition method was adopted to decompose the wind energy data into three constituent modes: the long-term component, the fluctuation component, and the random component. Subsequently, the Recurrent Neural Network (RNN), specifically the Long Short-Term Memory (LSTM) variant, was employed to deeply analyze the characteristics of these three constituent modes. Thanks to its unique forget gate and memory gate structure, the LSTM learns the association with long-term time series to construct a multi-step forecasting model.

The algorithm was developed using the Python programming language in the Google Colab development environment, aiming to leverage its computational power and user-friendly interface.

The accuracy of the VMD-LSTM model was evaluated, and the experimental results indicate that the proposed model provides superior performance in multi-step forecasting.

Keywords: Wind energy forecasting; Variational mode decomposition; Short-term memory neural network; Optimization; Recurrent neural network.

# Tabla de Contenido

# Lista de Figuras

# Lista de Tablas

# Prólogo

El presente trabajo de suficiencia profesional tiene como finalidad mejorar la programación de generación de energía en centrales eólicas mediante la predicción de generación de energía con un modelo neuronal basado en redes neuronales recurrentes de memoria a corto plazo (LSTM) y descomposición en modo variacional (VMD). Este trabajo abarca los siguientes capítulos:

El primer capítulo aborda el tema de la introducción, donde se explican las generalidades, la descripción del problema en el proceso de generación de energía eólica, el objetivo, los antecedentes investigativos nacionales e internacionales del presente trabajo.

El segundo capítulo aborda el marco teórico y conceptual, los cuales describen los fundamentos básicos de definición, clasificación y tipos de aprendizaje de las redes neuronales artificiales. Asimismo, se detallan las redes neuronales recurrentes y su tipo especial de red, LSTM, el cual es clave en la predicción de sistemas secuenciales y su variación mediante la descomposición en modo variacional de las variables de entrada.

El tercer capítulo contiene la hipótesis y operacionalización de variables dependiente e independiente con sus respectivos indicadores que se desarrollan en el trabajo de investigación.

El cuarto capítulo titulado metodología de la investigación, se explica el tipo y diseño de la investigación, la unidad de análisis (especifica donde se desarrolla la investigación) y la matriz de consistencia.

El quinto capítulo es el desarrollo del trabajo de investigación y éste abarca el procesamiento de la información recolectada para el diseño de la red neuronal recurrente tipo LSTM-VMD para mostrar los resultados conseguidos.

El sexto capítulo describe el análisis y discusión de resultados del trabajo desarrollado en el capítulo anterior. Asimismo, se realiza la contratación de hipótesis del trabajo de investigación. Finalmente, se exponen las conclusiones, recomendaciones, referencias bibliográficas, anexos y apéndices del trabajo de suficiencia profesional.

# Capítulo 1. Introducción

## 1.1 Generalidades

La energía eólica es una tecnología de origen renovable y de rápido crecimiento en los últimos 7 años (IRENA, 2021); la demanda de este tipo de energía está en aumento, alrededor del mundo, debido a la disminución de costos para la instalación de este tipo de centrales energéticas, así como al cuidado del medio ambiente. El Perú actualmente, cuenta con cuatro parques eólicos, localizados en la costa peruana (Ica, La Libertad y Piura); asimismo existen otros emplazamientos donde es posible aprovechar el potencial de generación de energía del Perú, como proyectos nuevos aprobados o en revisión para el 2023-2024 (Enel Green Power, 2021).

El principal problema la energía eólica, es que la energía eléctrica producida por los aerogeneradores, depende de las condiciones del clima y la intensidad del viento; tales condiciones son fluctuantes, lo que ocasiona desequilibrios entre carga eléctrica y la generación de la energía; por lo cual se reduce la capacidad de control de voltaje y frecuencia del sistema interconectado, lo que deteriora la capacidad total del sistema para manejar desbalances entre carga y generación; es por ello, que un mejor ajuste de modelos predictivos ayuda a reducir estos problemas de desbalance y evitar penalidades.

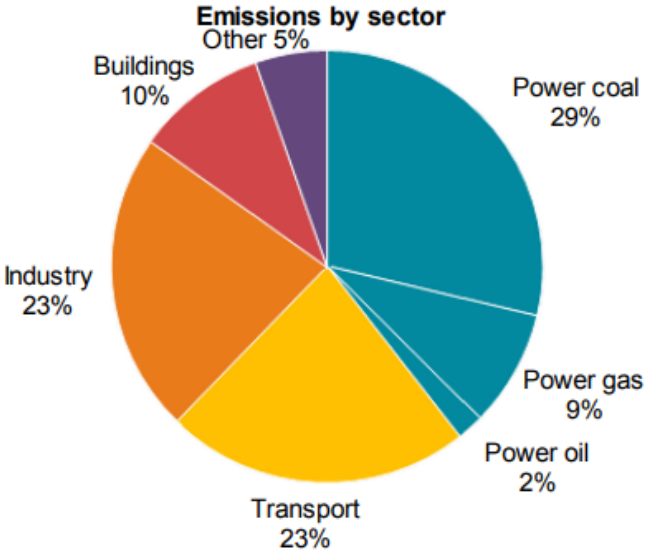
Debido a lo explicado anteriormente, una adecuada estimación (proyección) de la energía eléctrica producida por los aerogeneradores, permite una mejor programación del suministro de energía y estar preparados para los cambios debido a estas variables meteorológicas, además de una mejorar la eficiencia para la operación y mantenimiento.

## 1.2 Descripción del problema de investigación

A nivel global, la producción de electricidad se fundamenta esencialmente en la combustión de combustibles fósiles, factores tales como su coste económico reducido y la disponibilidad del carbón como recurso primordial contribuyen a esta dependencia (Liddle & Sadorsky, 2017). Las emanaciones del ámbito eléctrico global en 2018 marcaron un pico histórico, alcanzando 13,8 GtCO2. Como se refleja en la Figura 1, este valor constituye el 40% del total de emisiones de CO2. Por ende, siendo el sector eléctrico la fuente principal de emisiones de CO2, la necesidad de explorar métodos que permitan reducir dichas emisiones se torna indispensable.

**Figura 1**

*Emisión de CO2 por sector.*




*Nota*. El grafico representa las emisiones de CO2 por sector industrial del año 2020 (IEA, 2020).

La energía renovable aparece como una alternativa a las energías provenientes de la quema de combustibles fósiles. La comunidad internacional ha firmado compromisos para aumentar la generación de electricidad renovable con el objetivo principal de reducir las

emisiones de CO2 para 2050. Este objetivo requerirá duplicar la producción actual de energía renovable para 2020 (Direccion General de Electricidad, 2019). Sin embargo, las fuentes de energía renovable son de naturaleza intermitente, por lo que es una tarea desafiante integrar los recursos de energía renovable en la red eléctrica. Algunos de los desafíos y problemas asociados con la integración en la red de diversas fuentes de energía renovable, en particular, los sistemas de generación de energía solar fotovoltaica y eólica son descritos a continuación (Kumar, Pandey, & Sinha, 2016):

Problemas técnicos:

1. Problemas de integración en la red para la generación a pequeña escala (inferiores a 100MW):
   - ✓ Coste, fiabilidad y eficiencia de la de red
   - ✓ Congestión de la red, redes débiles
   - ✓ Variabilidad de la producción renovable
   - ✓ Baja calidad de la energía
   - ✓ Problemas de protección
   - ✓ Flujo de potencia bidireccional en la red de distribución
   - ✓ Problemas de estabilidad de tensión localizados
2. Problemas relacionados con la integración de la red de generación a gran escala (parques eólicos con capacidades superiores 100MW):
   - ✓ La necesidad de potencia reactiva para el soporte de la tensión es una de las cuestiones clave relacionadas con la generación de energía eólica.
   - ✓ Diseño de la electrónica de potencia de la turbina y optimización del controlador.

- ✓ Problemas de los parques eólicos conectados a sistemas compensados en serie.
- ✓ Problemas de calidad de la energía, incluido el parpadeo de la tensión.
- ✓ Arranque y sincronización de los parques eólicos con la red.
- ✓ Problemas de resonancia subsíncrona debidos a la interacción de la red eléctrica y el complejo sistema de ejes y engranajes del aerogenerador.

Problemas no técnicos:

1. Falta de personal técnico calificado
2. Menor disponibilidad de líneas de transmisión para dar cabida a las RER.
3. Las tecnologías de RER se excluyen de la competencia al darles prioridad en el despacho, lo que desalienta la instalación de nuevas centrales eléctricas con fines de reserva

Avanzando en esta dirección, en mayo de 2008, el Gobierno de Perú promulgó el Decreto Legislativo 1002. Este marco legal propone, mediante la implementación de subastas, la promoción de inversiones para la generación de electricidad con Recursos Energéticos Renovables (RER), también conocidos como fuentes de energía renovables no convencionales (ERNC). Entre estas se incluyen la energía eólica, solar, geotérmica, mareomotriz, la biomasa, y pequeñas hidroeléctricas con una capacidad instalada de hasta 20 MW. Este marco normativo establece que el objetivo es que la producción de electricidad mediante RER alcance hasta el 5% del consumo nacional de electricidad anual en cada año del primer quinquenio (Decreto Legislativo Nº 1002, 2010).

La generación eólica promovida por el estado continúa su crecimiento en las dos últimas décadas principalmente apoyadas mediante las subastas RER, lográndose instalar del 2014 al 2019, 372 MW de capacidad instalada, siendo de mayor tamaño, la C.E. Wayra

instalada en el año 2018 con una capacidad de 132 MW, en la Tabla 1 se observa las principales centrales eólicas del Perú. (DGEE, 2019).

**Tabla 1**

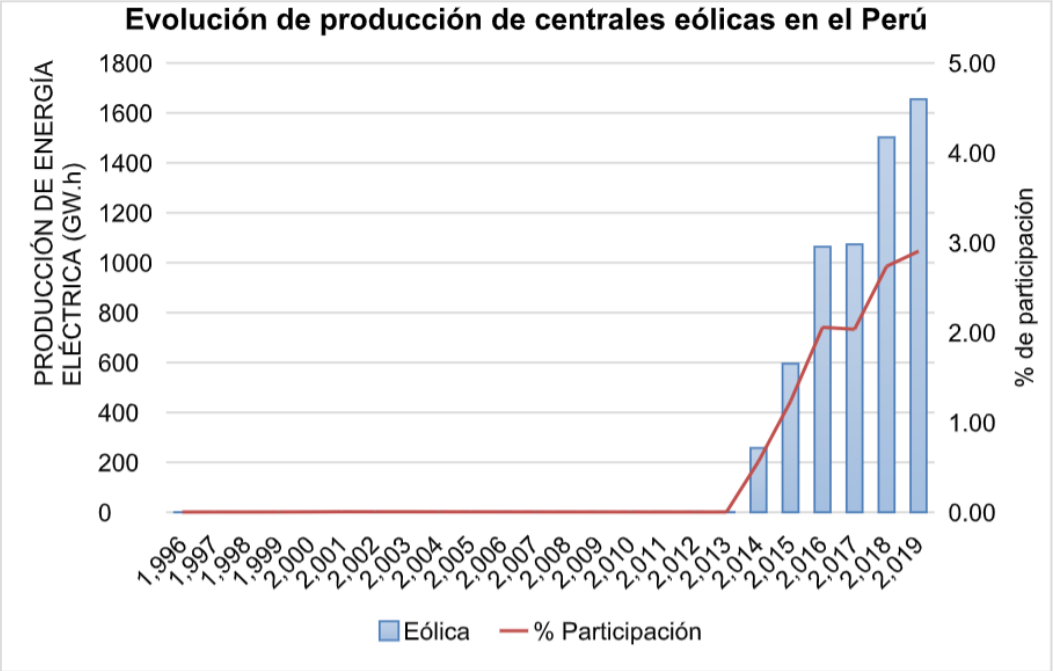
Las Centrales eólicas del mercado eléctrico peruano

| Empresa | Central | Ubicación | Producción | | Potencia instalada | |
|---|---|---|---|---|---|---|
| | | | Cantidad (MWh) | Part. | Cantidad (MW) | Part. |
| PARQUE EÓLICO MARCONA S.R.L. | MARCONA | Ica | 157 115 | 9,5% | 32,1 | 8,6% |
| ENERGÍA EÓLICA S.A. | CUSPINIQUE | La Libertad | 322 612 | 19,5% | 80,0 | 21,5% |
| ENERGÍA EÓLICA S.A. | TALARA | Piura | 128 005 | 7,7% | 30,0 | 8,1% |
| PARQUE EÓLICO TRES HERMANAS S.A.C. | TRES HERMANAS | Ica | 461 166 | 27,9% | 97,2 | 26,2% |
| ENEL GREEN POWER PERÚ S.A. | WAYRA I | Ica | 584 915 | 35,4% | 132,3 | 35,6% |
| **TOTAL** | | | **1 653 813** | **100,0%** | **371,6** | **100,0%** |

*Nota. Adaptado de (DGEE, 2019*

Figura **2**

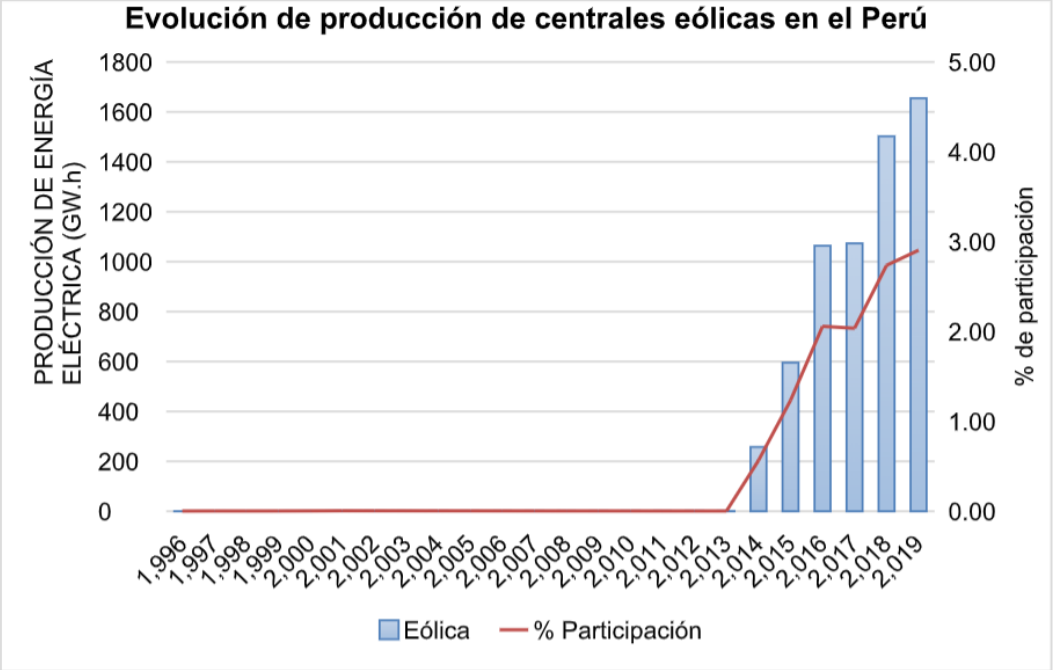
*Evolución de la generación eólica en el Perú.*



En la Figura 2, se observa el crecimiento de la participación de generación eólica en el Perú. El 2019 tuvo una participación del 2.01% con 1655.04 GW-H de un total de 56,968.50 GW-H. Por tanto, se espera que la generación Recursos Energéticos Renovables (RER) mantenga el crecimiento sostenido en los próximos años.

**Figura 2**

*Evolución de la generación eólica en el Perú.*




*Nota*. Adaptado de Estadísticas Reportadas del MINEM, por MINEM, 2019.

No obstante, las energías renovables difieren significativamente de las fuentes de generación de energía convencionales. A medida que aumenta la proporción de energías renovables variables, estas diferencias conducen a numerosos desafíos en los sistemas de energía. (Loaiza Muñoz, 2019)

En esa dirección, la predicción de la demanda de electricidad es un componente fundamental para el control diario, programación, operación, planificación y estabilidad del sistema eléctrico. (Chapagain, Kittipiyakul, & Kulthanavit, 2020). Desde el ámbito de los

mercados de energía, esta toma de decisiones está sujeta a factores como el clima, la anticipación para comprar, vender la energía o la alta penetración de la energía renovable, entre otros, introduciendo incertidumbre en la predicción de la demanda de energía y haciendo que esta tarea sea un gran reto. (Fan, Peng, & Hong, 2018)

El manual de procedimiento " Programación de la Operación de Corto Plazo (PR-01)" (COES, 2014) de El Comité de Operación Económica del Sistema (COES) establece los criterios técnicos y la metodología para la elaboración de los programas de la operación de Corto Plazo de las Unidades de Generación del Sistema Eléctrico Interconectado Nacional (SEIN), que es el proceso mediante el cual se definen para el o los días siguientes, según corresponda, los niveles de generación de cada una de las centrales del sistema, con el objeto de obtener la operación del sistema eléctrico seguro y económico.

En Perú, el COES, basándose en información estadística, determina la demanda de potencia para el día o días subsiguientes, a partir de lo cual establece la generación horaria de cada central de generación. El criterio más relevante para determinar el nivel de generación de las centrales radica en la utilización de una orden de mérito, concebida primordialmente en base a los costos variables de cada central. Otras consideraciones incluyen las programaciones de mantenimiento y las especificaciones técnicas, así como las restricciones de las instalaciones del sistema. Sin embargo, los parques eólicos gozan de la ventaja de un costo variable de generación igual a cero. Siendo este un factor determinante en la definición del programa diario, resulta habitual que estén en funcionamiento de manera constante, las 24 horas, los siete días de la semana (Dammert, Molinelli, & Carbajal, 2011).

El COES debe mantener un estándar de calidad de suministro para lo cual se utilizan reservas por sobre la demanda esperada (Dammert, Molinelli, & Carbajal, 2011). Al no conocerse con precisión el nivel de generación en un parque eólico (debido a la naturaleza estocástica del viento), no es posible determinar con exactitud cuánto es el nivel de generación

para cada central de manera tal de cumplir en todos los horarios con un despacho óptimo, entendiéndose éste como aquel que minimiza las reservas satisfaciendo la demanda plenamente.

De acuerdo a Spodniak et al. (2021) se estudiaron las interrelaciones entre los diferenciales de precios, efectos del pronóstico del viento y los errores de pronóstico de la demanda, y otras variables exógenas para los países de Dinamarca, Suecia y Finlandia. El estudio concluye que los errores de pronóstico del viento afectan los diferenciales de precios en áreas con proporción de generación eólica superior al 34%. Los errores de pronóstico de la demanda impactan en los diferenciales de precios, excepto en áreas con participación baja de 6.4%, donde no se muestra un claro efecto y concluyente.

En el estudio de Hu, X. et al. (2021) concluyen que los errores de pronóstico de energía eólica y las primas de precios intradía resultan negativas en tres de las cuatro áreas de estudio en Suecia. En primer lugar, indica que cuando la producción eólica real difiera de su pronóstico, el precio de la electricidad en el mercado intradiario va a incorporar esta información y se diferenciará del precio del mercado diario. En particular, cuando la producción real de energía eólica es mayor que su pronóstico, los participantes del mercado están dispuestos a pagar un precio de electricidad más bajo en el mercado intradiario en comparación con el precio de la electricidad en el mercado diario y viceversa. En segundo lugar, proporciona algunas ideas sobre cómo los costos de equilibrio en el mercado intradiario podrían depender de la señal de errores de pronóstico de energía eólica

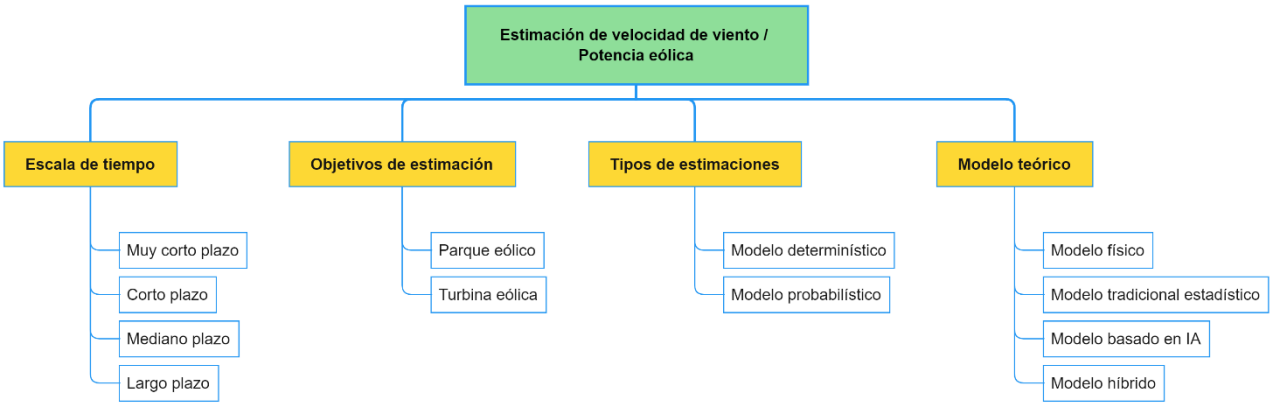
En el año 2011, el gobierno central de China estableció una exigencia para que cada central eólica implementara un sistema de pronóstico de energía eólica antes de junio de 2012, tarea que ya ha sido parcialmente completada. Esta demanda surgió a raíz de la significativa inexactitud de los sistemas de predicción y pronóstico de la energía eólica existentes en ese momento (Liu, Guan, & Hou, 2019).

En China en el 2019, de acuerdo con “Measures for the Management of Wind Farm Power Forecast", el error de pronóstico en tiempo real no debe ser más del 15%. (Li, Huitian, Rongchang, & Zhiyu, 2019)

Con el fin de abordar los principales retos del pronóstico de energía eólica se propusieron y aplicaron modelos bajo diferentes enfoques. De acuerdo con diferentes normas de clasificación, se divide en cuatro grupos, como se muestra en la Figura 3. Sin embargo, la energía eólica, es uno de los elementos meteorológicos más desafiantes de pronosticar. Cada metodología experimenta pros y contras, aplicables en casos específicos e inapropiados en otros.

**Figura 3**

Clasificación de modelos de pronóstico de nergía eólica.




*Nota. Adapatado de* (Wang, Zou, Liu, Zhang, & Liu, 2021).

En todos estos frentes se han producido mejoras desde los primeros modelos. Las cifras típicas en cuanto a precisión son un RMSE de aproximadamente el 9%-14% para un horizonte de 24 horas. (Giebel & Kariniotakis, 2017).

En Perú, para el pronóstico de energía se utilizan datos velocidad del viento de páginas web como Wisuki y Tu Tiempo, la velocidad del viento obtenidos de estos sitios web es

correlacionada con la curva de potencia de cada aerogenerador y de esa manera se determina la energía generada para cada aerogenerador y posteriormente el total de la central eólica, sin embargo, este tipo de prácticas asume interferencias como lejanía del centro meteorológico hacia el parque eólico e indisponibilidades de equipos. Otras centrales eólicas optan por contratar servicios para el pronóstico de energía con elevados costos.

Por lo tanto, al considerar las características actuales del pronóstico de energía eólica en el contexto del Perú, se presenta la necesidad y la oportunidad de rediseñar los modelos de estimación actuales que se realizan en los parques eólicos.

## 1.3 Formulación del problema

¿Cómo mejorar la precisión del pronóstico de la generación de energía eólica utilizando un modelo basado redes neuronales recurrentes de memoria a corto plazo (LSTM) y descomposición en modo variacional (VMD), en comparación con los métodos actuales?

## 1.4 Objetivo

### 1.4.1 *Objetivo general*

Desarrollar un modelo de pronóstico de energía eólica a corto plazo basado en redes neuronales recurrentes de memoria a corto plazo (LSTM) y descomposición en modo variacional (VMD) para mejorar la programación de generación de energía eólica en centrales eólicas.

## 1.5 Antecedentes referenciales

En esta sección se presenta diferentes estudios realizados de modelos de pronóstico de producción y demanda eléctrica, los cuales fueron publicados en la literatura internacional y nacional.

### 1.5.1 *Antecedentes internacionales*

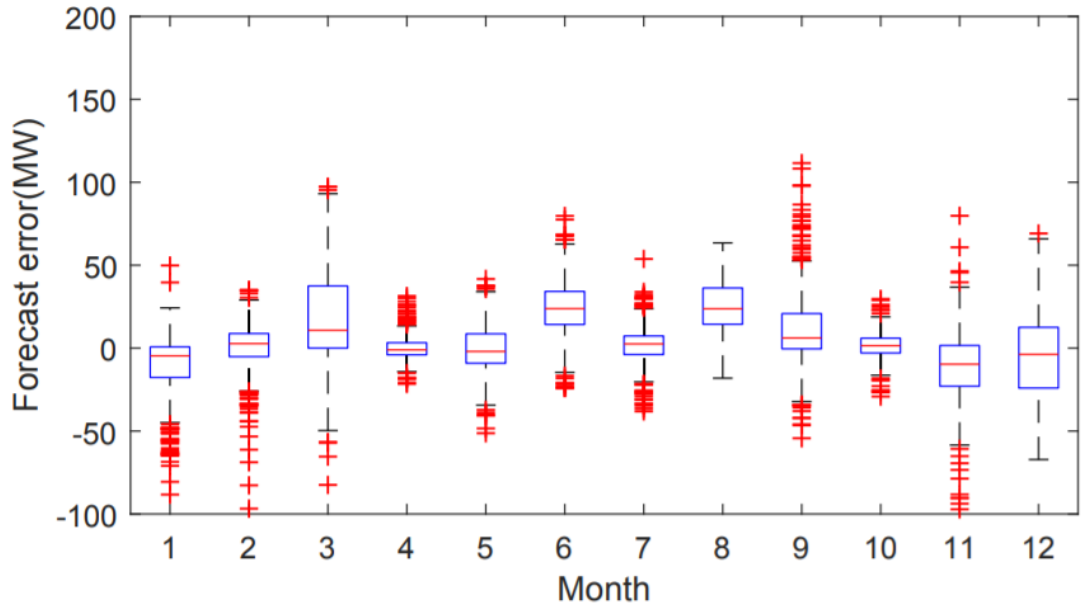
Han, L. et al. (2019) “Wind power forecast based on improved Long Short-Term Memory network”. Proponen un modelo de red neuronal de memoria corto plazo mejorada (ILSTM por sus siglas en inglés) para el pronóstico de energía eólica a muy corto plazo (horizonte de 1h, 2h, 3h y 4h), donde primeramente utilizaron el método de descomposición de método variacional (VMD por sus siglas en inglés) de 3 capas para obtener el componente a largo plazo, el componente de fluctuación a corto plazo y el componente aleatorio de la señal de energía eólica. El estudio se llevó a cabo utilizando los datos de Belgian ELIA website (ELIA, 2018). Durante el proceso de entrenamiento, se utilizaron la data de los primeros 1000 y los segundos 1000 datos de diciembre de 2018 para el entrenamiento y el test respectivamente.

En la Figura 4 y Figura 5, se observa un error positivo que supera los 150MW en el mes de enero y un error negativo de 100MW en el mes de diciembre para el modelo de pronóstico LSTM, sin embargo, el error puntual del modelo de pronóstico ILSTM en cada mes cambia dentro de un rango menor. Esto demuestra que el método propuesto pronóstica con un menor error medio porcentual absoluto (MAPE por sus siglas en ingles).

La red mejorada ILSTM también evita el sobreajuste causado por los componentes aleatorios, mejorando así la capacidad de generalización. Y se consigue un modelo de pronóstico preciso basado en ILSTM.

**Figura 4**

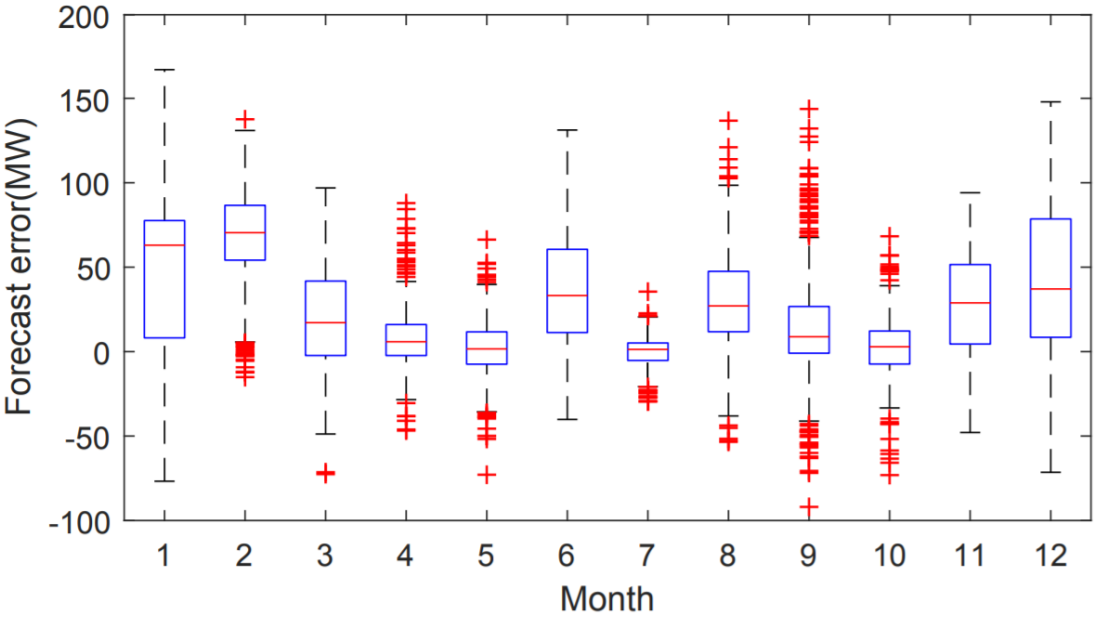
*Diagrama de caida de erro del pronóstico de LSTM*




*Nota*. Adaptado de (Li, Huitian, Rongchang, & Zhiyu, 2019).

**Figura 5**

*Diagrama de caja de error del pronóstico del ILSTM*




*Nota.*

*Nota. Adaptado de (*Li, Huitian, Rongchang, & Zhiyu, 2019).

Liu, Y. et al. (2019) “Wind Power Short-Term Prediction Based on LSTM and Discrete Wavelet Transform”. Propone un método de pronóstico a corto plazo de la energía eólica basado en la transformada wavelet discreta y las redes de memoria a corto plazo (DWT-LSTM por sus siglas en ingles).

Se seleccionaron tres series temporales de datos de energía eólica de tres parques eólicos diferente. Cada serie temporal abarcó 12 meses para que los experimentos fueran más sólidos. El intervalo de tiempo entre dos datos de potencia vecinos era de 15 minutos (35.000 puntos en total). El parámetro de los datos de energía eólica fue MW. Los tres parques eólicos estaban situados en el interior de Mongolia (parque 1), en los Países Bajos (parque 2) y en Yunnan, China (parque 3), respectivamente. Estos parques eólicos estaban lo suficientemente alejados como para garantizar que las tres series temporales de datos de energía eólica fueran diferentes.

La transformada wavelet discreta (DWT por sus siglas en inglés) se introduce para descomponer la serie temporal de energía eólica no estacionaria en varios componentes con mayor estacionalidad y por ser más fáciles de pronosticar. Cada componente es excavado por un LSTM INDEPENDIENTE. Los resultados de la predicción de la energía eólica se obtienen sintetizando los valores de predicción de todos los componentes. La precisión de la predicción se ha mejorado con el método propuesto, que fue validado por el error medio absoluto (MAE por sus siglas en inglés), error medio porcentual absoluto (MAPE por sus siglas en ingles) y error cuadrático medio (RMSE) de los resultados experimentales de tres parques eólicos como referencia. El pronóstico de la energía eólica basada en el método propuesto proporcionó una forma alternativa de mejorar la seguridad y la estabilidad de la red eléctrica con la alta penetración de la energía eólica.

Qu Xiaoyun. et al. (2016), "Short-term prediction of wind power based on deep Long Short-Term Memory". Este artículo propuso un modelo de predicción de la energía eólica

basado en el modelo de memoria a corto plazo (LSTM por sus siglas en inglés), un método de aprendizaje profundo. Se utilizó el análisis de componentes principales (PCA por sus siglas en inglés) para elegir las muestras de entrada y reducir las dimensiones de las variables de entrada del LSTM. del modelo de predicción LSTM basado en datos de predicción meteorológica numérica (NWP).

El estudio se llevó a cabo utilizando los datos del parque eólico de Manchester un parque eólico es de 120MW. Los datos de 2010 a 2011 contienen los datos de medición de la energía eólica y los datos NWP del parque eólico de Manchester, incluyendo la densidad del aire, presión, temperatura, velocidad del viento y dirección del viento en 100m. El intervalo de tiempo es de 5 minutos.

Mediante el entrenamiento y el aprendizaje del modelo LSTM, se pronostica la energía eólica futura de 24 horas. Los resultados de la simulación mostraron que en comparación con la red neuronal BP y el modelo SVM, la precisión de la predicción de PCA-LSTM es mucho mayor. EL MÉTODO LSTM analizó eficazmente datos masivos. Demostró tener la ventaja de la generalización y la capacidad de aproximación de alta dimensión. En consecuencia, se demostró que el modelo de memoria a largo plazo es avanzado y práctico en el campo del pronóstico de la energía eólica.

Liu, H. et al. (2018). “Smart multi-step deep learning model for wind speed forecasting based on variational mode decomposition, singular spectrum analysis, LSTM network and ELM”. Concluyó que una predicción precisa y robusta de la velocidad del viento es esencial para la planificación, programación y mantenimiento de la energía eólica. En este estudio se propuso un nuevo modelo de predicción de la velocidad del viento en varios pasos, combinando la descomposición del modo variable (VMD), el análisis del espectro singular (SSA), la red LSTM (Long Short Term Memory) y la máquina de aprendizaje extremo (ELM); la red LSTM se utilizó para completar el pronóstico de las subcapas de baja frecuencia

obtenidas por el VMD-SSA; y el ELM se utiliza para completar el pronóstico de las subcapas de alta frecuencia obtenidas por el VMD-SSA. Para investigar el rendimiento de la predicción en varios pasos de los modelos propuestos, se incluyeron ocho modelos en las comparaciones. Los resultados de los experimentos muestran que:

a. Entre todos los modelos implicados, el modelo propuesto tiene el mejor rendimiento de predicción en varios pasos;

b. En comparación con los otros modelos implicados, el modelo propuesto es más eficaz y robusto a la hora de extraer la información sobre la tendencia.

### 1.5.2 *Antecedentes nacionales*

Fernández Quiñones, E. R. (2014). “Pronóstico de la demanda eléctrica a corto plazo utilizando redes neuronales artificiales aplicado al Sistema Interconectado Nacional”. En la presente tesis se desarrolló la propuesta de pronóstico de la demanda eléctrica a corto plazo, basado en Redes Neuronales Artificiales y aplicado al Sistema Eléctrico Interconectado Nacional (SEIN), el cual es modelado como una sola barra; con el objetivo de minimizar las diferencias entre la demanda eléctrica proyectada y la demanda real. Esta nueva alternativa de pronóstico de demanda eléctrica es posible generalizar para cualquier barra de un Sistema Eléctrico de Potencia. En esta tesis se desarrolla una metodología para la construcción de una base de datos de perfiles de demanda eléctrica cada media hora, agrupados en días típicos, periodos estacionales y feriados. La validez de los resultados obtenidos se ha verificado con datos históricos reales. También se analiza el efecto que las variables climatológicas influyen en la demanda eléctrica.

