ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Estimación de la demanda de agua en cultivos de banano aplicando modelos de pronósticos para fincas de pequeños y medianos productores en la provincia de El Oro.

PROYECTO DE TITULACIÓN

Previo la obtención del Título de:

Magister en Ciencias de Datos

Presentado por:

Calvache Silvestre Andrés Francisco

Roldán Carranza Daniela Alexandra

GUAYAQUIL - ECUADOR

Año: 2024

DEDICATORIA

Dedico este trabajo a mi papá (†) y mamá, cuyo amor, sacrificio y apoyo incondicional me han guiado a lo largo de mi camino profesional y personal.

Andrés Calvache

A mi madre y mi tía (†) quienes me enseñaron que, con perseverancia y disciplina, lo podemos lograr todo.

Daniela Roldán.

AGRADECIMIENTOS

A mi madre, por su amor incondicional. A mi padre, por su guía eterna. A mi familia y amigos, por el apoyo recibido y por estar siempre en todo momento. A mis profesores, por impartir con paciencia y entusiasmo sus conocimientos.

Andrés Calvache

A mi familia y amigos por el soporte en este proceso. Al Observatorio Estadístico de Banano y AEBE por la apertura para desarrollar este proyecto integrador.

Finalmente, a los profesores de la MCD por la transferencia de conocimientos, de especial manera, nuestro tutor, por el apoyo en el desarrollo de este trabajo.

Daniela Roldán

DECLARACIÓN EXPRESA

“Los derechos de titularidad y explotación, nos corresponden conforme al reglamento de propiedad intelectual de la institución; *Calvache Silvestre Andrés Francisco; Roldán Carranza Daniela Alexandra,* damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual”.

Firmado electrónicamente por:
ANDRES FRANCISCO
CALVACHE SILVESTRE

Andrés F. Calvache Silvestre

Firmado electrónicamente por:
DANIELA ALEXANDRA
ROLDAN CARRANZA

Daniela A. Roldán Carranza

EVALUADORES

Firmado electrónicamente por:
EDUARDO SEGUNDO
CRUZ RAMIREZ

MsC. Eduardo Segundo Cruz Ramírez
TUTOR

Firmado electrónicamente por:
ALLAN ROBERTO
AVENDANO SUDARIO

MsC. Allan Roberto Avendaño Sudario
REVISOR

RESUMEN

La gestión eficiente del agua es crucial para la producción agrícola, especialmente en cultivos como el banano de exportación. La agricultura de precisión busca optimizar el uso de recursos naturales y químicos, pero su costo limita su acceso a pequeños productores.

En este contexto, el presente trabajo estimó la evapotranspiración (ETo) en cultivos de banano en la provincia de El Oro. Para ello, se analizó las tendencias climáticas y patrones relacionados con la ETo. Luego, se estimaron modelos de series de tiempo y redes neuronales LSTM para pronosticar la demanda hídrica. Una vez evaluado los criterios de ajustes, los resultados indican que las redes neuronales LSTM tienen un MAE inferior al 5%, por lo que se seleccionó este mecanismo para hacer el pronóstico de la demanda hídrica para los cultivos del cantón El Guabo en la provincia de El Oro.

Finalmente, se presentan los resultados en un *dashboard* donde se muestra estadísticas climáticas y estimaciones de consumo de agua por hectárea.

Palabras Clave: Evapotranspiración, redes neuronales, ARIMA, MAE, banano.

.

ABSTRACT

Efficient water management is critical for agricultural production, particularly in crops like export bananas. Precision agriculture aims to optimize the use of natural and chemical resources, but its cost remains a barrier for small-scale farmers.

In this context, our study estimated evapotranspiration (ETo) in banana crops in El Oro province. We analyzed climatic trends and patterns related to ETo. Subsequently, we applied time series models and Long Short-Term Memory (LSTM) neural networks to forecast water demand. After evaluating model performance, LSTM neural networks demonstrated an MAE below 5%, leading us to select this approach for predicting water demand in El Guabo canton, El Oro province.

Finally, we present the results through a dashboard displaying climatic statistics and water consumption estimates per hectare.

Keywords: *Evapotranspiration, neural networks, ARIMA, MAE, banana.*

ÍNDICE GENERAL

ABREVIATURAS

ETo	Evapotranspiración del cultivo de referencia
LSTM	Long-Short Term Memory
ETL	Extract, Transform and Load
SARIMA	Seasonal Autoregressive Integrated Moving Average
CNN	Convolutional neural network
LASSO	Least Absolute Shrinkage and Selection Operator
MODIS	Moderate Resolution Imaging Spectroradiometer
ADF	Augmented Dickey-Fuller
RMSE	Root Mean Square Error
MASE	Mean Absolute Scale Error

INDICE DE GRÁFICOS

ÍNDICE DE TABLAS

CAPÍTULO 1

1. PLANTEAMIENTO DEL PROBLEMA

La gestión eficiente del agua es uno de los retos en la agricultura, de especial manera en el cultivo de banano, donde la sobreutilización o subutilización de este recurso natural puede tener diferentes consecuencias en la calidad del banano. Cuando esto ocurre, se producen pérdidas económicas a todo nivel, pues el banano debe cumplir rigurosos estándares de calidad para ser exportados a las regiones de Europa, Estados Unidos y Rusia.

En este sentido, la adopción de tecnologías para optimizar el uso de los recursos naturales y químicos necesarios para la producción de banano es importante para minimizar el riesgo de efectos negativos y, consecuentemente, pérdidas económicas generadas por una producción de mala calidad.

1.1 Descripción del problema

La falta de un manejo adecuado de los recursos hídricos puede tener graves consecuencias en la producción de banano, ya que el agua es un recurso crítico para el crecimiento y desarrollo de la planta. La sobreutilización de agua en los cultivos puede generar pérdida de nutrientes, disminución de la calidad del fruto, disminución del rendimiento y daños en el suelo. Por otro lado, la subutilización del agua puede limitar el crecimiento de las plantas y la producción de frutas.

La región de El Oro enfrenta una problemática en la estimación de la demanda hídrica para los cultivos de banano. Los pequeños productores de la zona carecen de recursos financieros y conocimientos técnicos para emplear herramientas avanzadas de análisis de datos.

La falta de acceso a herramientas avanzadas de análisis de datos limita la gestión eficiente del agua en los cultivos de banano, lo que puede afectar negativamente la producción y calidad. Es importante considerar las limitaciones de los pequeños productores de la zona para implementar prácticas de agricultura inteligente. Por ello, se requiere un esfuerzo conjunto entre los productores, las autoridades locales y otras instituciones para promover y facilitar el acceso a estas herramientas, capacitaciones y financiamiento.

Además de las limitaciones de los pequeños productores, también existen otros factores que pueden afectar la implementación de prácticas de agricultura inteligente

en la zona. Por ejemplo, la falta de infraestructura y tecnología adecuada puede limitar la recolección y el análisis de datos, así como la comunicación y el acceso a información relevante. Asimismo, la falta de políticas y regulaciones claras en relación con el manejo del agua y la agricultura pueden generar incertidumbre y dificultades para la toma de decisiones.

Por tanto, resulta relevante establecer un mecanismo de acción para superar las barreras que suponen dichas limitaciones, partiendo desde el uso de datos libres hasta la estimación de la demanda hídrica de sus cultivos. De esta forma, se agrega valor al fomentar una cultura de gestión eficiente de los recursos hídricos en los cultivos de banano en la provincia de El Oro, lo que puede generar beneficios económicos, sociales y ambientales a largo plazo.

1.2 Justificación

El banano ecuatoriano es el segundo producto de exportación que mueve el ingreso de las exportaciones no petroleras del país. El ingreso FOB de este producto a diciembre de 2022 alcanzó los 3.144 millones de dólares, flujo monetario que corresponde al envío de 351 millones de cajas de banano a 73 destinos internacionales, esto según las cifras reportadas por el Banco Central de Ecuador.

A nivel de producción, según el último informe presentado por el Instituto Nacional de Estadísticas y Censos (INEC), el hectareaje nacional dedicado a cultivos permanentes alcanzó las 901 mil has, de ellas, el banano representa 12 % de la superficie plantada. La producción de banano se concentra principalmente en las provincias de Los Ríos (37.50 %), Guayas (28 %) y El Oro (25.10 %). A nivel de rendimientos productivos, Los Ríos y Guayas producen alrededor de 2.200 cajas por hectárea anualmente, mientras que El Oro alcanza un promedio de 1.600 cajas anuales en este indicador, ubicándose 600 mil cajas por debajo de sus principales provincias competidoras en producción nacional.

El rendimiento productivo en el cultivo de banano depende de diferentes factores, entre los principales se encuentran el clima, la aplicación de buenas prácticas agrícola, el control fitosanitario y el control de riego de la plantación. Dependiendo de la gestión dada a los factores mencionados, el rendimiento productivo de las plantaciones puede variar. Dentro de este orden de factores, no existe un consenso en formulas o ranking para la administración de los tópicos indicados, pues su control y aplicación dependen en gran medida de la experticia y recursos económicos disponibles del productor.

La gestión adecuada de los recursos hídricos en la agricultura es fundamental para maximizar la producción de cultivos, garantizar la seguridad alimentaria y preservar el medio ambiente. Los pequeños productores de banano enfrentan limitaciones para acceder a información relevante y oportuna sobre la cantidad de agua que sus cultivos necesitan, lo que resulta en un uso ineficiente del recurso y una menor producción.

El desarrollo de un modelo de pronósticos para la gestión de uno de los componentes de recursos hídricos, en este caso la evapotranspiración, puede contribuir significativamente a resolver esta limitación de los pequeños productores. El modelo proporcionará la cantidad de agua que podría necesitarse en el cultivo, tal que el productor implemente acciones preventivas dentro de la planificación de riego.

Además, el uso de tecnologías innovadoras como la agricultura inteligente, que integra herramientas digitales y modelos de aprendizaje profundo, así como estadísticos, puede ser una solución asequible y efectiva para los pequeños productores. Al permitirles acceder a información oportuna sin incurrir en costos significativos, podrán mejorar su productividad y sostenibilidad, lo que a su vez contribuirá al desarrollo económico y social de las zonas rurales donde se encuentran estos cultivos de banano.

1.3 Solución Propuesta

Ahora bien, una vez explicada justificada la importancia de abordar con agricultura inteligente la problemática planteada en esta investigación, se propone tomar el siguiente curso de acción para solucionar la planificación de demanda de agua en los cultivos de banano.

El trabajo se dividirá en tres etapas. La primera, se enfoca en la extracción, transformación y carga (ETL, por sus siglas en inglés) de los datos. Los datos que se emplearán provienen de bases de datos estructurados, pues corresponden tanto a coordenadas geográficas y datos climatológicos. En ambos casos, se realizará un proceso de revisión y depuración para garantizar la calidad de *inputs* a los modelos a desarrollar.

Luego, la segunda etapa del proyecto consiste la estimación de los modelos de demanda de agua. Proceso que se realizará en local, empleando la herramienta de *Google Colaboratory*, donde se entrenará tanto la red LSTM y el ARIMA. Finalmente, el proceso cierra en la tercera etapa con la evaluación del pronóstico de ambos

modelos. Se elegirá el que tenga menor margen de error, una vez que se haya realizado la selección, y los resultados serán presentados en una visualización en Power BI para que la empresa pueda hacer uso de la información.

1.4 Objetivos

1.4.1 Objetivo General

Estimar el factor de evapotranspiración a través de modelos de pronósticos estadísticos y de aprendizaje profundo como aproximación de la demanda hídrica en cultivos de banano en la provincia de El Oro.

1.4.2 Objetivos Específicos

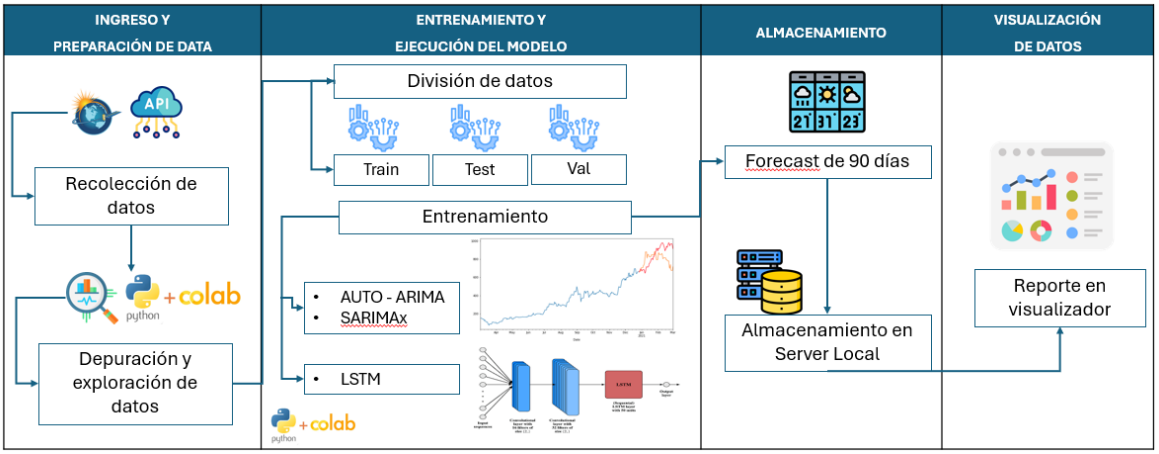
1. Analizar las tendencias y patrones de los factores climatológicos que componen el indicador de evapotranspiración para la comprensión del comportamiento del uso del agua en el cultivo.
2. Pronosticar el factor de evapotranspiración empleando los modelos estadísticos y de aprendizaje profundo.
3. Evaluar la precisión de los modelos aplicados para la aproximación de la demanda hídrica con aquel que tenga el mejor ajuste.

1.5 Metodología

El presente trabajo aspira desarrollar un modelo de pronóstico del Factor de Evapotranspiración (ETo) empleando métodos estadísticos, como el modelo ARIMA, así también, métodos de aprendizaje profundo como las redes neuronales denominadas “Long-Short Term Memory” (LSTM). Se comparará ambos modelos de pronósticos con el propósito de obtener la mayor precisión en la predicción del factor.

Al implementar el modelo de pronóstico del ETo, se puede planificar la demanda de agua y riego para el cultivo de acuerdo con la necesidad que muestre la proyección climatológica. Por otro lado, para garantizar la continuidad del modelo. este debe ser actualizado cada cierto tiempo, considerando la estacionalidad de las series.

La arquitectura de datos que se empleará es la siguiente:



1.6 Resultados esperados

Los pronósticos generados por el modelo serán presentados en Power BI para que la empresa pueda utilizar dicha información según su necesidad de consulta. Además, el código será entregado a la empresa para que pueda generar las actualizaciones pertinentes con su departamento de TI.

1.7 Dataset

La información que se empleará en este trabajo tiene dos componentes. Primero tenemos las coordenadas de latitud y longitud de las estaciones meteorológicas donde se encuentran instaladas las estaciones meteorológicas. Estas son proporcionadas por el Observatorio Estadístico de Banano. Luego están los datos climatológicos. Para efectos de este estudio, se emplearán los datos publicados por la Administración Nacional de Aeronáutica y el Espacio (N.A.S.A) en el portal POWER LARC. El horizonte de estudio de este trabajo es de cinco años con periodicidad diaria (2015 - 2023)

A continuación, se describen las variables empleadas en cada base de datos.

Fuente	Variable	Descripción
OEB	Latitud	Componente de coordenada
OEB	Longitud	Componente de coordenada
Power Larc	Irradiancia	Irradiancia descendente de onda corta de toda la superficie del cielo (MJ/m^2/día)
Power Larc	Irradiancia atmosférica	Irradiancia descendente de onda corta en la parte superior de la atmósfera (MJ/m^2/día)
Power Larc	Temperatura	Temperatura a 2 Metros (C) (Máximo, promedio y mínimo)
Power Larc	Humedad	Humedad Relativa a 2 Metros (%)
Power Larc	Viento	Velocidad del viento a 2 metros (m/s)

CAPÍTULO 2

2. ESTADO DEL ARTE

Este capítulo detalla un marco conceptual y referencial de artículos científicos relacionadas con la problemática a resolver. Se especifican las metodologías a utilizar para el preprocesamiento y análisis de datos que se necesitan para estimar los recursos necesarios en los cultivos ante variaciones atmosféricas.

2.1 Marco conceptual

2.1.1 Agricultura de precisión

La agricultura de precisión, también conocida como agricultura de precisión, es un enfoque innovador que utiliza tecnologías como el GPS y la digitalización para optimizar diversos aspectos de la producción de cultivos, como la aplicación de fertilizantes, la gestión de plagas y el uso del agua (Demestichas, Peppes, & Alexakis, 2020). Esta estrategia implica recopilar, procesar y analizar datos para tomar decisiones informadas que mejoren la utilización de los recursos, la productividad, la calidad, la rentabilidad y la sostenibilidad de la agricultura (Monteleone, y otros, 2020). El principio básico de la agricultura de precisión es aplicar el tratamiento adecuado en el lugar adecuado, en el momento adecuado y en la cantidad adecuada (St-Denis, Kneeshaw, & Messier, 2018). Se basa en información espacial detallada, tecnología de la información avanzada y sistemas de apoyo a la toma de decisiones para lograr sus objetivos (Neupane & Guo, 2019).

La agricultura de precisión, a veces denominada gestión específica del sitio, es una tecnología emergente que permite realizar ajustes para abordar la variabilidad dentro del campo en características como la fertilidad del suelo, la humedad del suelo, la intensidad de las malezas y la infestación de plagas de insectos (Demestichas, Peppes, & Alexakis, 2020). La tecnología tiene el potencial de reducir los costos de producción mediante una aplicación más eficiente y efectiva de insumos agrícolas. También reduce la degradación ambiental al permitir que los agricultores apliquen insumos agrícolas en proporciones adecuadas en los lugares donde sean necesarios (Monteleone, y otros, 2020). Los aspectos espaciales, temporales y predictivos de la variabilidad del suelo y de los cultivos son elementos vitales de la agricultura de precisión. Implica el muestreo, mapeo, análisis y manejo de áreas específicas dentro de un campo en reconocimiento de la variabilidad

espacial y temporal con respecto a la fertilidad del suelo, la disponibilidad de humedad, las características de los cultivos y la población de plagas de insectos (Monteleone, y otros, 2020). Durante mucho tiempo se ha encontrado una variabilidad a gran escala en diferentes prácticas de cultivo en diferentes regiones. Sin embargo, la agricultura de precisión responde a la variabilidad espacial dentro de campos o huertos individuales (Neupane & Guo, 2019).

La definición de la agricultura de precisión ha ido cambiando con el paso del tiempo y los avances tecnológicos (McBratney, Whelan, Ancev, & Bouma, 2005). En este sentido, abarca prácticas agrícolas que buscan optimizar las operaciones basados en características específicas del lugar (Shrestha & Khanal, 2020). La agricultura de precisión integra distintas tecnologías, tales como sistemas GPS, microordenadores y sistemas autónomos con el fin de mejorar la precisión de la administración de los recursos (Krupitzer & Stein, 2021). La agricultura de precisión no solo se limita a las prácticas agrícolas, sino que abarca otros sectores como la ganadería, lo que permite la recolección de información precisa para la toma de decisiones (Subach & Shmeleva, 2022).

En adición, la agricultura de precisión se enfoca en el manejo de las fluctuaciones espaciales y temporales vinculados con la producción agrícola para maximizar la rentabilidad y reducir impactos negativos al medio ambiente (Lawler, 2016). El objetivo principal de la agricultura de precisión es el manejo óptimo de los recursos y aumentar la calidad de los productos (Duerfeldt, 2011). Por medio de la adopción de estas prácticas, los agricultores pueden aumentar sus niveles de productividad.

2.1.2 Modelos de pronósticos

2.1.2.1 Modelos ARIMA, SARIMA y X-11

Los modelos de previsión ARIMA, SARIMA y X-11 se utilizan habitualmente en diversos campos para el análisis y la previsión de series temporales. Los modelos ARIMA explica la serie temporal considerada sobre la base de sus valores anteriores, es decir, sus rezagos y la predicción rezagada errores. Puede ser útil para la previsión futura de una situación no Serie de tiempo estacionaria que exhibe patrones y no es irregular. ruido blanco. Los 3 términos característicos del modelo ARIMA. son los parámetros (p, d, q) donde, cada uno de los términos son los órdenes del término AR, la diferenciación necesaria para cambiar la serie de tiempo a una estacionaria y el término MA respectivamente. Estos modelos son conocidos por su precisión y robustes estadística (Contreras, Espínola, Nogales, & Conejo,

2003). Además, los modelos ARIMA se han combinado con otras técnicas como GARCH para analizar la fluctuación de los precios internacionales del petróleo, lo que demuestra la adaptabilidad de estos modelos a diferentes contextos (Xiang, 2022).

Para construir un modelo de predicción son necesarias las series temporales estacionarias. Para eliminar la no estacionariedad de una serie, comúnmente se realiza la diferenciación. A veces, si la serie temporal es más compleja, puede ser necesaria más de una operación de diferencia. Por lo tanto, el valor de diferencia "d" es el número mínimo de diferenciaciones necesarias para convertir la serie temporal en estacionaria. El valor d sería 0, si la serie ya es estacionaria. Si una serie de tiempo es univariada y contiene componentes de tendencia y/o estacionales, entonces se utiliza el modelo ARIMA estacional (SARIMA) (Kam, Sung, & Park, 2010).

El procedimiento X11, una adaptación del programa de ajuste estacional X-11 de la Oficina del Censo de EE. UU., ajusta estacionalmente series temporales mensuales o trimestrales (Miao, Wang, & Khan, 2016). El procedimiento realiza ajustes aditivos o multiplicativos y crea un conjunto de datos de salida que contiene las series temporales ajustadas y los cálculos intermedios. El uso más común del procedimiento X11 es producir una serie desestacionalizada (Zhou, Sun, Sun, & Yang, 2015). Eliminar el componente estacional de una serie económica facilita la comparación entre meses o trimestres consecutivos. Un gráfico de la serie desestacionalizada suele ofrecer más información sobre las tendencias o la ubicación en un ciclo económico que un gráfico de la serie no ajustada (Najamuddin & Fatima, 2022).

2.1.2.2 Long Short-Term Memory (LSTM)

Las redes neuronales de memoria a largo plazo (LSTM) son un tipo de arquitectura de red neuronal recurrente (RNN) diseñada para abordar los retos del aprendizaje de dependencias a largo plazo en datos secuenciales. Las LSTM están equipadas con una célula de memoria que puede mantener la información durante periodos prolongados, lo que les permite capturar y recordar patrones en datos de series temporales (Sak, 2014). Esta capacidad de retener información durante secuencias largas hace que las LSTM sean particularmente eficaces para tareas que implican dependencias temporales, como el reconocimiento del habla, el reconocimiento de la escritura a mano y la predicción de series temporales.

La red de memoria a largo plazo (LSTM) es un tipo de RNN que se utiliza en DL, ya que se pueden entrenar grandes conjuntos de datos para obtener grandes precisiones. Este modelo tiene un mecanismo de aprendizaje para memorizar y comprender el mapeo de variables de entrada a variables de salida y determina qué contexto derivado de los datos de entrada es útil para realizar el mapeo y podría alterar dinámicamente el contexto según sea necesario.

La estructura única de las LSTM les permite mitigar los problemas de gradiente de fuga y explosión que se encuentran comúnmente en las RNN tradicionales, lo que las hace muy adecuadas para modelar datos secuenciales complejos. Al incorporar puertas que regulan el flujo de información dentro y fuera de la célula de memoria, las LSTM pueden retener o descartar selectivamente información en cada paso temporal, facilitando el aprendizaje de dependencias de largo alcance.

Los investigadores han aprovechado las LSTM en varios dominios, como el reconocimiento de actividades, las predicciones del mercado financiero y la predicción energética, mostrando su versatilidad y eficacia en diversas aplicaciones (Yu, 2022; Fischer & Krauß, 2018; Mekruksavanich, Jitpattanakul, Youplao, & Yupapin, 2020). Además, avances como la combinación de LSTM con redes neuronales convolucionales (CNN) han mejorado aún más su rendimiento en tareas como la estimación del estado de carga de las baterías y el reconocimiento de actividades (Liu & Li, 2017).

2.1.2.3 Random Forest

Los bosques aleatorios son una modificación significativa del método de embolsado, que crea una gran cantidad de árboles y luego los promedia. En muchas tareas, los bosques aleatorios son muy similares al método de impulso y son más fáciles de entrenar y configurar (Wang, Chen, Song, & Qi, 2023). Son una herramienta eficaz en la predicción. Gracias a la ley de los grandes números, no se desbordan. Introducir el tipo correcto de aleatoriedad los convierte en clasificadores y regresores precisos. Además, la estructura en términos de la fuerza de los predictores individuales y su correlación da una idea de la capacidad del bosque aleatorio para predecir en el trabajo (Bhimanpallewar & Rao, 2021).

El algoritmo de bosque aleatorio es el mejor en términos de precisión entre los algoritmos modernos; funciona eficazmente en grandes bases de datos; puede manejar miles de variables de entrada sin eliminarlos; proporciona estimaciones de qué variables son importantes para las clasificaciones; genera una estimación

interna insesgada del error de generalización a medida que crece el bosque; tiene un método eficiente para estimar los datos faltantes y mantiene la precisión cuando faltan grandes porciones de datos; tiene métodos para igualar errores en conjuntos de datos de clases no balanceadas; se calculan prototipos que brindan información sobre la relación entre variables y clasificación; calcula la cercanía entre pares de casos, que se puede utilizar para agrupar, encontrar valores atípicos o (mediante escalamiento) proporcionar información interesante sobre los datos; ofrece un método experimental para detectar interacciones variables (Sanit-in & Saikaew, 2019; Shafique & Hato, 2014; Wang, Chen, Song, & Qi, 2023).

2.1.2.4 Regresión LASSO (Least Absolute Shrinkage and Selection Operator)

La correlación en el análisis de regresión es una correlación que tiene como objetivo predecir el valor de variables no independientes en función de las variables independientes. De hecho, una correlación entre variables no solo consta de dos variables, sino que puede haber una correlación entre tres o más variables llamada regresión lineal múltiple (Tibshirani, 1996). El análisis de regresión lineal múltiple tiene muchas variables independientes, por lo que existe una correlación entre dos o más variables independientes. Esta variable independiente que se correlaciona entre sí se llama multicolinealidad. El método de regresión LASSO (Operador de selección y contracción mínima absoluta) puede ayudar a reducir la multicolinealidad y aumentar la precisión de los modelos de regresión lineal. Es un método de análisis de regresión que realiza tanto la selección de variables como la regularización para mejorar la precisión de la predicción y la interpretabilidad del modelo estadístico que produce (Mao, y otros, 2021).

