Universidad Nacional de Loja

Facultad de la Energía, las Industrias y los Recursos Naturales No Renovables

Carrera de Electromecánica

Modelo de predicción de irradiación solar basado en Machine Learning para la Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables de la Universidad Nacional de Loja.

Trabajo de Integración Curricular, previo a la obtención del título de Ingeniero Electromecánico

AUTOR:

Jhandry Juan Tapia Sarango

DIRECTOR:

Ing. Edwin Bladimir Paccha Herrera, PhD.

Loja-Ecuador

2025

Certificación

Universidad Nacional de Loja

1859

Sistema de Información Académico Administrativo y Financiero - SIAAF

CERTIFICADO DE CULMINACIÓN Y APROBACIÓN DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Yo, **Paccha Herrera Edwin Bladimir**, director del Trabajo de Integración Curricular denominado **MODELO DE PREDICCIÓN DE IRRADIACIÓN SOLAR BASADO EN MACHINE LEARNING PARA LA FACULTAD DE LA ENERGÍA, LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES DE LA UNIVERSIDAD NACIONAL DE LOJA**, perteneciente al estudiante **JHANDRY JUAN TAPIA SARANGO**, con cédula de identidad N° **1104983604**.

Certifico:

Que luego de haber dirigido el **Trabajo de Integración Curricular**, habiendo realizado una revisión exhaustiva para prevenir y eliminar cualquier forma de plagio, garantizando la debida honestidad académica, se encuentra concluido, aprobado y está en condiciones para ser presentado ante las instancias correspondientes.

Es lo que puedo certificar en honor a la verdad, a fin de que, de así considerarlo pertinente, el/la señor/a docente de la asignatura de **Integración Curricular**, proceda al registro del mismo en el Sistema de Gestión Académico como parte de los requisitos de acreditación de la Unidad de Integración Curricular del mencionado estudiante.

Loja, 4 de Marzo de 2024

FIRMADO ELECTRÓNICAMENTE POR: EDWIN BLADIMIR PACCHA HERRERA

F) ----------

DIRECTOR DE TRABAJO DE INTEGRACIÓN CURRICULAR

Certificado TIC/TT.: UNL-2024-000697

1/1

Educamos para **Transformar**

Autoría

Yo, **Jhandry Juan Tapia Sarango**, declaro ser autor del presente Trabajo de integración Curricular y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos, de posibles reclamos y acciones legales, por el contenido del mismo. Adicionalmente, acepto y autorizo a la Universidad Nacional de Loja, la publicación de mi Trabajo de Integración Curricular en el Repositorio Institucional - Biblioteca Virtual.

Firma:

Cédula de identidad: 1104983604.

Fecha: 22 de abril de 2025

Correo electrónico: jhandry.tapia@unl.edu.ec.

Teléfono: 0993069043.

Carta de autorización por parte del autor, para consulta, reproducción parcial o total y/o publicación electrónica del texto completo, del Trabajo de Integración Curricular o de Titulación

Yo, **Jhandry Juan Tapia Sarango**, declaro ser autor del Trabajo de Integración Curricular denominado: **"Modelo de predicción de irradiación solar basado en Machine Learning para la Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables de la Universidad Nacional de Loja"**, como requisito para optar el título de **Ingeniero Electromecánico**, autorizo al Sistema Bibliotecario de la Universidad Nacional de Loja para que, con fines académicos, muestre la producción intelectual de la Universidad, a través de la visibilidad de su contenido en el Repositorio Institucional.

Los usuarios pueden consultar los contenidos de este trabajo en el Repositorio Institucional, en las redes de información del país y del exterior con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del Trabajo de Integración Curricular que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, a los veintidós días del mes de abril de dos mil veinticinco.

Firma:
Autor: Jhandry Juan Tapia Sarango.
Cédula de identidad: 1104983604.
Dirección: FEIRNNR.
Correo electrónico: jhandry.tapia@unl.edu.ec
Teléfono: 0993069043.

DATOS COMPLEMENTARIOS
Director del Trabajo de Integración Curricular: Ing. Edwin Bladimir Paccha Herrera, PhD.

Dedicatoria

El presente proyecto de tesis va dedicado a mi madre María Cruz Sarango Tapia por haber luchado incansablemente por mí, cuando mi padre se hizo a un lado evitando todo tipo de responsabilidad, por inculcarme los valores de responsabilidad, puntualidad, junto a los mismos su afán incansable de apoyarme para la finalización de mis estudios, también gracias por enseñarme que con esfuerzo y dedicación todo es posible. De igual manera a mi tío Antonio José Sarango Tapia, que me guio muchas veces haciendo el papel de padre para mí, enseñándome el valor del trabajo y que para conseguir algo en la vida con perseverancia y paciencia se lo puede conseguir, gracias por siempre mostrarme su apoyo incondicional. A mis abuelitos Segundo Juan y María Victoria, quienes fueron los que me formaron como persona, nada de esto sería posible sin su guía para mi vida, gracias por dar tanto por mí y perdón por corresponderles con tan poco. También a mi familia, tíos, primos y demás familiares que me apoyaron moralmente para seguir con mis estudios y culminar mi carrera universitaria.

Por último y por ello no menos importante para ti Titha que fuiste el pilar fundamental en esta última etapa de mi estudio, la única persona que estuvo ahí para mi en todos los sentidos, la mujer que me hizo sentir que en verdad podía realizar todo lo que me proponía, este trabajo también es por ti y para ti.

Jhandry Juan Tapia Sarango

Agradecimiento

Mis sinceros agradecimientos a la Universidad Nacional de Loja (UNL), a la Facultad de la Energía, los Recursos Naturales no Renovables por acogerme y formarme como un profesional con estándares morales y éticos altos, de igual manera a toda la planta docente que siempre estuvieron con la intención de ayudarme y guiarme de la mejor manera para responder a todas mis preguntas y dar solución a los inconvenientes que se presentaron durante todo el transcurso de la carrera, pero de manera especial al Ing. Edwin Paccha que durante todo el tiempo compartió de sus conocimientos para el desarrollo del trabajo. De igual manera, debo agradecer al Centro de Investigaciones Tecnológicas y Energéticas (CITE) de la UNL, por brindarme la oportunidad de llevar a cabo el presente proyecto, donde logre obtener los conocimientos necesarios que todo ingeniero debe poseer, los mismos que han contribuido con mi formación profesional.

Jhandry Juan Tapia Sarango

Índice de Contenidos

Índice de Tablas:

Índice de Figuras:

Índice de Anexos:

Simbología

Acrónimos

Bi-LSTM	Red bidireccional de memoria a corto plazo
CITE	Centro de Investigaciones Tecnológicas y Energéticas
CNN	Red Neuronal Convolucional
DL	Deep Learning
FEIRNNR	Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables
GHI	Irradiación Horizontal Global
GRU	Gated Recurrent Unit
IA	Inteligencia artificial
IS	Irradiación Solar
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
ML	Machine Learning
MLP	Modelo de Perceptrón Multicapa
MSE	Mean Squared Error
R2	R-squared
RNN	Redes Neuronales Recurrentes
S2SAE	Autocodificador secuencia-secuencia apilado
SGM	Generación de tiempo solar

1. Título

Modelo de predicción de irradiación solar basado en Machine Learning para la Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables de la Universidad Nacional de Loja.

2. Resumen

La predicción de datos o forecasting, se emplea para anticipar la disponibilidad de energía solar en una región específica. En este trabajo, se llevó a cabo la predicción de irradiación solar en la Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables (FEIRNNR) mediante la implementación de Redes Neuronales, tales como Redes Neuronales Recurrentes (RNN), Long Short-Term Memory (LSTM) y Gated Recurrent Unit (GRU). Durante este proceso, se realizaron ajustes en hiperparámetros clave, como Dropout, el optimizador Adams, batch_size y las épocas. Los modelos ajustados, con cada uno de los hiperparámetros mencionados anteriormente, lograron una precisión del 85 % en todos los análisis realizados, lo cual es considerado aceptable en este ámbito. Se utilizó información proveniente de la estación meteorológica de la FEIRNNR, abarcando todo el año 2021 con una frecuencia de muestreo de cinco minutos. No obstante, para explorar posibles mejoras, se optó por calcular promedios cada 20 y 60 minutos respectivamente. Dada la diversidad climática de la ciudad de Loja, se consideró que esta estrategia proporcionaría resultados más precisos, resultando en un éxito particularmente notable para el periodo de 5 minutos, con mejoras mínimas con respecto a los resultados obtenidos con intervalos de cinco minutos. Este estudio se sometió a la evaluación de métricas como Mean Squared Error (MSE), Mean Absolute Error (MAE) y R-squared (R2). En última instancia, este trabajo contribuye al impulso de la utilización de fuentes de energía limpias y renovables. Para futuras mejoras, se sugiere explorar ajustes en las estructuras de los modelos, así como realizar modificaciones en los hiperparámetros para continuar optimizando la precisión y eficacia de las predicciones.

Palabras claves: Machine Learning, Deep Learning, Redes Neuronales Recurrentes, Long Short-Term Memory, Gated Recurrent Unit.

2.1. Abstract

Forecasting data is used to anticipate the availability of solar energy in a specific region. In this paper, solar irradiance forecasting was executed at the Faculty of Energy, Industries and Non-Renewable Natural Resources (FEIRNNR) by implementing Neural Networks, such as Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). During this process, adjustments were made to key hyperparameters, such as Dropout, the Adams optimiser, batch size and epochs. The adjusted models, with each of the hyperparameters mentioned above, achieved an accuracy of 85 % in all the analyses performed, which is considered acceptable in this field. Data from the FEIRNNR weather station were used, covering the entire year 2021 with a sampling frequency of five minutes. However, to explore possible improvements, it was decided to calculate averages every 20 and 60 minutes respectively. Given the climatic diversity of the city of Loja, it was considered that this strategy would provide more accurate results, resulting in particularly notable success for the 5-minute period, with minimal improvements over the results obtained with five-minute intervals. This study was subjected to the evaluation of metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE) and R-squared (R2). Ultimately, this work contributes to the promotion of the use of clean and renewable energy sources. For future improvements, it is suggested to explore adjustments to model structures, as well as modifications to hyperparameters to further optimise the accuracy and efficiency of predictions.

Keywords: Machine Learning, Deep Learning, Recurrent Neural Networks, Long Short-Term Memory, Gated Recurrent Unit..

3. Introducción

El aprendizaje automático (Machine Learning) constituye la base técnica de la minería de datos. Se utiliza para extraer información de los datos brutos de las bases de datos, información que se expresa de forma comprensible y puede utilizarse para diversos fines. Algunas aplicaciones de minería de datos se centran en la predicción: pronosticar lo que ocurrirá en situaciones nuevas a partir de datos que describen lo que ocurrió en el pasado, a menudo adivinando la clasificación de los nuevos eventos. El aprendizaje es adquirir conocimiento a través de experiencias, enseñanzas e investigaciones, y machine learning busca adquirir este conocimiento a través de patrones de manera automática, explicando datos y poder hacer predicciones de ello (Witten & Frank, 2005).

La principal barrera que enfrenta el desarrollo de la energía solar reside en la intermitencia de la irradiancia solar, lo cual genera una variabilidad significativa en la disponibilidad de energía tanto en el espacio como en el tiempo. Este desafío es compartido con la energía eólica, dificultando su implementación a gran escala, a menos que se pueda prever con alta precisión el recurso disponible en las próximas horas y días (Instituto de Ingeniería del conocimiento, 2023).

Para enfrentar este problema, los parques fotovoltaicos, necesitan poseer modelos de predicción que les permitan determinar la posible producción de energía eléctrica con una anticipación cercana a dos días. Diversos grupos de investigación se encuentran actualmente involucrados en el desarrollo de herramientas de este tipo. En los primeros estudios publicados, los modelos se centran en la predicción de los valores de irradiancia solar, y a partir de dicha predicción obtienen la producción de energía eléctrica (Rodríguez, 2021).

La presente investigación es importante, porque hará uso de la Inteligencia Artificial (Machine Learning) para la predicción de la (IS) en la FEIRNNR, y con ello, estimar la producción de energía eléctrica del sistema fotovoltaico instalado en la terraza del bloque A3. El modelo para la predicción de la Irradiancia Solar (IS) con herramientas de Machine Learning coadyuvará al desarrollo de proyectos de sostenibilidad energética en la Universidad Nacional de Loja.

En el campo de la predicción de irradiación solar, (Nawab et al., 2023), desarrollaron modelos de predicción de irradiación solar, dos modelos de memoria a corto plazo (LSTM) y de RNA, la LSTM superó a RNA. Los modelos se evaluaron mediante el coeficiente de determinación (R2), el error cuadrático medio (MSE), el error absoluto medio (MAE), de 0,93, 0,008, 0,17, respectivamente.

Otro caso para la predicción de irradiación solar (Gallo et al., 2025), haciendo uso de las redes neuronales desarrollo un modelo de Perceptrón Multicapa (MLP) para

estimar la irradiación solar terrestre, en términos de Irradiación Horizontal Global (GHI). Al concluir los valores de GHI estimados se comparan con los datos de GHI medidos en tierra, alcanzando un coeficiente de determinación (R2) de 0,929.

Para otro modelo de predicción para la irradiación solar (Mughees et al., 2023), usando la inteligencia artificial, desarrollado una novedosa «red bidireccional de memoria a corto plazo» (Bi-LSTM) basada en un autocodificador secuencia-secuencia apilado (S2SAE). Para predecir la irradiación solar a corto plazo los resultados simulados a partir de datos reales confirmaron que el modelo propuesto superaba los resultados propuestos alcanzando un R2 de 0.997, lo que evidencia la alta fiabilidad del modelo propuesto.

Finalmente, en relación con la predicción de la irradiación solar (Zhao et al., 2024), desarrolló un conjuntos de árboles/regresión de bosque aleatorio (RF), haciendo uso de redes neuronales. Los modelos se evaluaron mediante el R2, el (MSE), el (MAE), de 0,99, 40,083, 5,575, respectivamente.

En el campo de la energía eólica también son muy utilizados los algoritmos de machine learning. Por ejemplo, (Maldonado et al., 2021), desarrollaron un modelo híbrido conformado por los modelos de Perceptrón Multicapa (MLP), Memoria Larga a Corto Plazo (LSTM) y Redes Neuronales, Red Neuronal Convolucional (CNN), este modelo es el que mejor se adapta a la producción de energía del parque Eólico Villonaco.

Objetivo general

- Diseñar un modelo de predicción de irradiación solar basado en Machine Learning para la FEIRNNR.

Objetivos específicos

- Procesar estadísticamente los datos de irradiación solar de fuentes físicas cercanas a la FEIRNNR.
- Desarrollar un modelo basado en Machine Learning para la predicción de irradiación solar.
- Evaluar el desempeño del modelo de predicción de irradiación utilizando métricas usualmente empleadas en Machine Learning.

4. Marco Teórico

4.1. Capítulo I: Fundamentos Teóricos de los datos y de la Irradiancia Solar

4.1.1. Análisis de Datos

Resulta importante tener una buena comprensión de los datos de entrada y la variada terminología utilizada al describir los datos. Los datos pueden estructurarse en filas y columnas, como una tabla de base de datos o como una hoja de cálculo, Estos son conocidos como "Estructura tradicional de datos", y son comunes en el campo del aprendizaje de máquina. A continuación, se mencionan algunos conceptos importantes.

- **Observación.** Es la entidad más pequeña, con propiedades de interés para un estudio que puede ser registrado.
- **Características.** Son las propiedades o atributos de las observaciones que pueden ser útiles para el aprendizaje.
- **Conjunto de datos.** Una colección de observaciones es un conjunto de datos y, cuando se trabaja con métodos de aprendizaje máquina, generalmente se requieren algunos conjuntos de datos para diferentes propósitos.
- **Datos de entrenamiento**. Conforman un conjunto de datos que se incorpora al algoritmo de aprendizaje máquina para entrenar al modelo.
- **Datos de prueba.** Constituyen un conjunto de datos utilizado para validar la precisión del modelo, pero que no se emplea para entrenar al modelo. Se lo conoce también como "Conjunto de datos de validación". Para cumplir con los propósitos del análisis de datos y del modelado predictivo, resulta importante conocer el tipo de dato que se va a seleccionar, con el fin de ayudar a determinar el tipo de visualización, análisis de datos o modelo estadístico.
- **Continuo.** Lo constituyen los datos que pueden tomar cualquier valor dentro de un intervalo. Entre algunos ejemplos se encuentran la velocidad del viento, la distancia recorrida por un coche o la estatura de una persona.
- **Discreto.** Este tipo de datos puede tomar solo valores enteros; por ejemplo, el conteo de recurrencia de un evento o el número de clics de un sitio web.
- **Categórico.** Son datos que pueden tomar solo un conjunto específico de valores, los cuales representan un conjunto de categorías posibles, tales como el tipo de sangre o los estados de un país, entre otros.
- **Binario.** Los datos binarios suponen un caso especial dentro de los datos categóricos. Estos cuentan con solo dos categorías de valores: verdadero o falso; entre algunos ejemplos, enfermo/saludable, día/noche o activo/inactivo.
- **Ordinal.** Son datos categóricos que tienen un orden explicito; por ejemplo, la falla de ropa o la calificación numérica de un producto (1.°,2.°,3.°,4.°,5.°).

- **Recopilación de datos.** Este paso implica reunir el material de aprendizaje que implementará un algoritmo para generar un conocimiento procesable. En la mayoría de los casos, los datos deberán combinarse en una sola fuente, como un archivo de texto, hoja de cálculo o base de datos.
- **Exploración y preparación de datos.** La calidad de cualquier proyecto de aprendizaje automático se basa, en gran medida, en la calidad de sus datos de entrada. Para mejorar la calidad de los datos, estos requieren de una preparación especial para el proceso de aprendizaje. Esta preparación implica arreglar o limpiar los llamados datos" desordenados", eliminando datos innecesarios, y recopilando los datos para ajustarse a las entradas esperadas de la técnica de aprendizaje máquina (Jiménez, 2021).

4.1.2. Data Mining

Según Molina (2001) menciona que la Data Mining se refiere al proceso de extraer conocimiento de bases de datos. Su objetivo es descubrir situaciones anómalas y/o interesantes, tendencias, padrones y secuencias en los datos.

La Data Mining es una etapa dentro del proceso completo del descubrimiento del conocimiento, este intenta obtener patrones o modelos a partir de los datos recopilados. Decidir si los modelos obtenidos son útiles o no suele requerir una valoración subjetiva va por parte del usuario. Los algoritmos de Data Mining suelen tener tres componentes:

1. El modelo, que contiene parámetros que han de fijarse a partir de los datos de entrada.
2. El criterio de preferencia, que sirve para comparar modelos alternativos.
3. El algoritmo de búsqueda, que viene a ser como cualquier otro programa de inteligencia artificial (IA).

4.1.2.1 Herramientas de Data Mining. Las herramientas de data mining empleados en el proceso de KDD se pueden clasificar en dos grandes grupos:

- **Técnicas de verificación**, en las que el sistema se limita a comprobar hipótesis suministradas por el usuario.
- **Métodos de descubrimiento,** en los que se han de encontrar patrones potencialmente interesantes de forma automática, incluyendo en este grupo todas las técnicas de predicción. El resultado obtenido con la aplicación de algoritmos de data mining pertenecientes al segundo grupo, el de técnicas de descubrimiento, pueden ser de carácter descriptivo o predictivo. Las predicciones sirven para prever el comportamiento futuro de algún tipo de entidad, mientras que una descripción puede ayudar

a su comprensión.

4.1.2.2 Resumen. Se obtienen representaciones compactas para subconjuntos de los datos de entrada (análisis interactivo de datos, generación automática de informes, visualización de datos).

4.1.2.3 Modelado de Dependencias. Se obtienen descripciones de dependencias existentes entre variables. El análisis de relaciones (por ejemplo, las reglas de asociación), en el que se determinan relaciones existentes entre elementos de una base de datos, podría considerarse un caso particular de modelado de dependencias.

4.1.2.4 Análisis de Secuencias. Se intenta modelar la evolución temporal de alguna variable, con fines descriptivos o predictivos (redes neuronales multicapas (Violeta, 2004) .

4.1.3. Análisis de datos mediante gráficos

El lenguaje gráfico tiene un papel esencial en la organización, descripción y análisis de datos, al ser un instrumento de transnumeración. Esta es una de las formas básicas de razonamiento estadístico definidas por Wild y Pfannkuch (1999), que consiste en obtener una nueva información, al cambiar de un sistema de representación a otro. Por ejemplo, al pasar de un listado de datos a un histograma, el alumno puede percibir el valor de la moda, que antes no era visible en los datos brutos.

La construcción e interpretación de gráficos estadísticos es también parte importante de la cultura estadística a la que cada vez se dedica más atención y que se define como la unión de dos competencias relacionadas:

Interpretar y evaluar críticamente la información estadística, los argumentos apoyados en datos o los fenómenos estocásticos que las personas pueden encontrar en diversos contextos, incluyendo los medios de comunicación, pero no limitándose a ellos, y b) discutir o comunicar sus opiniones respecto a tales informaciones estadísticas cuando sea relevante.

4.1.3.1 Elementos y competencias en la lectura de gráficos estadísticos. Encontramos gráficos en la prensa diaria, en Internet y también en textos de materias como las ciencias sociales. Sería, por tanto, necesario que una persona culta fuese capaz de comprender la información expresada en los mismos, aunque esta competencia no es sencilla. Cuando se pide a un estudiante interpretar un gráfico, el estudiante debe realizar la traducción entre lo representado en el gráfico y la realidad. Pero esta traducción requiere conocimientos tanto sobre la realidad representada, como sobre los convenios de construc-

ción del gráfico que a veces el estudiante no posee. Un gráfico queda determinado por los siguientes elementos.

Las **palabras** que aparecen en el gráfico, como el título del gráfico, las etiquetas de los ejes y de las escalas, y que proporcionan las claves necesarias para comprender el contexto, las variables y las relaciones expresadas en el gráfico.

El **contenido matemático** subyacente en el gráfico. Por ejemplo, los conjuntos numéricos empleados y otros conceptos matemáticos implícitos en el gráfico que el estudiante ha de dominar para interpretarlo, como los de área en un gráfico de sectores, longitud en un gráfico de líneas o sistema de coordenadas cartesianas en un diagrama de dispersión.

Partiendo del análisis anterior, se identifican los siguientes elementos estructurales de un gráfico estadístico:

El **título** y las **etiquetas** indican el contenido contextual del gráfico y cuáles son las variables en él representadas. Será importante incluir un título y etiquetas no ambiguos.

El **marco** del gráfico, que incluye los ejes, escalas, y marcas de referencia en cada eje. Dicho marco proporciona información sobre las unidades de medida de 23 las magnitudes representadas. Puede haber diferentes tipos de marcos y sistemas de coordenadas (lineales, cartesianas, bidimensionales o multidimensionales, polares).

Los **especificadores** del gráfico son los elementos usados para representar los datos, como los rectángulos (en el histograma) o los puntos (en el diagrama de dispersión). Los autores nos alertan de que no todos los especificadores son igualmente sencillos de comprender, sugiriendo el siguiente orden de dificultad: Posición en una escala homogénea (gráficos de línea, de barras, de puntos, algunos pictogramas e histogramas); posición en una escala no homogénea (gráficos polares, gráficos bivariantes); longitud (gráficos poligonales o estrellados sin ejes de referencia, árboles), ángulo o pendiente (gráfico de sectores, discos), área (círculos, pictogramas), volumen (cubos, algunos mapas estadísticos), color (mapas estadísticos codificados mediante color).

En relación con los anteriores componentes del gráfico, su lectura y construcción, se requieren los siguientes tipos de competencias relacionadas con el lenguaje de los gráficos:

Reconocer los elementos estructurales del gráfico (ejes, escalas, etiquetas, elementos específicos) y sus relaciones. Esta competencia se adquiere cuando es posible distinguir cada uno de estos elementos y si cada elemento es o no apropiado en el gráfico particular.

Apreciar el impacto de cada uno de estos componentes sobre la presentación de la información en un gráfico, (por ejemplo, ser capaz de predecir como cambiaría el gráfico al variar la escala de un eje).

Traducir las relaciones reflejadas en el gráfico a los datos que se representan en el mismo y viceversa. Por ejemplo, cuando un diagrama de dispersión es creciente, comprender que la relación representada entre las dos variables es directa.

Reconocer cuando un gráfico es más útil que otro, en función del juicio requerido y de los datos representados, es decir, saber elegir el gráfico adecuado al tipo de variable y al tipo de problema (Artega, 2009).

4.1.4. Irradiación solar

La irradiación solar es la magnitud que mide la energía por unidad de superficie de radiación solar incidente en una superficie. Es decir, la potencia recibida durante un tiempo (J/m^2 o Wh/m^2).

El término de radiación solar es un concepto genérico, pero que no se cuantifica con ninguna magnitud. Las magnitudes que describen la radiación solar que llega a la superficie terrestre por metro cuadrado es la irradiancia y la irradiación solar.

4.1.4.1 Origen de la irradiancia solar El origen de la **radiación electromagnética** que llega a **la Tierra es el sol**. La radiación solar es la reacción de fusión nuclear que se produce en el interior del sol en la que se libera una gran cantidad de energía térmica.

La energía generada provoca que el Sol sea una gigantesca masa incandescente. La temperatura en su capa es aproximadamente de 5.505 grados Celsius.

Al ser un cuerpo incandescente, el Sol emite radiación electromagnética en un amplio rango de longitudes de onda.

4.1.4.2 Tipos de irradiancias solares. Hay varios tipos de irradiancia solar.

- **La irradiancia solar total**. La irradiación solar total es una medida de la potencia solar en todas las longitudes de onda por unidad de área incidente en la atmosfera superior de la Tierra. Se mide la perpendicular a la luz solar entrante.
- **La irradiancia normal directa**. La irradiación normal directa, o radiación de haz, se mide en la superficie de la Tierra en una ubicación dada con un elemento de superficie perpendicular al Sol. Este tipo de irradiación excluye la radiación solar difusa. La irradiancia directa es igual a la irradiancia en el exterior de la atmosfera terrestre, menos las pérdidas atmosféricas debidas a la absorción y dispersión. Las

pérdidas dependen principalmente de: La hora del día (longitud de la trayectoria de la luz a través de la atmósfera según el ángulo de elevación solar)

- La cantidad de nubes en el cielo
- La humedad.
- Otros aspectos como: aerosoles, moléculas de ozono, vapor de agua, etc.

La irradiancia sobre la atmósfera también varía con la época del año (debido a que la distancia al sol varía), aunque este efecto es generalmente menos significativo en comparación con el efecto de las pérdidas en la radiación directa.

La irradiancia directa normal que proviene del Sol es de 1367 W/m^2 a la superficie de la atmósfera. Esta magnitud se llama constante solar.

4.1.4.3 La irradiancia horizontal difusa o del cielo difuso. La irradiación horizontal difusa o la radiación de cielo difuso es la radiación en la superficie de la Tierra a partir de la luz dispersada por la atmósfera. Se mide en una superficie horizontal con radiación proveniente de todos los puntos del cielo, excluyendo la radiación solar proveniente del disco solar.

4.1.4.4 La irradiancia horizontal global. La radiación global incluye la recibida directamente del disco solar y también la radiación celeste difusa dispersa al atravesar la atmósfera. La irradiancia horizontal global es la irradiancia total del sol sobre una superficie horizontal en la Tierra. Es la suma de la irradiancia directa (después de tener en cuenta el ángulo cenital solar del Sol) y la irradiancia horizontal difusa (Planas, 2019) .

4.2. Capítulo II: Machine Learning y Deep Learning

Es una rama de la Inteligencia Artificial que se encarga de crear algoritmos que poseen la capacidad de aprender y no tener que programarlos de forma explícita. El desarrollador no tendrá que sentarse a programar por horas tomando en cuenta todos los escenarios posibles ni todas las excepciones posibles. Lo único que hay que hacer es alimentar el algoritmo con un volumen gigantesco de datos para que el algoritmo aprenda y sepa qué hacer en cada uno de estos casos. En la **Figura 1** se puede observar el desempeño del Machine Learning (ML) y Deep Learning (DL) en distintos campos de aplicación.

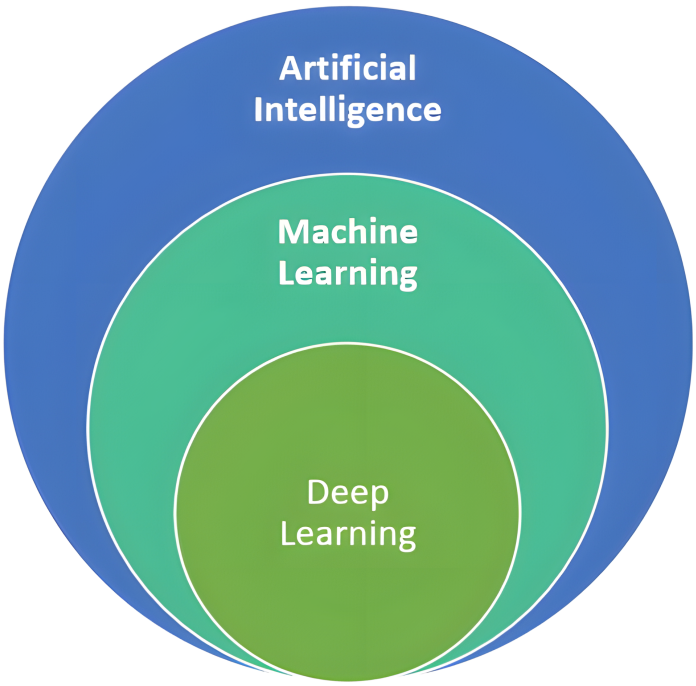

Figura 1. Machine Learning and Deep Learning.
Fuente: (Sandoval, 2018).

4.2.1. Conceptos Básicos de Python

4.2.1.1 Instrucciones y Funciones. Un programa Python está formado por instrucciones (a veces también son llamadas sentencias). Cada instrucción se escribe normalmente en una línea.

Si en un programa hay diversas instrucciones, estas se ejecutan secuencialmente, una tras otra. Una función puede ser llamada (también se dice invocada) desde un programa escribiendo su nombre y a continuación unos paréntesis de comienzo y cierre. En ocasiones, las funciones toman parámetros, que se escriben en la llamada dentro de los paréntesis, separados por comas si hay más de uno. Hay muchas funciones predefinidas, por ejemplo, la función help nos provee ayuda sobre cualquier otra función.

Por supuesto, la ayuda está en inglés. En el ejemplo anterior hemos usado el carácter almohadilla (#) para escribir un comentario sobre el código. Los comentarios se usan en los programas para hacer aclaraciones, explicar el funcionamiento y, en general, hacer más cómodo de entender nuestros programas. Las funciones anteriores se llaman funciones predefinidas porque ya vienen incluidas por defecto en Python.

4.2.1.2 Expresiones y Tipos. El uso de operadores es frecuente en todos los lenguajes de programación. Las expresiones están formadas por operadores (como el +), literales y variables. Una expresión es siempre evaluada por Python, para obtener un resultado,

antes de seguir ejecutando el resto de las instrucciones del programa.

En las expresiones lógicas se usan diferentes operadores lógicos (como el operador and) y operadores relacionales (como los operadores > o ==). Iremos viendo más operadores de estos tipos a medida que los vayamos requiriendo.

En las expresiones también podemos usar llamadas a funciones, siempre que sean funciones que devuelvan algún resultado. Para que una función devuelva un valor cuando sea invocada, utilizaremos la instrucción return en algún punto del cuerpo de la función (casi siempre, al final del cuerpo de la función).

Una función puede tomar diferentes parámetros. Definamos una función de nombre descuento que reciba dos parámetros: un valor y un porcentaje y devuelva el resultado de aplicar el descuento correspondiente al porcentaje sobre el valor indicado.

El resultado de una expresión podemos guardarlo para utilizarlo más adelante en nuestro programa. Para esto se usan las variables. Cada variable tiene un tipo.

4.2.1.3 Tipos Predefinidos. Las variables: tipo cadena de caracteres, tipo entero y tipo real. Cada uno de estos son tipos predefinidos en Python (built-in types). Hablamos de predefinidos porque Python también admite al programador crear sus propios tipos mediante clases. Los valores que hemos escrito para inicializar cada una de las variables se llaman literales.

Un tipo de datos está definido por un conjunto de posibles valores (lo que en matemáticas se conoce como dominio) y un conjunto de operaciones asociadas. Un literal (es decir, un valor determinado de un tipo) tiene asociado un tipo definitivo, dependiendo de cómo está escrito dicho literal. Por otra parte, para saber el tipo asociado a una variable, debemos fijarnos en el valor que ha sido almacenado en la misma.

Una función recibe uno o diversos valores de un tipo determinado, sus parámetros, y puede devolver un valor de este u otro tipo. Las llamadas a las funciones pueden estar representadas por un operador o por un identificador más unos parámetros reales, como veremos más adelante.

En las siguientes secciones se muestran distintos tipos predefinidos, la manera en que se escriben sus literales y sus operaciones asociadas más importantes.

4.2.1.4 Tipo lógico. El tipo lógico (bool) únicamente contiene dos valores en su dominio: verdadero (True) y falso (False). Estas dos palabras son precisamente los únicos literales lógicos permitidos en Python. El tipo lógico sirve para representar resultados de

expresiones lógicas.

4.2.1.5 Tipo Cadena. El tipo cadena de caracteres (str), o como se suele abreviar, tipo cadena, nos permite trabajar con cadenas de caracteres. Las literales cadenas de caracteres se escriben utilizando unas comillas simples o dobles para rodear al texto que queremos representar.

Una casilla no existe si su índice está fuera del rango permitido. Un índice i está en el rango permitido si $0 \leq$ i <len(texto). Python nos permite usar el operador + entre dos cadenas, y el operador * entre una cadena y un número entero. También es posible usar los operadores relacionales entre cadenas.