Pajares Correa, F. K. (2013). “Optimización del consumo y demanda de energía eléctrica utilizando redes neuronales artificiales y modelos dinámicos de simulación”. Este trabajo estudia la aplicación de técnicas modernas de predicción y simulación en la

optimización del uso de la energía eléctrica dentro de la planta industrial. La predicción de la demanda eléctrica es realizada utilizando redes neuronales dinámicas recurrentes, mediante las cuales se intenta prever el comportamiento de la potencia (MW) en el tiempo, empleando patrones definidos mediante un estudio de caso y referencias de trabajos previos. Este módulo ayudará a reorganizar la planificación eléctrica, de tal manera que se reduzcan al mínimo los picos de demanda. Las redes recurrentes son sistemas dinámicos no lineales capaces de descubrir regularidades temporales en las secuencias procesadas y aplicables, por lo tanto, a multitud de tareas de procesamiento de este tipo de secuencias. Esta tesis se centra en la aplicación de las redes neuronales recurrentes a la predicción de la demanda eléctrica. La simulación dinámica de la planta industrial es realizada utilizando técnicas modernas de estimación paramétrica, modelado por redes neuronales y ecuaciones matemáticas obtenidas por regresión estadística y modelos previamente elaborados en otros trabajos. Las mayores ventajas de la simulación desarrollada es que permite integrar los subsistemas más importantes dentro de un proceso industrial, como son: el eléctrico, el mecánico, el de instrumentación y el energético, con la finalidad de ensayar diversos escenarios de operación y control, con el fin de evaluar su impacto en todos los subsistemas mencionados. Esto constituye la segunda gran ventaja de la simulación propuesta, debido a que en los simuladores normales sólo se evalúa algún subsistema en particular o algún componente específico del proceso, dejándose de lado la influencia de tener un cambio, ya sea por motivos de mejora continua o reingeniería, en todo el proceso en general. Finalmente se plantean propuestas de ahorro energético en las zonas clave de la planta, simulando su aplicación y evaluando su impacto técnico y económico, tanto a nivel energético, como a nivel del proceso. La metodología utilizada para el desarrollo del trabajo, está enmarcada en las recomendaciones del Project Management Institute (PMI).

Blancas Sanchez, J. D. (2019). “Implementación del método razonamiento inductivo difuso para el pronóstico de la demanda eléctrica a corto plazo en el SEIN”. Enfocó su estudio en la aplicación del “razonamiento inductivo difuso” (FIR por sus siglas en inglés) al problema del pronóstico de la demanda eléctrica de corto plazo (Short Term Load Forecasting en inglés). El modelo FIR, que está basado en lógica difusa, aprende las relaciones pasadas de la demanda eléctrica (carga) y predice el comportamiento de la demanda a partir del último dato real agregado con el fin de establecer las desviaciones de la demanda programada versus la demanda real. El objetivo del estudio buscó determinar el menor error de pronóstico mediante el indicador “error porcentual absoluto medio” (MAPE por sus siglas en inglés), por lo cual en primer lugar se desarrollan los fundamentos de lógica difusa como base para comprender la metodología FIR y se propone un método de pronóstico previo solo utilizando la lógica difusa como herramienta. Posteriormente desarrolló la metodología FIR propuesta como mejora de la metodología con lógica difusa y que además fue complementada con el uso de un algoritmo evolutivo denominado “algoritmo de rebotes simulados” (SRA en inglés) que servirá como un método de optimización (minimización del MAPE) para determinar las relaciones lineales y no lineales entre las variables y así identificar el conjunto de variables de entrada que mejoran la precisión de la predicción. Entonces, tanto la metodología con lógica difusa como la metodología FIR complementada con la implementación de SRA, se aplicaron al sistema energético peruano mediante la utilización de los datos históricos de la demanda eléctrica del Sistema Eléctrico Interconectado Nacional para determinar el pronóstico de la demanda eléctrica del día siguiente (corto plazo). Finalmente, se demuestra que la metodología FIR ofrece errores menores en el pronóstico de la demanda eléctrica en comparación la metodología con lógica difusa desarrollada inicialmente y con la metodología actualmente utilizada por el operador del sistema eléctrico nacional (COES-SINAC), que utiliza ajuste por mínimos cuadrados como herramienta de pronóstico.

Jacinto, R., (2019). “Redes Neuronales para predicción de contaminación del aire en Carabayllo – Lima”, en su investigación realizó un modelo de pronóstico de predicción (con múltiples etapas adelantadas) del comportamiento de las variables de contaminación ambiental de material particulado $PM2.5$ en el distrito de Carabayllo – Lima. El modelo fue entrenado con datos reales de la estación automatizada de calidad de aire del distrito de Carabayllo en el intervalo de 2 años, y como variables de entrada se utilizaron los datos de concentraciones de contaminantes de material particulado ($PM2.5$ y $PM10$) y químicos ($CO$, $SO2$, $NO$) sobre tres diferentes algoritmos de retro propagación y dos modelos de neuronas en una única capa oculta para hallar parámetros de un modelo óptimo de predicción. La red neuronal fue aplicada sobre un grupo de 72 datos de prueba obteniendo resultados con un error porcentual medio de -0.1089% lo cual indicó un pronóstico preciso para el caso de estudio.

García Fernández, Leonardo Brain (2021). "Modelamiento del pronóstico de la demanda eléctrica diaria del sistema eléctrico interconectado nacional utilizando técnicas de MACHINE LEARNING". En esta investigación, se propuso un modelo computacional específico para el Sistema Eléctrico Interconectado Nacional (SEIN). El enfoque principal del estudio fue comparar el desempeño de dos avanzadas metodologías de Machine Learning: las redes neuronales de la Teoría de Resonancia Adaptativa (ARTMAP Fuzzy) y el modelo Neuro-Fuzzy (ANFIS).

Un aporte significativo de García Fernández fue la introducción de una metodología de pre-procesamiento para el conjunto de datos históricos, buscando mejorar la precisión en los resultados de pronóstico. Al evaluar estas técnicas en el pronóstico de demanda para los años 2019 y 2020, se utilizó el error porcentual medio absoluto (MAPE) como métrica principal. Los hallazgos del estudio sugieren una revisión y posible actualización del procedimiento técnico 03 (PR03) propuesto por el Comité de Operación Económica del Sistema Interconectado

Nacional (COES), subrayando la importancia de integrar y perfeccionar técnicas modernas de pronóstico en el sector eléctrico peruano.

Marcelo Barreto, Emilio Asunción et al (2021). "El modelo estocástico univariante ARIMA como herramienta predictiva de la demanda de energía eléctrica residencial del sistema eléctrico Cusco". Esta investigación se centró en el análisis de la serie temporal de la demanda de energía eléctrica residencial del Sistema Eléctrico Cusco, utilizando el análisis univariante ARIMA. Basándose en datos históricos mensuales obtenidos de OSINERGMIN desde enero de 1996 hasta junio de 2019, se buscó predecir la demanda para un horizonte de tres años.

El estudio demostró la eficiencia del modelo ARIMA al compararlo con el suavizado exponencial multiplicativo de Winter, una técnica determinística de predicción. Se identificó el modelo ARIMA Xt 0,5(0,1,1) (1,0,2) S como el más adecuado, logrando explicar de manera eficiente la demanda histórica residencial y su predicción para los tres años subsiguientes. Los resultados mostraron un error MAPE del 2.296% y se proyectó un crecimiento promedio del 7.825% en la demanda eléctrica residencial del Sistema Eléctrico Cusco para el período 2019-2022. Estos hallazgos ofrecen un sustento técnico valioso para la toma de decisiones en planificación e inversión, especialmente en la ampliación de redes de media y baja tensión en el área de influencia del sistema, beneficiando a Electro Sur Este al evitar inversiones inadecuadas.

# Capítulo 2. Marco teórico

## 2.1 Bases teóricas

Este apartado detalla los conceptos utilizados en el presente trabajo con el fin de profundizar el entendimiento del mismo, así como de la limitación de ambigüedades.

### 2.1.1 *Fundamentos de la Energía Eólica*

Dammert, A. et al (2011). “Fundamentos técnicos y económicos del sector eléctrico peruano”. Define que la energía eólica es la energía obtenida de la fuerza del viento, mediante la utilización de la energía cinética generada por las corrientes de aire.

En el estudio de Letcher (2017), menciona que la fuente de esta energía eólica, es la radiación solar. La radiación electromagnética del Sol calienta de forma desigual la superficie de la Tierra, más fuerte en los trópicos y más débil en las latitudes altas, como resultado de una absorción diferencial de la luz solar por el suelo, rocas, agua y la vegetación; el aire de las distintas regiones se calienta a diferente velocidad. Los patrones de calentamiento diferencial de la superficie terrestre, así como otros procesos térmicos como la evaporación, la precipitación, las nubes, la sombra y las variaciones de absorción de la radiación en la superficie aparecen en diferentes escalas espaciales y temporales. Estos se acoplan a las fuerzas dinámicas debidas de la Tierra para impulsar una variedad de procesos de generación de viento, lo que lleva a la existencia de una gran variedad de fenómenos eólicos.

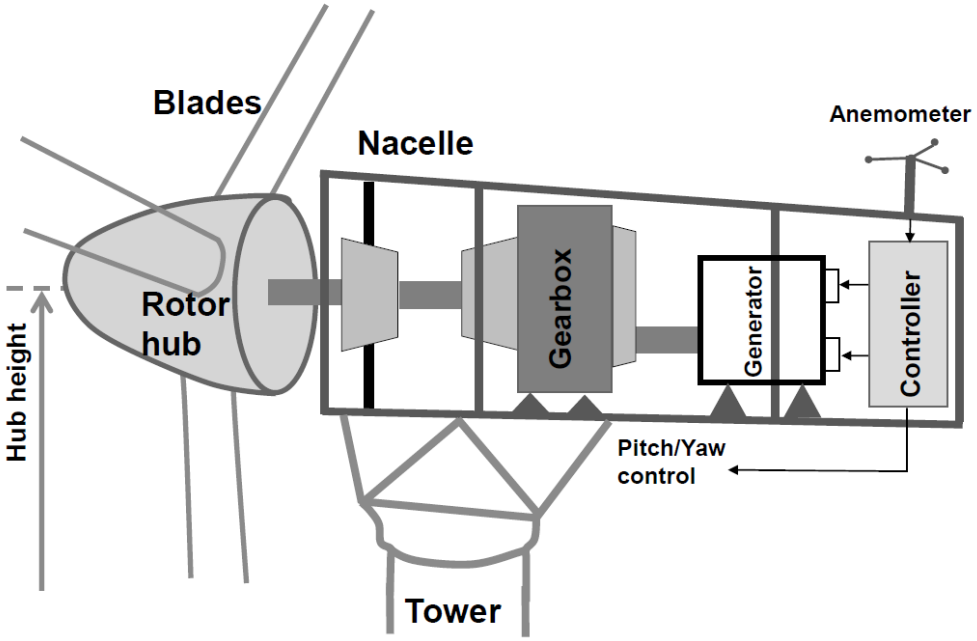
#### *2.1.1.1. Principios y componentes de un aerogenerador*

Ding, Y. (2020) “Data Science for Wind Energy”, Define que la turbina eólica es un dispositivo que convierte la energía cinética (del viento) en energía mecánica. Los aerogeneradores considerados en la presente investigación son los de eje horizontal como se

ilustra en la Figura 6, en el cual los principales componentes se encuentran en la góndola, el tren de transmisión y el sistema de control, incluyendo el reductor y el generador están dentro de la góndola. En la parte superior de la góndola, hacia su extremo posterior, se encuentran sensores para medir la velocidad y dirección del viento. En respuesta a los cambios en la dirección del viento, el sistema de control gira y orienta la góndola para absorber la energía cinética del viento

**Figura 6**

Esquema de las principales partes de un aerogenerador.




Nota. Adaptado *(Ding, 2020).*

## *2.1.1.2. Parques eólicos y su integración en la red eléctrica*

La integración de parques eólicos en la red eléctrica presenta varios desafíos que deben abordarse para garantizar la confiabilidad y la estabilidad del sistema eléctrico. Entre los desafíos se encuentran la intermitencia de la energía eólica, la calidad de la energía, la estabilidad de voltaje y frecuencia, la capacidad de atravesar fallas y la protección, entre otros.

Afortunadamente, existen soluciones disponibles para enfrentar estos desafíos, como los códigos de red [1]y los sistemas de almacenamiento de energía.

### *2.1.1.3. Factores que influyen en la generación de energía eólica*

Manwell, J. F., McGowan, J. G., & Rogers, A. L. (2009). “Wind Energy Explained: Theory, Design and Application”. Existen varios factores que influyen en la generación de energía eólica, algunos de los cuales incluyen:

- Velocidad y dirección del viento: La potencia generada es proporcional al cubo de la velocidad del viento, y la dirección afecta la orientación del aerogenerador.
- Altura de la torre: La velocidad del viento generalmente aumenta con la altura sobre el suelo debido a la menor fricción superficial, aumentando la generación de energía.
- Densidad del aire: La densidad del aire influye en la cantidad de energía que puede ser extraída del viento. A mayor densidad, mayor será la energía disponible.
- Rugosidad del terreno: El terreno circundante y las características topográficas influyen en la velocidad y turbulencia del viento.
- Diseño y eficiencia del aerogenerador: La eficiencia en la conversión de energía del viento a eléctrica depende del diseño y tecnología del aerogenerador.
- Disponibilidad y mantenimiento: El buen funcionamiento y mantenimiento adecuado de los aerogeneradores maximizan la producción de energía.

[1] Reglas técnicas para conexión de generadores eléctricos.

## 2.1.2 *Pronóstico de energía y planificación de la generación*

Ordoudis, Pinson, & Morales (2017) “An integrated market for electricity and natural gas systems with stochastic power producers”. El pronóstico de energía y la planificación de la generación implican predecir la demanda de energía y la producción de energía a partir de diversas fuentes, y luego utilizar estas predicciones para optimizar la asignación de recursos y el despacho de unidades de generación. Esto permite garantizar un suministro de energía confiable y eficiente, minimizar los costos de operación y las emisiones, y adaptarse a las fluctuaciones en la generación de energía, especialmente en el caso de las energías renovables.

La integración exitosa del pronóstico de energía en la planificación de la generación puede mejorar la eficiencia del sistema energético, facilitar la penetración de las energías renovables y respaldar la transición hacia una matriz energética más sostenible y resiliente.

### *2.1.2.1. Importancia y desafíos del pronóstico de energía a corto plazo*

Hong, T., Pinson, P., & Fan, S. (2016). “Global energy forecasting competition 2012”. La importancia del pronóstico de energía a corto plazo radica en su capacidad para mejorar la eficiencia y confiabilidad en la generación de energía, la distribución y la gestión de la demanda. Un pronóstico preciso permite una mejor asignación de recursos y una programación óptima de la generación. Esto resulta especialmente crítico en el caso de fuentes de energía renovable, como la eólica y la solar, que son variables e intermitentes por naturaleza.

Los desafíos del pronóstico de energía a corto plazo incluyen la variabilidad y la incertidumbre asociadas con las condiciones meteorológicas, la calidad de los datos de entrada y la selección de modelos de pronóstico adecuados.

### *2.1.2.2. Métodos tradicionales de pronóstico de energía*

Alfares, H. K., & Nazeeruddin, M. (2015). “Electric load forecasting: Literature survey and classification of methods”. Los métodos tradicionales de pronóstico de energía incluyen enfoques estadísticos y de series temporales, como la regresión lineal, el suavizado exponencial, los modelos autorregresivos integrados de promedios móviles (ARIMA) y el análisis espectral. Estos enfoques se han utilizado ampliamente en el pasado para predecir la demanda de energía y la generación de energía a partir de diversas fuentes, incluidas las energías renovables.

Sin embargo, estos enfoques tradicionales pueden tener dificultades para capturar la naturaleza no lineal y la complejidad de los sistemas energéticos modernos, especialmente con la creciente penetración de las energías renovables. En este contexto, se han desarrollado nuevos métodos de pronóstico basados en inteligencia artificial y aprendizaje automático, como las redes neuronales artificiales y las máquinas de vectores de soporte, para mejorar la precisión y la capacidad de adaptación de los modelos de pronóstico.

### *2.1.2.3. Integración del pronóstico en la programación de la generación*

Wang, Y., Zhang, N., & Kang, C. (2018). “Optimal scheduling of generation and demand response considering wind power uncertainty”. La integración del pronóstico en la programación de la generación implica utilizar las predicciones de la demanda y la generación de energía, incluyendo la incertidumbre asociada con la generación de energía renovable, para optimizar la asignación de recursos y el despacho de unidades de generación. La optimización considera la participación de diferentes fuentes de energía, como energías renovables y no renovables, y busca minimizar los costos de operación y las emisiones mientras se garantiza la confiabilidad del suministro.

La programación de la generación debe adaptarse a la incertidumbre en la generación de energía eólica, lo que requiere un enfoque más flexible y robusto. Esto puede incluir el uso de algoritmos de optimización estocástica, la consideración de múltiples escenarios y la implementación de mecanismos de reserva para compensar las fluctuaciones en la generación de energía.

### 2.1.3 *Aprendizaje automático y redes neuronales*

El Naqa y Murphy (2015) “What Is Machine Learning?”. Definen el aprendizaje automático o Machine Learning en inglés, como una rama en la evolución de los algoritmos computacionales que están diseñados basados en emular la inteligencia humana, aprendiendo del entorno circundante. Las técnicas basadas en el aprendizaje automático se han aplicado con éxito en diversos campos que van desde el reconocimiento de patrones, visión por computadora, series temporales, entre otros.

#### *2.1.3.1. Neurona biológica*

Tino et al. (2015) “Springer Handbook of Computational Intelligence”. Afirma que existe $\mathbf{10^{12}}$ células neuronales (neuronas) en el cerebro humano, donde dos tercios de las neuronas forman un córtex de 4 a 6 mm de grosor que se supone que es el centro de los procesos cognitivos. El cerebro es un conjunto de células cerebrales (neuronas) conectadas entre sí. Una neurona recibe impulsos eléctricos como señales, éstos procesan el mensaje recibido y lo transfieren a las siguientes neuronas.

Una neurona está compuesta de cuatro partes:

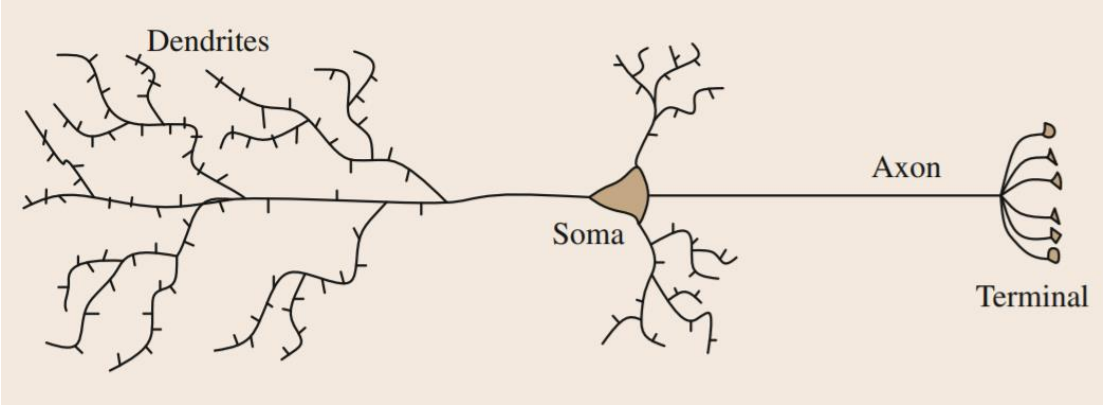
- Las dendritas aceptan los impulsos eléctricos de entrada.
- El soma procesa la información.

- El axón transforma la información de entrada para que sea recibida por la siguiente neurona.
- La terminal o sinápsis es el contacto electroquímico entre neuronas

Estos componentes se visualizan en la Figura 7.

**Figura 7**

*Neurona biológica.*




*Nota.* Adaptado de (Tino, Benuskova, & Sperduti, 2015).

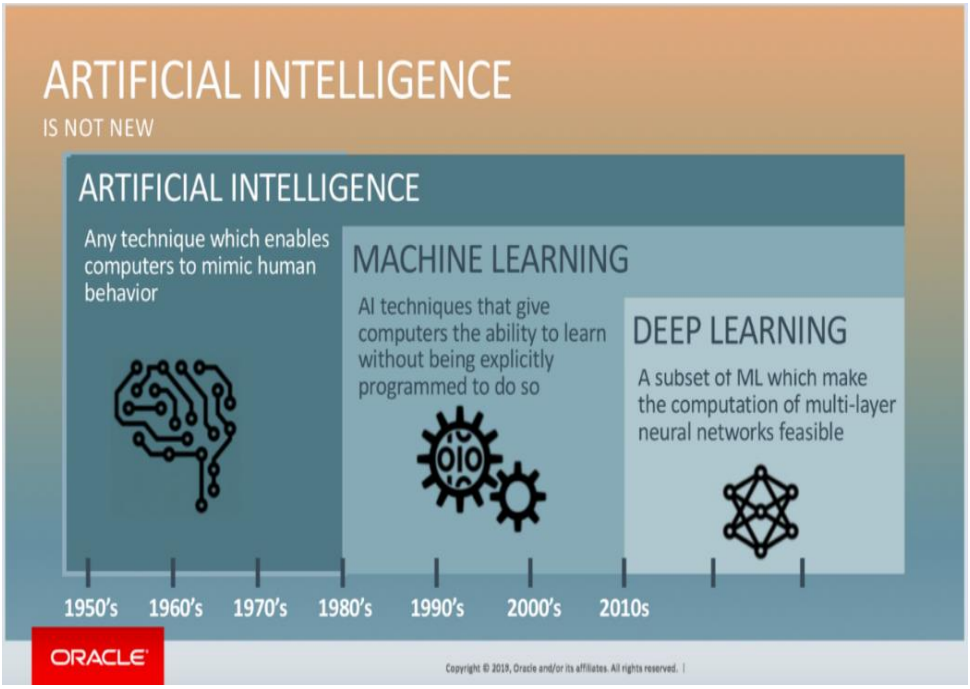
## *2.1.3.2. Deep learning (DL)*

Oracle (2018) “Oracle Big Data Blog”. Menciona que las redes neuronales, conocidas actualmente bajo el término paraguas de “deep learning”, proporciona uno de los mecanismos mediante el cual se consigue que un ordenador aprenda. DL es un subcampo del Aprendizaje Automático o Machine Learning (ML) en inglés y a la vez este es un subcampo de la Inteligencia Artificial (IA) como se detalla en la Figura 8.

La DL se ocupa de algoritmos inspirados en la estructura y función del cerebro, llamados redes neuronales. Por tanto, la DL utiliza las redes neuronales con más neuronas, capas e interconectividad.

**Figura 8**

Línea de tiempo de inteligencia artificial




*Nota. Adaptado de (Oracle, 2018).*

## *2.1.3.3. Algoritmo de aprendizaje*

Goodfellow et al. (2016) “Deep Learning”. Un algoritmo de aprendizaje automático es definido como un algoritmo que es capaz de aprender de los datos. Mitchell (1997) ofrece la siguiente definición: "Se dice que un programa informático aprende de la experiencia E con respecto a una clase de tareas T y una medida de rendimiento P, si su rendimiento en las tareas de T, medido por P mejora con la experiencia E".

En términos de aprendizaje automático, las tareas suelen describirse como la manera en que el sistema de aprendizaje automático procesa un ejemplo. Un ejemplo es una colección de características que son medidos cuantitativamente de algún objeto o evento que se pretende que el sistema de aprendizaje automático procese.

Tipos de algoritmos de aprendizaje automático

Antes de una explicación más profunda de los RNN, es necesario especificar los algoritmos de aprendizaje automático y los problemas que resuelven. Según su finalidad, el algoritmo de aprendizaje automático se divide en tres categorías:

- Aprendizaje supervisado: Entrena la máquina utilizando datos que están bien "etiquetados". Significa que algunos datos ya están etiquetados con la respuesta correcta y le ayuda a predecir los resultados de los datos imprevistos.
- Aprendizaje no supervisado: El aprendizaje no supervisado es una técnica de aprendizaje automático en la que no es necesario supervisar el modelo. En su lugar, debe permitir que el modelo funcione por sí solo para descubrir información. Se ocupa principalmente de los datos sin etiquetar.
- Aprendizaje reforzado: El aprendizaje por refuerzo es un método de entrenamiento de aprendizaje automático basado en recompensar los comportamientos deseados y / o castigar los no deseados. En general, un agente de aprendizaje por refuerzo es capaz de percibir e interpretar su entorno, tomar acciones y aprender mediante prueba y error.

En esta tesis, se utiliza el algoritmo de aprendizaje supervisado, por lo que se abordará más a fondo el algoritmo.

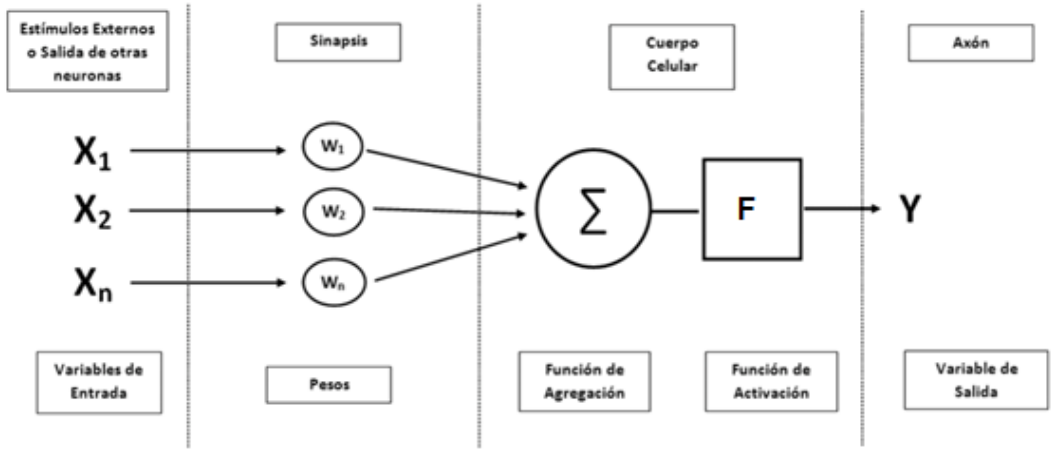
### 2.1.4 *Redes neuronales artificiales (ANN)*

Rico, C. et al (2009) “Modeling of the Hierarchical Structure of Freshwater Macroinvertebrates Using Artificial Neural Networks”. Una red neuronal (NN) es un modelo de aprendizaje automático inspirado en el funcionamiento y la estructura de un cerebro biológico, pero más simplificado.

El modelamiento de la red neuronal se observa en la Figura 9*Nota.* Adaptado de, donde cada neurona recibe y combina estímulos externos y/o salidas de otras neuronas, expresada bajo variables ($\boldsymbol{Xi}$), consecutivamente, multiplicados por una proporción o peso ($\boldsymbol{Wi}$) que indica la fuerza de la conexión sináptica entre las entradas y las dendritas de las neuronas. Posteriormente, las señales ponderadas llegan al cuerpo celular del modelo, en el que son primero acumulados a través de una suma ($\sum$) y posteriormente valorados a través de una función de activación ($\boldsymbol{f}$) que define el estado inhibido o excitado de la neurona; a partir de allí se da la trasmisión del resultado o salida a través del axón de la neurona ($\boldsymbol{Yi}$) hacia otras.

**Figura 9**

Modelo de neuroan artificial




*Nota.* Adaptado de (Rico, Paredes, & Fernández, 2009).

$$y(x) = f(w \odot x + b) \qquad (2.1)$$

## *2.1.4.1. Funciones de activación:*

(Siddharth, Simone, & Anidhya, 2020) Son funciones que se encuentran previa a la salida de la red neuronal cuya función es limitar los valores de salida en un rango finito. Las funciones de activación típicas incluyen: sigmoide logístico (σ), tanh y unidades lineales rectificadas (ReLU).

Las funciones están definidas por las ecuaciones 2.2, 2.3, 2.4 y 2.5 respectivamente.

$$\sigma_{(z)} = \frac{1}{1+e^{-z}} \quad (2.2)$$

$$tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}} \quad (2.3)$$

$$ReLU\ (z) = max(0, z) \quad (2.4)$$

$$a(z) = z \quad (2.5)$$

### *2.1.4.2. Función Costo*

Es una función que determina qué tan bien se desempeña un modelo de aprendizaje automático para un conjunto de datos determinado. La función de costo calcula la diferencia entre los valores anticipados y esperados y la muestra como un solo número real. Las funciones de coste crean en una variedad de métodos dependiendo de la situación. Para estimar qué tan mal funcionan los modelos, se emplean funciones de costo. La función de costo es una medida de la separación entre los valores esperados y reales.

Mínimo: cuando un valor se reduce a su forma más simple, se denomina costo, pérdida o error. El objetivo es identificar la configuración de los parámetros del modelo para los que la función de coste proporciona el menor número posible.

Máximo: cuando algo se maximiza, el valor que produce se denomina recompensa. El objetivo es descubrir los valores de los parámetros del modelo con un número devuelto tan grande como sea posible.

Algunas funciones de costo se mencionan:

- Mean Squared Error
- Hinge loss
- Mean Absolute Error (MAE)

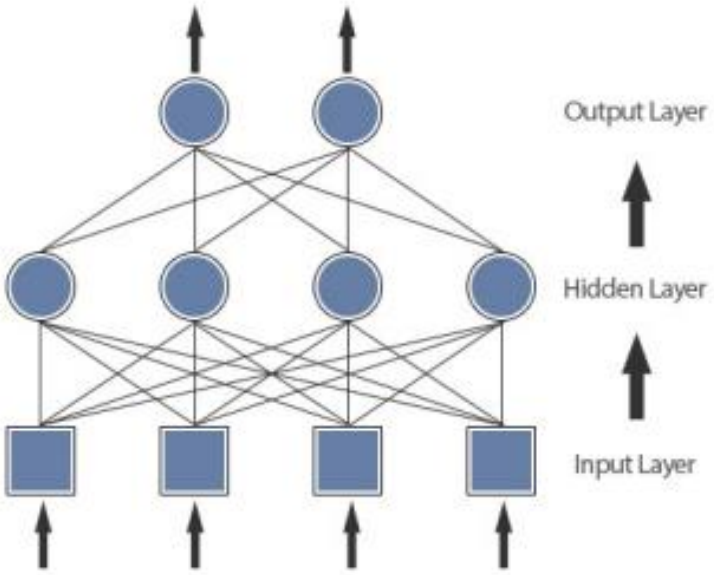
- Cross Entropy

### *2.1.4.3. Forward Propagation*

Lee, J. (2013) “Introduction to Artificial Neural Networks - Part 1”. La red básica feed-forward se muestra en la Figura 10 y consta de varias neuronas, organizadas en capas para formar una red. Las neuronas de cada capa están conectadas a todas las neuronas de la capa anterior mediante un conjunto de bordes dirigidos. Cada borde adquiere un peso correspondiente asociado. La primera capa recibe la entrada y se llama capa de entrada. La última capa denominada capa de salida produce la salida de la NN. Las capas restantes se denominan colectivamente capas ocultas. Dado que el flujo de información es de la capa de entrada a la capa de salida, una jerarquía está implícita en la estructura de la capa. La capa de entrada también se conoce como la capa inferior y la capa de salida como la capa superior. Las salidas de cada neurona se calculan mientras se mueven hacia arriba desde la capa inferior hasta que la salida de la red se produce en la capa superior.

**Figura 10**

Red feed forward con una capa oculta.



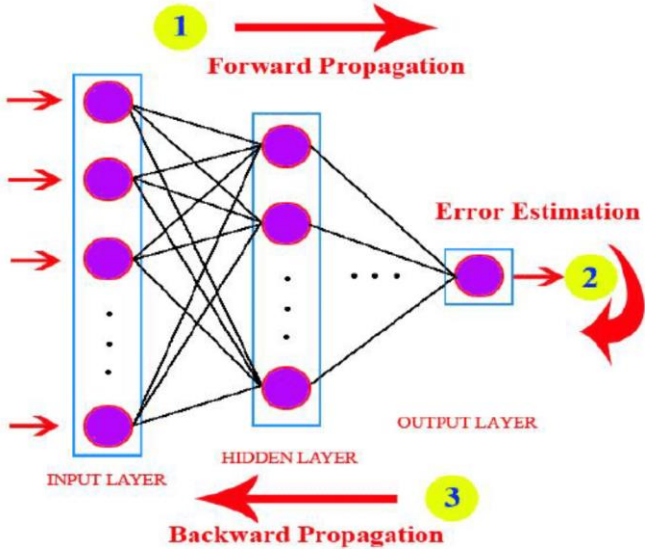

*Nota.* Adaptado de (Lee J., 2013).

### *2.1.4.4. Backward Propagation*

En el campo del aprendizaje automático, la Propagación Hacia Atrás (Backward Propagation) se destaca como un algoritmo esencial para el entrenamiento. Tras la ejecución de la red en modo feed-forward, se obtiene una salida prevista que se coteja con la salida deseada. A partir de esta comparación, se determina la pérdida total, que indica si el modelo está adecuadamente calibrado. Si no lo está, el valor de pérdida se emplea para reajustar los pesos en la siguiente iteración. Este proceso de reajuste de pesos se optimiza eficientemente mediante la Propagación Hacia Atrás, esto se evidencia en la Figura 11.

**Figura 11**

Arquitectura de ANN de feed'forward de tres capas con backward- propagation.




*Nota.* Adaptado de (Lee, J. ,2013).

### *2.1.4.5. Gradiente descendente*

Ray, S. (2019) "A Quick Review of Machine Learning Algorithms". Gradiente Descendente es un método iterativo en el que el objetivo es minimizar la función de coste. Se utiliza para identificar la menor cantidad de inexactitud en un modelo. Gradiente Descendente es considerada como el camino a recorrer para cometer la menor cantidad de errores. La

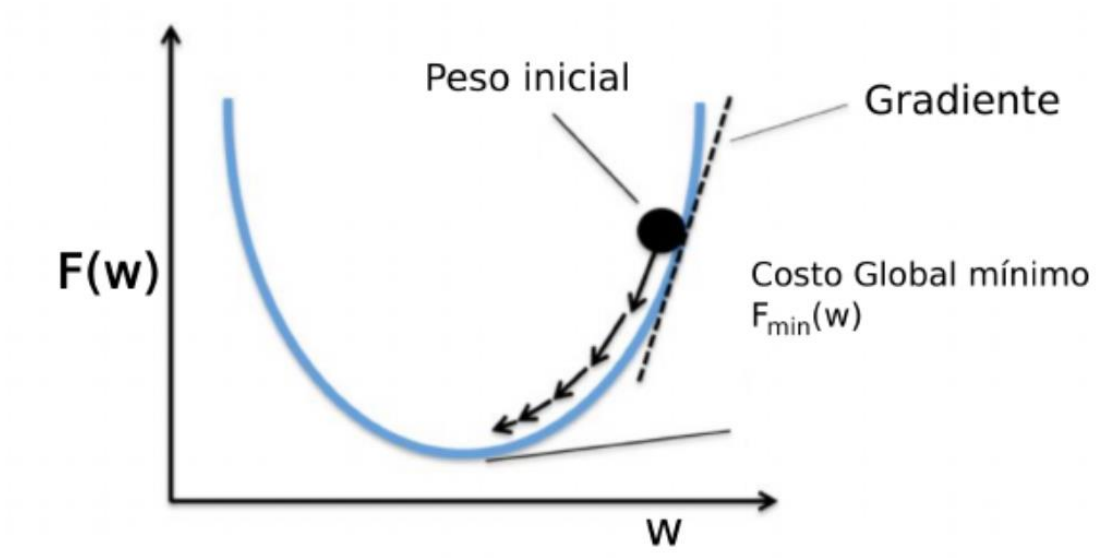
inexactitud del modelo puede variar en diferentes lugares, y debe descubrir el método más rápido para disminuirla a fin de evitar el desperdicio de recursos.