El LASSO es la técnica de operador de contracción más básica y mejor estudiada, reduce los coeficientes (parámetros) que se correlacionan con cero o cerca de cero, lo que da como resultados estimadores con variantes más pequeñas y un modelo final más representativo (Tibshirani, 1996). El método Lasso se dio a conocer después del algoritmo LAR en 2004. Las rutas de solución de LAR son lineales por partes y, por lo tanto, se pueden calcular de manera muy eficiente (Ranstam & Cook, 2018). La modificación de LAR (regresión de ángulo mínimo) para LASSO produce un algoritmo más eficiente para estimar la solución del estimador del coeficiente LASSO. El método LASSO puede reducir el coeficiente del método de mínimos cuadrados ordinarios a cero para poder seleccionar la variable fija. el

modelo producido por el método LASSO es más simple e indirectamente libre de multicolinealidad (Alhamzawi, Yu, & Benoit, 2012).

2.1.2.5 Redes Neuronales Artificiales (RNA)

Una neurona artificial es un modelo computacional inspirado en las neuronas naturales. Las neuronas naturales reciben señales a través de sinapsis ubicadas en las dendritas o membrana de la neurona. Cuando las señales recibidas son lo suficientemente fuertes (superan un cierto umbral), la neurona se activa y emite una señal a través del axón. Esta señal podría enviarse a otra sinapsis y activar otras neuronas (Hill, O'Connor, & Remus, 1996).

La complejidad de las neuronas reales queda muy abstraída al modelar neuronas artificiales. Estos consisten básicamente en entradas (como sinapsis), que se multiplican por pesos (fuerza de las señales respectivas) y luego se calculan mediante una función matemática que determina la activación de la neurona. Otra función (que puede ser la identidad) calcula la salida de la neurona artificial (a veces dependiendo de un cierto umbral). Las RNA combinan neuronas artificiales para procesar información.

Cuanto mayor sea el peso de una neurona artificial, más fuerte será la entrada que se multiplica por ella. Los pesos también pueden ser negativos, por lo que podemos decir que la señal es inhibida por el peso negativo. Dependiendo de los pesos, el cómputo de la neurona será diferente. Ajustando los pesos de una neurona artificial podemos obtener la salida que queremos para entradas específicas. Pero cuando tenemos una RNA de cientos o miles de neuronas, sería bastante complicado encontrar a mano todos los pesos necesarios. Pero podemos encontrar algoritmos que pueden ajustar los pesos de la ANN para obtener la salida deseada de la red. Este proceso de ajustar los pesos se llama aprendizaje o entrenamiento (Jentzen & Riekert, 2022).

2.1.3 Modelos de Evapotranspiración

Los modelos de evapotranspiración son cruciales en la agricultura de precisión para estimar la pérdida de agua por evaporación del suelo y transpiración de las plantas. Estos modelos ayudan a optimizar las prácticas de riego, gestionar eficazmente los recursos hídricos y predecir con exactitud las necesidades de agua de los cultivos. Se han desarrollado varios modelos teóricos y fórmulas de cálculo para estimar con precisión la evapotranspiración.

Un método muy utilizado para calcular la evapotranspiración es la ecuación de Penman-Monteith, recomendada por la Organización de las Naciones Unidas para la Agricultura y la Alimentación (FAO) Jin et al. (Jin, Shuyan, & Yin, 2022). Esta ecuación tiene en cuenta factores como la temperatura, la humedad, la velocidad del viento y la radiación solar para estimar la evapotranspiración potencial.

La ecuación de Penman-Monteith es una fórmula ampliamente utilizada para calcular la evapotranspiración (ET) a partir de datos meteorológicos. La versión simplificada de la ecuación de Penman-Monteith es la siguiente:

$$ET_o = \frac{0.408 \cdot \Delta \cdot (R_n - G) + \gamma \cdot \frac{900}{T + 273} \cdot u \cdot (e_s - e_a)}{\Delta + \gamma \cdot (1 + 0.34 \cdot u)}$$

Donde:

ET_o: $Evapotranspiración$ en $\frac{\text{mm}}{\text{día}}$

Δ: Pendiente de la curva de presión de vapor $(\text{kPa } °\text{C}^{-1})$

R_n: Radiación neta en la superficie $(\text{MJ } /\text{m}^2\text{día}^1)$

G: Flujo de calor del suelo $(\text{MJ } /\text{m}^2\text{día}^1)$

γ: Constante psicrométrica $(\text{kPa } °\text{C}^{-1})$

T: Temperatura media diaria del aire (°C)

u: velocidad del viento a 2 m de altura $\left(\frac{m}{\text{s}}\right)$

e_s: Presión de vapor de saturación (kPa)

e_a: Presión de vapor de actual (kPa)

Esta ecuación se utiliza para calcular la evapotranspiración de referencia (ET0), que es un indicador del consumo de agua de una superficie de césped hipotética que no está limitada por el agua. La ET0 se utiliza a menudo en la agricultura para estimar las necesidades de riego

2.1.4 MODIS

El sensor MODIS (Moderate Resolution Imaging Spectroradiometer) es un instrumento a bordo de los satélites de la NASA que recopila datos sobre la Tierra en una amplia variedad de bandas espectrales. Estos datos son utilizados para monitorear la cobertura terrestre, la temperatura de la superficie, la vegetación y

otros parámetros ambientales, lo que lo convierte en una fuente valiosa de información para el estudio del clima y la agricultura.

Los datos climatológicos de la NASA: son registros históricos y actuales de diversas variables climáticas, como temperaturas, precipitaciones, vientos y humedad, recopilados por la Administración Nacional de Aeronáutica y del Espacio de Estados Unidos (NASA). Estos datos son ampliamente utilizados en la investigación climática y en aplicaciones relacionadas con la agricultura y la gestión del agua. (NASA, s.f.)

2.2 Marco referencial

2.2.1 Modelos de predicción en la agricultura de precisión

Existen varios estudios relacionados con la aplicación de los modelos de pronósticos en la agricultura de precisión, abarcando distintos rubros como el manejo de cultivos, previsión de rendimientos, detección de enfermedades y control de medio ambiente. En este sentido, Sadiku et al. (2020), resaltaron la aplicación del big data en la agricultura inteligente y el uso de modelos predictivos. Por su parte, Bhimanpallewar y Rao (2021) estudiaron el uso de métodos de aprendizaje automático para la toma de decisiones, dentro de sus principales hallazgos, resaltaron la gestión de cultivos, suelo y agua.

Preethi et al. (2021) se enfocaron en la clasificación de maleza, detección de enfermedades foliares, control de drones mediante imágenes. El estudio resaltó el rol de la agricultura de precisión en el pronóstico de rendimientos de los cultivos como método de la optimización en los niveles de producción. Adicionalmente, Ruß y Brenning (2010), resaltaron el uso de variables espaciales para la previsión de niveles de rendimiento en cultivos agrícolas.

En otros estudios, se exploraron diferentes enfoques, como el uso de Redes Neuronales Artificiales (ANN) para predecir el rendimiento de la cebada de primavera y el trigo de invierno en la República Checa**Fuente especificada no válida.**. Se encontró que la precisión de la predicción mejoraba a medida que se incluían más entradas en las ANN, y ciertos períodos resultaron óptimos para la predicción del rendimiento.

Además, se investigó la integración de datos climáticos y satelitales en la predicción del rendimiento del trigo en Australia, utilizando métodos como Máquinas de Soporte Vectorial (SVM) y Bosques Aleatorios**Fuente especificada no válida.**. La

combinación de estos datos demostró una capacidad significativa para predecir el rendimiento del trigo antes de la cosecha, y los métodos de Aprendizaje Automático superaron a la regresión lineal en términos de rendimiento predictivo.

En los últimos años, se ha explorado el uso de redes neuronales LSTM y modelos de Aprendizaje Profundo para la predicción del rendimiento de cultivos, como el estudio de Mahdi et al. **Fuente especificada no válida.** en Bangladesh. Las LSTM son especialmente adecuadas para datos climáticos con propiedades temporales. Además, Cao et al. (2021b) compararon modelos de Aprendizaje Automático y Aprendizaje Profundo y encontraron que las redes LSTM superaron a otros en la predicción del rendimiento del arroz en China.

Por último, Ma et al. **Fuente especificada no válida.** presentaron un modelo llamado "the Stacked Sparse Auto-encoder (SSAE)" para estimar el rendimiento del arroz, utilizando datos climáticos y de imágenes satelitales. Este modelo superó a una red neuronal artificial (ANN) en términos de precisión, dependiendo de la combinación de datos seleccionada.

El clima, con eventos como temperaturas extremas y sequías, tiene un impacto sustancial en la producción de estos cultivos. El modelo de Random Forest se basa en la construcción de múltiples árboles de decisión y proporciona información confiable sobre la importancia de las variables, lo que permite estimar el error de manera efectiva. Los resultados mostraron que el modelo pudo predecir satisfactoriamente el rendimiento de los cultivos varios meses antes de la cosecha.

2.2.2 Aplicación de agricultura precisión en cultivos de banano

La agricultura de precisión se ha aplicado cada vez más en los cultivos de banano para mejorar la productividad, la sostenibilidad y la eficiencia de los recursos. Un estudio se centró en la monitorización de parámetros cruciales y el control de daños por salinidad en cultivos de banano utilizando redes de sensores inalámbricos (WSN) y el Internet de las cosas (IoT) Desai (Solomon, Zile, Swedberg, & McMurray, 2020). Mediante el desarrollo de un sistema de agricultura de precisión, la investigación tenía como objetivo mejorar el rendimiento de los cultivos de banano a través de la monitorización en tiempo real y las intervenciones oportunas basadas en conocimientos impulsados por datos.

Otro estudio destacó el uso de la agricultura de precisión para el control de plagas en cultivos orgánicos de banano (Manrique, Campos, Paiva, & Ipanaque, 2021).

Mediante el aprovechamiento de técnicas de aprendizaje automático, como la predicción de la incidencia de trips, se emplearon prácticas de agricultura de precisión para optimizar las estrategias de gestión de plagas en el cultivo de banano orgánico. Este enfoque demuestra el potencial de la agricultura de precisión para mejorar la salud y el rendimiento de los cultivos de una manera respetuosa con el medio ambiente.

Además, se ha explorado la aplicación de tecnologías de agricultura de precisión, como la fertilización de tasa variable, en sistemas de cultivo de banano (Wang, Hao, & Chen, 2015). Utilizando la tecnología del Sistema de Información Geográfica (SIG), los agricultores pueden optimizar las propiedades del suelo, ajustar los insumos de cultivo y mejorar la productividad del suelo, lo que conduce a una utilización eficiente de los recursos y a una producción sostenible de bananos.

2.2.3 Predicción de evapotranspiración en cultivos de banano

Según Vianny et al (2022), centraron su investigación en la optimización del sistema de gestión del agua y la energía en la agricultura actual, con un enfoque en la irrigación de precisión. Utilizando componentes del Internet de las Cosas (IoT) para observar diversas variables como la humedad del suelo, la temperatura del suelo, las condiciones climáticas y ambientales, se propuso un modelo híbrido para el sistema de riego que optimiza los requisitos y reduce la energía. Este modelo incluye K-Nearest Neighbors (KNN), Gradient Boosting-based Tree (GBT), Long Short-Term Memory (LSTM) y la correlación de rango de Spearman. Estas técnicas se utilizan para recopilar información de sensores cercanos, predecir valores reales basados en observaciones y realizar predicciones de series de tiempo. La implementación del trabajo propuesto se realizó con la ayuda de especies de banano en el período de tiempo entre febrero de 2020, diciembre de 2020 y 2021. Tras la implementación, se optimizó el 31.4% del requerimiento de agua para un solo árbol de banano en el período de tiempo de 2020, lo que redujo el uso excesivo de agua fresca y el desperdicio de energía.

Según Yohanani et al. (2022), se utilizó la evapotranspiración medida (LE) de plantaciones de banano en invernaderos para derivar y comparar dos tipos de modelos de aprendizaje automático: la red neuronal artificial (ANN) y la regresión lineal múltiple (MLR). Las mediciones se realizaron en dos plantaciones de banano en invernaderos similares durante dos temporadas consecutivas, 2016 y 2017. En la mayoría de los casos, el modelo ANN superó al MLR. Cuando se entrenaron y

validaron con el conjunto de datos completo de 2017, los modelos ANN y MLR proporcionaron R2 de 0.92 y 0.89, RMSE de 37.5 y 45.1 W m−2 y MAE de 21 y 27.2 W m−2, respectivamente. Los modelos podrían derivarse utilizando un conjunto de datos de entrenamiento tan corto como un mes y aún proporcionar estimaciones confiables. La radiación solar (Rg) fue la variable que más influyó en LE; usando Rg como la única variable de entrada, el modelo ANN resultó en R2, RMSE y MAE de 0.88, 47 W m−2 y 25.6 W m−2, respectivamente. Se concluye que los modelos ANN y MLR son opciones razonables para estimar la evapotranspiración en un invernadero de banano.

Según Jenitha & Rajesh (2024), utilizaron el algoritmo de memoria a largo plazo bidireccional profunda (DBLS-TM) para desarrollar una unidad de programación de riego. Se recopiló un conjunto de datos de la granja Virucham, en el distrito de Virudhunagar, Tamil Nadu, para entrenar la máquina. Basándose en estos datos, el modelo se entrenó y obtuvo un valor RMSE de 0.0088, un valor MSE de 7.75e−05 y un valor NRMSE de 0.0129 como métricas de rendimiento para la humedad del suelo. El sistema entrenado se implementó con un controlador de hardware en la granja. Los resultados revelaron que el D2LIAU basado en DBLS-TM propuesto proporciona una programación y monitoreo de riego eficientes, ahorrando entre 21.96% y 63.05% de agua en comparación con las prácticas de riego manual, basadas en el tiempo, basadas en difusas, FFANN, t-based y DLiSA, respectivamente.

CAPÍTULO 3

3. DISEÑO E IMPLEMENTACIÓN

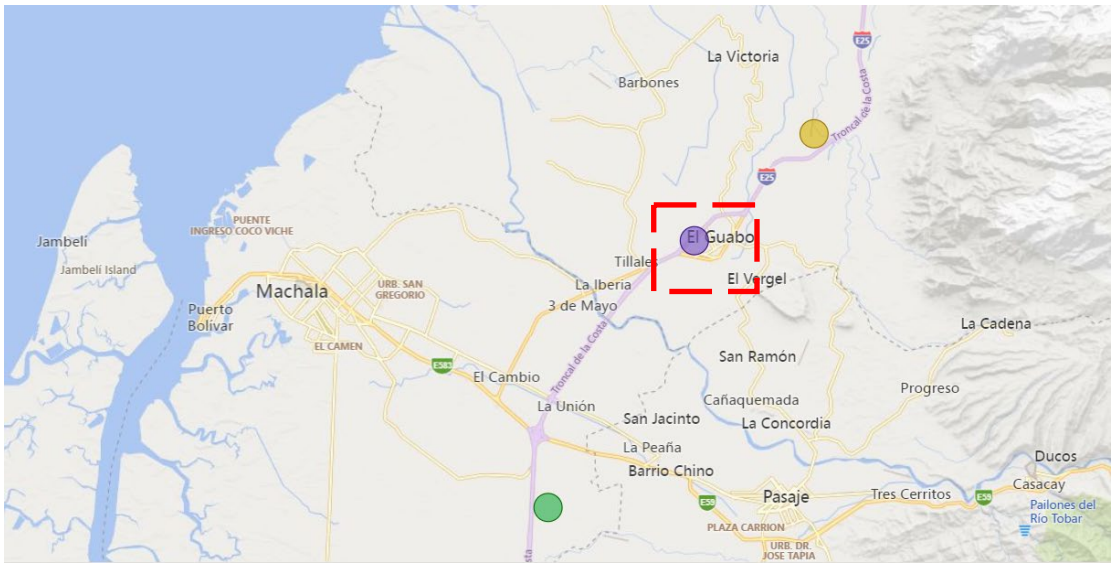
En este capítulo revisaremos la metodología desplegada para el diseño e implementación de los modelos de pronósticos. Adicional, revisaremos las estadísticas descriptivas de los indicadores climatológicos que componen la evapotranspiración.

3.1 Recolección y preprocesamiento de datos

El estudio se aplicó en la provincia de El Oro, puntualmente, en el cantón El Guabo, dado que, en este lugar es donde el Observatorio Estadístico de Banano concentra el 20% de productores que colaboran activamente con la institución. El alcance de los productores de la zona corresponde a 148 hectáreas.

El mapa muestra la georreferencia seleccionada para el presente estudio

Gráfico 1: Referencia geográfica del cantón El Guabo



Fuente: Elaboración de autores

En el marco de este estudio, se llevó a cabo una exhaustiva exploración y validación de los datos y fuentes utilizados para el análisis de la evapotranspiración (ETo) en el cantón Guabo. Los datos empleados provienen de fuentes públicas, específicamente del portal de libre acceso POWER LARC de la Administración Nacional de Aeronáutica y el Espacio (N.A.S.A). Este conjunto de datos abarca el período desde el 1 de enero de 2015 hasta el 31 de agosto de 2023, proporcionando observaciones diarias de variables climáticas clave, como precipitación, temperatura, radiación solar y humedad relativa. La selección de datos de esta

fuente se basó en su accesibilidad y calidad.

Un aspecto fundamental de la exploración de datos fue verificar la integridad y consistencia de la base de datos. Para ello, se llevó a cabo un proceso de control de calidad para identificar y abordar posibles datos faltantes o atípicos que podrían afectar el análisis.

En cuanto al procesamiento de datos, se emplearon lenguajes de programación como Python y/o RStudio, en función de las necesidades específicas de los algoritmos y la capacidad de cómputo disponible. Los datos originales se descargaron a nivel de días, y se realizó una transformación para trabajar con datos promediados, mínimos, máximos y acumulados, siguiendo las pautas sugeridas por la literatura especializada en la estimación de la evapotranspiración.

En total, se han recopilado 3165 observaciones correspondientes al Índice de Evapotranspiración del cantón El Guabo desde el 1 de enero de 2015 hasta el 31 de agosto de 2023. Para el desarrollo y validación de los modelos, la base de datos se dividirá en un conjunto de entrenamiento (80%) y un conjunto de prueba (20%), lo que se traduce en 2532 observaciones en el conjunto de entrenamiento y 633 observaciones en el conjunto de prueba. Esto con el fin de realizar una correcta ealuación de los modelos que se aplicarán.

Antes de continuar con los prototipos, modelos y módulos implementados para este estudio, resulta relevante presentar una breve descriptiva de los principales componentes de la Evapotranspiración y como se relacionan entre sí.

3.2 Exploración y validación de datos

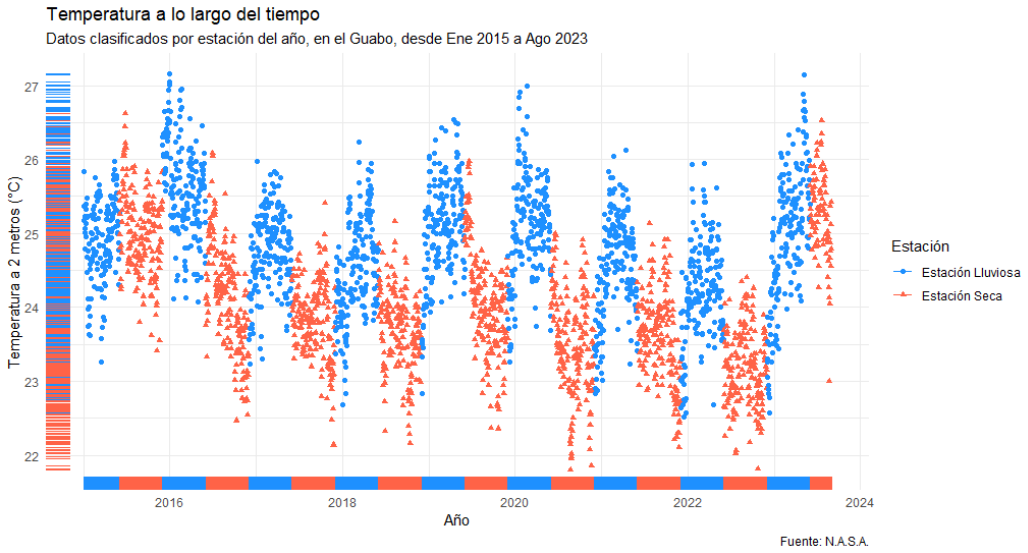
3.2.1 La Evapotranspiración y sus componentes

3.2.1.1 Temperatura a 2 metros

La temperatura a 2 metros (t2m) es una medida importante para el monitoreo de las condiciones climáticas locales, especialmente en el contexto agrícola. Esta variable proporciona información sobre la temperatura del aire cerca de la superficie, donde se desarrollan las plantas. En los gráficos, se observa la variabilidad de la temperatura a lo largo del tiempo, diferenciada por estaciones lluviosa y seca.

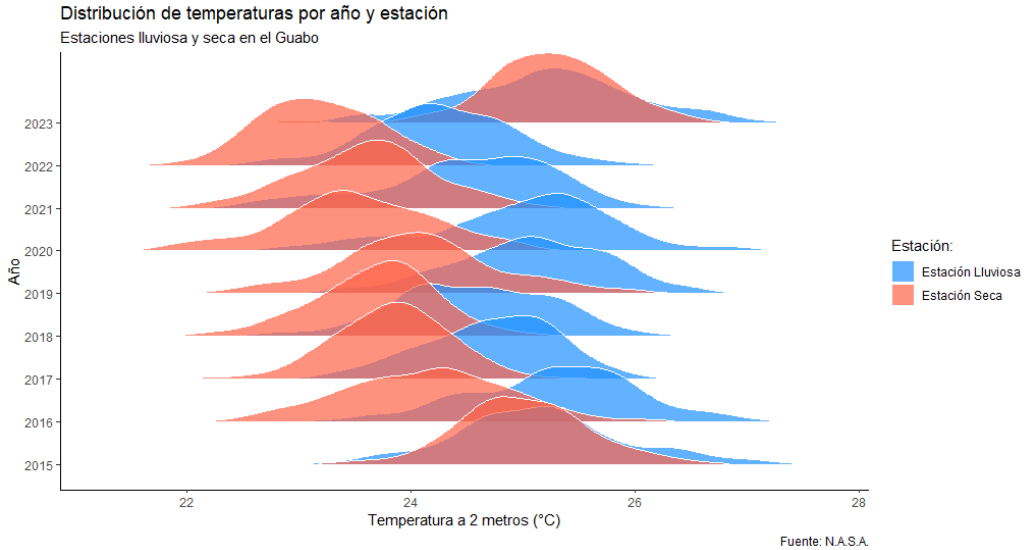
Durante la Estación Lluviosa, la temperatura tiende a ser más moderada debido a la mayor humedad y nubosidad, lo que reduce la radiación solar directa. En contraste, la Estación Seca muestra temperaturas un poco más bajas.

Gráfico 2: Temperatura a lo largo del tiempo y su distribución, en el cantón El Guabo



Nota: Estación Lluviosa comprende los meses de diciembre a mayo; mientras que, la Estación Seca corresponde a los meses de junio a noviembre.
Fuente: POWER LARC N.A.S.A
Elaboración: Autores

Gráfico 3: Distribución de la temperatura en el cantón El Guabo



Nota: Estación Lluviosa comprende los meses de diciembre a mayo; mientras que, la Estación Seca corresponde a los meses de junio a noviembre.
Fuente: POWER LARC N.A.S.A
Elaboración: Autores

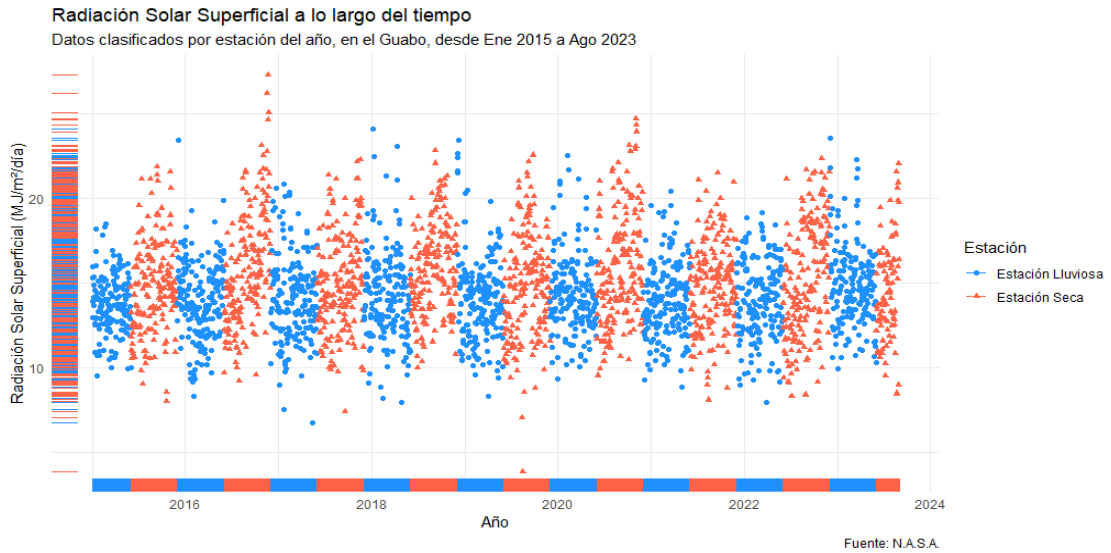
3.2.1.2 Radiación Solar Superficial

La radiación solar superficial (ALLSKY_SFC_SW_DWN) mide la cantidad de energía solar que llega a la superficie de los cultivos, importante para la fotosíntesis y, por ende, para el crecimiento de las plantas.