Las cadenas tienen los métodos: strip (), rstrip (), lstrip (). Estos métodos devuelven una nueva cadena, eliminando los espacios en blanco del principio y del final, solo del final o solo del principio respectivamente. Los espacios en blanco eliminados incluyen también a los tabuladores, $\backslash t$, y saltos de línea, $\backslash n$, return, $\backslash r$, entre otros. El método format de las cadenas devuelve una versión formateada de la misma al combinarla con algunos parámetros. Entre otras cosas, nos permite intercalar en una cadena los resultados de diversas expresiones, eligiendo el orden o el formato en que se representan dichos resultados. Esta flexibilidad hace de format una función adecuada, que podemos usar junto a print, para exponer mensajes más o menos complejos. El método format se aplica sobre una cadena con texto que tiene intercaladas parejas de llaves que pueden incluir un número. Devolverá una cadena en la que se sustituirán las llaves por los resultados de evaluar las expresiones que reciban como parámetros.

Python ofrece la posibilidad de usar f-string. Estas son cadenas de caracteres con el prefijo f o F. Estas cadenas pueden incluir pares de llaves con expresiones cuyo valor puede ser sustituido en tiempo de ejecución. Dentro de los pares de llaves también podemos especificar detalles del tipo de dato que se espera y la anchura del campo, entre otros (Bobadilla, 2021).

4.2.1.6 Tipos Numéricos. Existen tres tipos que permiten trabajar con números en Python: enteros (int), reales (float) y complejos (complex). También podemos usar el tipo Fraction. Las operaciones disponibles sobre valores de estos tipos incluyen a las operaciones aritméticas (suma, resta, multiplicación...), las operaciones relacionales (mayor que, menor que...), y algunas otras como el valor absoluto.

La operación inversa, convertir una cadena de caracteres en un valor de un tipo, se denomina parsing. Cada tipo viene dotado de una operación de este tipo. Para los tipos anteriores, las operaciones de parsing son los nombres de los tipos: int, float. Esta

operación de parsing la debemos hacer cuando leemos datos de un fichero o de la consola. La función bool aplicada a una cadena resulta en True si la cadena no está vacía y False en caso contrario.

Hay otros tipos que son usados en matemáticas como el tipo Fraction y el tipo Complex. El tipo de datos Fraction representa una fracción y Complex un número complejo. Los valores de Fraction y Complex pueden ser operados con los operadores aritméticos usuales: +, —, *, / ... Los valores de Fraction se pueden comparar con los operares, , >=. Los valores de complex no. Fraction tiene, entre otras, las propiedades numerador, denominador. Complex tiene las propiedades reales, imag que representan respectivamente la parte real y la imaginaria. El ángulo y el módulo pueden ser calculados mediante las funciones phase, abs.

4.2.1.7 Tipos para manejar fechas. Otros tipos de datos que usaremos a menudo son las fechas y las horas. Estos son los tipos datetime, date, time y timedelta. Los valores de datetime tienen entre otras las propiedades hour, month, year y se pueden comparar con los operadores relacionales. Mediante la función now podemos obtener la fecha y hora actual. Los valores del tipo datetime aceptan el operador + y - para sumar o restar días, etc. El tipo datetime representa fechas, horas, minutos, etc., date fechas y time horas, minutos y segundos. La conversión a cadenas de caracteres se hace con la función strptime (valor, formato) indicando el formato adecuado. El parsing de los valores de datetime se hace con el método strftime(formato) usando también el formato adecuado.

4.2.1.8 Clases y objetos. El tipo List ya viene predefinido en Python. Sus valores son objetos y estos objetos tienen métodos que se llaman con el operador, que aparta un objeto y un método. Si queremos definir un tipo nuevo debemos diseñar una clase. En esta se especifican las propiedades del objeto, sus métodos y el constructor. Internamente, en la definición de la clase, para acceder a las propiedades y métodos usamos self junto con la propiedad (Bobadilla, 2021).

4.2.2. Redes Neuronales Artificiales

Las redes artificiales de neuronas tratan, en cierto modo, de replicar el comportamiento del cerebro, donde tenemos millones de neuronas que se interconectan en red para enviarse mensajes unas a otras. Esta réplica del funcionamiento del cerebro humano es uno de los “modelos de moda” por las habilidades cognitivas de razonamiento que adquieren. El reconocimiento de imágenes o vídeos, por ejemplo, es un mecanismo complejo y una red neuronal es lo mejor para realizarlo. El problema, como ocurre con el cerebro humano, es que son lentas de entrenar y necesitan mucha capacidad de cómputo. Quizás sea uno de los modelos que más ha ganado con la “revolución de los datos” (Sandoval,

2018). En la **Figura 2**, se aprecia el funcionamiento de las redes Neuronales.

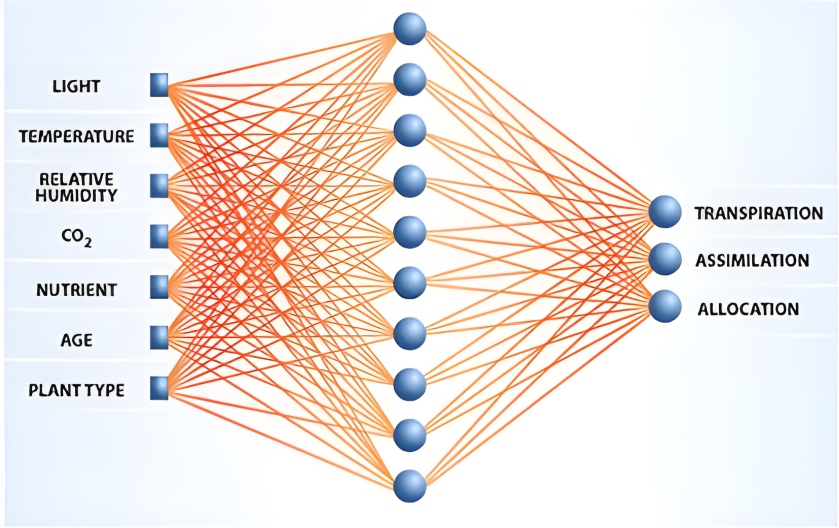

Figura 2. Gráfico de Redes Neuronales.
Fuente: (Sandoval, 2018).

4.2.3. Machine Learning con Python

4.2.3.1 Tipos de Machine Learning. Los conocimientos básicos incluyen la identificación de las tareas, empezando por la clasificación de los problemas de machine learning en alguno de los siguientes tipos:

≪ **Aprendizaje supervisado**

- Regresión
- Clasificación

≪ **Aprendizaje no supervisado**

- Clustering (agrupamiento)
- Reducción de dimensiones

≪ **Aprendizaje semi-supervisado**

≪ **Aprendizaje por refuerzo**

4.2.3.2 Aprendizaje supervisado. El aprendizaje supervisado en machine learning se aplica cuando cada dato, o conjunto de datos de entrada (muestra) tiene asociada una etiqueta. En la siguiente **Figura 3**, representa un gráfico tridimensional, podemos ver los datos correspondientes a un problema de regresión.

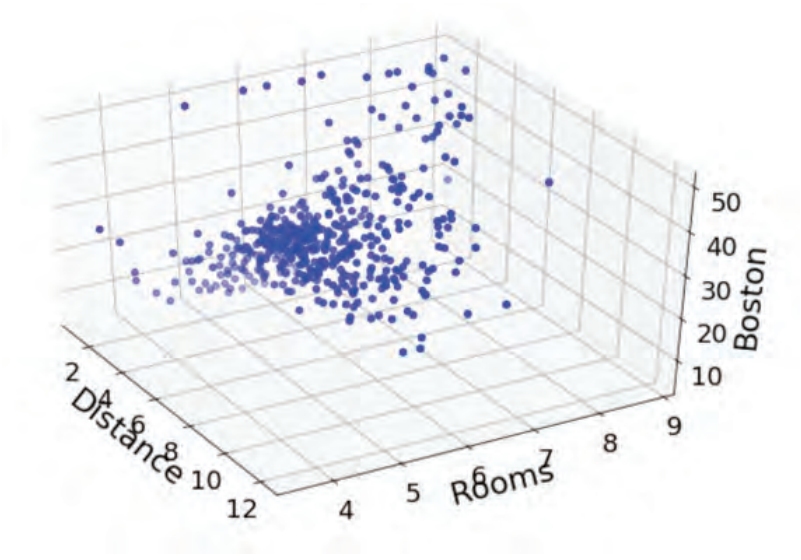

Figura 3. Gráfico Tridimensional de un Problema de Regresión.

Fuente: (Bobadilla, 2021).

4.2.3.3 Aprendizaje no supervisado. El aprendizaje no supervisado utiliza información no etiquetada. La aplicación más conocida del aprendizaje no supervisado es la de clustering (agrupamiento). El objetivo de la técnica de clustering es agrupar muestras: p, ej. En el siguiente **Figura 4**, muestra un esquema típico de clustering. Contiene tres clusters (grupos o clases) correspondientes a tres diferentes tipos de lirios.

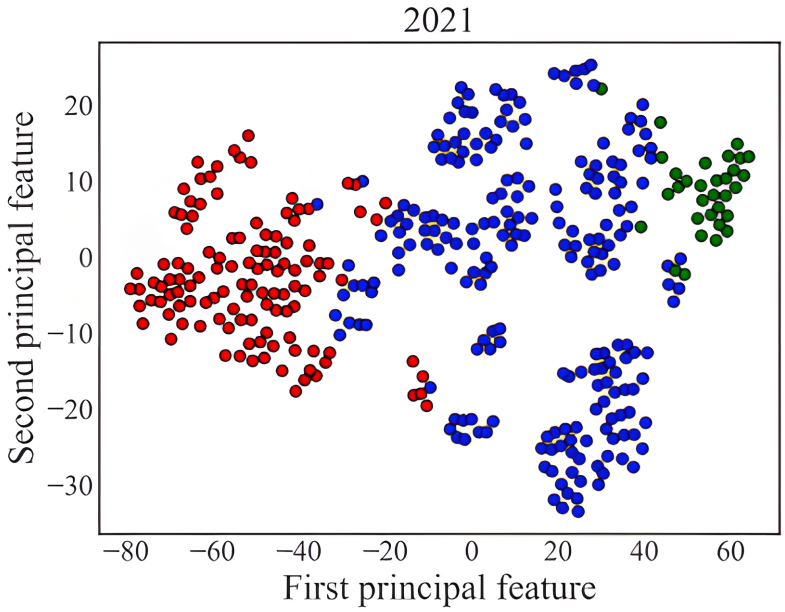

Figura 4. Esquema típico de clustering.

Fuente: (Bobadilla, 2021).

4.2.3.4 Aprendizaje semi-supervisado. El aprendizaje semi-supervisado trata con conjuntos de datos, en los que una porción de los datos está etiquetada y el resto no. Normalmente, la cantidad de muestras etiquetadas es mucho más pequeña que las no etiquetadas. La mayoría de los algoritmos de aprendizaje semi-supervisado son una mezcla de métodos supervisados y no supervisados. El aprendizaje por refuerzo es un área innovadora y con un gran futuro, ya que está inspirada en mecanismos naturales. En este caso, el algoritmo de aprendizaje recibe información de un entorno real o simulado. Cuando el sistema realiza una acción es recompensado o penalizado, tal y como pasa con los seres vivos. Tales algoritmos de aprendizaje se denominan agentes y pueden aprender siguiendo los principios de la evolución natural. Los agentes aprenden estrategias, denominadas "políticas", que maximizan las recompensas y minimizan las penalizaciones. La mayoría de los sistemas de inteligencia artificial actuales que están especializados en juegos, están basados en el enfoque de aprendizaje por refuerzo (Sandoval, 2018).

4.2.4. Bibliotecas de Python para Machine Learning

Las bibliotecas de aprendizaje automático de Python se han convertido en el lenguaje preferido para las implementaciones de algoritmos de aprendizaje automático. Echemos un vistazo a las principales bibliotecas de Python utilizadas para el aprendizaje automático. En la se describen las Bibliotecas de Python para Machine Learning.

En la **Tabla 1** se describen las Bibliotecas de Python para Machine Learning.

Tabla 1. Bibliotecas de Python para Machine Learning

Biblioteca	**Descripción**
NumPy	Proporciona funciones rápidas y precompiladas para rutinas numéricas, Computación orientada a matrices para una mayor eficiencia. Soporta un enfoque orientado a objetos, y Computaciones compactas y más rápidas con vectorización
SciPy	SciPy destaca por su conjunto de algoritmos basados en NumPy, comandos de alto nivel para manipulación y visualización de datos, procesamiento de imágenes multidimensionales a través de SciPy.ndimage, y funciones integradas para resolver ecuaciones diferenciales.
Scikit-Learn	Está construida sobre dos librerías básicas de Python, NumPy y SciPy. Scikit-learn soporta la mayoría de los algoritmos de aprendizaje supervisado y no supervisado. Scikit-learn también se puede utilizar para la minería de datos y análisis de datos, lo que lo convierte en una gran herramienta que está empezando con ML.

Continúa en la siguiente página

Tabla 1. Bibliotecas de Python para Machine Learning

Biblioteca	Descripción
Theano	Permite definir, evaluar y optimizar expresiones matemáticas que implican matrices multidimensionales de una manera eficiente. Se utiliza ampliamente para pruebas unitarias y autoverificación para detectar y diagnosticar diferentes tipos de errores.
TensorFlow	TensorFlow destaca por sus visualizaciones mejoradas de gráficos computacionales, logrando una reducción de error significativa, entre un 50 y un 60 por ciento, en el aprendizaje automático neuronal. Además, ofrece capacidades de computación paralela para ejecutar modelos complejos.
Keras	Es una biblioteca de aprendizaje automático muy popular para Python, ofrece una API de redes neuronales de alto nivel que es compatible con TensorFlow, CNTK y Theano. Destaca por su capacidad para ejecutarse sin problemas en CPU y GPU, facilitando a los principiantes en el aprendizaje automático la construcción y diseño de redes neuronales.
PyTorch	Es una librería de aprendizaje automático de código abierto implementada en C con una envoltura en Lua. Tiene una amplia gama de herramientas y bibliotecas que soportan Visión por Computador, Procesamiento del Lenguaje Natural (NLP) y muchos más programas de ML.
Pandas	Las características claves de Pandas son una sintaxis elocuente y funcionalidades ricas que le dan la libertad de tratar con datos que faltan, le permite crear su función y ejecutarla a través de una serie de datos, abstracción de alto nivel, y Contiene estructuras de datos de alto nivel y herramientas de manipulación.
Matplotlib	Matplotlib es particularmente útil para el análisis de correlación de variables, visualizar los intervalos de confianza del 95 por ciento de los modelos, la detección de valores atípicos mediante un gráfico de dispersión, y visualizar la distribución de los datos para obtener información instantánea.

Fuente: Adaptado de (Saabith et al., 2020).

4.2.5. Predicción de recurso solar

En la actualidad existen multitud de vías de aprovechamiento de recursos renovables para la generación eléctrica, donde cabe destacar la energía eólica, la energía solar termoeléctrica y la fotovoltaica, ya que todas ellas se caracterizan por poseer un grado de madurez lo suficientemente elevado como para ofrecer una alternativa sólida, eficaz y

rentable y competir con el resto de energías prescindiendo, en ocasiones, de incentivación económica para su instauración.

Estas fuentes energéticas abarcan tanto el ámbito industrial como el consumo propio, ya que energías como la solar fotovoltaica son muy modulares y permiten ser instaladas sobre pequeñas superficies, por lo que están al alcance de los consumidores minoritarios (mercado retail). Sin embargo, la mayoría de las energías renovables presentan desventajas notables que impiden la completa sustitución de los métodos de extracción tradicionales. Entre los más notables se encuentran:

4.2.5.1 Restricciones en la indisponibilidad del recurso. Por la propia naturaleza del recurso renovable del que se trate, este puede no estar disponible a lo largo de todo el día. Este hecho es obvio si se piensa en energía solar o en energía eólica que, en días nublados y sin ráfagas constantes de vientos, respectivamente, no disponen del recurso necesario para la producción energética.

4.2.5.2 Indisponibilidad geográfica. En ocasiones, el recurso renovable no está disponible en todas las zonas geográficas o no existen las condiciones óptimas o incluso mínimas que permitan la instalación de centrales capaces de aprovechar el recurso.

4.2.5.3 Límite de actuación dentro de un intervalo. Derivado del punto anterior, y aun en el supuesto de contar con condiciones favorables, las centrales eólicas y solares tienen límites de actuación impuestas por las variables meteorológicas que obligan, en el caso de sobrepasar unos límites establecidos, a parar la extracción energética.

4.2.5.4 Coste por encima de las fuentes tradicionales. Las tecnologías tradicionales de extracción energética se encuentran en un grado de madurez muy avanzado (referido a términos de economías de escala), por lo que los costes de producción se hallan muy depurados. Desde el punto de vista del autoconsumo, las energías alternativas, aunque competitivas, suponen un coste inicial muy superior, lo que es un obstáculo en su despliegue y comercialización.

4.2.5.5 Gestionabilidad y almacenamiento. El mercado energético actual se establece a través de la subestación de "lotes" de energía según la demanda esperada del recurso para la jornada y/o periodo de negociación. Cualquier variación del pronóstico de la demanda debe ser cubierta instantáneamente, obligando a los proveedores energéticos a aumentar/disminuir la producción. La mayoría de las energías renovables no son gestionables, ya que no se puede regular el recurso natural del que se sirven. Las alternativas que hacen posible la gestionabilidad pasan por métodos de almacenamiento que, actualmente, muchos de ellos se encuentran en fase de ensayo de laboratorio.

4.2.5.6 Incapacidad de predicción. Muchas fuentes energéticas renovables se sirven de recursos naturales que, a menudo, no son predecibles de forma exacta. La predicción del recurso natural es un factor clave de cara a la venta en el mercado energético, ya que, en caso de no proporcionar el "lote" que ha sido adjudicado a un proveedor, este está expuesto a ser penalizado. La predicción de variables como el viento o el sol es hoy en día una vía de investigación abierta (Benítez, 2013).

Para la predicción del recurso solar es necesario el diseño e implementación de un modelo de predicción del recurso solar irradiante sobre una determinada zona geográfica determinada, en este caso será la FEIRNNR. Los datos de partida son recogidos por instrumentación meteorológica, que arrojan datos en todo momento diversos indicativos del estado de la atmosfera. Además, el principal problema lo representan las nubes que, al actuar de forma transitoria por variar su posición en un intervalo de segundos o minutos, provocan cambios bruscos de temperatura (estrés térmico) y reducen la vida útil de los captadores, además de imposibilitar un cálculo preciso de la potencia solar ideal extraída. El desplazamiento de los transitorios repercute directamente en las labores de operación y mantenimiento de la planta.

4.3. Capítulo III: Evaluación de modelos de Machine Learning

4.3.1. Evaluación del modelo

Debido a que cada modelo da como resultado un sesgo en la construcción del modelo que otorga la solución al problema, resulta importante evaluar cómo de eficiente es el aprendizaje del algoritmo durante su etapa de entrenamiento. Dependiendo del tipo de modelo utilizado, es posible valorar la precisión del modelo mediante una evaluación con los datos de prueba. En algunos casos, se requiere del desarrollo de medidas de rendimiento específicas para la aplicación prevista.

4.3.1.1 Tratando con Datos. En machine learning, los datos son la base de todo; no habrá aprendizaje si no hay suficientes datos, o estos no son representativos o presentan información sesgada. Cuando la cantidad de datos es insuficiente, los algoritmos de machine learning no pueden generalizar los resultados: simplemente aprenden los patrones específicos de las muestras existentes.

Incluso si vamos a usar algún conjunto de datos que contiene información representativa y no sesgada, machine learning podría presentar fallos si la información no es de calidad.

Muchos de los algoritmos de machine learning funcionan mejor cuando todas las características y el valor numérico objeto están en el mismo rango. Además, evitando que los valores de entrada sean muy grandes, se ayuda a los algoritmos a encontrar la solución

y a encontrarla de manera más rápida. Por esta razón, de manera habitual, es necesario llevar a cabo un proceso de Normalización sobre las muestras. La Normalización más simple consiste en dividir cada muestra con el valor máximo de dichas muestras. Para obtener un rango que vaya de 0 a 1, en la **Ecuación 1** de la Normalización es:

$$X_{\mathrm{i}} = \frac{x_{\mathrm{i}} - \mathrm{mín}(X)}{\mathrm{máx}(X) - \mathrm{mín}(X)} \tag{1}$$

Donde:

$\mathrm{X_i} =$ es el valor normalizado,

$x_{\mathrm{i}} =$ es el valor original de la característica,

$\mathrm{mín}(X) =$ es el valor mínimo de la característica en el conjunto de datos,

$\mathrm{máx}(X) =$ es el valor máximo de la característica en el conjunto de datos.

La normalización puede fallar cuando hay información sesgada o existen valores atípicos (outliers). Por este motivo, la estandarización (normalización, z-score o standardization) es, muy a menudo, la mejor opción. La estandarización centra los datos en su media y los distribuye de acuerdo con el valor de la desviación típica. Un valor estandarizado puede ser interpretado como el número de desviaciones típicas que lo separa de la media. Se calculan con la **Ecuación 2**.

$$X_{\mathrm{i}} = \frac{x_{\mathrm{i}} - \mu\,(X)}{\sigma\,(X)} \tag{2}$$

Donde:

$\mu\,(X) =$ media de la característica,

$\sigma\,(X) =$ desviación estándar de la característica.

Para tratar con los valores vacíos en las características, los enfoques típicos son:

- Eliminar las muestras que contengan valores vacíos.
- Insertar los valores correctos cuando sea posible: p. ej: código postal o dirección postal.
- Utilización de la técnica de data imputación.

La técnica de data imputation consiste en reemplazar los valores vacíos en una característica por alguna predicción. El sistema más simple para predecir es la media de los valores de la característica: p. ej.: De esta manera, el método de machine learning

puede aprender que las muestras con este valor no contribuyen al modelo. De manera similar, se puede añadir una característica nueva para marcar los valores vacíos con el número uno y los no vacíos con el número cero. Finalmente, se puede usar una regresión para predecir los valores (Bobadilla, 2021).

4.3.1.2 Medición de la Calidad. Normalmente, no es posible saber visualmente si un resultado es mejor que otro; resulta necesario procesar algunas medidas de calidad para conocer la exactitud de los resultados. Dependiendo del enfoque de machine learning que estemos usando, podemos emplear diferentes medidas de calidad: algunas son aplicables a resultados de clasificación, otras pueden ser aplicables a regresión, etc. Vamos a explicar cada una de las medidas de calidad más importantes, pero primero expondremos la metodología.

La regla de oro en metodología de la validación es: no medir la calidad con el mismo conjunto de datos empleado para obtener el modelo. Queremos evitar situaciones en las que el modelo aprende cada una de las muestras de datos, pero es incapaz de tratar datos nuevos. Es necesario dividir las muestras de datos y los valores objetivo en varios conjuntos:

- Entrenamiento
- Validación
- Test

El conjunto de datos de entrenamiento es el mayor: un valor típico es que sea el 80 % del conjunto total de muestras. El conjunto de validación contiene las muestras usadas para mejorar el modelo a base de realizar un ajuste fino en los hiperparámetros (valores que controlan diferentes variaciones en el funcionamiento de los algoritmos). El conjunto test (o pruebas) nos permitirá medir la calidad del modelo utilizando las muestras que no hayan sido usadas para entrenar el modelo o para mejorarlo. Debemos asegurarnos de que los tres conjuntos tienen una distribución similar de los datos, que contienen datos representativos y que su intersección es nula. Una vez que hayamos entrenado el modelo con el conjunto de muestras de entrenamiento y lo hayamos mejorado el modelo con el conjunto de muestras de validación, podremos usar las muestras de test para hacer predicciones con estos datos, que no han sido previamente procesados. Los valores indicadores de la calidad serán tanto mayores cuanto más se asemejen las predicciones obtenidas a los valores objetivo existentes.

Dos medidas de calidad bien conocidas son el *Error Medio Absoluto (Mean Absolute Error o MAE) y el Error Cuadrático Medio (Mean Squared Differences o MSD)*. Ambas medidas de calidad penalizan la distancia que hay entra cada valor de una predicción y

el valor objetivo. Se calculan por medio de la **Ecuación 3**.

$$MSD\,(X, y) = \frac{1}{N}\sum_{\mathrm{i}=1}^{N}\left(\hat{y}_{\mathrm{i}} - p\,(X_{\mathrm{i}})\right)^2 \tag{3}$$

Donde:

$N =$ número de observaciones,

$\hat{y}_{\mathrm{i}} =$ es el valor original de la característica,

$p\,(X_{\mathrm{i}}) =$ valor predicho correspondiente.

Ambas ecuaciones devuelven el error medio cometido en las N muestras de test. El error en ambas ecuaciones es la diferente entre el valor objeto real y el valor predicho para la muestra X_{i}. MSD penaliza los errores con los valores grandes (los eleva al cuadrado) mucho más de lo que hace el MAR. La siguiente **Figura 5** muestra los errores en un modelo de regresión lineal: las líneas verticales que unen los valores reales con los predichos.

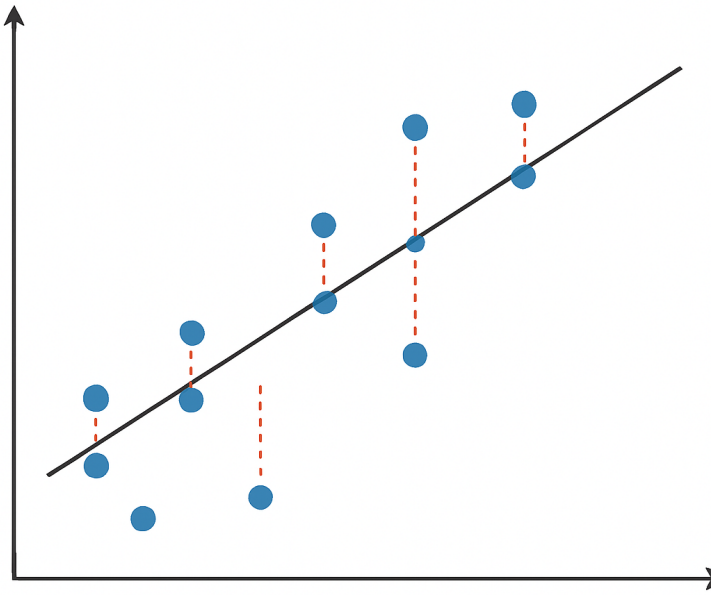

Figura 5. Muestra los errores en un modelo de regresión Lineal.
Fuente: (Bobadilla, 2021).

4.3.1.3 Mejora del Modelo. Para mejorar un modelo de machine learning se debe medir la calidad que obtiene al actuar sobre el conjunto de muestras de test (pruebas). En esta sección nos centramos en los conjuntos de datos de entrenamiento y de pruebas

(test), pero debemos recordar que los proyectos profesionales de machine learning deben usar también el conjunto de datos de validación. Primero se usa el conjunto de datos de entrenamiento para crear el modelo, y podemos, entonces, obtener sus medidas de calidad (aplicadas a las muestras de entrenamiento). Si las predicciones del modelo presentan muchos errores, las medidas de calidad devolverán valores muy bajos. Esto significa que los datos no son suficientemente numerosos, o presentan muchos valores atípicos, o los hiperparámetros de aprendizaje elegidos no son correctos. En estos casos, el modelo sufre sub ajuste (underfitting). Para ilustrar el caso, reproducimos la **Figura 6**, obtenido de datos generados. Se puede ver que el modelo lineal es demasiado simple para ajustar la distribución no lineal de los datos (Bobadilla, 2021).

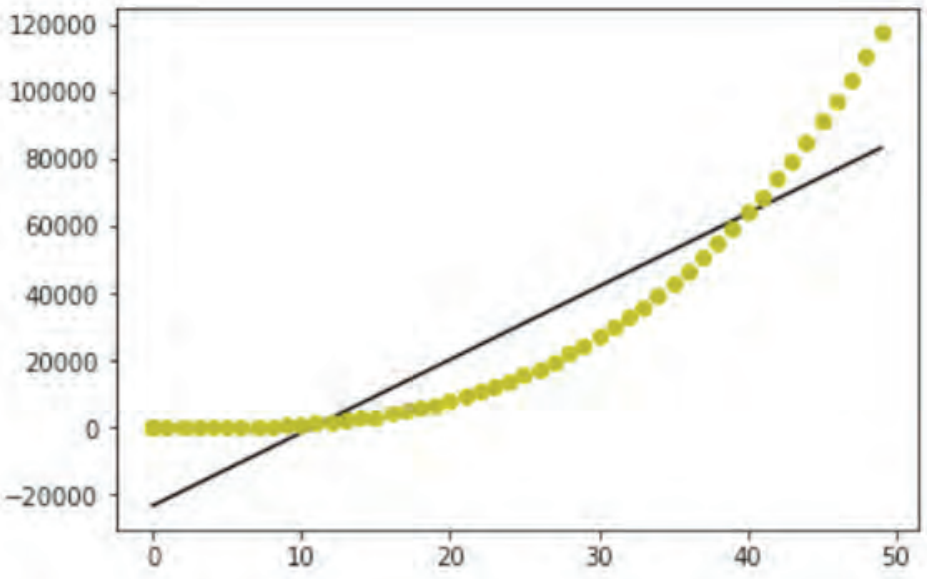

Figura 6. Muestra los errores en un modelo de regresión Lineal.
Fuente: (Bobadilla, 2021).

4.3.2. Reconstrucción de datos en radiación solar

Para reconstruir los datos de radiación solar, se requiere estimar la cantidad de radiación presente en ubicaciones y momentos específicos donde no hay mediciones directas disponibles. Esta etapa es de suma importancia en varios contextos, como la organización de proyectos solares, el estudio del clima y el análisis del potencial energético solar. A continuación, se describen algunos métodos comunes empleados en la reconstrucción de datos de radiación solar:

4.3.2.1 Modelos meteorológicos. Para prevenir la radiación solar, los modelos meteorológicos utilizan información tanto del pasado como del presente sobre el clima. Estos modelos toman en cuenta elementos como la localización geográfica, el nivel de altura, la pendiente del terreno y las condiciones climáticas. Varios modelos meteorológicos amplia-

mente utilizados incorporan el modelo de generación de tiempo solar (SGM) y el modelo de radiación solar desarrollado por la NASA (NASA SSE).

4.3.2.2 Sensores satelitales. Los satélites meteorológicos ofrecen una perspectiva desde lo alto, capturando la radiación solar es reflejada y emitida tanto por la atmósfera como por la superficie terrestre. Mediante el uso de algoritmos especializados, se analizan estos datos para calcular la radiación solar a nivel del suelo.

4.3.2.3 Estaciones meteorológicas. La radiación solar es medida directamente por las estaciones meteorológicas terrestres. Se emplean estos datos para ajustar y comprobar la precisión de modelos y métodos de cálculo. No obstante, la obtención de datos puede ser escasa en zonas alejadas.

4.3.2.4 Redes neuronales y aprendizaje automático. Los enfoques de aprendizaje automático, como las redes neuronales, pueden utilizarse para aprender patrones complejos en los datos climáticos y predecir la radiación solar. Estos modelos pueden aprovechar una diversidad de variables climáticas para realizar predicciones precisas.

4.3.2.5 Interpolación espacial y temporal. La interpolación es una técnica que se emplea para poder estimar los niveles de radiación solar en lugares y momentos donde no se cuentan con datos directos. Se pueden emplear técnicas como el kriging espacial y la interpolación temporal para completar las ausencias en los datos. Es relevante destacar que la precisión de las reconstrucciones varía según la calidad de los datos disponibles y el método seleccionado. Utilizar diversas fuentes de datos y enfoques puede mejorar la precisión de las estimaciones de radiación solar (Lalaleo, 2021).

5. Metodología

5.1. Área de estudio

El trabajo consistió en desarrollar un modelo de predicción de irradiación solar, el mismo que se realizó en y para la Facultad de Energía, las Industrias y los Recursos Naturales no Renovables de la U.N.L., haciendo uso de los algoritmos que ofrece machine learning, se tomó como un punto de partida, la obtención de los datos de irradiación solar, gracias a la estación meteorológica ubicada dentro de la misma facultad, esta investigación se la realizó durante un periodo de 12 meses desde enero del 2021 hasta diciembre del 2021.

La Facultad de Energía, Industrias y Recursos Naturales No Renovables está situada a una altitud de 2060 metros sobre el nivel del mar, entre las avenidas Reinaldo Espinoza y Eduardo Kingman. Su ubicación específica se caracteriza por las coordenadas geográficas XR92+J5C, C. 9, en la ciudad de Loja. En la **Figura 7** se encuentra el lugar donde se llevó a cabo la propuesta de investigación.

Figura 7. Ubicación de la Facultad de Energía.

Fuente: (Maps, 2024).

5.2. Materiales

En el presente trabajo se utilizó como los datos de entrada la fecha (fecha y hora) y como la variable objetiva de la irradiación solar proporcionada por la estación meteorológica de la facultad, del año 2021 de los cuales se realizó una limpieza de los mismos, para poder trabajar con los datos registrados a partir desde las 6 AM hasta las 6:30 PM. El presente proyecto requiere usar los siguientes recursos y materiales.

5.2.1. Recursos humanos

- Tutor de tesis

5.2.2. Recursos bibliográficos

- Datos de irradiación solar
- Libros de energía solar
- Libros de machine learning
- Libros de métricas de machine learning
- Tesis de Predicción utilizando machine learning
- Artículos científicos de forecasting de radiación solar

5.2.3. Recursos de oficina

- Paquete Office
- Software Python
- Equipos computacionales
- LaTeX

5.3. Metodología

5.3.1. Procedimiento

La presente investigación implica la recopilación de datos y la identificación de variables clave, utilizando tanto herramientas de *Machine Learning* como enfoques de *data mining*. En este contexto, se adoptará una orientación mixta. Las variables cualitativas estarán representadas por las herramientas de *Machine Learnin*g y *data mining*, mientras que las variables cuantitativas se basarán en los datos suministrados por el CITE(Centro de Investigaciones Tecnológicas y Energéticas), encargado de recopilar información sobre la irradiación solar a través de dispositivos instalados en la FEIRNNR. El principal objetivo es el desarrollo de uno o unos modelos de predicción, utilizando los datos recopilados de la estación meteorológica.