Debe ser posible calcular la derivada parcial de la función de coste que es la pendiente o gradiente. Los coeficientes se calculan en cada iteración tomando el negativo de la derivada y reduciendo los coeficientes en cada paso por una tasa de aprendizaje (tamaño del paso) multiplicado por la derivada, de modo que los mínimos locales se alcancen después de algunas iteraciones. Así que finalmente las iteraciones se detienen cuando se converge al valor mínimo de la función de coste después de lo cual no hay más reducción de la función de coste función. Existen tres diferentes métodos:

1. Gradiente descendente estocástica (SGD), ver Figura 12.
2. Gradiente descendente batch (BGD)
3. Gradiente descendente minibatch (MBGD)

**Figura 12**

Gradiente descendiente.




*Nota*: Busca la dirección hacia donde la función F(w) decrece más rápido. Las flechas que indican desplazamiento representan la tasa de aprendizaje.

### *2.1.4.6. Parámetros*

Los hiperparámetros principales de las redes neuronales LSTM, que controlan el algoritmo son:

- ✓ Tasa de aprendizaje: Controla la magnitud de los ajustes realizados durante el proceso de entrenamiento.
- ✓ long_sec: Longitud de la secuencia de entrada en pasos de tiempo.
- ✓ n_preds: Número de predicciones que se desean realizar.
- ✓ n_units: Número de unidades en la capa LSTM.
- ✓ n_units_hidden: Número de unidades en la capa oculta LSTM.
- ✓ epoch_number: Número de épocas para el entrenamiento, es decir, el número de veces que el modelo recorre todo el conjunto de entrenamiento.
- ✓ batch_size: Tamaño del lote utilizado para el entrenamiento, es decir, el número de muestras de entrenamiento que se utilizan en cada paso de actualización de los pesos del modelo.
- ✓ optimizer: Tipo de optimizador y tasa de aprendizaje utilizados para ajustar los pesos del modelo durante el entrenamiento.
- ✓ metrics: Métricas utilizadas para evaluar el rendimiento del modelo, como el error absoluto medio (mean absolute error).
- ✓ loss: Función de pérdida utilizada durante el entrenamiento para medir la discrepancia entre las predicciones del modelo y los valores reales.

Estos parámetros se ajustan para optimizar el rendimiento del modelo y deben ser explorados y probados para encontrar la configuración óptima en función de los requisitos y características del problema.

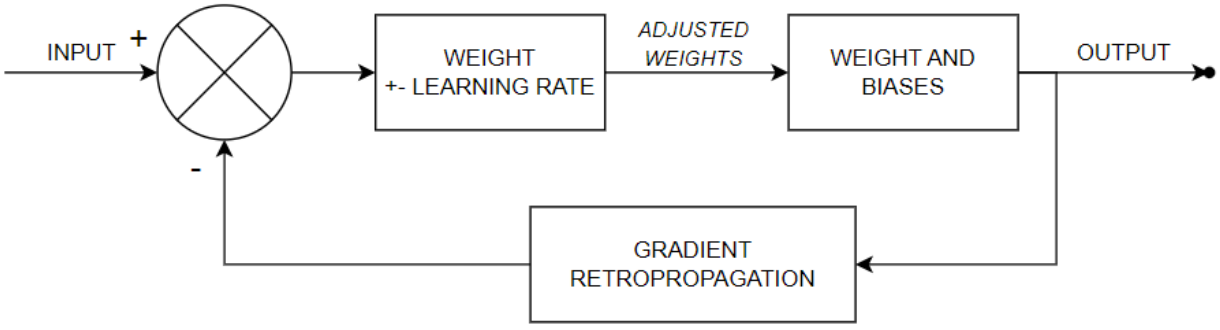
### *2.1.4.7. Entrenamiento de NN*

Queguiner, J. (2020). “What does Training Neural Networks mean?” El problema de aprendizaje se convierte en un ejercicio de optimización (minimización de errores) con el objetivo de minimizar la función de costo ajustando los parámetros del NN. El algoritmo de optimización utilizado para entrenar NN se llama descenso de gradiente. El descenso de gradientes implica calcular los gradientes de la función de costo con respecto a los parámetros de la red, es decir, pesos y sesgos. Se utiliza un valor escalar llamado tasa de aprendizaje (γ) para actualizar los parámetros (θ) en la dirección opuesta del gradiente, de acuerdo con la ecuación 2.8. El proceso se realiza de forma iterativa haciendo varias pasadas sobre los datos de entrenamiento. Un traspaso de datos de entrenamiento se denomina época y después de cada época los parámetros se acercan a sus valores óptimos, lo que minimiza la función de pérdida.

$$\theta = \theta - \gamma \frac{\partial L(\theta)}{\partial \theta} \qquad (2\text{-}8)$$

**Figura 13**

Paralelo entre sistemas de control y el proceso de entrenamiento de la red neuronal.




*Nota. Adaptado de (*Queguiner, J., 2020).

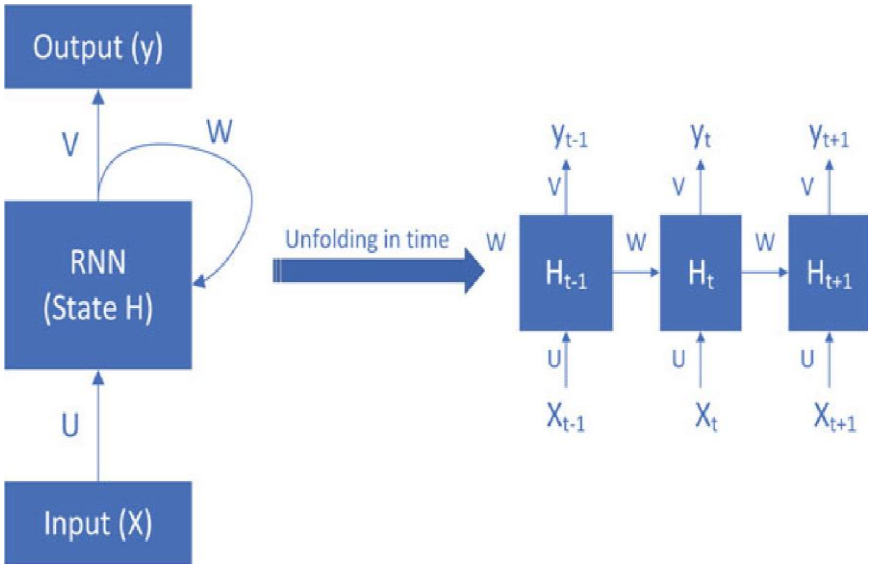
### 2.1.5 *Redes Neuronales Recurrentes*

Ameet, V. Joshi (2020) “Machine Learning and Artificial Intelligence”. Las redes neuronales recurrentes (RNN) son modelos de redes neuronales artificiales (ANN), propuestos en la década de 1980 (Rumelhart et al., 1986; Elman, 1990; Werbos, 1988). Las RNN son modelos que se adaptan mejor al procesamiento de datos secuenciales. Los datos de una serie o secuencia se observan en múltiples tipos de problemas, con datos de series de tiempo o secuencia de datos con propiedades repetitivas y correlacionadas.

La diferencia crucial entre las redes neuronales Feedforward (FNN) y las RNN es que las RNN cuentan con una memoria en la que almacenan información calculada a partir de entradas anteriores, la última salida está influenciada no solo por la entrada anterior sino por todas las entradas que se alimentaron en la red. En **Figura 14** la red neuronal recurrente (RNN) se esquematiza gráficamente por una retroalimentación de estados internos.

**Figura 14**

Arquitectura de RNN. El esquema de tiempo desplegado como se actualiza el estado de RNN en funcion de entradas secuenciales.




*Nota. Adaptado de (Ameet, 2020)*

La salida de la RNN es expresada bajo la siguiente ecuación:

$$y_t = f_y(V \cdot H_t) \qquad (2.9)$$

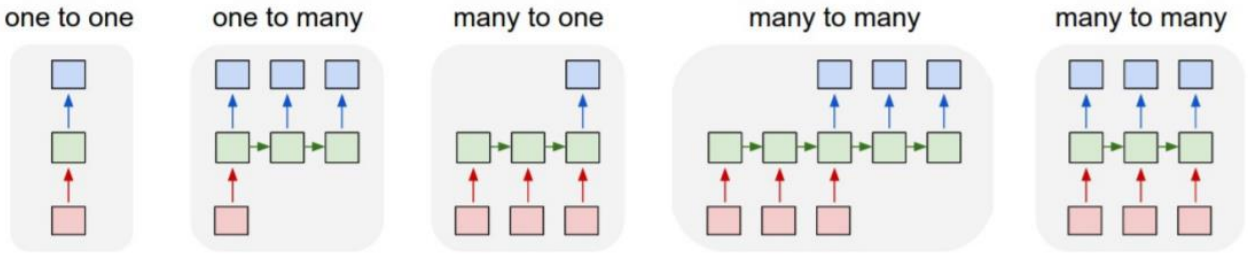
El estado de la RNN se actualiza en cada instancia de tiempo utilizando la entrada y el estado anterior, esto se define de la siguiente manera:

$$H_t = f_H(U \cdot X_t + W \cdot H_{t-1}) \qquad (2.10)$$

La **Figura 15**,*Nota. Adaptado* muestra algunos ejemplos de cómo se diseña la estructura de la red, dependiendo de si la entrada o la salida (o ambas) es una secuencia. Los rectángulos rojos representan vectores de entrada, los rectángulos azules representan vectores de salida y los rectángulos verdes representan bloques RNN (ocultos). Hay un flujo de datos no solo desde la capa de entrada a través de la capa oculta hasta la capa de salida, sino que las flechas verdes representan el flujo de influencia entre el bloque RNN y sus sucesores.

**Figura 15**

Tipos de estructura de RNN.




*Nota. Adaptado. (Karpathy, 2015).*

### 2.1.6 *Modelos de Redes Neuronales Recurrentes para pronóstico de energía eólica*

Yaghoubirad et al. (2023) “Deep learning-based multistep ahead wind speed and power generation forecasting using direct method”. En el contexto de la energía eólica, los Modelos de Redes Neuronales Recurrentes (RNN) son herramientas avanzadas de aprendizaje automático que pueden procesar datos secuenciales, como las series temporales de las condiciones meteorológicas, y hacer pronósticos precisos de la producción de energía eólica. Estos modelos, que incluyen variantes como Long Short-Term Memory (LSTM) y Gated Recurrent Units (GRU), son capaces de capturar las dependencias temporales en los datos, lo que es fundamental para predecir la producción futura de energía basada en las condiciones actuales y pasadas. Su uso en la predicción de la energía eólica es un área de investigación activa y emergente, que promete mejoras significativas en la eficiencia y la fiabilidad de la generación de energía eólica.

### 2.1.7 *Long short-term memory (LSTM)*

Zadranská, Lada (2019) “Time Series Forecasting using Deep Neural Networks”. Long Short-Term Memory (LSTM) son una clase de Redes Neuronales Recurrentes (RNN) diseñadas para manejar datos de series temporales. Estas redes fueron introducidas por Hochreiter y Schmidhuber en 1997 y han sido perfeccionadas en los años subsiguientes.

Las LSTM se han estudiado como una solución al problema del gradiente de fuga o "vanishing gradient" (un obstáculo para el aprendizaje de largas secuencias de datos) que presentan las RNN estándar (Hochreiter, 1998). Estas redes tienen la capacidad de preservar el error a través del tiempo y las capas, sin el riesgo de perder información relevante. Este

logro es posible gracias a mecanismos internos conocidos como puertas y estados de las células, que regulan el flujo de información.

Existen diversas variantes de modelos LSTM que se utilizan para diferentes problemas de pronóstico de series temporales. Según Brownlee (2019), estos modelos se pueden clasificar en cuatro categorías:

1. Modelo LSTM univariable
2. Modelo LSTM Multivariable
3. Modelo LSTM univariante Multi-step
4. Modelo LSTM Multivariable Multistep.

### *2.1.7.1. Arquitecturas LSTM*

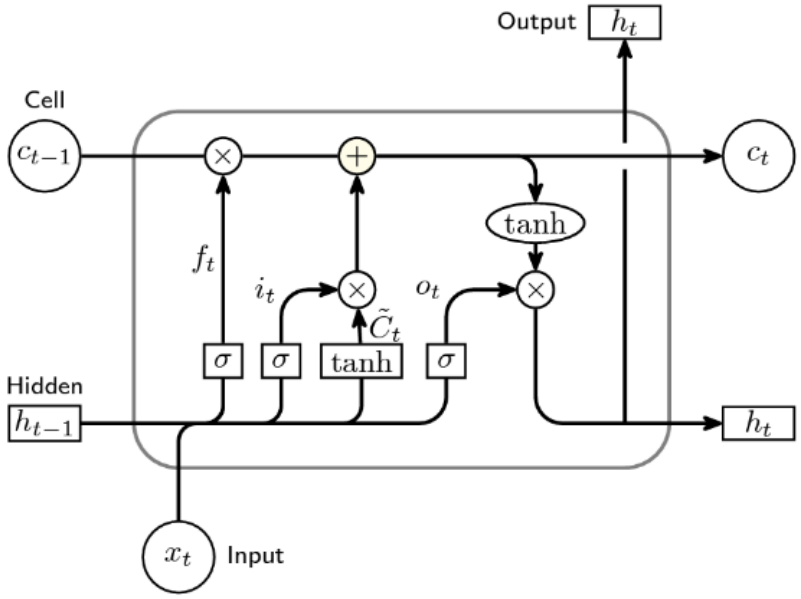
Fan et al. (2020) “Comparison of Long Short Term Memory Networks and the Hydrological Model in Runoff Simulation “. La arquitectura LSTM (Long Short-Term Memory) se distingue por la inclusión de tres puertas fundamentales: la puerta de entrada, la puerta de olvido y la puerta de salida. Cada una de estas puertas juega un papel estratégico en el manejo de la información a través de la red.

La puerta de entrada es la responsable de regular la incorporación de nuevas entradas a la memoria celular. Por otro lado, la puerta de olvido controla la eliminación de información irrelevante de la memoria celular. Finalmente, la puerta de salida determina qué información de la memoria celular se emitirá en la salida.

La Figura 16 proporciona una visualización gráfica de las tres puertas sigmoideas presentes en la arquitectura LSTM, proporcionando una representación intuitiva de sus roles respectivos en la gestión y manipulación de la información a través de la red.

**Figura 16**

Arquitectura de una célula de red LSTM




*Nota.* Adaptado de (Fan et al., 2020).

La clave de LSTM es el estado de la celda ($\boldsymbol{C_t}$), que permite que la información fluya sin cambios. El estado de la celda está regulado por tres puertas para permitir el paso de información opcionalmente.

- ✓ La primera puerta se llama puerta de olvido y controla qué elementos del vector de estado de celda $\boldsymbol{C_{t-1}}$ se olvidarán.

$$ft = \sigma(Wf \cdot [ht-1, xt] + bf) \qquad (2\text{-}11)$$

Donde $\boldsymbol{f_t}$ es un vector de salida de la capa sigmoidea con valores que van de 0 a 1, lo que indica el grado olvidado. $\boldsymbol{W_f}$ y $\boldsymbol{b_f}$ definen el conjunto de parámetros entrenables para la puerta de olvido.

- ✓ La segunda puerta se llama puerta de entrada donde se decide qué valor se actualizará:

$$it = \sigma(Wi \cdot [ht - 1, xt] + bi) \qquad (2\text{-}12)$$

Donde $\boldsymbol{it}$ es una variable de salida con un valor que va de 0 a 1. $\boldsymbol{W_i}$ y $\boldsymbol{b_i}$ son parámetros entrenables. Luego, un vector potencial del estado de la celda se calcula mediante la entrada actual ($\boldsymbol{x_t}$) y el último estado oculto $\boldsymbol{h_{t-1}}$.

$$\tilde{C} = tanh(WC \cdot [ht - 1, xt] + bC) \qquad (2\text{-}13)$$

Donde $\widetilde{\boldsymbol{C}}$ es un vector con valores que van de 0 a 1, $\boldsymbol{tanh}$ es la tangente hiperbólica y $\boldsymbol{W_C}$ y $\boldsymbol{b_C}$ son los parámetros entrenables.

Después de eso, se actualiza el estado de celda antiguo $\boldsymbol{C_{t-1}}$al nuevo estado de celda $\boldsymbol{C_t}$ mediante la multiplicación de elementos:

$$Ct = ft * Ct - 1 + it * C\tilde{}t \qquad (2\text{-}14)$$

- ✓ Finalmente, la puerta de salida decide cuál será la salida de una capa sigmoidea:

$$Ot = \sigma(Wo[ht - 1, xt] + bo) \qquad (2\text{-}15)$$

donde $\boldsymbol{O_t}$es un vector con valores que van de 0 a 1. Wo y bo son parámetros entrenables definidos para la puerta de salida. El nuevo estado oculto $\boldsymbol{ht}$ se calcula combinando las ecuaciones *2-14 y 2-15*.

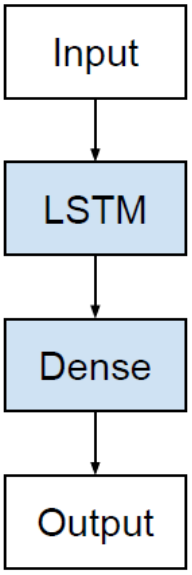
$$ht = ot * tanh(Ct) \qquad (2\text{-}16)$$

## *2.1.7.2. Arquitectura estándar LSTM*

La arquitectura estándar de la red LSTM se estructura en tres capas fundamentales: una capa de entrada, una capa LSTM oculta y una capa de salida. Como ilustra la Figura 17.

**Figura 17**

*Arquitectura estándar*



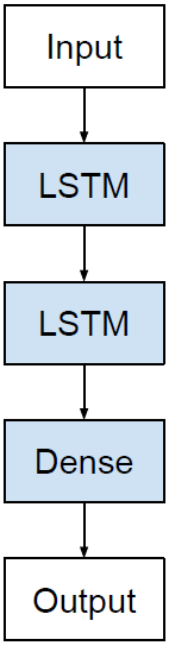

*Nota*. Adaptado de (Brownlee, 2017).

### *2.1.7.3. Arquitectura Stacked LSTM*

La Arquitectura Stacked LSTM, también conocida como LSTM apilada, es una variante de la red LSTM que incorpora múltiples capas de unidades LSTM. Cada capa de unidades LSTM transmite su salida a la siguiente capa, permitiendo a la red capturar relaciones más complejas en los datos de entrada. Esta arquitectura se diferencia al incluir una capa oculta LSTM adicional, como se ilustra en la Figura 18. Aunque esta arquitectura puede modelar relaciones más complejas, también es más susceptible al sobreajuste, requiere más datos para entrenarse de manera efectiva y es más intensiva en términos computacionales.

**Figura 18**

Arquitectura stacked




*Nota.* Adaptado de (Brownlee, 2017).

### *2.1.7.4. Entrenamiento de modelos LSTM*

El entrenamiento de un modelo de Redes de Memoria a Largo y Corto Plazo (LSTM, por sus siglas en inglés) consiste en ajustar los pesos del modelo para minimizar una función de pérdida específica, un proceso que incluye dos etapas clave: Feedforward y Backpropagation (Goodfellow et al., 2016).

En la fase de Feedforward, durante cada epoch, se suministran los datos de entrada al modelo LSTM. Posteriormente, se determina la pérdida, que es una medida cuantitativa de cuánto se desvían las predicciones del modelo de los valores verdaderos.

En la fase subsiguiente de Backpropagation, se usa un algoritmo específico para calcular los gradientes de la pérdida con respecto a los pesos del modelo. Entonces, los pesos se ajustan en la dirección que reduce la pérdida.

Este ciclo de Feedforward y Backpropagation se repite durante múltiples epochs, donde un epoch se define como una pasada completa a través de los datos de entrenamiento. A lo largo de este proceso de entrenamiento, es común utilizar un conjunto de validación para monitorear el rendimiento del modelo. Este conjunto permite ajustar los hiperparámetros del modelo, como la tasa de aprendizaje y los parámetros de regularización, para optimizar el rendimiento.

Las LSTM, como una variante de las Redes Neuronales Recurrentes (RNN), se han mostrado especialmente útiles para tareas de modelado de secuencias, como el procesamiento del lenguaje natural y la predicción de series temporales, debido a su capacidad para aprender dependencias a largo plazo (Hochreiter & Schmidhuber, 1997).

### 2.1.8 *Hiperparámetros en LSTM*

Los hiperparámetros son las variables necesarias para estimar mediante un algoritmo de optimización y se actualizan durante el entrenamiento, como se menciona en las Secciones 2.2 y 3 (ponderaciones y sesgos), los hiperparámetros no se estiman directamente a partir de los datos. Son externos al modelo y se configuran antes de que comience la fase de aprendizaje, aunque algunos de ellos se pueden ajustar durante el entrenamiento. La configuración de los hiperparámetros efectúa un gran impacto sobre si el modelo aprende, cómo y cuáles son las predicciones. En esta sección, se define los hiperparámetros que se utilizaran en el modelo definido en el Capítulo III.

#### *2.1.8.1. Numero de capas ocultas*

El número de capas ocultas en una red neuronal se refiere a las capas que están entre la capa de entrada y la capa de salida. Estas capas ocultas son donde se realiza la mayor parte del procesamiento a través de las conexiones ponderadas.

La elección del número de capas ocultas depende en gran medida de la complejidad del problema a resolver. Un problema simple puede requerir solo una o dos capas ocultas, mientras que un problema más complejo puede requerir varias capas ocultas. Sin embargo, agregar más capas ocultas no siempre mejora el rendimiento del modelo y puede llevar a un sobreajuste si el modelo se vuelve demasiado complejo.

### *2.1.8.2. Número de neuronas ocultas*

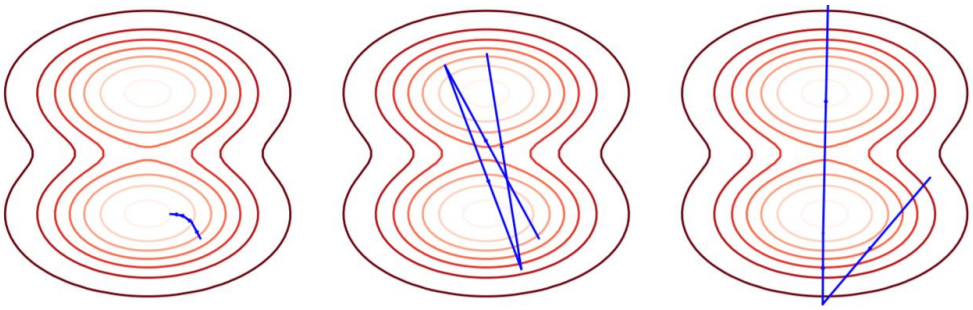
Como se evidencia en la investigación realizada por Zadranská (2019), la dependencia de este hiperparámetro varía en función del modelo y tipo de Red Neuronal Artificial (ANN) que se utilice. En el caso de las Redes Neuronales Recurrentes (RNN), existe cierta discrepancia en la definición de lo que constituye una capa oculta. Algunas fuentes sostienen que el número de capas ocultas en el modelo es equivalente a la cantidad de pasos de tiempo.

### *2.1.8.3. Learning rate*

Bengio, Y. (2012). “Practical Recommendations for Gradient-Based Training of Deep Architectures”, Menciona que el learning rate es uno de los hiperparámetro más importantes y siempre debe asegurarse de que se haya ajustado al modelo. Los valores típicos para una red neuronal con entradas estandarizadas (o entradas asignadas al intervalo (0,1)) son menores que 1 y mayores que 10−6, pero estos no deben tomarse como rangos estrictos y dependen en gran medida de la parametrización del modelo. Un buen valor por defecto y recomendado como partida es de 0.01.

**Figura 19**

Explicación de descenso gradual



*Nota.* De izquierda a derecha: 1) la tasa de aprendizaje es lo suficientemente pequeña para converger alrededor de un mínimo, 2) moderada para que rebote entre los mínimos, 3) demasiado grande para converger.

## *2.1.8.4. Función costo*

Según se destaca en la investigación de Zadranská (2019), la función objetivo es una medida que se busca maximizar o minimizar, y se emplea como evaluación de cuán bien un modelo se ajusta a los datos. Cuando se trabaja con datos etiquetados, es imprescindible minimizar la discrepancia entre la predicción del modelo y el valor real. Los parámetros de la función objetivo abarcan todos los parámetros que un modelo puede aprender. Dada la función objetivo L, donde θ representa el conjunto de todos los pesos y sesgos en el modelo, el objetivo es hallar estos pesos y sesgos de tal manera que se satisfaga la condición establecida por la Ecuación 2-17.

$$arg\,min_{\theta}\,(L(\theta)) \qquad (2\text{-}17)$$

En el contexto de la presente investigación, que se centra en un problema de optimización, se utiliza la Función de costo con el propósito de minimizarla. Para evaluar si un

modelo está aprendiendo, se aconseja presentar la función de costo en cada iteración y, por regla general, este valor debería disminuir.

En este tipo de problemas, las medidas para evaluar la calidad de los estimadores suelen ser la SSE (Suma de Errores al Cuadrado) (Ecuación 2.19) y el MSE (Error Cuadrático Medio).

$$SSE = \sum_{t=1}^{N}(y_t - \tilde{y}_t)^2 \qquad (2\text{-}18)$$

$$SSE = \frac{1}{N}\sum_{t=1}^{N}(y_t - \tilde{y}_t)^2 \qquad (2\text{-}19)$$

Dado que se busca determinar cuán distantes están en promedio los valores predichos de los valores reales.

### *2.1.8.5. Optimizador*

(Gupta, 2023) Un optimizador es una función o un algoritmo que modifica los atributos de la red neuronal, como los pesos y la tasa de aprendizaje. Por lo tanto, reduce la pérdida general y mejora la precisión.

Se utilizan diferentes optimizadores para realizar cambios en los pesos y tasa de aprendizaje, los cuales son mencionados:

1. Gradient Descent
2. Stochastic Gradient Descent
3. Stochastic Gradient descent with momentum
4. Mini-Batch Gradient Descent
5. Adagrad

6. RMSProp
7. AdaDelta
8. Adam

EL algoritmo de optimización afecta el modelo de DL en términos de precisión, velocidad y eficiencia. Por ello, plantea la necesidad de elegir un algoritmo de optimización adecuado para cada aplicación.

### *2.1.8.6. Mini batch size*

Como se constata en la investigación de Zadranská (2019). El “Batch Size” es un hiperparámetro de descenso de gradiente que controla la cantidad de muestras de entrenamiento para trabajar antes de que se actualicen los pesos y sesgos del modelo.

Se elige típicamente entre 1 y algunos cientos (256). El impacto que genera es principalmente computacional, es decir, un Batch Size más grande produce un cálculo más rápido (con implementaciones apropiadas) pero requiere visitar más ejemplos para alcanzar el mismo error, debido a que hay menos actualizaciones por época. En teoría, este hiperparámetro debería afectar el tiempo de entrenamiento y no tanto el rendimiento de la prueba, por lo que se puede optimizar por separado de los otros hiperparámetros, comparando las curvas de entrenamiento (error de entrenamiento y validación frente a la cantidad de tiempo de entrenamiento), después de otro.

### *2.1.8.7. Numero de épocas*

El número de épocas es un hiperparámetro de gradiente de descenso que controla el número de pases completos a través del conjunto de datos de entrenamiento.

Una época significa que cada muestra del conjunto de datos de entrenamiento ha tenido la oportunidad de actualizar los parámetros internos del modelo (sesgo y pesos). Una época se compone de uno o más Batch Sizes.

El número de épocas es tradicionalmente grande, a menudo cientos o miles, lo que permite que el algoritmo de aprendizaje se ejecute hasta que el error del modelo se haya minimizado lo suficiente.

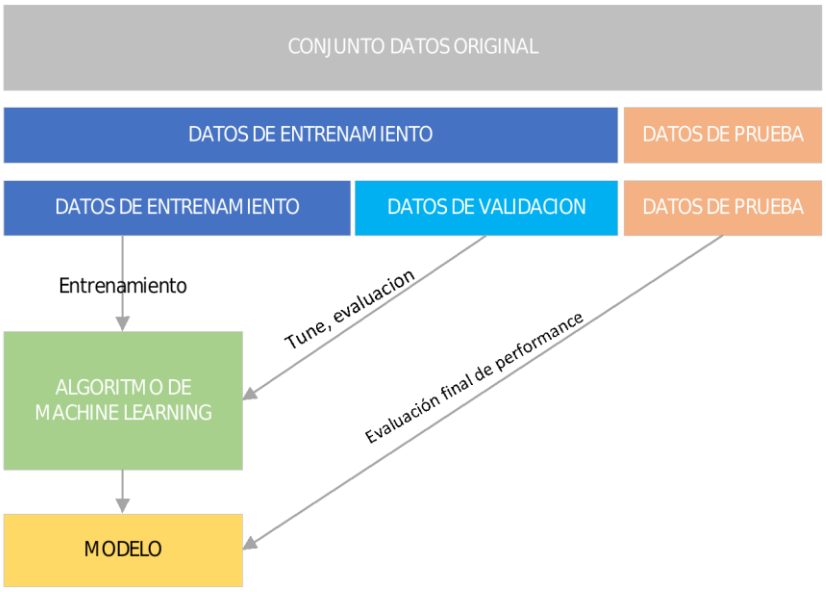
### 2.1.9 *Validación cruzada*

Sidhu (2020) “Use of Cross Validation in Machine Learning”. Destaca la validación cruzada como una técnica esencial para evaluar los conjuntos de datos en modelos de aprendizaje automático. La relevancia de la validación cruzada radica en su capacidad para abordar el ajuste de los modelos. Aunque un modelo puede adaptarse adecuadamente a su conjunto de datos de entrenamiento, puede enfrentar desafíos al ser expuesto a datos inéditos o de distribuciones distintas. Estos desafíos se manifiestan en forma de sesgo, cuando el modelo no se ajusta adecuadamente a los datos, y varianza, cuando el modelo se ajusta en exceso.

En la Figura 20, se muestra como los datos de entrenamiento se utilizan para ajustar los parámetros del modelo seleccionado, mientras que el conjunto de datos de prueba está destinado a verificar qué tan bien se desempeña el modelo en los datos que no fueron utilizados durante el entrenamiento. A veces, también se utiliza un conjunto de datos de validación, construido a partir de los datos de entrenamiento. Este conjunto no se utiliza durante la fase de aprendizaje, está destinado a actualizar correctamente los hiperparámetros.

**Figura 20**

Partición de datos de analisis.




*Nota.* Elaboración propia.

## 2.1.10 *Descomposición en modo variacional (VMD)*

Dragomiretskiy y Zosso (2014) “Variational Mode Decomposition”. Los autores definen a la Descomposición en modo variacional (VMD), como la nueva técnica de procesamiento de señales desarrollada para descomponer la señal de datos original en una secuencia de submódulos discretos con cierta banda de frecuencia. Cada subcomponente se compacta en el área circundante de la frecuencia central en el proceso de optimización. Para lograr resultados de descomposición satisfactorios, se emplean los siguientes tres pasos en cada modo obtenido en VMD.

1. La transformación de Hibbert se utiliza para producir un espectro de frecuencia unilateral.
2. Se adopta un ajuste exponencial para mapear el espectro obtenido en la banda base.

3. Se adopta la suavidad gaussiana de la señal demodulada para calcular el ancho de banda

Entonces, el problema de variación restringida se expresa de la siguiente manera:

$$\min_{u_k(t),w_k}\left\{\sum_{k=1}^{R}\left\|\partial_t\left[\left(\delta(t)+\frac{j}{\pi t}\right)\otimes u_k(t)\right]e^{-jw_k t}\right\|_2^2\right\} \qquad (2\text{-}20)$$

Donde:

$$\text{subject to}\ \sum_{k=1}^{R}u_k(t)=f(t) \qquad (2\text{-}21)$$

$\boldsymbol{R}$ = número de modos.

$\boldsymbol{\partial_t}$ = derivada parcial de la función en base al tiempo

$\boldsymbol{\delta(t)}$= Distribución Dirac

$\boldsymbol{j}$ = unidad imaginaria

$\otimes$ = Operador convolucional

$\boldsymbol{u_k(t)}$ =$\boldsymbol{tth}$ data del $\boldsymbol{kth}$ del modo descompuesto

$\boldsymbol{w_k}$ = centro de frecuencia de la $\boldsymbol{kth}$ del modo descompuesto

$\|.\|_{\boldsymbol{p}}$ = norma de vector usual $\boldsymbol{l_p}$

$\boldsymbol{f(t)}$ = señal original

Generalmente, es difícil encontrar directamente la solución para el problema de optimización restringida en la Ecuación 2-20) debido a sus características complejas (como la

concavidad o convexidad) tanto en las funciones objetivas como en las restricciones. Con el fin de reducir la dificultad de la solución (Fang et al.2019), la función de penalización cuadrática que garantiza la fidelidad de la reconstrucción y el multiplicador de Lagrange cumple estrictamente las restricciones, se combinan para transformar el problema anterior en uno de optimización sin restricciones, que se describe de la siguiente manera:

$$Z(u_k(t), w_k, \lambda) = a\sum_{k=1}^{R}\left\|\partial_t\left[\left(\delta(t)+\frac{j}{\pi t}\right)\otimes u_k(t)\right]e^{-jw_k t}\right\|_2^2 + \left\|\sum_{k=1}^{R}u_k(t)-f(t)\right\|_2^2 + \left\langle \lambda(t), f(t)-\sum_{k=1}^{R}u_k(t)\right\rangle \qquad (2\text{-}22)$$

Donde:

Z: Función de Lagrange aumentada.

$\Lambda$: Multiplicador de Lagrange.

$a$: Parámetro de penalización.

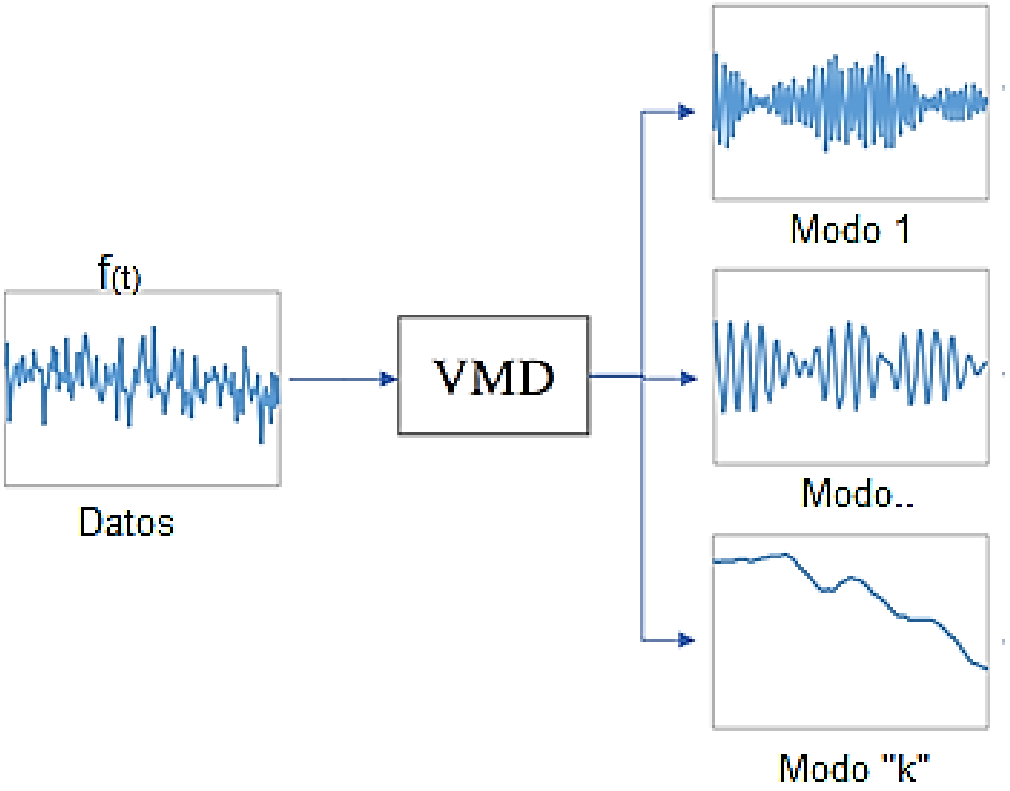
Luego, se adopta el método de dirección alternativa del método de multiplicadores para optimizar el problema no restringido modificado en la Ecuación 2-22. La solución correspondiente se expresa de la siguiente manera:

$$\hat{u}_k^{n+1}(w) = \frac{\hat{f}(w)-\sum_{i=1,i\neq k}^{R}\hat{u}_i(w)+0.5\hat{\lambda}^n(w)}{1+2a(w-w_k^n)^2} \qquad (2\text{-}23)$$

$$w_k^{n+1} = \frac{\int_0^\infty w|\hat{u}_k^{n+1}(w)|^2dw}{\int_0^\infty |\hat{u}_k^{n+1}(w)|^2dw}$$

**Figura 21**

Esquema de un modelo VMD.