Se observó que, durante la Estación Seca, los niveles de radiación solar son

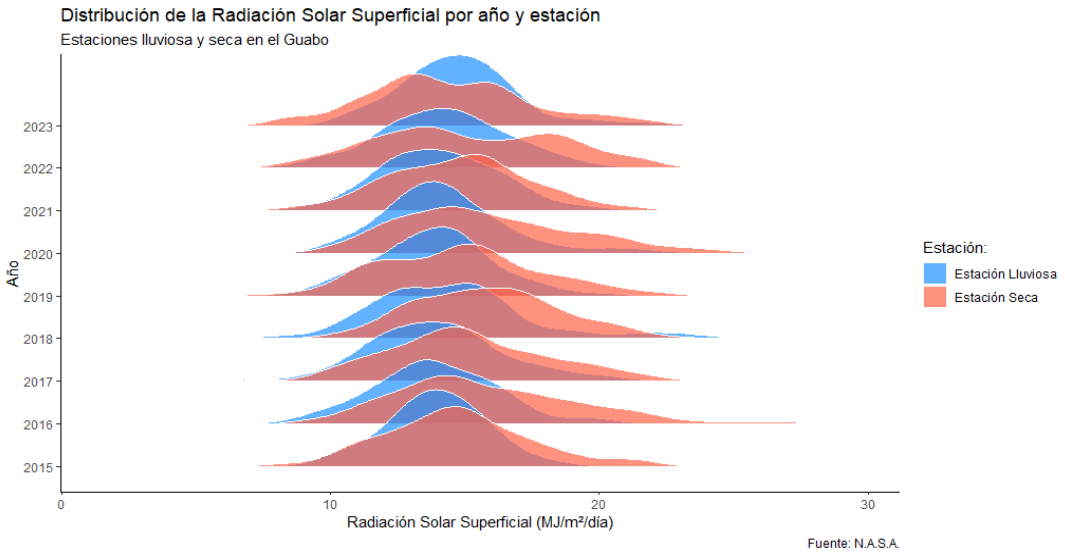
generalmente más altos debido a la menor nubosidad, lo que permite una mayor penetración de la luz solar.

Gráfico 4: Radiación Solar Superficial a lo largo del tiempo en el cantón El Guabo



Nota: Estación Lluviosa comprende los meses de diciembre a mayo; mientras que, la Estación Seca corresponde a los meses de junio a noviembre.
Fuente: POWER LARC N.A.S.A
Elaboración: Autores

Gráfico 5: Distribución de la Radiación Solar Superficial en el cantón El Guabo



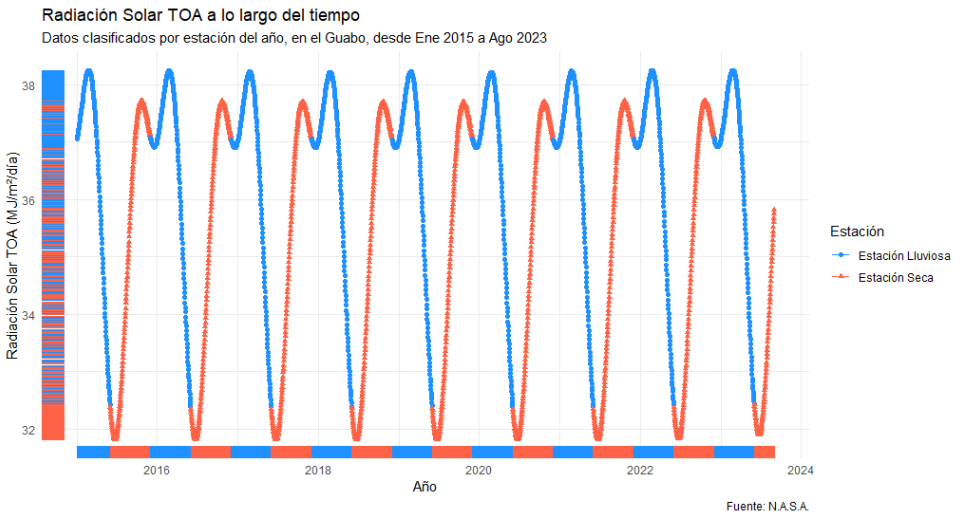
Nota: Estación Lluviosa comprende los meses de diciembre a mayo; mientras que, la Estación Seca corresponde a los meses de junio a noviembre.
Fuente: POWER LARC N.A.S.A
Elaboración: Autores

3.2.1.3 Radiación Solar en la Cima de la Atmósfera

La radiación solar en la cima de la atmósfera (TOA_SW_DWN) representa la cantidad de energía solar que llega a la atmósfera superior antes de ser absorbida o reflejada. Dicha variable sirve para entender el balance energético del planeta y los procesos climáticos. En los gráficos, se puede observar que la radiación en la

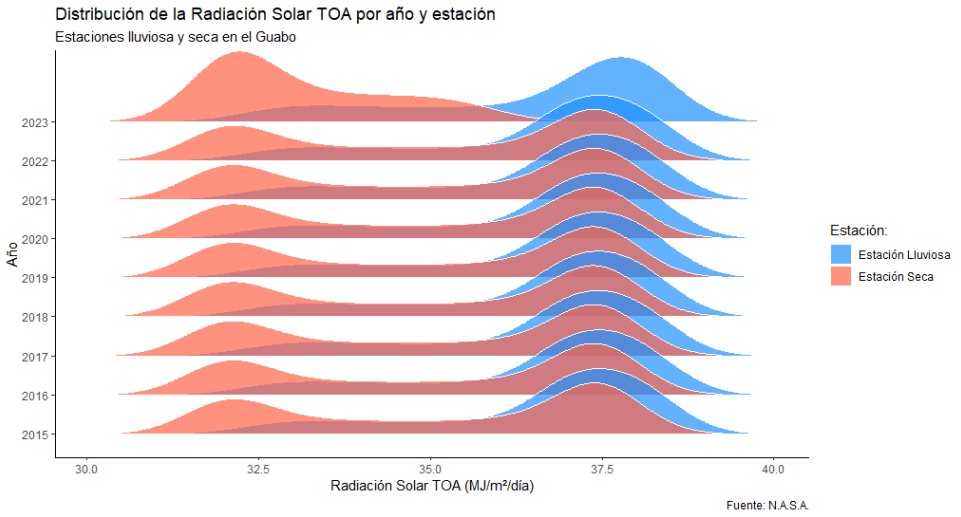
cima de la atmósfera es menos afectada por las estaciones en comparación con la radiación superficial, ya que no sufre directamente los efectos de la nubosidad y la humedad en la superficie de los cultivos. Sin embargo, las estaciones pueden influir en la cantidad de energía que finalmente llega a la superficie.

Gráfico 6: Radiación Solar en la cima de la atmósfera a lo largo del tiempo



Nota: Estación Lluviosa comprende los meses de diciembre a mayo; mientras que, la Estación Seca corresponde a los meses de junio a noviembre.
Fuente: POWER LARC N.A.S.A
Elaboración: Autores

Gráfico 7: Distribución de la Radiación Solar en la cima de la atmósfer



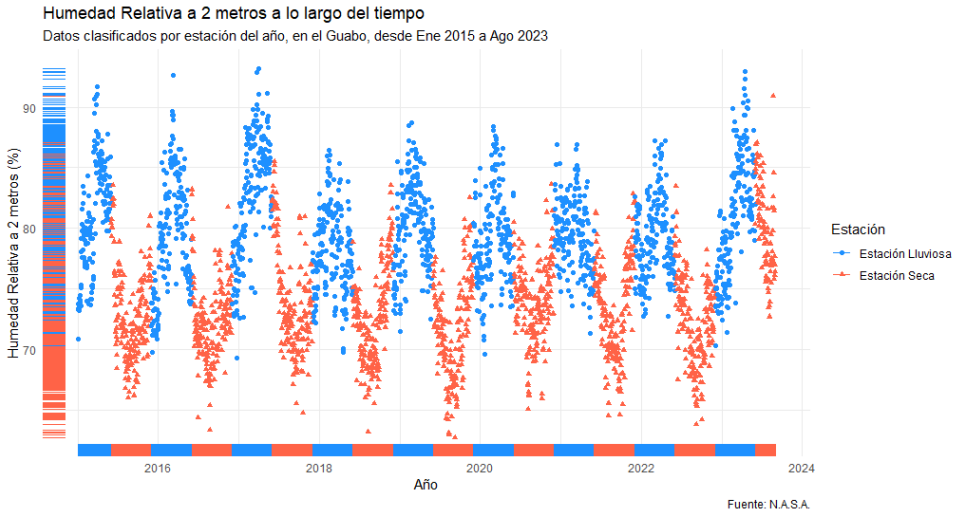
Nota: Estación Lluviosa comprende los meses de diciembre a mayo; mientras que, la Estación Seca corresponde a los meses de junio a noviembre.
Fuente: POWER LARC N.A.S.A
Elaboración: Autores

3.2.1.4 Humedad Relativa a 2 metros

La humedad relativa a 2 metros (RH2M) representa la cantidad de vapor de agua presente en el aire en comparación con la cantidad que podría contener a una temperatura específica, lo que permite monitorear el confort térmico y la salud de los cultivos.

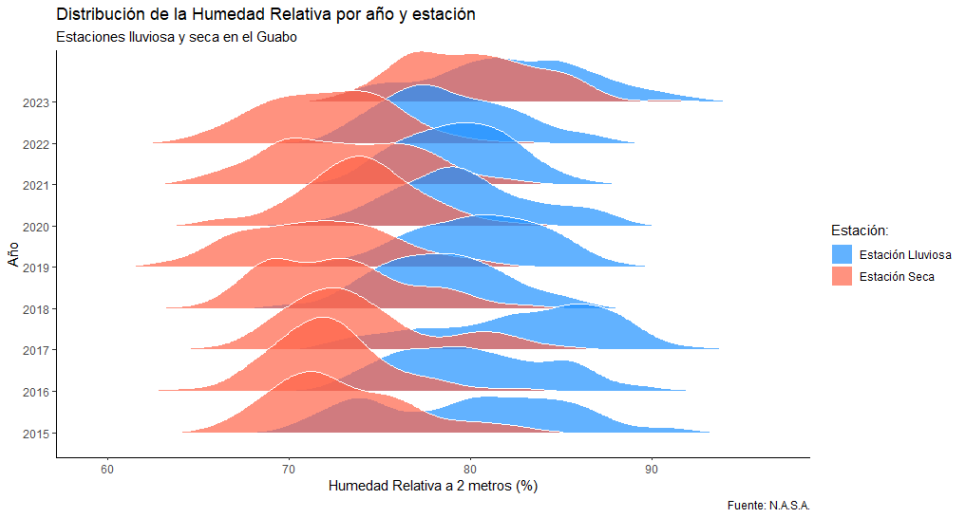
En los gráficos, se observó que la humedad relativa es generalmente más alta durante la Estación Lluviosa debido a las mayores precipitaciones y la humedad del suelo. En la Estación Seca, la humedad relativa tiende a ser más baja debido a la menor cantidad de precipitaciones y mayor evaporación.

Gráfico 8: Humedad Relativa a lo largo del tiempo en el cantón El Guabo



Nota: Estación Lluviosa comprende los meses de diciembre a mayo; mientras que, la Estación Seca corresponde a los meses de junio a noviembre.
Fuente: POWER LARC N.A.S.A
Elaboración: Autores

Gráfico 9: Distribución de la Humedad Relativa en el cantón El Guabo



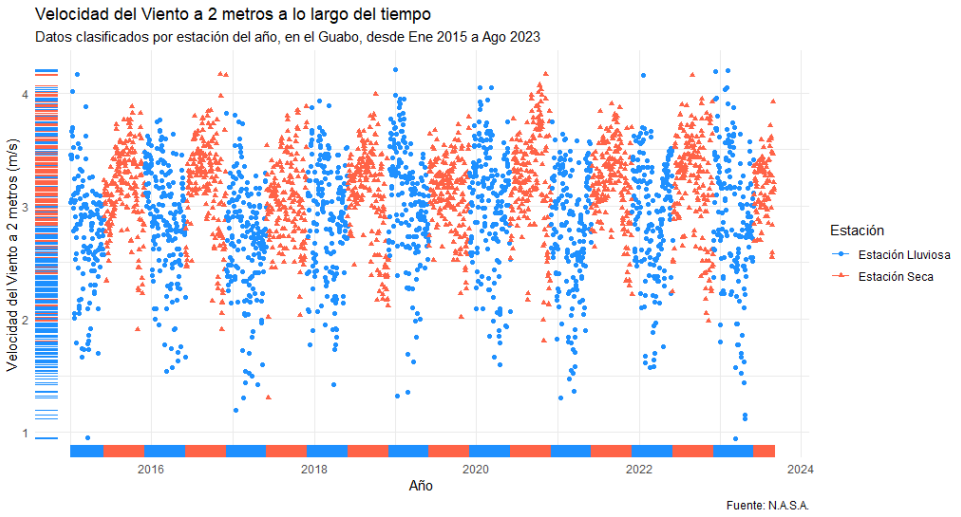
Nota: Estación Lluviosa comprende los meses de diciembre a mayo; mientras que, la Estación Seca corresponde a los meses de junio a noviembre.
Fuente: POWER LARC N.A.S.A
Elaboración: Autores

3.2.1.5 Velocidad del Viento a 2 metros

La velocidad del viento a 2 metros (WS2M) es una medida de la velocidad del aire cerca de la superficie de los cultivos. En los gráficos, se puede evidenciar que la velocidad del viento varía entre estaciones, concentrando valores generalmente más altos durante la Estación Seca debido a las condiciones más despejadas y

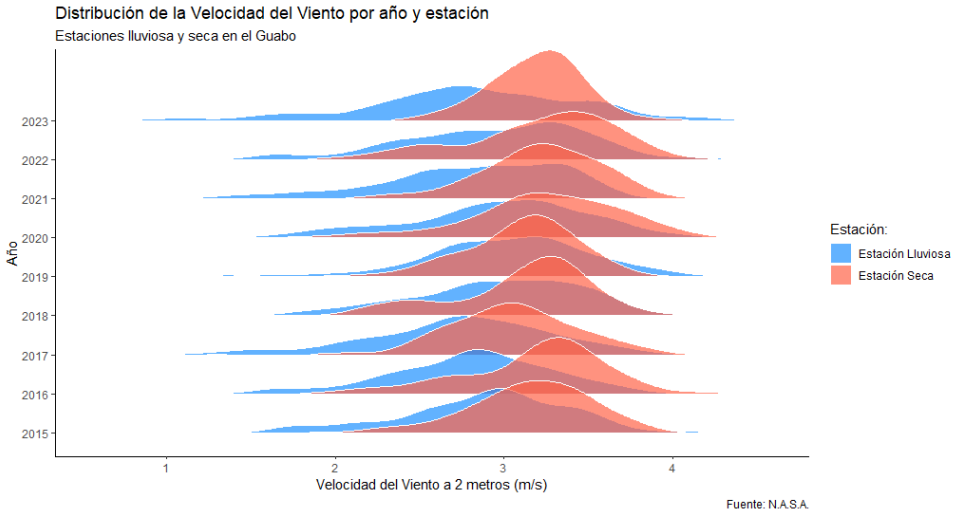
menos obstrucciones por la nubosidad y lluvia. La Estación Lluviosa muestra una mayor volatilidad en la velocidad del viento, la cual es influenciada por eventos de tormentas y lluvia.

Gráfico 10: Velocidad del viento a lo largo del tiempo en el cantón El Guabo



Nota: Estación Lluviosa comprende los meses de diciembre a mayo; mientras que, la Estación Seca corresponde a los meses de junio a noviembre.
Fuente: POWER LARC N.A.S.A
Elaboración: Autores

Gráfico 11: Distribución de la Velocidad del viento en el cantón El Guabo



Nota: Estación Lluviosa comprende los meses de diciembre a mayo; mientras que, la Estación Seca corresponde a los meses de junio a noviembre.
Fuente: POWER LARC N.A.S.A
Elaboración: Autores

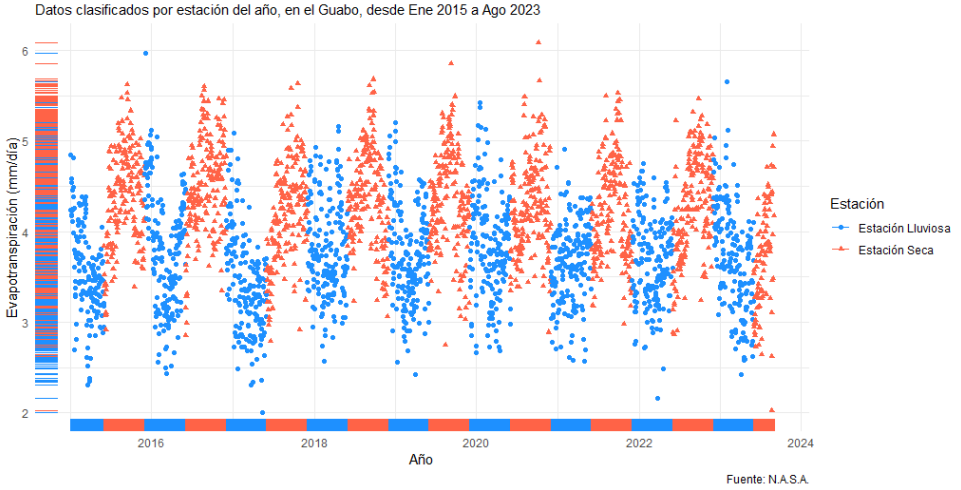
3.2.1.6 Evapotranspiración

La evapotranspiración (ETo (mm/día)) es la combinación de la evaporación del agua del suelo y la transpiración de las plantas.

En los gráficos, se observó que la evapotranspiración es mayor durante la Estación Seca debido a las temperaturas más altas y la menor humedad relativa, lo que aumenta la demanda de agua de las plantas. Conocer la variabilidad estacional de

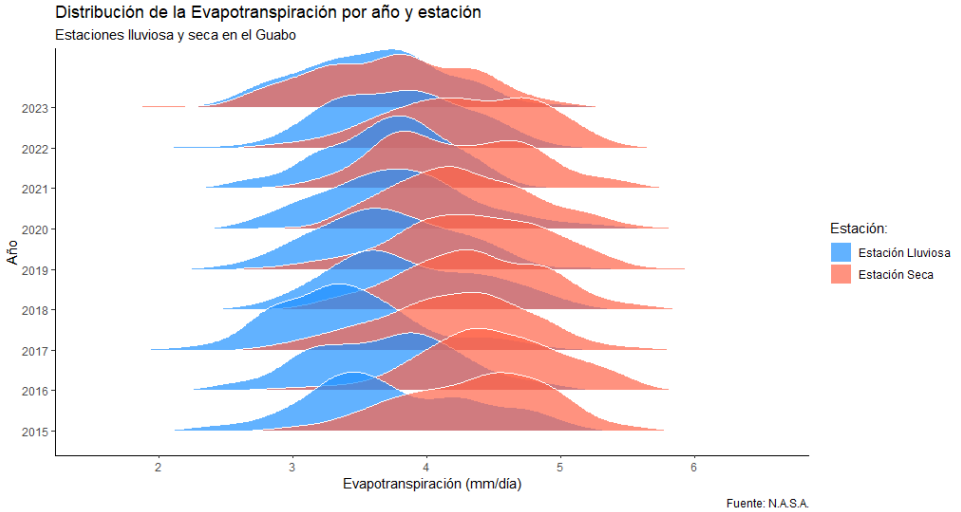
la evapotranspiración es primordial para optimizar los recursos hídricos en los cultivos y planificar adecuadamente los calendarios de riego.

Gráfico 12: Evapotranspiración a lo largo del tiempo en el cantón El Guabo



Nota: Estación Lluviosa comprende los meses de diciembre a mayo; mientras que, la Estación Seca corresponde a los meses de junio a noviembre.
Fuente: POWER LARC N.A.S.A
Elaboración: Autores

Gráfico 13: Distribución de la Evapotranspiración en el cantón El Guabo



Nota: Estación Lluviosa comprende los meses de diciembre a mayo; mientras que, la Estación Seca corresponde a los meses de junio a noviembre.
Fuente: POWER LARC N.A.S.A
Elaboración: Autores

3.2.1.7 Análisis de correlación

El análisis de correlación muestra relaciones importantes entre las variables meteorológicas y climáticas analizadas. La allsky_sfc_sw_dwn (radiación solar descendente en superficie) tiene una fuerte correlación positiva con la ETo (mm/día) (evapotranspiración diaria) de 0.70, lo que significa que a niveles más altos de radiación solar aumenta la evapotranspiración en los cultivos. Por lo que, una mayor radiación incrementaría la pérdida de agua del suelo y de las plantas, afectando las

necesidades hídricas de los cultivos.

Otra correlación fuerte y negativa es entre la rh2m (humedad relativa a 2 metros) y la ETo (mm/día) (-0.84). Esto evidencia que, a medida que aumenta la humedad relativa, la evapotranspiración disminuye considerablemente. Es decir que, una mayor humedad puede disminuir la necesidad de riego, ayudando a conservar recursos hídricos y optimizar el uso del agua en los campos.

Además, la rh2m muestra una correlación negativa moderada con la ws2m (velocidad del viento) de -0.41. Esto sugiere que velocidades de viento más altas tienden a reducir la humedad relativa, lo que puede aumentar la evapotranspiración.

Esta información resulta valiosa para los agricultores, ya que permite ajustar las prácticas de riego según las condiciones del viento, evitando el estrés hídrico en los cultivos. En conjunto, estas correlaciones destacan la interdependencia de las variables climáticas y su impacto en la agricultura, proporcionando una base para una gestión agrícola más informada y eficiente.

Gráfico 14: Correlación entre los componentes de la Evapotranspiración

Fuente: POWER LARC N.A.S.A
Elaboración: Autores

3.3 Prototipos de Algoritmos, Modelos y Módulos del Sistema

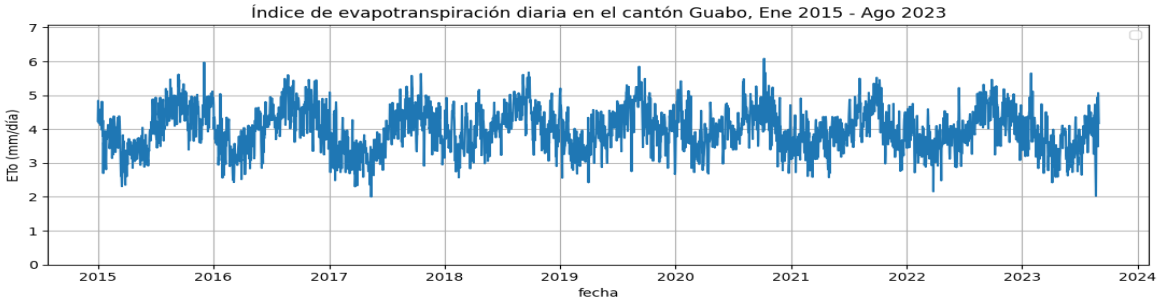
Los prototipos de algoritmos y modelos que se implementarán en este estudio serán relevantes en la estimación de la Evapotranspiración en el cantón El Guabo. Tres enfoques principales se han seleccionado para llevar a cabo el pronóstico de ETo: AUTO-ARIMA, SARIMA y la red neuronal *Long Short-Term Memory* (LSTM).

Pero, antes de explicar el detalle de los componentes utilizados en cada modelo resulta relevante dar una breve descripción de la serie temporal objeto de estudio,

para así tener una noción inicial de su comportamiento.

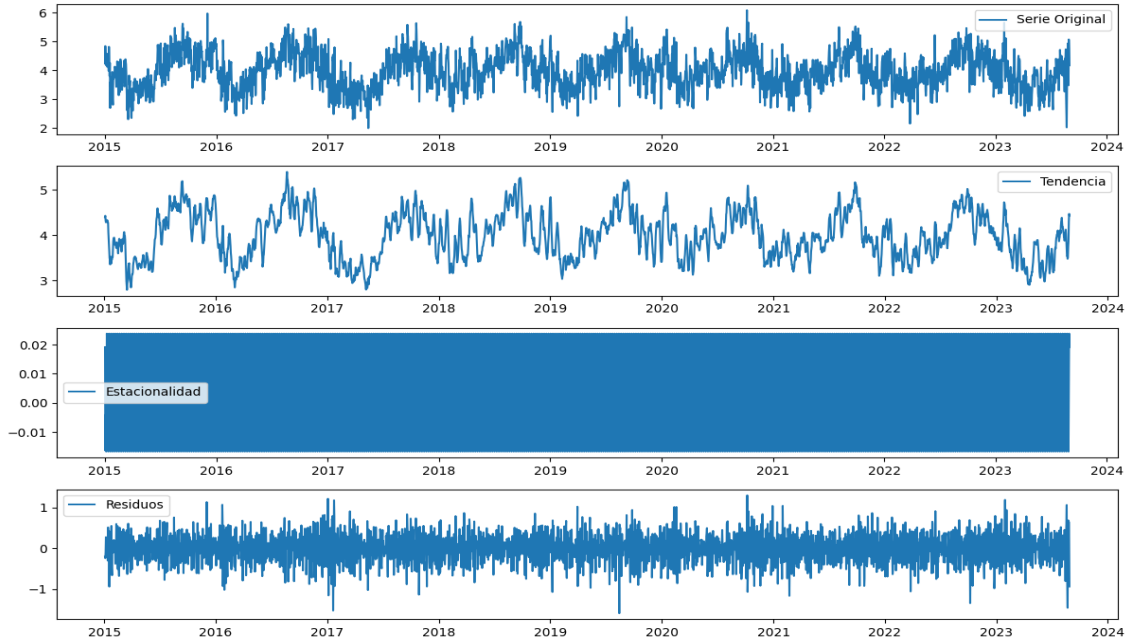
Es así como, analizando la serie de la ETo del cantón El Guabo entre el periodo de enero de 2015 a agosto de 2023, se observó que este indicador ha fluctuado durante dicho periodo en un rango de 2 a 6 mm/día. Al descomponer la serie de forma gráfica, se evidencia que no presenta tendencia y posiblemente sea estacionaria. Sin embargo, se observa estacionalidad que podría repetirse cada 15 o 30 días, junto con un ciclo anual distintivo. Este ciclo anual se manifiesta con valores más bajos a principios de cada año, seguidos de un aumento gradual hasta mediados y finalmente termina con valores bajos al finalizar el año.

Gráfico 15: Índice de evapotranspiración del cantón El Guabo



Fuente: POWER LARC N.A.S.A
Elaboración: Autores

Gráfico 16: Descomposición de la serie temporal del índice de evapotranspiración del cantón El Guabo



Fuente: POWER LARC N.A.S.A
Elaboración: Autores

3.3.1 Modelo Auto-ARIMA

El modelo AUTO-ARIMA se basa en la metodología ARIMA *(AutoRegressive*

Integrated Moving Average) y emplea un enfoque automatizado para la selección de los órdenes del modelo. Su simplicidad y capacidad de adaptación a diferentes series temporales hacen que sean un de los modelos econométricos más utilizados, aunque su precisión puede variar según la complejidad de los datos.