Para esto, se utilizó las siguientes técnicas:

Recolección y limpieza de datos: Se adquirieron datos de la estación meteorológica; sin embargo, no eran los datos que necesitamos de irradiación solar, solo existían datos de radiación solar, por lo que se tuvo que multiplicar por un factor de tiempo, el cual fue de (5/60)h para poder obtener la irradiación solar, posteriormente a ello realizo la limpieza de datos, los cuales podrían estar adulterados o erróneos, esto se lo realizo por medio del *Software Python* que a su vez se lo realizo gracias al Google Colab, que es una extensión del mismo y gracias a esto obtener los datos cuantitativos de las variables de interés.

Modelos *Machine Learning*: se desarrolló los modelos de *machine learning* que se encargan de realizar la predicción de la irradiación solar, haciendo uso de las variables cuantitativas, las que usaremos luego de realizar la limpieza de datos, las cuales son la fecha y la irradiación solar promedio.

Evaluación de los modelos con las métricas seleccionadas: Para finalizar, luego de terminar los modelos de predicción, se aplicó las diferentes métricas seleccionadas para evaluar el rendimiento de cada uno de los modelos de predicción realizados.

Durante el período establecido, se llevaron a cabo todas las actividades correspondientes con el fin de alcanzar cada uno de los objetivos planteados, como se observa en la **Figura 8**. Se implementaron estrategias específicas y se asignaron recursos adecuados para garantizar la ejecución exitosa de cada paso.

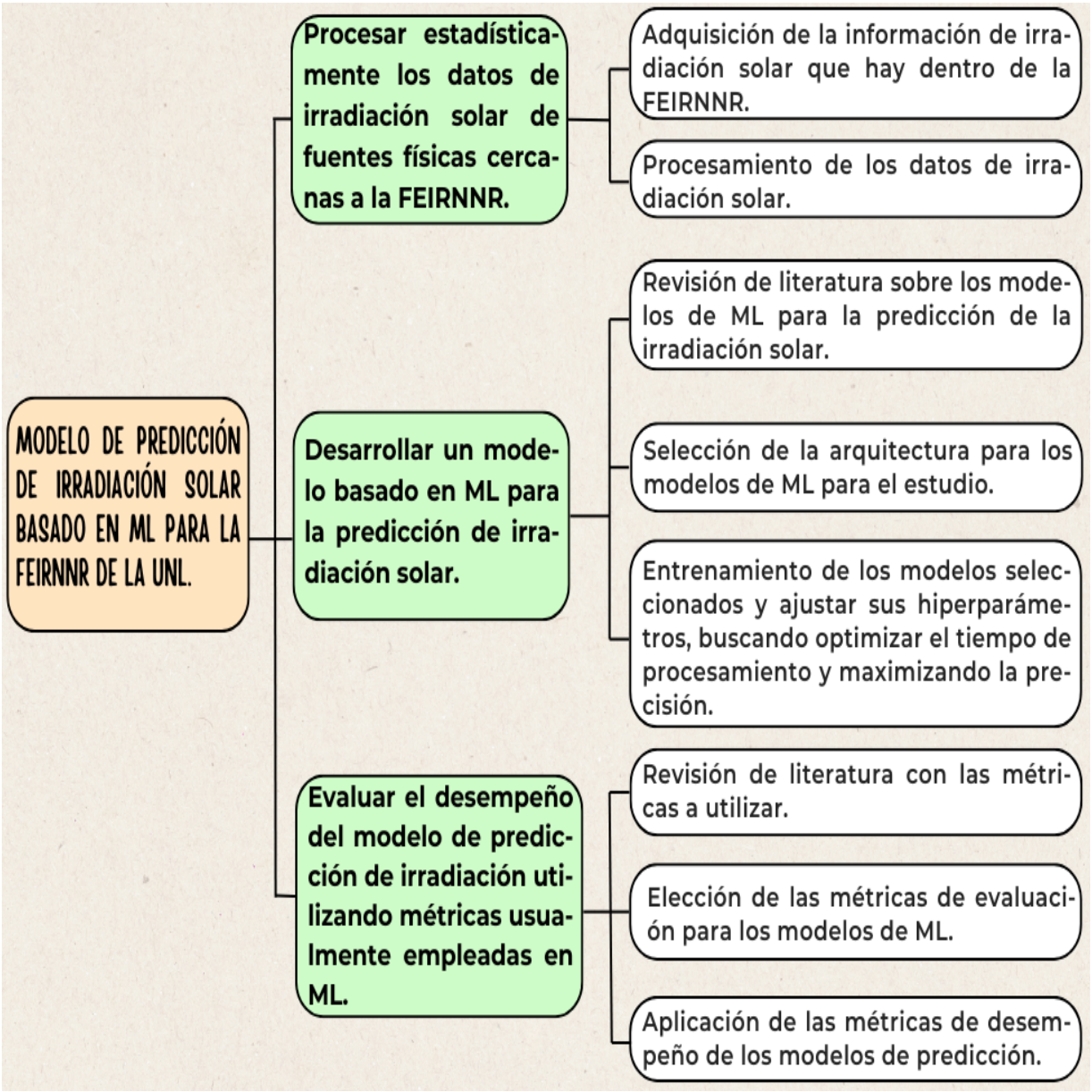

Figura 8. Procedimiento realizado en la investigación.

5.3.1.1 Procesar estadísticamente los datos de irradiación solar de fuentes físicas cercanas a la FEIRNNR. Para dar cumplimiento el primer objetivo se siguieron las siguientes actividades que se muestran en la **Figura 9**.

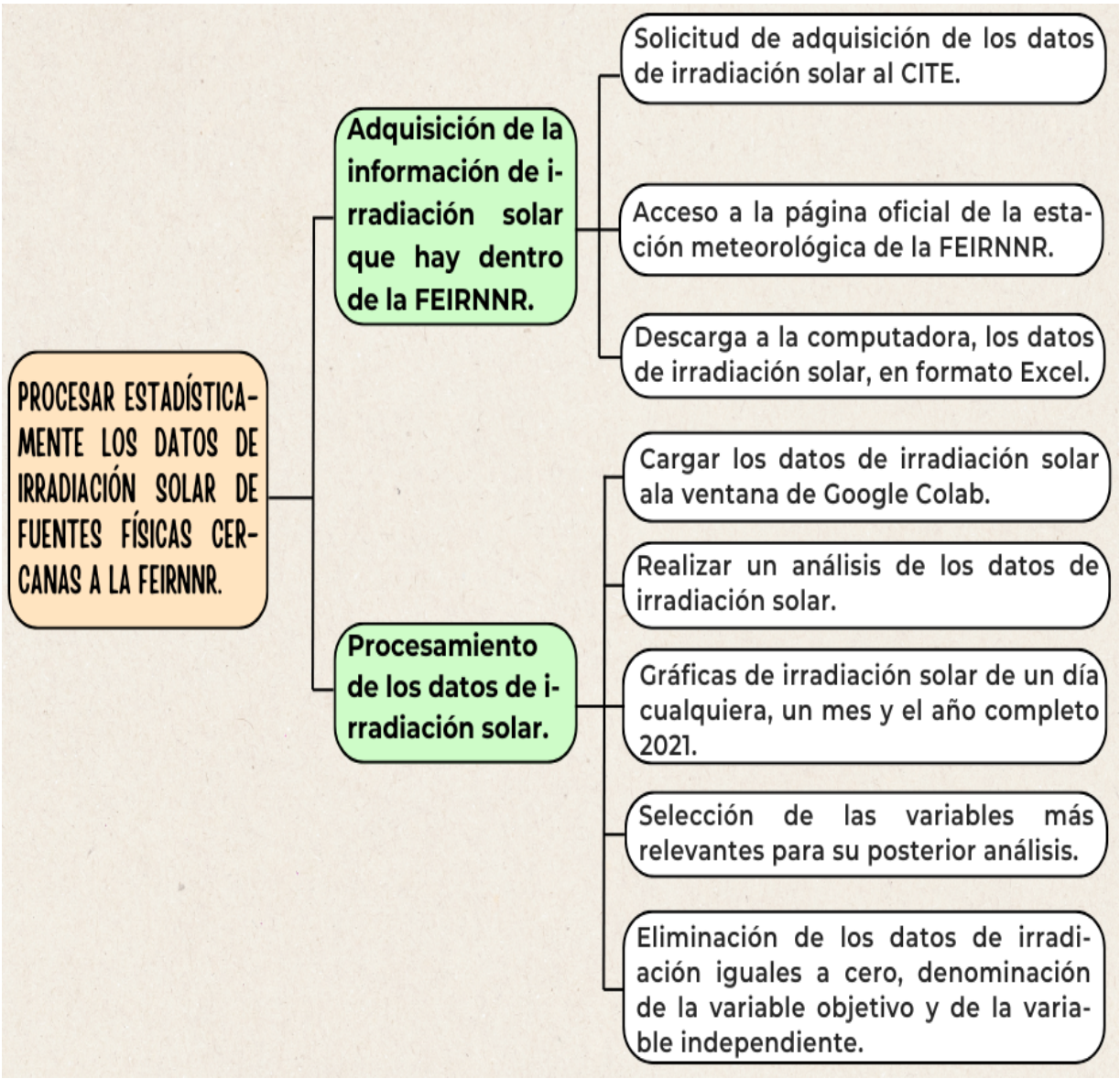

Figura 9. Procedimiento del primer objetivo.

En esta actividad se siguieron los siguientes pasos.

1. Adquisición de la información de irradiación solar que hay dentro de FEIRNNR.
   (a) Solicitud de adquisición de los datos de irradiación solar al CITE: se realizó una solicitud para la adquisición de datos de la radiación, dirigida al Ingeniero Juan Carlos Solano.
   (b) Acceso a la página oficial de la estación meteorología de la FEIRNNR: Luego de entregar la solicitud al Ingeniero Juan Carlos Solano, se me permitió el acceso a la página de la estación meteorológica.
   (c) Luego del acceso éxito a la página oficial de la estación meteorológica, se procedió a la descarga de los datos que nos ofrece, gracias a los diferentes equipos y sensores tales como el Datalogger QML201 Vaisala, Módem GSM y GPRS

Sierra Wireless Fastrack FXT009, Piranómetro Kipp Zonen CMP 3, Sensor ultrasónico de viento Young 86000, Sonda de humedad y Temperatura Vaisala HMP110, Phocos CML10 Solar Charge Controller 12-24V, 10 A y Baterías de plomo-ácido PS-12260, permitiendo así un seguimiento detallado de las condiciones meteorológicas, estos datos se descargaron en un formato Excel directamente a la computadora. La información recopilada considera toma de datos cada cinco minutos y por el lapso de un año correspondiente al año 2021.

2. Procesamiento de los datos de irradiación solar.
   (a) Cargar los datos de irradiación solar a la ventana de Google Colab: Al contar con los datos que nos ofrece la estación meteorológica, se los sube al entorno de Google Colab, haciendo uso de las líneas de código necesarias para subirlos a este entorno virtual.
   (b) Realizar un análisis de los datos de irradiación solar: Ya con los datos subidos al entorno del Google Colab, se saca todas las características de los datos, de esta manera se podrá ver si existen datos adulterados o valores nulos, puedan entorpecer el desarrollo de la investigación planteada.
   (c) Gráficas de irradiación solar de un día cualquiera, un mes y del año completo 2021: Para una mayor compresión de la variable de análisis, seleccionamos la variable de irradiación solar global promedio junto a la fecha y realizamos unas gráficas por separado.
   (d) Selección de las variables más relevantes para su posterior análisis: Luego del análisis de las gráficas de la irradiación solar promedio, se selecciona las columnas más relevantes, en este caso serán las de irradiación solar promedio junto a la de las fechas, esta selección se tomará a partir desde las 6 AM hasta las 6:30 PM.
   (e) Eliminación de los datos de irradiación iguales a cero, denominación de la variable objetivo y un conjunto de variables independientes: Luego de la selección de las columnas seleccionadas se elimina los valores de irradiación que son iguales a cero, de esta manera se invitara entorpecer el desarrollo del modelo de predicción. Finalmente, denominamos las variables, donde (y) es la variable objetivo y la variable (X) es el conjunto de variables independientes.

5.3.1.2 Desarrollar un modelo basado en Machine Learning para la predicción de irradiación solar. Para dar cumplimiento el segundo objetivo se siguieron las siguientes actividades que se muestran en la **Figura 10**.

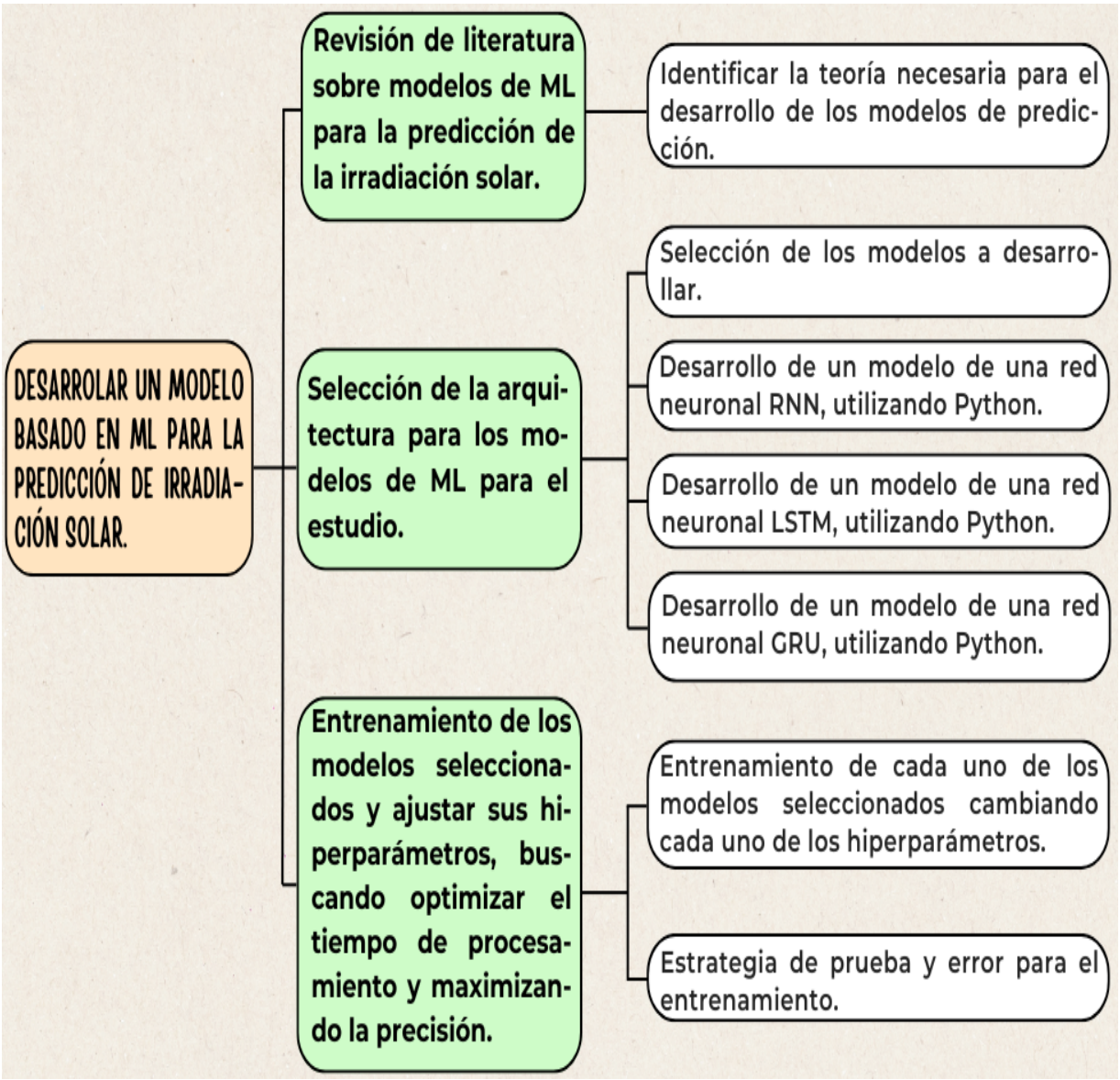

Figura 10. Procedimiento del segundo objetivo.

En esta actividad se siguieron los siguientes pasos.

1. Revisión de literatura sobre modelos de Machine Learning para la predicción de la irradiación solar.
   (a) Identificar la teoría necesaria para el desarrollo de los modelos de predicción: Se realizó una búsqueda profunda en donde se encontró la información de los posibles modelos de predicción para la irradiación que se podrían utilizar a lo largo de nuestra investigación.
   (b) Revisión de artículos científicos de Forescasting para la irradiación solar: se buscó información pertinente en varios artículos científicos donde se encuentran muy detallados unos modelos de predicción que se podría utilizar para la el desarrollo de nuestros modelos de predicción.

2. Selección de la arquitectura para los modelos de Machine Learning para el estudio.
   (a) Selección de los modelos a desarrollar: Luego de la revisión bibliográfica, se optó por el desarrollo de tres modelos de predicción, los cuales son:
   (b) Red neuronal recurrente.
   (c) Memoria a corto plazo.
   (d) Unidad recurrente cerrada.
3. Entrenamiento de los modelos seleccionados y ajustar sus hiperparámetros, buscando optimizar el tiempo de procesamiento y maximizando la precisión:
   (a) Entrenamiento de cada uno de los modelos seleccionados, cambiando cada uno de los hiperparámetros: Al finalizar cada uno de los modelos de predicción, se procedió al entrenamiento de cada uno de los modelos variando sus hiperparámetros.
   (b) Buscar que cada uno de los modelos desarrollados, se entrenen en el menor tiempo posible: Al variar cada uno de los hiperparámetros el entrenamiento de los modelos cambia constantemente, entonces se buscó la combinación que menos tiempo tarde en entrenar.

```
# Pérdida de entrenamiento
plt.plot(history.history['loss'], label='Entrenamiento',
color='blue')
# Pérdida de validación
plt.plot(history.history['val_loss'], label='Validación',
color='red')
plt.title('Pérdida de entrenamiento y validación a lo largo de
las épocas')
plt.xlabel('Épocas')
plt.ylabel('Pérdida (MSE)')
plt.legend()
plt.show()
```

Para dar cumplimiento el tercer objetivo se siguieron las siguientes actividades que se muestran en la **Figura 11**.

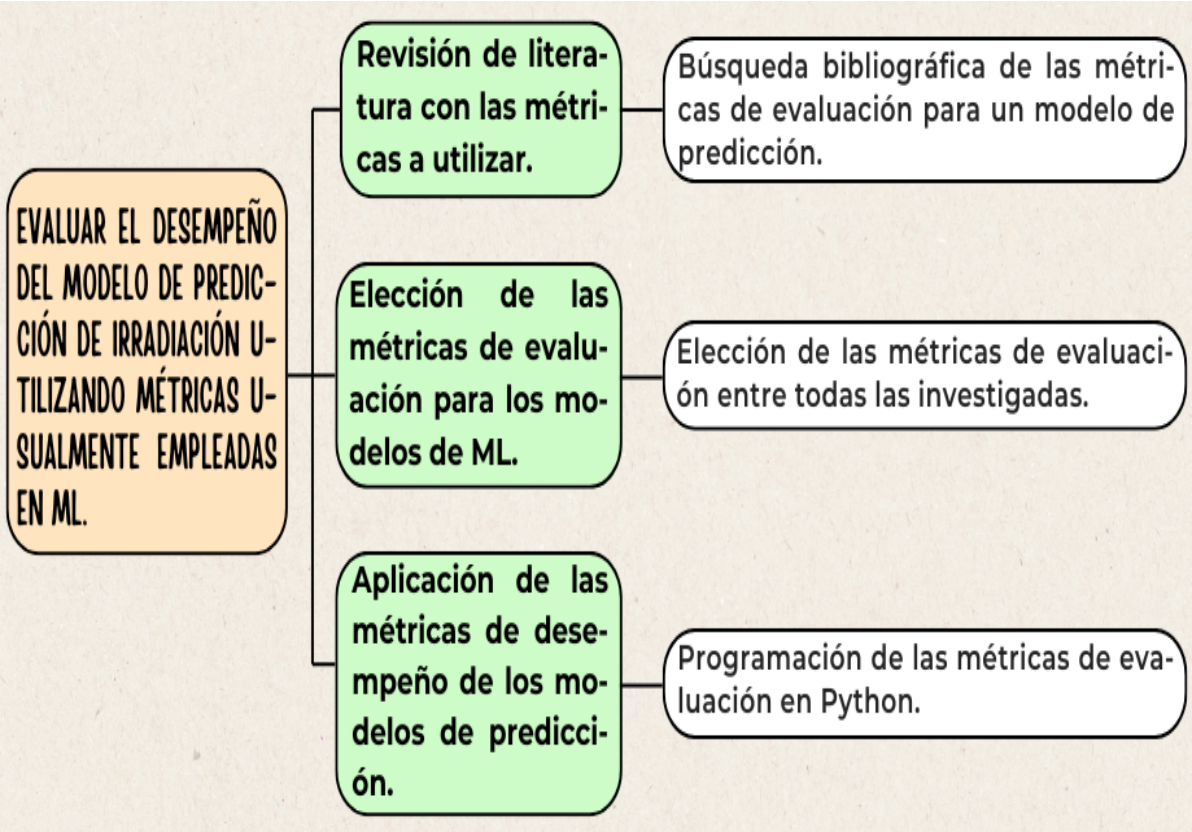

Figura 11. Procedimiento del tercer objetivo.

En esta actividad se siguieron los siguientes pasos.

(a) Búsqueda bibliográfica de las métricas de evaluación para un modelo de predicción: Al finalizar la construcción y entrenamiento de los tres modelos desarrollos es necesario aplicar unas cuantas métricas de evaluación a los modelos desarrollados para comprobar su eficiencia y de esta manera poder garantizar que el trabajo realizado tiene un buen desempeño y de tal manera puede contribuir al desarrollo de nuevos trabajos. Por tal motivo se realizó una búsqueda profunda de las diferentes métricas que existen, de las cuales se realizara una respectiva selección, ya, para dar veracidad a nuestro trabajo no es necesario de aplicar todas las métricas que existen.

(b) Elección de las métricas de evaluación entre todas las consultadas: Luego de revisar las diferentes métricas que existen, se declinó la selección de las tres métricas, las cuales son: revisar en el procedimiento.

(c) Programación de las métricas de evaluación en Python: Al aplicar las métricas seleccionadas se obtuvo unos resultados aceptables, ya que cumplían con lo requerido, el cual era de dar un alto rendimiento a los modelos de predicción, ya que cada uno de los modelos realizados está entre el 85 %.

5.4. Procesamiento y análisis de datos

5.3.1.3 Evaluar el desempeño del modelo de predicción de radiación utilizando métricas usualmente empleadas en Machine Learning.

5.4.1. Procesar estadísticamente los datos de irradiación solar de fuentes físicas cercanas a la FEIRNNR

Para dar inicio a la investigación, se llevaron a cabo diversas actividades, las cuales se detallan a continuación:

- **Solicitud de adquisición de los datos de irradiación solar al CITE.**

Para iniciar la investigación, se requiere adquirir los datos de la estación meteorológica. Para ello, se formuló una solicitud dirigida al Ingeniero Juan Carlos Solano para obtener dicha información„ se puede observar el modelo de la solicitud en el **Anexo 3**.

- **Acceso a la página oficial de la estación meteorología de la FEIRNNER.**

Luego de la entrega de la solicitud al ingeniero antes mencionado, se me hizo la entrega del link oficial de la estación meteorológica, donde pude observar los datos que nos ofrece, partiendo desde temperaturas hasta la radiación solar.

- **Descarga a la computadora de los datos de radiación solar, en formato Excel.**

Una vez que se pudo ingresar a la página web de la estación meteorológica se seleccionó el rango de los datos a descargar, la base de datos más completa fue la del año 2021, por este motivo se optó por descargar solamente la de este año, los mismos que se descargaron en un formato Excel directo a la computadora de trabajo.

Procesamiento de los datos de irradiación solar.

- **Cargar los datos de irradiación solar a la ventana de Google Colab.**

Después de adquirir los datos de la estación, se procedió a cargarlos en el entorno de Google Colab. Para realizar esta tarea, se requiere importar diversas bibliotecas en la ventana de programación. Para cargar los datos desde el archivo Excel, es necesario utilizar la función pd.read_excel y así incorporarlos al entorno de programación de Google Colab, lo cual lo podemos observar en las siguientes líneas de programación.

Código en Python.

```
#Importar todas la librerias necesarias para el modelo.
import datetime as df
```

```
from datetime import datetime
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import tensorflow as tf
%matplotlib inline
from pandas import DataFrame
from sklearn.preprocessing import MinMaxScaler

#Se carga el archivo excel con los datos de Irradiación
df = pd.read_excel('/content/Datos 2021.xlsx')
```

Es muy importante realizar una inspección de la data set, ya que se debe comprobar si existen valores adulterados.Para realizar la inspecion hacemos uso del siguiente código desarrollado en Python.

Código en Python.

```
#Se describre todo el DataFrame del Excel
df.describe()
```

En la siguiente **Figura 12** se puede observar un resumen estadístico de la base de datos.

	TEMPERATURA DEL AIRE PROMEDIO (ºC)	TEMPERATURA DEL AIRE MÁXIMA (ºC)	TEMPERATURA DEL AIRE MÍNIMA (ºC)	RADIACIÓN SOLAR GLOBAL PROMEDIO (W/m^2)	INTERVALO DE TIEMPO PARA LA IRRADIACIÓN	IRRADIACIÓN SOLAR GLOBAL PROMEDIO (Wh/m^2)
count	150134.000000	150134.000000	150134.000000	150134.000000	1.000000	150134.000000
mean	9.703747	10.348158	9.621637	136.458098	0.083333	11.371508
std	9.018671	9.578002	8.937209	223.787961	NaN	18.648997
min	-18.500000	0.000000	-40.000000	0.000000	0.083333	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.083333	0.000000
50%	13.900000	15.000000	13.800000	1.035000	0.083333	0.086250
75%	16.800000	17.900000	16.700000	207.000000	0.083333	17.250000
max	27.900000	29.200000	27.600000	999.000000	0.083333	83.250000

Figura 12. Resumen estadístico de todas las columnas numéricas en la data set.

También es muy importante observar las diferentes variables que existen dentro de la data sed, de la misma manera se debe revisar si en la base de datos existen datos erróneos o nulos, es por ello que es necesario realizar una inspección a la base de datos, para realizarla se la hace por medio de la función df.info(). En la siguiente **Figura 13** se puede observar la información de la data set.

Código en Python.

```python
#Se describre toda la información en el DataFrame del Excel
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150134 entries, 0 to 150133
Data columns (total 19 columns):
 #   Column                                    Non-Null Count   Dtype
---  ------                                    --------------   -----
 0   FECHA Y HORA                              150134 non-null  object
 1   TEMPERATURA DEL AIRE PROMEDIO (ºC)        150134 non-null  float64
 2   TEMPERATURA DEL AIRE MÁXIMA (ºC)          150134 non-null  float64
 3   TEMPERATURA DEL AIRE MÍNIMA (ºC)          150134 non-null  float64
 4   RADIACIÓN SOLAR GLOBAL PROMEDIO (W/m^2)   150134 non-null  float64
 5   INTERVALO DE TIEMPO PARA LA IRRADIACIÓN   1 non-null       float64
 6   IRRADIACIÓN SOLAR GLOBAL PROMEDIO (Wh/m^2) 150134 non-null float64
 7   RADIACIÓN SOLAR GLOBAL MÁXIMA (W/m^2)     150134 non-null  float64
 8   RADIACIÓN SOLAR GLOBAL MÍNIMA (W/m^2)     150134 non-null  float64
 9   RADIACIÓN SOLAR SUMATORIA (W/m^2)         150134 non-null  float64
 10  DIRECCIÓN DEL VIENTO PROMEDIO             150134 non-null  int64
 11  DIRECCIÓN DEL VIENTO MÁXIMA               150134 non-null  int64
 12  DIRECCIÓN DEL VIENTO MÍNIMO               150134 non-null  int64
 13  VELOCIDAD DEL VIENTO PROMEDIO (m/s)       150134 non-null  float64
 14  VELOCIDAD DEL VIENTO MÁXIMA (m/s)         150134 non-null  float64
 15  VELOCIDAD DEL VIENTO MÍNIMA (m/s)         150134 non-null  float64
 16  WIND POWER (W/m^2)                        150134 non-null  float64
 17  RECORRIDO DEL VIENTO (m/s)                150134 non-null  float64
 18  VOLTAJE DE LA BATERÍA (V)                 150134 non-null  int64
dtypes: float64(14), int64(4), object(1)
memory usage: 21.8+ MB
```

Figura 13. Inspección de la data set.

Luego de revisar y ver que no existen valores nulos, se realizó un análisis de todos los datos por medio de gráficas.

- **Gráficas de irradiación solar de un día cualquiera, un mes y del año completo 2021.**

Para una mayor compresión de la irradiación solar que existe dentro de la facultad se realizó una gráfica del primer día del año 2021, al igual que el mes (julio) del año mencionado. Se puede observar en la **Figura 14** y **Figura 15**.

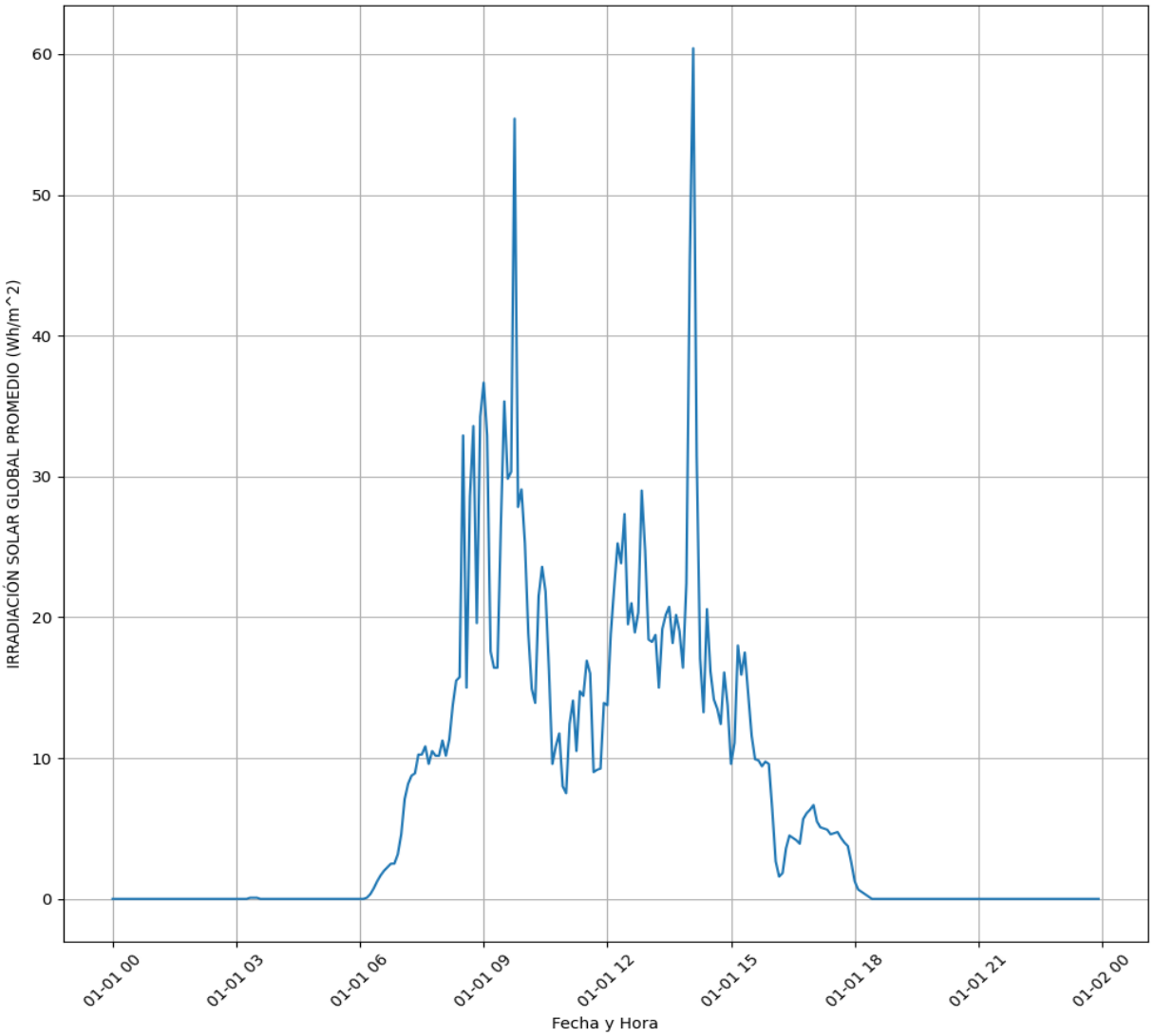

Figura 14. Irradiación solar promedio del primer día del año 2021.