*Nota.* Adaptado de (Han et al. ,2019)

## 2.1.11 *Pronósticos de series de tiempo*

Los pronósticos de series de tiempo se clasifican de la siguiente manera.

### *2.1.11.1. Clasificación de predicción de energía eólica según escalas de tiempo*

Chang, E. (2014) “A Literature Review of Wind Forecasting Methods”, Consultó varias literaturas disponibles y afirma que existe varios modelos de pronóstico según al horizonte de tiempo. Estos son agrupados en 4 categorías, ultracorto plazo, corto plazo, medio plazo y largo plazo, los cuales se muestran en la Tabla 2:

**Tabla 2**

Clasificación de horizontes de pronóstico.

| Escala de tiempo | Rango |
|---|---|
| Ultracorto plazo | Desde unos minutos hasta una hora |
| Corto plazo | Desde 1 hora hasta varias hora |
| Medio plazo | Desde varias horas hasta una semana |
| Largo plazo | De 1 semana a 1 año o más por delante |

*Fuente:* Adaptado de (Chang, 2014).

## 2.1.12 *Métricas de evaluación de modelos de pronóstico*

Hyndman & Koehler. (2006). "Another look at measures of forecast accuracy". Introduce la notación $\boldsymbol{Y_t}$ para denotar una observación en un tiempo específico $\boldsymbol{t}$ y $\boldsymbol{F_t}$ para representar el pronóstico correspondiente. La diferencia entre la observación real y su pronóstico se define como el error de pronóstico.

$$e_t = |Y_t - F_t| \qquad (2\text{-}24)$$

$\boldsymbol{Yt: Valor\ verdadero}$

$\boldsymbol{Ft: Valor\ de\ pronóstico}$

Los pronósticos pueden calcularse a partir de un tiempo base común y tener horizontes de pronóstico variables.

### *2.1.12.1. Error dependiendo de la escala*

Existen ciertas métricas de precisión cuya escala es directamente proporcional a la de los datos. Aunque resultan útiles al contrastar distintos métodos sobre un mismo conjunto de datos, no son apropiadas para comparar conjuntos de datos con escalas divergentes.

$$Error\ absoluto\ medio\ (MAE) = \ Media(|e_t|) \qquad (2\text{-}25)$$

$$Error\ cuadrático\ medio\ (MSE) = \ Media({e_t}^2) \qquad (2\text{-}26)$$

$$Raíz\ del\ Error\ Cuadrático\ Medio\ (RMSE) = \sqrt{MSE} \qquad (2\text{-}27)$$

### *2.1.12.2. Error porcentual*

El enfoque que utiliza errores porcentuales destaca por su consistencia independientemente de la escala, siendo frecuentemente adoptado para evaluar la precisión de pronósticos en diversas series de datos.

$$Porcentaje\ de\ error\ (p_t) = 100\frac{e_t}{Y_t} \qquad (2\text{-}28)$$

La exactitud de los pronósticos generados por diversos modelos se evalúa, en gran medida, a través del Error Porcentual Absoluto Medio (MAPE). Investigadores como Law (2000), Song et al. (2000) y Kon y Turner (2005) han utilizado este indicador.

El error porcentual absoluto medio se define como:

$$MAPE = \frac{100\%}{n} \sum \left| \frac{y - \widehat{y}}{y} \right| \qquad (2\text{-}29)$$

Donde $\widehat{\boldsymbol{y}}_{\boldsymbol{t}}$ denota el número previsto del modelo, $\boldsymbol{y}_{\boldsymbol{t}}$ representa el número real.

**Tabla 3**

Criterios para evaluar la precisión de un modelo de pronóstico utilizando MAPE.

| MAPE | PRECISIÓN DE PRONÓSTICO |
|---|---|
| **<10%** | Previsión de alta precisión |
| **10%~20%** | Buen pronóstico |
| **20%~50%** | Previsión razonable |
| **>50%** | Pronóstico débil e inexacto |

*Fuente:* Adaptado de (Lewis, 1982).

## 2.2 Marco conceptual

- Pronóstico: En el presente trabajo de investigación es el proceso de determinar un futuro evento basado en datos históricos.
- Normalizar: En el presente trabajo de investigación es el proceso que se encarga de reducir los rangos de los datos de entrada a la red neuronal entre [0,1] o [-1,1].
- Capa: En el presente trabajo de investigación se refiere a un conjunto de neuronas cuyas entradas provienen de una capa anterior (o de los datos de entrada en el caso de la primera capa) y cuyas salidas son la entrada de una capa posterior
- Memoria: En el presente trabajo de investigación es preservar un estado a través del tiempo. recordar información relevante sobre la entrada que recibieron, lo que les permite ser más precisas en la predicción de lo que vendrá después manteniendo información del contexto.

- Entrenamiento: En el presente trabajo de investigación consiste en ajustar cada uno de los pesos de las entradas de todas las neuronas que forman parte de la red neuronal, para que las respuestas de la capa de salida se ajusten lo más posible a los datos conocidos.

- Ajustar: En el presente trabajo de investigación significa modificar los valores de los parámetros e hiperparametros (definidos en la sección 2.1) con el fin de minimizar el error de predicción.

- Multistep: En el presente trabajo de investigación es la tarea de predecir una secuencia de valores en una serie de tiempo (predicción de múltiples etapas), es aplicar un modelo predictivo paso a paso y usar el valor predicho del paso de tiempo actual para determinar su valor en el siguiente paso de tiempo.

- Topología: La topología de una red neuronal representa la forma en que las neuronas se conectan para formar una red. En otras palabras, la topología de la red neuronal puede verse como la relación entre las neuronas mediante sus conexiones. La topología de una red neuronal desempeña un papel fundamental en su funcionalidad y rendimiento

- Etiquetado: En el presente trabajo de investigación es el proceso de identificar datos sin procesar (imágenes, archivos de texto, videos, etc.) y agregar una o más etiquetas significativas e informativas para proporcionar contexto para que un modelo de aprendizaje automático pueda aprender de ellos.

- Webscraping: Técnica utilizada para extraer contenido y datos de un sitio web.

- Ventaneo: Técnica de segmentar datos en "ventanas" consecutivas para entrenar o predecir con un modelo. Estas ventanas pueden desplazarse o expandirse a lo largo de la serie.

# Capítulo 3. Hipótesis y operacionalización de variables

## 3.1 Hipótesis

El desarrollo de un modelo de pronóstico de energía eólica a corto plazo basado en redes neuronales recurrentes de memoria a corto plazo (LSTM) y descomposición en modo variacional (VMD) mejorará la programación de generación de energía de centrales eólicas en comparación con los métodos actuales.

## 3.2 Variables e indicadores

Variable Independiente (VI): Modelo de pronóstico basado en redes LSTM y descomposición en modo variacional.

Variable Dependiente (VD): Precisión en la predicción de la generación de energía eólica.

**Tabla 4**

Relación de variables e indicadores.

| VARIABLES | INDICADORES |
|---|---|
| **VI: Modelo de pronóstico basado en redes neuronales LSTM y VMD.** | ✓ Tasa de error: Porcentaje de desviación entre las predicciones y los valores reales. MSE y MAE.<br>✓ Velocidad de convergencia del modelo: Tiempo requerido para que el modelo LSTM-VMD alcance una precisión deseada. |
| **VD: Precisión en la programación de generación de energía y lucro cesante asociado.** | ✓ Error de Pronóstico del Modelo Propuesto: Diferencia entre la energía pronosticada por el modelo LSTM-VMD y la energía realmente generada. Uso de MSE y MAE. |

| | |
|---|---|
| | ✓ Error de Pronóstico del Método Actual: Diferencia entre la energía pronosticada por el método actual y la energía realmente generada. Uso de MSE y MAE.<br>✓ Mejora en la Precisión: Diferencia entre el error de pronóstico del método actual y el error de pronóstico del modelo propuesto. |

Fuente: Elaboración propia.

# Capítulo 4. Metodología de La Investigación

## 4.1 Tipo y diseño de la investigación

El presente trabajo de investigación desde el punto de vista de enfoque, su naturaleza es cuantitativo, porque cumple las siguientes características:

- La percepción de la realidad posee objetividad porque el problema investigado no ha sido manipulado ni afectado por terceros.
- El razonamiento es deductivo porque se contrasta la hipótesis y los estudios previos presentados en la teoría.
- La finalidad sigue un patrón predecible y estructurado.
- Tiene una orientación de explicar y predecir el fenómeno estudiado.
- El principio de verdad es de validación, confiabilidad, y las conclusiones son derivadas de la generación de un nuevo conocimiento.
- La perspectiva del investigador ocurre en la realidad externa del investigador.

El tipo de esta investigación es explicativo, debido a que existe una relación entre la variable dependiente, modelo de pronóstico de energía, basado en redes neuronales recurrentes, y la variable independiente, programación de energía en centrales eólicas, además se identificará el grado de relación entre estas variables.

El diseño de la investigación es experimental, debido a que cuando se modifica el número de neuronas, épocas y número de componentes VMD de la variable independiente, se observará el efecto en la variable dependiente, programación de generación de energía en centrales eólicas.

## 4.2 Unidad de análisis

La presente investigación desarrolla como unidad de análisis al parque eólico de Talara. El parque se encuentra ubicado en la costa peruana, en el departamento de Piura, en la provincia de Pariñas, distrito de Talara. Se encuentra ubicado a 11 msnm, en la pampa denominada La Campana a 10 km de la ciudad de Talara, la **Figura 22** proporciona el plano ubicación del parque eólico. Esta central eólica de 30.86MW de potencia nominal, cuenta con 17 aerogeneradores de la marca VESTAS de 1.8MW cada uno.

**Figura 22**

*Plano de ubicación de PE Talara.*


*Nota.* Adaptado de (Osinergmin, 2018).

## 4.3 Matriz de consistencia

Desarrollo de un modelo de pronóstico de energía a corto plazo, basado en redes neuronales recurrentes, para mejorar la programación de generación de energía en centrales eólicas

**Tabla 5**

*Matriz de concistencia.*

| PROBLEMA | OBJETIVO | HIPÓTESIS | VARIABLES | INDICADOR | TECNICAS E INSTRUMENTOS DE RECOLECION DE DATOS |
|---|---|---|---|---|---|
| ¿Cómo mejorar la precisión del pronóstico de la generación de energía eólica utilizando un modelo basado redes neuronales recurrentes de memoria a corto plazo (LSTM) y descomposición en modo variacional (VMD), en comparación con los métodos actuales? | Desarrollar un modelo de pronóstico de energía eólica a corto plazo basado en redes neuronales recurrentes de memoria a corto plazo (LSTM) y descomposición en modo variacional (VMD) para mejorar la programación de generación de energía en centrales eólicas. | El desarrollo de un modelo de pronóstico de energía eólica a corto plazo basado en redes neuronales recurrentes de memoria a corto plazo (LSTM) y descomposición en modo variacional (VMD) mejorará la programación de generación de energía de centrales eólicas en comparación con los métodos actuales. | **Independiente:**<br><br>Modelo de pronóstico de energía a corto plazo basado en LSTM y VMD.<br><br>**Dependiente:**<br><br>Precisión en la programación de generación de energía en centrales eólicas. | - Error absoluto medio (MAE)<br>- Error cuadrático medio (MSE)<br>- Tiempo de convergencia. | - Código de programación de webscraping.<br>- Tablas de recolección de pronóstico actual (incluye variables meteorológicas y potencia cada 30min). Ver Anexo 01.<br>- Formato de reporte de generación de energía. Ver Anexo 02. |

*Nota*.: Elaboración propia

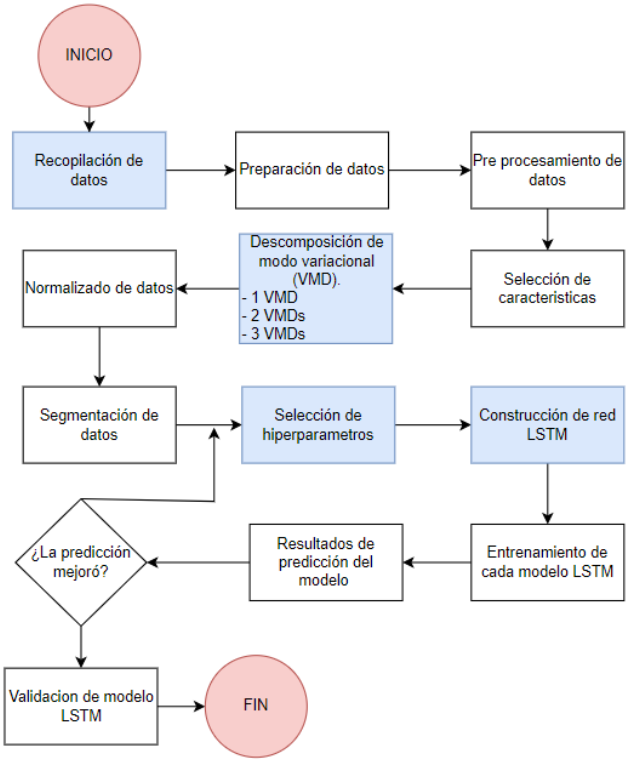
# Capítulo 5. Desarrollo del trabajo de investigación

A partir de la teoría establecida sobre redes neuronales, se procedió a la fase empírica de la investigación. Se extrajeron datos relevantes del sitio web del COES mediante técnicas avanzadas de webscraping. Estos datos fueron sometidos a Descomposición Modo Variacional (VMD). Luego, se implementó una Red Neuronal Recurrente del tipo LSTM para cada conjunto de datos resultante del VMD. Al concluir, las predicciones de las redes se consolidaron en un pronóstico integrado. La Figura 23 y predicción.

**Figura 24**, ilustra el cuerpo principal del capítulo, desde la recolección de datos hasta la evaluación final.

**Figura 23**

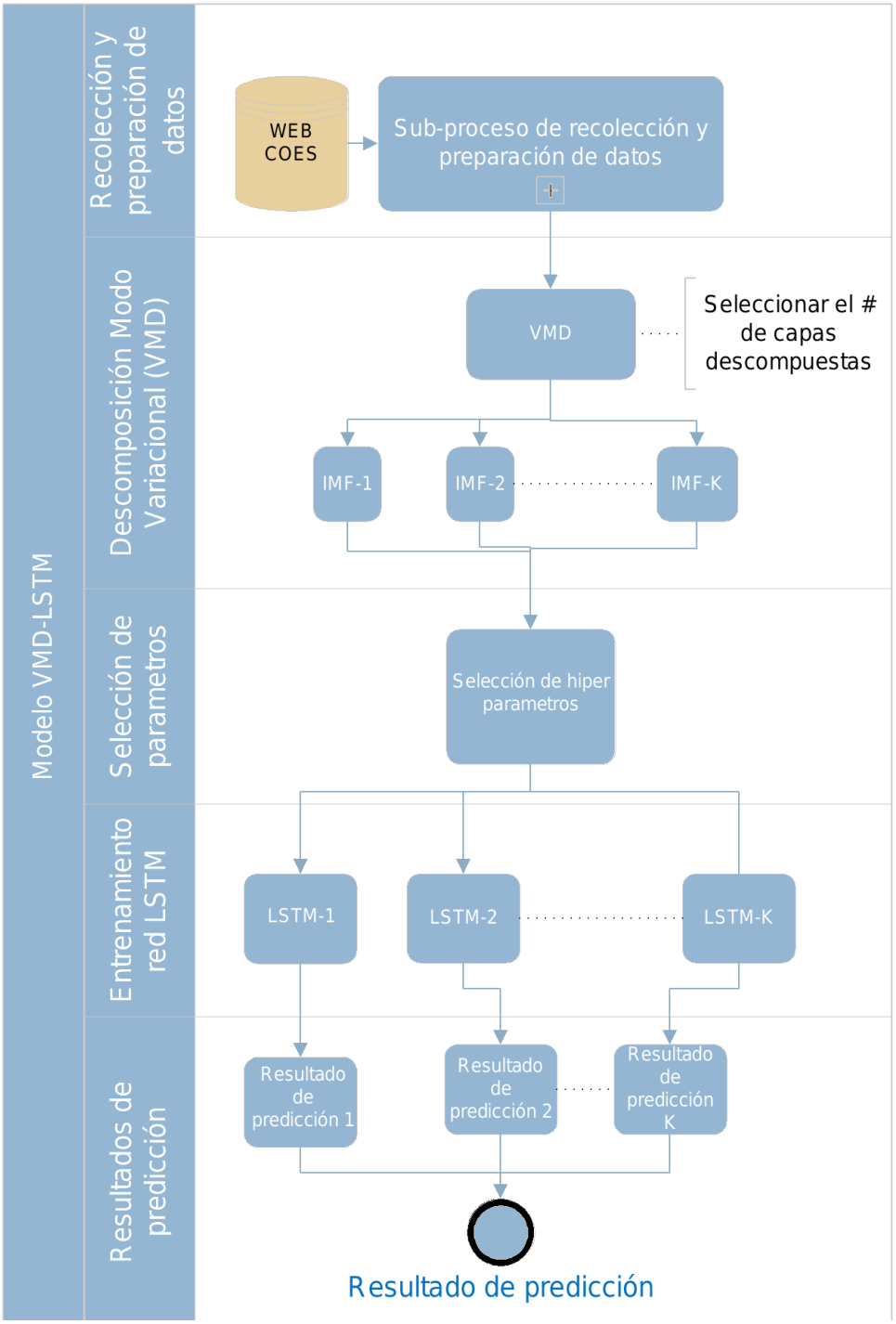
*Diagrama de flujo de desarrollo de trabajo de suficiencia*



La Figura 24, ilustra detalladamente la construcción interna del modelo para su entrenamiento y la posterior reconstrucción del resultado de predicción.

**Figura 24**

Flujo de proceso construcción del modelo propuesto

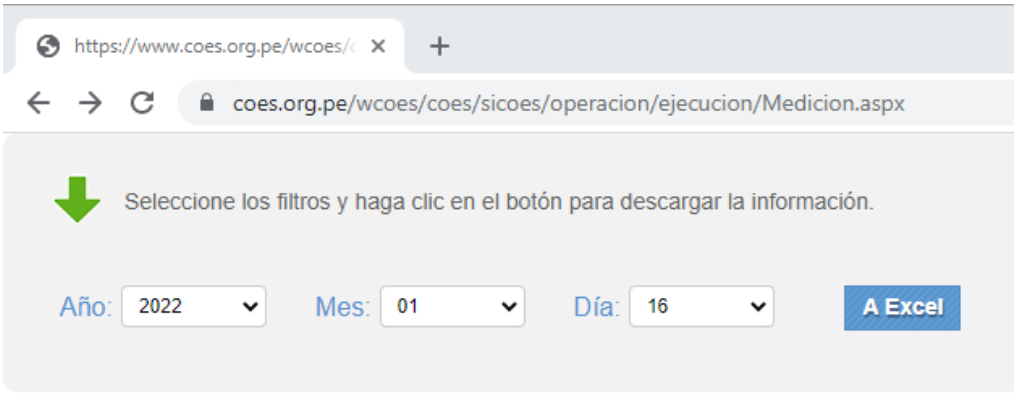


## 5.1 Recopilación y preparación de datos

Se recolectaron datos de la base del COES, ver Figura 25, un portal web que proporciona registros de diversas centrales de generación en Perú. Aunque la base ofrece principalmente datos de potencia, sería valioso contar con información adicional como velocidad y dirección del viento, humedad relativa, entre otros. No obstante, en este estudio, el foco principal será la potencia. Para la recolección masiva de esta información, se empleó la técnica de webscraping en el sitio web del COES. Los datos obtenidos tienen una resolución temporal de 30 minutos y abarcan desde el 01 de enero de 2016 hasta el 02 de julio de 2020.

**Figura 25**

*Sitio web del COES*




*Nota.* Sitio web utilizado para la descarga de datos históricos.

Después, se desarrolló un script en Python, empleando la librería Selenium, para llevar a cabo el webscraping

```
# Direccionamos hacia url y ejecutamos comandos
driver.get('https://www.coes.org.pe/wcoes/coes/sicoes/operacion/ejecucion/Medicion.aspx')
year_element=driver.find_element_by_id("DropDownListYears")
year=Select(year_element)
mes_element=driver.find_element_by_id("DropDownListMonths")
mes=Select(mes_element)
dia_element=driver.find_element_by_id("DropDownListDay")
dia=Select(dia_element)
link=driver.find_element_by_id("ButtonGenerarMedicion")
#Link.click()
```

```
rng = pd.date_range(inicio, periods=delta.days, freq='D')
df = pd.DataFrame({ 'Date': rng})
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day
df['Month'] = df['Date'].dt.strftime('%m')
df['Day'] = df['Date'].dt.strftime('%d')
```

```
# Ejecutamos ciclo for para descargar datos de periodo definido
for i in range(0, len(df), 1):
    year.select_by_visible_text(str(df.loc[ i,'Year' ]))
    mes.select_by_visible_text(str(df.loc[ i,'Month' ]))
    dia.select_by_visible_text(str(df.loc[ i,'Day' ]))
    link.click()
```

Se utilizó el siguiente código para generar la base de datos específica de nuestra unidad de estudio, el Parque Eólico de Talara. Los datos resultantes se almacenaron en un archivo denominado “WBS_Talara.csv”.

```
import pandas as pd
import glob

path = r'C:\Users\nisanchez\Desktop\Scrap-ejecutado' # use your path
all_files = glob.glob(path + "/*.csv")

li = []

for filename in all_files:
    df = pd.read_csv(filename, skiprows=2,index_col=None, header=0)
    li.append(df)

frame = pd.concat(li, axis=0, ignore_index=True)
```

```
df=frame[['Hora','PQE-EOLICO-TALARA']]
df.to_csv("WBS_Talara.csv")
```

| | Hora | PQE-EOLICO-TALARA |
|---|---|---|
| 0 | 2016-01-01 00:30:00 | 14.33674 |
| 1 | 2016-01-01 01:00:00 | 14.35624 |
| 2 | 2016-01-01 01:30:00 | 13.45760 |
| 3 | 2016-01-01 02:00:00 | 13.98033 |
| 4 | 2016-01-01 02:30:00 | 17.73316 |
| ... | ... | ... |
| 78859 | 2020-07-01 22:00:00 | 24.64462 |
| 78860 | 2020-07-01 22:30:00 | 24.59254 |
| 78861 | 2020-07-01 23:00:00 | 20.93133 |
| 78862 | 2020-07-01 23:30:00 | 17.81378 |
| 78863 | 2020-07-02 00:00:00 | 10.12704 |

78864 rows × 2 columns

### 5.1.1 *Preparación de datos*

La preparación de los datos siguió cuatro pasos principales: la etapa de pre procesamiento, paso de caracterización, fase de descomposición y finalmente escalamiento de características para posterior entrenamiento del modelo, tal como se muestra en la Figura 26.

1. Etapa de preprocesamiento: En esta etapa se llevaron a cabo tareas como la limpieza y selección de datos, la eliminación de valores atípicos y la eliminación de variables irrelevantes.

2. Paso de caracterización: En esta etapa se analizaron las propiedades estadísticas de los datos y se identificaron posibles relaciones entre las variables.

3. Fase de descomposición: En esta etapa se aplicó el VMD a las variables de entrada, lo que permitió identificar patrones o modos en los datos y mejorar la precisión de la predicción.

4. Escalamiento de características: Finalmente, en esta etapa se escalaron las variables de entrada para mejorar la capacidad de la red neuronal para generalizar y mejorar la precisión de la predicción.

Estos cuatro pasos son importantes para asegurarse de que los datos estén en un formato adecuado para su uso en la red neuronal y para mejorar la capacidad de la red neuronal para aprender patrones en los datos y predecir con precisión.

### 5.1.2 *Pre procesamiento de datos*

Este paso se centró en la limpieza de los datos. En primer lugar, los valores que faltan en las series temporales de potencia se han sustituido por el valor del día anterior. Para ello se definió la función ‘fill_missing’ cuyo código es:

```
# llenar los valores que faltan con un valor a la misma hora hace un día
def fill_missing(dataset):
  one_day = 60 * 24
  for row in range(1,dataset.shape[0]):
    #La primera columna es la fecha, empieza de 1
    for col in range(dataset.shape[1]):
      if pd.isnull(dataset.iloc[row, col]):
        dataset.iloc[row, col] = dataset.iloc[row - one_day, col]
```

En segundo lugar, los valores negativos se han sustituido por 0. Por último, se fijó un umbral equivalente a la máxima potencia que genera el parque eólico y se seleccionó aquéllos patrones y targets cuya producción sean datos anómalos.

El umbral seleccionado fue de 3,300 KW. El código usado fue el siguiente:

```
#Cargamos el archivo .csv
df= pd.read_csv(file_id[1],header=0, sep=',',usecols = [1,2],low_memory=False,parse_dates={'datetime':[0]})
df.columns = ['Date','Despacho_ejecutado']
df.loc[df['Despacho_ejecutado'] > 33, 'Despacho_ejecutado'] = '?'
df.replace('?', nan, inplace=True)

#Variables para el entrenamiento (univariante)
cols = list(df)[1]
#Convertimos a data numerico
df[cols] = df[cols].astype('float32')

# llama la función fill_missing
fill_missing(df)
# Dataframe con feautures de entrenamiento y test (no incluye Date)
df_feautures=df[cols]
print(df.head())
print(df_feautures.head())
```

### 5.1.3 *Selección de características*

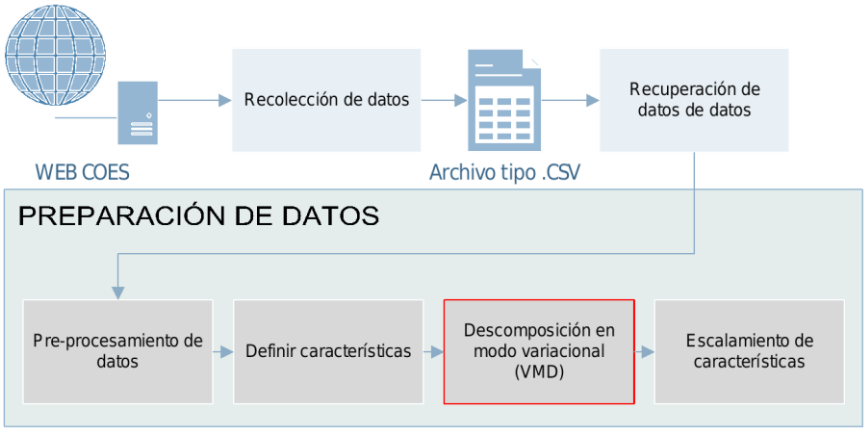
Uno de los pasos de mayor complejidad para la construcción de un buen modelo de red neuronal es la selección y clasificación de los patrones óptimos, sin embargo, para el siguiente estudio se utilizó un modelo univariante, es decir considerando solo valores de potencia históricos con un periodo de muestreo 30 minutos.

## 5.2 Descomposición en modo variacional (VMD)

Los datos históricos, una vez preparados, se descomponen en un conjunto discreto de Funciones Modales Intrínsecas (IMF). Aunque un mayor número de IMFs, denotado como “K”, puede mejorar la precisión, un exceso en la cantidad de capas de descomposición puede generar problemas, como una mayor carga computacional y la introducción de errores. Con base en esta consideración, se optó por seleccionar y comparar dos y tres IMFs, respectivamente, para posteriormente evaluar su impacto en el estudio

**Figura 26**

*Sub-proceso de recoleccion y preparacion de datos.*

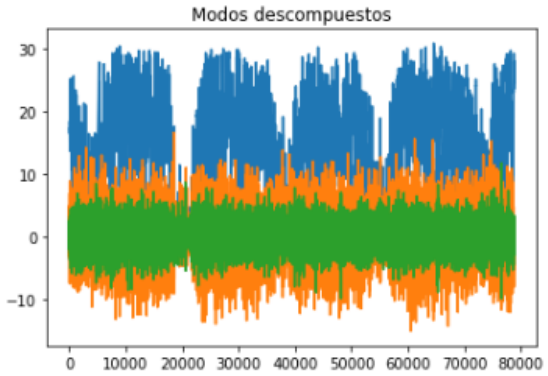



*Nota*. Elaboración propia.

El código que se presenta a continuación se utilizó para descomponer la señal original en sus respectivos modos.

```
# La data original se descompone en "K" Modos (Puede considerarse como parametros)
# Instalamos libreria para VMD
!pip install vmdpy

# some sample parameters for VMD
from vmdpy import VMD
import random

f=set_power.values
alpha = 200       # moderate bandwidth constraint
tau = 0.            # noise-tolerance (no strict fidelity enforcement)
K = 3               # 3 modes
DC = 0              # no DC part imposed
init = 1            # initialize omegas uniformly
tol = 1e-7

# Run actual VMD code
u, u_hat, omega = VMD(f, alpha, tau, K, DC, init, tol)

# Simple Visualization of decomposed modes
plt.figure()
plt.plot(u.T)
plt.title('Modos descompuestos')
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: vmdpy in /usr/local/lib/python3.8/dist-packages (0.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (from vmdpy) (1.22.4)
Text(0.5, 1.0, 'Modos descompuestos')
```




### 5.2.1 *Normalizado*

Se normalizaron los valores utilizando la función 'MinMaxScaler', escalándolos entre -1 y 1. La normalización permitió agilizar el entrenamiento y reducir la probabilidad de estancarse en óptimos locales. Además, se logró robustez frente a desviaciones estándar pequeñas de las variables mediante entradas estandarizadas. A continuación, se presenta el código utilizado para el escalamiento:

```
scaler = MinMaxScaler(feature_range=(-1, 1))
train_scaled = scaler.fit_transform(data_train.values.reshape(-1, 1))
test_scaled = scaler.transform(data_test.values.reshape(-1, 1))
```

### 5.2.2 *Segmentación de datos para entrenamiento y prueba*

Se dividió el conjunto de datos en dos secciones, una de entrenamiento (train) y otra de prueba (test). La proporción recomendada fue de 80 y 20, es decir del total de valores del conjunto de datos, el 80% fue utilizado para el entrenamiento y el otro 20% para prueba.

```
# Guardamos los conjuntos de entrenamiento y prueba en archivos CSV
TRAIN_SIZE = 0.8
idx = round(len(data_VMD)*TRAIN_SIZE)
```

```
data_train = data_VMD[:idx]
data_test=data_VMD[idx:]

# Escribe los datos de entrenamiento en un archivo CSV
train_file = '/content/drive/MyDrive/Colab Notebooks/Webscrap Data COES/Data '+ str(m) +' Modo/train_file_' + str(m) +'modos.csv'
data_train.to_csv(train_file, index=True)

# Escribe los datos de prueba en un archivo CSV
test_file = '/content/drive/MyDrive/Colab Notebooks/Webscrap Data COES/Data '+ str(m) +' Modo/test_file_' + str(m) + 'modos.csv'
data_test.to_csv(test_file, index=True)
```

```
# Primero dividamos los sets entre train y test
TRAIN_SIZE = 0.8
idx = round(len(data_VMD)*TRAIN_SIZE)
data_train = data_VMD[:idx]
data_test=data_VMD[idx:]

# escribe los datos de entrenamiento en un archivo CSV en la unidad de Google Drive
train_file = '/content/drive/MyDrive/Colab Notebooks/Modelos LSTM/train_file.csv'
data_train.to_csv(train_file, index=False)

# escribe los datos de prueba en un archivo CSV en la unidad de Google Drive
test_file = '/content/drive/MyDrive/Colab Notebooks/Modelos LSTM/test_file.csv'
data_test.to_csv(test_file, index=False)
```

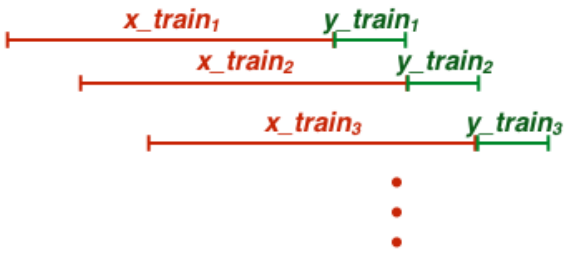
Se empleó una función específica, cuyo código se presenta a continuación, para estructurar los conjuntos de datos de entrenamiento y prueba en ventanas de tiempo. Este proceso, también conocido como "ventaneo", implica determinar cuántos datos históricos o valores previos son necesarios para predecir el siguiente valor.

```
def get_sequence_data(data_item, long_sequence, n_preds):
    X, Y = [], []
    for i in range(0, len(data_item) - long_sequence - n_preds + 1, n_preds):
        X.append(data_item[i:i + long_sequence].reshape((long_sequence, -1)))
        Y.append(data_item[i + long_sequence:i + long_sequence + n_preds].reshape((-1, n_preds, 1)))
    X = np.array(X)
    Y = np.array(Y)
    return X, Y
```

Como entrada al modelo se utilizó bloques de X datos consecutivos (X_train) y el modelo deberá aprender a predecir el dato Y de la secuencia (Y_train).

**Figura 27**

*Ventaneo de set de datos de entrenamiento.*




## 5.3 Selección de hiperparametros

El proceso de entrenamiento y optimización de modelos LSTM requiere ajustar pesos y sesgos a través de iteraciones sucesivas. Es esencial seleccionar hiperparámetros adecuados, validar el rendimiento del modelo y detener el entrenamiento una vez que se logra la convergencia, todo con el objetivo de obtener predicciones precisas en la generación de energía eólica.

Para el modelo de pronóstico LSTM-VMD enfocado en la generación de energía, los hiperparámetros que se considerarán para ajustes son los siguientes:

**Tabla 6**

Parámetros a modificar en el entrenamiento

| Hiperparámetros | Símbolo |
|---|---|
| Cantidad de neuronas de capa entrada | N_UNITS |
| Cantidad de neuronas de capa oculta | N_UNITS_HIDDEN |
| Longitud de secuencia | LONG_SEC |
| Número de épocas | epoch_number |
| Tipo de optimizador | Optimizer, ADAM |

Fuente: Elaboración propia.