En este caso, los resultados del modelo reflejaron una estructura SARIMAX(2, 0, 1). Este está compuesto por un componente autorregresivo (AR) de orden 2 y un componente de media móvil (MA) de orden 1. La estimación de los coeficientes revela que el término AR(1) es positivo y altamente significativo, lo que sugiere una fuerte dependencia lineal en los datos con un rezago de 1 periodo.

El término AR(2) es negativo y significativo, indicando que hay una dependencia lineal negativa a 2 periodos atrás. El término MA(1) es negativo y también significativo, lo que sugiere la presencia de una relación lineal negativa con el término de media móvil anterior. Además, el valor del logaritmo de la verosimilitud *(Log Likelihood)* es -1364.912, y los criterios de información AIC y BIC son 2739.824 y 2769.008 respectivamente. Estos valores indican un buen ajuste del modelo a los datos, siendo menores que los criterios de modelos alternativos. Sin embargo, el valor del estadístico de Jarque-Bera es 30.85, con una probabilidad asociada de 0.00, lo que sugiere que los residuos del modelo no siguen una distribución normal. Además, la kurtosis es 3.41, indicando una mayor concentración de valores en comparación con una distribución normal. En resumen, el modelo SARIMAX es efectivo para capturar la estructura de dependencia en los datos, pero los residuos no son normalmente distribuidos, lo que podría sugerir una mayor exploración de la validez de las suposiciones subyacentes del modelo.

En relación con la prueba de autocorrelación, se observa que el valor de Ljung-Box (L1) (Q) es bajo, 0.02, con una probabilidad asociada de 0.87, lo que indica que no existe una autocorrelación significativa en los residuos en el primer rezago. Además, el estadístico de heteroscedasticidad (H) es 0.98 con una probabilidad asociada de 0.74, lo que sugiere que no hay evidencia suficiente de heteroscedasticidad en los residuos, lo que respalda la suposición de homocedasticidad del modelo. La asimetría (Skew) es -0.18, indicando una leve asimetría hacia la izquierda en la distribución de los residuos.

En conjunto, estos resultados sugieren que el modelo SARIMAX(2, 0, 1) tiene un buen ajuste en términos de autocorrelación y heteroscedasticidad, lo que respalda su validez en ese aspecto. Sin embargo, la desviación significativa de la normalidad

en los residuos, como lo indica el valor de Jarque-Bera, sugiere que se deben tener precauciones adicionales al interpretar las estimaciones y al realizar inferencias basadas en este modelo, especialmente si se asumen distribuciones normales en los análisis posteriores.

Tabla 1: Resultados del modelo Auto-ARIMA

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	2532	
Model:	SARIMAX(2, 0, 1)	**Log Likelihood**	-1364.912	
Date:	Thu, 30 May 2024	**AIC**	2739.824	
Time:	8:09:27	**BIC**	2769.008	
Sample:	0 - 2532	**HQIC**	2750.413	
Covariance Type:	opg			

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
intercept	0.0609	0.016	3.727	0.00	0.029	0.093
ar.L1	1.3879	0.029	47.572	0.00	1.331	1.445
ar.L2	-0.4031	0.027	-15.131	0.00	-0.455	-0.351
ma.L1	-0.8549	0.021	-41.107	0.00	-0.896	-0.814
sigma2	0.1719	0.004	38.515	0.00	0.163	0.181

Ljung-Box (L1) (Q):	0.02	**Jarque-Bera (JB):**	30.85
Prob(Q):	0.87	**Prob(JB):**	0
Heteroskedasticity (H):	0.98	**Skew:**	-0.18
Prob(H) (two-sided):	0.74	**Kurtosis:**	3.41

Warnings: [1] Covariance matrix calculated using the outer product of gradients (complex-step).

Elaborado por: Autores

3.3.2 Modelo SARIMA (Manual)

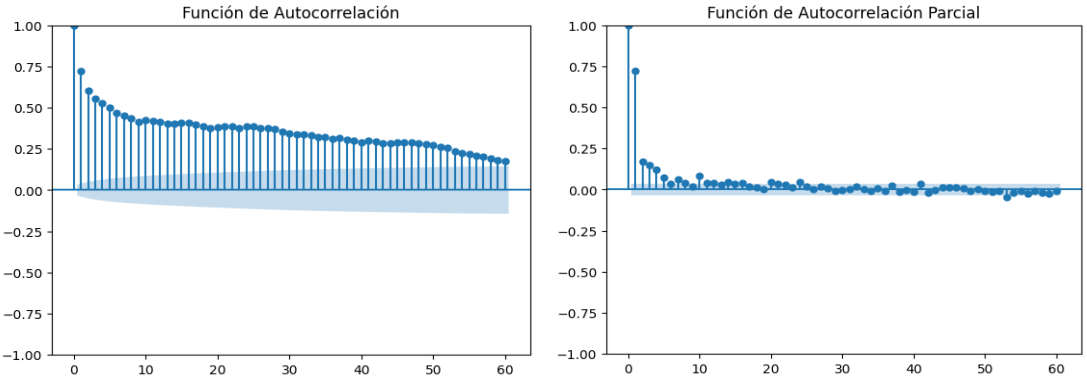
Por su parte, el modelo SARIMA incorpora una especificación manual de los órdenes de auto regresión, integración y media móvil, así como componentes estacionales. Esto permite una adaptación más precisa a las características específicas de la serie temporal, lo que puede resultar en pronósticos más robustos en comparación con el AUTO-ARIMA.

Para determinar el orden autorregresivo (p) y el orden de la media móvil (q) se analizó las gráficas de autocorrelación parcial (PACF) y función de autocorrelación (ACF) respectivamente.

En este caso, se observó que los rezagos de la ACF sobresalen hasta el rezago 60, lo que indica una correlación significativa con valores pasados en un rango amplio de períodos (los cuales están en días). Por otro lado, los rezagos de la PACF sobresalen solo hasta el rezago 6, lo que sugiere que solo los últimos seis períodos tienen una correlación directa con el valor en el período actual.

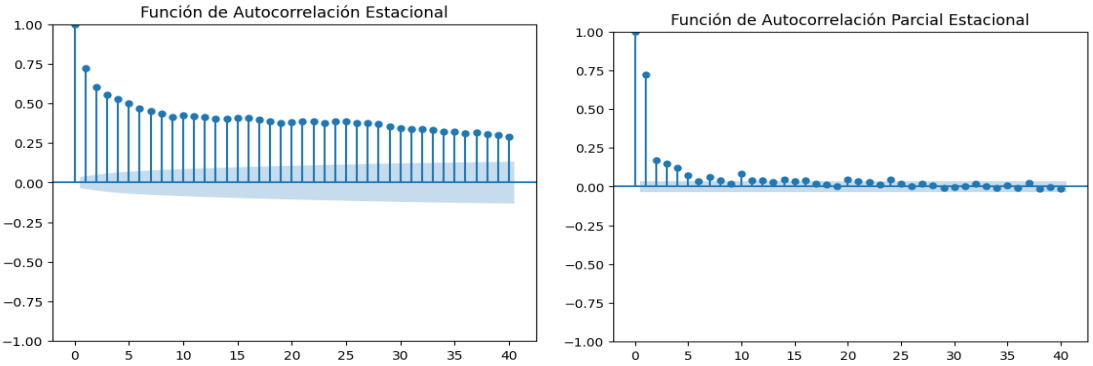
Esta discrepancia sugiere la posible presencia de un modelo ARMA (Autorregresivo de Media Móvil) para capturar la estructura de dependencia en la serie temporal de manera adecuada. Además, si se observan picos en múltiples rezagos, podría ser necesario considerar un modelo SARMA (Autorregresivo de Media Móvil Estacional) para abordar la estacionalidad en los datos.

Gráfico 17: Funciones de Autocorrelación y Autocorrelación parcial de la serie de evapotranspiración del cantón Guabo.



Fuente: POWER LARC N.A.S.A
Elaboración: Autores

Gráfico 18: Funciones de autocorrelación estacional y autocorrelación parcial estacional de la serie de evapotranspiración del cantón Guabo.



Fuente: POWER LARC N.A.S.A
Elaboración: Autores

Los resultados de la prueba de Estacionariedad ADF *(Augmented Dickey-Fuller)* son de gran importancia en el análisis de series temporales. En este caso, el estadístico ADF tiene un valor de -4.78, mientras que el valor p asociado es de 6.01e-05, que es notablemente menor que el nivel de significancia comúnmente utilizado de 0.05. Esta diferencia indica que la serie temporal es estacionaria. La estacionariedad implica que las propiedades estadísticas de la serie no varían con el tiempo. Esto simplifica el modelado y análisis, ya que muchas técnicas y suposiciones en estadísticas y econometría requieren que los datos sean estacionarios.

Durante el proceso de modelado de la serie temporal de evapotranspiración (ETo), se llevaron a cabo diversas pruebas e iteraciones en la búsqueda de un modelo SARIMA (*Seasonal Autoregressive Integrated Moving Average*) que se ajustara adecuadamente a los datos. Se exploraron varios órdenes y componentes estacionales, en un esfuerzo por capturar la complejidad de la serie. Sin embargo, a pesar de los esfuerzos y pruebas exhaustivas, los modelos previamente considerados no lograron proporcionar predicciones precisas y métricas de validación satisfactorias. Este desafío es común en el análisis de series temporales, ya que muchas series presentan patrones y comportamientos intrínsecos que pueden ser difíciles de capturar de manera efectiva.

Finalmente, se probó un modelo de SARIMAX(5, 0, 1)x(1, 1, [] 30) que parecía ofrecer un mejor ajuste a la serie de ETo. Este modelo especifica un componente autorregresivo (AR) de orden 5, un componente de media móvil (MA) de orden 1 y un componente estacional con un período de 30. Los resultados del modelo indican un valor de logaritmo de verosimilitud (*Log Likelihood*) de -1873.18 y criterios de información AIC y BIC de 3762.363 y 3808.962, respectivamente.

Al examinar los coeficientes del modelo, se observa que los términos autorregresivos (ar.L1 a ar.L5) tienen valores en el rango de 0.0878 a 0.2450, lo que indica una dependencia lineal con los valores pasados de la serie y estadísticamente significativos solo los rezagos del 3 al 5. Además, el componente estacional (ar.S.L30) tiene un valor de -0.4821, lo que sugiere una fuerte influencia estacional en el modelo, con un período de 30 días. El valor de sigma2 (0.2608) representa la varianza de los residuos del modelo.

Es importante señalar que, a diferencia de los modelos anteriores, el valor de Jarque-Bera (JB) en este modelo es 4.06, con un valor p de 0.13, lo que indica que los residuos podrían tener una distribución más cercana a la normalidad en comparación con los modelos anteriores. Además, el estadístico de Ljung-Box (L1) muestra un valor de 0.00, lo que sugiere una baja autocorrelación en los residuos en el primer rezago.

En resumen, el modelo SARIMAx(5, 0, 1)x(1, 1, [], 30) parece ser el más adecuado para capturar la dinámica de la serie de ETo, con coeficientes significativos y un mejor ajuste según los criterios de información.

Tabla 2: Resultados del modelo SARIMA-Manual

SARIMAX Results						
	Dep. Variable:	ETo (mm/día)		**No. Observations:**	2532	
	Model:	SARIMAX(5, 0, 1)x(1, 1, [], 30)		**Log Likelihood**	-1873.182	
	Date:	Sun, 22 Oct 2023		**AIC**	3762.363	
	Time:	2:12:10		**BIC**	3808.962	
	Sample:	42005		**HQIC**	3779.28	
		-2039				
	Covariance Type:	opg				
	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
ar.L1	0.15	0.31	0.48	0.63	-0.47	0.77
ar.L2	0.25	0.17	1.45	0.15	-0.09	0.58
ar.L3	0.09	0.03	3.57	0.00	0.04	0.14
ar.L4	0.10	0.03	3.05	0.00	0.04	0.17
ar.L5	0.09	0.03	2.86	0.00	0.03	0.15
ma.L1	0.38	0.32	1.22	0.22	-0.24	1.00
ar.S.L30	-0.48	0.02	-28.09	0.00	-0.52	-0.45
sigma2	0.26	0.01	35.79	0.00	0.25	0.28
Ljung-Box (L1) (Q):	0		**Jarque-Bera (JB):**	4.06		
Prob(Q):	0.99		**Prob(JB):**	0.13		
Heteroskedasticity (H):	1.02		**Skew:**	-0.09		
Prob(H) (two-sided):	0.78		**Kurtosis:**	3.07		

Warnings: [1] Covariance matrix calculated using the outer product of gradients (complex-step).

Elaborado por: Autores

3.3.3 Modelo LSTM

Dado el problema que surgía en los modelos anteriores, se ha desarrollado un último modelo de pronóstico para la serie de evapotranspiración (ETo) basado en redes neuronales LSTM (*Long Short-Term Memory*), con el fin de capturar y modelar las relaciones temporales y las dependencias complejas presentes en la serie temporal de ETo, lo que permitirá realizar pronósticos más precisos y detallados.

El proceso comienza con la carga de datos de ETo desde un archivo Excel y su división en conjuntos de entrenamiento y prueba, donde el 70% de los datos se utiliza para entrenar el modelo, el 15% para validación y el 15% restante para testeo. Es importante destacar que antes de utilizar los datos, se procede a su normalización, lo que implica escalar los valores en un rango entre 0 y 1. Esto facilita el proceso de entrenamiento y mejora la convergencia de los datos.

Para iniciar el modelo LSTM, se generan secuencias de datos a partir de los datos normalizados. Cada secuencia se compone de tres pasos de tiempo anteriores y

se utiliza para predecir el siguiente valor. Las secuencias se dividen en características (input) y etiquetas (output), y se convierten en tensores de TensorFlow para su uso en el modelo. El modelo LSTM consta de una capa con 50 unidades y una función de activación ReLU, seguida de una capa de salida densa de una unidad.

Tras la construcción del modelo, se procede a su compilación utilizando el optimizador 'adam' y la función de pérdida *'Mean Absolute Error' (MAE)*. El optimizador 'adam' es conocido por su eficiencia en el entrenamiento de redes neuronales. Luego, el modelo se entrena durante 100 épocas con un tamaño de lote igual a 1. Durante el entrenamiento, el modelo ajusta sus pesos para minimizar la función de pérdida, permitiendo que aprenda a realizar predicciones robustas (*para ver detalle del código dirigirse al anexo 1*).

Este modelo presenta una mejora significativa en la predicción de la evapotranspiración del cantón El Guabo (*ver Capítulo 4*), ya que se adapta a la complejidad de las series temporales. Sin embargo, es importante tener en cuenta que, dada la inherente variabilidad y desafíos en la predicción de series temporales, pueden ser necesarias iteraciones adicionales y ajustes para optimizar aún más el modelo y evaluar su desempeño en datos de testeo independientes.

3.4 Métricas y presentación de resultados

Para efectos de este estudio, se llevará a cabo una evaluación visual de los resultados. Los pronósticos generados por cada modelo se compararán con los valores reales de ETo en el conjunto de testeo. Esto permitirá una comprensión intuitiva de cuán bien se ajustan los modelos al comportamiento de la serie temporal y cómo se comparan entre sí. En cuanto a las métricas de rendimiento, se evaluarán métricas como el Error Cuadrático Medio de la Raíz (RMSE) y/o el Error Medio Absoluto de Escala (MASE). Estas métricas proporcionarán una medida cuantitativa de la precisión de los modelos y permitirán comparar su rendimiento.

La comunicación de los resultados se realizará de manera efectiva, destacando el modelo que mejor se ajuste al comportamiento de la serie temporal en el cantón El Guabo. Ello, respaldará las conclusiones y permitirían a los tomadores de decisiones comprender la calidad de los modelos y su priorización.

3.5 Plataformas de Visualización

Para la presentación de resultados, se utilizará la plataforma de *Power BI*, que

facilita la creación de visualizaciones interactivas y comprensibles. La elección de Power BI se basa en su capacidad para transformar datos en información accesible y relevante para una amplia audiencia, lo que es esencial en la comunicación de resultados en un entorno práctico y de toma de decisiones.

CAPÍTULO 4

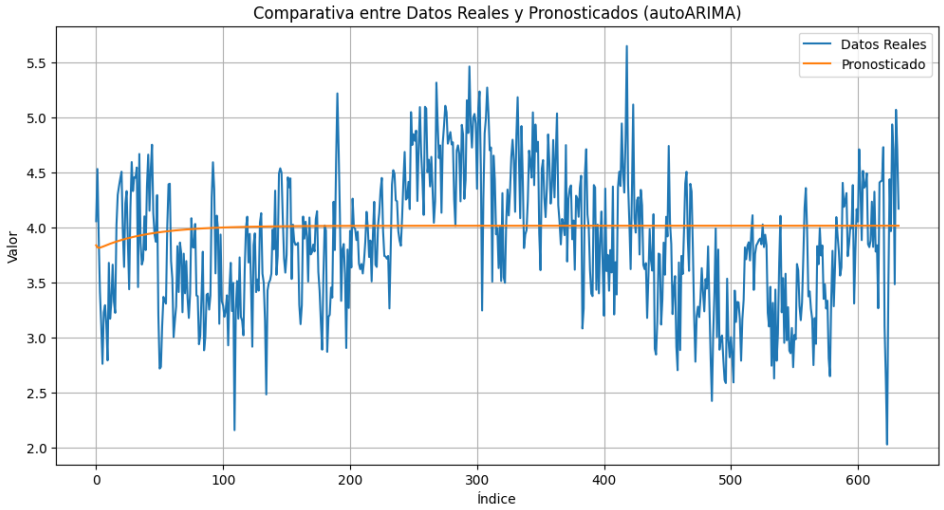
4. ANÁLISIS DE RESULTADOS

A continuación, se presentan los resultados obtenidos junto con una validación de estos resultados que cuantifica el error en comparación con diferentes metodologías y elección del mejor modelo (considerando las métricas señaladas en el apartado 3.4) para la predicción del Índice de Evapotranspiración en los cultivos del cantón el Guabo.

4.1 Rendimiento del Modelo Auto-ARIMA

Como se esperaba, los resultados del modelo utilizando el data set de testeo reflejan pronósticos que no capturan el comportamiento de la serie. Los valores proyectados muestran una tendencia prácticamente lineal y no logran reflejar las fluctuaciones y patrones inherentes a la serie. Esta discrepancia entre los pronósticos del modelo y la realidad de la serie subraya la necesidad de una revisión y ajuste del modelo para mejorar su capacidad predictiva y garantizar que refleje con precisión el comportamiento observado en la serie de tiempo.

Gráfico 19: Comparación de valores pronosticados vs valores reales del modelo Auto-ARIMA, con el dataset de testeo.



Fuente: POWER LARC N.A.S.A
Elaboración: Autores

Dado los resultados obtenidos en el modelo automático SARIMAX(2, 0, 1) y la desviación significativa de la normalidad en los residuos según el valor del estadístico de Jarque-Bera, se ha tomado la decisión de llevar a cabo un análisis más exhaustivo. Este análisis se centrará en una evaluación detallada de los componentes de orden q, I y p, además de considerar otros factores que pueden

ejercer influencia en el comportamiento de la serie.

El fin de esto es encontrar un modelo mejor ajustado a los datos, que ofrezca una representación más precisa de la estructura subyacente en la serie de tiempo. La desviación de la normalidad en los residuos, como lo indica el valor de Jarque-Bera, sugiere que podrían existir aspectos no capturados por el modelo actual que requieren una exploración más profunda. Este enfoque permitirá una comprensión más sólida de la serie y, en última instancia, mejorar la calidad de las inferencias y predicciones realizadas en este contexto.

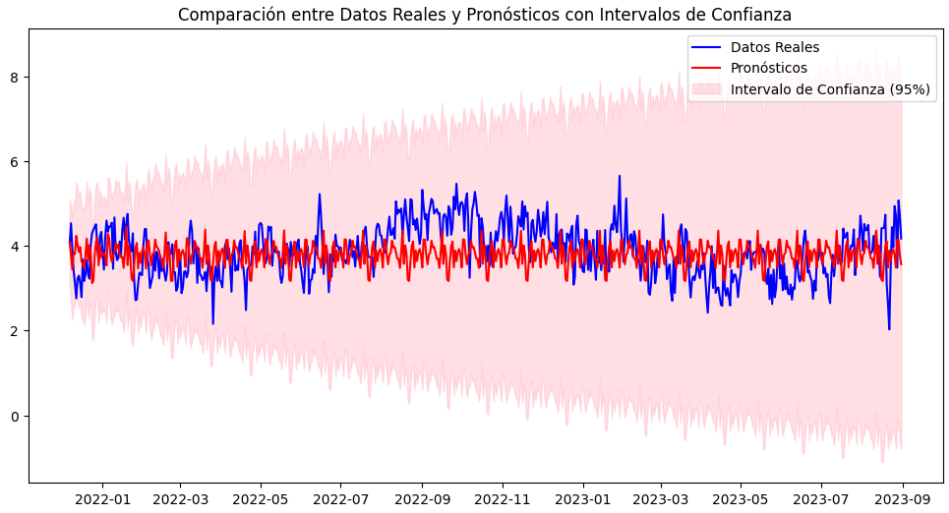
4.2 Rendimiento del Modelo SARIMA (Manual)

El modelo SARIMAX(5, 0, 1)x(1, 1, [], 30) que se obtuvo presenta un avance significativo en la capacidad de pronosticar la serie temporal de evapotranspiración (ETo). Al evaluar su desempeño con respecto a los datos de testeo, se observó una mejora notable en los valores pronosticados en comparación con el modelo previamente analizado.

Los pronósticos se alinean mejor con los valores observados, lo que es fundamental para cualquier aplicación que requiera predicciones precisas de la evapotranspiración, como la gestión de recursos hídricos, la agricultura y la climatología.

Sin embargo, a pesar de las mejoras evidentes, este modelo presenta un desafío que se relaciona con el crecimiento del intervalo de confianza a medida que se avanza en el horizonte de pronóstico. Esto significa que, a medida que nos alejamos en el tiempo desde el último punto de datos observados, la incertidumbre en las predicciones aumenta gradualmente. Este fenómeno es común en pronósticos de series temporales y se debe en parte a la influencia de los términos estacionales. A medida que avanzamos en el tiempo, la incertidumbre sobre cómo evolucionarán los patrones estacionales futuros se vuelve más relevante.

Gráfico 20: Comparación entre valores pronosticados y valores reales del modelo SARIMA manual



Fuente: POWER LARC N.A.S.A
Elaboración: Autores

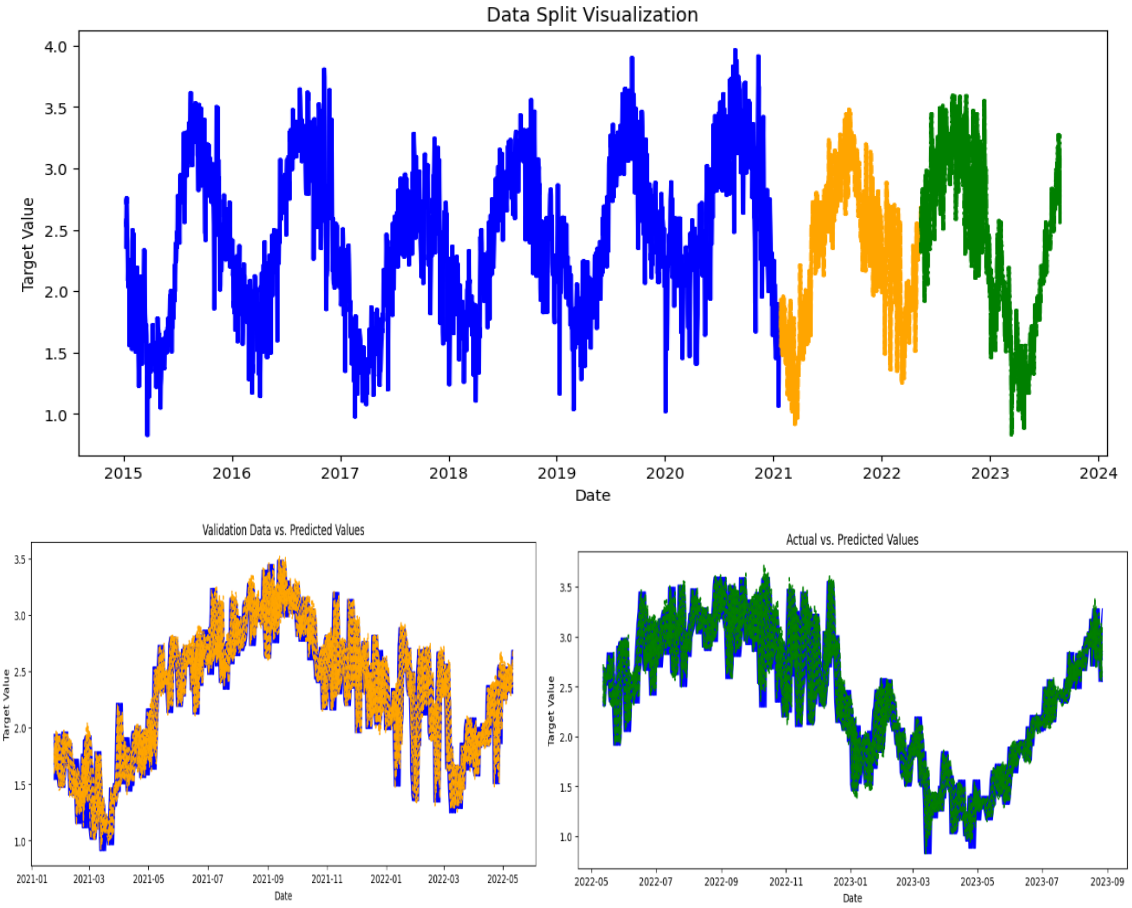
Para abordar esta limitación y mejorar aún más la precisión de las predicciones, se requeriría una iteración continua en la búsqueda de modelos más sofisticados o una refinación de los parámetros. Esto podría implicar la consideración de modelos más avanzados, como los modelos de redes neuronales recurrentes (RNN) o las redes de memoria de corto-largo plazo (LSTM), o bien la inclusión de variables exógenas adicionales que puedan influir en la evapotranspiración.