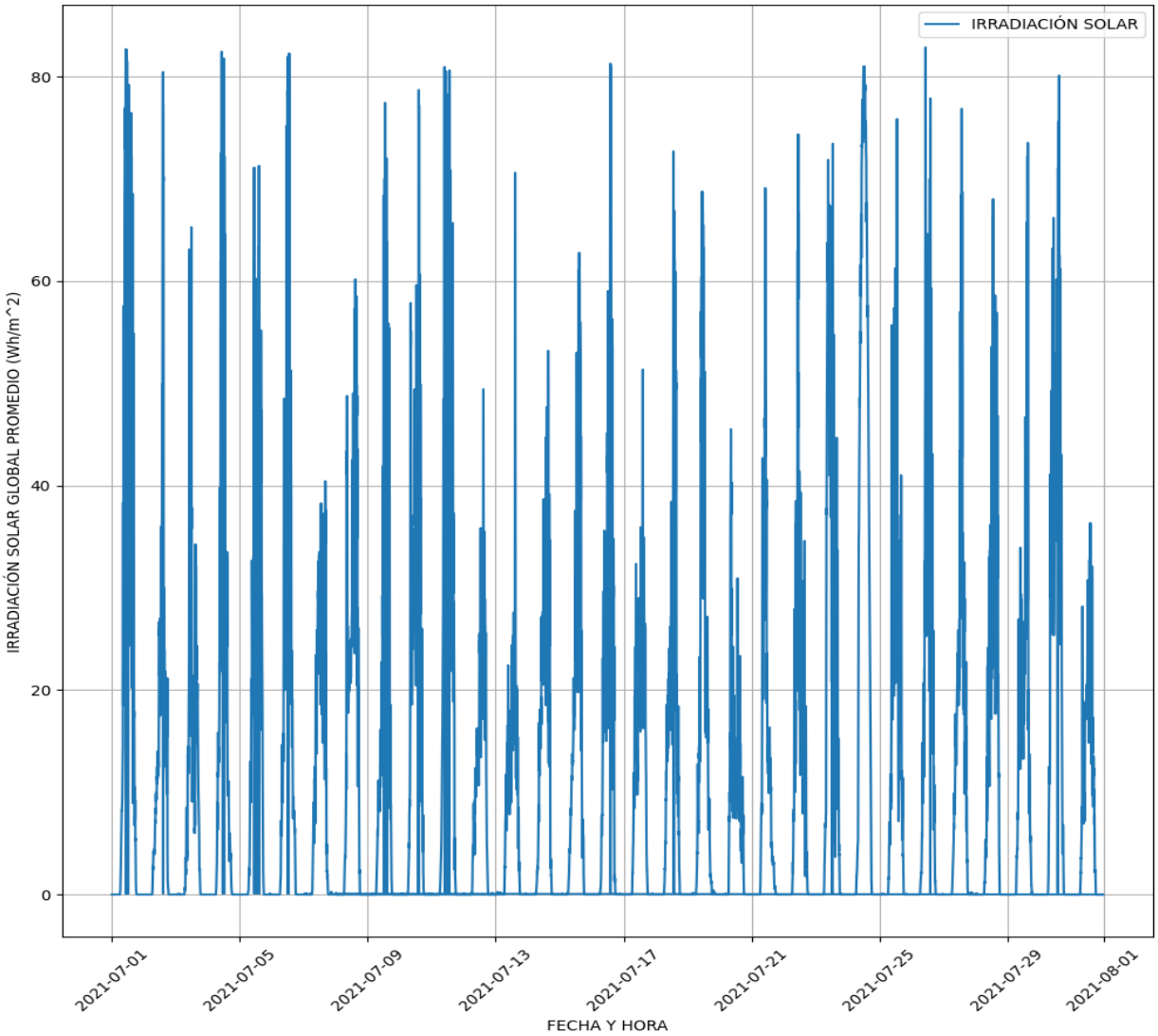

Figura 15. Irradiación solar promedio del mes de julio del 2021.

De la misma manera se realizó una gráfica de todo el año 2021 donde se puede observar en la **Figura 16** de toda la irradiación solar promedio.

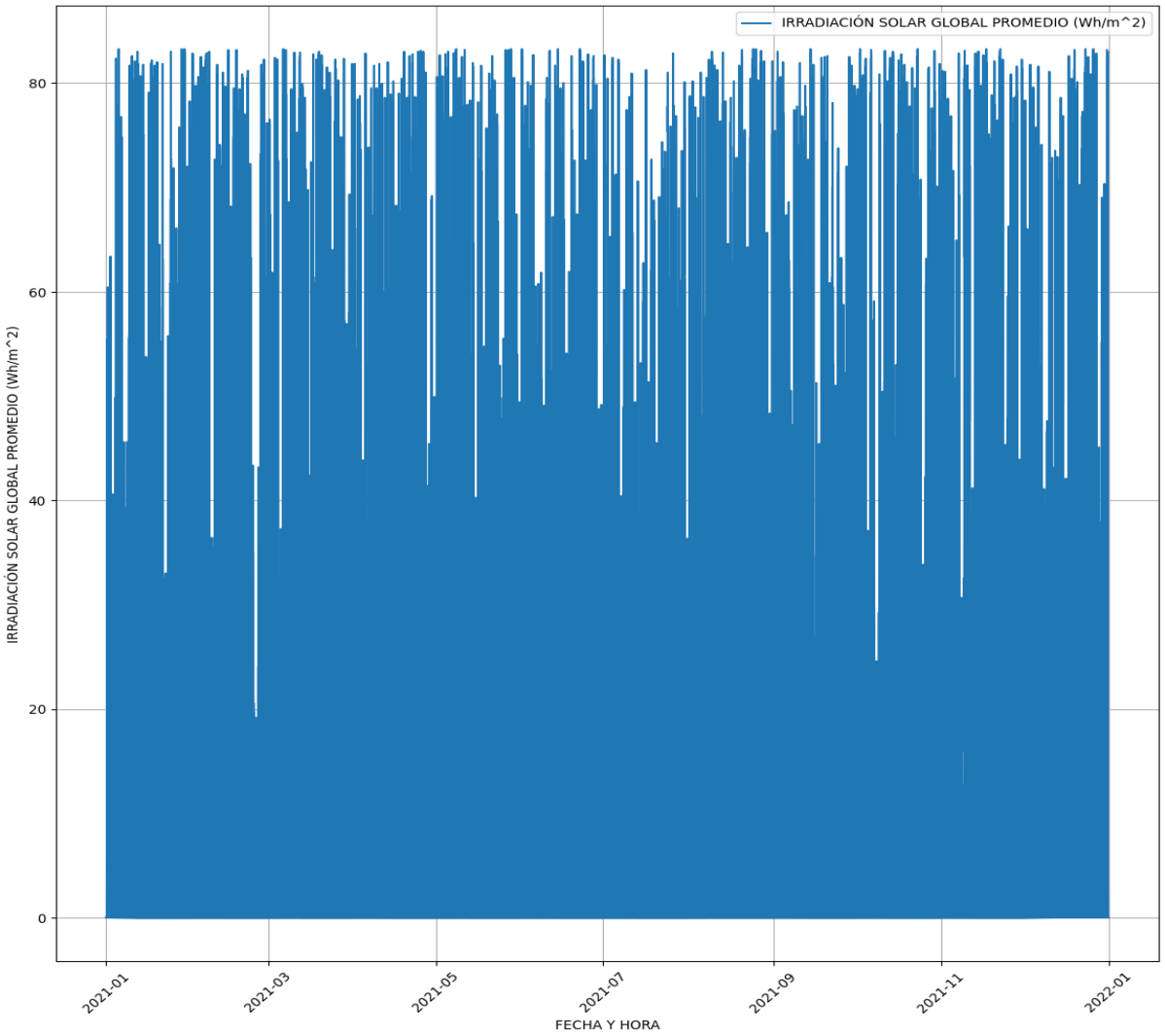

Figura 16. Irradiación solar promedio de todo el año 2021.

- **Selección de las variables más relevantes para su posterior análisis.**

Para iniciar la construcción de modelos de predicción con técnicas de machine learning es vital realizar una selección de variables, puesto que estas determinan la calidad de las interferencias que el modelo podría generar. Para este caso, se prefirió por usar solo dos variables: **FECHA Y HORA** como representación temporal, e **IRRADIACIÓN SOLAR GLOBAL PROMEDIO** como la variable objetivo. La selección se basa en el hecho que la **IRRADIACIÓN SOLAR GLOBAL PROMEDIO** está claramente influenciada por patrones temporales recurrentes, como el ciclo diario y estacional, lo que permite que la variable temporal (**FECHA Y HORA**) contenga una gran cantidad de información importante para el aprendizaje del modelo. Según (Géron, 2019), los modelos de aprendizaje automático consiguen obtener un rendimiento significativo cuando se entrenan con variables que reflejan relaciones temporales claramente definidas, especialmente en fenómenos físicos que dependen del tiempo, como la energía solar.Al reducir el número de variables a solo aquellas que capturan la dimensión temporal y la respuesta

a predecir, se busca minimizar el ruido y la complejidad del modelo, facilitando así un entrenamiento más eficiente y menos propenso al sobreajuste.

Al observar, que en la data set no existen valores adulterados o nulos, es necesario realizar un filtrado de datos con respecto al tiempo de interés para nuestra investigación, en el siguiente apartado se puede apreciar el código desarrollado en Python, para realizar el filtrado desde las 6 AM hasta las 6:30 PM.

Código en Python

```
    # Se filtran los datos de 6 AM a 6:30 PM.
df['FECHA Y HORA'] = pd.to_datetime(df['FECHA Y HORA'], format='%d/%m
/%Y%I:%M %p')
data = df[(df['FECHA Y HORA'].dt.hour >= 6) & ((df['FECHA Y HORA'].dt.
hour < 18) | ((df['FECHA Y HORA'].dt.hour == 18) & (df['FECHA Y HORA'].
dt.minute <= 30)))]
data.head()
```

Después del filtrado de las horas de interés es necesario realizar un filtrado de las columnas de interés para la investigación, las siguientes líneas de código se encuentran desarrolladas en Google Colab para la selección de dichas columnas, que en este caso son: la columna uno que es la fecha y hora, mientras que la columna dos es la irradiación solar promedio, de la misma manera se puede observar que los datos aparecen a partir desde las 6 de la mañana hasta las 6:30 de la tarde, en la siguiente **Figura 17** se puede observar lo anteriormente mencionado.

Código en Python

```
#Seleccionamos las Columas de 'Fecha y Hora' y 'IRRADIACIÓN SOLAR GLOBAL
    PROMEDIO (Wh/m^2)'
columnas_seleccionadas = data[['FECHA Y HORA','IRRADIACIÓN SOLAR GLOBAL
   PROMEDIO (Wh/m^2)']]
columnas_seleccionadas.head()
```

	FECHA Y HORA	IRRADIACIÓN SOLAR GLOBAL PROMEDIO (Wh/m^2)
73	2021-01-01 06:00:00	0.000000
74	2021-01-01 06:05:00	0.000000
75	2021-01-01 06:10:00	0.083333
76	2021-01-01 06:15:00	0.333333
77	2021-01-01 06:20:00	0.750000
...	...	...
150064	2021-12-31 18:10:00	2.166667
150065	2021-12-31 18:15:00	1.333333
150066	2021-12-31 18:20:00	0.833333
150067	2021-12-31 18:25:00	0.416667
150068	2021-12-31 18:30:00	0.083333

Figura 17. Selección de las columnas de interés.

- **Eliminación de los datos de irradiación iguales a cero.**

Una vez seleccionado el rango de las horas es necesario realizar, una eliminación de los datos que son igual a cero en cuanto a la irradiación solar promedio, para un mejor análisis del mismo, en el siguiente apartado se puede apreciar el código desarrollado en Python.

Código en Python

```
    # Eliminación de los datos iguales a cero.
# Mostrar el tama o del DataFrame antes de eliminar los ceros
print(f'Tama o del set antes de eliminar los datos iguales a cero:{
    columnas_seleccionadas.shape}')
# Eliminar los datos iguales a cero en la columna de irradiación solar
columnas_seleccionadas=columnas_seleccionadas[columnas_seleccionadas['
    IRRADIACIÓN SOLAR GLOBAL PROMEDIO (Wh/m^2)']>0]
# Mostrar el tama o del DataFrame después de eliminar los ceros
print(f'Tama o del set despues de eliminar los datos iguales a cero:{
    columnas_seleccionadas.shape}')
```

En la siguiente **Figura 18** se puede observar la limpieza de los mismos.

```
Tamaño del set antes de eliminar los datos iguales a cero:(52920, 2)
Tamaño del set despues de eliminar los datos iguales a cero:(52028, 2)
```

Figura 18. Eliminación de los datos de irradiación iguales a cero.

Por último, y no menos importantes, se debe determinar las variables (X) y (y) para la posterior construcción de los modelos de predicción. En las siguientes lineas se puede observar la programación realizada en Python.

Código en Python

```
# Definición de las variable X e y.
y=columnas_seleccionadas['IRRADIACIÓN SOLAR GLOBAL PROMEDIO (Wh/m^2)']
X=columnas_seleccionadas.drop(['IRRADIACIÓN SOLAR GLOBAL PROMEDIO
(Wh/m^2)'], axis=1)
y.head()
X.head()
```

En la siguiente **Figura 19** Se puede apreciar en las variables (X) y (y)

	FECHA Y HORA	IRRADIACIÓN SOLAR GLOBAL PROMEDIO (Wh/m^2)
75	2021-01-01 06:10:00	0.083333
76	2021-01-01 06:15:00	0.333333
77	2021-01-01 06:20:00	0.750000
78	2021-01-01 06:25:00	1.250000
79	2021-01-01 06:30:00	1.666667
...	...	...
150064	2021-12-31 18:10:00	2.166667
150065	2021-12-31 18:15:00	1.333333
150066	2021-12-31 18:20:00	0.833333
150067	2021-12-31 18:25:00	0.416667
150068	2021-12-31 18:30:00	0.083333

75720 rows × 2 columns

Figura 19. Conjunto de datos en una variable objetivo (y) y un conjunto de variables independientes (X).

5.4.2. Desarrollar un modelo basado en Machine Learning para la predicción de irradiación solar

Revisión de literatura sobre modelos de Machine Learning para la predicción de la irradiación solar.

- **Identificar la teoría necesaria para el desarrollo de los modelos de predicción.**

Tras realizar una revisión absoluta de la literatura, se identificaron varios modelos de predicción empleando Machine Learning. Los detalles de estos modelos, incluyendo los

nombres de los autores, los tipos de modelos y sus respectivos resultados, se presentan en la **Tabla 2**. Esta información bibliográfica resultará de gran ayuda para enriquecer el contexto de nuestro estudio en este campo.

5.4.2.1 Modelos de Predicción de irradiación solar En la **Tabla 2** se observa diferentes modelos de predicción, haciendo uso de la Inteligencia Artificial (Machine Learning).

Tabla 2. Modelos de predicción, haciendo uso de la Inteligencia Artificial (Machine Learning).

Autor	Modelo	Resultado
1.(Nawab et al., 2023)	Modelos de memoria a corto plazo (LSTM) y de RNA para la predicción de irradiación solar.	La LSTM superó a RNA. Los modelos se evaluaron mediante el coeficiente de determinación (R2), el error cuadrático medio (MSE), el error absoluto medio (MAE), de 0,93, 0,008, 0,17, respectivamente, en la Universidad de Duzce (Turquía).
2.(Gallo et al., 2025)	Redes neuronales para estimar la irradiación solar superficial a partir de imágenes de satélite.	Se obtuvo un coeficiente de determinación (R2) de 0,929, en 16 ubicaciones de Europa, África y Sudamérica.
3.(Mughees et al., 2023)	Autocodificador apilado profundo secuencia a secuencia basado en Bi-LSTM para predicción de la irradiación solar.	Los resultados simulados a partir de datos reales confirmaron que el modelo propuesto superaba las alternativas propuestas alcanzando un R2 de 0.997, lo que evidencia la alta fiabilidad del modelo propuesto.

Continúa en la siguiente página

Tabla 2 – Continuación

Autor	Modelo	Resultado
4.(Zhao et al., 2024)	Predicción a gran escala de la irradiación solar, impactos de sombreado y en la fachada de los edificios mediante indicadores morfológicos urbanos: A enfoque de aprendizaje automático.	Un conjunto de árboles/regresión de bosque aleatorio (RF), haciendo uso de redes neuronales. Los modelos se evaluaron mediante el (R2), el (MSE), el (MAE), de 0,99, 40,083, 5,575, respectivamente.
5.(Faisal et al., 2022)	Predicción multivariante de series temporales de radiación solar basada en Redes Neuronales a partir de datos meteorológicos de distintas ciudades de Bangladesh (Red Neuronal Recurrente (RNN), la Memoria Larga a Corto Plazo (LSTM) y la Unidad Recurrente Cerrada (GRU)).	Haciendo uso de las redes neuronales, desarrollo tres modelos RNN, LSTM y GRU. Al concluir el análisis, el modelo GRU produjo el mejor resultado entre los tres modelos, con una puntuación de MAPE (error porcentual absoluto medio) del 19.28 % en comparación de los valores reales.

Continúa en la siguiente página

Tabla 2 – Continuación

Autor	Modelo	Resultado
6.(Tian et al., 2023)	Predicción multiescala de radiación solar y potencia fotovoltaica con algoritmos de aprendizaje automático en entorno urbano (Modelo RNA (Red Neuronal Artificial) con múltiples capas).	Usando la inteligencia artificial, incluidos los algoritmos de aprendizaje automático (ML), desarrollo tres arquitecturas DL (Deep Learning) (ANN (Redes neuronales artificiales), CNN (Redes neuronales convolucionales) y LSTM (Redes de memoria a largo plazo)) para predecir la irradiancia horizontal global. Los modelos de DL obtuvieron los mejores resultados para la predicción a corto plazo con 1 h y 3 h de antelación, con puntuaciones R2 superiores a 0,90.
7.(Sehrawat et al., 2023)	Modelos de predicción de la irradiancia solar mediante técnicas de aprendizaje automático y gemelo digital (Modelos LR, XGBRF, RF, XGB Boost, CatBoost, DT, GB y LGBM).	Desarrolló un nuevo modelo denominado 8-Stacking Regression Cross Validation (8 STR-CV) para estimar la irradiancia solar. Este enfoque combina ocho modelos de aprendizaje automático, y los resultados de la investigación indican que el XGBoost Regressor superó a todos los demás modelos cuando se empleó como metaregresor o estimador final en este estudio. Se logró una impresionante precisión del 98.8 % en las ubicaciones seleccionadas.

Selección de la arquitectura para los modelos de Machine Learning para el estudio.

- **Red neuronal recurrente.**

Las redes neuronales recurrentes (RNN) pueden tratar eficazmente datos secuenciales. Las RNN indican el tiempo utilizando aristas adyacentes a los pasos temporales. Estas redes neuronales basadas en estados pasados y datos de entrada actuales pueden cambiar el estado actual, lo que se hace a través de una conexión cíclica. Las RNN contienen estados ocultos, que son esencialmente células de memoria. Estas células de memoria conservan su estado en forma vectorial. El estado oculto actual puede calcularse mediante la siguiente ecuación. Donde x_t se denota como la entrada actual, S_t es el estado oculto actual, y S_{t-1} es el anterior.

La red produce una salida en cada paso de tiempo. Mediante un bucle, la salida se devuelve al estado que calcula el siguiente paso temporal inmediato. Como se muestra en la ecuación, el estado oculto en un paso temporal puede calcularse multiplicando la entrada y los estados con matrices de pesos.

$$S_t = \varphi\left(S_{t-1}, X_t\right) \tag{4}$$

La red produce una salida en cada paso temporal. Mediante un bucle, la salida se devuelve al estado, que calcula el siguiente paso temporal inmediato. Como se muestra en la ecuación, el estado oculto en un paso temporal puede calcularse multiplicando la entrada y los estados con matrices de pesos.

$$s_t = \varphi\left(W_{S_{t-1}}, U_{X_t}\right) \tag{5}$$

$$o_t = \Psi\left(V s_t\right) \tag{6}$$

En la siguiente **Figura 20** muestra una representación de una red neuronal recurrente desplegada. La primera configuración de la red neuronal consta de cuatro capas totalmente conectadas.

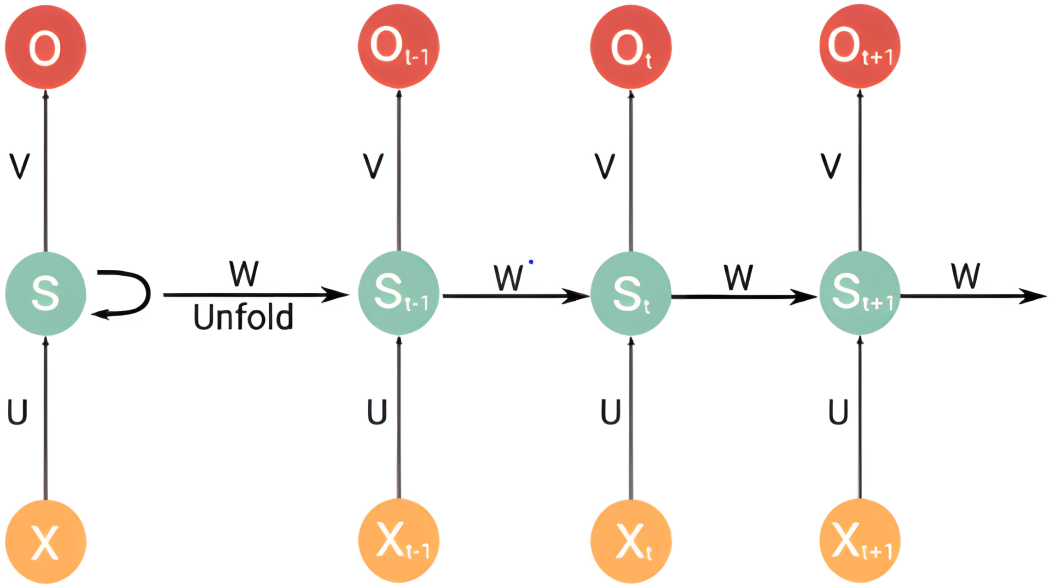

Figura 20. Principio de funcionamiento de una red neuronal recurrente desplegada. Fuente: (Faisal et al., 2022).

Las capas totalmente conectadas son las capas de una red en la que todas las entradas de una capa están conectadas a las unidades de activación de la capa siguiente. La primera capa consta de 128 unidades, la segunda de 64 y la tercera de 32 unidades. Estas tres primeras capas tienen ReLu como función de activación. La última capa consta de sólo 1 unidad y una función lineal como función de activación. Es necesario utilizar un método de regularización con una red RNN para evitar el sobreajuste del modelo. Como método de regularización utilizamos el abandono, que elimina entradas y conexiones probabilísticamente durante el entrenamiento. La tasa de abandono para esta configuración se establece en "0.2". También se utilizó un algoritmo de optimización por descenso de gradiente, Adam.

5.4.3. Memoria a corto plazo

La LSTM aborda un problema muy perceptible con una red RNN, la capacidad de memoria a corto plazo. Las RNN pueden representar información contextual almacenando datos de entrada, pero esto no permite formar dependencias a largo plazo. LSTM promete una solución a este problema. Propuesta por primera vez por Hochreiter y Schmidhuber, la LSTM introduce un mecanismo de compuertas para controlar el flujo de entrada y almacenar estados anteriores para formar una capacidad de memoria a largo plazo. La LSTM supera los problemas de desvanecimiento de gradiente y explosión que se producen en las RNN. A continuación, se presentan las ecuaciones del mecanismo de compuerta.

$$\mathrm{i} = \sigma\left(W_{\mathrm{i}} h_{t-1} + u_{\mathrm{i}} x_{\mathrm{i}}\right) \tag{7}$$

$$f = \sigma\left(W_f h_{t-1} + u_f x_{\mathrm{i}}\right) \tag{8}$$

$$o = \sigma\left(W_o h_{t-1} + u_o x_{\mathrm{i}}\right) \tag{9}$$

$$g = \varphi\left(W_g h_{t-1} + u_g s_t\right) \tag{10}$$

$$C_t = \varphi\left(C_{t-1} \bigotimes f\right) \bigotimes \left(g \bigotimes \mathrm{i}\right) \tag{11}$$

$$h_t = \varphi\left(C_t\right) \bigotimes o \tag{12}$$

Aquí, i representa la puerta de entrada, f para la puerta de olvido y o es la puerta de salida. Además, g es el estado oculto, $c_t$es el estado de acarreo y h_t es la siguiente celda de memoria. W_{i}, W_f, W_o y W_g son los pesos de la puerta correspondiente.

En la **Figura 21** observa la arquitectura básica de una unidad de memoria a corto y largo plazo LSTM (Long Short-Term Memory), sus partes constan de 3 compuertas la cuales son:compuerta de olvido, compuerta de entrada y compuerta de salida. Estas compuertas se utilizan para determinar si se almacenan u olvidan estados anteriores, estructurando así una relación temporal más larga. .Nuestra configuración de red basada en LSTM también consta de cuatro capas.

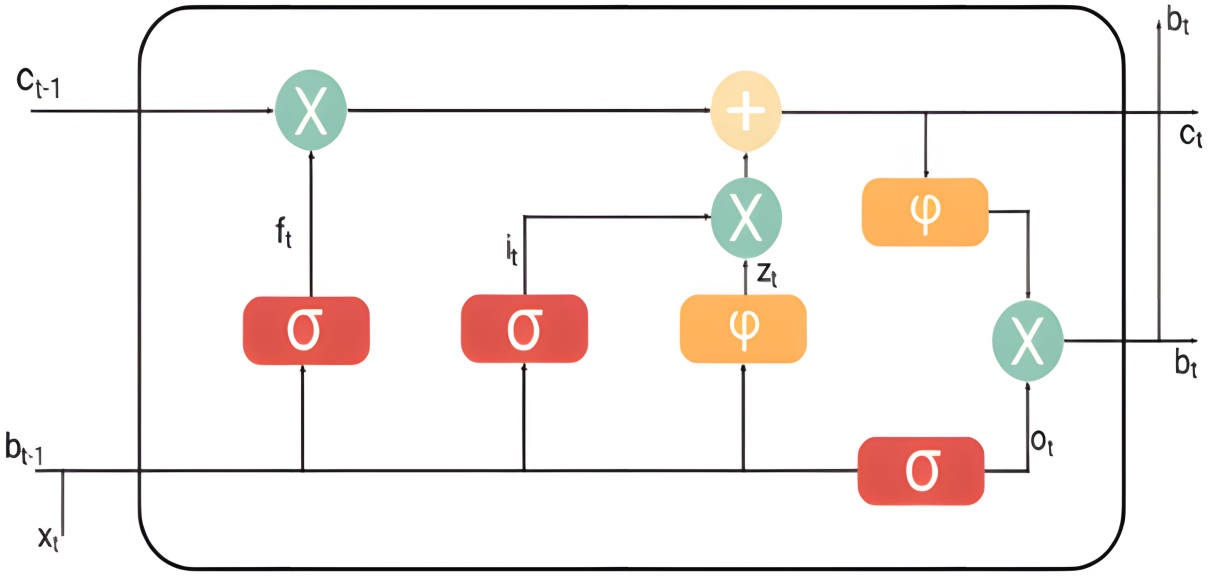

Figura 21. Arquitectura de una unidad de memoria a corto plazo (LSTM).
Fuente: (Faisal et al., 2022).

La primera capa aquí es una capa LSTM de 128 unidades. Esta primera capa está configurada para emitir todos los estados ocultos en lugar de sólo el último estado oculto. La segunda capa consta de 64 unidades LSTM. Las dos últimas capas son capas RNN totalmente conectadas. La tercera capa consta de 16 unidades y la cuarta de 1 unidad. Como las unidades LSTM ya tienen internamente funciones sigmoideas y Tanh, y como se trata de un problema de regresión, decidimos utilizar la función lineal básica como función de activación entre todas las capas. La tasa de abandono para esta configuración se establece en '0.2 Adam, similar a la primera configuración, también se utilizó aquí como un algoritmo de optimización de descenso de gradiente.

5.4.4. Unidad recurrente cerrada

La Unidad Recurrente Cerrada (GRU) fue diseñada por Cho et al. (2014) para resolver el problema de la alta carga computacional que puede ralentizar una red LSTM. Una unidad GRU estándar combina la compuerta de olvido y la compuerta de entrada de una unidad LSTM, por lo que una unidad GRU solo tiene dos compuertas, una compuerta de reinicio y una compuerta de actualización. Esta sencilla arquitectura hace que las GRU sean más rápidas que las LSTM, pero menos potentes en potencia computacional. Por tanto, las redes basadas en GRU son totalmente incompetentes para algunos problemas. Las ecuaciones matemáticas de las GRU son las siguientes.

$$r_t = \sigma\left(W_{zh} h_{t-1} + W_{zx} x_t + b_z\right) \tag{13}$$

$$r_t = \sigma\left(W_{rh} h_{t-1} + W_{rx} x_t + b_r\right) \tag{14}$$

$$\tilde{h_t} = \varphi\left(W_{\tilde{hh}}\left(r_t . h_{t-1}\right) + W_{\tilde{hh}} x_t + b_z\right) \tag{15}$$

$$h_t = \left(1 - z_t\right) . h_{t-1} + z_t . \tilde{ht} \tag{16}$$

Aquí, r_t es la puerta de reinicio, z_t es la puerta de actualización, h_t es la salida, y x_t representa la entrada.

La principal característica de las GRU es que no solo procesan secuencias de manera temporal, sino que también cuentan compuertas que controlan el flujo de información, mejorando la capacidad de la red para aprender dependencias a largo plazo.En la **Figura 22** se puede observar la arquitectura básica de una Unidad Recurrente Cerrada (GRU).

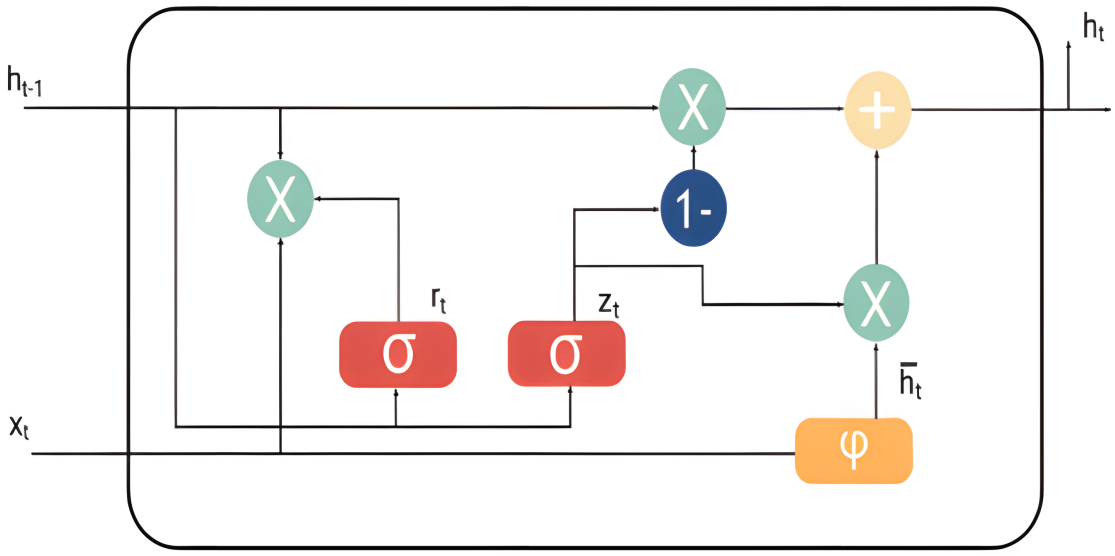

Figura 22. Arquitectura de una unidad recurrente controlada (GRU).
Fuente: (Faisal et al., 2022).

Nuestra red basada en GRU consta de cuatro capas. La primera capa es una capa GRU de 128 unidades, y la segunda capa es una capa GRU de 64 unidades. La primera capa de nuestra red basada en LSTM está configurada para emitir todos los estados ocultos en lugar de sólo el último estado oculto. La tercera capa consta de 32 unidades GRU. Para estas tres primeras capas se ha establecido un dropout de '0,2'. La función de activación de estas tres capas es "tanh". Como capa final para la regresión se utiliza

una capa RNN de 1 unidad totalmente conectada. Adam se utiliza como un algoritmo de optimización de descenso de gradiente, como las redes anteriores (Faisal et al., 2022).

Justificación de la elección de los modelos a desarrollar.

La elección de los modelos de predicción se fundamentó en la naturaleza secuencial de los datos utilizados, donde la **FECHA Y HORA** representan una dimensión temporal clave, y la **IRRADIACIÓN SOLAR GOBAL PROMEDIO** es una variable objetivo con patrones recurrentes a lo largo del tiempo. En este contexto, se seleccionaron tres arquitecturas de redes neuronales recurrentes: una RNN con capas LSTM, un modelo LSTM puro, y un modelo basado en GRU, debido a su capacidad para modelar dependencias temporales y patrones no lineales complejos en series temporales.

El modelo RNN con capas LSTM fue considerado como una arquitectura base mejorada, dado que las celdas LSTM permiten superar las limitaciones clásicas de las RNN simples, como el desvanecimiento del gradiente, y son eficaces para capturar dependencias a largo plazo en datos secuenciales. Por su parte, el modelo LSTM se seleccionó por su probada eficacia en problemas de predicción temporal, donde se requiere memoria de largo plazo para aprender la dinámica del sistema. Finalmente, se incorporó el modelo GRU debido a su arquitectura más simple y eficiente computacionalmente, pero que mantiene un rendimiento comparable al LSTM en muchas tareas de predicción secuencial.

Según (Géron, 2019), tanto LSTM como GRU son extensiones de las redes neuronales recurrentes diseñadas específicamente para manejar secuencias largas de datos sin perder la información relevante, lo que las hace especialmente útiles para tareas como la predicción de series temporales meteorológicas, en las que los datos dependen fuertemente del contexto anterior.

Entrenamiento de los modelos seleccionados y ajustar sus hiperparámetros, buscando optimizar el tiempo de procesamiento y maximizando la precisión.

- **Entrenamiento de cada uno de los modelos seleccionados, cambiando cada uno de los hiperparámetros.**
- **Buscar que cada uno de los modelos desarrollados, se entrenen en el menor tiempo posible.**

El primer modelo de red neuronal recurrente (RNN) utilizando la arquitectura LSTM (Long Short-Term Memory) para predecir la irradiación solar global promedio. Aquí está una explicación detallada de cada parte del código para comenzar se realizó. Estas importaciones preparan el entorno para construir, entrenar y evaluar un modelo de

red neuronal recurrente para la predicción de irradiación solar.

PRIMER MODELO: Red Neuronal RNN.

1. Importar bibliotecas:

Código en Python.

```
# Primer Modelo
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import mean_squared_error, mean_absolute_error
,r2_score
import matplotlib.pyplot as plt
import numpy as np
```

- **import pandas as pd:** La Biblioteca Panda se la importa y se le da un 'pd' de seudónimo, a menudo utilizado para el procesamiento y el análisis de datos de Python.
- **from sklearn.model_selection import train_test_split:** Importar la función de train_test_split desde scikit-learn utilizada para dividir el conjunto de datos en el kit de entrenamiento y prueba.
- **from sklearn.preprocessing import MinMaxScaler:** MinmaxScaler Class Imports de ScikitLearn, que se usa para normalizar los datos, dentro del rango de [0, 1].
- **from tensorflow.keras.models import Sequential:** Se debe importar la clase Keras Sequential para la construcción de modelos de redes neuronales, apilando capas de manera secuencial.
- **from tensorflow.keras.layers import LSTM, Dense, Dropout:** Se realiza la importación de las capas LSTM (Long Short-Term Memory),Dense y Dropout de Keras, que son los componentes importantes para la construcción de modelos en redes neuronales.
- **from tensorflow.keras.optimizers import Adam:** Importar el optimizador Adam desde Keras, ya que es un algoritmo de optimización popular utilizado para ajustar el peso de la red neuronal durante el entrenamiento.
- **from tensorflow.keras.callbacks import EarlyStopping:** Se debe Importar la clase EarlyStopping desde Keras, ya que se debe detener el entrenamiento prematuramente en ausencia a las mejoras en una determinada métrica.