En este caso, se modificaron 5 parámetros de entrenamiento. El código utilizado es el siguiente:

```
# Diccionario de parámetros
params = [
    {
        'LONG_SEC': 3*48,  # Longitud de la secuencia de entrada
        'N_UNITS': 64,  # Número de unidades en la capa LSTM
        'N_UNITS_HIDDEN': 32,  # Número de unidades en la capa oculta LSTM
        'epoch_number': 15,  # Número de épocas para el entrenamiento
        'batch_size': 256,  # Tamaño del lote para el entrenamiento
        'NN_model': {  # Parámetros para el modelo de red neuronal
            'optimizer': {'type': 'Adam', 'lr': 0.001},  # Optimizador y tasa de aprendizaje
            'metrics': {'metrics': ['mean_absolute_error']},  # Métricas para evaluar el modelo
            'loss': {'loss': ['mse']}  # Función de pérdida
        },
        'n_preds': 24,  # Número de predicciones a realizar
        'seed': 88  # Semilla para la generación de números aleatorios
    },
    {
        'LONG_SEC': 3*48,
        'N_UNITS': 4,
        'N_UNITS_HIDDEN': 2,
        'epoch_number': 10,
        'batch_size': 128,
        'NN_model': {
            'optimizer': {'type': 'Adam', 'lr': 0.001},
            'metrics': {'metrics': ['mean_absolute_error']},
            'loss': {'loss': ['mse']}
        },
        'n_preds': 24,  # Número de predicciones
        'seed': 88
    },
    {
        'LONG_SEC': 3*48,
        'N_UNITS': 128,
        'N_UNITS_HIDDEN': 64,
        'epoch_number': 5,
        'batch_size': 256,
        'NN_model': {
            'optimizer': {'type': 'Adam', 'lr': 0.01},
            'metrics': {'metrics': ['mean_absolute_error']},
            'loss': {'loss': ['mse']}
        },
        'n_preds': 24,  # Número de predicciones
        'seed': 88
    }
]

# Selecciona los primeros 'num_columnas' elementos de params
params = params[:num_columnas]
```

Estos parámetros son flexibles a cambios y modificaciones continuas, debido a que la red se está probando constantemente, por lo que solo son referenciales.

## 5.4 Construcción de la red LSTM

En esta sección, se estableció la arquitectura de la red LSTM, adaptando las entradas y salidas de entrenamiento a estructuras específicas y definiendo sus atributos y parámetros. Se introduce la clase denominada ‘LSTM_net’.

### 5.4.1 *Diseño de la arquitectura*

Se especificó la estructura y configuración de la red neuronal destinada al pronóstico de generación de energía eólica. Esto abarca la determinación del número de capas LSTM, la definición de unidades por capa, la organización de conexiones intercapas y la selección de funciones de activación."

```python
def build_model(params):
    # Obtención de los parámetros para la construcción del modelo de la red LSTM
    N_UNITS = params['N_UNITS']
    N_UNITS_HIDDEN = params['N_UNITS_HIDDEN']
    LONG_SEC = params['LONG_SEC']
    optimizador = params['NN_model']['optimizer']['type']
    loss = params['NN_model']['loss']['loss']
    metrics = params['NN_model']['metrics']['metrics']

    # Semilla de los generadores aleatorios
    tf.random.set_seed(params['seed'])
    np.random.seed(params['seed'])

    # Definición del modelo de la red LSTM
    modelo = Sequential()
    modelo.add(LSTM(N_UNITS, input_shape=(LONG_SEC, 1), return_sequences=True, activation='tanh')) # Capa LSTM con unidades N_UNITS
    modelo.add(Dropout(0.2))

    modelo.add(LSTM(N_UNITS_HIDDEN, return_sequences=True, activation='tanh')) # Capa LSTM oculta con unidades N_UNITS_HIDDEN y activación relu
    modelo.add(Dropout(0.2)) # Capa de regularización de Dropout

    modelo.add(LSTM(N_UNITS_HIDDEN, activation='tanh')) # Capa LSTM oculta con unidades N_UNITS_HIDDEN y activación relu
    modelo.add(Dropout(0.2)) # Capa de regularización de Dropout

    # Se agrega la capa TimeDistributed para que la capa Dense actúe sobre cada valor de predicción por separado
    modelo.add(Dense(units=params['n_preds'], activation='tanh'))

    modelo.add(Dense(units=params['n_preds'], activation='linear'))

    # Compilación del modelo
    modelo.compile(optimizer=optimizador, loss=loss, metrics=metrics)
    print(modelo.summary())
    return modelo
```

### 5.4.2 *Diseño de algoritmo de entrenamiento de las redes LSTM*

Se llevó a cabo mediante un proceso iterativo de ajuste de los pesos y sesgos en la red para minimizar el error de predicción.

```python
def train_model(params, data_train, data_test, name):
    EPOCHS = params['epoch_number']
    BATCH_SIZE = params['batch_size']
    LONG_SEC = params['LONG_SEC']
    n_preds = params['n_preds']  # Número de predicciones

    scaler = MinMaxScaler(feature_range=(-1, 1))
    train_scaled = scaler.fit_transform(data_train.values.reshape(-1, 1))
    test_scaled = scaler.transform(data_test.values.reshape(-1, 1))

    x_train, y_train = get_sequence_data(train_scaled, LONG_SEC, n_preds)
    x_test, y_test = get_sequence_data(test_scaled, LONG_SEC, n_preds)

    y_train = np.reshape(y_train, (y_train.shape[0], y_train.shape[2]))
    y_test = np.reshape(y_test, (y_test.shape[0], y_test.shape[2]))

    print("Dimensiones de x_train y y_train:", x_train.shape, y_train.shape)
    print("Dimensiones de x_test y y_test:", x_test.shape, y_test.shape)

    model = build_model(params)

    history = model.fit(x_train, y_train, batch_size=BATCH_SIZE, epochs=EPOCHS, validation_data=(x_test, y_test))

    model.save(f'/content/drive/MyDrive/Colab Notebooks/Webscrap Data COES/Multi step model/{name}.h5')
    with open(f'/content/drive/MyDrive/Colab Notebooks/Webscrap Data COES/Multi step model/scaler_{name}.pkl', 'wb') as f:
        pickle.dump(scaler, f)

    show_history(history.history, EPOCHS)
    print(f'[Model] Training Completed. Model saved as {name}')
    return model, scaler, x_test, y_test
```

```python
# Entrenamiento de modelos
models = {}     # Diccionario para almacenar los modelos entrenados
scalers = {}    # Diccionario para almacenar los escaladores utilizados

for i, param in enumerate(params, 1):
    if f'Mode {i}' in data_train.columns and f'Mode {i}' in data_test.columns:
        data_train_mode = data_train[f'Mode {i}']        # Datos de entrenamiento para el modo i
        data_test_mode = data_test[f'Mode {i}']          # Datos de prueba para el modo i
        model_name = f'modelo_lstm_{i}_{i}'              # Nombre del modelo

        model, scaler, x_test, y_test = train_model(param, data_train_mode, data_test_mode, model_name)  # Entrenamiento del modelo
        models[model_name] = model                    # Almacenar el modelo en el diccionario
        scalers[model_name] = scaler                  # Almacenar el escalador en el diccionario
```

### 5.4.3 *Definición de salidas (targets)*

De acuerdo al objetivo del proyecto las salidas fueron:

- Producción en un horizonte de 01 paso: Es un vector 1 x 1; Es decir 30 min en adelante.
- Producción en horizonte de 24 pasos: Es un vector de 1 x N, en donde N es el número de muestras que existen en el próximo día (24 pasos o 12 horas).
- Producción en horizonte de 48 pasos: Es un vector de 1 x N, en donde N es el número de muestras que existen en el próximo día (48 pasos o 12 horas).

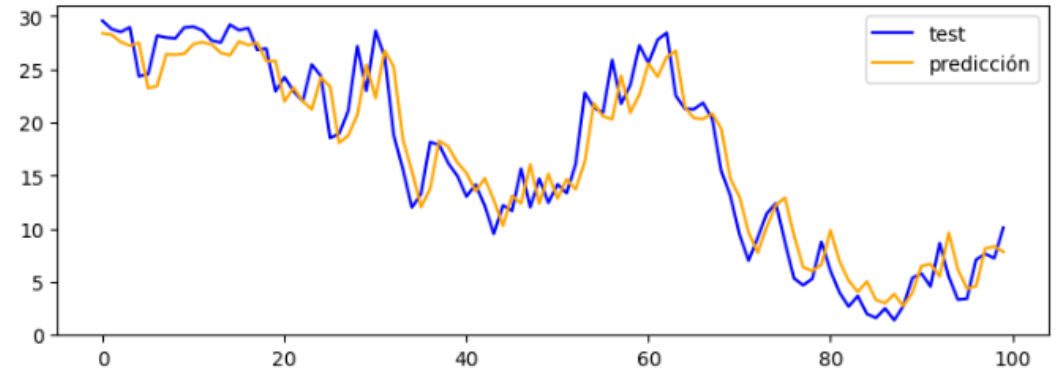
## 5.5 Resultado del modelo LSTM

Para validar y evaluar la precisión del modelo LSTM, se utilizó un conjunto de datos independientes que no se utilizaron anteriormente en el proceso de entrenamiento.

### 5.5.1 *Pronóstico single-step (1 step)*

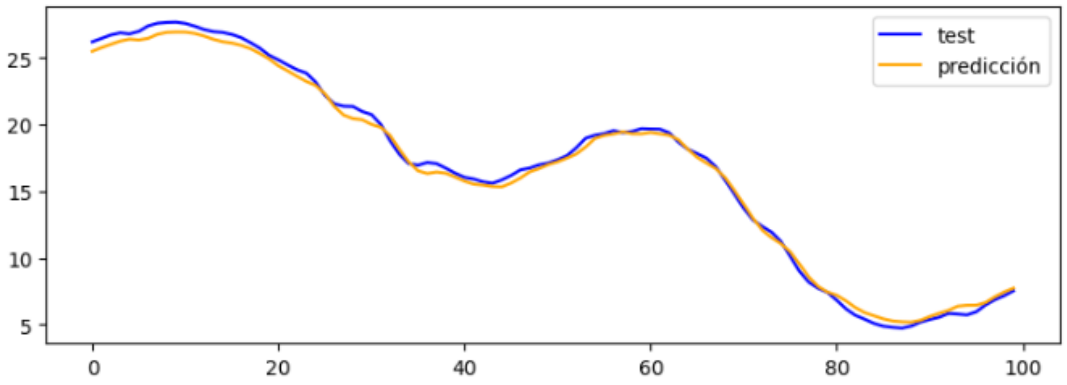
Para la evaluación de pronóstico Simple-step con el Modo 1, que corresponde a la descomposición variacional de frecuencia de la señal original, se utilizó una métrica de evaluación comúnmente utilizada en machine learning, el error cuadrático medio (MSE) y error absoluto medio (MAE).

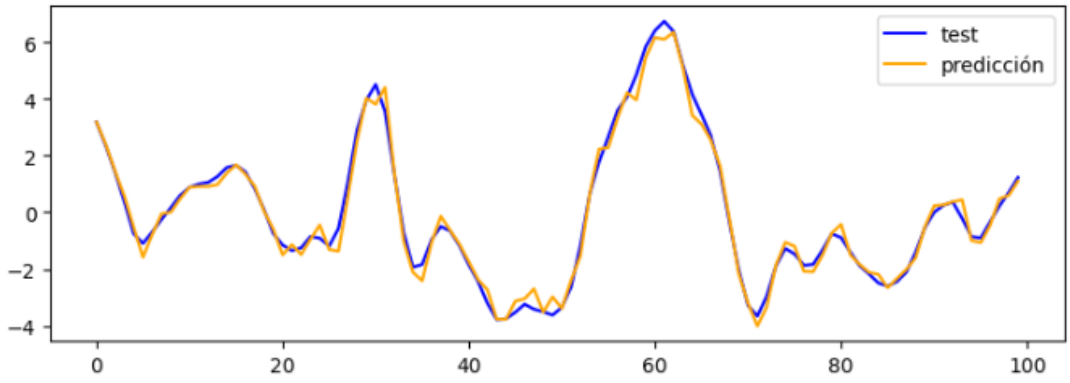
```
489/489 [==============================] - 3s 5ms/step
Model 1 MSE: 9.361338669986637
Model 1 MAE: 2.304618720374096
Combined models MSE: 9.361338669986637
Combined models MAE: 2.304618720374096
```



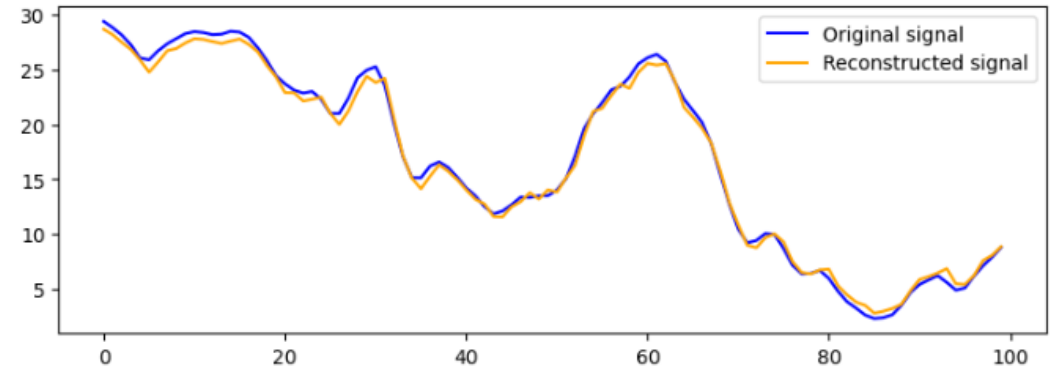

Para la evaluación de pronóstico Simple-step con el Modo 1y Modo 2, que corresponde a la descomposición variacional de frecuencia de la señal original, se utilizó una métrica de evaluación comúnmente utilizada en machine learning, el error cuadrático medio (MSE). Este indicador cuantifica la diferencia entre los valores estimados por el modelo y los valores reales.

```
489/489 [==============================] - 2s 5ms/step
489/489 [==============================] - 3s 5ms/step
Model 1 MSE: 0.25055290185454215
Model 1 MAE: 0.4173667676957631
Model 2 MSE: 0.1377656240851911
Model 2 MAE: 0.28141829422278336
Combined models MSE: 0.4349656865287261
Combined models MAE: 0.5444297408925843
```



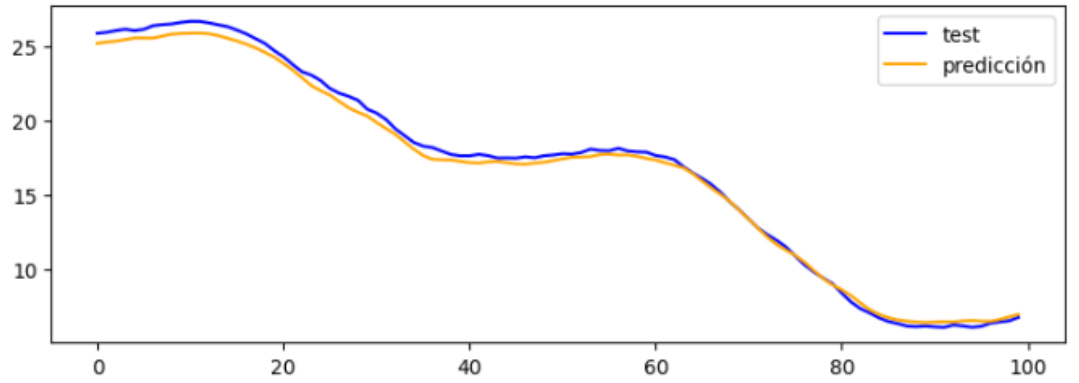

```
Combined model MAE: 0.5444297408925843
Combined model MSE: 0.4349656865287261
```

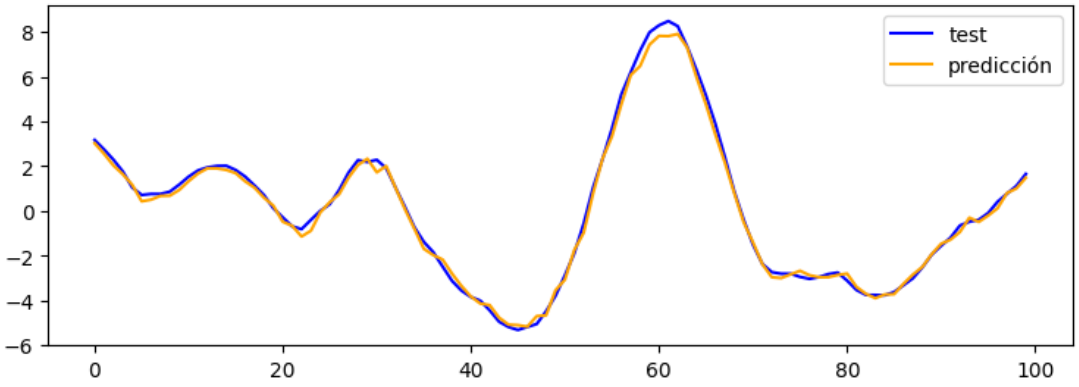


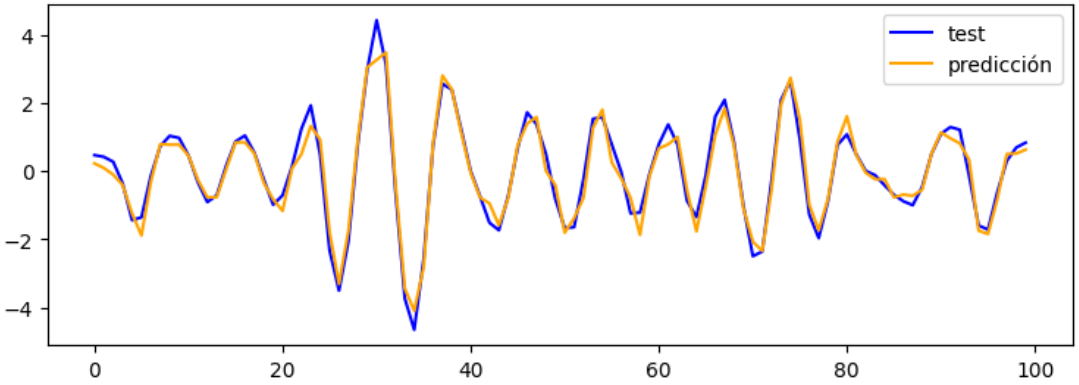

Para la evaluación de pronóstico Simple-step con el Modo 1, Modo 2 y Modo 3, que corresponde a la descomposición variacional de frecuencia de la señal original, se utilizó una métrica de evaluación comúnmente utilizada en machine learning, el error cuadrático medio

(MSE). Este indicador cuantifica la diferencia entre los valores estimados por el modelo y los valores reales.

```
489/489 [==============================] - 3s 5ms/step
489/489 [==============================] - 3s 5ms/step
489/489 [==============================] - 4s 6ms/step
Model 1 MSE: 0.2279761127195631
Model 1 MAE: 0.3998012312640419
Model 2 MSE: 0.06516621064366519
Model 2 MAE: 0.1967323545721449
Model 3 MSE: 0.1267267661135771
Model 3 MAE: 0.26821294845511634
Combined models MSE: 0.6044286665705488
Combined models MAE: 0.6228398863527684
```

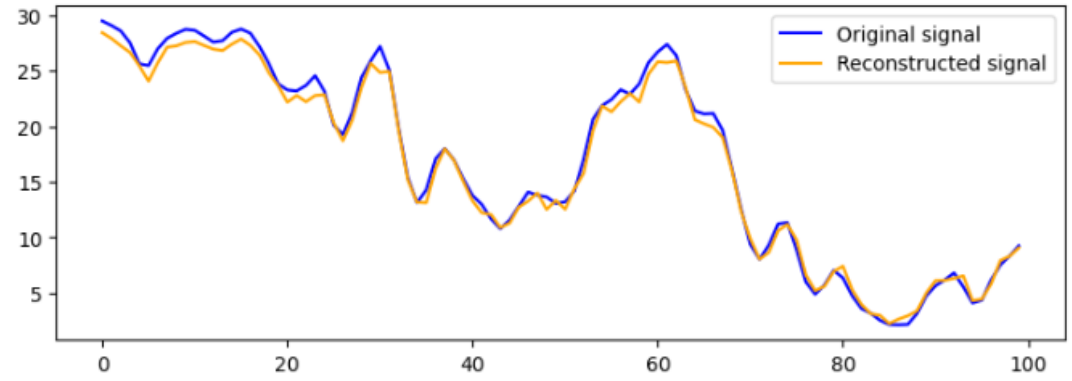


Modelo general reconstruido combinado:

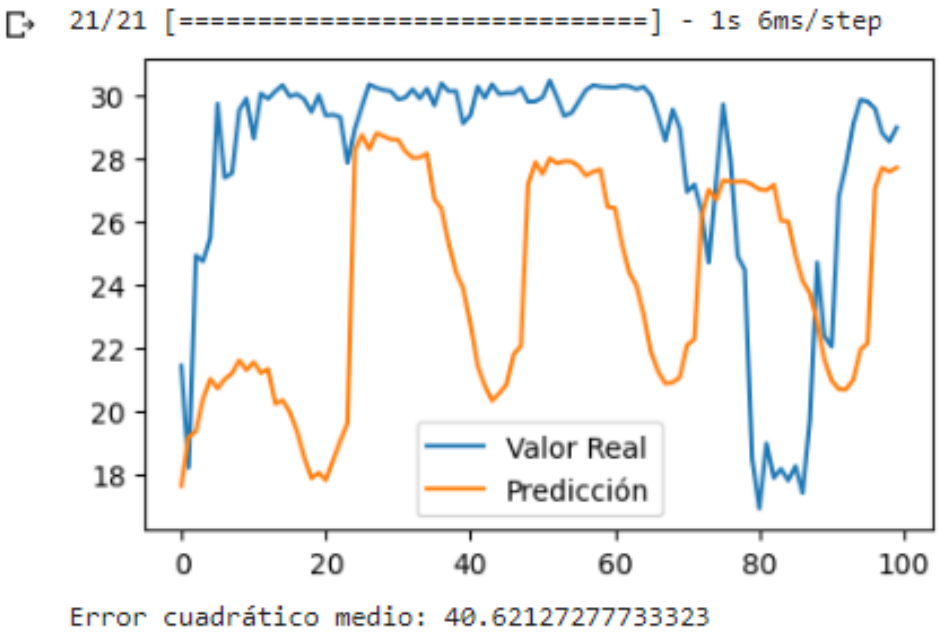
```
Combined model MAE: 0.6228398863527684
Combined model MSE: 0.6044286665705488
```



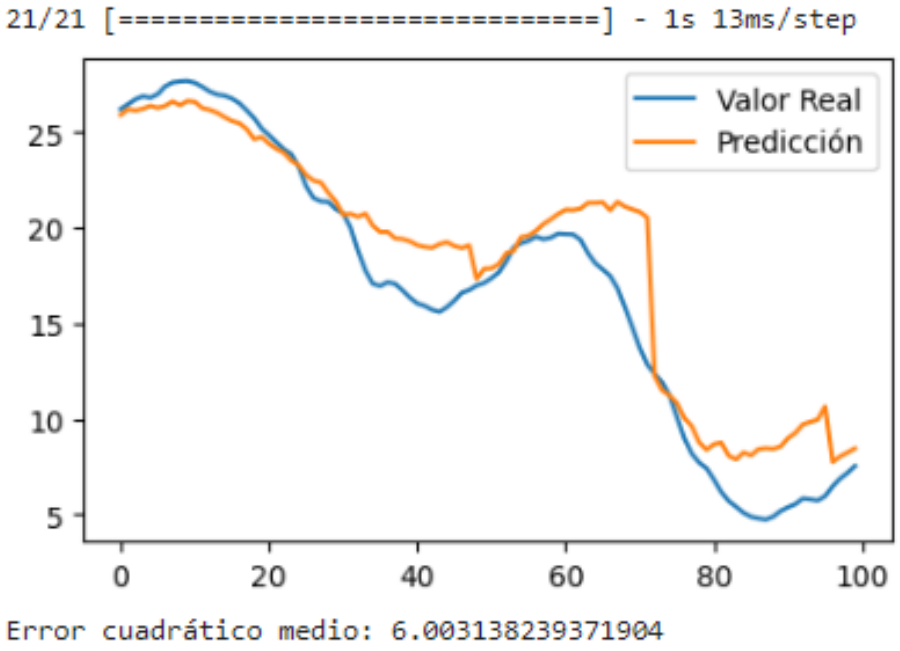

### 5.5.2 *Pronóstico Multi-step (24 steps)*

Este estudio implementa un enfoque de pronóstico multi-step, o por bloques, que busca predecir una secuencia completa de valores futuros en una sola ejecución, en lugar de un único valor.

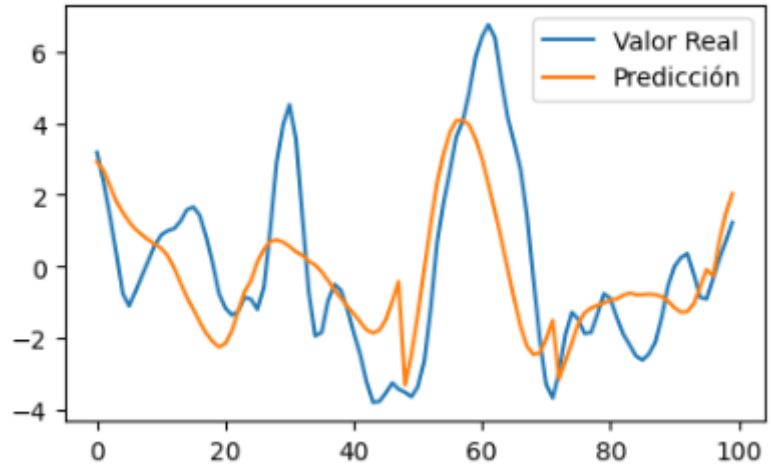
En el análisis unimodal, se utilizó un solo modo de Descomposición Modo Variacional (VMD) para entrenar una red LSTM. El objetivo era predecir una secuencia de valores futuros, aplicando una estrategia de pronóstico directo o por bloques. El modelo estaba diseñado para predecir una secuencia de 24 pasos, equivalente a 12 horas, en una sola ejecución, y mostró un desempeño notablemente efectivo.




Progresando al análisis bimodal, se incorporó un segundo modo de Descomposición Modo Variacional (VMD). Este enfoque mejoró la predicción de la misma secuencia de 24 pasos, teniendo en cuenta las características derivadas de dos modos variacionales.



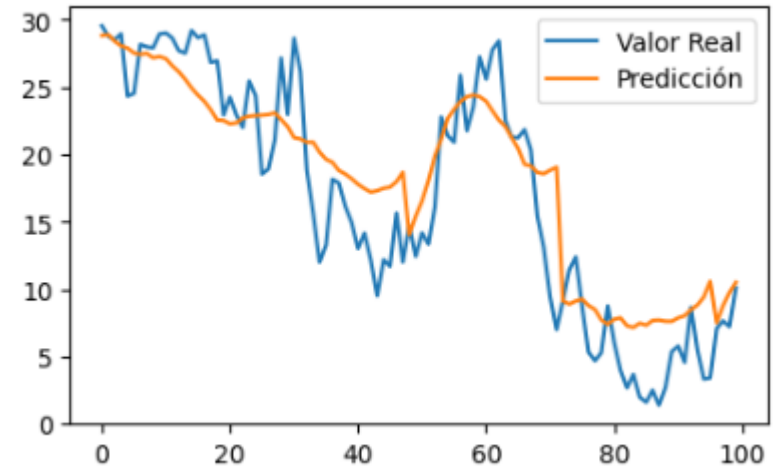
```
21/21 [==============================] - 1s 6ms/step
```




```
Error cuadrático medio: 3.5445986147827 65
```

Modelo reconstruido combinado:

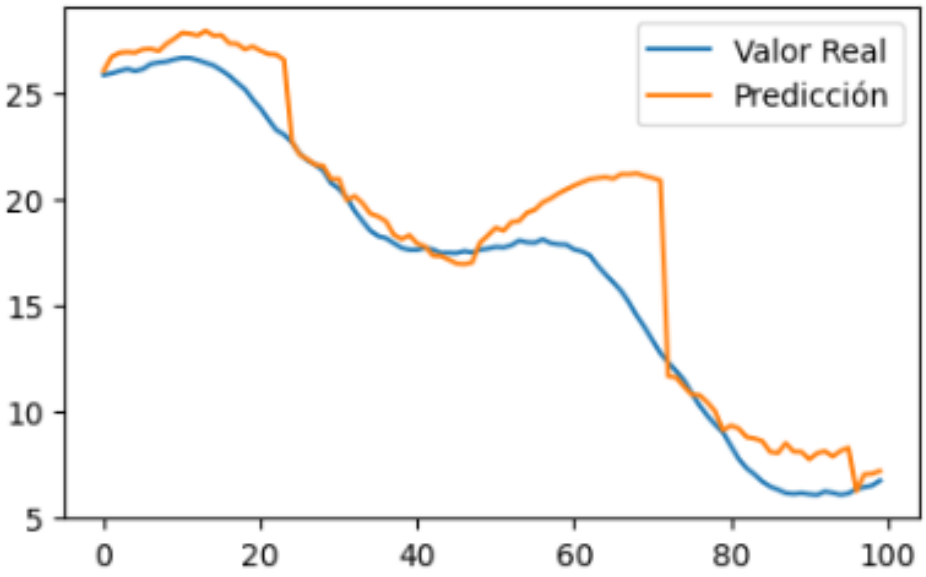
```
[15624 rows x 4 columns]
```




```
Error cuadrático medio: 14.891440380682857
```

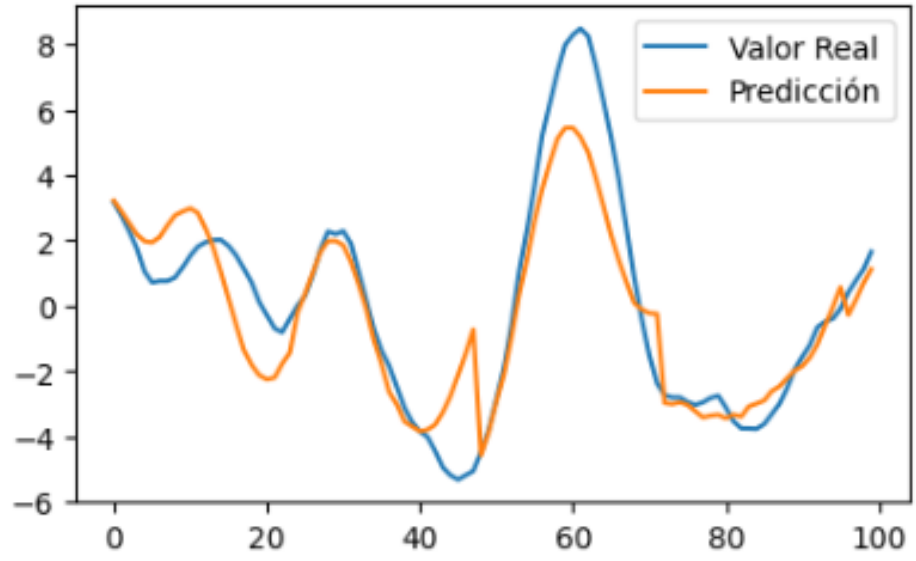
Por último, se exploró un análisis trimodal, donde se implementaron tres modos de Descomposición Modo Variacional (VMD). Esta estrategia multimodal mejoró aún más la capacidad predictiva del modelo, aprovechando la diversidad de patrones capturados por los tres modos variacionales.

```
21/21 [==============================] - 2s 11ms/step
```



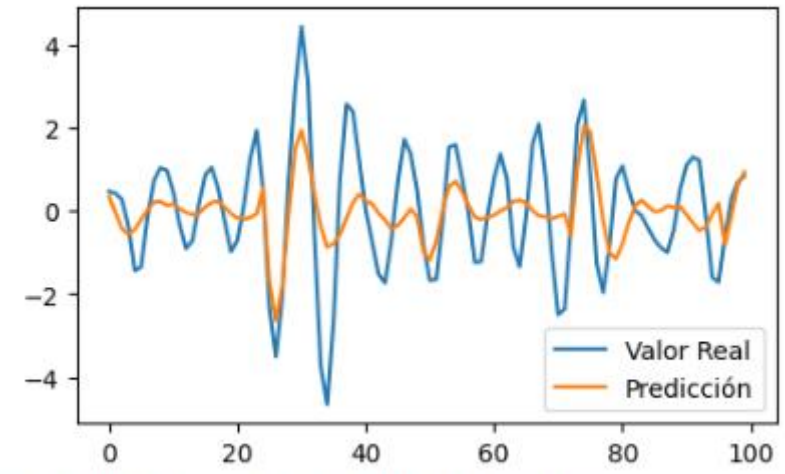

```
Error cuadrático medio: 5.436821706051129
```

```
21/21 [==============================] - 1s 8ms/step
```




```
Error cuadrático medio: 2.0959245558582613
```

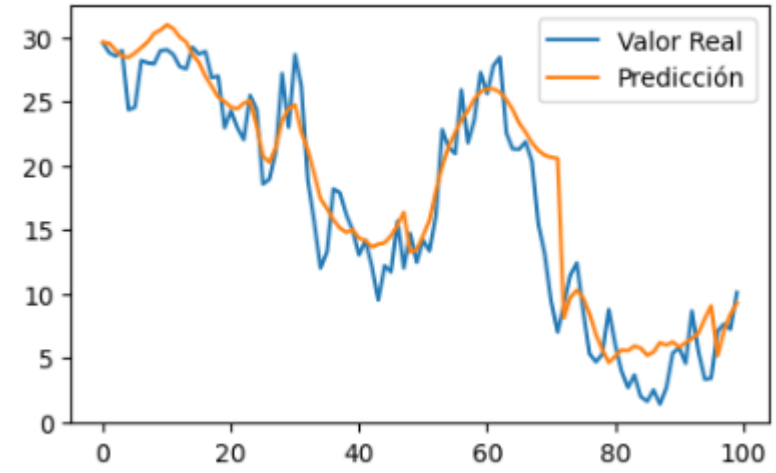
```
21/21 [==============================] - 1s 8ms/step
```




```
Error cuadrático medio: 1.5145112104141132
```

Modelo reconstruido combinado:

```
[15624 rows x 5 columns]
```



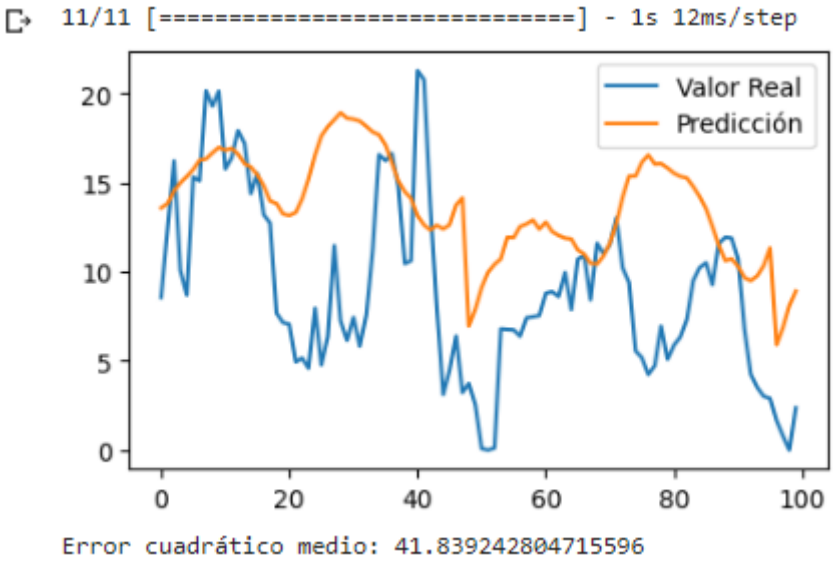

```
Error cuadrático medio: 9.411700377524067
```

### 5.5.3 *Pronóstico Multi-step (48 steps)*

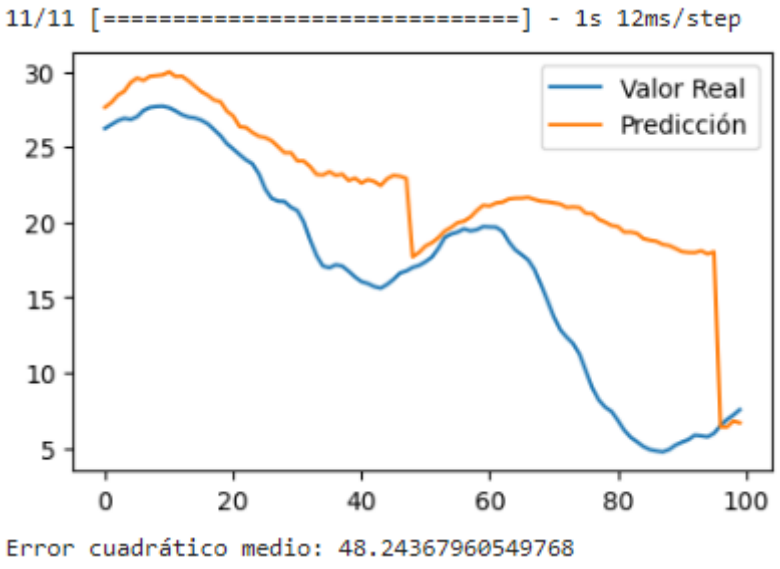
Este estudio implementa un enfoque de pronóstico multi-step, o por bloques, que busca predecir una secuencia completa de valores futuros en una sola ejecución, en lugar de un único valor.