Además, es importante evaluar y ajustar los parámetros estacionales y la longitud de los intervalos de confianza en función de las necesidades específicas de la aplicación. Si bien el modelo actual representa un progreso, siempre existe margen para la mejora continua en la búsqueda de la precisión y la confiabilidad en las predicciones de series temporales.

4.3 Rendimiento del Modelo LSTM

La aplicación del modelo de red LSTM a la serie temporal de evapotranspiración (ETo) del cantón el Guabo ha resultado en una notable mejora en la calidad de los valores pronosticados. Esta mejora se refleja de manera gráfica al observar cómo las predicciones se alinean de manera más precisa con los valores reales de ETo en el conjunto de testeo. Se observa una convergencia más ajustada entre las líneas de pronóstico y las observaciones reales, lo que indica que el modelo LSTM ha logrado capturar con mayor robustez la complejidad de la serie temporal.

Gráfico 21: Comparación entre valores pronosticados y valores reales del modelo LSTM



Nota: Los datos en color azul se refiere a los datos de entrenamiento, en amarillo a los de validación y en verde a los de testeo.
Fuente: POWER LARC N.A.S.A
Elaboración: Autores

Pese a ello, es fundamental tener en cuenta que, a medida que se extiende el horizonte de pronóstico en el tiempo, la incertidumbre aumenta gradualmente, como es típico en las predicciones de series temporales. Por lo tanto, aunque se ha logrado una gran mejora en la precisión de los pronósticos, aún es necesario considerar la incertidumbre inherente al proyectar a largo plazo.

4.4 Comparación y elección del mejor modelo

La comparación de los tres modelos implementados, el AutoARIMA, el SARIMAX manual y el modelo LSTM, en el contexto del pronóstico de la evapotranspiración (ETo) arroja resultados significativos y revela ventajas y desventajas distintivas para cada uno.

Tabla 3: Rendimiento de los modelos aplicados

Modelo	Métricas de rendimiento	
	RMSE	MASE
Auto-Arima	0.68	1.45
SARIMAx - Manual	0.62	1.40
LSTM	0.11	0.35

El modelo AutoARIMA, que se basa en la selección automática de los órdenes del modelo ARIMA, presenta la ventaja de su simplicidad y automatización en la elección de los parámetros del modelo. Este enfoque es adecuado para aquellos casos en los que se necesita un modelo rápido y básico. Sin embargo, en el caso de la serie de ETo, el modelo AutoARIMA puede no ser lo suficientemente sofisticado para capturar la complejidad de las dependencias temporales presentes en los datos. Esto se reflejó en un valor de RMSE y MASE más elevados (0.68 y 1.45, respectivamente), lo que indica una precisión relativamente baja en las predicciones.

Por otro lado, el modelo SARIMAX manual, que involucra un enfoque más detallado y específico para modelar la serie temporal, logró un mejor ajuste. Se ajustaron manualmente los órdenes de auto regresión, integración y media móvil, así como los componentes estacionales. Esto permitió capturar con mayor precisión las estructuras de dependencia en los datos.

El modelo LSTM, basado en redes neuronales, demostró un rendimiento destacable en la predicción de la serie temporal de ETo. Sus ventajas radican en su capacidad para modelar dependencias temporales complejas y no lineales. Esto se tradujo en un valor de RMSE sustancialmente más bajo (0.45) y un MASE de aproximadamente 0.0, lo que indica una predicción muy precisa. Además, se observó un MAPE del 9.797%, lo que refleja una buena capacidad del modelo para prever valores cercanos a los reales. Además, la construcción y entrenamiento de modelos LSTM requiere más tiempo y recursos en comparación con los enfoques ARIMA.

En resumen, el modelo AutoARIMA es adecuado para aplicaciones simples y rápidas, pero puede carecer de precisión en series temporales complejas como la ETo. El SARIMAX manual ofrece una mejora en la predicción al ajustar manualmente los parámetros del modelo. El modelo LSTM destaca por su capacidad para manejar dependencias temporales complejas y proporciona predicciones más precisas.

En el caso del presente trabajo de investigación, se elige el modelo LSTM como el ideal para realizar el pronóstico de la evapotranspiración de los cultivos del cantón El Guabo.

4.5 Estimación de la demanda hídrica

Para estimar la cantidad de agua requerida, en función del ETo calculado, es menester aclarar dos aspectos importantes.

i) Requerimiento de agua:

La ETo en mm/día se refiere a la cantidad de agua que se evapora y transpira por día por metro cuadrado, por lo tanto, esto no significa que toda el agua necesaria para cubrir determinada cantidad de hectáreas deba ser considerada en este contexto. La ETo es una tasa local que aplica a cualquier superficie, independientemente de su tamaño.

Para calcular la cantidad de agua necesaria para una hectárea (Ha) con una ETo de mm/día promedio, haremos el siguiente ejercicio práctico, con una ETo promedio de 2 mm/día:

Si:

$$1\,Hectárea = 10.000\,m^2$$

y

$$2\frac{mm}{día} = 2litros/m^2/día$$

Entonces,

$$\frac{2litros}{\frac{m^2}{día}} \times 10.000\,m^2 = 20.000\frac{litros}{día}Ha$$

Además, si:

$$1m^3 = 1.000\,litros$$

Tenemos que:

$$20.000\,litros = 20\,m^3$$

Por lo tanto, para una hectárea de cultivo con una ETo de 2 mm/día, se necesitarían 20 m³ de agua por día.

ii) Proyección sobre la extensión de área de la plantación

Dada la cobertura espacial, cada punto de datos de MERRA-2 representa un área de aproximadamente 308,000 hectáreas, pero esto es simplemente la resolución del dato, no la cantidad de agua requerida para esa área.

El ETo es un valor local aplicado a cada metro cuadrado en el área representada por el punto de datos. Así que, si se quiere estimar la necesidad total de agua para una determinada cantidad de hectáreas, seguiremos los siguientes pasos:

- Cantidad total de agua para 10 hectáreas:

$$10\ hectáreas\ *\ 20\ m^3/ha/día\ = 200\ m^3/día$$

Este cálculo se basa en la suposición de que toda el área necesita el mismo riego según la ETo promedio.

- Demanda hídrica en cultivos de banano.

De acuerdo con las estimaciones realizadas por la FAO (2006) en su estudio "Evapotranspiración del cultivo: Guías para la determinación de los requerimientos de agua de los cultivos", el Banano tiene un factor de ajuste adimensional de demanda de agua (K_c) de 1.10 el nivel de ETo en zonas húmedas tropicales.

Por lo tanto, el nivel de Evapotranspiración total del cultivo ETc, se obtiene:

$$ET_c = ET_o \times K_c$$

$$ET_c = 200\frac{m^3}{día} \times 1.10$$

$$ET_c = 220\frac{m^3}{día}$$

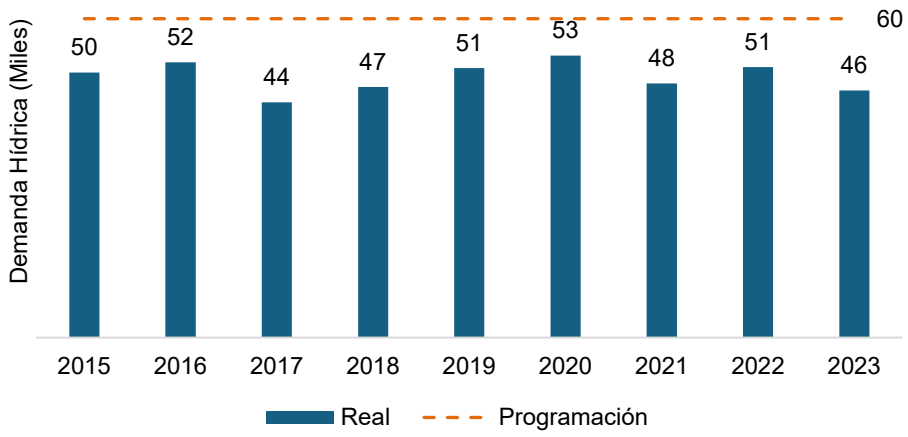
Una vez realizado este ajuste, el ETc es la estimación final de la demanda hídrica que se requiere en el cultivo de banano.

4.6 Aplicación de negocios

En el sector de producción de banano existe un valor estándar de riego de 4 mm por día, este valor, es estándar, es decir, es un criterio que no se cuestiona y se aplica en la programación de riego de las fincas (Caicedo-Caposano, Balmaseda-Espinosa, & Proaño-Saraguro, 2015). Si bien existen esfuerzos por inteligenciar la distribución del riego de acuerdo con las horas luz del día, la cantidad de agua que se proporciona al cultivo es vital, pues mucha o poca agua puede generar estrés

hídrico. En el caso de tener deficiencia hídrica, la planta no se desarrollará adecuadamente, afectando la calidad final del banano producido; mientras que, un sobre riego aumenta el riesgo de dañar las raíces, eliminando la vida útil de la planta.

Gráfico 19: Demanda programada versus demanda real.



Fuente: POWER LARC N.A.S.A
Elaboración: Autores

La administración del agua de riego es crucial particularmente en la época seca (julio a noviembre), dado que los niveles de la ETo son más altos en comparación a la época lluviosa. Siguiendo el criterio estándar, en una finca de 10 hectáreas durante el periodo seco se riegan 60 mil m^3 de agua, sin embargo, cuando se hace la estimación con el ETo real, se aprecia que esta programación se encuentra 6 mil a 15 mil m^3 por encima de la cantidad de agua que el cultivo realmente necesitó en dichos periodos, diferencia que impacta tanto a la economía del agricultor, así como a la naturaleza al emplear ineficientemente un recurso valioso.

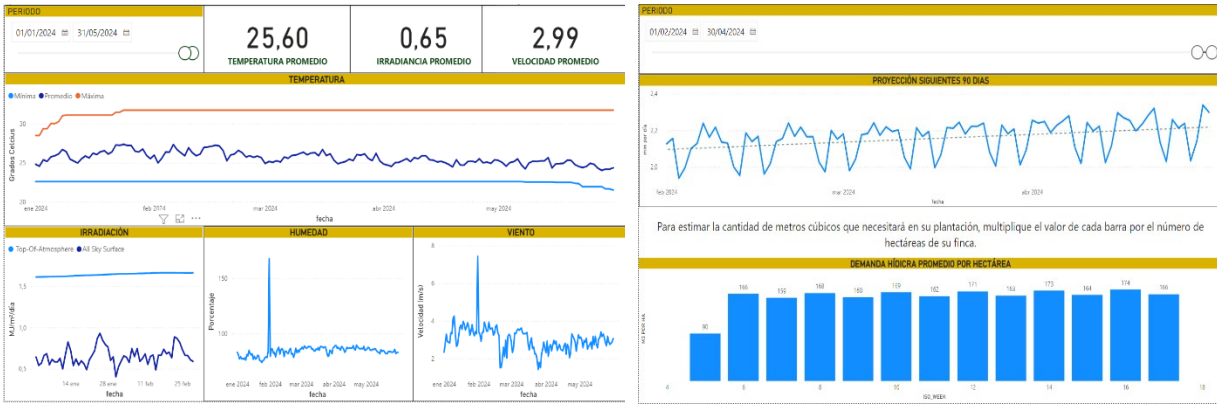
4.7 Presentación de resultados en Power BI

El propósito principal del presente trabajo de investigación es aportar a los pequeños productores de banano, quienes dada su limitación económica ven complicado implementar agricultura de precisión dentro de sus cultivos.

Por ello, bajo el marco del manejo del agua, se ha desarrollado un *dashboard* con libre acceso para que los productores tengan acceso a las estadísticas básicas de los indicadores relacionados a la ETo, como temperatura, radiación solar, entre otros. Así como a la proyección de la demanda hídrica en cultivos de 10 hectáreas, dado que, se considera como pequeño productor hasta dicho nivel de hectareaje. A continuación, se adjuntan las capturas de pantalla de las dos vistas del

dashboard, así como el enlace de acceso (https://shorturl.at/WVawP).

Gráfico 20: PÁGINAS DEL DASHBOARD DESARROLLADO



Datos: POWER LARC N.A.S.A
Elaboración: Autores

CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

El modelo LSTM basado en redes neuronales recurrentes ha sobresalido al capturar de manera efectiva las complejas dependencias temporales presentes en la serie de la ETo del cantón El Guabo, en comparación a los otros modelos analizados. Esto se refleja en un valor de RMSE sustancialmente más bajo y un MASE de 0.0, indicando la capacidad del modelo para realizar pronósticos muy precisos.

El modelo SARIMAX manual también ha demostrado ser una opción viable para mejorar la precisión en la predicción de la ETo. A través de la selección manual de órdenes y componentes estacionales, se logró una mejora en el rendimiento en comparación con el modelo AutoARIMA. Esto resalta la importancia de una especificación más detallada en la modelización de series temporales en los que se requiera una comprensión más profunda de la interacción de sus componentes temporales.

A pesar de sus ventajas, el modelo LSTM podría presentar limitaciones debido a la creciente incertidumbre en las predicciones a largo plazo. Además, requiere más recursos computacionales y tiempo para su construcción y entrenamiento en comparación con los modelos ARIMA. La elección del modelo dependerá de la aplicación específica y de los recursos disponibles.

En general, es fundamental reconocer que, independientemente del modelo seleccionado, el análisis y pronóstico de series temporales es un campo complejo que requiere una comprensión profunda de las características de los datos y la iteración constante en la búsqueda de modelos más precisos y adecuados. Además, es esencial evaluar el rendimiento en datos de prueba independientes para garantizar la generalización de los resultados.

5.2 RECOMENDACIONES

En aplicaciones críticas como la gestión de recursos hídricos y la agricultura, se recomienda considerar el uso del modelo LSTM, especialmente cuando se buscan pronósticos de alta precisión. Sin embargo, se debe tener en cuenta la

incertidumbre en las proyecciones a largo plazo y asignar recursos adecuados para el entrenamiento y evaluación de este modelo.

Para casos en los que se requiera una aproximación más sencilla y rápida, el modelo SARIMAX con selección manual de órdenes puede proporcionar mejoras significativas en comparación con los modelos automáticos. Es importante explorar diferentes combinaciones de órdenes y componentes estacionales para obtener el mejor ajuste posible.

El proceso de pronóstico de series temporales es dinámico y requiere una constante retroalimentación y ajuste. Se recomienda la implementación de sistemas de monitoreo y actualización periódica de modelos a medida que se disponga de nuevos datos, lo que contribuirá a mantener la precisión de las predicciones a lo largo del tiempo.

REFERENCIAS BIBLIOGRÁFICAS

Alhamzawi, R., Yu, K., & Benoit, D. (2012). Bayesian adaptive lasso quantile regression. . *Statistical Modelling, 12*(3), 279-297. doi:10.1177/1471082x1101200304

Bhimanpallewar, R., & Rao, M. (2021). Precision in Agriculture Decision Making Based on Machine Learning. *Technology in Agriculture*, 437. doi:10.5772/intechopen.98787

Caicedo-Caposano, O., Balmaseda-Espinosa, C., & Proaño-Saraguro, J. (2015). Programación del riego del banano (Musa paradisiaca) en finca San José 2, Los Ríos, Ecuador. *Revista Ciencias Técnicas Agropecuarias*, 18-22.

Contreras, J., Espínola, R., Nogales, F., & Conejo, A. (2003). Arima models to predict next-day electricity prices. *Ieee Transactions on Power Systems, 18*(3), 1014-1020. doi:10.1109/tpwrs.2002.804943

Demestichas, K., Peppes, N., & Alexakis, T. (2020). Survey on security threats in agricultural iot and smart farming. *Sensors, 20*(22), 6458. doi:10.3390/s20226458

Duerfeldt, K. (2011). Use of geographic information systems at Gifft Hill School, St. John, US Virgin Islands. *Iowa State University.* doi:10.31274/etd-180810-1619

FAO. (1990). *Evapotranspiración del cultivo: Guía para la determinación de los requerimientos de agua de los cultivos.* Obtenido de https://www.fao.org/3/x0490s/x0490s.pdf

FAO. (2006). *Evapotranspiración del cultivo: Guías para la determinación de los requerimientos de agua de los cultivos.* Obtenido de Chapter 6 - ETc - Single crop coefficient (Kc): https://www.fao.org/4/X0490E/x0490e0b.htm

Fischer, T., & Krauß, C. (2018). Deep learning with long short-term memory networks for financial market predictions. . *European Journal of Operational Research, 270*(2), 654-669. . doi:10.1016/j.ejor.2017.11.054

Hill, T., O'Connor, M., & Remus, W. (1996). Neural network models for time series forecasts. . *Management Science, 42*(7), 1082-1092. doi:10.1287/mnsc.42.7.1082

Jenitha, R., & Rajesh, K. (2024). Intelligent irrigation scheduling scheme based on deep bi-directional LSTM technique. . *International Journal of Environmental Science and Technology, 21*(2), 1905-1922.

Jentzen, A., & Riekert, A. (2022). A proof of convergence for stochastic gradient descent in the training of artificial neural networks with relu activation for constant target

functions. *Zeitschrift Für Angewandte Mathematik Und Physik, 72*(5). doi:https://doi.org/10.1007/s00033-022-01716-w

Jin, J., Shuyan, Y., & Yin, H. (2022). Impact of land use/land cover types on surface humidity in northern china in the early 21st century. . *Journal of Arid Land, 14*(7), 705-718. . doi:10.1007/s40333-022-0055-3

Kam, H., Sung, J., & Park, R. (2010). Prediction of daily patient numbers for a regional emergency medical center using time series analysis. . *Healthcare Informatics Research, 16*(3), 158. doi:10.4258/hir.2010.16.3.158

Krupitzer, C., & Stein, A. (2021). Food informatics—Review of the current state-of-the-art, revised definition, and classification into the research landscape. *Foods, 10*(11), 2889. doi:10.3390/foods10112889

Lawler, T. (2016). Spatially consistent corn yield variability on the Loess Hills of Northwest Iowa: a critical look. doi:10.31274/etd-180810-4597

Liu, J., & Li, C. (2017). The short-term power load forecasting based on sperm whale algorithm and wavelet least square support vector machine with dwt-ir for feature selection. *Sustainability, 9*(7), 1188. doi:10.3390/su9071188

Manrique, J., Campos, J., Paiva, E., & Ipanaque, W. (2021). Thrips incidence prediction in organic banana crop with machine learning. . *Heliyon. , 7*(12). doi:10.1016/j.heliyon.2021.e08575

Mao, Y., Xu, L., Xue, T., Liang, J., Lin, W., Wen, J., & Chen, G. (2021). Novel nomogram for predicting the 3-year incidence risk of osteoporosis in a chinese male population. . *Endocrine Connections, 10*(9), 1111-1124. doi:10.1530/ec-21-0330

McBratney, A., Whelan, B., Ancev, T., & Bouma, J. (2005). Future directions of precision agriculture. *Precision agriculture, 6*(1), 7-23. doi:10.1007/s11119-005-0681-8

Mekruksavanich, S., Jitpattanakul, A., Youplao, P., & Yupapin, P. (2020). Enhanced hand-oriented activity recognition based on smartwatch sensor data using lstms. *Symmetry, 12*(9), 1570. doi:10.3390/sym12091570

Miao, T., Wang, P., & Khan, J. (2016). Drought forecasting with vegetation temperature condition index using arima models in the guanzhong plain. . *Remote Sensing, 8*(9), 690. doi:10.3390/rs8090690

Monteleone, S., Moraes, E., Faria, B., Aquino, P., Maia, R., Neto, A., & Toscano, A. (2020). Exploring the adoption of precision agriculture for irrigation in the context of

agriculture 4.0: the key role of internet of things. *Sensors, 20*(24), 7091. doi:10.3390/s20247091

Najamuddin, M., & Fatima, S. (2022). Hybrid brnn-arima model for financial time series forecasting. . *Sukkur Iba Journal of Computing and Mathematical Sciences, 6*(1), 62-71. doi:10.30537/sjcms.v6i1.1027

NASA. (s.f.). *Moderate Resolution Imaging Spectroradiometer*. Obtenido de https://modis.gsfc.nasa.gov/about/

Neupane, J., & Guo, W. (2019). Agronomic basis and strategies for precision water management: a review. *Agronomy, 9*(2), 87. doi:10.3390/agronomy9020087

Preethi, C., Nc, B., & Ck, Y. (2021). An comprehensive survey on applications of precision agriculture in the context of weed classification, leave disease detection, yield prediction and uav image analysis. *Computing Technologies and Application, 40*, 296. doi:10.3233/apc210152

Ranstam, J., & Cook, J. (2018). Lasso regression. *British Journal of Surgery,, 105*(10), 1348-1348. doi:10.1002/bjs.10895

Ruß, G., & Brenning, A. (2010). Spatial variable importance assessment for yield prediction in precision agriculture. *Springer Berlin Heidelberg, 21*, 184-195. doi:10.1007/978-3-642-13062-5_18

Sak, H. (. (2014). Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402*, 1128. doi:10.48550/arxiv.1402.1128

Sanit-in, Y., & Saikaew, K. (2019). Prediction of waiting time in one-stop service. *International Journal of Machine Learning and Computing, 9*(3), 322-327. doi:10.18178/ijmlc.2019.9.3.805

Shafique, M., & Hato, E. (2014). Use of acceleration data for transportation mode prediction. . *Transportation, 42*(1), 163-188. doi:10.1007/s11116-014-9541-6

Shrestha, M., & Khanal, S. (2020). Future prospects of precision agriculture in nepal. *Archives of Agriculture and Environmental Science, 5*(3), 397-405. doi:10.26832/24566632.2020.0503023

Solomon, S. D., Zile, M., Swedberg, K., & McMurray, J. (2020). Sacubitril/valsartan across the spectrum of ejection fraction in heart failure. *Circulation, 141*(5), 352-361. doi:10.35940/ijrte.f7280.038620

St-Denis, A., Kneeshaw, D., & Messier, C. (2018). Effect of predation, competition, and facilitation on tree survival and growth in abandoned fields: towards precision restoration. . *Forests, 9*(11), 692. doi:10.3390/f9110692

Subach, T., & Shmeleva, Z. (2022). Introduction of digital innovations in livestock farming. *Iop Conference Series Earth and Environmental Science, 1112*(1), 012079. doi:10.1088/1755-1315/1112/1/012079

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. . *Journal of the Royal Statistical Society Series B (Statistical Methodology, 58*(1), 267-288. doi:10.1111/j.2517-6161.1996.tb02080.x

Vianny, D., John, A., Mohan, S. K., Sarlan, A., & Ahmadian, A. (2022). Water optimization technique for precision irrigation system using IoT and machine learning. . *Sustainable Energy Technologies and Assessments, 52*, 102307.

Wang, G., Hao, Y., & Chen, Y. (2015). Research on the model of variable-rate fertilization in maize based on geographic information system. *6th International Conference on Manufacturing Science and Engineering*, 1756-1763. doi:10.2991/icmse-15.2015.318

Wang, R., Chen, J., Song, Z., & Qi, Z. (2023). Bridging machine learning and redlich–kister theory for solid–liquid equilibria prediction of binary eutectic solvent systems. *Industrial & Engineering Chemistry Research, 62*(12), 5382-5393. doi:10.1021/acs.iecr.3c00054

Xiang, Y. (2022). Using arima-garch model to analyze fluctuation law of international oil price. *Mathematical Problems in Engineering, 2022*, 1-7. doi:10.1155/2022/3936414

Yohanani, E., Frisch, A., Lukyanov, V., Cohen, S., Teitel, M., & Tanny, J. (2022). Estimating evapotranspiration of screenhouse banana plantations using artificial neural network and multiple linear regression models. . *Water, 14*(7), 1130.

