UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

"MODELO DE MACHINE LEARNING PARA EL PRONÓSTICO DE PRECIPITACIONES EN LA ESTACIÓN JORGE BASADRE DE LA REGIÓN TACNA"

PARA OPTAR:

TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS

PRESENTADO POR:

Bach. YOBER NAIN CATARI CABRERA

Bach. NILTON EDY PEREZ MAMANI

TACNA – PERÚ

2026

UNIVERSIDAD PRIVADA DE TACNA
FACULTAD DE INGENIERÍA
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

"MODELO DE MACHINE LEARNING PARA EL PRONÓSTICO DE PRECIPITACIONES EN LA ESTACIÓN JORGE BASADRE DE LA REGIÓN TACNA"

Tesis sustentada y aprobada el 01 de abril de 2026; estando el jurado calificador integrado por:

PRESIDENTE : M Sc. HAYDEE RAQUEL SISA YATACO

SECRETARIO : Mtro. ALBERTO JOHNATAN FLOR RODRÍGUEZ

VOCAL : Dr. LUIS ALFREDO FERNANDEZ VIZCARRA

ASESOR : Mtro. ISRAEL NAZARETH CHAPARRO CRUZ

DECLARACIÓN JURADA DE ORIGINALIDAD

Nosotros, Yober Nain Catari Cabrera y Nilton Edy Perez Mamani, egresados, de la Escuela Profesional de Ingeniería de Sistemas de la Facultad de Ingeniería de la Universidad Privada de Tacna, identificados con DNI 72567658 y 70938731 respectivamente, así como Israel Nazareth Chaparro Cruz con DNI 48584646; declaramos en calidad de autores y asesor que:

1. Somos los autores de la tesis titulado: *Modelo de Machine Learning para el Pronóstico de Precipitaciones en la Estación Jorge Basadre de la Región Tacna*, la cual presentamos para optar el Título de Ingeniero de Sistemas
2. La tesis es completamente original y no ha sido objeto de plagio, total ni parcialmente, habiéndose respetado rigurosamente las normas de citación y referencias para todas las fuentes consultadas.
3. Los datos presentados en los resultados son auténticos y no han sido objeto de manipulación, duplicación ni copia.

En virtud de lo expuesto, asumimos frente a *La Universidad* toda responsabilidad que pudiera derivarse de la autoría, originalidad y veracidad del contenido de la tesis, así como por los derechos asociados a la obra.

En consecuencia, nos comprometemos ante a *La Universidad* y terceros a asumir cualquier perjuicio que pueda surgir como resultado del incumplimiento de lo aquí declarado, o que pudiera ser atribuido al contenido de la tesis, incluyendo cualquier obligación económica que debiera ser satisfecha a favor de terceros debido a acciones legales, reclamos o disputas resultantes del incumplimiento de esta declaración.

En caso de descubrirse fraude, piratería, plagio, falsificación o la existencia de una publicación previa de la obra, aceptamos todas las consecuencias y sanciones que puedan derivarse de nuestras acciones, acatando plenamente la normatividad vigente.

Tacna, 02 de setiembre de 2025

Yober Nain Catari Cabrera
DNI: 72567658

Nilton Edy Perez Mamani
DNI :70938731

Israel Nazareth Chaparro Cruz
DNI: 48584646

DEDICATORIA

Este trabajo es para mis padres, mis queridos hermanos y mi familia; las personas más significativas en mi vida. Me ayudan a alcanzar mis metas día a día; siempre están ahí para motivarme y apoyarme, no importa qué situación se me presente. Nada de esto sería posible sin mis padres Enrique y Porfiria Rosa, y mis hermanos y hermanas Horlando, Beto y Lucy Yovana. ¡Mucho amor, gracias por estar allí para mí!

Yober Nain Catari Cabrera

DEDICATORIA

Este trabajo va dedicado a mis queridos padres Valentín y Lourdes, mi hermano William y a mi amada esposa Fabiola, quienes siempre han estado encargados de proporcionarme cariño y acompañamiento en todas las etapas de mi existencia. A mis maestros por su constante esfuerzo y guía en mi educación. A mis amigos, por estar a mi lado en los momentos más difíciles y compartir su amistad conmigo. A mi familia, por creer en mí y darme fuerzas para seguir adelante. Gracias a todos por su apoyo.

Nilton Edy Perez Mamani

AGREDECIMIENTO

Primero, quiero dar las gracias a Dios y a mis padres, quienes siempre me han apoyado sin condiciones para que pueda alcanzar todas mis metas personales y académicas. Estoy muy agradecido con mi asesor por su dedicación y paciencia. Sin las palabras y correcciones adecuadas, no habría podido terminar esta tesis tan esperada. Para terminar, quiero agradecer a la universidad por los constantes desafíos que me ha presentado y por darme la oportunidad de obtener el título que siempre he deseado.