En el análisis unimodal, se utilizó un solo modo de Descomposición Modo Variacional (VMD) para entrenar una red LSTM. El objetivo era predecir una secuencia de valores futuros,

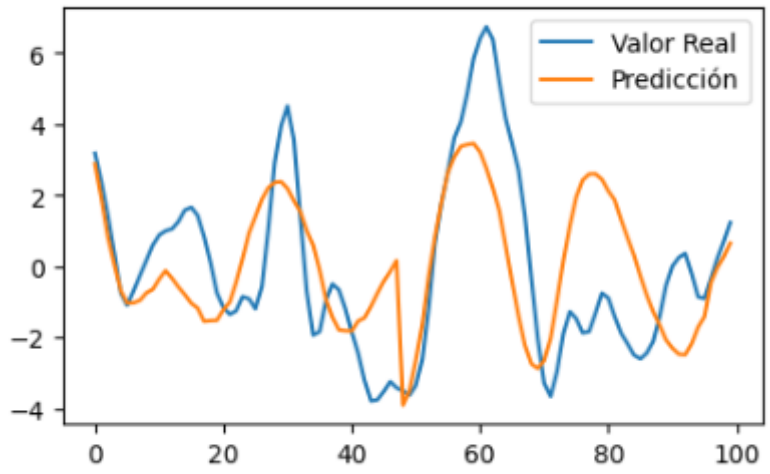
aplicando una estrategia de pronóstico directo o por bloques. El modelo estaba diseñado para predecir una secuencia de 48 pasos, equivalente a 24 horas, en una sola ejecución.




Progresando al análisis bimodal, se incorporó un segundo modo de Descomposición Modo Variacional (VMD). Este enfoque mejoró la predicción de la misma secuencia de 48 pasos, teniendo en cuenta las características derivadas de dos modos variacionales.



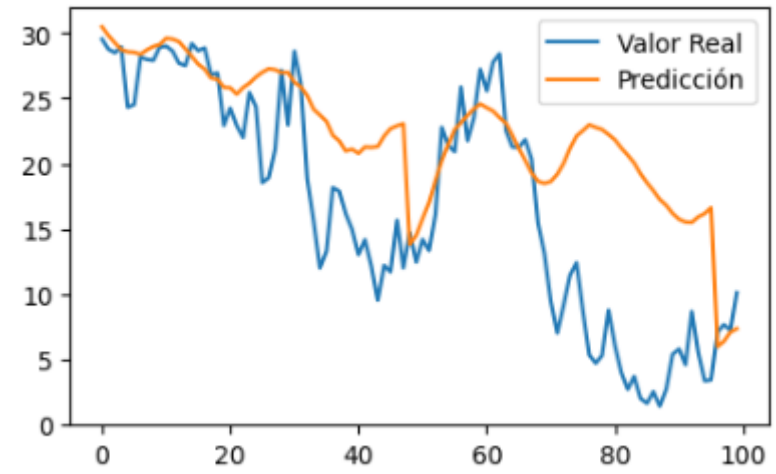
```
11/11 [==============================] - 1s 9ms/step
```




```
Error cuadrático medio: 4.474625553713441
```

Modelo reconstruido:

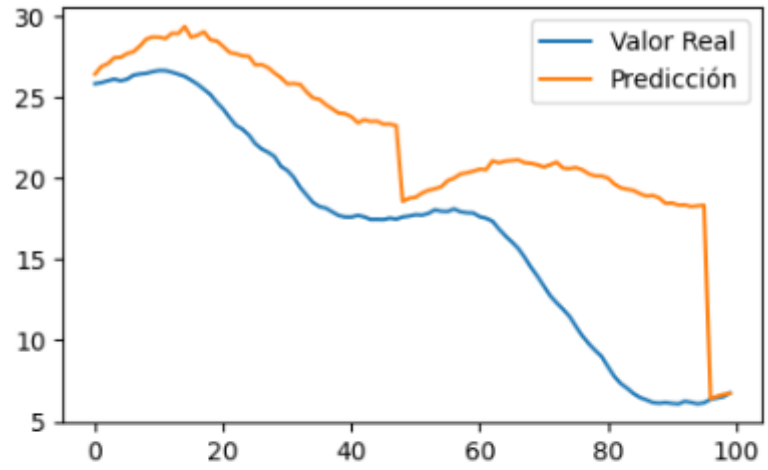
```
[15600 rows x 4 columns]
```



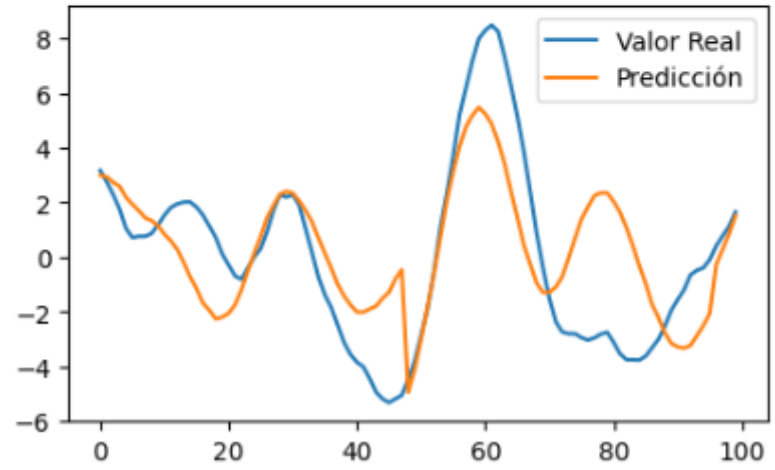

```
Error cuadrático medio: 67.10273377157831
```

Por último, se exploró un análisis trimodal, donde se implementaron tres modos de Descomposición Modo Variacional (VMD). Esta estrategia multimodal mejoró aún más la capacidad predictiva del modelo, aprovechando la diversidad de patrones capturados por los tres modos variacionales.

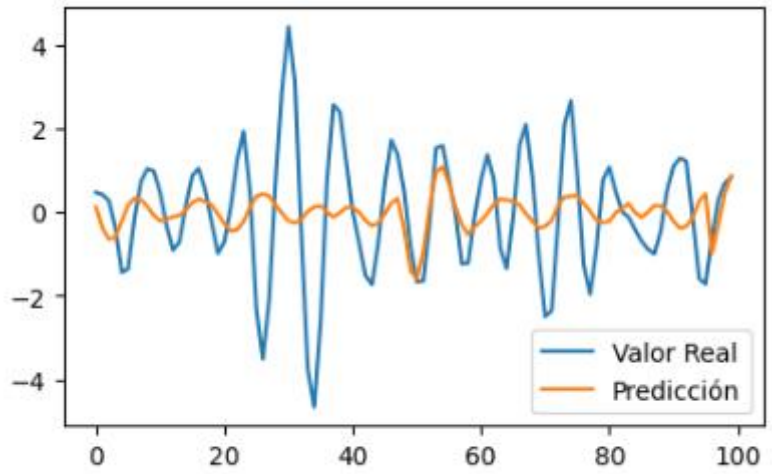
```
11/11 [==============================] - 1s 12ms/step
```




```
Error cuadrático medio: 46.08287344965249
11/11 [==============================] - 1s 10ms/step
```

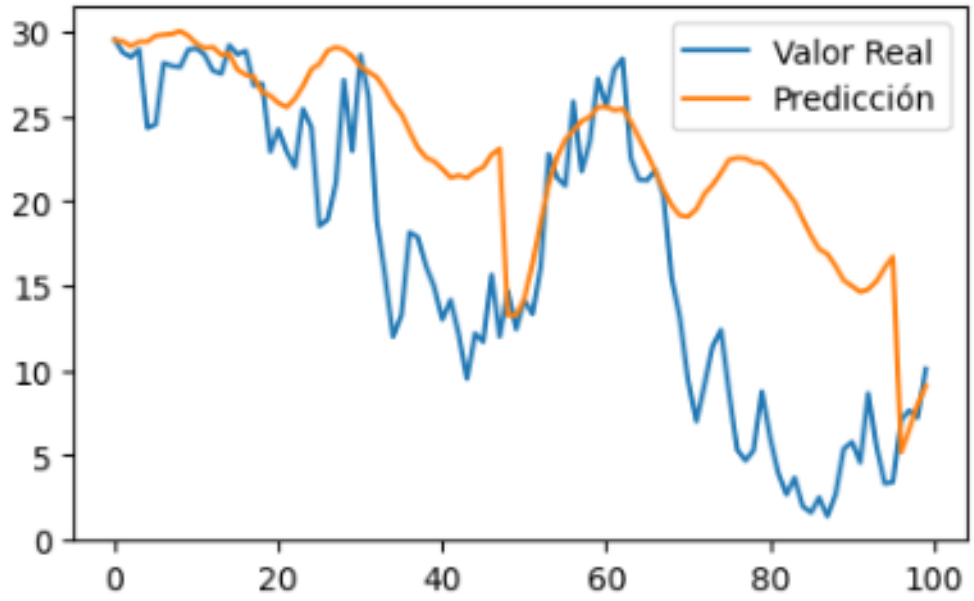



```
Error cuadrático medio: 5.950231198207376
```




```
Error cuadrático medio: 2.239059205881306
```

Modelo reconstruido:




```
Error cuadrático medio: 68.94679814076046
```

### 5.5.4 *Validación del modelo*

Con el fin de verificar la validez y para evaluar el rendimiento de cada uno de los métodos de predicción propuestos, se seleccionaron el índice de evaluación: el error medio absoluto (MAE) y el error cuadrático medio (MSE), fueron seleccionados para evaluar el rendimiento de la predicción y se escogió, el de mejor desempeño.

## 5.6 Herramientas y librerías utilizadas

El desarrollo y la implementación eficaz de modelos de aprendizaje automático, como las redes LSTM-VMD que forman parte de la investigación, requieren una serie de herramientas y librerías especializadas. Estas herramientas no solo facilitan la construcción y el entrenamiento de los modelos, sino que también proporcionan funciones para la limpieza y la preparación de los datos, la evaluación de los modelos y la visualización de los resultados.

1. Google Colab: Una plataforma de Google para escribir y ejecutar código de Python en la nube. Se utiliza para montar Google Drive y autenticar el usuario.
2. PyDrive: Una biblioteca de Python para interactuar con la API de Google Drive. Se utiliza para autenticar y crear el cliente de Google Drive.
3. oauth2client: Una biblioteca para implementar el protocolo OAuth 2.0 en Python. Se utiliza para obtener las credenciales predeterminadas de la aplicación.
4. os: Un módulo de Python que proporciona funciones para interactuar con el sistema operativo.
5. pandas: Una biblioteca de manipulación y análisis de datos de Python.
6. seaborn: Una biblioteca de visualización de datos de Python basada en matplotlib. Proporciona una interfaz de alto nivel para dibujar gráficos estadísticos atractivos.
7. numpy: Una biblioteca de Python para trabajar con matrices de datos de gran tamaño y matrices matemáticas de alto nivel.
8. matplotlib: Una biblioteca de trazado en 2D de Python que produce figuras de calidad de publicación en una variedad de formatos de papel y entornos interactivos.
9. sklearn (Scikit-learn): Una biblioteca de aprendizaje automático de Python que proporciona herramientas simples y eficientes para la minería y análisis de datos. Aquí se utiliza para el preprocesamiento de datos y métricas de error.
10. Keras: Una biblioteca de redes neuronales de Python de alto nivel, escrita en Python y capaz de ejecutarse sobre TensorFlow. Se utiliza para construir y entrenar modelos de redes neuronales.

11. TensorFlow: Una biblioteca de código abierto para el aprendizaje automático y otras tareas que requieren operaciones matemáticas intensivas. Se utiliza para construir y entrenar modelos de redes neuronales.

12. pickle: Un módulo de Python que implementa protocolos binarios para serializar y deserializar la estructura de un objeto Python.

# Capítulo 6. Análisis y discusión de resultados

En este capítulo se procedió a comparar los tres modelos del capítulo anterior y se seleccionó el modelo óptimo para la predicción para cada steps evaluado.

## 6.1 Evaluación del modelo optimo Single Step (1 step)

Se llevó a cabo una evaluación comparativa de los modelos basados en un único paso adelante dentro de los diversos modos de variación proporcionados VMD.

- ✓ Modelo uni-modal, de la Tabla 7, se selecciona los modelos de mejor desempeño y este se evalua en los datos de prueba y la Figura 28 resume el error promediado mensual.

**Tabla 7**

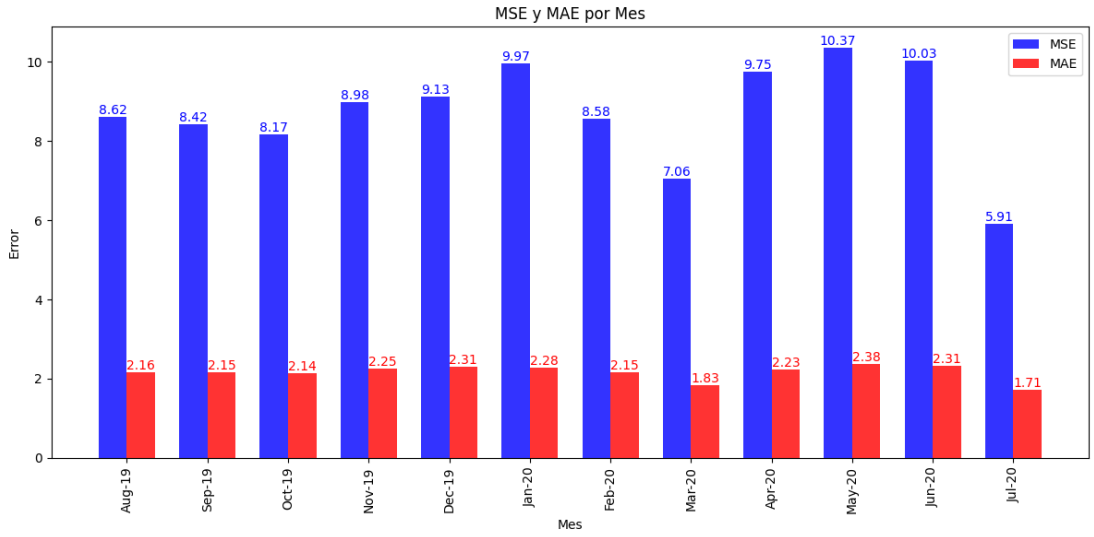
*Desempeño de modelo single step (1 pasos= con 1 VMD).*

| | Modelo | Hiperparametro | | | | Val_MAE | Val_mse |
|---|---|---|---|---|---|---|---|
| | | long_seq | n_units | n_hideen | Optimizer | | |
| 1 MODO | VMD 1 | 48 | 128 | 64 | Adam | 0.1446 | 0.0390 |
| | VMD 1 | 192 | 128 | 64 | Adam | 0.1439 | 0.0386 |
| | **VMD 1** | **336** | **128** | **64** | **Adam** | **0.1442** | **0.0383** |
| | VMD 1 | 144 | 256 | 64 | Adam | 0.1457 | 0.0388 |
| | VMD 1 | 144 | 128 | 64 | Adam | 0.1456 | 0.0385 |
| | VMD 1 | 144 | 32 | 64 | Adam | 0.1455 | 0.0386 |
| | VMD 1 | 144 | 32 | 32 | Adam | 0.1451 | 0.0385 |
| | VMD 1 | 144 | 32 | 16 | Adam | 0.1462 | 0.0387 |
| | VMD 1 | 144 | 32 | 8 | Adam | 0.1485 | 0.0390 |
| | VMD 1 | 144 | 32 | 32 | Adam | 0.1451 | 0.0385 |
| | VMD 1 | 144 | 32 | 32 | SGD | 0.1526 | 0.0421 |
| | VMD 1 | 144 | 32 | 32 | RMSProp | 0.1437 | 0.0388 |

Fuente: Elaboración propia.

**Figura 28**

*Agrupación de errores MSE y MAE por mes evaluado a 1 paso y 1 VMD.*




*Nota*. Elaboración propia.

✓ Modelo Bi Modal, de la Tabla 8 se selecciona los modelos de mejor desempeño y este se evalua en los datos de prueba y la Figura 29 resume el error promediado mensual.

**Tabla 8**

*Desempeño de modelo single-step (1 pasos) con 2 VMDs.*

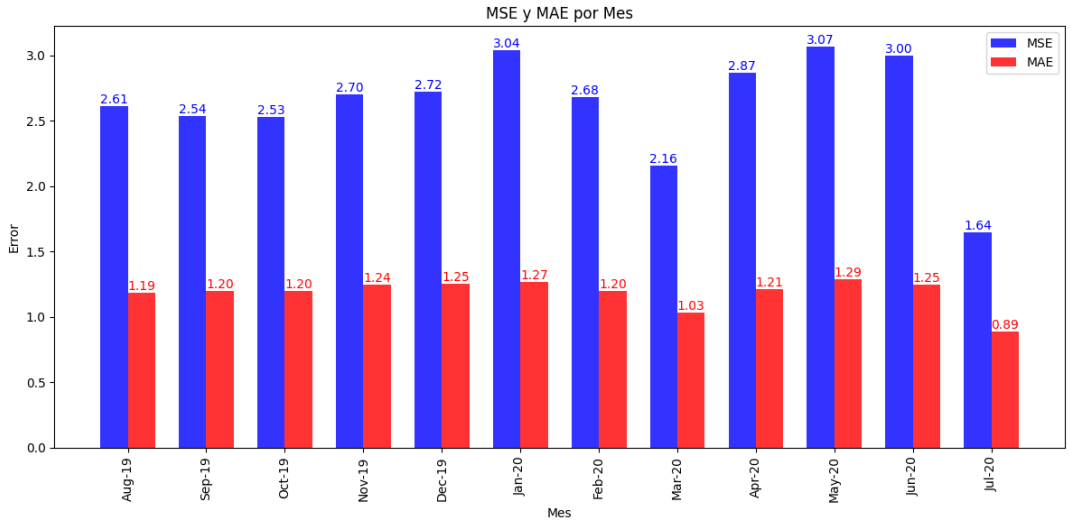
| | Modelo | Hiperparametro | | | | Val_MAE | Val_mse |
|---|---|---|---|---|---|---|---|
| | | long_seq | n_units | n_hideen | Optimizer | | |
| **2 MODOS** | 1 VMD | 48 | 128 | 64 | Adam | 0.0103 | 0.00018 |
| | 1 VMD | 192 | 128 | 64 | Adam | 0.0098 | 0.00017 |
| | 1 VMD | 336 | 128 | 64 | Adam | 0.0165 | 0.00049 |
| | 1 VMD | 192 | 256 | 64 | Adam | 0.0099 | 0.00016 |
| | 1 VMD | 192 | 128 | 64 | Adam | 0.0098 | 0.00017 |
| | 1 VMD | 192 | 32 | 64 | Adam | 0.0122 | 0.00023 |
| | **1 VMD** | **192** | **32** | **32** | **Adam** | **0.0100** | **0.00017** |
| | 1 VMD | 192 | 32 | 16 | Adam | 0.0141 | 0.00033 |
| | 1 VMD | 192 | 32 | 8 | Adam | 0.0293 | 0.00120 |
| | 1 VMD | 192 | 128 | 64 | Adam | 0.0098 | 0.00017 |
| | 1 VMD | 192 | 128 | 64 | SGD | 0.0482 | 0.00370 |
| | 1 VMD | 192 | 128 | 64 | RMSProp | 0.0187 | 0.00056 |
| | 2 VMD | 48 | 128 | 64 | Adam | 0.0168 | 0.00050 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 2 VMD | 192 | 128 | 64 | Adam | 0.0168 | 0.00049 |
| | 2 VMD | 336 | 128 | 64 | Adam | 0.0165 | 0.00049 |
| | **2 VMD** | **192** | **256** | **64** | **Adam** | **0.0165** | **0.00049** |
| | 2 VMD | 192 | 128 | 64 | Adam | 0.0168 | 0.00049 |
| | 2 VMD | 192 | 32 | 64 | Adam | 0.0166 | 0.00049 |
| | 2 VMD | 192 | 32 | 32 | Adam | 0.0174 | 0.00052 |
| | 2 VMD | 192 | 32 | 16 | Adam | 0.0175 | 0.00053 |
| | 2 VMD | 192 | 32 | 8 | Adam | 0.0177 | 0.00054 |
| | 2 VMD | 192 | 128 | 64 | Adam | 0.0168 | 0.00049 |
| | 2 VMD | 192 | 128 | 64 | SGD | 0.1144 | 0.02200 |
| | 2 VMD | 192 | 128 | 64 | RMSProp | 0.0168 | 0.00050 |

Fuente: Elaboración propia.

**Figura 29**

*Agrupación de errores MSE y MAE por mes evaluado a 1 paso y 2 VMDs.*




*Nota*. Elaboración propia.

## 6.2 Evaluación del modelo optimo Multi Step (24 steps ahead)

A continuación, se realizó una comparación de los tres modelos, tomando en cuenta los valores de rendimiento obtenidos en el proceso de validación.

- ✓ La Tabla 9, muestra los resultados de aplicar un solo modo de Descomposición Modo Variacional (VMD) en una red LSTM. Tras evaluar diversos hiperparámetros, se logró la máxima eficiencia con la siguiente configuración: longitud de secuencia de 48, 256

unidades, 64 capas ocultas y el optimizador Adam. Este conjunto logró minimizar el Error Absoluto Medio de Validación (Val_MAE) a 0.3895 y el Error Cuadrático Medio de Validación (Val_MSE) a 0.2303.

**Tabla 9**

Desempeño de modelo multi-step (24 pasos) con 1 VMD.

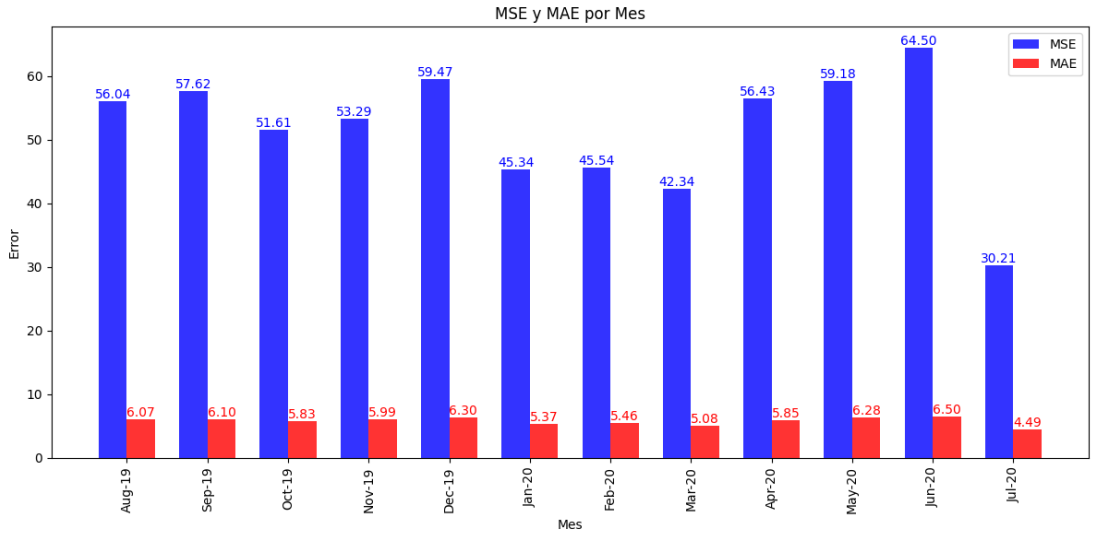
| | Modelo | Hiperparametro | | | | 24 steps ahead | |
|---|---|---|---|---|---|---|---|
| | | long_seq | n_units | n_hideen | Optimizer | Val_MAE | Val_mse |
| 1 MODO | VMD 1 | 48 | 128 | 64 | Adam | 0.3910 | 0.2320 |
| | VMD 1 | 192 | 128 | 64 | Adam | 0.3950 | 0.2366 |
| | VMD 1 | 336 | 128 | 64 | Adam | 0.3936 | 0.2381 |
| | **VMD 1** | **48** | **256** | **64** | **Adam** | **0.3895** | **0.2303** |
| | VMD 1 | 48 | 128 | 64 | Adam | 0.3912 | 0.2330 |
| | VMD 1 | 48 | 32 | 64 | Adam | 0.3936 | 0.2369 |
| | VMD 1 | 48 | 32 | 32 | Adam | 0.3967 | 0.2386 |
| | VMD 1 | 48 | 32 | 16 | Adam | 0.4002 | 0.2408 |
| | VMD 1 | 48 | 32 | 8 | Adam | 0.4044 | 0.2431 |
| | VMD 1 | 48 | 32 | 32 | Adam | 0.3997 | 0.2421 |
| | VMD 1 | 48 | 32 | 32 | SGD | 0.4750 | 0.3108 |
| | VMD 1 | 48 | 32 | 32 | RMSProp | 0.4006 | 0.2415 |

Fuente: Elaboración propia.

De la Tabla 9 se selecciona los modelos de mejor desempeño y este se evalúa en los datos de prueba y la Figura 30 resume el error promediado mensual.

**Figura 30**

Agrupación de errores MSE y MAE por mes evaluado a 24 pasos y 1 VMD.




Nota. Elaboración propia.

- ✓ La Tabla 10, muestra los resultados derivados de la aplicación de dos modos de Descomposición Modo Variacional (VMD) en una red LSTM. En ambos modos, se usó una longitud de secuencia (long_seq) de 144, con el optimizador Adam y diferentes configuraciones para el número de unidades (n_units) y capas ocultas (n_hidden).

  En el Modo 1, la configuración con 256 unidades y 32 capas ocultas produjo el menor Error Absoluto Medio de Validación (Val_MAE) y Error Cuadrático Medio de Validación (Val_MSE), con valores de 0.1367 y 0.0353, respectivamente.

  Por otro lado, el Modo 2 con 32 unidades y 16 capas ocultas, también consiguió reducir los errores de validación, reportando un Val_MAE de 0.1371 y un Val_MSE de 0.0341.

**Tabla 10**

*Desempeño de modelo multi-step (24 pasos) con 2 VMDs.*

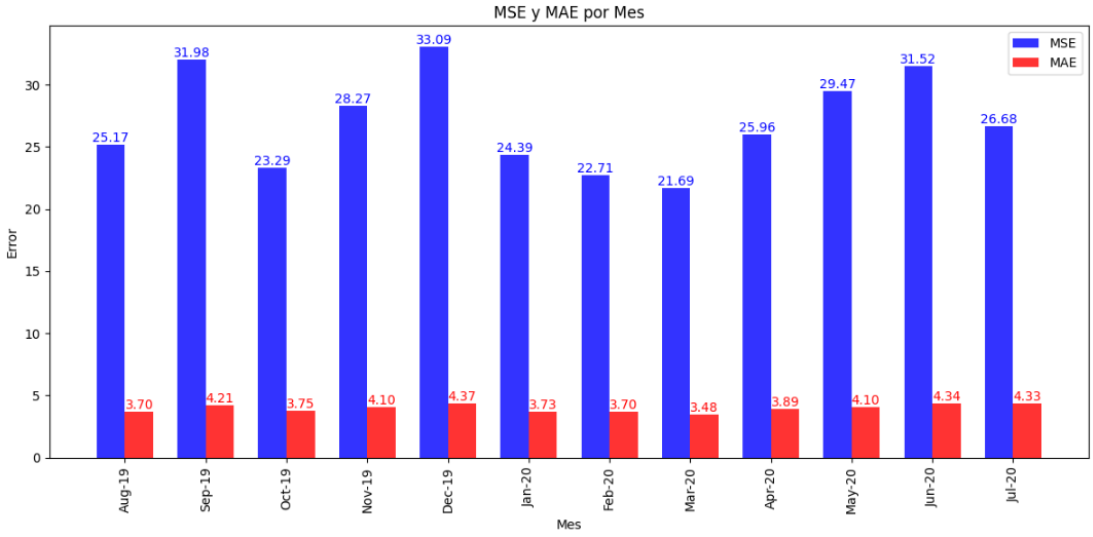
| | Modelo | Hiperparametro | | | | 24 steps ahead | |
|---|---|---|---|---|---|---|---|
| | | long_seq | n_units | n_hideen | Optimizer | Val_MAE | Val_mse |
| 2 MODOS | 1 VMD | 48 | 128 | 64 | Adam | 0.1447 | 0.0409 |
| | 1 VMD | 144 | 128 | 64 | Adam | 0.1388 | 0.0365 |
| | 1 VMD | 336 | 128 | 64 | Adam | 0.1314 | 0.0349 |
| | 1 VMD | 144 | 256 | 64 | Adam | 0.1360 | 0.0356 |
| | 1 VMD | 144 | 128 | 64 | Adam | 0.1425 | 0.0379 |
| | 1 VMD | 144 | 32 | 64 | Adam | 0.1417 | 0.0374 |
| | **1 VMD** | **144** | **256** | **32** | **Adam** | **0.1367** | **0.0353** |
| | 1 VMD | 144 | 256 | 16 | Adam | 0.1464 | 0.0400 |
| | 1 VMD | 144 | 256 | 8 | Adam | 0.1568 | 0.0442 |
| | 1 VMD | 144 | 256 | 32 | Adam | 0.1367 | 0.0353 |
| | 1 VMD | 144 | 256 | 32 | SGD | 0.3499 | 0.1752 |
| | 1 VMD | 144 | 256 | 32 | RMSProp | 0.1527 | 0.0430 |
| | 2 VMD | 48 | 128 | 64 | Adam | 0.1432 | 0.0368 |
| | 2 VMD | 144 | 128 | 64 | Adam | 0.1436 | 0.0369 |
| | 2 VMD | 336 | 128 | 64 | Adam | 0.1433 | 0.0370 |
| | 2 VMD | 144 | 256 | 64 | Adam | 0.1376 | 0.0347 |
| | 2 VMD | 144 | 128 | 64 | Adam | 0.1425 | 0.0365 |
| | 2 VMD | 144 | 32 | 64 | Adam | 0.1382 | 0.0346 |
| | 2 VMD | 144 | 32 | 32 | Adam | 0.1405 | 0.0355 |
| | **2 VMD** | **144** | **32** | **16** | **Adam** | **0.1371** | **0.0341** |
| | 2 VMD | 144 | 32 | 8 | Adam | 0.1429 | 0.0364 |
| | 2 VMD | 144 | 32 | 16 | Adam | 0.1371 | 0.0341 |
| | 2 VMD | 144 | 32 | 16 | SGD | 0.1889 | 0.0583 |
| | 2 VMD | 144 | 32 | 16 | RMSProp | 0.1443 | 0.0370 |

*Fuente*: Elaboración propia.

De la Tabla 10,se selecciona los modelos de mejor desempeño y este se evalúa en los datos de prueba y la Figura 31 resume el error promediado mensual.

**Figura 31**

Agrupación de errores MSE y MAE por mes evaluado a 24 pasos y 2 VMDs.




*Nota.* Elaboración propia.

- ✓ En la Tabla 11 se profundizo el análisis explorando tres variantes del método de Descomposición Modo Variacional (VMD) implementados en una red LSTM. Consistentemente con análisis anteriores, en todas las configuraciones se conservó una longitud de secuencia (long_seq) de 144 y se utilizó el optimizador Adam.

  En la variante 1, la configuración que incorporó 256 unidades y 32 capas ocultas generó los menores valores de Error Absoluto Medio de Validación (Val_MAE) y Error Cuadrático Medio de Validación (Val_MSE), reportando 0.1004 y 0.0205, respectivamente.

  En contraste, la variante 2, con una estructura de 128 unidades y 64 capas ocultas, logró igualmente reducir los errores de validación, obteniendo un Val_MAE de 0.1154 y un Val_MSE de 0.0261.

Finalmente, la variante 3, también con 64 capas ocultas, pero con 256 unidades, demostró una disminución en los errores de validación, presentando un Val_MAE de 0.1056 y un Val_MSE de 0.0215

**Tabla 11**

Desempeño de modelo multi-step (24 pasos) con 3 VMDs.