Yu, Y. (2022). Gdp economic forecasting model based on improved rbf neural network. . *Mathematical Problems in Engineering, 2022*, 1-11. doi:10.1155/2022/7630268

Zhou, Y., Sun, T., Sun, H., & Yang, F. (2015). Research on combinational forecast models for the traffic flow. . *Mathematical Problems in Engineering, 2015*, 1-10. doi:10.1155/2015/201686

APÉNDICES

Anexo 1: CODIGO DE MODELOS

LIBRARIES

In [ ]:

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
import json
import warnings
from datetime import datetime, date, timedelta
import requests
warnings.filterwarnings("ignore")
```

DATA EXTRACTION

The code that follows up is the conection to the NASA's API called NASA POWER LARC: https://power.larc.nasa.gov/beta/data-access-viewer/. In this site, we find climate data from all around the world. In this particular case, we are extracting the components of the Evapotranspiration Indicator (ETo) using the geographical coordinates from El Guabo, corresponding to El Oro province.

The study period spans from January 1, 2015, to May 15, 2024. The API provides data on an hourly basis, but for our analysis, we need to aggregate it to a daily period. To achieve this, we calculate the mean value per day for each feature. Additionally, for temperature, we consider both the minimum and maximum values per day. The climate variables used in the calculation of ETo (Evapotranspiration) are as follows:

- ALLSKY_SFC_SW_DWN: CERES SYN1deg All Sky Surface Shortwave Downward Irradiance (MJ/m^2/day)
- TOA_SW_DWN: CERES SYN1deg Top-Of-Atmosphere Shortwave Downward Irradiance (MJ/m^2/day)
- T2M: MERRA-2 Temperature at 2 Meters (C)
- RH2M: MERRA-2 Relative Humidity at 2 Meters (%)
- WS2M: MERRA-2 Wind Speed at 2 Meters (m/s)

In [ ]:

```
# Now in format YYYYMMDD
fecha_inicio = date(2015, 1, 1)
fecha_final = date(2016, 1, 1)
fecha_actual = date.today()
dict_result = {}
dict_ac = {}
while fecha_final <= fecha_actual:
  start_date = fecha_inicio
  end_date = fecha_final
  latitude = "-3.207"
  longitude = "-79.7893"
  parameters = "ALLSKY_SFC_SW_DWN,TOA_SW_DWN,T2M,RH2M,WS2M"
  community = "ag"
```

```
header = "true"
time_standard = "lst"

request = requests.get(
    url="https://power.larc.nasa.gov/api/temporal/hourly/point",
    headers={"accept": "application/json"},
    params={
        "start": start_date.strftime('%Y%m%d'),
        "end": end_date.strftime('%Y%m%d'),
        "latitude": latitude,
        "longitude": longitude,
        "parameters": parameters,
        "community": community,
        "header": header,
        "time-standard": time_standard
        }
    )

data = request.json()["properties"]["parameter"]

for i in data:
    if i not in dict_result:
      dict_result[i] = {}
    anterior = ""
    for j in data[i]:
        if data[i][j] == -999.0:
            continue
        actual = j
        if anterior == "":
            if i == "T2M":
                dict_result[i][actual[:8]] = {}
            else:
                dict_result[i][actual[:8]] = 0
            acum = 0
            anterior = actual
            total_local = 0
            list_result = []
        else:
            if actual[:8] == anterior[:8]:

                if i == "T2M":
                    acum += data[i][j]
                else:
                    dict_result[i][actual[:8]] += data[i][j]
                list_result.append(data[i][j])
                total_local += 1
                anterior = actual
            else:
                if i == "T2M":
                    dict_result[i][anterior[:8]
                                   ]["average"] = acum / total_local
```

```
                    dict_result[i][anterior[:8]]["max"] = max(list_result)
                    dict_result[i][anterior[:8]]["min"] = min(list_result)
                else:
                    dict_result[i][anterior[:8]
                                ] = dict_result[i][anterior[:8]
                                                   ] / total_local
                if i == "T2M":
                    dict_result[i][actual[:8]] = {}
                else:
                    dict_result[i][actual[:8]] = 0
                total_local = 0
                acum = 0
                anterior = actual
  fecha_inicio = fecha_final
  fecha_final = fecha_final + timedelta(days=365)

start_date = fecha_inicio
end_date = fecha_actual
latitude = "-3.207"
longitude = "-79.7893"
parameters = "ALLSKY_SFC_SW_DWN,TOA_SW_DWN,T2M,RH2M,WS2M"
community = "ag"
header = "true"
time_standard = "lst"

request = requests.get(
    url="https://power.larc.nasa.gov/api/temporal/hourly/point",
    headers={"accept": "application/json"},
    params={
        "start": start_date.strftime('%Y%m%d'),
        "end": end_date.strftime('%Y%m%d'),
        "latitude": latitude,
        "longitude": longitude,
        "parameters": parameters,
        "community": community,
        "header": header,
        "time-standard": time_standard
        }
    )

data = request.json()["properties"]["parameter"]

for i in data:
    if i not in dict_result:
      dict_result[i] = {}
    anterior = ""
    for j in data[i]:
        if data[i][j] == -999.0:
            continue
        actual = j
        if anterior == "":
```

```
            if i == "T2M":
                dict_result[i][actual[:8]] = {}
            else:
                dict_result[i][actual[:8]] = 0
            acum = 0
            anterior = actual
            total_local = 0
            list_result = []
        else:
            if actual[:8] == anterior[:8]:

                if i == "T2M":
                    acum += data[i][j]
                else:
                    dict_result[i][actual[:8]] += data[i][j]
                list_result.append(data[i][j])
                total_local += 1
                anterior = actual
            else:
                if i == "T2M":
                    dict_result[i][anterior[:8]
                                   ]["average"] = acum / total_local
                    dict_result[i][anterior[:8]]["max"] = max(list_result)
                    dict_result[i][anterior[:8]]["min"] = min(list_result)
                else:
                    dict_result[i][anterior[:8]
                                   ] = dict_result[i][anterior[:8]
                                                      ] / total_local
                if i == "T2M":
                    dict_result[i][actual[:8]] = {}
                else:
                    dict_result[i][actual[:8]] = 0
                total_local = 0
                acum = 0
                anterior = actual



dict_new = {"T2MAVG": {}, "T2MMAX": {}, "T2MMIN": {}}
for i in dict_result:
    if i == "T2M":
        for j in dict_result[i]:
            if dict_result[i][j] == {}:
                continue
            dict_new["T2MAVG"][j] = dict_result[i][j]["average"]
            dict_new["T2MMAX"][j] = dict_result[i][j]["max"]
            dict_new["T2MMIN"][j] = dict_result[i][j]["min"]

dict_result["T2MAVG"] = dict_new["T2MAVG"]
dict_result["T2MMAX"] = dict_new["T2MMAX"]
dict_result["T2MMIN"] = dict_new["T2MMIN"]
```

```
dict_result.pop("T2M")
data = pd.read_json(json.dumps(dict_result))
```

In [ ]:

```
data
```

Out[ ]:

	ALLSKY_SFC_SW_DWN	TOA_SW_DWN	RH2M	WS2M	T2MAVG	T2MMAX	T2MMIN
20150101	0.604348	1.610870	73.287391	3.065217	25.270000	29.83	22.48
20150102	0.694783	1.610870	70.306522	3.494348	25.953043	30.47	22.48
20150103	0.663478	1.612609	72.963913	3.460435	25.461739	30.47	22.48
20150104	0.563043	1.613043	72.679130	3.441304	25.263913	30.47	22.43
20150105	0.615217	1.614783	72.681304	3.092174	25.320870	30.47	22.26
...	...	...	...	...	...	...	...
20240522	NaN	NaN	83.556957	3.307826	24.523043	31.74	22.33
20240523	NaN	NaN	83.388261	3.143043	24.390435	31.74	21.94
20240524	NaN	NaN	83.461304	2.857391	24.533913	31.74	21.94
20240525	NaN	NaN	83.127391	2.677391	24.943478	31.74	21.94
20240526	NaN	NaN	1453.310000	58.920000	NaN	NaN	NaN

3434 rows × 7 columns

Since we need all the components to calculate the ETo, we're going to check when was the last capture of the irradiance features.

In [ ]:

```
# Validating
last_valid_index = data['ALLSKY_SFC_SW_DWN'].last_valid_index()
print(f"The last valid index for column 'ALLSKY_SFC_SW_DWN' is at row
{last_valid_index}.")
The last valid index for column 'ALLSKY_SFC_SW_DWN' is at row 20240201.
```

In [ ]:

```
# Filtering the data
data = data.iloc[:3318]
data
```

Out[ ]:

	ALLSKY_SFC_SW_DWN	TOA_SW_DWN	RH2M	WS2M	T2MAVG	T2MMAX	T2MMIN
20150101	0.604348	1.610870	73.287391	3.065217	25.270000	29.83	22.48
20150102	0.694783	1.610870	70.306522	3.494348	25.953043	30.47	22.48
20150103	0.663478	1.612609	72.963913	3.460435	25.461739	30.47	22.48
20150104	0.563043	1.613043	72.679130	3.441304	25.263913	30.47	22.43
20150105	0.615217	1.614783	72.681304	3.092174	25.320870	30.47	22.26
...	...	...	...	...	...	...	...
20240127	0.795217	1.642174	78.089130	3.394783	26.493913	31.74	22.60
20240128	0.765652	1.643478	78.638261	3.213913	26.387391	31.74	22.60
20240129	0.603913	1.644783	78.175652	3.435217	26.766087	31.74	22.60
20240130	0.644783	1.645652	83.692609	3.317826	25.948261	31.74	22.60

	ALLSKY_SFC_SW_DWN	TOA_SW_DWN	RH2M	WS2M	T2MAVG	T2MMAX	T2MMIN
20240131	0.403043	1.646957	84.112609	3.718696	25.521304	31.74	22.60

3318 rows × 7 columns

Now, once we have the data in order, we can proceed to used a specialized software to calculate the ETo using the Penman-Montheinth. Once we have done that, we re-upload our data. And The final dataframe is the folloing:

In [ ]:

```
data = pd.read_excel('data.xlsx')
data = data.loc[~data['ETO'].isna()].reset_index(drop=True)
```

MODELOS ARIMA

In [ ]:

```
data = data['ETO'].values
# Divide los datos en conjuntos de entrenamiento y prueba
train_size = int(len(data) * 0.8)
train_data, test_data = data[:train_size], data[train_size:]
```

In [ ]:

```
from pmdarima.arima import auto_arima
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.pyplot as plt
from pmdarima import auto_arima
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.stats.stattools import durbin_watson
from scipy.stats import kstest, norm
```

PRUEBA 1: AUTO-ARIMA

In [ ]:

```
model = auto_arima(train_data, seasonal=True, suppress_warnings=True,
stepwise=True)
model_fit=model.fit(train_data)
```

In [ ]:

```
# Realiza predicciones en el conjunto de prueba
y_pred, conf_int = model.predict(n_periods=len(test_data), return_conf_int=True)
```

In [ ]:

```
model.summary()
```

Out[ ]:

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	2652
Model:	SARIMAX(2, 1, 2)	**Log Likelihood**	194.004
Date:	Thu, 30 May 2024	**AIC**	-378.007
Time:	01:15:55	**BIC**	-348.594

Sample:	0 - 2652			**HQIC**	-367.361	
Covariance Type:	opg					
	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
ar.L1	-0.0339	0.176	-0.192	0.848	-0.379	0.311
ar.L2	0.3398	0.118	2.886	0.004	0.109	0.571
ma.L1	-0.2319	0.172	-1.352	0.176	-0.568	0.104
ma.L2	-0.5815	0.157	-3.708	0.000	-0.889	-0.274
sigma2	0.0506	0.001	46.471	0.000	0.048	0.053

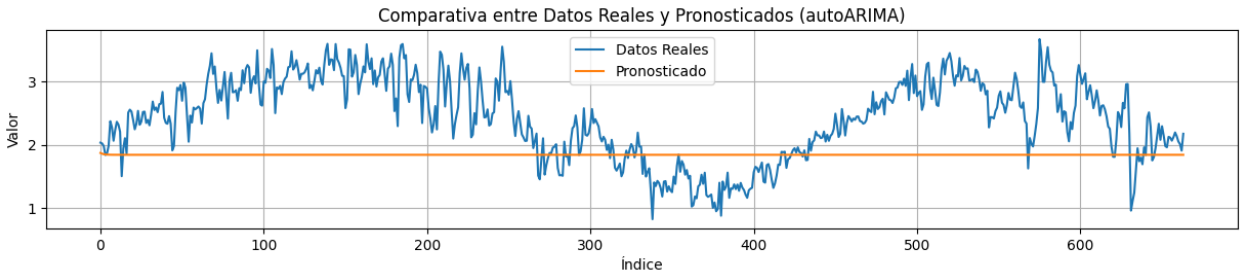
Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	233.97
Prob(Q):	0.99	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.28	**Skew:**	-0.35
Prob(H) (two-sided):	0.00	**Kurtosis:**	4.27

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [ ]:

```
# Grafica los datos reales y las predicciones

indices = np.arange(len(test_data))
plt.figure(figsize=(15, 2.5))
plt.plot(indices, test_data, label="Datos Reales")
plt.plot(indices, y_pred, label="Pronosticado")
plt.title("Comparativa entre Datos Reales y Pronosticados (autoARIMA)")
plt.xlabel("Índice")
plt.ylabel("Valor")
plt.legend()
plt.grid(True)
plt.show()
```



EVALUACIÓN DE LOS RESIDUOS DEL MODELO

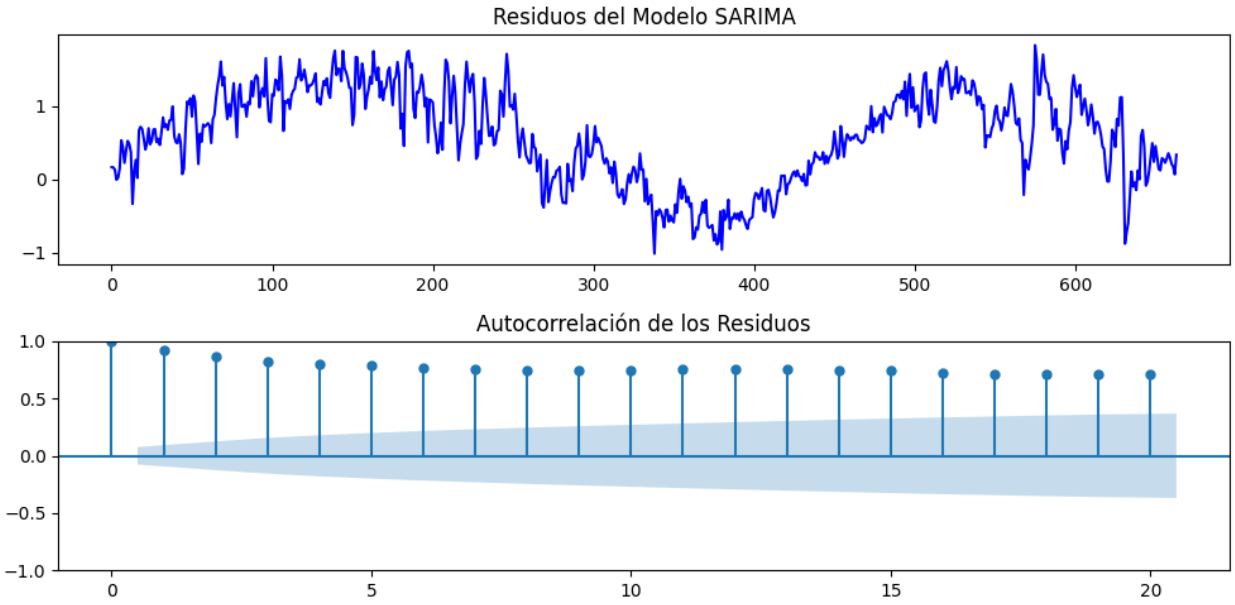
In [ ]:

```
# Paso 1: Ajustar el modelo SARIMA
model = auto_arima(train_data, seasonal=True, suppress_warnings=True,
                   stepwise=True)
model_fit = model.fit(train_data)

# Paso 2: Predecir los valores para los datos de prueba
y_pred, conf_int = model.predict(n_periods=len(test_data), return_conf_int=True)
```

```
# Paso 3: Calcular los residuos
residuals = test_data - y_pred
residuals = np.array(residuals, dtype=np.float64)
```

In [ ]:

```
# Paso 4: Evaluar la autocorrelación de los residuos utilizando ACF y gráficos
facet
fig, axes = plt.subplots(2, 1, figsize=(10, 5))

# Graficar los residuos del modelo SARIMA
axes[0].plot(residuals, color='blue')
axes[0].set_title('Residuos del Modelo SARIMA')

# Graficar la autocorrelación de los residuos
plot_acf(residuals, lags=20, ax=axes[1])
axes[1].set_title('Autocorrelación de los Residuos')

plt.tight_layout()
plt.show()
```



In [ ]:

```
# Paso 5: Prueba de Durbin-Watson para evaluar la autocorrelación
dw_statistic = durbin_watson(residuals)
print(f'Prueba de Durbin-Watson:')
print(f'Estadístico DW: {dw_statistic}')

# Paso 6: Prueba de normalidad de los residuos (Kolmogorov-Smirnov)
residuals_standardized = (residuals - np.mean(residuals)) / np.std(residuals)
ks_statistic, ks_pvalue = kstest(residuals_standardized, 'norm')
print(f'Prueba de Normalidad (Kolmogorov-Smirnov):')
print(f'Estadístico KS: {ks_statistic}')
print(f'Valor p: {ks_pvalue}')
Prueba de Durbin-Watson:
```

```
Estadístico DW: 0.0811099585997244
Prueba de Normalidad (Kolmogorov-Smirnov):
Estadístico KS: 0.07014118703378425
Valor p: 0.002761107143673073
```

PRUEBA 2: SARIMA MANUAL

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.metrics import mean_squared_error
from math import sqrt
```

In [ ]:

```
data = pd.read_excel('data.xlsx')
data = data.loc[~data['ETO'].isna()].reset_index(drop=True)
df = data[['fecha', 'ETO']]
df
```

Out[ ]:

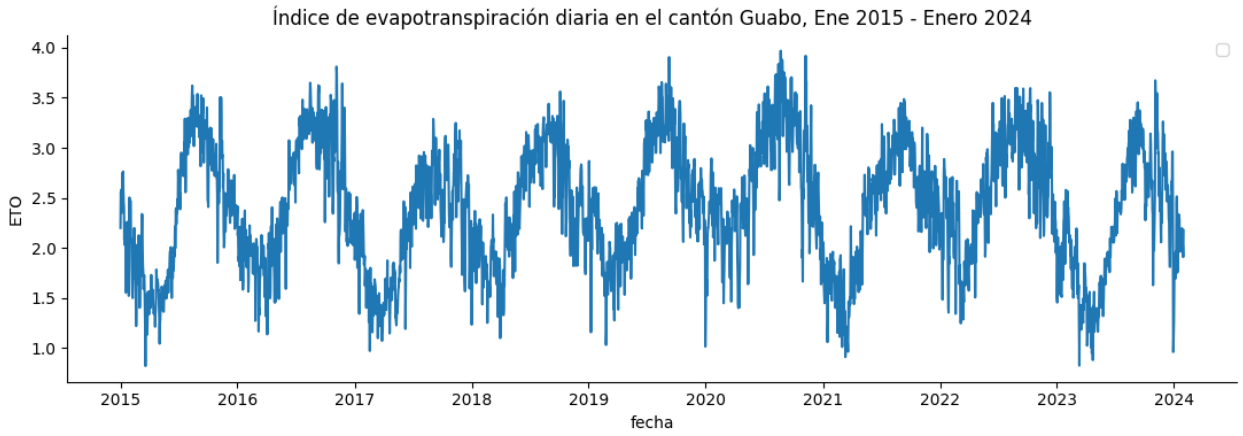
	fecha	ETO
0	2015-01-01	2.200864
1	2015-01-02	2.554714
2	2015-01-03	2.529351
3	2015-01-04	2.584235
4	2015-01-05	2.352408
...	...	...
3311	2024-01-27	2.140035
3312	2024-01-28	2.055444
3313	2024-01-29	2.013823
3314	2024-01-30	1.910303
3315	2024-01-31	2.177274

3316 rows × 2 columns

In [ ]:

```
# Convierte la columna 'fecha' a un formato de fecha
df['fecha'] = pd.to_datetime(df['fecha'])

# Función para graficar la serie temporal
def time_series_multiline(df, timelike_colname, value_colname, series_colname,
                          figscale=1, mpl_palette_name='Dark2'):
    figsize = (11 * figscale, 4 * figscale)
    palette = list(sns.color_palette(mpl_palette_name))

    fig, ax = plt.subplots(figsize=figsize)
    df = df.sort_values(timelike_colname, ascending=True)
```

```
    if series_colname:
        for series_name, series in df.groupby(series_colname):
            plt.plot(series[timelike_colname], series[value_colname],
                     label=series_name)
    else:
        plt.plot(df[timelike_colname], df[value_colname])

    plt.legend()
    sns.despine()
    plt.xlabel(timelike_colname)
    plt.ylabel(value_colname)
    plt.title('Índice de evapotranspiración diaria en el cantón Guabo, Ene 2015 -
Enero 2024')

# Llama a la función para crear la gráfica
time_series_multiline(df, 'fecha', 'ETO', None)

# Muestra la gráfica
plt.tight_layout()
plt.show()
```



INSPECCION DE COMPONENTES DE LA SERIE

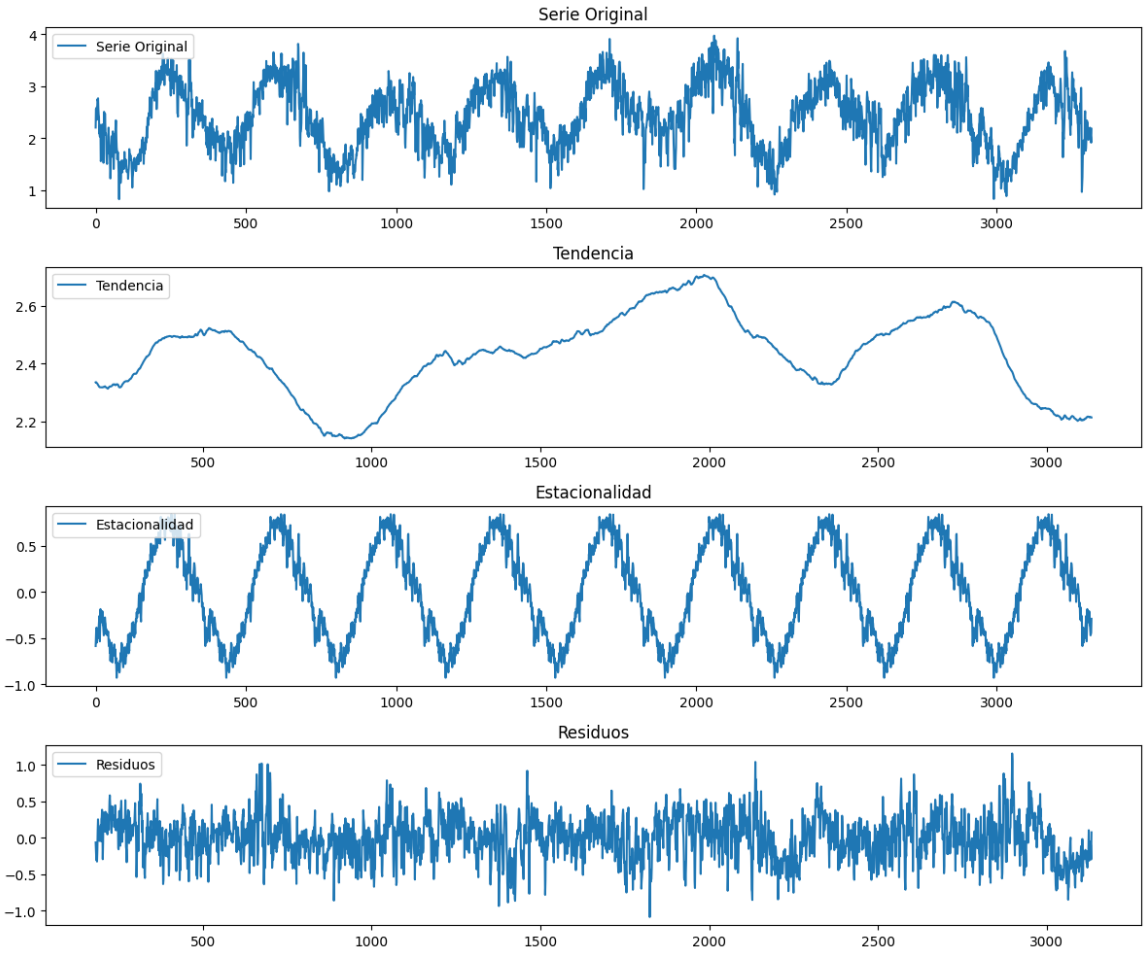
In [ ]:

```
from statsmodels.tsa.seasonal import seasonal_decompose
```

In [ ]:

```
# Descomposición de la serie temporal
result = seasonal_decompose(df['ETO'], model='additive', period=365)  # Ajusta
'period' según tu serie temporal

# Gráficos de tendencia, estacionalidad y residuos
plt.figure(figsize=(12, 10))

plt.subplot(411)
plt.plot(df.index, df['ETO'], label='Serie Original')
plt.legend(loc='upper left')
```

```
plt.title('Serie Original')

plt.subplot(412)
plt.plot(result.trend, label='Tendencia')
plt.legend(loc='upper left')
plt.title('Tendencia')

plt.subplot(413)
plt.plot(result.seasonal, label='Estacionalidad')
plt.legend(loc='upper left')
plt.title('Estacionalidad')

plt.subplot(414)
plt.plot(result.resid, label='Residuos')
plt.legend(loc='upper left')
plt.title('Residuos')

plt.tight_layout()
plt.show()
```

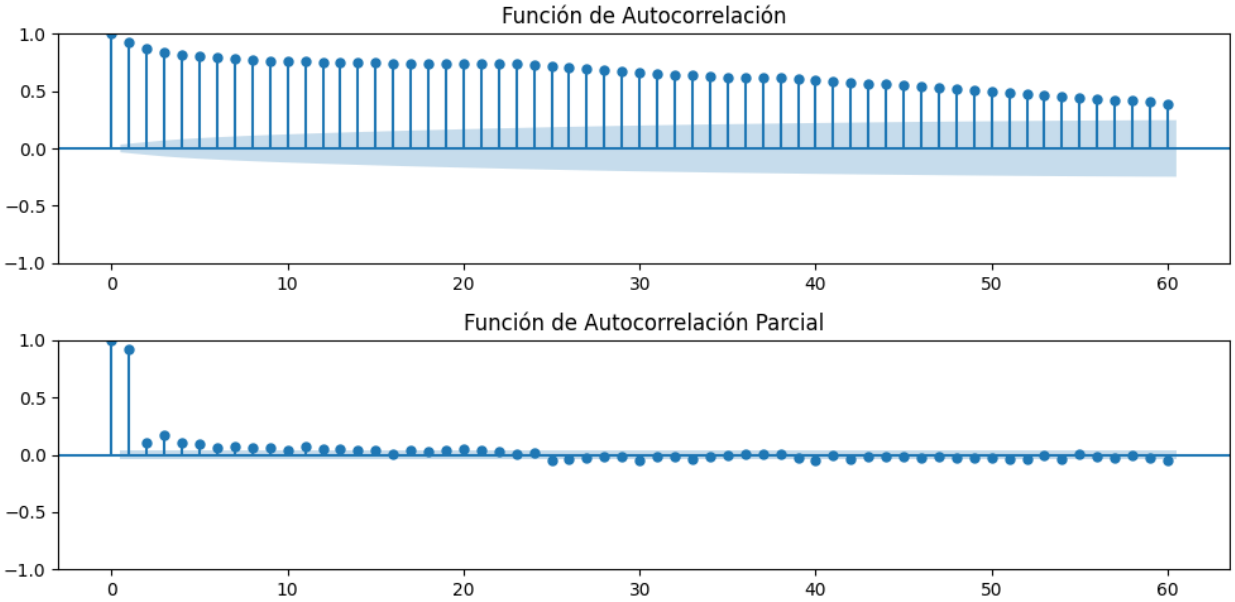


In [ ]:

```
from statsmodels.tsa.stattools import acf, pacf
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import statsmodels.api as sm
```

In [ ]:

```
fig, axes = plt.subplots(2, 1, figsize=(10, 5))

plot_acf(df['ETO'], lags=60, ax=axes[0])
axes[0].set_title('Función de Autocorrelación')

plot_pacf(df['ETO'], lags=60, ax=axes[1])
axes[1].set_title('Función de Autocorrelación Parcial')

plt.tight_layout()
plt.show()
```



TEST ESTADISTICOS: RAIZ UNITARIA

In [ ]:

```
from statsmodels.tsa.stattools import adfuller
result = adfuller(df['ETO'])
adf_statistic = result[0]
p_value = result[1]
critical_values = result[4]

print(f'Estadística ADF: {adf_statistic}')
print(f'Valor p: {p_value}')
print('Valores críticos:')
for key, value in critical_values.items():
    print(f'   {key}: {value}')

if p_value <= 0.05:
    print('La serie es estacionaria (p-valor <= 0.05).')
else:
    print('La serie no es estacionaria (p-valor > 0.05).')
Estadística ADF: -3.487465490209176
Valor p: 0.008314902946253647
```

```
Valores críticos:
   1%: -3.4323416055066382
   5%: -2.86241997328205
   10%: -2.5672384281310743
La serie es estacionaria (p-valor <= 0.05).
```

A partir de las gráficas de autocorrelación (MA), autocorrelación parcial (AR), y prueba de estacionariedad elige los órdenes p, d y q.