Yober Nain Catari Cabrera

AGREDECIMIENTO

Primeramente, ante todo, quiero agradecer a Dios por ser mi luz y mi guía a lo largo de mi vida. Por darme la fuerza y el conocimiento para superar los desafíos y alcanzar mis metas. Por otra parte, estoy agradecido por mi familia, amigos y maestros que me brindaron el apoyo necesario para recorrer el camino elegido. Nunca olvidaré su amor incondicional y sus continuas presencias entre mis días; esta tesis es mi muestra de agradecimiento hacia ustedes. Por último, pero no menos importante, estoy increíblemente agradecido con mi familia y seres queridos por su inquebrantable apoyo emocional y comprensión en los momentos de arduo trabajo. Sus palabras motivadoras y su amor único me dieron fuerzas para seguir adelante e incluso en los momentos más desafiantes. Gracias por estar allí.

.

Nilton Edy Perez Mamani

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN

En el presente estudio, se creó un modelo de machine learning para poder hacer pronóstico meteorológico sobre la precipitación en la estación de Jorge Basadre en la región de Tacna. Este trabajo se centra en la predicción de precipitaciones en la estación meteorológica, empleando modelos de machine learning aplicados a series temporales climáticas. La investigación responde a la necesidad de contar con pronósticos más precisos que permitan anticipar eventos de lluvia y así reducir riesgos, optimizar la gestión de recursos hídricos y apoyar la planificación agrícola. Se utilizaron datos históricos comprendidos entre 2021 y 2025, que incluyen registros de temperatura, humedad relativa, presión atmosférica y precipitaciones. Como punto de referencia, se implementaron modelos estadísticos tradicionales como ARIMA y Prophet para establecer una base comparativa inicial. La metodología siguió el enfoque CRISP-DM, que abarcó desde la comprensión del problema y el análisis exploratorio de datos hasta la modelación, evaluación y despliegue. El preprocesamiento incluyó limpieza de valores nulos y atípicos, normalización de variables y generación de secuencias temporales. Se seleccionaron características relevantes para reducir la complejidad y se entrenaron modelos basados en redes neuronales recurrentes LSTM y GRU. La optimización de hiperparámetros, tamaño de ventana, unidades recurrentes, tamaño de lote y tasa de aprendizaje se realizó con el fin de maximizar el rendimiento. La evaluación se llevó a cabo mediante métricas como MAE, MSE, RMSE y MAPE, empleando además validación cruzada K-Fold (TimeSeriesSplit), para garantizar la robustez de los resultados. Los hallazgos mostraron que el modelo GRU superó a LSTM en precisión, obteniendo menores valores de error y demostrando una mayor capacidad para capturar dependencias temporales y patrones no lineales propios de la variabilidad climática. No obstante, ambos modelos superaron ampliamente a las aproximaciones estadísticas tradicionales. En conclusión, este estudio confirma la efectividad de las redes neuronales recurrentes, en especial GRU, como herramienta para el pronóstico de precipitaciones, ofreciendo un marco metodológico sólido y aplicable a la gestión climática, la prevención de desastres y el desarrollo sostenible de la región de Tacna; asimismo, se recomienda que futuras investigaciones incorporen datos meteorológicos de al menos diez años, con el fin de mejorar la precisión y confiabilidad de los modelos predictivos.

Palabras claves: Machine Learning; redes neuronales; data; algoritmos; modelo.

ABSTRACT

In this study, a machine learning model was created to forecast precipitation at the Jorge Basadre weather station in the Tacna region. This work focuses on predicting precipitation at the meteorological station using machine learning models applied to climate time series. The research addresses the need for more accurate forecasts that allow for anticipating rainfall events, thereby reducing risks, optimizing water resource management, and supporting agricultural planning. Historical data from 2021 to 2025 were used, including records of temperature, relative humidity, atmospheric pressure, and precipitation. As a reference point, traditional statistical models such as ARIMA and Prophet were implemented to establish an initial comparative baseline. The methodology followed the CRISP-DM approach, encompassing everything from problem understanding and exploratory data analysis to modeling, evaluation, and deployment. Preprocessing included cleaning null and outlier values, normalizing variables, and generating time series. Relevant features were selected to reduce complexity, and models based on LSTM and GRU recurrent neural networks were trained. Hyperparameter optimization, window size, recurrent units, batch size, and learning rate were performed to maximize performance. Evaluation was carried out using metrics such as MAE, MSE, RMSE, and MAPE, and K-Fold (TimeSeriesSplit) cross-validation was employed to ensure the robustness of the results. The findings showed that the GRU model outperformed LSTM in accuracy, obtaining lower error values and demonstrating a greater capacity to capture temporal dependencies and nonlinear patterns characteristic of climate variability. However, both models significantly outperformed traditional statistical approaches. In conclusion, this study confirms the effectiveness of recurrent neural networks, especially GRU, as a tool for precipitation forecasting, offering a solid methodological framework applicable to climate management, disaster prevention, and sustainable development in the Tacna region. Furthermore, it is recommended that future research incorporate meteorological data of at least ten years, in order to improve the accuracy and reliability of predictive models.