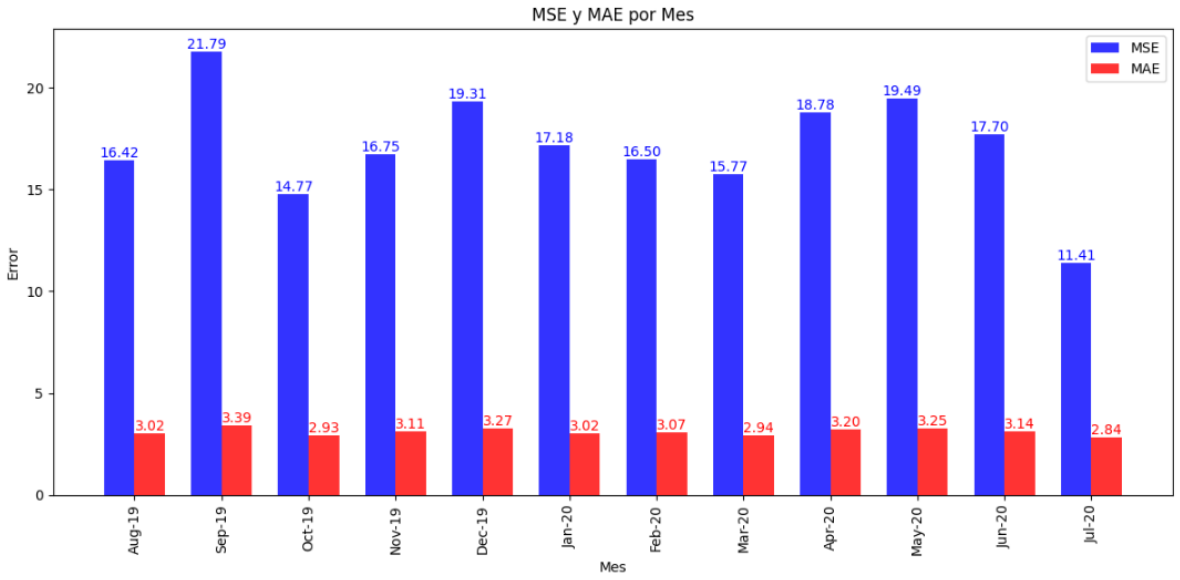
| | Modelo | Hiperparametro | | | | 24 steps ahead | |
|---|---|---|---|---|---|---|---|
| | | long_seq | n_units | n_hideen | Optimizer | Val_MAE | Val_mse |
| 3 MODOS | 1 VMD | 48 | 256 | 64 | Adam | 0.1010 | 0.0219 |
| | 1 VMD | 144 | 256 | 64 | Adam | 0.1036 | 0.0215 |
| | 1 VMD | 336 | 256 | 64 | Adam | 0.1046 | 0.0225 |
| | 1 VMD | 144 | 256 | 64 | Adam | 0.1036 | 0.0215 |
| | 1 VMD | 144 | 128 | 64 | Adam | 0.1116 | 0.0233 |
| | 1 VMD | 144 | 32 | 64 | Adam | 0.1139 | 0.0240 |
| | **1 VMD** | **144** | **256** | **32** | **Adam** | **0.1004** | **0.0205** |
| | 1 VMD | 144 | 256 | 16 | Adam | 0.1158 | 0.0255 |
| | 1 VMD | 144 | 256 | 8 | Adam | 0.1292 | 0.0307 |
| | 1 VMD | 144 | 128 | 64 | Adam | 0.1102 | 0.0234 |
| | 1 VMD | 144 | 128 | 64 | SGD | 0.2787 | 0.1159 |
| | 1 VMD | 144 | 128 | 64 | RMSProp | 0.1265 | 0.0281 |
| | 2 VMD | 48 | 256 | 64 | Adam | 0.1208 | 0.0278 |
| | 2 VMD | 144 | 256 | 64 | Adam | 0.1174 | 0.0267 |
| | 2 VMD | 336 | 256 | 64 | Adam | 0.1169 | 0.0268 |
| | 2 VMD | 144 | 256 | 64 | Adam | 0.1174 | 0.0267 |
| | **2 VMD** | **144** | **128** | **64** | **Adam** | **0.1154** | **0.0261** |
| | 2 VMD | 144 | 32 | 64 | Adam | 0.1181 | 0.0268 |
| | 2 VMD | 144 | 256 | 32 | Adam | 0.1207 | 0.0273 |
| | 2 VMD | 144 | 256 | 16 | Adam | 0.1246 | 0.0288 |
| | 2 VMD | 144 | 256 | 8 | Adam | 0.1218 | 0.0273 |
| | 2 VMD | 144 | 128 | 64 | Adam | 0.1192 | 0.0270 |
| | 2 VMD | 144 | 128 | 64 | SGD | 0.1979 | 0.0632 |
| | 2 VMD | 144 | 128 | 64 | RMSProp | 0.1258 | 0.0303 |
| | 3 VMD | 48 | 256 | 64 | Adam | 0.1071 | 0.0218 |
| | **3 VMD** | **144** | **256** | **64** | **Adam** | **0.1056** | **0.0215** |
| | 3 VMD | 336 | 256 | 64 | Adam | 0.1067 | 0.0218 |
| | 3 VMD | 144 | 256 | 64 | Adam | 0.1056 | 0.0215 |
| | 3 VMD | 144 | 128 | 64 | Adam | 0.1067 | 0.0218 |
| | 3 VMD | 144 | 32 | 64 | Adam | 0.1086 | 0.0224 |
| | 3 VMD | 144 | 256 | 32 | Adam | 0.1065 | 0.0218 |
| | 3 VMD | 144 | 256 | 16 | Adam | 0.1074 | 0.0220 |
| | 3 VMD | 144 | 256 | 8 | Adam | 0.1092 | 0.0225 |
| | 3 VMD | 144 | 128 | 64 | Adam | 0.1068 | 0.0218 |
| | 3 VMD | 144 | 128 | 64 | SGD | 0.1316 | 0.0307 |
| | 3 VMD | 144 | 128 | 64 | RMSProp | 0.1301 | 0.0302 |

Fuente: Elaboración propia.

De la Tabla 11 se selecciona los modelos de mejor desempeño y este se evalúa en los datos de prueba y la Figura 32 resume el error promediado mensual.

**Figura 32**

Agrupación de errores MSE y MAE por mes evaluado a 24 pasos y 3 VMDs.




*Nota.* Elaboración propia.

## 6.3 Evaluación del modelo optimo Multi Step (48 steps ahead)

A continuación, se realizó una comparación de los tres modelos, tomando en cuenta los valores de rendimiento obtenidos en el proceso de validación.

- ✓ La Tabla 12, muestra los resultados de aplicar un solo modo de Descomposición Modo Variacional (VMD) en una red LSTM. Tras evaluar diversos hiperparámetros, se logró la máxima eficiencia con la siguiente configuración: longitud de secuencia de 336, 128 unidades, 64 capas ocultas y el optimizador Adam. Este conjunto logró minimizar el Error Absoluto Medio de Validación (Val_MAE) a 0.4366 y el Error Cuadrático Medio de Validación (Val_MSE) a 0.2801.

**Tabla 12**

*Desempeño de modelo multi-step (48 pasos) con 1 VMD.*

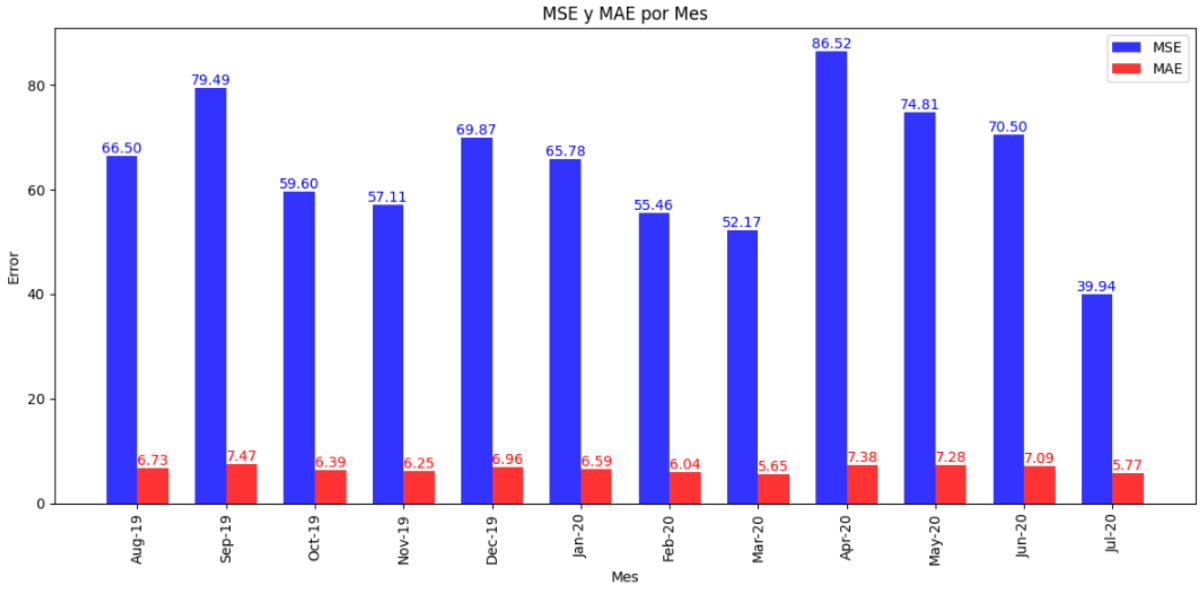
| | Modelo | Hiperparametro | | | | 48 steps | |
|---|---|---|---|---|---|---|---|
| | | long_seq | n_units | n_hideen | Optimizer | Val_MAE | Val_MSE |
| 1 MODO | VMD 1 | 48 | 128 | 64 | Adam | 0.4416 | 0.2847 |
| | VMD 1 | 144 | 128 | 64 | Adam | 0.4412 | 0.2849 |
| | **VMD 1** | **336** | **128** | **64** | **Adam** | **0.4366** | **0.2801** |
| | VMD 1 | 144 | 256 | 64 | Adam | 0.4435 | 0.2834 |
| | VMD 1 | 144 | 128 | 64 | Adam | 0.4390 | 0.2813 |
| | VMD 1 | 144 | 32 | 64 | Adam | 0.4456 | 0.2867 |
| | VMD 1 | 144 | 128 | 32 | Adam | 0.4436 | 0.2851 |
| | VMD 1 | 144 | 128 | 16 | Adam | 0.4507 | 0.2940 |
| | VMD 1 | 144 | 128 | 8 | Adam | 0.4547 | 0.2971 |
| | VMD 1 | 144 | 128 | 32 | Adam | 0.4436 | 0.2851 |
| | VMD 1 | 144 | 128 | 32 | SGD | 0.5011 | 0.3419 |
| | VMD 1 | 144 | 128 | 32 | RMSProp | 0.4519 | 0.2941 |

Fuente: Elaboración propia.

De la Tabla 12 se selecciona los modelos de mejor desempeño y este se evalúa en los datos de prueba y la Figura 33 resume el error promediado mensual.

**Figura 33**

*Agrupación de errores MSE y MAE por mes evaluado a 48 pasos y 1 VMD.*




*Nota.* Elaboración propia.

- ✓ La Tabla 13 muestra los resultados derivados de la aplicación de dos modos de Descomposición Modo Variacional (VMD) en una red LSTM, ampliando el análisis unimodal previamente presentado en la Tabla 12. En ambos modos, se usó una longitud de secuencia (long_seq) de 144, con el optimizador Adam y diferentes configuraciones para el número de unidades (n_units) y capas ocultas (n_hidden).

  En el Modo 1, la configuración con 256 unidades y 16 capas ocultas produjo el menor Error Absoluto Medio de Validación (Val_MAE) y Error Cuadrático Medio de Validación (Val_MSE), con valores de 0.2264 y 0.0893, respectivamente.

  Por otro lado, el Modo 2 con 32 unidades y 16 capas ocultas, también consiguió reducir los errores de validación, reportando un Val_MAE de 0.1655 y un Val_MSE de 0.0474.

**Tabla 13**

Desempeño de modelo multi-step (48 pasos) con 2 VMDs.

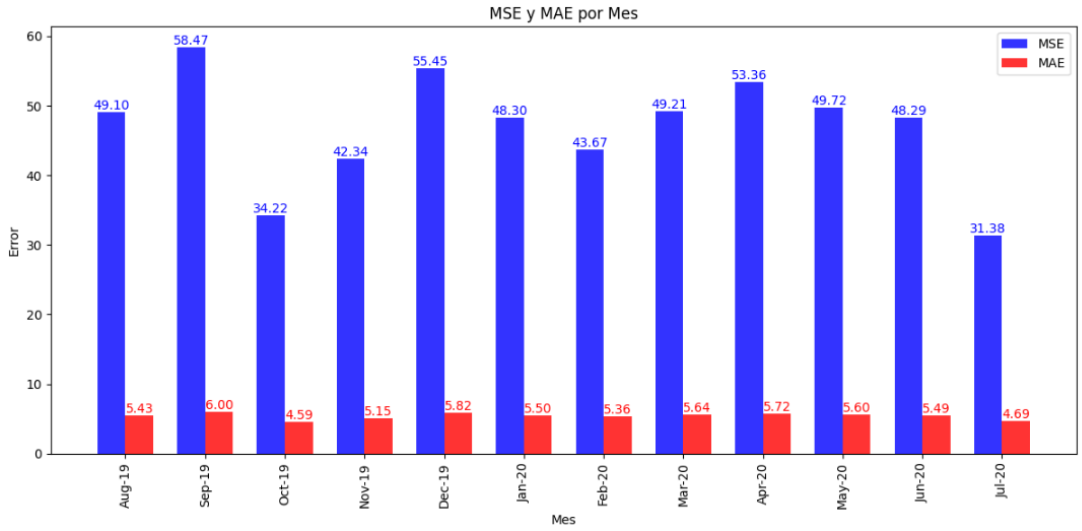
| | Modelo | Hiperparametro | | | | 48 steps | |
|---|---|---|---|---|---|---|---|
| | | long_seq | n_units | n_hideen | Optimizer | Val_MAE | Val_MSE |
| 2 MODOS | 1 VMD | 48 | 128 | 64 | Adam | 0.2343 | 0.0953 |
| | 1 VMD | 144 | 128 | 64 | Adam | 0.2281 | 0.0895 |
| | 1 VMD | 336 | 128 | 64 | Adam | 0.2249 | 0.0888 |
| | 1 VMD | 144 | 256 | 64 | Adam | 0.2218 | 0.0898 |
| | 1 VMD | 144 | 128 | 64 | Adam | 0.2337 | 0.0944 |
| | 1 VMD | 144 | 32 | 64 | Adam | 0.2239 | 0.0876 |
| | 1 VMD | 144 | 256 | 32 | Adam | 0.2231 | 0.0894 |
| | **1 VMD** | **144** | **256** | **16** | **Adam** | **0.2264** | **0.0893** |
| | 1 VMD | 144 | 256 | 8 | Adam | 0.2384 | 0.0975 |
| | 1 VMD | 144 | 256 | 32 | Adam | 0.2231 | 0.0894 |
| | 1 VMD | 144 | 256 | 32 | SGD | 0.3821 | 0.2062 |
| | 1 VMD | 144 | 256 | 32 | RMSProp | 0.2410 | 0.1014 |
| | 2 VMD | 48 | 128 | 64 | Adam | 0.1673 | 0.0481 |
| | 2 VMD | 144 | 128 | 64 | Adam | 0.1673 | 0.0482 |
| | 2 VMD | 336 | 128 | 64 | Adam | 0.1691 | 0.0489 |
| | 2 VMD | 144 | 256 | 64 | Adam | 0.1674 | 0.0482 |
| | **2 VMD** | **144** | **128** | **64** | **Adam** | **0.1655** | **0.0474** |
| | 2 VMD | 144 | 32 | 64 | Adam | 0.1709 | 0.0494 |
| | 2 VMD | 144 | 128 | 32 | Adam | 0.1674 | 0.0480 |
| | 2 VMD | 144 | 128 | 16 | Adam | 0.1688 | 0.0487 |
| | 2 VMD | 144 | 128 | 8 | Adam | 0.1713 | 0.0499 |
| | 2 VMD | 144 | 128 | 32 | Adam | 0.1674 | 0.0480 |
| | 2 VMD | 144 | 128 | 32 | SGD | 0.1791 | 0.0532 |
| | 2 VMD | 144 | 128 | 32 | RMSProp | 0.1824 | 0.0546 |

Fuente: Elaboración propia.

De la Tabla 13 se selecciona los modelos de mejor desempeño y este se evalúa en los datos de prueba y la Figura 34 resume el error promediado mensual.

**Figura 34**

Agrupación de errores MSE y MAE por mes evaluado a 48 pasos 2 VMDs.




Nota. Elaboración propia.

- ✓ En la Tabla 14, se profundizo el análisis explorando tres variantes del método de Descomposición Modo Variacional (VMD) implementados en una red LSTM. Consistentemente con análisis anteriores, en todas las configuraciones se conservó una longitud de secuencia (long_seq) de 144 y se utilizó el optimizador Adam.

  En la variante 1, la configuración que incorporó 256 unidades y 64 capas ocultas generó los menores valores de Error Absoluto Medio de Validación (Val_MAE) y Error Cuadrático Medio de Validación (Val_MSE), reportando 0.2010 y 0.0762, respectivamente.

  En contraste, la variante 2, con una estructura de 256 unidades y 64 capas ocultas, logró igualmente reducir los errores de validación, obteniendo un Val_MAE de 0.1624 y un Val_MSE de 0.0474.

Finalmente, la variante 3, también con 256 capas ocultas, pero con 64 unidades, demostró una disminución en los errores de validación, presentando un Val_MAE de 0.1185 y un Val_MSE de 0.0257.

**Tabla 14**

Desempeño de modelo multi-step (48 pasos) con 3 VMDs.

| | Modelo | Hiperparametro | | | | 48 steps | |
|---|---|---|---|---|---|---|---|
| | | long_seq | n_units | n_hideen | Optimizer | Val_MAE | Val_MSE |
| 3 MODOS | 1 VMD | 48 | 256 | 64 | Adam | 0.2026 | 0.0800 |
| | **1 VMD** | **144** | **256** | **64** | **Adam** | **0.2010** | **0.0762** |
| | 1 VMD | 336 | 256 | 64 | Adam | 0.1987 | 0.0769 |
| | 1 VMD | 144 | 256 | 64 | Adam | 0.2082 | 0.0820 |
| | 1 VMD | 144 | 128 | 64 | Adam | 0.2115 | 0.0835 |
| | 1 VMD | 144 | 32 | 64 | Adam | 0.2013 | 0.0766 |
| | 1 VMD | 144 | 256 | 32 | Adam | 0.2005 | 0.0771 |
| | 1 VMD | 144 | 256 | 16 | Adam | 0.2066 | 0.0823 |
| | 1 VMD | 144 | 256 | 8 | Adam | 0.2099 | 0.0836 |
| | 1 VMD | 144 | 128 | 64 | Adam | 0.2115 | 0.0835 |
| | 1 VMD | 144 | 128 | 64 | SGD | 0.3583 | 0.1838 |
| | 1 VMD | 144 | 128 | 64 | RMSProp | 0.2250 | 0.0937 |
| | 2 VMD | 48 | 256 | 64 | Adam | 0.1650 | 0.0481 |
| | 2 VMD | 144 | 256 | 64 | Adam | 0.1633 | 0.0480 |
| | 2 VMD | 336 | 256 | 64 | Adam | 0.1659 | 0.0492 |
| | **2 VMD** | **144** | **256** | **64** | **Adam** | **0.1624** | **0.0474** |
| | 2 VMD | 144 | 128 | 64 | Adam | 0.1630 | 0.0480 |
| | 2 VMD | 144 | 32 | 64 | Adam | 0.1690 | 0.0504 |
| | 2 VMD | 144 | 256 | 32 | Adam | 0.1654 | 0.0489 |
| | 2 VMD | 144 | 256 | 16 | Adam | 0.1748 | 0.0535 |
| | 2 VMD | 144 | 256 | 8 | Adam | 0.1758 | 0.0540 |
| | 2 VMD | 144 | 128 | 64 | Adam | 0.1630 | 0.0480 |
| | 2 VMD | 144 | 128 | 64 | SGD | 0.1938 | 0.0610 |
| | 2 VMD | 144 | 128 | 64 | RMSProp | 0.1913 | 0.0599 |
| | 3 VMD | 48 | 256 | 64 | Adam | 0.1175 | 0.0254 |
| | **3 VMD** | **144** | **256** | **64** | **Adam** | **0.1185** | **0.0257** |
| | 3 VMD | 336 | 256 | 64 | Adam | 0.1167 | 0.0252 |

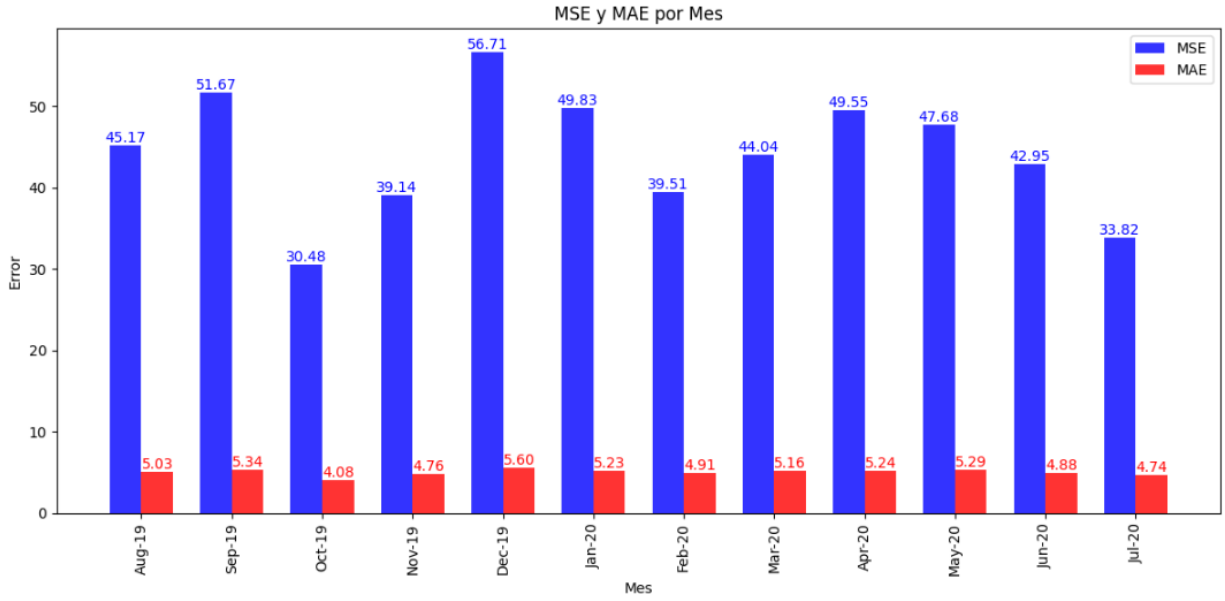
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 3 VMD | 144 | 256 | 64 | Adam | 0.1189 | 0.0259 |
| | 3 VMD | 144 | 128 | 64 | Adam | 0.1192 | 0.0260 |
| | 3 VMD | 144 | 32 | 64 | Adam | 0.1196 | 0.0261 |
| | 3 VMD | 144 | 256 | 32 | Adam | 0.1189 | 0.0258 |
| | 3 VMD | 144 | 256 | 16 | Adam | 0.1197 | 0.0262 |
| | 3 VMD | 144 | 256 | 8 | Adam | 0.1208 | 0.0265 |
| | 3 VMD | 144 | 256 | 64 | Adam | 0.1185 | 0.0257 |
| | 3 VMD | 144 | 256 | 64 | SGD | 0.1349 | 0.0318 |
| | 3 VMD | 144 | 256 | 64 | RMSProp | 0.1303 | 0.0303 |

Fuente: Elaboración propia.

De la Tabla 14 se selecciona los modelos de mejor desempeño y este se evalúa en los datos de prueba y la Figura 35 resume el error promediado mensual.

**Figura 35**

Agrupación de errores MSE y MAE por mes evaluado a 48 pasos y 3 VMDs.



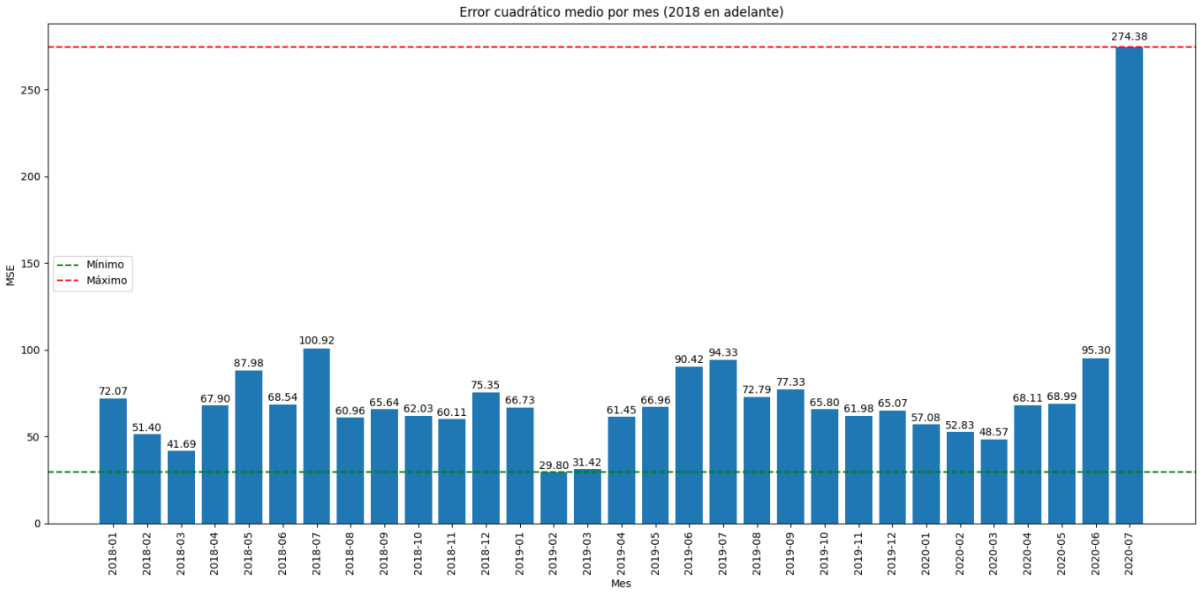

Nota. Elaboración propia.

## 6.4 Evaluación de exactitud del modelo de pronóstico actual

Desde enero 2018 a julio 2020, se evaluó la exactitud del modelo actual utilizado en la central eólica. Como se puede observar en la Figura 36 y Figura 37.

**Figura 36**

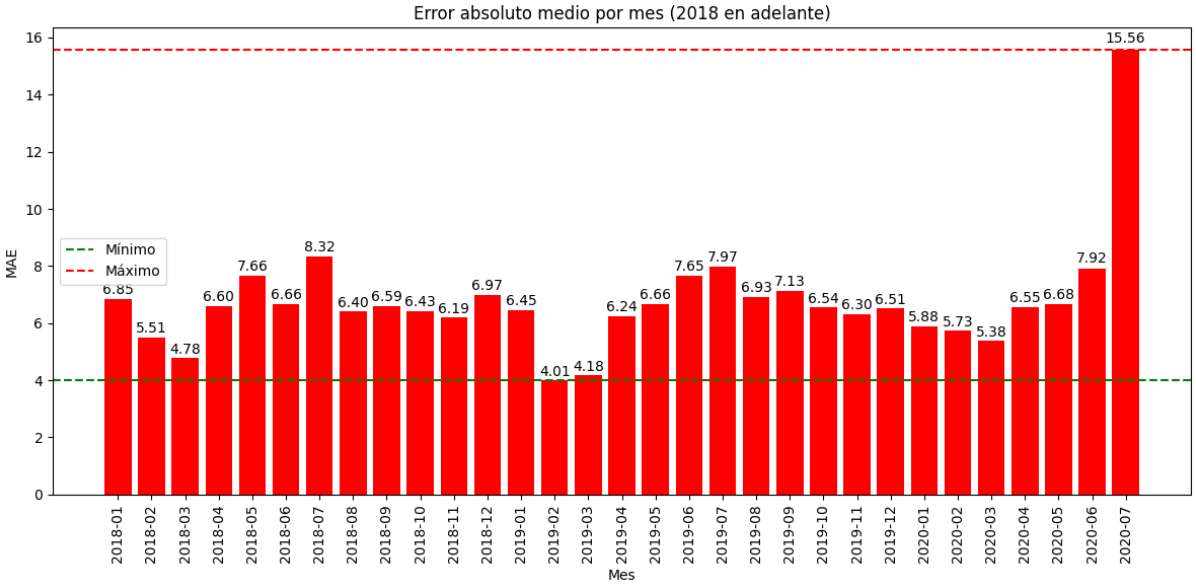
Agrupación de errores MSE por mes evaluado a 48 pasos de modelo actual




Nota. *Elaboración propia.*

**Figura 37**

Agrupación de errores MAE por mes evaluado a 48 pasos de modelo actual.




Nota. Elaboración propia.

## 6.5 Contrastación de hipótesis

En este estudio, la hipótesis formulada fue que el modelo de red neuronal VMD-LSTM reducirá el error en las predicciones de energía eólica en comparación con el método actual. Para verificar esta hipótesis, se llevó a cabo los siguientes pasos:

1. Se presenta el desempeño del modelo de pronóstico actualmente en uso en el parque eólico para los informes diarios de generación de energía. A continuación, se proporciona en la Tabla 15 y Tabla 16 el error MSE y MAE promediados mensualmente.

**Tabla 15**

*Error MSE del modelo actual de pronóstico promediado por mes.*

| | Erro MSE Modelo Actual Central Eólica | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Steps ahead** | **Aug-19** | **Sep-19** | **Oct-19** | **Nov-19** | **Dec-19** | **Jan-20** | **Feb-20** | **Mar-20** | **Apr-20** | **May-20** | **Jun-20** | **Jul-20** |
| 48 | 72.79 | 77.33 | 65.8 | 61.98 | 65.07 | 57.08 | 52.83 | 48.57 | 68.11 | 68.99 | 95.3 | 274.38 |

Fuente: Elaboración propia.

**Tabla 16**

*Error MAE del modelo actual de pronóstico promediado por mes.*

| | Erro MAE Modelo Actual Central Eólica | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Steps ahead** | **Aug-19** | **Sep-19** | **Oct-19** | **Nov-19** | **Dec-19** | **Jan-20** | **Feb-20** | **Mar-20** | **Apr-20** | **May-20** | **Jun-20** | **Jul-20** |
| 48 | 6.93 | 7.13 | 6.54 | 6.30 | 6.51 | 5.88 | 5.73 | 5.38 | 6.55 | 6.68 | 7.92 | 15.56 |

Fuente: Elaboración propia.

2. Posteriormente, se seleccionó y evaluó el modelo de mejor desempeño un LSTM con 3 VMDs. Cada modelo descompuesto se entrenó de forma independiente y, a continuación, se combinaron para formar un modelo integrado, en la Tabla 17 y Tabla 18, se observa el desempeño, donde observamos una reducción en el nivel de error en comparación con los resultados de la Tabla 15 y Tabla 16.

**Tabla 17**

*Error MSE promedio por mes del modelo entrenado de 3VMDs.*

| | Erro MSE con 3 VMD | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Steps ahead** | **Aug-19** | **Sep-19** | **Oct-19** | **Nov-19** | **Dec-19** | **Jan-20** | **Feb-20** | **Mar-20** | **Apr-20** | **May-20** | **Jun-20** | **Jul-20** |
| 1 | 2.14 | 1.96 | 2.07 | 2.16 | 2.11 | 2.40 | 2.10 | 1.63 | 2.23 | 2.45 | 2.36 | 1.17 |
| 24 | 16.43 | 21.34 | 14.56 | 16.07 | 18.81 | 17.23 | 16.53 | 14.88 | 17.85 | 19.45 | 16.88 | 9.4 |
| 48 | 47.57 | 54.02 | 31.3 | 40.33 | 57.63 | 46.79 | 38.79 | 40.95 | 50.19 | 45.99 | 44.5 | 37.15 |

Fuente: Elaboración propia.

**Tabla 18**

*Error MAE promedio por mes del modelo entrenado de 3VMDs.*

| | Erro MSE con 3 VMD | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Steps ahead** | **Aug-19** | **Sep-19** | **Oct-19** | **Nov-19** | **Dec-19** | **Jan-20** | **Feb-20** | **Mar-20** | **Apr-20** | **May-20** | **Jun-20** | **Jul-20** |
| 1 | 1.07 | 1.04 | 1.07 | 1.12 | 1.10 | 1.11 | 1.05 | 0.89 | 1.06 | 1.15 | 1.08 | 0.78 |
| 24 | 3.02 | 3.39 | 2.93 | 3.11 | 3.27 | 3.02 | 3.07 | 2.94 | 3.20 | 3.25 | 3.14 | 2.84 |
| 48 | 5.03 | 5.34 | 4.08 | 4.76 | 5.60 | 5.23 | 4.91 | 5.16 | 5.24 | 5.29 | 4.88 | 4.74 |

Fuente: Elaboración propia.

**Tabla 19**

*Tiempo que tarda el modelo LSTM con 3 VMDs en ser entrenado en los 03 horizontes temporales.*

| **Steps ahead** | **Tiempo de entrenamiento de modelo (segundos)** |
|---|---|
| 1 | 5307.0 |
| 24 | 66.1 |
| 48 | 39.7 |

Fuente: Elaboración propia.

Los datos presentados en las tablas anteriores corroboran la hipótesis inicial. Por ende, para el propósito de este estudio, la hipótesis se considera validada.

# Conclusiones

- El estudio evidencia la robustez y mayor precisión de los modelos VMD-LSTM en contraste con los modelos LSTM tradicionales (basados en un solo VMD) y el modelo que se utiliza actualmente en la central eólica. Esto sugiere una mejora significativa en el pronóstico de la producción de energía eólica. Las Tabla 15 y Tabla 17, muestran esta mejora evaluando el error MSE, mientras que la Tabla 16 y Tabla 18 lo hacen a través del error MAE.

- Al evaluar el desempeño de modelos LSTM-VMD para diferentes horizontes temporales, se consideraron resoluciones de 1, 24 y 48 pasos. Se observó que la precisión del modelo tiende a disminuir conforme se amplía el horizonte temporal, evidenciado en el rendimiento del modelo en la Tabla 18.

- La aplicación de la Descomposición de Modo Variacional (VMD) mejora el pronóstico de energía eólica en horizontes temporales más extensos. Aunque en un solo paso se observó una mejora con 2 modos, al introducir un tercer modo, el error aumentó y la precisión del pronóstico disminuyó.

- El tiempo necesario para entrenar un modelo está determinado tanto por el tamaño del conjunto de datos utilizado como por la demanda computacional en plataformas como Google Colab para el presente estudio. Al trabajar con conjuntos de datos más reducidos, el tiempo por época tiende a disminuir. Según la Tabla 19, los modelos destinados a pronósticos de 24 y 48 steps ahead adoptan un ventaneo con saltos que coinciden con su horizonte de predicción, es decir, saltos de 24 y 48, respectivamente. Esta estrategia previene la superposición de datos durante el entrenamiento. Por otro

lado, el modelo de 1 step ahead progresa de forma incremental, avanzando paso a paso.

- El avance en técnicas de aprendizaje automático, ejemplificado por los modelos VMD-LSTM, mostró una mejora significativa en la precisión de los pronósticos de la producción de energía eólica. La implementación de estas técnicas avanzadas puede aliviar algunas de las consecuencias negativas de las imprecisiones en los pronósticos, que incluyen programación ineficiente de mantenimiento, inestabilidad en la red eléctrica y desequilibrios entre la oferta y demanda.

- Para concluir, el enfoque adoptado en este estudio establece un marco sólido para futuras investigaciones. Estos estudios futuros podrían centrarse en optimizaciones adicionales del modelo, como la implementación de técnicas de aprendizaje automático más sofisticadas o el uso de conjuntos de datos más extensos y variados. Esto con el objetivo de mejorar aún más la precisión y la capacidad de generalización del modelo.

# Recomendaciones

- Optimizar los modelos propuestos mediante pruebas que ajusten los parámetros de sus respectivas capas.
- Evaluar los modelos LSTM-VMD utilizando diferentes técnicas de ventaneo, como el deslizante y el expansivo.
- Incorporar variables predictoras adicionales, como velocidad y dirección del viento, humedad relativa y temperatura, para enriquecer y mejorar la precisión de las predicciones.
- Potenciar el desempeño de las redes diseñadas ajustando parámetros clave. En este sentido, se recomienda considerar cambios en:
    - Long_seq
    - n_preds
    - Optimizador.
    - Learning rate.
    - Número de épocas y tamaño del batch.
    - etc.
- Llevar a cabo un estudio de los días en los que la red comete mayores errores en el pronóstico. Es esencial comprender la razón detrás de estas imprecisiones. Una vez identificados los patrones o características que hacen que esos días sean diferentes de los días normales, se pueden considerar nuevas variables para mejorar el aprendizaje de la red.

- Realizar un análisis detallado de los días y meses que presentan los mayores errores de predicción, con el objetivo de comprender sus particularidades. Posteriormente, se podría considerar la inclusión de variables nuevas que faciliten a la red neuronal distinguir estos días de los normales, mejorando de esta manera la precisión del modelo.

# Bibliografía

*Access to electricity – SDG7: Data and Projections – Analysis*. (2020). (IEA) Retrieved 10 2021, 11, from https://www.iea.org/reports/sdg7-data-and-projections/access-to-electricity

Alfares, H., & Nazeeruddin, M. (2015). ). Electric load forecasting: Literature survey and classification of methods. *Operations & Logistics*, 23-34.

Ameet, V. (2020). *Machine Learning and Artificial Intelligence.* Redmond, WA, USA: Springer.

Bengio, Y. (2012). Practical Recommendations for Gradient-Based Training of Deep Architectures. *Lecture Notes in Computer Science*, 437-478.

Brownlee, J. (2017). *Long Short-Term Memory Networks With Python.* Machine Learning Mastery.

Brownlee, J. (2019). *Deep Learning for Time Series Forecasting.* Sarah Martin.

Chang, W.-Y. (2014). A Literature Review of Wind Forecasting Methods. *Journal of Power and Energy Engineering*, 161-168. doi:http://dx.doi.org/10.4236/jpee.2014.24023

Chapagain, K., Kittipiyakul, S., & Kulthanavit, P. (2020). Short-Term Electricity Demand Forecasting: Impact Analysis of Temperature for Thailand. *Energies, 14*, 2498. doi:https://doi.org/10.3390/en13102498

COES. (2014). *Programación de la operacipon a corto plazo (PR-01).*

Dammert, A., Molinelli, F., & Carbajal, M. (2011). *Fundamentos técnicos y económicos del sector eléctrico peruano.*

Decreto Legislativo Nº 1002 . (2010, Setiembre 10). Diario Oficial El Peruano. Perú.