- p = 5 # Orden autoregresivo (AR)
- d = 0 # Orden de diferenciación (I)
- q = 1 # Orden de la media móvil (MA)

En cuanto al orden de la estacionalidad, se observa el siguiente comportamiento:.

- P = 1 # Orden de la estacionalidad autoregresiva (SAR)
- D = 1 # Orden de diferenciación estacional (SI)
- Q = 0 # Orden de la media móvil estacional (SMA)
- s = 30 # Frecuencia estacional (por ejemplo, 30 para datos mensuales)

ENTRENAMIENTO DEL MODELO

In [ ]:

```
# Dividir los datos en conjuntos de entrenamiento (train) y prueba (test)
train_size = int(len(df) * 0.8)  # Por ejemplo, usando el 80% de los datos para
entrenamiento
train, test = df[:train_size], df[train_size:]
```

In [ ]:

```
p = 5 # Orden autoregresivo (AR)
d = 0 # Orden de diferenciación (I)
q = 1 # Orden de la media móvil (MA)
P = 1 # Orden de la estacionalidad autoregresiva (SAR)
D = 1 # Orden de diferenciación estacional (SI)
Q = 0 # Orden de la media móvil estacional (SMA)
s = 30 # Frecuencia estacional (por ejemplo, 30 para datos mensuales)
```

In [ ]:

```
# Ajustar el modelo ARIMA o SARIMA al conjunto de entrenamiento
if s > 0:
    model = sm.tsa.SARIMAX(train['ETO'], order=(p, d, q), seasonal_order=(P, D, Q,
s))
else:
    model = sm.tsa.SARIMAX(train['ETO'], order=(p, d, q), seasonal_order=(0, 0, 0,
0))  # Componente estacional desactivado
model_fit = model.fit(disp=False)
/usr/local/lib/python3.10/dist-packages/statsmodels/base/model.py:607:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

In [ ]:

```
model_fit.summary()
```

Out[ ]:

SARIMAX Results

Dep. Variable:	ETO	**No. Observations:**	2652
Model:	SARIMAX(5, 0, 1)x(1, 1, [], 30)	**Log Likelihood**	-368.533
Date:	Thu, 30 May 2024	**AIC**	753.066
Time:	01:18:21	**BIC**	800.039
Sample:	0	**HQIC**	770.078
	- 2652		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	1.6157	0.027	59.375	0.000	1.562	1.669
ar.L2	-0.6875	0.038	-18.100	0.000	-0.762	-0.613
ar.L3	0.1237	0.039	3.152	0.002	0.047	0.201
ar.L4	-0.0706	0.036	-1.947	0.051	-0.142	0.000
ar.L5	0.0141	0.021	0.688	0.491	-0.026	0.054
ma.L1	-0.9098	0.021	-43.093	0.000	-0.951	-0.868
ar.S.L30	-0.4909	0.015	-31.968	0.000	-0.521	-0.461
sigma2	0.0773	0.002	39.221	0.000	0.073	0.081

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	28.20
Prob(Q):	0.99	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.33	**Skew:**	-0.16
Prob(H) (two-sided):	0.00	**Kurtosis:**	3.40

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

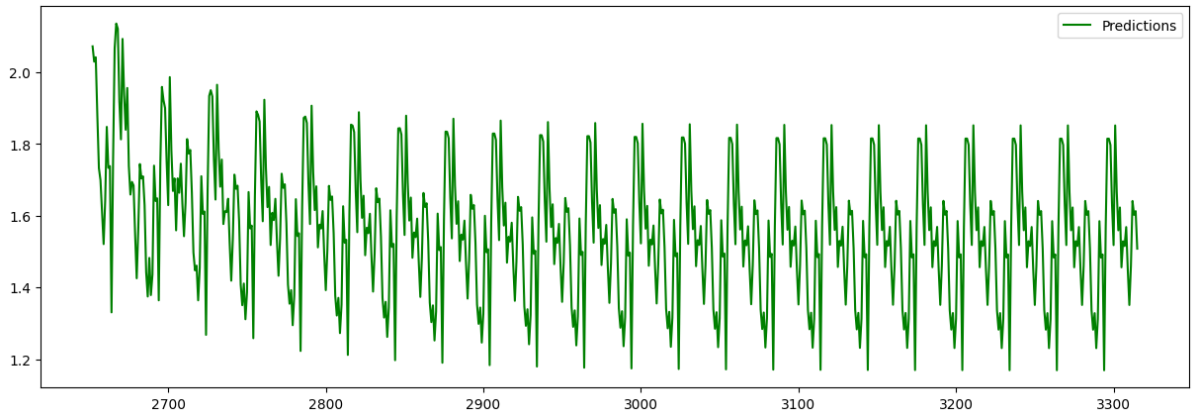
FORECAST

In [ ]:

```
# Paso 4: Realizar pronósticos en el conjunto de prueba
y_pred = model_fit.get_forecast(len(test.index))
y_pred_df = y_pred.conf_int(alpha = 0.05)
y_pred_df["Predictions"] = model_fit.predict(start = y_pred_df.index[0], end =
y_pred_df.index[-1])
y_pred_df.index = test.index
y_pred_out = y_pred_df["Predictions"]
```

In [ ]:

```
plt.figure(figsize=(15, 5))
plt.plot(y_pred_out, color='green', label = 'Predictions')
plt.legend()
```



INSPECCION GRAFICA DEL AJUSTE

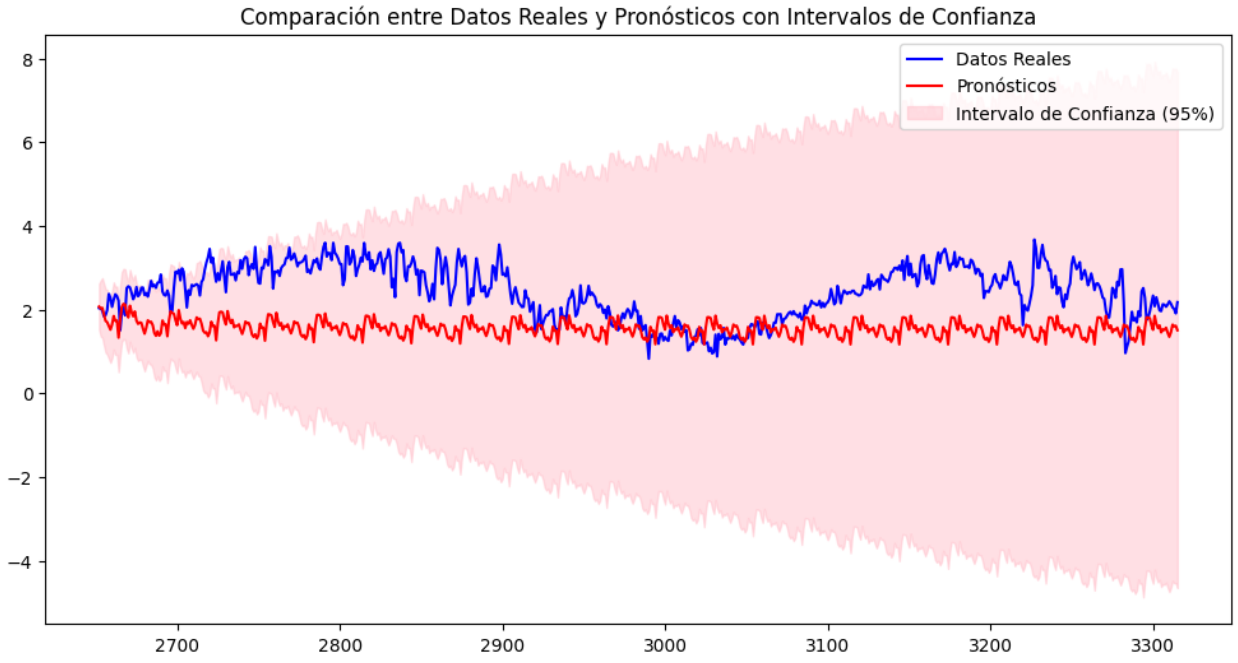
In [ ]:

```
from sklearn.metrics import mean_squared_error, mean_absolute_error
import statsmodels.api as sm
```

In [ ]:

```
def sarimax_forecast(df, p=5, d=0, q=1, P=1, D=1, Q=0, s=30):
    # Dividir los datos en conjuntos de entrenamiento (train) y prueba (test)
    train_size = int(len(df) * 0.8)  # Por ejemplo, usando el 80% de los datos
para entrenamiento
    train, test = df[:train_size], df[train_size:]

    # Ajustar el modelo ARIMA o SARIMA al conjunto de entrenamiento
    if s > 0:
        model = sm.tsa.SARIMAX(train['ETO'], order=(p, d, q), seasonal_order=(P,
D, Q, s))
    else:
        model = sm.tsa.SARIMAX(train['ETO'], order=(p, d, q), seasonal_order=(0,
0, 0, 0))  # Componente estacional desactivado

    model_fit = model.fit(disp=False)

    # Realizar pronósticos en el conjunto de prueba
    y_pred = model_fit.get_forecast(len(test.index))
    forecast_ci = y_pred.conf_int()
    y_pred_df = forecast_ci.copy()
    y_pred_df["Predictions"] = y_pred.predicted_mean
    y_pred_df.index = test.index
    y_pred_out = y_pred_df["Predictions"]

    # Gráfico de comparación entre datos reales y pronósticos con intervalos de
confianza
    plt.figure(figsize=(12, 6))
    plt.plot(test.index, test['ETO'], label='Datos Reales', color='blue')
    plt.plot(test.index, y_pred_df["Predictions"], label='Pronósticos',
color='red')
```

```
    plt.fill_between(test.index, forecast_ci['lower ETO'], forecast_ci['upper
ETO'], color='pink', alpha=0.5, label='Intervalo de Confianza (95%)')
    plt.legend()
    plt.title('Comparación entre Datos Reales y Pronósticos con Intervalos de
Confianza')
    plt.show()

    return y_pred_out

# Ejecutar la función
predictions = sarimax_forecast(df)
/usr/local/lib/python3.10/dist-packages/statsmodels/base/model.py:607:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```



In [ ]:

```
def calculate_metrics(y_true, y_pred):
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_true, y_pred)
    mase = mae / np.mean(np.abs(np.diff(y_true)))

    print(f'MSE: {mse}')
    print(f'RMSE: {rmse}')
    print(f'MASE: {mase}')

    return mse, rmse, mase


# Eliminar la observación correspondiente a la fecha '2022-04-08' de las
predicciones
```

```
# predictions = predictions.drop(pd.Timestamp('2022-04-08'))

# Asegurarse de que 'test' y 'predictions' tienen la misma longitud
if len(test['ETO']) > len(predictions):
    test = test.iloc[:len(predictions)]
elif len(test['ETO']) < len(predictions):
    predictions = predictions[:len(test['ETO'])]

# Ahora puedes calcular las métricas
mse, rmse, mase = calculate_metrics(test['ETO'], predictions)
MSE: 1.2020850356860753
RMSE: 1.0963963862062276
MASE: 5.132077887206841
```

MODELO LONG SHORT TERM MEMORY (LSTM)

In [ ]:

```
data = pd.read_excel('data.xlsx')
data = data.loc[~data['ETO'].isna()].reset_index(drop=True)
df = data[['fecha', 'ETO']]
df.index = df.pop('fecha')
df
```

Out[ ]:

	ETO
fecha	
2015-01-01	2.200864
2015-01-02	2.554714
2015-01-03	2.529351
2015-01-04	2.584235
2015-01-05	2.352408
...	...
2024-01-27	2.140035
2024-01-28	2.055444
2024-01-29	2.013823
2024-01-30	1.910303
2024-01-31	2.177274

3316 rows × 1 columns

PRUEBA 1: RED SECUENCIAL

TRANSFORMACIÓN DE DATAFRAME

A continuación, se muestra la transformación del dataset (df) a para realizar el entrenamiento de la red LSTM de manera secuencial. El modelo tomará los valores de 7 días para realizar la predicción

de los 7 siguientes datos. Es decir, en función de la ETO correspondiente a la ventana de los últimos 7 días, se predicen los valores para los días siguiente semana.

In [ ]:

```
import numpy as np
import pandas as pd
import datetime
```

In [ ]:

```
def str_to_datetime(date_str):
    return datetime.datetime.strptime(date_str, '%Y-%m-%d')

def df_to_windowed_df(dataframe, first_date_str, last_date_str, n=7):
    first_date = str_to_datetime(first_date_str)
    last_date  = str_to_datetime(last_date_str)

    target_date = first_date

    dates = []
    X, Y = [], []

    last_time = False
    while True:
        df_subset = dataframe.loc[:target_date].tail(n+1)

        if len(df_subset) != n+1:
            print(f'Warning: Not enough data for date {target_date}. Reducing window size to {len(df_subset)-1}.')
            n_actual = len(df_subset)-1
            values = df_subset['ETO'].to_numpy()
        else:
            n_actual = n
            values = df_subset['ETO'].to_numpy()

        x, y = values[:-1], values[-1]

        dates.append(target_date)
```

```
        X.append(x)
        Y.append(values[-7:])  # Cambio: Seleccionar los últimos 7 valores

        next_week = dataframe.loc[target_date:target_date+datetime.timedelta(days=7)]
        next_datetime_str = str(next_week.head(2).tail(1).index.values[0])
        next_date_str = next_datetime_str.split('T')[0]
        year_month_day = next_date_str.split('-')
        year, month, day = year_month_day
        next_date = datetime.datetime(day=int(day), month=int(month), year=int(year))

        if last_time:
            break
        target_date = next_date

        if target_date == last_date:
            last_time = True
    ret_df = pd.DataFrame({})
    ret_df['Target Date'] = dates

    X = np.array(X)
    for i in range(0, n_actual):
        X[:, i]
        ret_df[f'Target-{n_actual-i}'] = X[:, i]

    for i in range(1, 8):  # Cambio: Agregar los 7 objetivos
        ret_df[f'Target+{i}'] = np.array(Y)[:, -i]

    return ret_df

# Start day second time around: '2021-03-25'
windowed_df = df_to_windowed_df(df,
                                '2015-01-08',
                                '2023-08-27',
                                n=7)
```

```
windowed_df
```

Out[ ]:

	Target Date	Target-7	Target-6	Target-5	Target-4	Target-3	Target-2	Target-1	Target+1	Target+2	Target+3	Target+4	Target+5	Target+6	Target+7
0	2015-01-08	2.200864	2.554714	2.529351	2.584235	2.352408	2.561922	2.744921	2.714976	2.744921	2.561922	2.352408	2.584235	2.529351	2.554714
1	2015-01-09	2.554714	2.529351	2.584235	2.352408	2.561922	2.744921	2.714976	2.764029	2.714976	2.744921	2.561922	2.352408	2.584235	2.529351
2	2015-01-10	2.529351	2.584235	2.352408	2.561922	2.744921	2.714976	2.764029	2.565488	2.764029	2.714976	2.744921	2.561922	2.352408	2.584235
3	2015-01-11	2.584235	2.352408	2.561922	2.744921	2.714976	2.764029	2.565488	2.453584	2.565488	2.764029	2.714976	2.744921	2.561922	2.352408
4	2015-01-12	2.352408	2.561922	2.744921	2.714976	2.764029	2.565488	2.453584	2.407247	2.453584	2.565488	2.764029	2.714976	2.744921	2.561922
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3147	2023-08-23	2.709467	3.076919	3.279738	2.820754	3.097499	2.772122	2.824146	2.851183	2.824146	2.772122	3.097499	2.820754	3.279738	3.076919
3148	2023-08-24	3.076919	3.279738	2.820754	3.097499	2.772122	2.824146	2.851183	2.554509	2.851183	2.824146	2.772122	3.097499	2.820754	3.279738
3149	2023-08-25	3.279738	2.820754	3.097499	2.772122	2.824146	2.851183	2.554509	2.642612	2.554509	2.851183	2.824146	2.772122	3.097499	2.820754
3150	2023-08-26	2.820754	3.097499	2.772122	2.824146	2.851183	2.554509	2.642612	3.080105	2.642612	2.554509	2.851183	2.824146	2.772122	3.097499
3151	2023-08-27	3.097499	2.772122	2.824146	2.851183	2.554509	2.642612	3.080105	3.277544	3.080105	2.642612	2.554509	2.851183	2.824146	2.772122

3152 rows × 15 columns

In [ ]:

```
def windowed_df_to_date_X_y(windowed_dataframe):
    df_as_np = windowed_dataframe.to_numpy()

    dates = df_as_np[:, 0]

    X = df_as_np[:, 1:-7, np.newaxis]    # Seleccionar las columnas correspondientes a las características X
    y = df_as_np[:, -7:]

    return dates, X.astype(np.float32), y.astype(np.float32)
```

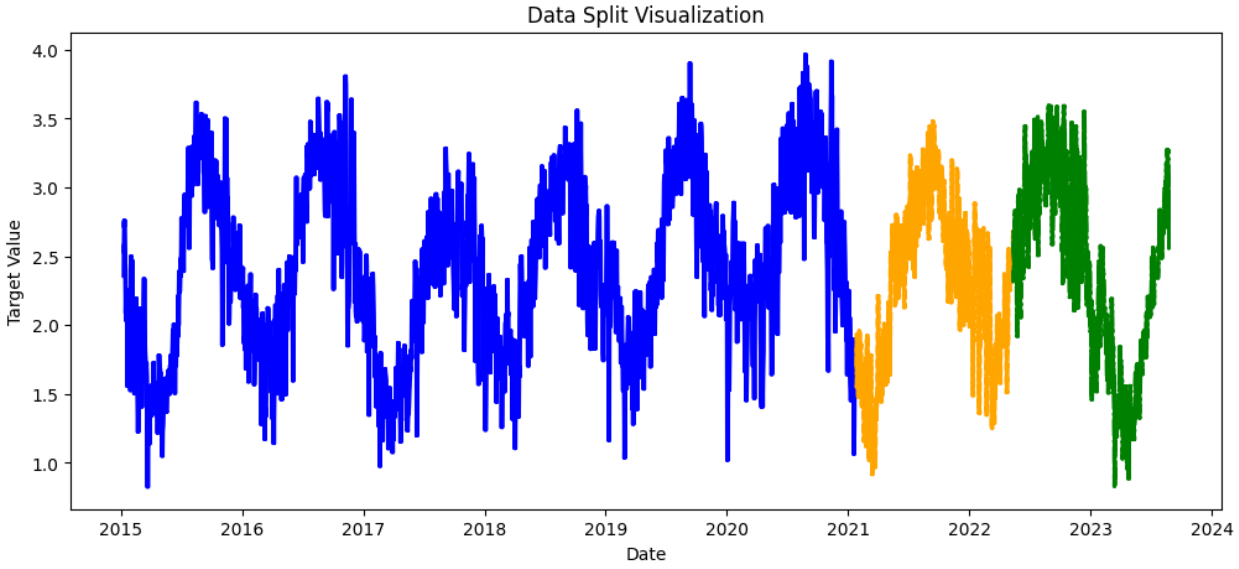
```
dates, X, y = windowed_df_to_date_X_y(windowed_df)

dates.shape, X.shape, y.shape
```

Out[ ]:

```
((3152,), (3152, 7, 1), (3152, 7))
```

TRAIN, VALIDATION, TEST.

```
# Calculate the split points
q_70 = int(len(dates) * 0.7)
q_85 = int(len(dates) * 0.85)
# Split the data
dates_train, X_train, y_train = dates[:q_70], X[:q_70], y[:q_70]
dates_val, X_val, y_val = dates[q_70:q_85], X[q_70:q_85], y[q_70:q_85]
dates_test, X_test, y_test = dates[q_85:], X[q_85:], y[q_85:]

# Create the plot
plt.figure(figsize=(12, 5))  # Adjust the figure size if needed
plt.plot(dates_train, y_train, label='Train', linestyle='-', color='blue')
plt.plot(dates_val,     y_val,     label='Validation',     linestyle='--',
color='orange')
plt.plot(dates_test, y_test, label='Test', linestyle='-.', color='green')

# Customize the plot
plt.xlabel('Date')
plt.ylabel('Target Value')
plt.title('Data Split Visualization')

# Show the plot
plt.show()
```



DEFINICIÓN Y ENTRENAMIENTO DE LA RED

In [ ]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import layers
```

In [ ]:

```
model = Sequential([
    layers.Input((7, 1)),
    layers.LSTM(64),
    layers.Dense(32, activation='relu'),
    layers.Dense(32, activation='relu'),
    layers.Dense(7)  # Corregir el número de unidades en la última capa
])
model.compile(loss='mse',
              optimizer=Adam(learning_rate=0.001),
              metrics=['mean_absolute_error'])

model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=100)
```

```
Epoch 1/100
69/69 [==============================] - 4s 22ms/step - loss: 3.0936 - mean_absolute_error: 1.4451 - val_loss: 0.2046 - val_mean_absolute_error: 0.3714
Epoch 2/100
```

```
69/69 [==============================] - 1s 11ms/step - loss: 0.0934 -
mean_absolute_error: 0.2275 - val_loss: 0.0460 - val_mean_absolute_error:
0.1649

Epoch 3/100

69/69 [==============================] - 1s 11ms/step - loss: 0.0468 -
mean_absolute_error: 0.1646 - val_loss: 0.0425 - val_mean_absolute_error:
0.1559

Epoch 4/100

69/69 [==============================] - 1s 10ms/step - loss: 0.0438 -
mean_absolute_error: 0.1582 - val_loss: 0.0389 - val_mean_absolute_error:
0.1491

Epoch 5/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0409 -
mean_absolute_error: 0.1525 - val_loss: 0.0368 - val_mean_absolute_error:
0.1451

Epoch 6/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0380 -
mean_absolute_error: 0.1464 - val_loss: 0.0345 - val_mean_absolute_error:
0.1406

Epoch 7/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0361 -
mean_absolute_error: 0.1429 - val_loss: 0.0333 - val_mean_absolute_error:
0.1380

Epoch 8/100

69/69 [==============================] - 1s 7ms/step - loss: 0.0340 -
mean_absolute_error: 0.1383 - val_loss: 0.0335 - val_mean_absolute_error:
0.1392

Epoch 9/100

69/69 [==============================] - 1s 7ms/step - loss: 0.0326 -
mean_absolute_error: 0.1354 - val_loss: 0.0301 - val_mean_absolute_error:
0.1315

Epoch 10/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0314 -
mean_absolute_error: 0.1327 - val_loss: 0.0295 - val_mean_absolute_error:
0.1305

Epoch 11/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0298 -
mean_absolute_error: 0.1294 - val_loss: 0.0287 - val_mean_absolute_error:
0.1288

Epoch 12/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0288 -
mean_absolute_error: 0.1271 - val_loss: 0.0272 - val_mean_absolute_error:
```

```
0.1252

Epoch 13/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0275 -
mean_absolute_error: 0.1235 - val_loss: 0.0262 - val_mean_absolute_error:
0.1231

Epoch 14/100

69/69 [==============================] - 1s 8ms/step - loss: 0.0257 -
mean_absolute_error: 0.1190 - val_loss: 0.0242 - val_mean_absolute_error:
0.1181

Epoch 15/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0236 -
mean_absolute_error: 0.1138 - val_loss: 0.0216 - val_mean_absolute_error:
0.1099

Epoch 16/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0230 -
mean_absolute_error: 0.1121 - val_loss: 0.0222 - val_mean_absolute_error:
0.1131

Epoch 17/100

69/69 [==============================] - 1s 7ms/step - loss: 0.0222 -
mean_absolute_error: 0.1100 - val_loss: 0.0219 - val_mean_absolute_error:
0.1123

Epoch 18/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0215 -
mean_absolute_error: 0.1076 - val_loss: 0.0202 - val_mean_absolute_error:
0.1060

Epoch 19/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0213 -
mean_absolute_error: 0.1070 - val_loss: 0.0220 - val_mean_absolute_error:
0.1115

Epoch 20/100

69/69 [==============================] - 1s 8ms/step - loss: 0.0208 -
mean_absolute_error: 0.1054 - val_loss: 0.0205 - val_mean_absolute_error:
0.1063

Epoch 21/100

69/69 [==============================] - 1s 9ms/step - loss: 0.0207 -
mean_absolute_error: 0.1054 - val_loss: 0.0206 - val_mean_absolute_error:
0.1086

Epoch 22/100

69/69 [==============================] - 1s 13ms/step - loss: 0.0208 -
mean_absolute_error: 0.1058 - val_loss: 0.0203 - val_mean_absolute_error:
0.1064

Epoch 23/100
```

```
69/69 [==============================] - 1s 18ms/step - loss: 0.0204 -
mean_absolute_error: 0.1044 - val_loss: 0.0201 - val_mean_absolute_error:
0.1054

Epoch 24/100

69/69 [==============================] - 2s 33ms/step - loss: 0.0202 -
mean_absolute_error: 0.1034 - val_loss: 0.0208 - val_mean_absolute_error:
0.1082

Epoch 25/100

69/69 [==============================] - 2s 23ms/step - loss: 0.0201 -
mean_absolute_error: 0.1033 - val_loss: 0.0203 - val_mean_absolute_error:
0.1064

Epoch 26/100

69/69 [==============================] - 1s 16ms/step - loss: 0.0200 -
mean_absolute_error: 0.1031 - val_loss: 0.0196 - val_mean_absolute_error:
0.1050

Epoch 27/100

69/69 [==============================] - 1s 15ms/step - loss: 0.0201 -
mean_absolute_error: 0.1035 - val_loss: 0.0207 - val_mean_absolute_error:
0.1081

Epoch 28/100

69/69 [==============================] - 1s 12ms/step - loss: 0.0198 -
mean_absolute_error: 0.1023 - val_loss: 0.0202 - val_mean_absolute_error:
0.1057