- **from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score:** Importar las funciones métricas de regresión desde Scikit-Learn, que se utilizarán para evaluar el rendimiento del modelo.
- **import Matplotlib.pyplot as plt:** La biblioteca Matplotlib se le da un módulo de visualización 'plt' de seudónimo que se utilizará para crear gráficos y visualizaciones de los mismos.
- **import numpy as np:** La Biblioteca Numpy se le da seudónimo "np", a menudo utilizado para operaciones de matemáticas y la manipulación de eventos para Python.

2. Cargar los datos desde el Excel:

Código en Python.

```
# Cargar datos desde Excel
columnas_seleccionadas = pd.read_excel('Datos 2021.xlsx')
```

- **pd.read_excel('Datos 2021.xlsx'):** Se usa la función read_excel de pandas para cargar datos de un archivo de Excel llamado 'Datos 2021.xlsx'. Esta característica crea un marco de datos, que es la estructura de la tabla de dos dimensiones, donde cada columna puede tener diferente (por ejemplo, fechas, números, texto). Este marco de datos se asigna a la variable columnas_seleccionadas se asigna a este DataFrame, y puede contener los datos presentes en el archivo Excel.

3. Selección de columnas relevantes y preprocesamiento:

Código en Python.

```
columnas_seleccionadas_selected=columnas_seleccionadas[['FECHA Y
HORA','IRRADIACIÓN SOLAR GLOBAL PROMEDIO (Wh/m^2)']]
# Asegurar que la columna de fecha y hora sea de tipo datetime
columnas_seleccionadas_selected['FECHA Y HORA']=pd.to_datetime
(columnas_seleccionadas_selected['FECHA Y HORA'])
# Ordenar por fecha y hora
columnas_seleccionadas_selected=columnas_seleccionadas_selected.
sort_values(by='FECHA Y HORA')
# Normalizar los datos
scaler = MinMaxScaler()
columnas_seleccionadas_selected['IRRADIACIÓN SOLAR GLOBAL PROMEDIO
(Wh/m^2)'] = scaler.fit_transform(columnas_seleccionadas_selected
['IRRADIACIÓN SOLAR GLOBAL PROMEDIO (Wh/m^2)'].values.
reshape(-1,1))
```

- **columnas_seleccionadas_selected = columnas_seleccionadas [['FECHA Y HORA' , 'IRRADIACIÓN SOLAR GLOBAL PROMEDIO ($\mathbf{Wh/m^2}$)']]** : Se escoge las columnas relevantes del DataFrame original (columnas_seleccionadas),seleccionando solo las columnas 'FECHA Y HORA' y

'IRRADIACIÓN SOLAR GLOBAL PROMEDIO (Wh/m^2)'.La nueva variable columnas_seleccionadas_selected almacena este subconjunto de datos.

- **columnas_seleccionadas_selected ['FECHA Y HORA']= pd.to_datetime (columnas_seleccionadas_selected['FECHA Y HORA']):** Convierte la columna **'FECHA Y HORA'** del subconjunto de datos a tipo datetime datetime haciendo uso de la función **pd.to_datetime()**. Esto es útil para trabajar con fechas y horas de manera efectiva en Python.
- **columnas_seleccionadas_selected = columnas_seleccionadas_selected. sort_values (by='FECHA Y HORA'):** Ordena el DataFrame según la columna **'FECHA Y HORA'**, asegurándose de que los datos estén en orden cronológico.
- **scaler = MinMaxScaler():** Crea un objeto MinMaxScaler de scikit-learn, que se utilizará para normalizar los datos. El MinMaxScaler escala los valores al rango [0, 1].
- **columnas_seleccionadas_selected['IRRADIACIÓN SOLAR GLOBAL PROMEDIO(Wh/m^2)'] = scaler.fit_transform(columnas_seleccionadas_selected['IRRADIACIÓN SOLAR GLOBAL PROMEDIO (Wh/m^2)'].values.reshape(-1,1)):** Normaliza la columna 'IRRADIACIÓN SOLAR GLOBAL PROMEDIO (Wh/m^2)' utilizando el MinMaxScaler. La normalización es una técnica común para lograr todo el valor en una escala comparable, en este caso 0 a 1.

4. Creación de secuencias y objetivos:

Código en Python.

```
# Definir la longitud de la secuencia temporal
seq_length = 10
# Crear secuencias y objetivos
sequences = []
targets = []
for i in range(len(columnas_seleccionadas_selected) - seq_length):
seq = columnas_seleccionadas_selected['IRRADIACIÓN SOLAR GLOBAL
PROMEDIO (Wh/m^2)'].iloc[i:i + seq_length].values
target = columnas_seleccionadas_selected['IRRADIACIÓN SOLAR GLOBAL
PROMEDIO (Wh/m^2)'].iloc[i + seq_length]
sequences.append(seq)
targets.append(target)
X = np.array(sequences)
y = np.array(targets)
```

- **seq_length = 10:** Define la longitud del orden temporal. En este caso, son 10, lo que significa que habrá 10 elementos de la serie temporal en cada orden.
- **sequences = [ ] y targets = [ ]:** Inicializan listas vacías para guardar las

secuencias y los objetivos.

- **El bucle for i in range(len(columnas_seleccionadas_selected)-seq_length):** itera a través de los índices de la serie temporal menos la longitud de la secuencia. Esto asegura que haya suficientes elementos en la serie temporal para formar secuencias de la longitud especificada.
- **seq = columnas_seleccionadas_selected['IRRADIACIÓN SOLAR GLOBAL PROMEDIO (Wh/m^2)'].iloc[i:i+seq_length].values:** Crea una secuencia tomando los valores de la columna 'IRRADIACIÓN SOLAR GLOBAL PROMEDIO (Wh/m^2)' desde el índice i hasta i + seq_length. Estos valores se extraen como un array de NumPy.
- **target = columnas_seleccionadas_selected['IRRADIACIÓN SOLAR GLOBAL PROMEDIO (Wh/m^2)'].iloc[i + seq_length]:**Concreta el objetivo como el valor en la posición siguiente a la secuencia actual.
- **sequences.append(seq) y targets.append(target):** Añaden la secuencia y el objetivo a las listas correspondientes.
- **X = np.array(sequences):** Convierte las secuencias en un array de NumPy, que será utilizado como entrada para el modelo.
- **y = np.array(targets):** Convierte los objetivos en un array de NumPy, que será la salida deseada para el modelo.

5. División de datos en conjuntos de entrenamiento, validación y prueba:

Código en Python.

```
# Dividir los datos en conjuntos de entrenamiento, validación y
prueba
X_train,X_temp,y_train,y_temp=train_test_split(X,y,test_size=0.4,
random_state=42)
X_val,X_test,y_val,y_test=train_test_split(X_temp,y_temp,test_size
=0.5, random_state=42)
```

- **X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.4, random_state=42):** Hace uso de la función train_test_split desde la biblioteca scikit-learnpara luego dividir un conjunto de datos, en conjuntos de entranmiento (train) y temporal (temp). La entrada X son las secuencias y y son los objetivos. Otro parámetro test_size establece en 0.4, significa que el 40 % de los datos serán asignados al conjunto temporal y el 60 % restante se lo asignará al conjunto de entrenamiento. El parámetro random_state , a sido establecido en 42 para asegurar reproducibilidad, por ejemplo, si ejecutas el código varias veces, obtendrás la misma partición de datos.
- **X_val,X_test,y_val,y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42):** Se utiliza nuevamente la función train_-

test_split , para la división del conjunto temporal (temp) en conjuntos de validación (val) y prueba (test). El otro parámetro test_size se estableció en 0.5, lo que representa la mitdad de los datos temporales se asignarán al conjunto de prueba y la otra mitad al conjunto de validación. El parámetro random_state lo dejamos en 42 para asegurar la consistencia.

6. Reshape para coincidir con la entrada de la red:

Código en Python.

```
# Reshape para que coincida con la entrada de la red
X_train = X_train.reshape(-1, seq_length, 1)
X_val = X_val.reshape(-1, seq_length, 1)
X_test = X_test.reshape(-1, seq_length, 1)
```

- **X_train = X_train.reshape(-1, seq_length, 1):**Damos uso al método reshape de Numpy para cambiar la forma de los datos de entrenamiento (X_train)). La nueva forma se da en (número de muestras, longitud de la secuencia ,1). Esto es necesario para que los datos se ajusten a la entrada de la red neuronal, donde cada muestra reside en secuencias de longitud seq_length y una sola característica.
- **X_val = X_val.reshape(-1, seq_length, 1):**Igual al paso anterior, cambia la forma de los datos de validación (X_val) para que concuerden con la estructura requerida por la red neuronal.
- **X_test = X_test.reshape(-1, seq_length, 1):**Al igual que los dos pasos anteriores, cambia la forma de los datos de prueba (X_test) para que se acomoden a la estructura necesaria para la entrada en la red neuronal.
- Estas líneas de código nos aseguran que los datos de entrada estén ordenados en lotes de secuencias de longitud **seq_length**, y con una sola característica en cada paso de tiempo, preparándolos para ser utilizados como entrada en una red neuronal recurrente (LSTM en este caso).

7. Construcción del modelo RNN con capas LSTM:

Código en Python.

```
# Construir el modelo RNN con capas LSTM y capas totalmente
    conectadas
model = Sequential()
model.add(LSTM(units=128, activation='relu',
input_shape=(seq_length,1), return_sequences=True))
model.add(Dropout(0.15))
model.add(LSTM(units=64, activation='relu', return_sequences=True))
model.add(Dropout(0.15))
model.add(LSTM(units=32, activation='relu'))
model.add(Dropout(0.15))
model.add(Dense(units=64, activation='relu'))
```

```
model.add(Dropout(0.15))
model.add(Dense(units=1, activation='linear'))
```

- **model = Sequential():** Comienza un modelo secuencial de Keras, que es una pila lineal de capas.
- **model.add(LSTM(units=128, activation='relu', input_shape=(seq_length, 1), return_sequences=True)):** Agrega una capa LSTM con 128 unidades, función de activación 'relu', y una forma de entrada detallada por **input_shape=(seq_length, 1)**. La opción **return_sequences=True** indica que la capa debe devolver secuencias en lugar de un solo vector de salida.
- **model.add(Dropout(0.15)):** Aumenta una capa de Dropout con una tasa del 15 % para ayudar a prevenir el sobreajuste.
- **model.add(LSTM(units = 64, activation = 'relu', return_sequences = True)):** Aumenta otra capa LSTM con 64 unidades, función de activación 'relu', y **return_sequences=True**.
- **model.add(Dropout(0.15)):** Aumenta otra capa de Dropout con una tasa del 15 %.
- **model.add(LSTM(units=32, activation='relu')):** Aumenta una tercera capa LSTM con 32 unidades y función de activación 'relu'. Como no se especifica **return_sequences=True**, esta capa no devolverá secuencias.
- **model.add(Dropout(0.15)):** Aumenta otra capa de Dropout con una tasa del 15 %.
- **model.add(Dense(units=64, activation='relu')):** Aumenta una capa totalmente conectada (Dense) con 64 unidades y función de activación 'relu'.
- **model.add(Dropout(0.15)):** Aumenta otra capa de Dropout con una tasa del 15 %.
- **model.add(Dense(units=1, activation='linear')):** Aumenta la capa de salida con una sola unidad y función de activación 'linear' para regresión, ya que se está prediciendo un valor numérico continuo.

8. Compilación del modelo:

Código en Python.

```
# Compilar el modelo
model.compile(optimizer=Adam(learning_rate=0.0001),loss=
'mean_squared_error')
```

- **model.compile(optimizer = Adam(learning_rate = 0.0001), loss = 'mean_squared_error'):** Compila el modelo con la configuración detallada.
- **optimizer = Adam(learning_rate = 0.0001):** Escoge el optimizador a usar durante el entrenamiento. En este caso, se utiliza el optimizador Adam

con una tasa de aprendizaje (learning rate) de 0.0001. Adam es un algoritmo de optimización popular utilizado para ajustar los pesos de la red durante el entrenamiento.

- **loss = 'mean_squared_error':** Define la función de pérdida a minimizar durante el entrenamiento. En este caso, se utiliza el error cuadrático medio (mean squared error), que es comúnmente utilizado en problemas de regresión cuando se está prediciendo un valor numérico continuo. El modelo ajustará sus pesos para minimizar esta función de pérdida durante el entrenamiento.
- Compilar el modelo es un paso muy importante antes de entrenarlo, ya que configura aspectos como el optimizador y la función de pérdida. Despues, el modelo está listo para ser entrenado utilizando datos de entrada (X_train) y objetivos (y_train).

9. Definir el Callback EarlyStopping:

Código en Python.

```
# Definir el callback EarlyStopping para detener el entrenamiento
    si la pérdida en el conjunto de validación deja de disminuir
early_stopping = EarlyStopping(monitor='val_loss', patience=15,
restore_best_weights=True)
```

- **early_stopping = EarlyStopping(monitor = 'val_loss', patience = 15,restore_best_weights = True):** Crea una solicitud de la clase EarlyStopping con los siguientes parámetros:
- **monitor = 'val_loss':** Detalla la métrica que se debe monitorear para tomar decisiones sobre si se debe parar el entrenamiento o no. En este caso, la métrica es la pérdida en el conjunto de validación (val_loss).
- **patience = 15:** Representa la cantidad de épocas (iteraciones completas a través de los datos) durante las cuales la métrica monitorizada (val_loss) puede no mejorar antes de detener el entrenamiento. Para este caso, el entrenamiento se detendrá si la pérdida en el conjunto de validación no mejora durante 15 épocas consecutivas.
- **restore_best_weights=True:** Enseña que se deben restaurar los pesos del modelo a los mejores obtenidos durante el entrenamiento cuando se detiene debido a la falta de mejora. Esto evita que el modelo se sobreajuste a los datos de entrenamiento y a tener una versión del modelo que generaliza bien a nuevos datos.
- El EarlyStopping es un mecanismo utilizado para evitar el sobreajuste y detener el entrenamiento cuando el rendimiento del modelo en el conjunto de validación deje de mejorar. Esto ayuda a prevenir que el modelo continúe ajustándose a ruido en los datos y permite conservar una versión del modelo que

generaliza bien a datos no vistos. Este callback se suele utilizar en conjunto con el entrenamiento de modelos de aprendizaje profundo para mejorar su capacidad de generalización.

10. Entrenamiento del Modelo:

Código en Python.

```
# Entrenar el modelo y almacenar el historial de entrenamiento
history=model.fit(X_train,y_train,epochs=100,batch_size=168,
validation_data=(X_val, y_val), callbacks=[early_stopping])
```

- **history = model.fit(X_train,y_train, epochs = 100,batch_size = 168,validation_data = (X_val, y_val), callbacks = [early_stopping]):** Comienza el proceso de entrenamiento del modelo.
- **X_train, y_train:** Son los datos de entrada (secuencias) y los objetivos (etiquetas) del conjunto de entrenamiento, proporcionalmente.
- **epochs = 100:** Detalla la cantidad de épocas (iteraciones completas a través de los datos de entrenamiento) que el modelo se entrenará.
- **batch_size = 168:** Define el tamaño del lote (batch) de datos utilizado en cada paso del entrenamiento. Para esto se utilizan 168 muestras en cada paso. El tamaño del lote influye en la eficiencia del entrenamiento y puede afectar la memoria requerida.
- **validation_data = (X_val, y_val):** Facilita datos de validación durante el entrenamiento. El modelo evaluará su rendimiento en este conjunto después de cada época.
- **callbacks = [early_stopping]:** Hace uso del callback de EarlyStopping previamente definido. Este callback monitorizará la pérdida en el conjunto de validación (val_loss) y pondrá fin al entrenamiento si la pérdida no mejora durante un número detallado de épocas (según la configuración de patience).
- **history:** Guarda el historial del entrenamiento, que contiene la información sobre la pérdida y la métrica (si se especifica) en cada época tanto para el conjunto de entrenamiento como para el conjunto de validación. Se puede utilizar este historial para visualizar cómo evolucionó el rendimiento del modelo durante el entrenamiento.
- Para el segundo y tercer modelo se realiza el mismo procedimiento, desde el principio lo que cambia es a partir de la construcción del modelo, a continuación, se describen los modelos dos y tres.

SEGUNDO MODELO: LSTM

1. Construcción del modelo LSTM:

Código en Python

```
# Construir el modelo LSTM
model = Sequential()
model.add(LSTM(units=128,activation='relu',input_shape=
(seq_length,1),return_sequences=True))
model.add(Dropout(0.1))
model.add(LSTM(units=64, activation='relu', return_sequences=True))
model.add(Dropout(0.1))
model.add(LSTM(units=32, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(units=64, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(units=1, activation='linear'))
```

- **model = Sequential():** Comienza un modelo secuencial de Keras, que es una pila lineal de capas.
- **model.add(LSTM(units = 128, activation = 'relu', input_shape = (seq_length, 1), return_sequences = True)):** Agrega una capa LSTM con 128 unidades, con la función de activación 'relu', y una forma de entrada especificada por input_shape = (seq_length,1). La opción return_sequences=True muestra que la capa debe devolver secuencias en lugar de un solo vector de salida.
- **model.add(Dropout(0.1)):** Aumenta una capa de Dropout con una tasa del 10 % para ayudar a prevenir el sobreajuste.
- **model.add(LSTM(units = 64, activation = 'relu', return_sequences = True)):** Aumenta otra capa LSTM con 64 unidades, función de activación 'relu', y return_sequences = True.
- **model.add(Dropout(0.1)):** Aumenta otra capa de Dropout con una tasa del 10 %.
- **model.add(LSTM(units = 32, activation = 'relu')):** Aumenta una tercera capa LSTM con 32 unidades y función de activación 'relu'. Como no se especifica return_sequences = True, esta capa no devolverá secuencias.
- **model.add(Dropout(0.1)):** Aumenta otra capa de Dropout con una tasa del 10 %.
- **model.add(Dense(units = 64, activation = 'relu')):** Aumenta una capa totalmente conectada (Dense) con 64 unidades y función de activación 'relu'.
- **model.add(Dropout(0.1)):** Aumenta otra capa de Dropout con una tasa del 10 %.
- **model.add(Dense(units = 1, activation = 'linear')):** Aumenta la capa de salida con una sola unidad y función de activación 'linear' para regresión,

ya que se está prediciendo un valor numérico continuo.

2. Compilación del modelo:

Código en Python.

```
# Compilar el modelo
model.compile(optimizer=Adam(learning_rate=0.0001),
loss='mean_squared_error')
```

- **model.compile:** La función configura el modelo para el entrenamiento. Detalla la función de pérdida, el optimizador y las métricas a utilizar durante el entrenamiento y la evaluación.
- **optimizer=Adam(learning_rate=0.0001):** Se detalla el optimizador a usar. En este caso, se utiliza el optimizador Adam con una tasa de aprendizaje (learning rate) de 0.0001. Adam es un optimizador tradicional en el aprendizaje profundo.
- **loss='mean_squared_error':** Se detalla la función de pérdida a optimizar durante el entrenamiento. Para este caso, se utiliza la pérdida cuadrática media (mean squared error), que es común en problemas de regresión, donde se busca minimizar la diferencia cuadrática entre las predicciones y los valores reales.

3. Entrenamiento del Modelo.

Código en Python.

```
# Entrenar el modelo
history = model.fit(X_train, y_train, epochs=120, batch_size=150,
validation_data=(X_val, y_val), verbose=1)
```

- **history = model.fit(X_train, y_train, epochs=120, batch_size=150, validation_data=(X_val, y_val), verbose=1):** Comienza el proceso de entrenamiento del modelo.
- **X_train, y_train:** Estos son los datos de entrada (secuencias) y los objetivos (etiquetas) del conjunto de entrenamiento, respectivamente.
- **epochs=120:** Detalla la cantidad de épocas (iteraciones completas a través de los datos de entrenamiento) que el modelo se entrenará.
- **batch_size=150:** Explica el tamaño del lote (batch) de datos utilizado en cada paso del entrenamiento. Para este caso, se utilizan 150 muestras en cada paso. El tamaño del lote influye en la eficiencia del entrenamiento y puede afectar la memoria requerida.
- **validation_data=(X_val, y_val):** Facilita los datos de validación durante el entrenamiento. El modelo evaluará su rendimiento en este conjunto después de cada época.

- **verbose=1:** Controla la cantidad de información que se muestra durante el entrenamiento. Un valor de 1 significa que indicara una barra de progreso con información sobre la pérdida y la métrica en cada época.
- **history:** Guarda el historial del entrenamiento, que contiene información sobre la pérdida y la métrica en cada época tanto para el conjunto de entrenamiento como para el conjunto de validación. Se puede utilizar este historial para visualizar cómo evolucionó el rendimiento del modelo durante el entrenamiento.

TERCER MODELO: GRU

1. Construir el modelo GRU:

Código en Python

```
# Construir el modelo GRU
model = Sequential()
model.add(GRU(units=128, input_shape=(seq_length, 1),
return_sequences=True))
model.add(Dropout(0.1))
model.add(GRU(units=64, return_sequences=True))
model.add(Dropout(0.1))
model.add(GRU(units=32, return_sequences=True))
model.add(Dropout(0.1))
model.add(GRU(units=16))
model.add(Dropout(0.1))
model.add(Dense(units=1))
```

- **model = Sequential():** Comienza un modelo secuencial de Keras, que es una pila lineal de capas.
- **model.add(GRU(units = 128,input_shape = (seq_length,1),return_sequences=True)):** Agrega una capa GRU con 128 unidades y una forma de entrada especificada por **input_shape=(seq_length,1)**. La opción **return_sequences=True** indica que la capa tienes que devolver secuencias en lugar de un solo vector de salida.
- **model.add(Dropout(0.1)):** Aumenta una capa de Dropout con una tasa del 10 % para ayudar a prevenir el sobreajuste.
- **model.add(GRU(units = 64, return_sequences=True)):** Aumenta otra capa GRU con 64 unidades y **return_sequences=True**.
- **model.add(Dropout(0.1)):** Aumenta otra capa de Dropout con una tasa del 10 %.
- **model.add(GRU(units=32, return_sequences=True)):**Aumenta una tercera capa GRU con 32 unidades y **return_sequences=True**.
- **model.add(Dropout(0.1)):**Aumenta otra capa de Dropout con una tasa del

10 %.

- **model.add(GRU(units=16)):**Aumenta una cuarta capa GRU con 16 unidades. Como no se especifica **return_sequences=True**, esta capa no devolverá secuencias.
- **model.add(Dropout(0.1)):**Aumenta otra capa de Dropout con una tasa del 10 %.
- **model.add(Dense(units=1)):**Aumenta la capa de salida con una sola unidad, ya que se está prediciendo un valor numérico continuo. No se da una función especifica de activación, lo que implica una activación lineal por defecto.

2. Compilación del modelo:

Código en Python

```
# Compilar el modelo
model.compile(optimizer=Adam(learning_rate=0.001),
loss='mean_squared_error')
```

- **model.compile(optimizer = Adam(learning_rate = 0.001), loss = 'mean_squared_error'):** Compila el modelo con la configuración especificada.
- **optimizer = Adam(learning_rate = 0.001):** Se elige el optimizador a utilizar durante el entrenamiento. Para este caso, se utiliza el optimizador Adam con una tasa de aprendizaje (learning rate) de 0.001. Adam es un algoritmo de optimización muy utilizado para ajustar los pesos de la red durante el entrenamiento. El learning rate controla el tamaño de los pasos de actualización de los pesos.
- **loss ='mean_squared_error':** Detalla la función de pérdida a minimizar durante el entrenamiento. En este caso, se utiliza el error cuadrático medio (mean squared error), que es comúnmente utilizado en problemas de regresión cuando se está prediciendo un valor numérico continuo. El modelo ajustará sus pesos para minimizar esta función de pérdida durante el entrenamiento.

3. Definir el Callback EarlyStopping:

Código en Python

```
# Definir el callback EarlyStopping para detener el entrenamiento
si la pérdida en el conjunto de validación deja de disminuir
early_stopping=EarlyStopping(monitor='val_loss',patience=15,
restore_ best_weights=True)
```

- **early_stopping:** Se crea un objeto de la clase **EarlyStopping**, que es un callback en Keras. Los callbacks son funciones que se ejecutan durante el entrenamiento en diferentes puntos, como al final de cada época.

- **monitor='val_loss':** Se detalla que el monitoreo para detener el entrenamiento se ejecutara según la pérdida en el conjunto de validación (val_loss). La optimización se detendrá si la pérdida en el conjunto de validación no mejora.
- **patience=15:** Detalla la cantidad de épocas durante las cuales se espera que la pérdida en el conjunto de validación no mejore antes de detener el entrenamiento. En este caso, se interrumpirá después de 15 épocas sin mejora.
- **restore_best_weights=True:** Enseña que se deben restaurar los pesos del modelo al mejor punto durante el entrenamiento cuando se detiene. Esto garantiza que el modelo final tenga los pesos correspondientes al mínimo de pérdida en el conjunto de validación.

4. Entrenamiento del Modelo:

Código en Python

```
# Entrenar el modelo
history = model.fit(X_train, y_train, epochs=100, batch_size=84,
validation_data=(X_val, y_val), callbacks=[early_stopping])
```

- **history = model.fit(X_train,y_train, epochs = 100,batch_size = 84,validation_data = (X_val, y_val), callbacks = [early_stopping]):** Comienza el proceso de entrenamiento del modelo.
- **X_train, y_train:** Estos son los datos de entrada (secuencias) y los objetivos (etiquetas) del conjunto de entrenamiento, respectivamente.
- **epochs = 100:** Detalla la cantidad de épocas (iteraciones completas a través de los datos de entrenamiento) que el modelo se entrenará.
- **batch_size = 84:** Detalla el tamaño del lote (batch) de datos utilizado en cada paso del entrenamiento. Para este caso, se utilizan 84 muestras en cada paso. El tamaño del lote influye en la eficiencia del entrenamiento y puede afectar la memoria requerida.
- **validation_data = (X_val, y_val):** Facilita los datos de validación durante el entrenamiento. El modelo evaluará su rendimiento en este conjunto después de cada época.
- **callbacks = [early_stopping]:** Hace uso del callback de EarlyStopping previamente definido. Este callback monitorizará la pérdida en el conjunto de validación (val_loss) y finalizará el entrenamiento si la pérdida no mejora durante un número especificado de épocas (según la configuración de patience).
- **history:** Guarda el historial del entrenamiento, que contiene información sobre la pérdida y la métrica (si se especifica) en cada época tanto para el conjunto de entrenamiento como para el conjunto de validación. Se puede utilizar este

historial para visualizar cómo evolucionó el rendimiento del modelo durante el entrenamiento.

5.4.5. Evaluar el desempeño del modelo de predicción de radiación utilizando métricas usualmente empleadas en Machine Learning.

Búsqueda bibliográfica de las métricas de evaluación para un modelo de predicción:

- **El error absoluto medio (MAE: Mean absolute error).**

Mide qué tan cerca están las predicciones de los resultados reales; por lo tanto, un menor puntaje es mejor. Se calcula como la sumatoria de valor absoluto de la diferencia entre el valor absoluto y estimado, multiplicado por la inversa del tamaño de la muestra, todo eso se lo realiza con la **Ecuación 17**.

$$MAE = \frac{1}{n}\sum_{\mathrm{i}=1}^{n}|y_{\mathrm{i}} - \hat{y}_l| \tag{17}$$

Donde:

n = Tamaño de la muestra,

y_{i} = Valor real,

$\hat{y}_l$ = Valor estimado.

- **El error cuadrático medio (MSE: Mean Squared Error).**

Crea un valor único que resume el error en el modelo. Mide la media de las diferencias al cuadrado entre los valores reales y los estimados. Véase **Ecuación 18**.

$$MSE = \frac{1}{n}\sum_{\mathrm{i}=1}^{n}(y_{\mathrm{i}} - \hat{y}_{\mathrm{i}})^2 \tag{18}$$

Donde:

n = Tamaño de la muestra,

y_{i} = Valor real,

$\hat{y}_{\mathrm{i}}$ = Valor estimado.

- **El error absoluto relativo (RAE: Relative absolute error).**

Es una diferencia absoluta relativa entre los valores esperados y reales; relativo porque la diferencia de medias se divide por la media aritmética. Se lo realiza por medio de la **Ecuación 19**.

$$RAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_l|}{\sum_{i=1}^{n} |\bar{y} - y_i|} \tag{19}$$

Donde:

n = Tamaño de la muestra,

y_i = Valor real,

$\hat{y}_l$ = Valor estimado,

$\bar{y}$ = Promedio valor real.

- **El error al cuadrado relativo (RSE: Relative squared error).**

Normaliza de manera similar el error al cuadrado total de los valores predichos por dividiendo por el error cuadrado total de los valores reales. Véase **Ecuación 20**.

$$RSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_l)^2}{\sum_{i=1}^{n} (\bar{y} - y_i)^2} \tag{20}$$

Donde:

n = Tamaño de la muestra,

y_i = Valor real,

$\hat{y}_l$ = Valor estimado,

$\bar{y}$ = Promedio valor real.

- **El coeficiente de determinación (R2).**

A menudo denominado R2, representa el poder predictivo del modelo como valor entre 0 y 1. Cero significa que el modelo es aleatorio (no explica nada); 1 significa que hay un ajuste perfecto. Sin embargo, se debe tener precaución al interpretar los valores de R, ya que los valores bajos pueden ser totalmente normales y altos. Los valores pueden ser sospechosos. Se los puede realizar por medio las **Ecuación 21** y **Ecuación 22**.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_l)^2}{\sum_{i=1}^{n} (\bar{y} - y_i)^2} \tag{21}$$

$$R^2 = 1 - \frac{MSE_P}{MSE_M} \tag{22}$$

Donde:

n = Tamaño de la muestra,

y_i = Valor real,

$\hat{y}_l$ = Valor estimado,

$\bar{y}$ = Promedio valor real,

MSE_P = Error cuadrático medio de los valores estimados,

MSE_M = Error cuadrático medio de la muestra.

- **Tipos de Predicciones.**

En la **Figura 23** podemos observar los distintos valores para la evaluación de los distintos métodos, en la denominada matriz de confusión.

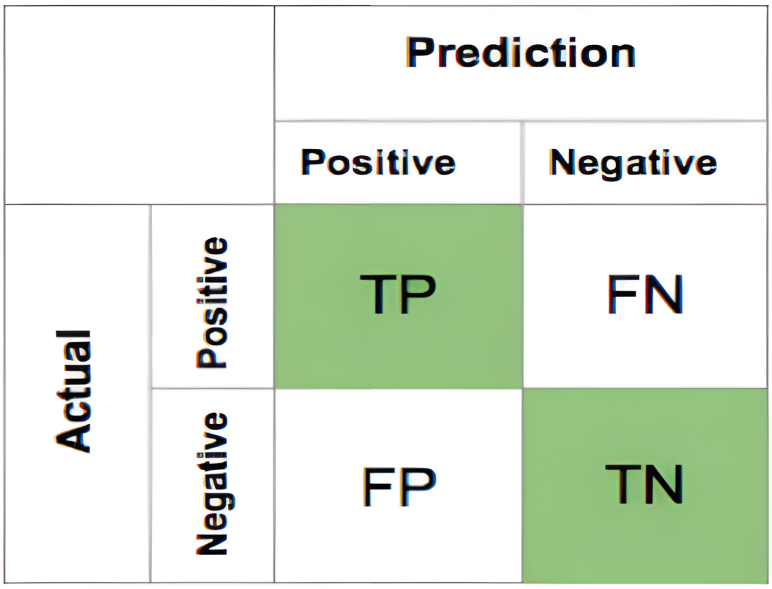

Figura 23. Valores para la evaluación de los distintos métodos.
Fuente: (Alfadda et al., 2018).

- Verdadero Positivo o “True Positive” (TP): ejemplo positivo, predicho positivo.
- Falso Positivo o “False Positive” (FP): ejemplo negativo, predicho positivo.
- Verdadero Negativo o “True Negative” (TN): ejemplo negativo, predicho negativo.
- Falso Negativo o “False Negative” (FN): ejemplo positivo, predicho negativo.

- **Accurary o Exactitud.**

Es una de las métricas más utilizadas, la cual mide el porcentaje cuando una clase pronosticada no coincide con la clase esperada, se la pueda calcular mediante la siguiente **Ecuación 23**.

$$\text{Accuracy} = \frac{\text{Predicciones Correctas}}{\text{Total de Predicciones}} = \frac{\text{TP} + \text{TN}}{TP + FP + FN + TN} \tag{23}$$

- **Recall, Sensibilidad.**

Es el número de elementos identificados correctamente como positivos del total de positivos verdaderos. Se la puede realizar por medio de la siguiente **Ecuación 24**.

$$\text{Recall} = \frac{\text{TP}}{TP + FN} \tag{24}$$

- **F1 Score.**

Esta es otra métrica muy empleada porque nos resume la precisión y sensibilidad en una sola métrica. Por ello es de gran utilidad cuando la distribución de las clases es desigual, por ejemplo, cuando el número de pacientes con una condición es del 15 % y el otro es 85 %, lo que en el campo de la salud es bastante común. Véase **Ecuación 25**.

$$\text{PuntajeF1} = \frac{2 * \text{Precision} * \text{Sensibilidad}}{Precision + Sensibilidad} \tag{25}$$

- **Curva ROC.**

La curva ROC (Curva Característica Operativa Receptor) es un gráfico que resume el rendimiento de los modelos sobre la clase positiva, el eje X indica la tasa de falsos positivos y el eje e indica la tasa positivos verdaderos, la **Figura 24** se define en el intervalo de 0 a 1, en la se logra observar cómo trabaja la curva ROC.