DGEE. (2019). *Balance Nacional de Energía.*

Ding, Y. (2020). *Data Science for Wind Energy.* Taylor & Francis.

Direccion General de Electricidad. (2019). *Anuario Ejecutivo de Electricidad.*

Dragomiretskiy, K., & Zosso, D. (2014). Variational Mode Decomposition. *IEEE Transactions on Signal Processing*, 531-544. doi:10.1109/tsp.2013.2288675

El Naqa, I., & Murphy, M. (2015). What Is Machine Learning? *Machine Learning in Radiation Oncology*, 3-11.

ELIA. (2018). *Wind-power generation data [EB/OL]*. Retrieved from http://www.elia.be

Enel Green Power. (2021). *Comentarios y propuestas de solución a los problemas de transmisión detectados en el informe de diagnóstico de las condiciones operativas del SEIN periodo 2023-2032 .*

Fan, G., Peng, L., & Hong, W. (2018). Short term load forecasting based on phase space reconstruction algorithm and bi-square kernel regression model. *ScienceDirect*, 13-33. doi:https://doi.org/10.1016/j.apenergy.2018.04.075

García Fernández, L. B. (2021). Modelamiento del pronóstico de la demanda eléctrica diaria del sistema eléctrico interconectado nacional utilizando técnicas de MACHINE LEARNING. *Universidad Nacional de Ingeniería*.

Giebel, G., & Kariniotakis, G. (2017). *Wind power forecasting—a review of the state of the art.* Renewable Energy Forecasting. doi:10.1016/b978-0-08-100504-0.00003-2

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press.

Gupta, A. (2023, 04 26). A Comprehensive Guide on Optimizers in Deep Learning. *Analytics Vidhya*. Retrieved from https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/#:~:text=An%20optimizer%20is%20a%20function,loss%20and%20improve%20the%20accuracy

Haykin, S. (2008). *Neural Networks and Learning Machines.* Pearson.

Hochreiter, S. (1998). The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6*(2). doi:10.1142/s0218488598000094

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation, 9*(8), 1735-1780. doi:10.1162/neco.1997.9.8.1735

Hong, T., Pinson, P., & Fan, S. (2016). Global energy forecasting competition 2012. *International Journal of Forecasting*, 896-913.

Hyndman, R., & Koehler, A. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*. doi:10.1016/J.IJFORECAST.2006.03.001

IEA. (2020). *The role of CCUS in low-carbon power systems.* Paris. Retrieved from https://www.iea.org/reports/the-role-of-ccus-in-low-carbon-power-systems

International Energy Agency. (2020). *Electricity Information: Overview.*

IRENA. (2021). *Renewable energy statistics 2021.*

Karpathy, A. (2015, Mayo 21). *The Unreasonable Effectiveness of Recurrent Neural Networks*. Retrieved from https://karpathy.github.io/2015/05/21/rnn-effectiveness/

Kumar, V., Pandey, A., & Sinha, S. (2016). Grid integration and power quality issues of wind and solar energy system: A review. *2016 International Conference on Emerging Trends in Electrical Electronics & Sustainable Energy Systems (ICETEESES)*, 71-80. doi:10.1109/iceteeses.2016.7581355

Lee, J. (2013, Diciembre 5). Introduction to Artificial Neural Networks - Part 1. Retrieved from https://www.theprojectspot.com/tutorial-post/introduction-to-artificial-neural-networks-part-1/7

Li, H., Huitian, J., Rongchang, Z., & Zhiyu, G. (2019). Wind power forecast based on improved Long Short Term Memory. *Elsevier, 189*. doi:https://doi.org/10.1016/j.energy.2019.116300

Liu, Y., Guan, L., & Hou, C. (2019). Wind Power Short-Term Prediction Based on LSTM and Discrete Wavelet Transform. *Applied Sciences*, 1108. doi:https://doi.org/10.3390/app9061108

Loaiza Muñoz, J. E. (2019). *PREDICCIÓN A CORTO PLAZO DE LA DEMANDA DE ENERGÍA.*

Manwell, J., McGowan, J., & Rogers, A. (2009). *Wind Energy Explained: Theory, Design and Application.* Chichester, UK: John Wiley & Sons.

Marcelo Barreto, E. A., & VillarrealEscate, L. G. (2021). El modelo estocástico univariante ARIMA como herramienta predictiva de la demanda de energía eléctrica residencial del sistema eléctrico Cusco. *Universidad Nacional de Ingeniería*.

MINEM. (2016). *Atlas Eólico del Perú.* Lima.

Oracle. (2018, Julio 11). *Oracle Big Data Blog*. Retrieved from https://blogs.oracle.com/bigdata/difference-ai-machine-learning-deep-learning

Ordoudis, C., Pinson, P., & Morales, J. (2017). An integrated market for electricity and natural gas systems with stochastic power producers. *Applied Energy*, 230-240.

Osinergmin. (2018, Marzo). *Osinergmin*. Retrieved from https://www.osinergmin.gob.pe/seccion/centro_documental/electricidad/Documentos/PROYECTOS%20GFE/Acorde%C3%B3n/Generaci%C3%B3n/1.7.3.pdf

Osinergmin. (2020, 12 26). *https://www.osinergmin.gob.pe/*. Retrieved from https://www.osinergmin.gob.pe/: https://www.osinergmin.gob.pe/empresas/energias-renovables/energia-eolica/que-es-la-energia-eolica

Pereira De Azevedo, L. (2016). Aplicación De Redes Neuronales Artificiales En El Proceso De Orquídeas Del Género Cattleya. Retrieved from https://www.ifmg.edu.br/sabara/biblioteca/trabalhos-de-conclusao-de-curso/tcc-documentos/TCCLucasAzevedo.pdf

Queguiner, J.-L. (2020). *What does Training Neural Networks mean?* OVHcloud Blog. Retrieved from https://blog.ovhcloud.com/what-does-training-neural-networks-mean/

Ray, S. (2019). A Quick Review of Machine Learning Algorithms. *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*. doi:10.1109/comitcon.2019.8862451

Rico, C., Paredes, M., & Fernandez, N. (2009). *Modeling of the Hierarchical Structure of Freshwater Macroinvertebrates Using Artificial Neural Networks.* Colombia: Acta Biológica Colombiana.

Siddharth, S., Simone, S., & Anidhya, A. (2020). Activation functions in neural. *International Journal of Engineering Applied Sciences and Technology*, 310-316.

Sidhu, R. (2020). *Use of Cross Validation in Machine Learning.* Retrieved from AI Graduate: https://aigraduate.com/use-of-cross-validation-in-machine-learning/

Tino, P., Benuskova, L., & Sperduti, A. (2015). *Springer Handbook of Computational Intelligence.* doi:10.1007/978-3-662-43505-2_27

V Joshi, A. (2020). *Machine Learning and Artificial Intelligence.* Springer. doi:https://doi.org/10.1007/978-3-030-26622-6

Wang, Y., Zhang, N., & Kang, C. (2018). Optimal scheduling of generation and demand response considering wind power uncertainty. *Applied Energy*, 924-933.

Wang, Y., Zou, R., Liu, F., Zhang, L., & Liu, Q. (2021). A review of wind speed and wind power forecasting with deep neural networks. *Applied Energy*, 117766. doi:10.1016/j.apenergy.2021.117766

Worl Economic Forum. (2021). *Fostering Effective Energy Transition.* Worl Economic Forum.

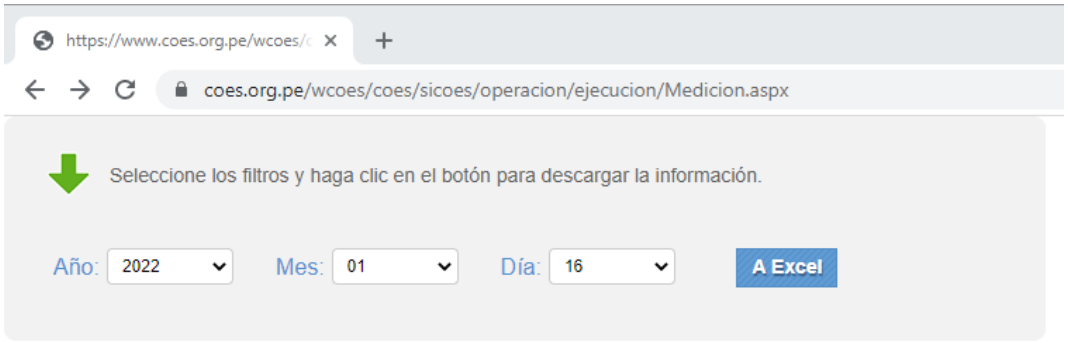
Yaghoubirad, M., Azizi, N., Farajollahi, M., & Ahmadi, A. (2023). Deep learning-based multistep ahead wind speed and power generation forecasting using direct method. *Energy Conversion and Management*, 281, 116760. Retrieved from https://doi.org/10.1016/j.enconman.2023.116760

Zadranská, L. (2019). Time Series Forecasting using Deep Neural Networks. *University of West Bohemia*.

# Anexos

Anexo 1: Código de programación para webscraping

El siguiente código es utilizado para recuperar solo la información referente a la central eólica de talara.




```
# importamos librerias a utilizar
from selenium import webdriver
from selenium.webdriver.support.ui import Select
import pandas as pd
import os
from webdriver_manager.chrome import ChromeDriverManager
from datetime import datetime, timedelta
```

```
# simulamos entorno chrome para webscraping
driver = webdriver.Chrome(ChromeDriverManager().install())
```

```
# Direccionamos hacia url y ejecutamos comandos
driver.get('https://www.coes.org.pe/wcoes/coes/sicoes/operacion/ejecucion/Medicion.aspx')
year_element=driver.find_element_by_id("DropDownListYears")
year=Select(year_element)
mes_element=driver.find_element_by_id("DropDownListMonths")
mes=Select(mes_element)
dia_element=driver.find_element_by_id("DropDownListDay")
dia=Select(dia_element)
link=driver.find_element_by_id("ButtonGenerarMedicion")
#link.click()
```

```
# Definimos periodo a descargar
inicio = datetime(2016,1,1)
fin    = datetime(2020,7,2)
print(inicio)
print(fin)
delta = fin - inicio
print (delta.days)
```

```
rng = pd.date_range(inicio, periods=delta.days, freq='D')
df = pd.DataFrame({ 'Date': rng})
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day
df['Month'] = df['Date'].dt.strftime('%m')
df['Day'] = df['Date'].dt.strftime('%d')
```

```
# Ejecutamos ciclo for para descargar datos de periodo definido
for i in range(0, len(df), 1):
    year.select_by_visible_text(str(df.loc[ i,'Year' ]))
    mes.select_by_visible_text(str(df.loc[ i,'Month' ]))
    dia.select_by_visible_text(str(df.loc[ i,'Day' ]))
    link.click()
```

```
import pandas as pd
import glob

path = r'C:\Users\nisanchez\Desktop\Scrap-ejecutado' # use your path
all_files = glob.glob(path + "/*.csv")

li = []

for filename in all_files:
    df = pd.read_csv(filename, skiprows=2,index_col=None, header=0)
    li.append(df)

frame = pd.concat(li, axis=0, ignore_index=True)
```

```
df=frame[['Hora','PQE-EOLICO-TALARA']]
df.to_csv("WBS_Talara.csv")
```

| | Hora | PQE-EOLICO-TALARA |
|---|---|---|
| **0** | 2016-01-01 00:30:00 | 14.33674 |
| **1** | 2016-01-01 01:00:00 | 14.35624 |
| **2** | 2016-01-01 01:30:00 | 13.45760 |
| **3** | 2016-01-01 02:00:00 | 13.98033 |
| **4** | 2016-01-01 02:30:00 | 17.73316 |
| **...** | ... | ... |
| **78859** | 2020-07-01 22:00:00 | 24.64462 |
| **78860** | 2020-07-01 22:30:00 | 24.59254 |
| **78861** | 2020-07-01 23:00:00 | 20.93133 |
| **78862** | 2020-07-01 23:30:00 | 17.81378 |
| **78863** | 2020-07-02 00:00:00 | 10.12704 |

78864 rows × 2 columns

Anexo 2: Código de programación de preparación de datos

```
# Importamos librerías
!pip install vmdpy
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
from vmdpy import VMD
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from oauth2client.client import GoogleCredentials

# Montamos Google Drive
from google.colab import drive
drive.mount("/content/drive", force_remount=True)

# Autentificamos y creamos el cliente PyDrive
from google.colab import auth
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)

# Función para cargar los datos
def load_data(id, usecols, new_col_name):
    df = pd.read_csv(id,header=0, sep=',', usecols=usecols, low_memory=False, parse_dates={'datetime':[0]}, index_col=['datetime'])
    df.columns = [new_col_name]
    df.loc[df[new_col_name] > 33, new_col_name] = np.nan
    df = df.astype('float32')
    return df

# Función para llenar los valores faltantes
def fill_missing(values):
    one_day = 60 * 24
    for row in range(values.shape[0]):
        for col in range(values.shape[1]):
            if np.isnan(values[row, col]):
                values[row, col] = values[row - one_day, col]

# Dirección para obtener el id de los datos históricos
listed = drive.ListFile({'q': "title contains '.csv' and '1Ins2lYJnbwT7XxeEG-uXdo2Vlkd_ZGOx' in parents"}).GetList()

```

```python
# Listamos los archivos tipo ".csv"
file_id=[]
downloaded=[]
for file in listed:
  file_id.append(file['id'])
  downloaded = drive.CreateFile({'id': file['id']})
  downloaded.GetContentFile(file['id'])

  print('title {}, id {}'.format(file['title'], file['id']))

# Lista de los archivos ".csv"
file_id=[]
downloaded=[]
for file in listed:
    file_id.append(file['id'])
    downloaded = drive.CreateFile({'id': file['id']})
    downloaded.GetContentFile(file['id'])

# Cargamos los datos
dataset_despacho_ejecutado = load_data(file_id[0], [1,2], 'Despacho_ejecutado')
dataset_despacho_programado = load_data(file_id[1], [1,2], 'Despacho_programado')
dataset_despacho_ejecutado.head()

dataset_despacho_programado.head()

# Rellenamos los valores faltantes
fill_missing(dataset_despacho_ejecutado.values)
fill_missing(dataset_despacho_programado.values)
```

```python
def perform_vmd(set_power, K=2, alpha=200, tau=0., DC=0, init=1, tol=1e-7):
    '''
    Aplica la Descomposición de Modos Variacionales (VMD) a los datos de entrada.

    Parámetros:
        set_power (DataFrame): Los datos a descomponer.
        K (int): El número de modos en los que descomponer los datos. Si K=1, la señal original será la salida.
        alpha (float): Parámetro de ancho de banda.
        tau (float): Parámetro de tolerancia al ruido.
        DC (int): Si se impone una parte de DC.
        init (int): Si se inicializan las omegas de manera uniforme.
        tol (float): Tolerancia para la convergencia del algoritmo.

    Devuelve:
        DataFrame: Un DataFrame que contiene los modos descompuestos, la señal reconstruida y la señal original.
    '''
    # Si K=1, retornamos la señal original
    if K == 1:
        data_VMD = set_power.copy()
        data_VMD.columns = ['Original']
        return data_VMD

    # Ejecutamos VMD
    f = set_power.values.flatten()
    u, u_hat, omega = VMD(f, alpha, tau, K, DC, init, tol)

    # Comprobamos si los datos y el índice tienen la misma longitud
    if u.T.shape[0] != len(set_power):
        u = u[:,:-1] if u.T.shape[0] > len(set_power) else np.hstack((u, np.zeros((K,1))))

    # Preparamos las columnas del DataFrame de salida
    Column = ['Mode '+str(i) for i in range(1, K+1)]
    data_VMD = pd.DataFrame(u.T, columns=Column, index=set_power.index)

    # Calculamos la señal reconstruida como la suma de los modos
    data_VMD['Reconstructed'] = np.sum(data_VMD, axis=1)

    # Añadimos la señal original al DataFrame
    data_VMD['Original'] = set_power.values

    return data_VMD
```

```python
# Aplicamos VMD
m=1
dataset_despacho_ejecutado = dataset_despacho_ejecutado.dropna()
data_VMD = perform_vmd(dataset_despacho_ejecutado, K=m)

# Crear una nueva columna 'date' con los valores del índice actual
data_VMD['date'] = dataset_despacho_ejecutado.index

# Convertir la columna 'date' al formato datetime
data_VMD['date'] = pd.to_datetime(data_VMD['date'])

# Reordenar las columnas para que 'date' sea la primera
cols = data_VMD.columns.tolist()
cols = cols[-1:] + cols[:-1]  # Coloca 'date' al principio
data_VMD = data_VMD[cols]

# Resetear el índice del DataFrame a números enteros por defecto
data_VMD = data_VMD.reset_index(drop=True)

data_VMD.head()

# Guardamos los conjuntos de entrenamiento y prueba en archivos CSV
TRAIN_SIZE = 0.8
idx = round(len(data_VMD)*TRAIN_SIZE)

data_train = data_VMD[:idx]
data_test=data_VMD[idx:]

# Escribe los datos de entrenamiento en un archivo CSV
train_file = '/content/drive/MyDrive/Colab Notebooks/Webscrap Data COES/Data '+ str(m) +' Modo/train_file_' + str(m) +'modos.csv'
data_train.to_csv(train_file, index=True)

# Escribe los datos de prueba en un archivo CSV
test_file = '/content/drive/MyDrive/Colab Notebooks/Webscrap Data COES/Data '+ str(m) +' Modo/test_file_' + str(m) + 'modos.csv'
data_test.to_csv(test_file, index=True)
```

Anexo 3: Código de programación de modelo LSTM-VMD

```python
# Selecciona el numero de modos
num_columnas = 3

archivo_train_csv = f'/content/drive/MyDrive/Colab Notebooks/Webscrap Data COES/Data {num_columnas} Modo/train_file_{num_columnas}modos.csv'
archivo_test_csv = f'/content/drive/MyDrive/Colab Notebooks/Webscrap Data COES/Data {num_columnas} Modo/test_file_{num_columnas}modos.csv'

cols = [1,2] if num_columnas == 1 else [i for i in range(1, num_columnas + 4)]
data_train = pd.read_csv(archivo_train_csv, usecols=cols, index_col=0)
data_test = pd.read_csv(archivo_test_csv, usecols=cols, index_col=0)

if num_columnas == 1:
  # Cambiar el nombre de una columna en data_train
  data_train = data_train.rename(columns={'Original': 'Mode 1'})
  data_test = data_test.rename(columns={'Original': 'Mode 1'})

data_train.head()

data_test.head()
```

```python
# Diccionario de parámetros
params = [
  {
        'LONG_SEC': 3*48,  # Longitud de la secuencia de entrada
        'N_UNITS': 256,  # Número de unidades en la capa LSTM
        'N_UNITS_HIDDEN': 32,  # Número de unidades en la capa oculta LSTM
        'epoch_number': 100,  # Número de épocas para el entrenamiento
        'batch_size': 256,  # Tamaño del lote para el entrenamiento
        'NN_model': {  # Parámetros para el modelo de red neuronal
            'optimizer': {'type': 'Adam', 'lr': 0.001},  # Optimizador y tasa de aprendizaje
            'metrics': {'metrics': ['mean_absolute_error','mse']},  # Métricas para evaluar el modelo
            'loss': {'loss': ['mse']}  # Función de pérdida
        },
        'n_preds': 48,  # Número de predicciones a realizar
        'seed': 88  # Semilla para la generación de números aleatorios
    },
    {
        'LONG_SEC': 3*48,
        'N_UNITS': 128,
        'N_UNITS_HIDDEN': 64,
        'epoch_number': 100,
        'batch_size': 32,
        'NN_model': {
            'optimizer': {'type': 'Adam', 'lr': 0.001},
            'metrics': {'metrics': ['mean_absolute_error','mse']},
            'loss': {'loss': ['mse']}
        },
        'n_preds': 48,  # Número de predicciones
        'seed': 88
    },
    {
        'LONG_SEC': 3*48,
        'N_UNITS': 256,
        'N_UNITS_HIDDEN': 64,
        'epoch_number': 100,
        'batch_size': 256,
        'NN_model': {
            'optimizer': {'type': 'Adam', 'lr': 0.001},
            'metrics': {'metrics': ['mean_absolute_error','mse']},
            'loss': {'loss': ['mse']}
        },
        'n_preds': 48,  # Número de predicciones
        'seed': 88
    }
]
```

```python
# Selecciona los primeros 'num_columnas' elementos de params
params = params[:num_columnas]

def get_sequence_data(data_item, long_sequence, n_preds):
    X, Y = [], []
    for i in range(0, len(data_item) - long_sequence - n_preds + 1, n_preds):
        X.append(data_item[i:i + long_sequence].reshape((long_sequence, -1)))
        Y.append(data_item[i + long_sequence:i + long_sequence + n_preds].reshape((-1, n_preds, 1)))
    X = np.array(X)
    Y = np.array(Y)
    return X, Y

def build_model(params):
    # Obtención de los parámetros para la construcción del modelo de la red LSTM
    N_UNITS = params['N_UNITS']
    N_UNITS_HIDDEN = params['N_UNITS_HIDDEN']
    LONG_SEC = params['LONG_SEC']
    optimizador = params['NN_model']['optimizer']['type']
    loss = params['NN_model']['loss']['loss']
    metrics = params['NN_model']['metrics']['metrics']

    # Semilla de los generadores aleatorios
    tf.random.set_seed(params['seed'])
    np.random.seed(params['seed'])

    # Definición del modelo de la red LSTM
    modelo = Sequential()
    modelo.add(LSTM(N_UNITS, input_shape=(LONG_SEC, 1), return_sequences=True, activation='tanh')) # Capa LSTM con unidades N_UNITS
    modelo.add(Dropout(0.2))

    modelo.add(LSTM(N_UNITS_HIDDEN, return_sequences=True, activation='tanh')) # Capa LSTM oculta con unidades N_UNITS_HIDDEN y activación relu
    modelo.add(Dropout(0.2)) # Capa de regularización de Dropout

    modelo.add(LSTM(N_UNITS_HIDDEN, activation='tanh')) # Capa LSTM oculta con unidades N_UNITS_HIDDEN y activación relu
    modelo.add(Dropout(0.2)) # Capa de regularización de Dropout

    # Se agrega la capa TimeDistributed para que la capa Dense actúe sobre cada valor de predicción por separado
    modelo.add(Dense(units=params['n_preds'], activation='tanh'))

    modelo.add(Dense(units=params['n_preds'], activation='linear'))

    # Compilación del modelo
    modelo.compile(optimizer=optimizador, loss=loss, metrics=metrics)
    print(modelo.summary())
    return modelo
```

```python
def train_model(params, data_train, data_test, name):
    EPOCHS = params['epoch_number']
    BATCH_SIZE = params['batch_size']
    LONG_SEC = params['LONG_SEC']
    n_preds = params['n_preds']  # Número de predicciones

    scaler = MinMaxScaler(feature_range=(-1, 1))
    train_scaled = scaler.fit_transform(data_train.values.reshape(-1, 1))
    test_scaled = scaler.transform(data_test.values.reshape(-1, 1))

    x_train, y_train = get_sequence_data(train_scaled, LONG_SEC, n_preds)
    x_test, y_test = get_sequence_data(test_scaled, LONG_SEC, n_preds)

    y_train = np.reshape(y_train, (y_train.shape[0], y_train.shape[2]))
    y_test = np.reshape(y_test, (y_test.shape[0], y_test.shape[2]))

    print("Dimensiones de x_train y y_train:", x_train.shape, y_train.shape)
    print("Dimensiones de x_test y y_test:", x_test.shape, y_test.shape)

    model = build_model(params)

    history = model.fit(x_train, y_train, batch_size=BATCH_SIZE, epochs=EPOCHS, validation_data=(x_test, y_test))

    model.save(f'/content/drive/MyDrive/Colab Notebooks/Webscrap Data COES/Multi step model/{name}.h5')
    with open(f'/content/drive/MyDrive/Colab Notebooks/Webscrap Data COES/Multi step model/scaler_{name}.pkl', 'wb') as f:
        pickle.dump(scaler, f)

    show_history(history.history, EPOCHS)
    print(f'[Model] Training Completed. Model saved as {name}')
    return model, scaler, x_test, y_test

def show_history(history_dict, epochs):
    loss = history_dict['loss']
    val_loss = history_dict['val_loss']
    epochs = np.arange(0, epochs)
    plt.figure()
    plt.plot(epochs, loss, 'bo', label='Training loss')
    plt.plot(epochs, val_loss, 'b', label='Validation loss')
    plt.title('Training and validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
```

```python
def plot_prediction(y_test_inv, y_pred_inv, num_samples):
    plt.figure(figsize=(5, 3))
    plt.plot(y_test_inv[:num_samples], label='Valor Real')
    plt.plot(y_pred_inv[:num_samples], label='Predicción')
    plt.legend()
    plt.show()

def calculate_mse(y_test_inv, y_pred_inv, num_samples):
    y_test_inv = y_test_inv[:num_samples]
    y_pred_inv = y_pred_inv[:num_samples]
    mse = mean_squared_error(y_test_inv, y_pred_inv)
    print(f'Error cuadrático medio: {mse}')

# Entrenamiento de modelos
models = {}     # Diccionario para almacenar los modelos entrenados
scalers = {}    # Diccionario para almacenar los escaladores utilizados

for i, param in enumerate(params, 1):
    if f'Mode {i}' in data_train.columns and f'Mode {i}' in data_test.columns:
        data_train_mode = data_train[f'Mode {i}']        # Datos de entrenamiento para el modo i
        data_test_mode = data_test[f'Mode {i}']          # Datos de prueba para el modo i
        model_name = f'modelo_lstm_{i}_{i}'              # Nombre del modelo

        model, scaler, x_test, y_test = train_model(param, data_train_mode, data_test_mode, model_name)  # Entrenamiento del modelo
        models[model_name] = model                       # Almacenar el modelo en el diccionario
        scalers[model_name] = scaler                     # Almacenar el escalador en el diccionario

# Predicción y evaluación de modelos
predictions = []

for i, param in enumerate(params, 1):
    model_name = f'modelo_lstm_{i}_{i}'              # Nombre del modelo
    model = models[model_name]                       # Obtener el modelo correspondiente al nombre
    scaler = scalers[model_name]                     # Obtener el escalador correspondiente al nombre
    data_test_mode = data_test[f'Mode {i}']          # Datos de prueba para el modo i

    x_test, y_test_seq = get_sequence_data(scaler.transform(data_test_mode.values.reshape(-1, 1)), param['LONG_SEC'], param['n_preds'])
    y_test_seq = scaler.inverse_transform(y_test_seq.reshape(-1, param['n_preds'])).reshape(-1, 1)

    y_pred = scaler.inverse_transform(model.predict(x_test).reshape(-1, param['n_preds'])).reshape(-1, 1)   # Realizar la predicción y desescalar los resultados
    predictions.append(y_pred)
    plot_prediction(y_test_seq, y_pred, 500)            # Graficar la predicción y los valores reales
    calculate_mse(y_test_seq, y_pred, 500)              # Calcular el error cuadrático medio
```

```python
for i, pred in enumerate(predictions):
    print(f"Forma de la predicción {i+1}: {pred.shape}")

# Asegurar que todas las predicciones tengan la misma longitud
min_length = min(len(pred) for pred in predictions)
predictions = [pred[:min_length] for pred in predictions]
print ('Tamano data test',len(data_test))
# Sumar las predicciones
total_predictions = np.sum(predictions, axis=0)
print ('total_predictions length:',len(total_predictions))
# Obtener el desfase más grande entre todos los parámetros
start_point = max(param['LONG_SEC'] for param in params)
print('start_point:',start_point)

# Considerar el desfase en las predicciones totales y en los valores originales
original_values = data_test['Original'].values[start_point:len(total_predictions)+start_point]
print('original_values length',original_values.shape)

# Convertir las predicciones individuales en un DataFrame
df_predictions = pd.DataFrame([pred.flatten() for pred in predictions]).T

# Agregar las predicciones totales y los valores originales al DataFrame
df_predictions['Total Predictions'] = total_predictions
df_predictions['Original Values'] = original_values

# Nombrar las columnas
df_predictions.columns = ['Prediction 1', 'Prediction 2', 'Prediction 3' ,'Total Predictions', 'Original Values']

# Mostrar el DataFrame
print(df_predictions)

# Graficar y calcular el error cuadrático medio
plot_prediction(original_values, total_predictions, 1440)
calculate_mse(original_values, total_predictions, 1440)
# Calcular el error cuadrático medio global
global_mse = mean_squared_error(original_values, total_predictions)

# Crear el rango de fechas
dates_range = data_test.index[start_point:len(total_predictions)+start_point]

# Asegurarse de que las predicciones y el rango de fechas tengan la misma longitud
assert len(dates_range) == len(total_predictions), "Las longitudes no coinciden"

# Convertir las predicciones individuales en un DataFrame
df_predictions = pd.DataFrame([pred.flatten() for pred in predictions]).T
```

```python
# Agregar las predicciones totales y los valores originales al DataFrame
df_predictions['Total Predictions'] = total_predictions
df_predictions['Original Values'] = original_values
df_predictions.index = dates_range  # Asegurarse de que las predicciones y los valores originales tengan el mismo índice

# Nombrar las columnas
df_predictions.columns = ['Prediction 1', 'Prediction 2', 'Prediction 3' ,'Total Predictions', 'Original Values']

# Mostrar el DataFrame
print(df_predictions)

df_predictions.describe()

# Asegurarse de que el índice es una fecha
df_predictions.index = pd.to_datetime(df_predictions.index)

# Calcula el error cuadrático
df_predictions['Squared Error'] = np.square(df_predictions['Total Predictions'] - df_predictions['Original Values'])
# Agrupa por mes y calcula el error cuadrado medio
mse_by_month = df_predictions['Squared Error'].resample('M').mean()

# Prepara los datos para el gráfico
mse_by_month = mse_by_month.reset_index()
mse_by_month['date'] = mse_by_month['date'].dt.strftime('%b-%y')

# Crea el gráfico
plt.figure(figsize=(10, 6))
plt.bar(mse_by_month['date'], mse_by_month['Squared Error'], width=0.6)

# Agrega etiquetas de error sobre las barras
for i, value in enumerate(mse_by_month['Squared Error']):
    plt.annotate(f"{value:.2f}", (mse_by_month['date'][i], value), ha='center', va='bottom')

# Configura el gráfico
plt.title('MSE por Mes')
plt.xlabel('Mes')
plt.ylabel('MSE')
plt.xticks(rotation=90)

# Muestra el gráfico
plt.tight_layout()
plt.show()
```

Anexo 4: Formato de pronóstico (condición actual).

| Fecha y hora | C.E. TALARA | | | | Estimación Wisuki | |
|---|---|---|---|---|---|---|
| | V (m/s) | Pot Unit (MW) | Pot (MW) | Factor de Energía | Hr | m/s |
| 27/03/2017 00:30 | | | | | 00:00 | |
| 27/03/2017 01:00 | | | | | 01:00 | |
| 27/03/2017 01:30 | | | | | 02:00 | |
| 27/03/2017 02:00 | | | | | 03:00 | |
| 27/03/2017 02:30 | | | | | 04:00 | |
| 27/03/2017 03:00 | | | | | 05:00 | |
| 27/03/2017 03:30 | | | | | 06:00 | |
| 27/03/2017 04:00 | | | | | 07:00 | |
| 27/03/2017 04:30 | | | | | 08:00 | |
| 27/03/2017 05:00 | | | | | 09:00 | |
| 27/03/2017 05:30 | | | | | 10:00 | |
| 27/03/2017 06:00 | | | | | 11:00 | |
| 27/03/2017 06:30 | | | | | 12:00 | |
| 27/03/2017 07:00 | | | | | 13:00 | |
| 27/03/2017 07:30 | | | | | 14:00 | |
| 27/03/2017 08:00 | | | | | 15:00 | |
| 27/03/2017 08:30 | | | | | 16:00 | |
| 27/03/2017 09:00 | | | | | 17:00 | |
| 27/03/2017 09:30 | | | | | 18:00 | |
| 27/03/2017 10:00 | | | | | 19:00 | |
| 27/03/2017 10:30 | | | | | 20:00 | |
| 27/03/2017 11:00 | | | | | 21:00 | |
| 27/03/2017 11:30 | | | | | 22:00 | |
| 27/03/2017 12:00 | | | | | 23:00 | |
| 27/03/2017 12:30 | | | | | 00:00 | |
| 27/03/2017 13:00 | | | | | | |
| 27/03/2017 13:30 | | | | | | |
| 27/03/2017 14:00 | | | | | | |
| 27/03/2017 14:30 | | | | | | |
| 27/03/2017 15:00 | | | | | | |
| 27/03/2017 15:30 | | | | | | |
| 27/03/2017 16:00 | | | | | | |
| 27/03/2017 16:30 | | | | | | |
| 27/03/2017 17:00 | | | | | | |
| 27/03/2017 17:30 | | | | | | |
| 27/03/2017 18:00 | | | | | | |
| 27/03/2017 18:30 | | | | | | |
| 27/03/2017 19:00 | | | | | | |
| 27/03/2017 19:30 | | | | | | |
| 27/03/2017 20:00 | | | | | | |
| 27/03/2017 20:30 | | | | | | |
| 27/03/2017 21:00 | | | | | | |
| 27/03/2017 21:30 | | | | | | |
| 27/03/2017 22:00 | | | | | | |
| 27/03/2017 22:30 | | | | | | |
| 27/03/2017 23:00 | | | | | | |
| 27/03/2017 23:30 | | | | | | |
| 28/03/2017 00:00 | | | | | | |

Anexo 5: Formato de reporte de generación

| Hora | Pro. Diaria (MW) | P. Activa (MW) | P. Reactiva (Mbar) | Factor de Potencia | Velo. Viento (m/s) | Disponibilidad de WTG's |
|---|---|---|---|---|---|---|
| 00:30 | | | | | | |
| 01:00 | | | | | | |
| 01:30 | | | | | | |
| 02:00 | | | | | | |
| 02:30 | | | | | | |
| 03:00 | | | | | | |
| 03:30 | | | | | | |
| 04:00 | | | | | | |
| 04:30 | | | | | | |
| 05:00 | | | | | | |
| 05:30 | | | | | | |
| 06:00 | | | | | | |
| 06:30 | | | | | | |
| 07:00 | | | | | | |
| 07:30 | | | | | | |
| 08:00 | | | | | | |
| 08:30 | | | | | | |
| 09:00 | | | | | | |
| 09:30 | | | | | | |
| 10:00 | | | | | | |
| 10:30 | | | | | | |
| 11:00 | | | | | | |
| 11:30 | | | | | | |
| 12:00 | | | | | | |
| 12:30 | | | | | | |
| 13:00 | | | | | | |
| 13:30 | | | | | | |
| 14:00 | | | | | | |
| 14:30 | | | | | | |
| 15:00 | | | | | | |
| 15:30 | | | | | | |
| 16:00 | | | | | | |
| 16:30 | | | | | | |
| 17:00 | | | | | | |
| 17:30 | | | | | | |
| 18:00 | | | | | | |
| 18:30 | | | | | | |
| 19:00 | | | | | | |
| 19:30 | | | | | | |
| 20:00 | | | | | | |
| 20:30 | | | | | | |
| 21:00 | | | | | | |
| 21:30 | | | | | | |
| 22:00 | | | | | | |
| 22:30 | | | | | | |
| 23:00 | | | | | | |
| 23:30 | | | | | | |
| 24:00 | | | | | | |