Epoch 29/100

69/69 [==============================] - 1s 13ms/step - loss: 0.0198 -
mean_absolute_error: 0.1022 - val_loss: 0.0198 - val_mean_absolute_error:
0.1028

Epoch 30/100

69/69 [==============================] - 1s 16ms/step - loss: 0.0193 -
mean_absolute_error: 0.1004 - val_loss: 0.0184 - val_mean_absolute_error:
0.0995

Epoch 31/100

69/69 [==============================] - 1s 13ms/step - loss: 0.0197 -
mean_absolute_error: 0.1019 - val_loss: 0.0185 - val_mean_absolute_error:
0.1002

Epoch 32/100

69/69 [==============================] - 1s 12ms/step - loss: 0.0192 -
mean_absolute_error: 0.1005 - val_loss: 0.0182 - val_mean_absolute_error:
0.0992

Epoch 33/100

69/69 [==============================] - 1s 12ms/step - loss: 0.0191 -
mean_absolute_error: 0.0999 - val_loss: 0.0185 - val_mean_absolute_error:
```

```
0.0994

Epoch 34/100

69/69 [==============================] - 1s 9ms/step - loss: 0.0188 -
mean_absolute_error: 0.0988 - val_loss: 0.0182 - val_mean_absolute_error:
0.0996

Epoch 35/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0188 -
mean_absolute_error: 0.0990 - val_loss: 0.0180 - val_mean_absolute_error:
0.0989

Epoch 36/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0188 -
mean_absolute_error: 0.0986 - val_loss: 0.0192 - val_mean_absolute_error:
0.1026

Epoch 37/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0184 -
mean_absolute_error: 0.0973 - val_loss: 0.0181 - val_mean_absolute_error:
0.0985

Epoch 38/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0181 -
mean_absolute_error: 0.0964 - val_loss: 0.0183 - val_mean_absolute_error:
0.1012

Epoch 39/100

69/69 [==============================] - 1s 8ms/step - loss: 0.0178 -
mean_absolute_error: 0.0956 - val_loss: 0.0173 - val_mean_absolute_error:
0.0953

Epoch 40/100

69/69 [==============================] - 1s 10ms/step - loss: 0.0180 -
mean_absolute_error: 0.0959 - val_loss: 0.0173 - val_mean_absolute_error:
0.0966

Epoch 41/100

69/69 [==============================] - 1s 10ms/step - loss: 0.0177 -
mean_absolute_error: 0.0950 - val_loss: 0.0169 - val_mean_absolute_error:
0.0937

Epoch 42/100

69/69 [==============================] - 1s 11ms/step - loss: 0.0173 -
mean_absolute_error: 0.0933 - val_loss: 0.0167 - val_mean_absolute_error:
0.0934

Epoch 43/100

69/69 [==============================] - 1s 11ms/step - loss: 0.0170 -
mean_absolute_error: 0.0922 - val_loss: 0.0178 - val_mean_absolute_error:
0.0981

Epoch 44/100
```

```
69/69 [==============================] - 1s 11ms/step - loss: 0.0174 -
mean_absolute_error: 0.0943 - val_loss: 0.0161 - val_mean_absolute_error:
0.0909

Epoch 45/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0166 -
mean_absolute_error: 0.0911 - val_loss: 0.0161 - val_mean_absolute_error:
0.0909

Epoch 46/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0165 -
mean_absolute_error: 0.0904 - val_loss: 0.0166 - val_mean_absolute_error:
0.0922

Epoch 47/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0167 -
mean_absolute_error: 0.0917 - val_loss: 0.0164 - val_mean_absolute_error:
0.0928

Epoch 48/100

69/69 [==============================] - 1s 7ms/step - loss: 0.0160 -
mean_absolute_error: 0.0884 - val_loss: 0.0153 - val_mean_absolute_error:
0.0870

Epoch 49/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0160 -
mean_absolute_error: 0.0887 - val_loss: 0.0159 - val_mean_absolute_error:
0.0912

Epoch 50/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0159 -
mean_absolute_error: 0.0879 - val_loss: 0.0150 - val_mean_absolute_error:
0.0870

Epoch 51/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0154 -
mean_absolute_error: 0.0855 - val_loss: 0.0150 - val_mean_absolute_error:
0.0859

Epoch 52/100

69/69 [==============================] - 1s 7ms/step - loss: 0.0157 -
mean_absolute_error: 0.0873 - val_loss: 0.0163 - val_mean_absolute_error:
0.0952

Epoch 53/100

69/69 [==============================] - 1s 7ms/step - loss: 0.0153 -
mean_absolute_error: 0.0857 - val_loss: 0.0154 - val_mean_absolute_error:
0.0872

Epoch 54/100

69/69 [==============================] - 1s 8ms/step - loss: 0.0151 -
mean_absolute_error: 0.0847 - val_loss: 0.0143 - val_mean_absolute_error:
```

```
0.0827

Epoch 55/100

69/69 [==============================] - 1s 8ms/step - loss: 0.0147 -
mean_absolute_error: 0.0837 - val_loss: 0.0141 - val_mean_absolute_error:
0.0816

Epoch 56/100

69/69 [==============================] - 1s 8ms/step - loss: 0.0148 -
mean_absolute_error: 0.0830 - val_loss: 0.0147 - val_mean_absolute_error:
0.0871

Epoch 57/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0150 -
mean_absolute_error: 0.0847 - val_loss: 0.0144 - val_mean_absolute_error:
0.0842

Epoch 58/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0147 -
mean_absolute_error: 0.0836 - val_loss: 0.0138 - val_mean_absolute_error:
0.0809

Epoch 59/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0140 -
mean_absolute_error: 0.0803 - val_loss: 0.0132 - val_mean_absolute_error:
0.0773

Epoch 60/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0138 -
mean_absolute_error: 0.0792 - val_loss: 0.0146 - val_mean_absolute_error:
0.0885

Epoch 61/100

69/69 [==============================] - 1s 7ms/step - loss: 0.0138 -
mean_absolute_error: 0.0797 - val_loss: 0.0131 - val_mean_absolute_error:
0.0797

Epoch 62/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0134 -
mean_absolute_error: 0.0774 - val_loss: 0.0126 - val_mean_absolute_error:
0.0756

Epoch 63/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0132 -
mean_absolute_error: 0.0765 - val_loss: 0.0136 - val_mean_absolute_error:
0.0813

Epoch 64/100

69/69 [==============================] - 1s 7ms/step - loss: 0.0131 -
mean_absolute_error: 0.0761 - val_loss: 0.0134 - val_mean_absolute_error:
0.0829

Epoch 65/100
```

```
69/69 [==============================] - 1s 11ms/step - loss: 0.0132 -
mean_absolute_error: 0.0766 - val_loss: 0.0123 - val_mean_absolute_error:
0.0753

Epoch 66/100

69/69 [==============================] - 1s 10ms/step - loss: 0.0125 -
mean_absolute_error: 0.0732 - val_loss: 0.0120 - val_mean_absolute_error:
0.0737

Epoch 67/100

69/69 [==============================] - 1s 11ms/step - loss: 0.0122 -
mean_absolute_error: 0.0717 - val_loss: 0.0116 - val_mean_absolute_error:
0.0721

Epoch 68/100

69/69 [==============================] - 1s 11ms/step - loss: 0.0119 -
mean_absolute_error: 0.0701 - val_loss: 0.0122 - val_mean_absolute_error:
0.0726

Epoch 69/100

69/69 [==============================] - 1s 10ms/step - loss: 0.0120 -
mean_absolute_error: 0.0709 - val_loss: 0.0119 - val_mean_absolute_error:
0.0718

Epoch 70/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0117 -
mean_absolute_error: 0.0690 - val_loss: 0.0114 - val_mean_absolute_error:
0.0711

Epoch 71/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0117 -
mean_absolute_error: 0.0693 - val_loss: 0.0122 - val_mean_absolute_error:
0.0761

Epoch 72/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0117 -
mean_absolute_error: 0.0694 - val_loss: 0.0110 - val_mean_absolute_error:
0.0666

Epoch 73/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0120 -
mean_absolute_error: 0.0719 - val_loss: 0.0114 - val_mean_absolute_error:
0.0703

Epoch 74/100

69/69 [==============================] - 1s 7ms/step - loss: 0.0114 -
mean_absolute_error: 0.0679 - val_loss: 0.0112 - val_mean_absolute_error:
0.0691

Epoch 75/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0112 -
mean_absolute_error: 0.0668 - val_loss: 0.0116 - val_mean_absolute_error:
```

```
0.0713

Epoch 76/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0112 -
mean_absolute_error: 0.0673 - val_loss: 0.0110 - val_mean_absolute_error:
0.0667

Epoch 77/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0111 -
mean_absolute_error: 0.0662 - val_loss: 0.0109 - val_mean_absolute_error:
0.0678

Epoch 78/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0109 -
mean_absolute_error: 0.0646 - val_loss: 0.0108 - val_mean_absolute_error:
0.0674

Epoch 79/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0112 -
mean_absolute_error: 0.0674 - val_loss: 0.0107 - val_mean_absolute_error:
0.0655

Epoch 80/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0109 -
mean_absolute_error: 0.0647 - val_loss: 0.0105 - val_mean_absolute_error:
0.0639

Epoch 81/100

69/69 [==============================] - 1s 7ms/step - loss: 0.0108 -
mean_absolute_error: 0.0645 - val_loss: 0.0113 - val_mean_absolute_error:
0.0732

Epoch 82/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0111 -
mean_absolute_error: 0.0664 - val_loss: 0.0105 - val_mean_absolute_error:
0.0644

Epoch 83/100

69/69 [==============================] - 1s 8ms/step - loss: 0.0107 -
mean_absolute_error: 0.0634 - val_loss: 0.0103 - val_mean_absolute_error:
0.0630

Epoch 84/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0111 -
mean_absolute_error: 0.0662 - val_loss: 0.0104 - val_mean_absolute_error:
0.0641

Epoch 85/100

69/69 [==============================] - 1s 7ms/step - loss: 0.0107 -
mean_absolute_error: 0.0639 - val_loss: 0.0114 - val_mean_absolute_error:
0.0731

Epoch 86/100
```

```
69/69 [==============================] - 0s 7ms/step - loss: 0.0108 -
mean_absolute_error: 0.0648 - val_loss: 0.0104 - val_mean_absolute_error:
0.0648

Epoch 87/100

69/69 [==============================] - 1s 7ms/step - loss: 0.0107 -
mean_absolute_error: 0.0643 - val_loss: 0.0113 - val_mean_absolute_error:
0.0693

Epoch 88/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0109 -
mean_absolute_error: 0.0652 - val_loss: 0.0102 - val_mean_absolute_error:
0.0619

Epoch 89/100

69/69 [==============================] - 0s 7ms/step - loss: 0.0108 -
mean_absolute_error: 0.0635 - val_loss: 0.0102 - val_mean_absolute_error:
0.0625

Epoch 90/100

69/69 [==============================] - 1s 10ms/step - loss: 0.0106 -
mean_absolute_error: 0.0635 - val_loss: 0.0108 - val_mean_absolute_error:
0.0695

Epoch 91/100

69/69 [==============================] - 1s 10ms/step - loss: 0.0105 -
mean_absolute_error: 0.0631 - val_loss: 0.0103 - val_mean_absolute_error:
0.0631

Epoch 92/100

69/69 [==============================] - 1s 10ms/step - loss: 0.0110 -
mean_absolute_error: 0.0655 - val_loss: 0.0103 - val_mean_absolute_error:
0.0639

Epoch 93/100

69/69 [==============================] - 1s 11ms/step - loss: 0.0105 -
mean_absolute_error: 0.0624 - val_loss: 0.0100 - val_mean_absolute_error:
0.0618

Epoch 94/100

69/69 [==============================] - 1s 11ms/step - loss: 0.0105 -
mean_absolute_error: 0.0623 - val_loss: 0.0103 - val_mean_absolute_error:
0.0657

Epoch 95/100

69/69 [==============================] - 1s 11ms/step - loss: 0.0109 -
mean_absolute_error: 0.0657 - val_loss: 0.0109 - val_mean_absolute_error:
0.0663

Epoch 96/100

69/69 [==============================] - 1s 11ms/step - loss: 0.0108 -
mean_absolute_error: 0.0642 - val_loss: 0.0113 - val_mean_absolute_error:
```

```
0.0727

Epoch 97/100

69/69 [==============================] - 1s 10ms/step - loss: 0.0107 - mean_absolute_error: 0.0639 - val_loss: 0.0100 - val_mean_absolute_error: 0.0627

Epoch 98/100

69/69 [==============================] - 1s 11ms/step - loss: 0.0107 - mean_absolute_error: 0.0637 - val_loss: 0.0108 - val_mean_absolute_error: 0.0693

Epoch 99/100

69/69 [==============================] - 1s 11ms/step - loss: 0.0106 - mean_absolute_error: 0.0634 - val_loss: 0.0102 - val_mean_absolute_error: 0.0636

Epoch 100/100

69/69 [==============================] - 1s 10ms/step - loss: 0.0106 - mean_absolute_error: 0.0634 - val_loss: 0.0104 - val_mean_absolute_error: 0.0643
```

Out[ ]:

```
<keras.src.callbacks.History at 0x7a61e3ee4a90>
```

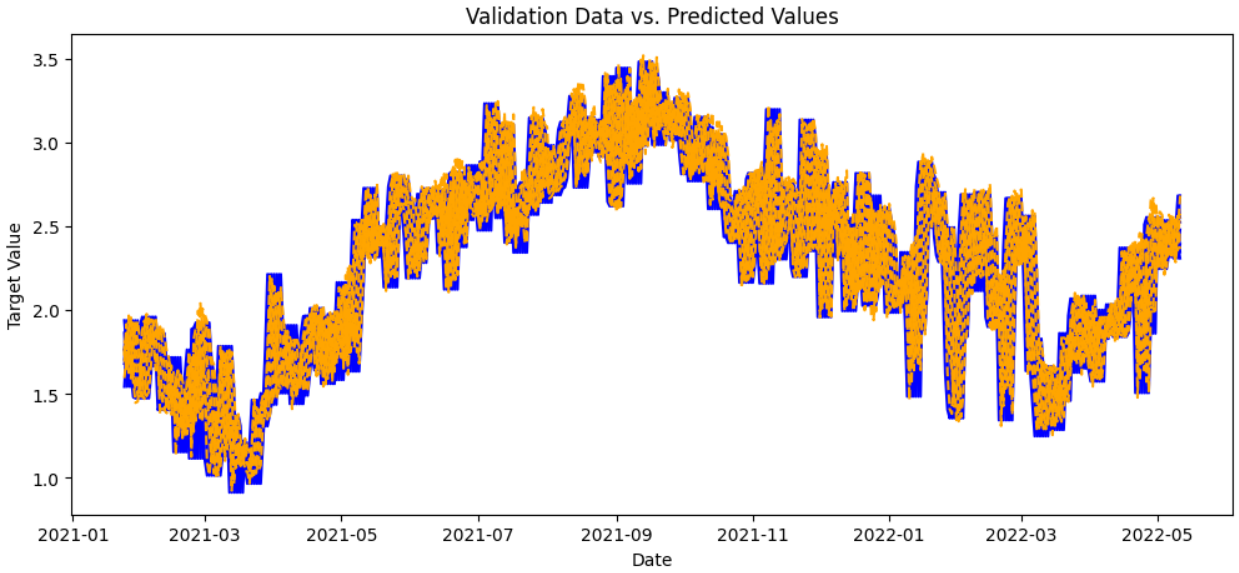
INSPECCIÓN VISUAL DEL AJUSTE

In [ ]:

```
# Generate predictions for the validation set

y_val_pred = model.predict(X_val)


# Create a plot

plt.figure(figsize=(12, 5))  # Adjust the figure size if needed


# Plot actual validation data

plt.plot(dates_val, y_val, label='Validation (Actual)', linestyle='-', color='blue')


# Plot predicted values for validation data

plt.plot(dates_val, y_val_pred, label='Validation (Predicted)', linestyle='--', color='orange')


# Customize the plot

plt.xlabel('Date')
```

```
plt.ylabel('Target Value')
plt.title('Validation Data vs. Predicted Values')

# Show the plot
plt.show()
```

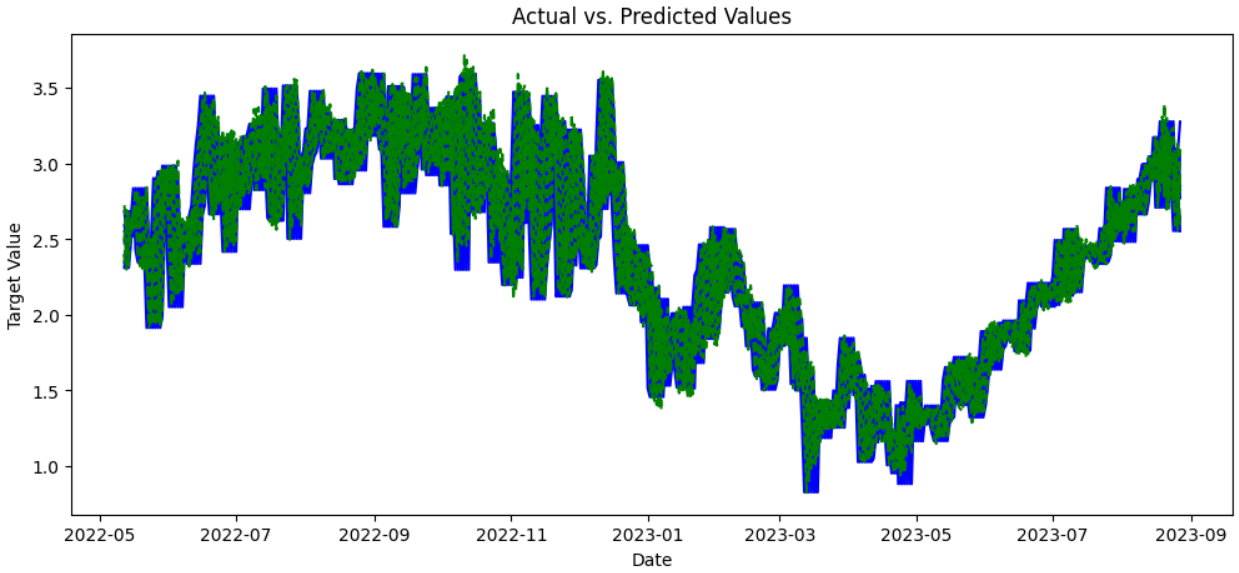
```
15/15 [==============================] - 1s 3ms/step
```



In [ ]:

```
# Assuming you've already trained your model as shown in your provided code snippet

# Generate predictions for the test set
y_pred = model.predict(X_test)

# Create a plot
plt.figure(figsize=(12, 5))  # Adjust the figure size if needed

# Plot actual test data
plt.plot(dates_test, y_test, label='Actual', linestyle='-', color='blue')

# Plot predicted values
plt.plot(dates_test, y_pred, label='Predicted', linestyle='--', color='green')

# Customize the plot
plt.xlabel('Date')
```

```
plt.ylabel('Target Value')
plt.title('Actual vs. Predicted Values')

# Show the plot
plt.show()
15/15 [==============================] - 0s 4ms/step
```



ESTADISTICOS DE AJUSTE

In [ ]:

```
import numpy as np

def calculate_metrics(y_true, y_pred):
    """
    Calcula los estadísticos de error: MSE, RMSE y MASE.

    Args:
        y_true (array-like): Valores reales.
        y_pred (array-like): Valores predichos.

    Returns:
        dict: Un diccionario con los valores de los estadísticos.
    """
    # Error Cuadrático Medio (MSE)
    mse = np.mean((y_true - y_pred) ** 2)
```

```
    # Raíz del Error Cuadrático Medio (RMSE)
    rmse = np.sqrt(mse)

    # Error Absoluto Medio Escalado (MASE)
    mae = np.mean(np.abs(y_true - y_pred))
    mase = mae / np.mean(np.abs(np.diff(y_true))) if
np.mean(np.abs(np.diff(y_true))) != 0 else np.nan

    # Crear un diccionario con los resultados
    metrics = {
        'MSE': mse,
        'RMSE': rmse,
        'MASE': mase
    }

    return metrics

# Generar predicciones para los conjuntos de validación y prueba
y_val_pred = model.predict(X_val)
y_test_pred = model.predict(X_test)

# Calcular los estadísticos de error
metrics_val = calculate_metrics(y_val, y_val_pred)
metrics_test = calculate_metrics(y_test, y_test_pred)

print("Estadísticos de error para el conjunto de validación:")
for metric, value in metrics_val.items():
    print(f"{metric}: {value:.4f}")

print("\nEstadísticos de error para el conjunto de prueba:")
for metric, value in metrics_test.items():
    print(f"{metric}: {value:.4f}")
15/15 [==============================] - 0s 3ms/step
15/15 [==============================] - 0s 4ms/step
Estadísticos de error para el conjunto de validación:
```

```
MSE: 0.0104
RMSE: 0.1021
MASE: 0.3442

Estadísticos de error para el conjunto de prueba:
MSE: 0.0111
RMSE: 0.1055
MASE: 0.3551
```

PREDICCIÓN

In [ ]:

```
# Function to make predictions for the next n days
def predict_next_days(model, initial_input, days_to_predict=90):
    predictions = []
    current_input = initial_input

    # Number of complete weeks to predict
    num_weeks = days_to_predict // 7

    for _ in range(num_weeks):
        prediction = model.predict(current_input[np.newaxis, :, np.newaxis])
        predictions.append(prediction.flatten())
        current_input = np.concatenate((current_input.flatten()[7:], prediction.flatten()))

    # Handle any remaining days if days_to_predict is not a multiple of 7
    remaining_days = days_to_predict % 7
    if remaining_days > 0:
        prediction = model.predict(current_input[np.newaxis, :, np.newaxis])
        predictions.append(prediction.flatten()[:remaining_days])

    return np.concatenate(predictions)

# Assuming df is your original DataFrame and it has a 'Date' column and
```

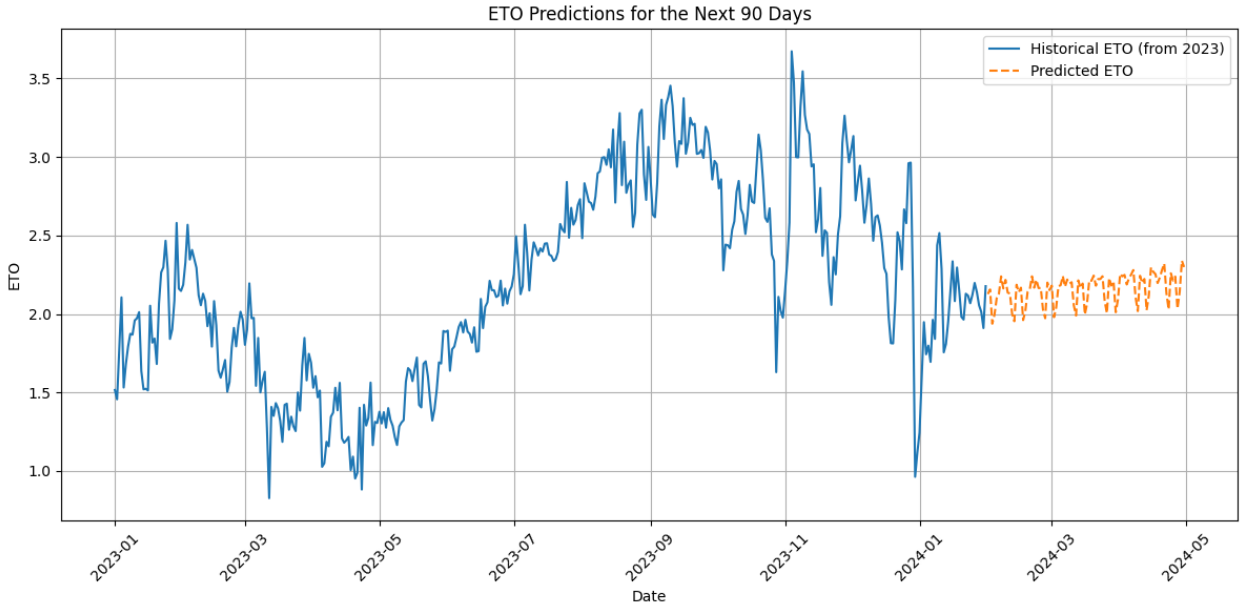
```
'ETO' values
last_window = df['ETO'].values[-7:]

# Predict the next 90 days
next_90_days_predictions = predict_next_days(model, last_window, days_to_predict=90)

# Create a DataFrame for the predictions
last_date = df.index[-1]
prediction_dates = [last_date + datetime.timedelta(days=i+1) for i in range(90)]

# Ensure the lengths match
assert len(prediction_dates) == len(next_90_days_predictions), "Lengths of dates and predictions do not match!"

# Create the DataFrame
predictions_df = pd.DataFrame({'Date': prediction_dates, 'Predicted_ETO': next_90_days_predictions})

print(predictions_df)
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 66ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 39ms/step
         Date  Predicted_ETO
0  2024-02-01       2.126815
1  2024-02-02       2.156197
```

```
2  2024-02-03       1.938094
3  2024-02-04       1.995584
4  2024-02-05       2.100743
..        ...            ...
85 2024-04-26       2.238451
86 2024-04-27       2.034199
87 2024-04-28       2.140674
88 2024-04-29       2.338817
89 2024-04-30       2.299124

[90 rows x 2 columns]
```

In [ ]:

```
# Filter the historical data to include only entries from 2023 onwards
historical_data_from_2023 = df[df.index >= '2023-01-01']

# Combine the filtered historical data and predicted data for plotting
combined_df = pd.concat([historical_data_from_2023[['ETO']],
predictions_df.set_index('Date')], axis=0)

# Plot the historical and predicted ETO values
plt.figure(figsize=(12, 6))
plt.plot(historical_data_from_2023.index,
historical_data_from_2023['ETO'], label='Historical ETO (from 2023)')
plt.plot(predictions_df['Date'], predictions_df['Predicted_ETO'],
label='Predicted ETO', linestyle='--')

# Add titles and labels
plt.title('ETO Predictions for the Next 90 Days')
plt.xlabel('Date')
plt.ylabel('ETO')
plt.legend()
plt.grid(True)

# Rotate date labels for better readability
plt.xticks(rotation=45)
```

```
# Show the plot
plt.tight_layout()
plt.show()
```



In [ ]:

```
combined_df.to_excel('lstm_secuencial.xlsx')
```