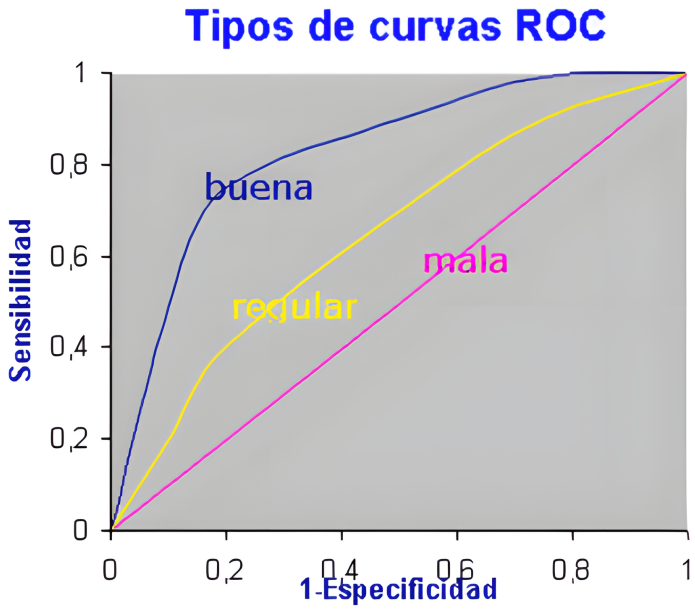

Figura 24. Tipos de Curvas ROC.

Fuente: (Alfadda et al., 2018).

- **Elección de las métricas de evaluación entre todas las consultadas:**

Para poder evaluar el desempeño de los modelos de predicción realizados los cuales son: **RNN** con capas **LSTM**, LSTM puro y **GRU** se escogieron tres métricas ampliamente reconocidas: el Error Absoluto Medio **(MAE)**, el Error Cuadrático Medio **(MSE)** y el Coeficiente de Determinación **(R^2)**. La elección de estas métricas se basa en su capacidad para brindar una visión complementaria y robusta del rendimiento de modelos de regresión sobre series temporales.

El **MAE** permite medir el promedio de los errores absolutos entre los valores predichos y los reales, proporcionando una interpretación directa en las mismas unidades que la variable de salida. Por su parte, el **MSE** penaliza de forma más severa los errores grandes, ya que eleva al cuadrado las diferencias, siendo útil para identificar desviaciones significativas que puedan afectar la precisión del modelo. Finalmente, el **R^2** evalúa la proporción de la varianza de la variable dependiente que es explicada por el modelo, brindando una métrica estandarizada para comparar el ajuste del modelo a los datos.

Según (Kuhn & Johnson, 2013), una evaluación completa de modelos predictivos pide considerar múltiples métricas que no solo cuantifiquen el error, sino que también permitan interpretar qué tan bien el modelo captura la estructura subyacente de los datos. Por ello, el uso conjunto de **MAE**, **MSE** y **R^2** brinda un enfoque equilibrado entre simplicidad interpretativa, sensibilidad a errores y calidad de ajuste.

- **El error absoluto medio (MAE: Mean absolute error).**

La primera métrica la apreciamos en la **Ecuación 26**.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_l| \tag{26}$$

- **El error cuadrático medio (MSE: Mean Squared Error).**

La segunda métrica la apreciamos en la **Ecuación 27**.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{27}$$

- **El coeficiente de determinación (R2).**

La tercera métrica la apreciamos en la **Ecuación 28** y **Ecuación 29**.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_l)^2}{\sum_{i=1}^{n}(\bar{y} - y_i)^2} \tag{28}$$

$$R^2 = 1 - \frac{MSE_P}{MSE_M} \tag{29}$$

- **Programación de las métricas de evaluación en Python:**

Luego de las métricas seleccionadas es necesario realizar la programación en Python para cada uno de los modelos desarrollados, los pasos detallados podemos observarlos a continuación:

PRIMER MODELO: Red Neuronal RNN.

1. Evaluar el modelo en el conjunto de prueba.

Código en Python

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

- **model.evaluate(X_test, y_test):** Evalúa el rendimiento del modelo en el conjunto de prueba. Calcula la pérdida del modelo en los datos de prueba.
- **loss = ...:** Guarda el valor de la pérdida obtenida en la evaluación en la variable **loss**.
- **print(f'Loss en el conjunto de prueba: loss'):** Imprime en la consola el valor de la pérdida en el conjunto de prueba. La f-string **(f'...')** se usa para formatear la cadena e incrustar el valor de **loss** en ella.

2. Realizar Predicciones.

Código en Python

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

- **predictions = model.predict(X_test):** Usa el modelo (model) para realizar predicciones en el conjunto de prueba (X_test). El resultado se guarda en la variable predictions, que contendrá las predicciones del modelo para cada muestra en el conjunto de prueba.

3. Desnormalizar las predicciones.

Código en Python

```
# Desnormalizar las predicciones y los objetivos para su evaluación
predictions_denormalized = scaler.inverse_transform(predictions)
y_test_denormalized = scaler.inverse_transform(y_test.reshape(-1,
1))
```

- **predictions_denormalized = scaler.inverse_transform(predictions):** Usa el objeto **scaler** (previamente utilizado para normalizar los datos) para revertir la normalización aplicada a las predicciones (**predictions**). Esto es transcendental para tener las predicciones en la escala única de la variable de interés.
- **y_test_denormalized = scaler.inverse_transform(y_test.reshape(-1, 1)):** Ejecuta un proceso semejante para desnormalizar los objetivos del conjunto de prueba (**y_test**). Aquí, **y_test** se remodela a una matriz de una sola columna antes de emplear la inversa de la transformación de escala.

4. Calcular y mostrar métricas de evaluación.

Código en Python

```
# Calcular y mostrar métricas de evaluación
mse=mean_squared_error(y_test_denormalized,predictions_denormalized)
mae=mean_absolute_error(y_test_denormalized,predictions_denormalized)
r2=r2_score(y_test_denormalized,predictions_denormalized)
# Mostrar métricas de evaluación
print(f'Mean Squared Error (MSE): {mse}')
print(f'Mean Absolute Error (MAE): {mae}')
print(f'R-squared (R2): {r2}')
```

- **mse = mean_squared_error(y_test_denormalized,predictions_denormalized):** Calcula el Error Cuadrático Medio (MSE) entre las predicciones desnormalizadas (predictions_denormalized) y los valores reales desnormalizados del conjunto de prueba (y_test_denormalized).

- **mae = mean_absolute_error(y_test_denormalized,predictions_denormalized):** Calcula el Error Absoluto Medio (MAE) entre las predicciones desnormalizadas y los valores reales desnormalizados del conjunto de prueba.
- **r2 = r2_score(y_test_denormalized, predictions_denormalized):** Calcula el coeficiente de determinación R-cuadrado (R^2) entre las predicciones desnormalizadas y los valores reales desnormalizados del conjunto de prueba.
- **print(f'Mean Squared Error (MSE): mse'):** Imprime el valor del MSE.
- **print(f'Mean Absolute Error (MAE): mae'):** Imprime el valor del MAE.
- **print(f'R-squared (R2): r2'):** Imprime el valor del R^2.

Estas métricas proporcionan información sobre la calidad de las predicciones del modelo.

SEGUNDO MODELO: LSTM.

1. Evaluar el modelo en el conjunto de prueba.

Código en Python

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

   - **loss = model.evaluate(X_test, y_test):** Evalúa el modelo en el conjunto de prueba (X_test, y_test) y calcula la pérdida. La pérdida es una medida de cuánto difieren las predicciones del modelo de los valores reales en el conjunto de prueba. El resultado se guarda en la variable loss.
   - **print(f'Loss en el conjunto de prueba: loss'):** Imprime en la consola el valor de la pérdida en el conjunto de prueba. Esto suministra información sobre qué tan bien está realizando el modelo en datos no vistos.

2. Realizar Predicciones.

Código en Python

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

   - **predictions = model.predict(X_test):** Usa el modelo entrenado (model) para realizar predicciones en el conjunto de prueba (X_test). La variable predictions sujetará las predicciones del modelo para cada muestra en el conjunto de prueba.

3. Desnormalizar las predicciones.

Código en Python

```
# Desnormalizar las predicciones y los objetivos para su evaluación
predictions_denormalized = scaler.inverse_transform(predictions)
y_test_denormalized = scaler.inverse_transform(y_test.reshape(-1,
1))
```

- **predictions_denormalized = scaler.inverse_transform(predictions):** Desnormaliza las predicciones del modelo utilizando el objeto scaler inverso. Durante la preparación de los datos, la columna de salida ('IRRADIACIÓN SOLAR GLOBAL PROMEDIO (Wh/m^2)') fue normalizada para el entrenamiento del modelo. Estas predicciones normalizadas se están transformando de nuevo a la escala original.
- **y_test_denormalized = scaler.inverse_transform(y_test.reshape(-1, 1)):** Desnormaliza las etiquetas del conjunto de prueba (y_test). Al igual que con las predicciones, se usa el objeto scaler inverso para transformar las etiquetas del conjunto de prueba de vuelta a la escala original.

4. Calcular y mostrar métricas de evaluación:

Código en Python

```
# Calcular métricas de evaluación
mse=mean_squared_error(y_test_denormalized,predictions_
denormalized)
mae=mean_absolute_error(y_test_denormalized,predictions_
denormalized)
r2=r2_score(y_test_denormalized,predictions_denormalized)
# Mostrar métricas de evaluación
print(f'Mean Squared Error (MSE): {mse}')
print(f'Mean Absolute Error (MAE): {mae}')
print(f'R-squared (R2): {r2}')
```

- **mse = mean_squared_error(y_test_denormalized,predictions_denormalized):** Calcula el Mean Squared Error (MSE), que es una medida de la magnitud del error cuadrático promedio entre las predicciones y los valores reales desnormalizados del conjunto de prueba.
- **mae = mean_absolute_error(y_test_denormalized,predictions_denormalized):** Calcula el Mean Absolute Error (MAE), que es la media de la magnitud de los errores absolutos entre las predicciones y los valores reales desnormalizados del conjunto de prueba.
- **r2 = r2_score(y_test_denormalized, predictions_denormalized):** Calcula el coeficiente de determinación R-squared (R2), que muestra la proporción de la varianza en la variable dependiente que es predecible a partir de las variables independientes. Es una medida de la bondad del ajuste del modelo.
- **print(f'Mean Squared Error (MSE): mse'):** Imprime el valor del Mean Squared Error.

- **print(f'Mean Absolute Error (MAE): mae'):** Imprime el valor del Mean Absolute Error.
- **print(f'R-squared (R2): r2'):** Imprime el valor del coeficiente de determinación R-squared.

MODELO TRES: GRU

1. Evaluar el modelo en el conjunto de prueba.

Código en Python

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

- **loss = model.evaluate(X_test, y_test):** Esta línea evalúa el modelo en el conjunto de prueba usando los datos de entrada (X_test) y los valores reales de salida (y_test). El resultado es el valor de la función de pérdida del modelo en el conjunto de prueba.
- **print(f'Loss en el conjunto de prueba: loss'):** Imprime en la consola el valor de la pérdida (loss) calculado en la línea anterior. La pérdida es una medida de cuánto se desvían las predicciones del modelo en cotejo con los valores reales en el conjunto de prueba.

2. Realizar Predicciones.

Código en Python

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

- **predictions = model.predict(X_test):** Esta línea usa el modelo entrenado (model) para ejecutar predicciones en el conjunto de prueba (X_test). El resultado, guardado en la variable predictions, son las predicciones del modelo para los datos de entrada del conjunto de prueba.

3. Desnormalizar las predicciones.

Código en Python

```
# Desnormalizar las predicciones y los objetivos para su evaluación
predictions_denormalized=scaler.inverse_transform(predictions.
    reshape(-1, 1))
y_test_denormalized = scaler.inverse_transform(y_test.reshape(-1,
    1))
```

- **predictions_denormalized = scaler.inverse_transform(predictions.reshape(-1, 1)):** Usa el objeto **scaler** (normalizador) para deshacer la normalización aprovechada previamente a las predicciones (**predictions**). La función

inverse_transform revierte la transformación, y **reshape(-1, 1)** se usa para asegurar que las dimensiones sean relacionadas con la entrada original.

- **y_test_denormalized = scaler.inverse_transform(y_test.reshape(-1, 1)):** Equivalente a la línea anterior, desnormaliza los objetivos del conjunto de prueba **(y_test)** usando el mismo objeto **scaler**.

4. Desnormalizar las predicciones.

Código en Python

```
# Calcular y mostrar métricas de evaluación
mse=mean_squared_error(y_test_denormalized,predictions_
denormalized)
mae=mean_absolute_error(y_test_denormalized,predictions_
denormalized)
r2=r2_score(y_test_denormalized, predictions_denormalized)
# Mostrar métricas de evaluación
print(f'Mean Squared Error (MSE): {mse}')
print(f'Mean Absolute Error (MAE): {mae}')
print(f'R-squared (R2): {r2}')
```

- **mse = mean_squared_error(y_test_denormalized,predictions_denormalized):** Calcula el Mean Squared Error (MSE), que es una medida de la magnitud del error cuadrático promedio entre las predicciones y los valores reales desnormalizados del conjunto de prueba.
- **mae = mean_absolute_error(y_test_denormalized,predictions_denormalized):** Calcula el Mean Absolute Error (MAE), que es la media de la magnitud de los errores absolutos entre las predicciones y los valores reales desnormalizados del conjunto de prueba.
- **r2 = r2_score(y_test_denormalized, predictions_denormalized):** Calcula el coeficiente de determinación R-squared (R2), que muestra la proporción de la varianza en la variable dependiente que es predecible a partir de las variables independientes. Es una medida de la bondad del ajuste del modelo.
- **print(f'Mean Squared Error (MSE): mse'):** Imprime el valor del Mean Squared Error.
- **print(f'Mean Absolute Error (MAE): mae'):** Imprime el valor del Mean Absolute Error.
- **print(f'R-squared (R2): r2'):** Imprime el valor del coeficiente de determinación R-squared.

Además de lo propuesto anteriormente, se llevó a cabo la elaboración de promedios de la irradiación solar. Dado que los datos son registrados por la estación meteorológica cada 5 minutos, se optó por calcular promedios a intervalos de 20 minutos y 60 minutos.

Esta decisión se tomó con el objetivo de evaluar si dichos promedios podrían evidenciar mejoras en cada uno de los modelos desarrollados.

A continuación, se presenta el código implementado en el entorno de Python para calcular los promedios de irradiación solar en intervalos de 20 y 60 minutos.

Código en Python

```
# Cargar el archivo Excel en un DataFrame de pandas
df = pd.read_excel('Datos 2021.xlsx')
# Convertir la columna de fecha a formato datetime
df['FECHA Y HORA'] = pd.to_datetime(df['FECHA Y HORA'])
# Establecer 'Fecha' como índice para usar el método resample
df.set_index('FECHA Y HORA', inplace=True)
# Realizar el promedio cada 10 minutos
df_resample = df.resample('20T').mean()
# Restablecer el índice si lo prefieres
df_resample.reset_index(inplace=True)
# Imprimir el resultado o guardar en un nuevo archivo Excel
print(df_resample)
```

- **df = pd.read_excel('Datos 2021.xlsx'):** Carga un archivo Excel ('Datos 2021.xlsx') en un DataFrame de pandas llamado df. Este DataFrame contendrá los datos que se trabajarán.
- **df['FECHA Y HORA'] = pd.to_datetime(df['FECHA Y HORA']):** Convierte la columna 'FECHA Y HORA' del DataFrame a un formato de datetime. Esto es preciso para que pandas pueda reconocer y manejar las fechas de manera adecuada
- **df.set_index('FECHA Y HORA', inplace = True):** Establece la columna 'FECHA Y HORA' como índice del DataFrame.Se prepara el DataFrame para el uso de la función resample y otras operaciones temporales.
- **df_resample = df.resample('20T').mean():** Usa el método resample para calcular el promedio de los datos en intervalos de 20 minutos. Esta línea agrupa los datos temporalmente y calcula el promedio para cada intervalo de 20 minutos.
- **df_resample.reset_index(inplace = True):** Restablece el índice del DataFrame resultante después de usar la operación de resampling. Esto puede ser opcional dependiendo de las preferencias del usuario. En este caso, se ha restablecido el índice para obtener un DataFrame con un índice numérico consecutivo.
- **print(df_resample):** Imprime en la consola el DataFrame resultante después de la operación de resampling y promedio. Esto permite visualizar los datos procesados en la consola.

Para calcular el promedio de 60 minutos, simplemente modifica la línea df_resample =df.resample('20T').mean() a df_resample=df.resample('60T').mean(). Este ajuste sencillo permitirá realizar el promedio en intervalos de 60 minutos en lugar de 20 minutos. Para realizar el filtrado de datos, basta con ampliar la variable df_resample. A continuación, se presenta el código modificado con el pequeño ajuste mencionado anteriormente. Cada una de las líneas ya ha sido descrita previamente, por lo que se puede referir al proceso de procesamiento de datos.

Código en Python

```
#Se filtran los datos de 6 AM a 6:30 PM.
df_resample['FECHA Y HORA'] = pd.to_datetime(df_resample['FECHA Y HORA']
, format='%d/%m/%Y %I:%M %p')
data = df_resample[(df_resample['FECHA Y HORA'].dt.hour >= 6) &
((df_resample['FECHA Y HORA'].dt.hour < 18) | ((df_resample['FECHA Y
HORA'].dt.hour == 18) & (df_resample['FECHA Y HORA'].dt.minute <= 30)))]
#Seleccionamos las Columas de 'FECHA Y HORA'y'IRRADIACIÓN SOLAR GLOBAL
   PROMEDIO (Wh/m^2)
columnas_seleccionadas = data[['FECHA Y HORA' , 'IRRADIACIÓN SOLAR
   GLOBAL PROMEDIO
(Wh/m^2)']]
columnas_seleccionadas.head()
print(f'Tama o del set antes de eliminar registros del dataset:
{columnas_seleccionadas.shape}')
columnas_seleccionadas=columnas_seleccionadas[columnas_seleccionadas
['IRRADIACIÓN SOLAR GLOBAL PROMEDIO (Wh/m^2)']>0]
print(f'Tama o del set despues de eliminar registros del dataset:
{columnas_seleccionadas.shape}')
y=columnas_seleccionadas['IRRADIACIÓN SOLAR GLOBAL PROMEDIO (Wh/m^2)']
X=columnas_seleccionadas.drop(['IRRADIACIÓN SOLAR GLOBAL PROMEDIO
(Wh/m^2)'], axis=1)
columnas_seleccionadas.head()
```

6. Resultados

Primer Modelo

Para poder visualizar los resultados es necesario realizar una gráfica entre los datos reales junto a los datos predichos, para obtener esta gráfica es necesario realizar un código el mismo que es utilizado para los tres modelos y que lo podemos encontrar en el **Anexo 1**:

El siguiente modelo posee el rendimiento tomando en cuenta las siguientes métricas de evaluación:

- Mean Squared Error (MSE): 51.582
- Mean Absolute Error (MAE): 2.714
- R-squared (R2): 0.8524

En la **Figura 25** se puede observar la comparativa entre los valores reales y los valores predichos.

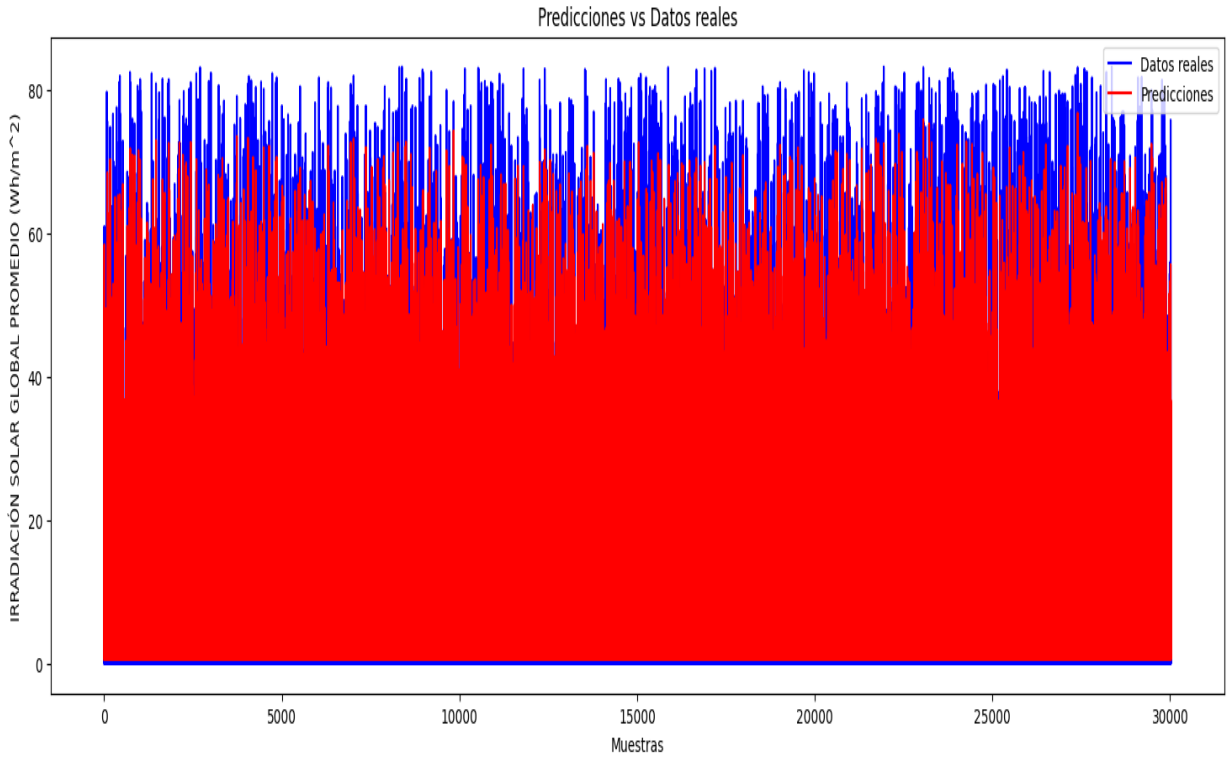

Figura 25. Valores reales vs Valores predichos.

También se realizó una gráfica de pérdida entre el entrenamiento y la validación, en la siguiente **Figura 26**, podemos apreciar estos valores.

Para obtener la gráfica antes mencionada es necesario realizar la siguiente programación, la cual la se detalló en el **Anexo 2**, este código sirve para los tres modelos realizados:

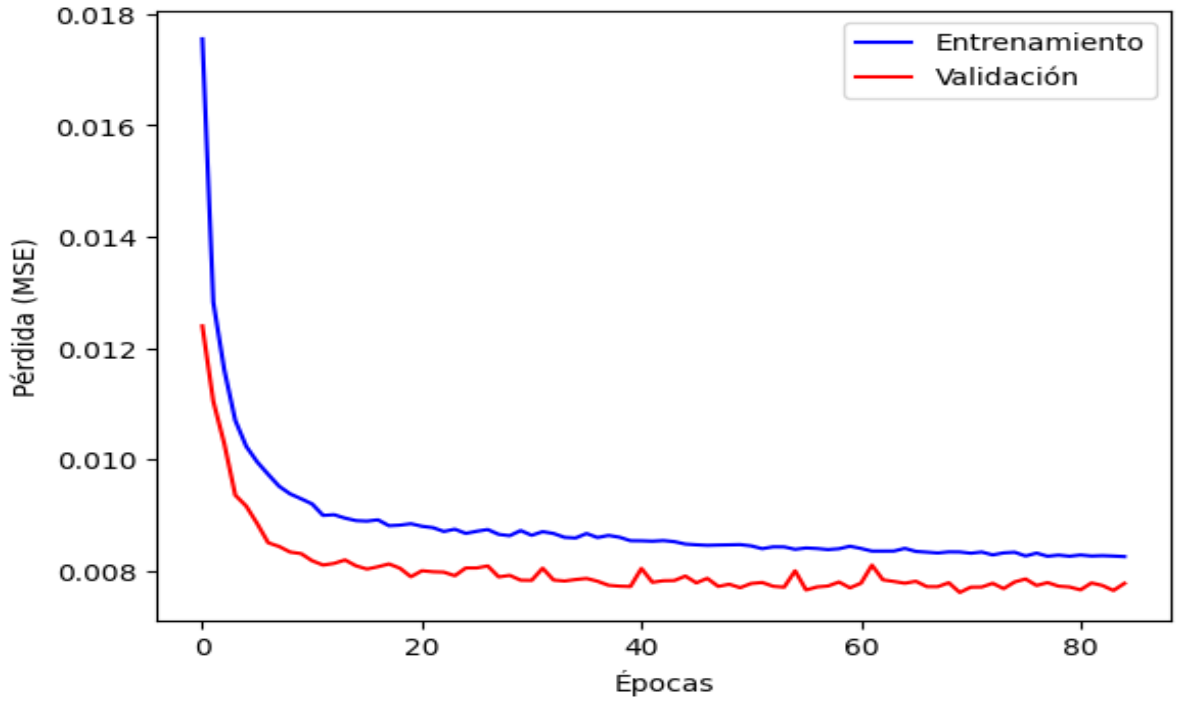

Figura 26. Pérdida entre el entrenamiento y validación a lo largo de las épocas.

Segundo Modelo

El siguiente modelo posee un rendimiento tomando en cuenta las siguientes métricas de evaluación:

- Mean Squared Error (MSE): 51.704
- Mean Absolute Error (MAE): 2.563
- R-squared (R2): 0.8521

En la **Figura 27** se puede observar la comparativa entre los valores reales y los valores predichos.

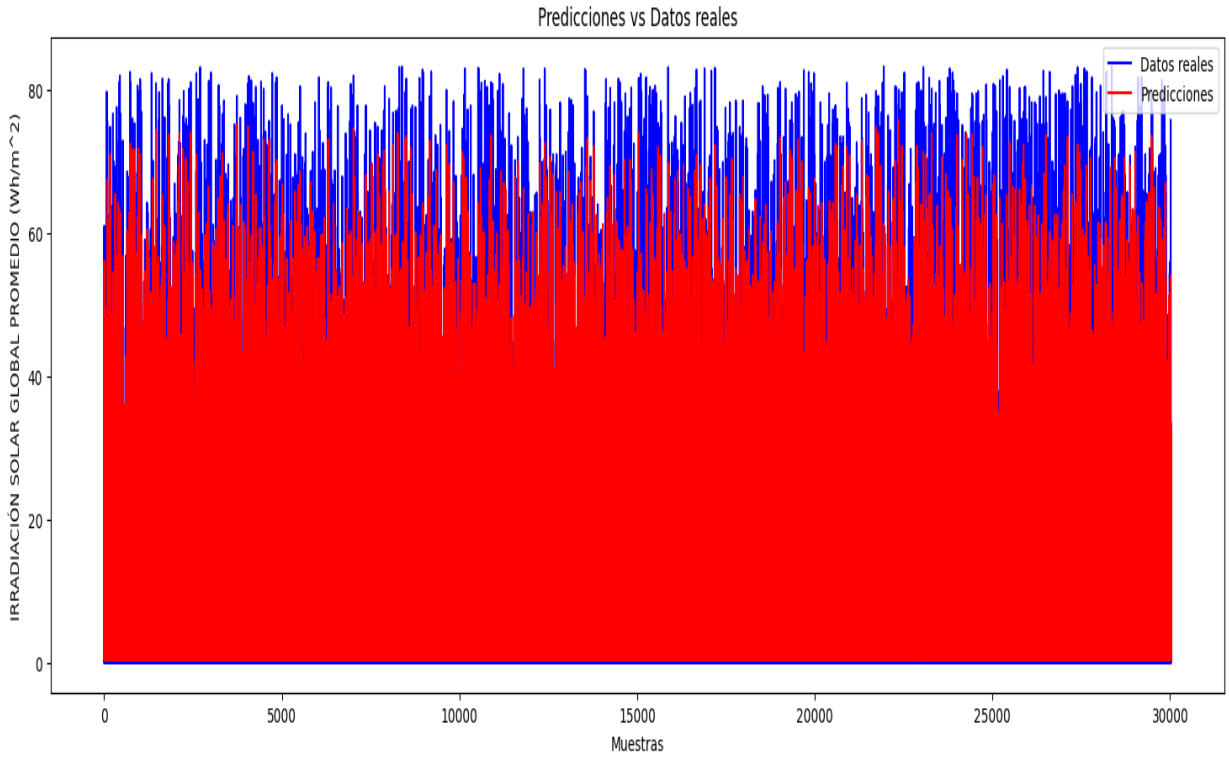

Figura 27. Valores reales vs Valores predichos.

También se realizó una gráfica de pérdida entré en entrenamiento y la validación, en la siguiente **Figura 28**, podemos apreciar estos valores.

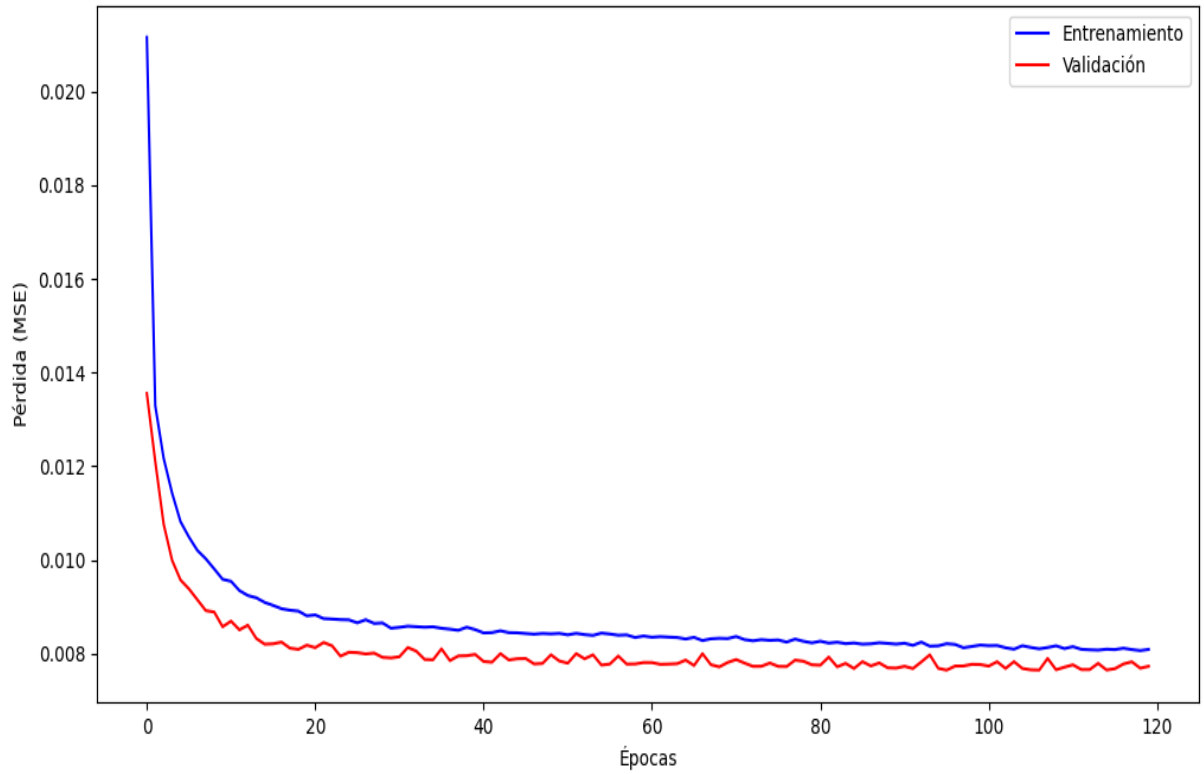

Figura 28. Pérdida entre el entrenamiento y validación a lo largo de las épocas.

Tercer modelo

El siguiente modelo posee el rendimiento tomando en cuenta las siguientes métricas de evaluación:

- Mean Squared Error (MSE): 51.678
- Mean Absolute Error (MAE): 2.566
- R-squared (R2): 0.8522

En la **Figura 29** se puede observar la comparativa entre los valores reales y los datos predichos.

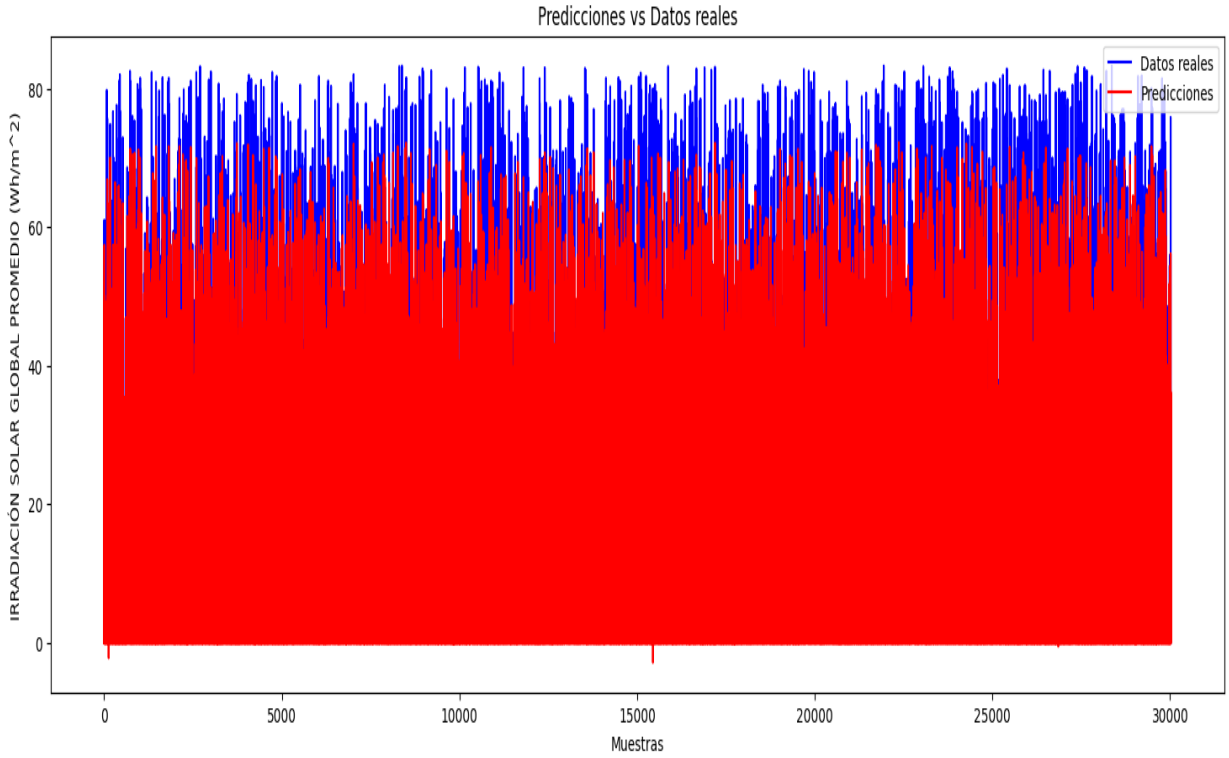

Figura 29. Valores reales vs Valores predichos.

También se realizó una gráfica de pérdida entré el entrenamiento y la validación, en la siguiente **Figura 30**, podemos apreciar estos valores.

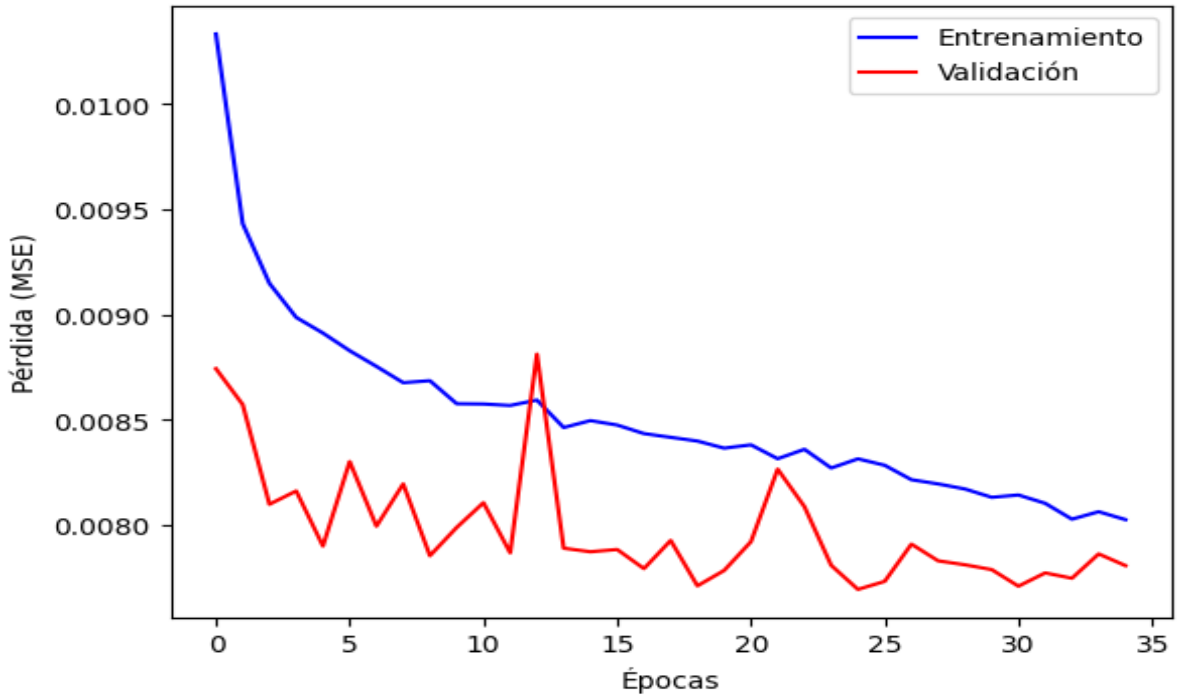

Figura 30. Pérdida entre el entrenamiento y validación a lo largo de las épocas.

Luego de explicar cada parte del código, de cada uno de los modelos desarrollados, se va a presentar cada uno de los resultados con los promedios de la irradiación solar de 20 y 60 minutos respectivamente.

Cada 20 minutos.

Primer Modelo: Red Neuronal RNN

En la ***Tabla 3*** se puede visualizar los diferentes hiperparámetros que posee este modelo junto a las diferentes métricas utilizadas anteriormente.

Tabla 3. Resultado de la predicción de irradiación solar utilizando el primer modelo en un intervalo de 20 minutos.

Modelo	Dropout	Adam	épocas	seq_-length	batch_-size	(MSE)	(MAE)	(R^2)
RNN	0.1	0.0001	100	10	84	51.812	2.638	0.8518

En la **Figura 31** se puede observar la comparativa entre los valores reales y los datos predichos.

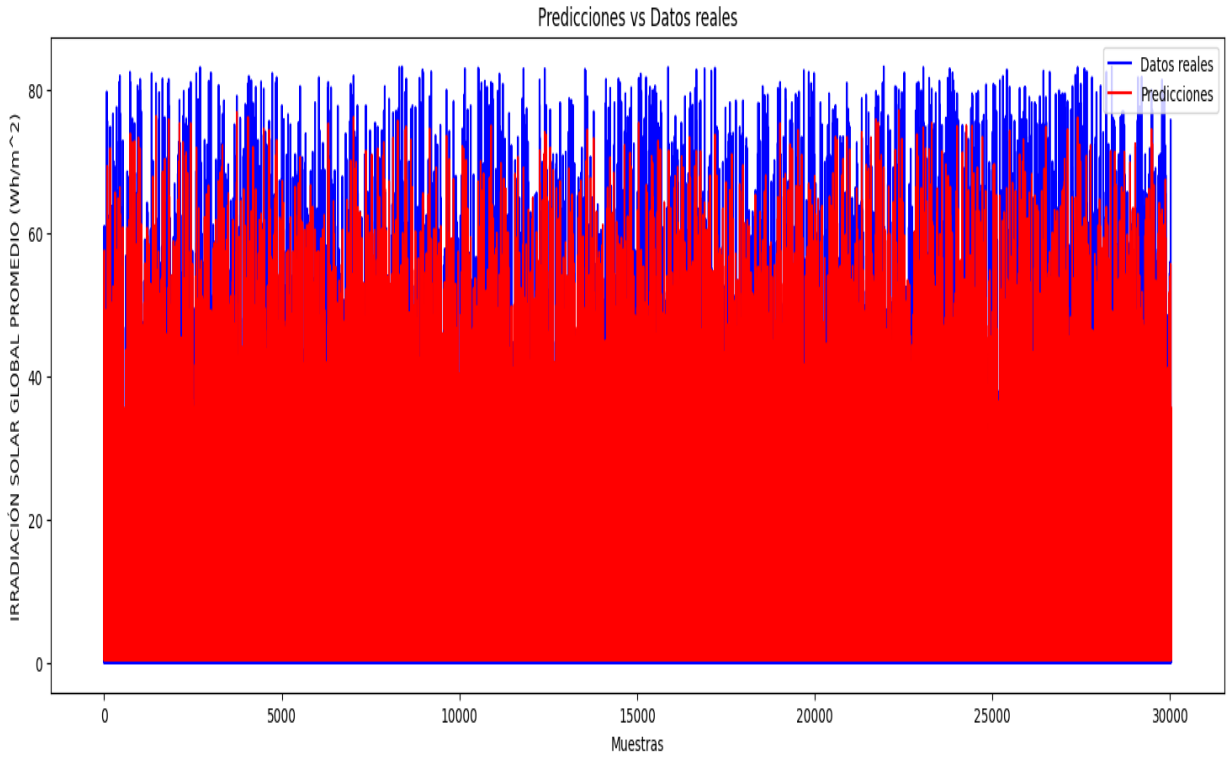

Figura 31. Valores reales vs Valores predichos.

También se realizó una gráfica de pérdida, entre el entrenamiento y la validación, en la siguiente **Figura 32**, podemos apreciar estos valores.

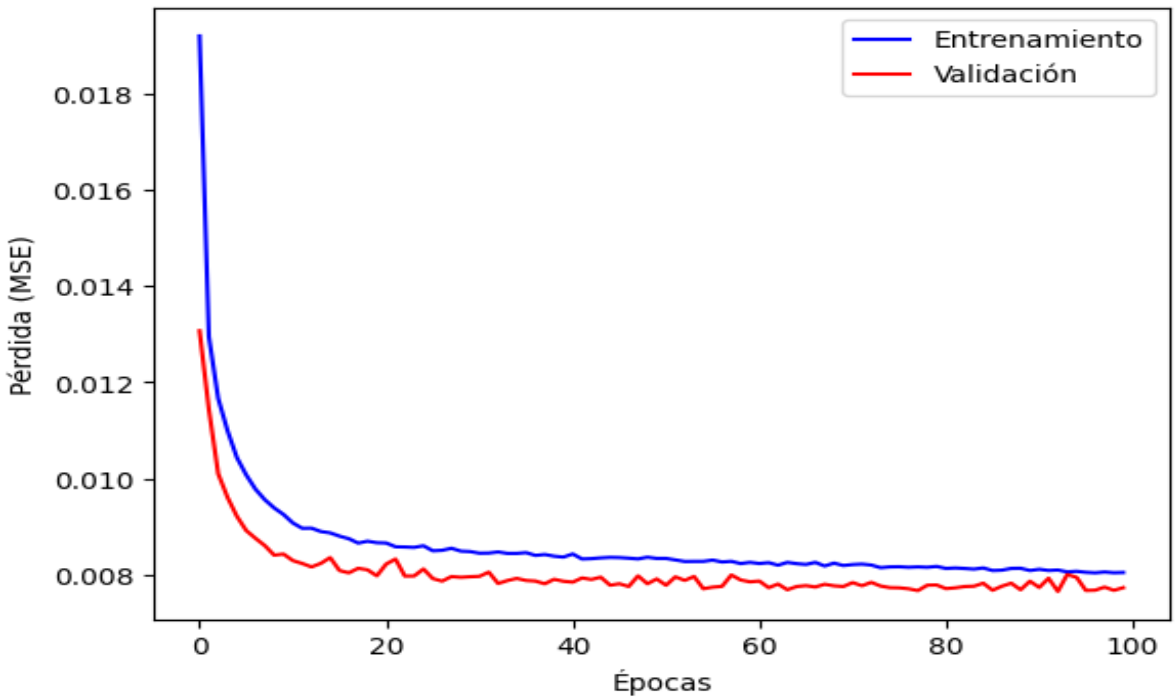

Figura 32. Pérdida entre el entrenamiento y validación a lo largo de las épocas.

Segundo Modelo: LSTM

En la **Tabla 4** se puede visualizar los diferentes hiperparámetros que posee este modelo junto a las diferentes métricas utilizadas anteriormente. ***Tabla 4:*** Resultado de la predicción de irradiación solar utilizando el segundo modelo en un intervalo de 20 minutos.

Tabla 4. Resultado de la predicción de irradiación solar utilizando el segundo modelo en un intervalo de 20 minutos.

Modelo	**Dropout**	**Adam**	**épocas**	**seq_-length**	**batch_-size**	**(MSE)**	**(MAE)**	**(R^2)**
LSTM	0.15	0.0001	150	10	168	52.043	2.743	0.8511

En la **Figura 33** se puede observar la comparativa entre los valores reales y los valores predichos.

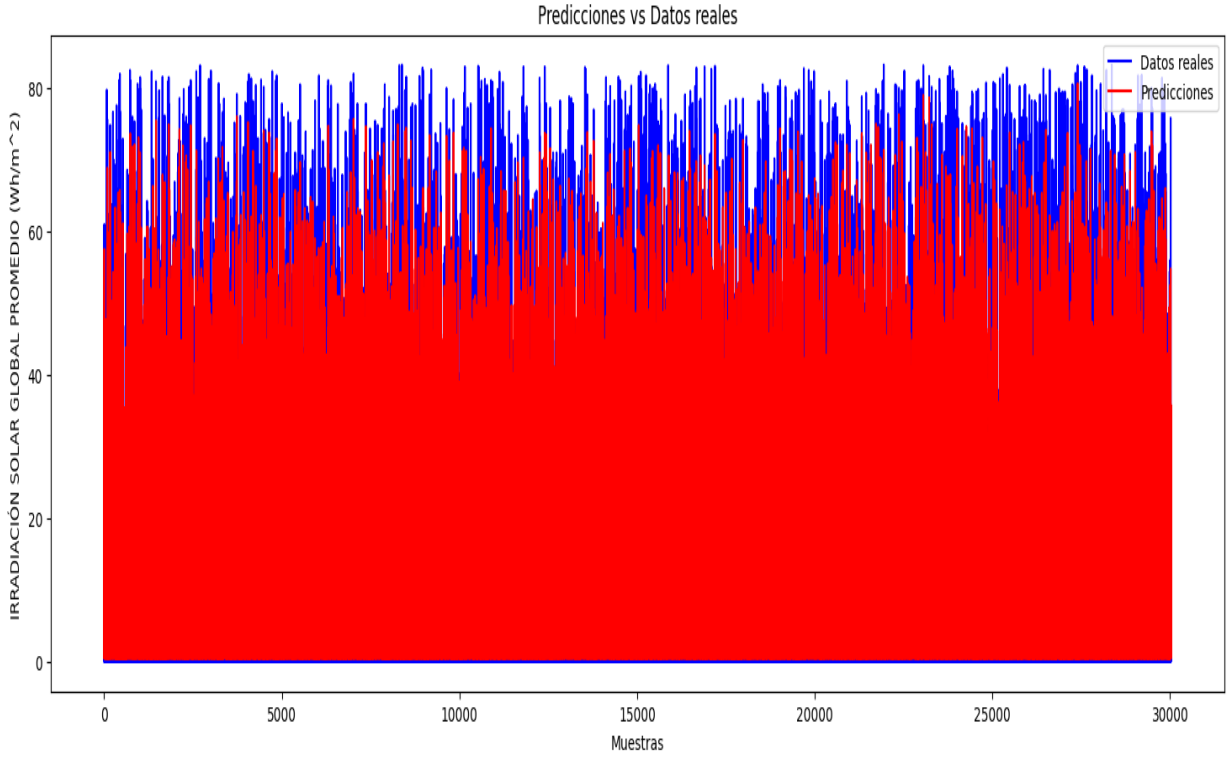

Figura 33. Valores reales vs Valores predichos.

También se realizó una gráfica de pérdida, entre en entrenamiento y la validación, en la siguiente **Figura 34**, podemos apreciar estos valores.

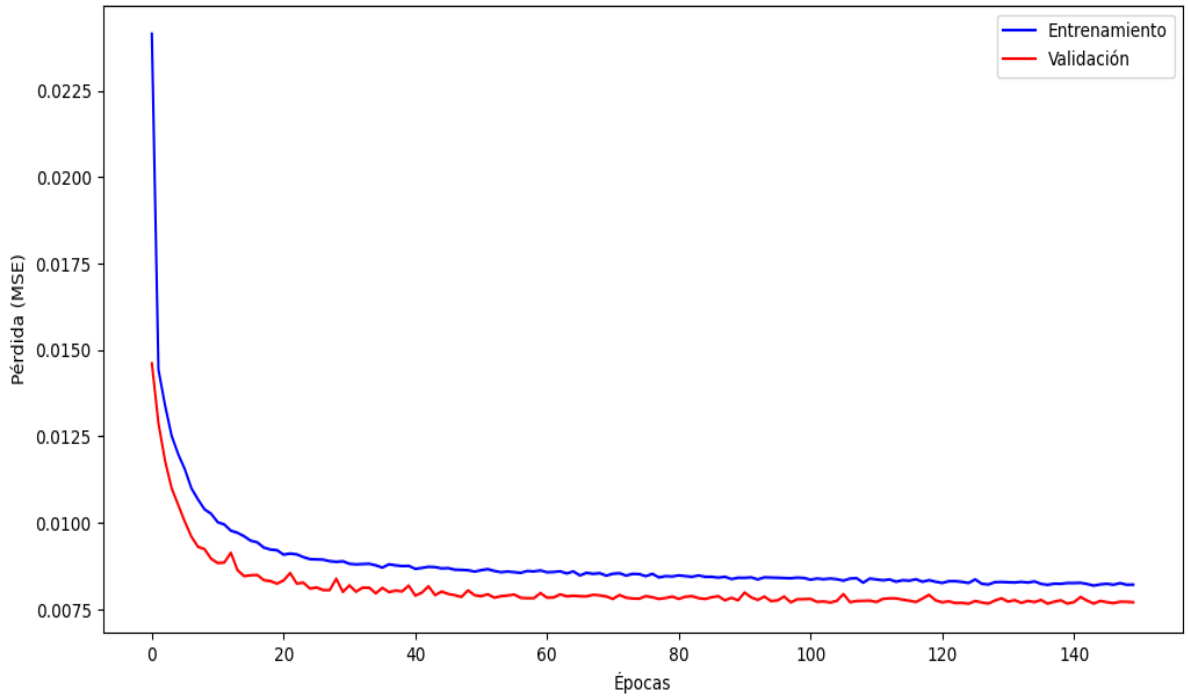

Figura 34. Pérdida entre el entrenamiento y validación a lo largo de las épocas.

Tercer Modelo: GRU

En la ***Tabla 5*** se puede visualizar los diferentes hiperparámetros que posee este modelo junto a las diferentes métricas utilizadas anteriormente.

Tabla 5. Resultado de la predicción de irradiación solar utilizando el tercer modelo en un intervalo de 20 minutos.

Modelo	Dropout	Adam	épocas	seq_-length	batch_-size	(MSE)	(MAE)	(R^2)
GRU	0.2	0.001	100	10	84	52.381	2.717	0.8504

En la **Figura 35** se puede observar la comparativa entre los valores reales y los valores predichos.

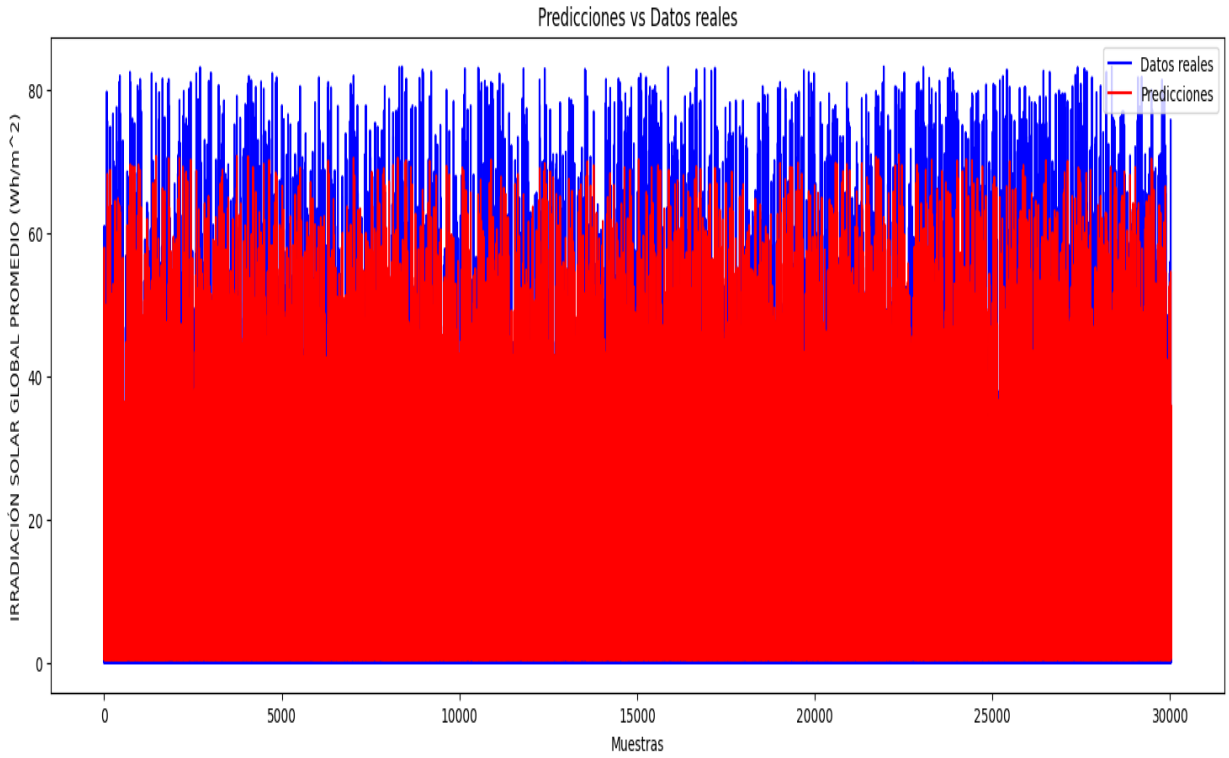

Figura 35. Valores reales vs Valores predichos.

También se realizó una gráfica de pérdida, entre en entrenamiento y la validación, en la siguiente **Figura 36**, podemos apreciar estos valores.

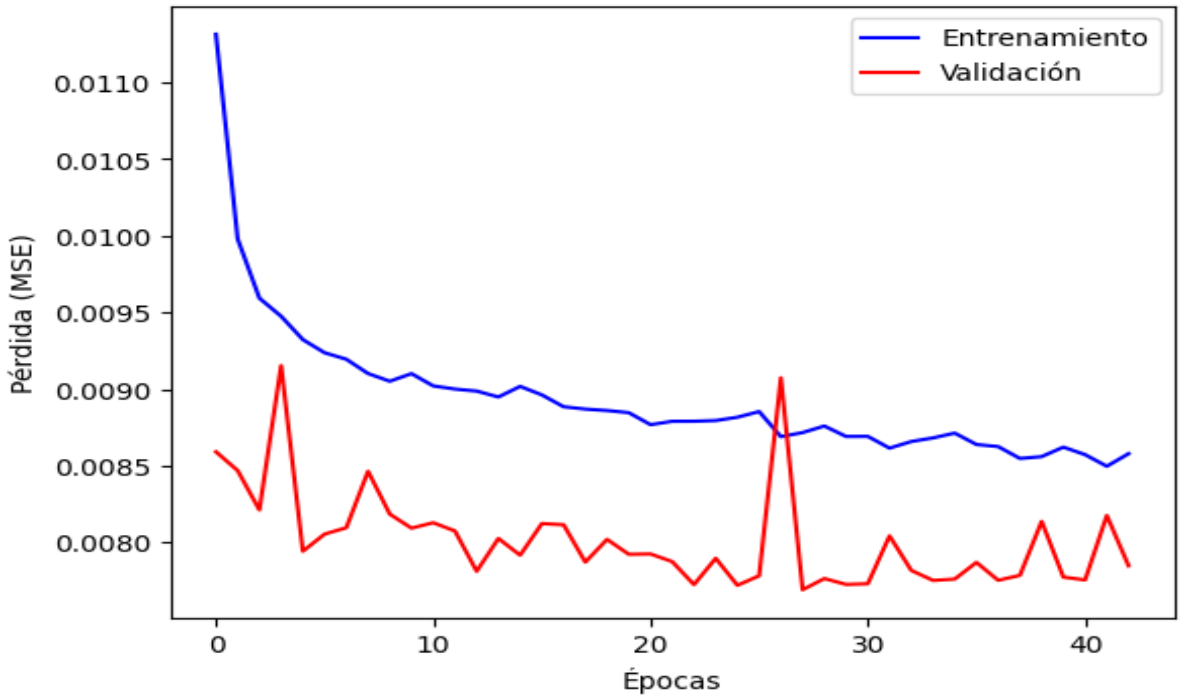

Figura 36. Pérdida entre el entrenamiento y validación a lo largo de las épocas.

Cada 60 minutos.

Primer Modelo: Red Neuronal RNN

En la ***Tabla 6*** se puede visualizar los diferentes hiperparámetros que posee este modelo junto a las diferentes métricas utilizadas anteriormente.

Tabla 6. Resultado de la predicción de irradiación solar utilizando el primer modelo en un intervalo de 60 minutos.

Modelo	Dropout	Adam	épocas	seq_-length	batch_-size	(MSE)	(MAE)	(R^2)
RNN	0.1	0.0001	84	10	150	52.004	2.58	0.8512

En la **Figura 37** se puede observar la comparativa entre los valores reales y los valores predichos.

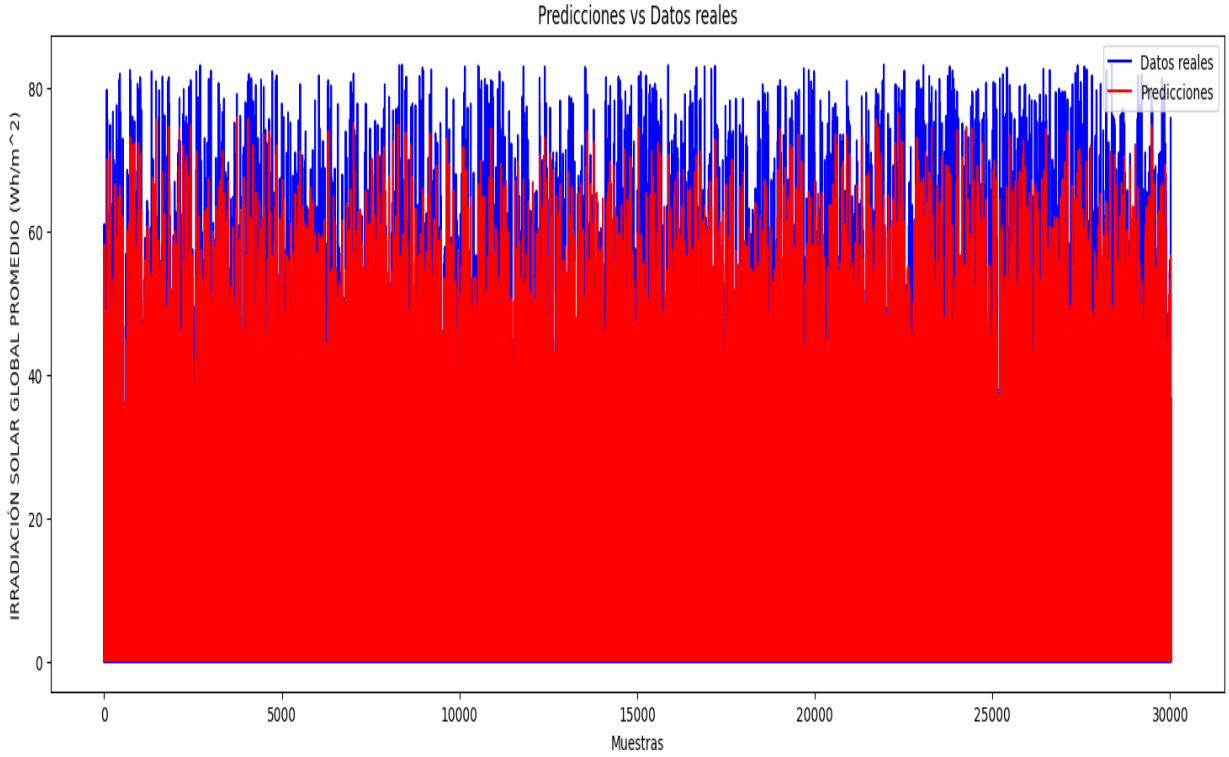

Figura 37. Valores reales vs Valores predichos.

También se realizó una gráfica de pérdida, entre en entrenamiento y la validación, en la siguiente **Figura 38**, podemos apreciar estos valores.

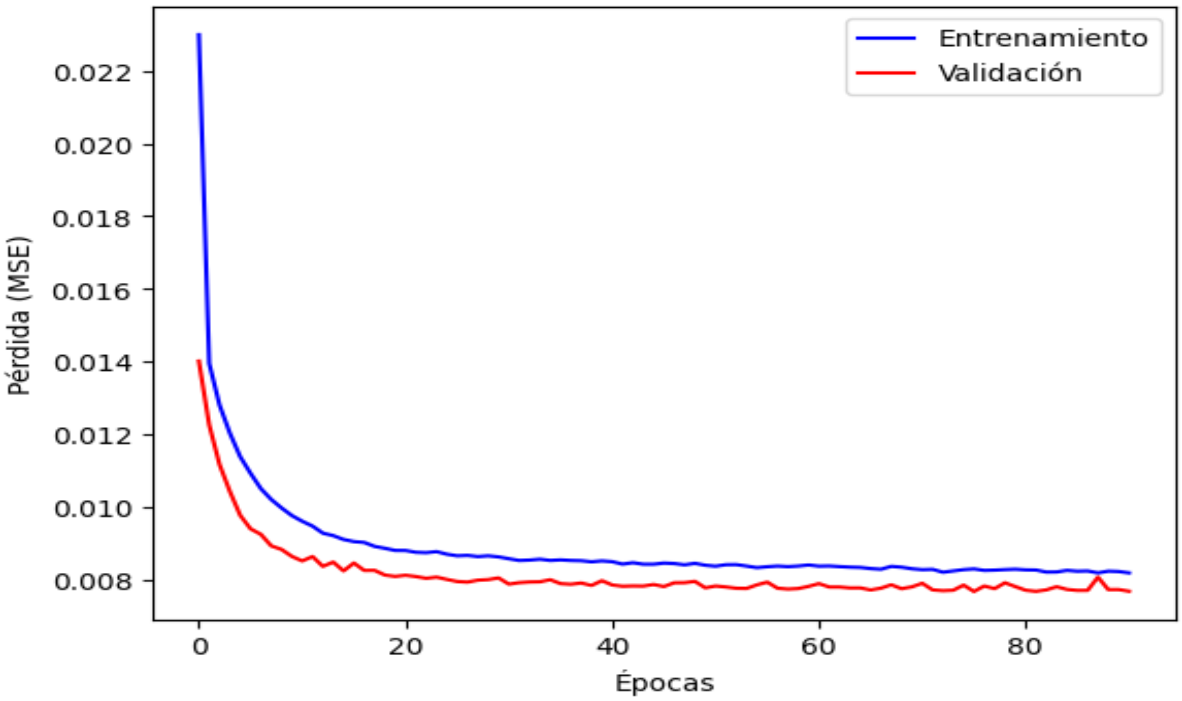

Figura 38. Pérdida entre el entrenamiento y validación a lo largo de las épocas.

Segundo Modelo: LSTM

En la ***Tabla 7*** se puede visualizar los diferentes hiperparámetros que posee este modelo junto a las diferentes métricas utilizadas anteriormente.

Tabla 7. Resultado de la predicción de irradiación solar utilizando el segundo modelo en un intervalo de 60 minutos.

Modelo	Dropout	Adam	épocas	seq_-length	batch_-size	(MSE)	(MAE)	(R^2)
LSTM	0.15	0.0001	150	10	168	52.286	2.712	0.8504

En la **Figura 39** se puede observar la comparativa entre los valores reales y los valores predichos.

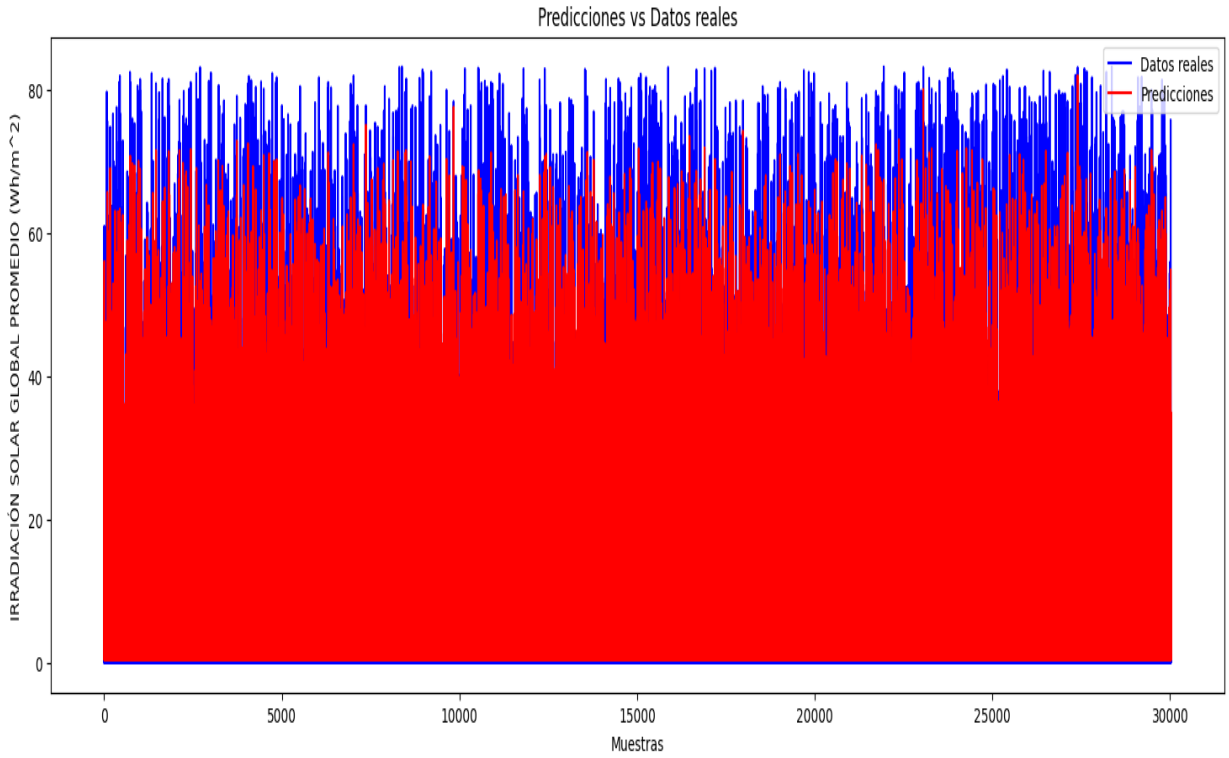

Figura 39. Valores reales vs Valores predichos.

También se realizó una gráfica de pérdida, entre en entrenamiento y la validación, en la siguiente **Figura 40**, podemos apreciar estos valores.

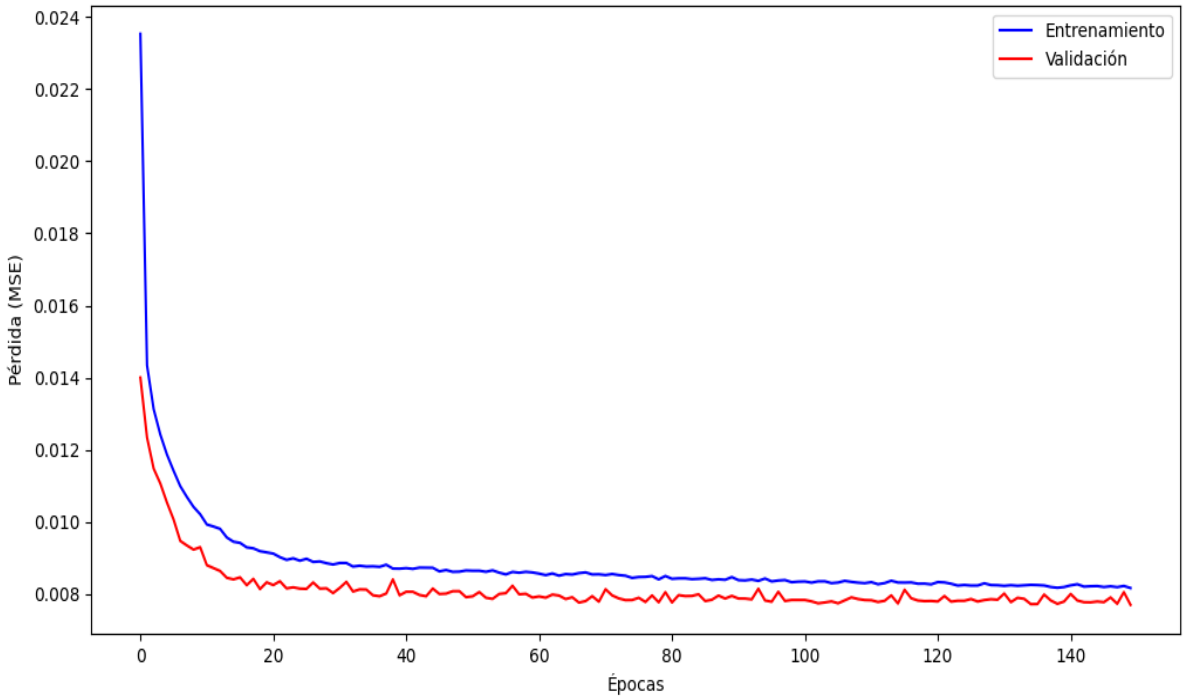

Figura 40. Pérdida entre el entrenamiento y validación a lo largo de las épocas.

Tercer Modelo: GRU

En la ***Tabla 8*** se puede visualizar los diferentes hiperparámetros que posee este modelo junto a las diferentes métricas utilizadas anteriormente.

Tabla 8. Resultado de la predicción de irradiación solar utilizando el tercer modelo en un intervalo de 60 minutos.

Modelo	Dropout	Adam	épocas	seq_-length	batch_-size	(MSE)	(MAE)	(R^2)
GRU	0.2	0.001	35	10	32	52.003	2.441	0.8512

En la **Figura 41** se puede observar la comparativa entre los valores reales y los valores predichos.

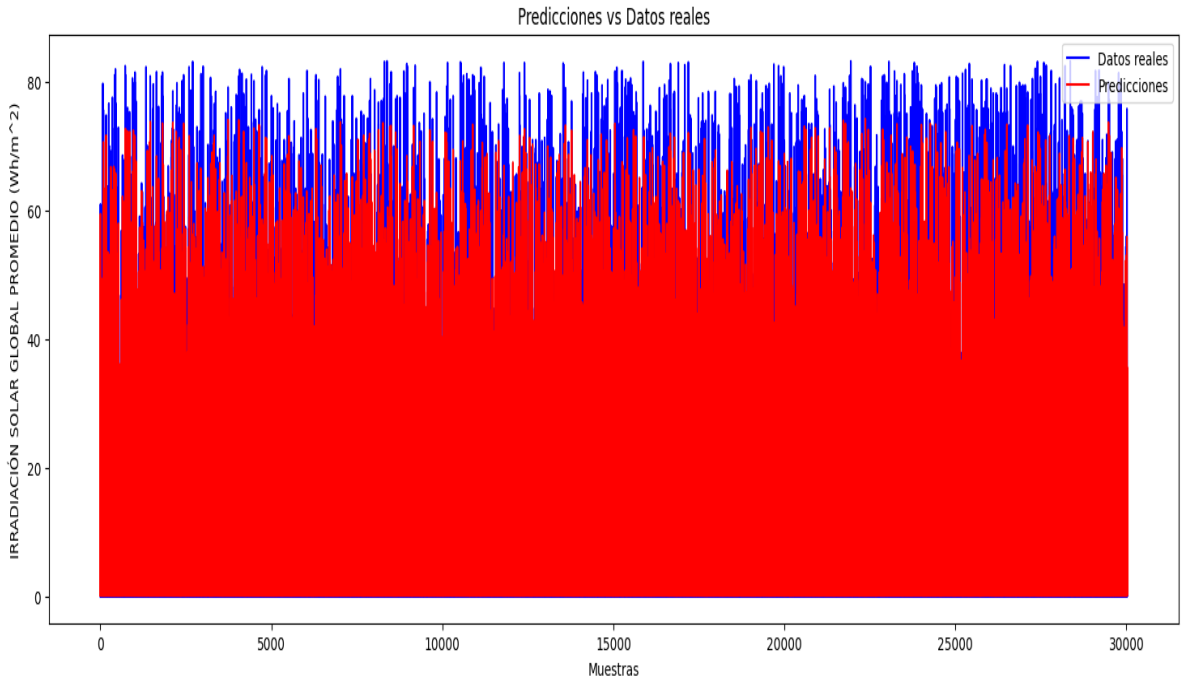

Figura 41. Valores reales vs Valores predichos.

También se realizó una gráfica de pérdida, entre en entrenamiento y la validación, en la siguiente **Figura 42**, podemos apreciar estos valores.

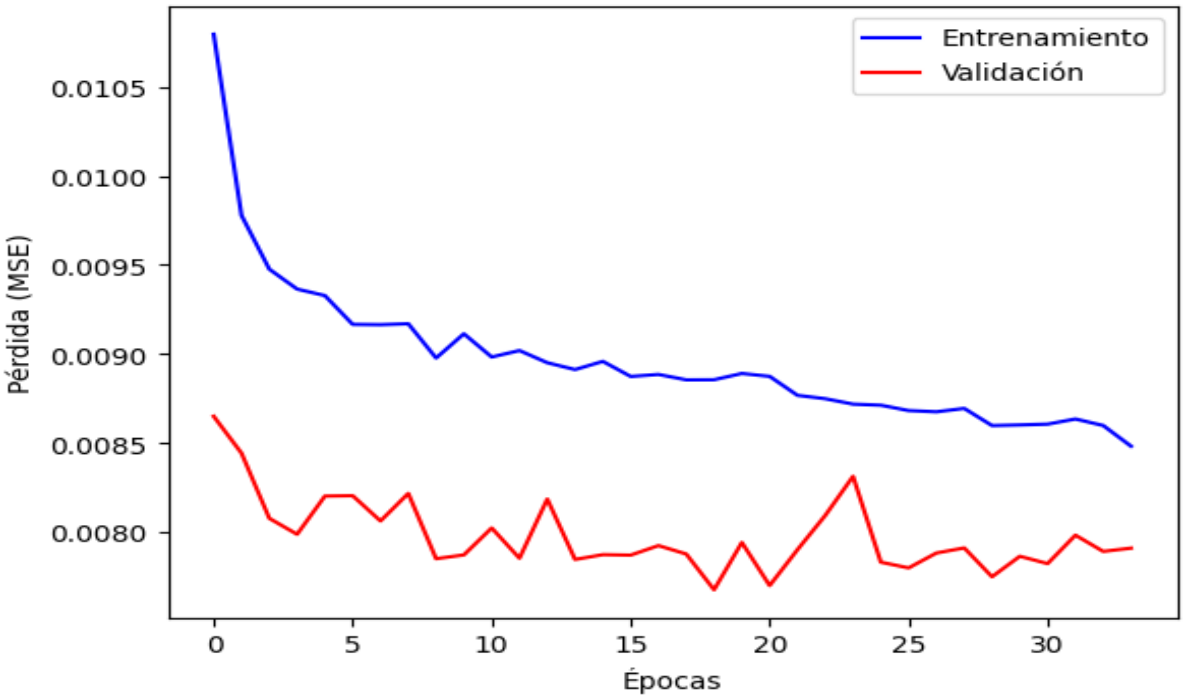

Figura 42. Pérdida entre el entrenamiento y validación a lo largo de las épocas.

En la ***Tabla 9*** se detalla un resumen de todos los modelos con sus diferentes hiperparámetros, al igual que la aplicación de las diferentes métricas utilizadas a lo largo del procesamiento.

Tabla 9. Resultado de la predicción de irradiación solar utilizando todos los modelos en diferentes intervalos de tiempo.

Modelo	Tiempo	Mean Squared Error	Mean Absolute Error	R-squared
RNN	5 minutos	51.582	2.714	0.8524
RNN	20 minutos	51.812	2.638	0.8518
RNN	60 minutos	52.004	2.58	0.8512
LSTM	5 minutos	51.704	2.563	0.8521
LSTM	20 minutos	52.043	2.743	0.8511
LSTM	60 minutos	52.286	2.712	0.8504
GRU	5 minutos	51.678	2.566	0.8522
GRU	20 minutos	52.381	2.717	0.8504
GRU	60 minutos	52.003	2.441	0.8512

7. Discusión

En nuestro estudio, se adquirieron datos de la estación meteorológica situada dentro de la FEIRNNR. Posteriormente, se realizó una minuciosa limpieza de los datos con el propósito de detectar posibles valores nulos o incorrectos. Al revisar la información, se confirmó la ausencia de datos faltantes o mediciones atípicas que excedieran el rango máximo de 83.25 Wh/m^2 visualizado. Los registros, recopilados cada 5 minutos durante las 24 horas del año 2021, se utilizaron para generar tres gráficas que ofrecen una comprensión detallada de la irradiación solar global promedio en este lugar, abarcando un día, un mes y todo el año 2021. Es importante destacar que se eliminaron los datos de irradiación solar igual a cero. En cuanto al cálculo de promedios, se optó por no realizar una media aritmética para reemplazar los valores nulos, dado que la variabilidad climática de Loja desaconseja este enfoque. En su lugar, se seleccionaron las columnas de FECHA, HORA y IRRADIACIÓN SOLAR PROMEDIO Wh/m^2. Estos datos se dividieron en conjuntos de entrenamiento (60 %), validación (20 %) y prueba (20 %). Además, se llevó a cabo un promedio de 20 y 60 minutos, respectivamente, durante el proceso de entrenamiento de los modelos desarrollados.

En el caso de (Nawab et al., 2023), su base de datos abarca cuatro años desde 2014 a 2018, los datos se dividieron en 3 años para el entrenamiento y 1 año para las pruebas, los datos de irradiación solar utilizados en el presente estudio proceden de tres tipos diferentes de paneles fotovoltaicos conectados a la red. Los paneles se instalaron en el tejado superior del Centro de Investigación y Aplicación Científica y Tecnológica de la Universidad de Duzce, los datos se recogieron diariamente con un intervalo de 5 minutos. Por lo tanto, para maximizar la información de nuestros datos, filtramos el periodo entre la puesta y la salida del sol, que varía entre el invierno y el verano. Añadimos los datos meteorológicos que consistían en la nubosidad, la humedad relativa y la hora del sol, a los datos de los paneles.

Para el caso de (Zhao et al., 2024), los datos de irradiación solar se recopilaron en cinco ciudades de Bangladesh (Bogra, Chuadanga, Dhaka, Dinajpur y Satkhira) al Sur de Asia, tomando mediciones entre las 05:00 y las 20:00 en catorce intervalos. Además de la irradiación solar, se recopilaron datos de otros parámetros meteorológicos desde 2014 hasta 2019. Se trató las excepciones como valores nulos y se eliminaron durante el preprocesamiento. Se realizaron ajustes para garantizar la calidad del conjunto de datos, eliminando entradas inválidas y estableciendo rangos para las variables. Después de estos pasos, el conjunto de datos se redujo a 3,997 datos, dividiéndose en conjuntos de entrenamiento (75 %) y prueba (25 %). Se asignaron números a cada estación correspondiente a las ciudades.

La inclusión de múltiples variables directamente relacionadas con la irradiación solar podría mejorar significativamente la precisión en la predicción de este recurso solar para la producción de energía. Al incorporar un mayor número de entradas, el modelo desarrollado tendría la capacidad de operar de manera más efectiva, aprovechando la información adicional proporcionada por estas variables relacionadas con la irradiación solar.

El modelo basado en RNN se ejecutó durante 84 épocas las capas totalmente conectadas son las capas de una red en la que todas las entradas de una capa están conectadas a las unidades de activación de la capa siguiente. La red neuronal tiene tres capas: la primera con 128 unidades, la segunda con 64 unidades, la tercera con 32 unidades y la cuarta con 64 unidades, todas con ReLu como función de activación. La última capa tiene una sola unidad y utiliza la función lineal. Se implementa el método de regularización llamado abandono, con un Dropout de 0,15, para prevenir el sobreajuste del modelo. Además, se utiliza el algoritmo de optimización Adam de 0,0001, el modelo basado en LSTM durante 120 épocas. La red neuronal consta de dos capas LSTM, la primera con 128 unidades emitiendo todos los estados ocultos, la segunda con 64 unidades y la tercera con 32 unidades. Las dos últimas capas son RNN totalmente conectadas, con la cuarta capa de 64 unidades y la quinta de 1 unidad. Se utiliza la función lineal como activación entre capas, ya que las unidades LSTM, todas con ReLu como función de activación. La tasa de abandono con un Dropout de 0,1, para prevenir el sobreajuste del modelo. Además, se utiliza el algoritmo de optimización Adam de 0,0001y el modelo basado en GRU durante 35 épocas, la red GRU tiene cuatro capas: la primera GRU con 128 unidades, la segunda con 64 unidades, la tercera con 32 y la cuarta con 16 unidades, La última capa con una sola unidad, ya que se está prediciendo un valor numérico continuo. No se especifica una función de activación, lo que implica una activación lineal por defecto. Se implementa el método de regularización llamado abandono, con un Dropout de 0,1, para prevenir el sobreajuste del modelo. Además, se utiliza el algoritmo de optimización Adam de 0,001.

En el caso de (Nawab et al., 2023),la red Neuronal LSTM (Long Short-Term Memory), en la Universidad de Duzce (Turquía), este algoritmo realiza predicciones de irradiación solar por hora, focalizándose en el intervalo de las 6 am a las 18 pm debido a la falta de sol en otras horas. La construcción del modelo se basa en tres años de datos (2014,2015,2016,2017). Se entrenó con datos de 2014 y 2026, reservando 2017 para la comparación. La red neuronal tiene tres capas: entrada (64 neuronas), oculta (32 neuronas) y salida. Se aplica Dropout al 25 % para regularizar y evitar el sobreajuste. La función de activación es tangente hiperbólica, usando el optimizador Adam y pérdida de MSE. Se afinó el modelo probando diferentes hiperparámetros.

Para el caso de (Zhao et al., 2024), El modelo basado en RNN se ejecutó durante

100 épocas las capas totalmente conectadas son las capas de una red en la que todas las entradas de una capa están conectadas a las unidades de activación de la capa siguiente. La red neuronal tiene tres capas: la primera con 128 unidades, la segunda con 64 y la tercera con 32, todas con ReLu como función de activación. La última capa tiene una sola unidad y utiliza la función lineal. Se implementa el método de regularización llamado abandono, con una tasa del 0,2, para prevenir el sobreajuste del modelo. Además, se utiliza el algoritmo de optimización Adam., el modelo basado en LSTM durante 125 épocas. La red neuronal consta de dos capas LSTM, la primera con 128 unidades emitiendo todos los estados ocultos, y la segunda con 64 unidades. Las dos últimas capas son RNN totalmente conectadas, con la tercera capa de 16 unidades y la cuarta de 1 unidad. Se utiliza la función lineal como activación entre capas, ya que las unidades LSTM internamente tienen funciones sigmoideas y Tanh. La tasa de abandono se fija en 0,2 y se emplea el algoritmo de optimización Adam, similar a la primera. Haciendo uso de las redes neuronales desarrollo de árboles/regresión de bosque aletorio RNN, LSTM.

Los tres modelos predictivos desarrollados, en comparación con otros investigados, emplean arquitecturas de redes neuronales recurrentes para predecir la irradiación solar. Sin embargo, se observan diferencias en la complejidad y la configuración de hiperparámetros específicos entre estos modelos. Por lo tanto, se debe mencionar que los tres modelos de predicción construidos pueden proporcionar resultados superiores. Esta posibilidad se fundamenta en el hecho de que durante su desarrollo solo se utilizó el año 2021, a diferencia e otros modelos que se basaron en tres años completos. Además, estos modelos incorporan una capa adicional que fortalece el conjunto, mejorando su rendimiento. Es destacable que la diferencia de pérdida entre las líneas de entrenamiento y validación no es significativa, indicando ausencia de sobreajuste y una adecuada configuración de los hiperparámetros.

Los tres modelos de prediccion desarrollados por medio de redes neuronales recurrentes RNN, LSTM, GRU mediante Machine Learning, obtuvieron una precisión del 85 % en el R-squared de cada uno. El mismo resultado se obtuvo utilizando intervalos de tiempo de 20 y 60 minutos. (Nawab et al., 2023), EL modelo se evaluó mediante mediante el coeficiente de determinación (R2), el error cuadrático medio (MSE), el error absoluto medio (MAE), de 0,93, 0,008, 0,17, respectivamente, en la Universidad de Duzce (Turquía). (Zhao et al., 2024), Un conjunto de árboles/regresión de bosque aleatorio (RF), haciendo uso de redes neuronales. Los modelos se evaluaron mediante el coeficiente de determinación (R2), el error cuadrático medio (MSE), el error absoluto medio (MAE), de 0,99, 40,083, 5,575, respectivamente.

Los modelos RNN, LSTM y GRU presentan un rendimiento superior en el R-squared, superando el 85 % de precisión de 5 minutos, como se detalla en la Tabla 9.

Estos modelos emergen como las opciones más adecuadas para este lugar, considerando la marcada variabilidad climática de Loja. La elección de estos modelos, junto con la configuración específica de sus hiperparámetros para este intervalo de tiempo, se sugiere como la estrategia más acertada.

8. Conclusiones

- El desarrollo del modelo de predicción de irradiación solar basado en Machine Learning para la FEIRNR refleja un progreso significativo en la optimización de la gestión de los recursos energéticos en este contexto particular. La implementación de varias redes neuronales recurrentes, como lo son las RNN, LSTM y GRU, ha dado como resultado modelos estables adaptados a la variación climática en la ciudad de Loja. La selección cuidadosa y el ajuste de los hiperparámetros, así como la inclusión de las variables correspondientes, han contribuido a un rendimiento único, especialmente destacado en el caso de 5 minutos.
- Usando técnicas estadísticas, se obtuvo un valioso conocimiento de los patrones y variaciones de la irradiación solar en el área alrededor de FEIRNNR. Este análisis estadístico ha creado una base estable que admite tanto el diseño como la implementación de los modelos de irradiación solar basados en Machine Learning. La integración de estos métodos estadísticos ha mejorado nuestra capacidad para beneficiarnos completamente de los datos disponibles y, por lo tanto, establecer las bases para un enfoque más preciso y eficiente para la planificación y el uso de energía solar en este contexto especial.
- La ejecución de estos modelos han demostrado ser una herramienta bien valiosa para anticipar y comprender los patrones de la irradiación solar, que facilita la toma de decisiones, ya que se basa más en la planificación y el uso de la energía solar en este entorno particular. El uso exitoso de las técnicas de Machine Learning no solo ha mejorado la precisión de las predicciones, sino que también ha establecido una base para un enfoque más sostenible y eficiente para la gestión de los recursos energéticos solares.
- La evaluación haciendo uso de las métricas establecidas ha proporcionado una comprensión detallada de la eficiencia y precisión de los modelos de irradiación solar en la FEIRNR, que destaca su rendimiento, superando el umbral del 85 % en cada uno de ellos. Estas evaluaciones han establecido una base sólida para toma de decisiones encaminada a la selección y arreglo de los modelos, que promueve un enfoque más estable y eficiente para la gestión de los recursos energéticos basados en la predicción de la irradiación solar.

9. Recomendaciones

- Ampliar la cantidad de datos de irradiación solar utilizados en el entrenamiento de los tres modelos desarrollados, estos ayudarán a mejorar significativamente su precisión. Este enfoque le permite optimizar el rendimiento de los modelos sin realizar cambios en su estructura actual.
- Continuar explorando y experimentando con varias arquitecturas de los modelos de Machine Learning, como lo son las redes neuronales recurrentes (RNN), Long Short-Term Memory (LSTM) y Gated Recurrent Unit (GRU). Refinar y ajustar los hiperparámetros de los modelos para mejorar aún más su rendimiento en la precisión.
- Evaluar la posibilidad de incluir más variables que pueden ayudar el pronóstico de la irradiación solar para mejorar la corpulencia del modelo. Averiguar la inclusión de las técnicas de ensemble learning para unir varios modelos y así mejorar la generalización del modelo.

10. Bibliografía

Alfadda, A., Rahman, S., & Pipattanasomporn, M. (2018). Solar irradiance forecast using aerosols measurements: A data driven approach. *Solar Energy*, *170*, 924-939. https://doi.org/https://doi.org/10.1016/j.solener.2018.05.089

Artega, P. (2009). ANÁLISIS DE GRÁFICOS ESTADÍSTICOS ELABORADOS EN UN PROYECTO DE ANÁLISIS DE DATOS, 21-23. https://www.ugr.es/~batanero/pages/ARTICULOS/trabajomasterPedro.pdf

Benítez, J. (2013). *SISTEMA DE PREDICCIÓN DEL RECURSO SOLAR APLICADO A CENTRALES TERMOSOLARES*. Universidad de Córdoba Campus de Rabanales, Edificio Leonardo Da Vinci 14071 CÓRDOBA (España). https://helvia.uco.es/xmlui/bitstream/handle/10396/12800/PFM_JavierRodriguezBenitez.pdf?sequence=1&isAllowed=y

Bobadilla, J. (2021). *Machine Learning y Deep Learning: Usando Python, Scikit y Keras*. Ediciones de la U. https://books.google.com.ec/books?id=iAAyEAAAQBAJ

Faisal, F., Rahman, A., Habib, M. T. M., Siddique, A. H., Hasan, M., & Khan, M. M. (2022). Neural networks based multivariate time series forecasting of solar radiation using meteorological data of different cities of Bangladesh. *Results in Engineering*, *13*, 100365. https://doi.org/https://doi.org/10.1016/j.rineng.2022.100365

Gallo, R., Castangia, M., Macii, A., Patti, E., & Aliberti, A. (2025). Neural networks for estimating surface solar irradiation from satellite images. *Engineering Applications of Artificial Intelligence*, *144*, 110101. https://doi.org/https://doi.org/10.1016/j.engappai.2025.110101

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (2.ª ed.). O'Reilly Media. http://14.139.161.31/OddSem-0822-1122/Hands-On_Machine_Learning_with_Scikit-Learn-Keras-and-TensorFlow-2nd-Edition-Aurelien-Geron.pdf

Instituto de Ingeniería del conocimiento. (2023). *Inteligencia Artificial*. https://www.iic.uam.es/

Jiménez, E. (2021). *Introducción al Machine Learning con MATLAB*. Marcombo. https://books.google.com.ec/books?id=Ek1OEAAAQBAJ

Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling*. Springer. https://vuquangnguyen2016.wordpress.com/wp-content/uploads/2018/03/applied-predictive-modeling-max-kuhn-kjell-johnson_1518.pdf

Lalaleo, D. (2021). *DISEÑO DE UN ALGORITMO UTILIZANDO MACHINE LEARNING PARA LA PREDICCIÓN DE LA RADIACIÓN SOLAR EN EL SECTOR DE LASSO*. [PROYECTO DE INVESTIGACIÓN, UNIVERSIDAD TÉC-

NICA DE COTOPAXI] .repositorio.utc.edu.ec. https://repositorio.utc.edu.ec/bitstream/27000/8014/1/MUTC-001017.pdf

Maldonado, J., Valdiviezo, M., Viñan, M. S., Samaniego, C., & Rojas, M. (2021). Wind power forecasting for the Villonaco wind farm. *Wind Engineering*, *45*, 1145-1159. https://doi.org/10.1177/0309524X20968817

Maps, G. (2024). *Facultad de Energía* [Consultado el 8 de noviembre de 2024]. https://www.google.com/maps

Mughees, N., Jaffery, M. H., Mughees, A., Mughees, A., & Ejsmont, K. (2023). Bi-LSTM-Based Deep Stacked Sequence-to-Sequence Autoencoder for Forecasting Solar Irradiation and Wind Speed. *Computers, Materials and Continua*, *75*, 6375-6393. https://doi.org/https://doi.org/10.32604/cmc.2023.038564

Nawab, F., Abd Hamid, A. S., Ibrahim, A., Sopian, K., Fazlizan, A., & Fauzan, M. F. (2023). Solar irradiation prediction using empirical and artificial intelligence methods: A comparative review. *Heliyon*, *9*, e17038. https://doi.org/https://doi.org/10.1016/j.heliyon.2023.e17038

Planas, O. (2019). Irradiación solar. https://solar-energia.net/que-es-energia-solar/radiacion-solar/irradiacion-solar

Rodríguez, J. (2021). *Predicción de la radiación global horizontal mediante el uso de redes neuronales*. Universidad de Córdoba Campus de Rabanales, Edificio Leonardo Da Vinci 14071 CÓRDOBA (España). https://idus.us.es/bitstream/handle/11441/126605/TFM-2036-RODRIGUEZ%20LUCENA.pdf?sequence=1&isAllowed=y

Saabith, A. L. S., Vinothraj, T., & Fareez, M. (2020). POPULAR PYTHON LIBRARIES AND THEIR APPLICATION DOMAINS. *International Journal of Advance Engineering and Research Development*, *7*. https://www.researchgate.net/publication/349828209

Sandoval, L. J. (2018). ALGORITMOS DE APRENDIZAJE AUTOMÁTICO PARA ANÁLISIS Y PREDICCIÓN DE DATOS. *REVISTA TECNOLÓGICA*. http://www.redicces.org.sv/jspui/bitstream/10972/3626/1/Art6_RT2018.pdf

Sehrawat, N., Vashisht, S., & Singh, A. (2023). Solar irradiance forecasting models using machine learning techniques and digital twin: A case study with comparison. *International Journal of Intelligent Networks*, *4*, 90-102. https://doi.org/https://doi.org/10.1016/j.ijin.2023.04.001

Tian, J., Ooka, R., & Lee, D. (2023). Multi-scale solar radiation and photovoltaic power forecasting with machine learning algorithms in urban environment: A state-of-the-art review. *Journal of Cleaner Production*, *426*, 139040. https://doi.org/https://doi.org/10.1016/j.jclepro.2023.139040

Violeta, V. A. (2004). Data Mining y el descubrimiento del conocimiento. *Industrial Data*. https://www.redalyc.org/articulo.oa?id=81670213

Witten, I., & Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. Elsevier Science. https://books.google.com.ec/books?id=QTnOcZJzlUoC

Zhao, H., Liu, C., Jing Yang, R., & Sun, C. (2024). Large-scale prediction of solar irradiation, shading impacts, and energy generation on building Façade through urban morphological indicators: A machine learning approach. *Energy and Buildings*, *323*, 114797. https://doi.org/https://doi.org/10.1016/j.enbuild.2024.114797

11. Anexos

Anexo 1. Gráfico de Resultados.

Código en Python.

```
# Graficar resultados
plt.figure(figsize=(18, 6))
plt.plot(y_test_denormalized, label='Datos reales', color='blue')
plt.plot(predictions_denormalized, label='Predicciones',
color='red')
plt.title('Predicciones vs Datos reales')
plt.xlabel('Muestras')
plt.ylabel('IRRADIACIÓN GLOBAL PROMEDIO (Wh/m^2)')
plt.legend()
plt.show()
```

- **plt.figure(figsize=(18, 6)):** Crea un gráfico para la visualización con un tamaño de 18 unidades de ancho y 6 unidades de alto.
- **plt.plot(y_test_denormalized, label='Datos reales', color='blue'):** Agrega una línea de trazado para los datos reales del conjunto de prueba (y_test_denormalized). Se detalla que el color de la línea es azul.
- **plt.plot(predictions_denormalized, label = 'Predicciones', color = 'red'):** Aumenta una línea de trazado para las predicciones del modelo (predictions_denormalized). Se detalla que el color de la línea es rojo.
- **plt.title('Predicciones vs Datos reales'):** Crea el título de la gráfica como "Predicciones vs Datos reales".
- **plt.xlabel('Muestras'):** Etiqueta el eje x de la gráfica como "Muestras".
- **plt.ylabel('IRRADIACIÓN GLOBAL PROMEDIO (Wh/m^2)′)** : Etiqueta el eje y de la gráfica como "IRRADIACIÓN GLOBAL PROMEDIO (Wh/m^2)".
- **plt.legend():** Muestra una leyenda en la gráfica para identificar las líneas de datos reales y predicciones.
- **plt.show():** Enseña la gráfica.

Anexo 2. Gráfico de la Pérdida de entrenamiento y validación a lo largo de las épocas.

Código en Python.

```
# Pérdida de entrenamiento
plt.plot(history.history['loss'], label='Entrenamiento',
color='blue')
# Pérdida de validación
plt.plot(history.history['val_loss'], label='Validación',
color='red')
plt.title('Pérdida de entrenamiento y validación a lo largo de
las épocas')
plt.xlabel('Épocas')
plt.ylabel('Pérdida (MSE)')
plt.legend()
plt.show()
```

- **plt.plot(history.history['loss'], label='Entrenamiento', color='blue'):** Grafica la pérdida de entrenamiento en función del número de épocas. history.history['loss'] incluye la pérdida calculada en cada época durante el entrenamiento. Se muestra con una línea azul en el gráfico y se etiqueta como "Entrenamiento".
- **plt.plot(history.history['val_loss'], label ='Validación', color ='red'):** Grafica la pérdida de validación en función del número de épocas. history.history ['val_-loss'] incluye la pérdida calculada en cada época durante la validación. Se muestra con una línea roja en el gráfico y se etiqueta como "Validación".
- **plt.title('Pérdida de entrenamiento y validación a lo largo de las épocas'):** Crea el título del gráfico como "Pérdida de entrenamiento y validación a lo largo de las épocas".
- **plt.xlabel('Épocas'):** Etiqueta el eje x del gráfico con "Épocas", mostrando el número de veces que el modelo ha pasado por todo el conjunto de datos de entrenamiento durante el entrenamiento.
- **plt.ylabel('Pérdida (MSE)'):** Etiqueta el eje y del gráfico con "Pérdida (MSE)", mostrando que la pérdida se mide utilizando el Error Cuadrático Medio.
- **plt.legend():** Aumenta una leyenda al gráfico para diferenciar entre las líneas de entrenamiento y validación.
- **plt.show():** Enseña el gráfico. Este comando suele ser útil para visualizar la representación gráfica en la interfaz de usuario.

Anexo 3. Solicitud de adquisición de los datos de irradiación solar al CITE.

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Loja, 08 de noviembre de 2023

Sr. Ing.
Juan Carlos Solano, PhD.
DIRECTOR DEL PROYECTO: DESARROLLO DE UN SISTEMA DE SOPORTE DE DECICIONES PARA EL AUTOCONSUMO FOTOVOLTAICO EN EL ECUADOR. CASO PRÁCTICO EN LA REGIÓN SUR.

Ciudad. –

De mi consideración:

Ciudad. –

De mi especial consideración:

Yo, **Jhandry Juan Tapia Sarango**, portador de la cédula de identidad nro. **1104983604**; solicito muy comedidamente a usted, se me haga la entrega los datos del **recurso solar**, con la finalidad de realizar mi tema de tesis, el cual adjunto su nombre **"MODELO DE PREDICCIÓN DE IRRADIACIÓN SOLAR BASADO EN MACHINE LEARNING PARA LA FACULTAD DE LA ENERGÍA, LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES DE LA UNIVERSIDAD NACIONAL DE LOJA".**

Esperando se sirva disponer el preferente y oportuno trámite, me anticipo en presentarle mis cumplidos agradecimientos.

Por la atención prestada de antemano le expreso mi más sincero agradecimiento.

Atentamente,

……

Estudiante: Jhandry Juan Tapia Sarango
Cl: 1104983604
Teléfono: 0993069043
Correo: Jhandry.tapia@unl.edu.ec

Anexo 4. Certificación de traduación del resumen.

Juan Pablo Ordóñez Salazar

CELTA-Certified English Teacher, traductor e intérprete.

Certificación de traducción al idioma inglés.

JUAN PABLO ORDÓÑEZ SALAZAR.

CELTA-certified English teacher, traductor e intérprete.

CERTIFICA:

Que la presente traducción de español a inglés del resumen del Trabajo de Integración Curricular previo a la obtención del título de Ingeniero Electromecánico titulado **"Modelo de predicción de irradiación solar basado en Machine Learning para la Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables de la Universidad Nacional de Loja"**, de autoría del estudiante **Jhandry Juan Tapia Sarango**, portador de la cédula de identidad número **1104983604**, estudiante de la Carrera de Ingeniería Electromecánica de la Facultad de la Energía, las Industrias y los Recursos Naturales No Renovables, Carrera de Electromecánica de la Universidad Nacional de Loja, fue realizada y revisada por Juan Pablo Ordóñez Salazar, perito traductor e intérprete del Consejo de la Judicatura, con certificado número 12298374.

Lo certifico en honor a la verdad, y autorizo al interesado hacer uso del presente en lo que a sus intereses convenga.

Loja, 21 de abril del 2025

Juan Pablo Ordóñez Salazar

DNI: 110360109-0

Código de Perito de la Judicatura: 12298374

Celular: +593 994290147

CELTA – CERTIFIED ENGLISH TEACHER, TRADUCTOR E INTÉRPRETE

juanpabloorsal@gmail.com | O99-429-0147 | 717-53 Miguel Morelos St., Loja- EC 11O111