Keywords: Machine Learning; neural networks; data; algorithms; model

INTRODUCCIÓN

En la actualidad, el departamento de Tacna, situado en el extremo sur del Perú y en la zona inicial del desierto de Atacama, presenta una marcada escasez de recursos hídricos. No obstante, en determinados años se han registrado precipitaciones intensas que ocasionaron impactos significativos en poblaciones asentadas en cauces de quebradas y ríos de carácter temporal. Un ejemplo histórico relevante ocurrió en 1927, cuando, tras las severas inundaciones que afectaron al norte del país debido al fenómeno El Niño de 1925, se produjeron lluvias torrenciales en Tacna, incrementando el caudal de los ríos Caplina y Caramolle y activando quebradas como Del Diablo, lo que generó daños en la ciudad de Tacna y en la localidad de Mirave, ubicada sobre el cono aluvial de la quebrada homónima. Cabe señalar que los episodios de precipitaciones extremas que han afectado a Tacna, Mirave y otros centros poblados no se relacionan exclusivamente con eventos extraordinarios de El Niño, sino que responden principalmente a lluvias estacionales u otros fenómenos climatológicos asociados a la variabilidad climática propia de la región sur (IGP, 2021).

En este contexto, el presente estudio se centra en la predicción de precipitaciones en la estación meteorológica Jorge Basadre, ubicada en la región de Tacna. La población de estudio corresponde a los registros proporcionados por el SENAMHI, comprendidos entre el 1 de enero de 2021 y el 31 de marzo de 2025. Para abordar el problema, se propusieron e implementaron cuatro modelos de machine learning y de series temporales: LSTM, GRU, ARIMA y Prophet, seleccionados por su eficacia en pronósticos meteorológicos.

El proyecto está estructurado en cinco capítulos, que ayudarán, paso a paso, en el análisis y comprensión del problema.

En el capítulo I, se aborda el problema de investigación, el cual incluye una descripción de este, la materia de problema formulado, su justificación e importancia y, finalmente, el cumplimiento de los objetivos generales y de los específicos que están, en su mayor parte, bien delineados. Además, se incorpora hipótesis generales y específicas que están bien formuladas y sustentadas.

En el capítulo II, se elabora el marco teórico, en el cual se analizan con rigor los antecedentes de investigación y las teorías que fundamentan el proyecto, ofreciendo el marco teórico con los antecedentes y las teorías que fundamentan el proyecto. También

se da una definición de los términos que se utilizarán, así como las palabras y su significado, lo que da soporte conceptual para la tesis elaborada.

En el capítulo III, El marco metodológico, se encuentra el diseño del estudio, las actividades que se llevarán a cabo, los materiales y los instrumentos que fueron seleccionados con precisión. Límite geográficamente. También se determina la población y/o la muestra de estudio que se elige, se operacionalizan las variables y se describen las técnicas de procesamiento y análisis; las técnicas de análisis son elaboradas en un marco metodológico sólido que se constituye en pilar del desarrollo y la validación de la tesis.

En los capítulos IV y V, en estas secciones, explicaré los hallazgos y su discusión, con especial atención a la exposición, las fases de análisis e interpretación en relación con los datos recopilados, la validación de las hipótesis planteadas y, posteriormente, la discusión de los hallazgos más importantes. Por último, argumentaré las conclusiones de la investigación, las recomendaciones relevantes, las fuentes bibliográficas consultadas y los documentos complementarios que añaden valor al trabajo.

CAPÍTULO I: EL PROBLEMA DE INVESTIGACIÓN

1.1. Descripción del problema

Según Taguela (2025), analiza cómo varía la precipitación en el espacio y el tiempo en África, un continente que depende en gran medida de este recurso para múltiples actividades económicas y sociales. A partir de datos observacionales y simulaciones del Proyecto de Intercomparación de Modelos Acoplados Fase 6 (CMIP6), los resultados evidencian patrones de precipitación distintos entre las subregiones africanas. En África Occidental, las lluvias más intensas coinciden con la temporada del monzón de junio a septiembre, mientras que en África Central las precipitaciones son constantes durante todo el año. En África Oriental, el norte experimenta una estación lluviosa entre mayo y septiembre, a diferencia del sur, que atraviesa una estación seca en ese mismo periodo. Por otro lado, el sur de África presenta su mayor volumen de lluvias durante el verano austral. Los modelos climáticos de CMIP6 y su promedio en conjunto logran representar en términos generales los picos estacionales de precipitación, aunque presentan desviaciones significativas. Por ejemplo, tienden a subestimar la intensidad de las lluvias en el norte de África y a sobreestimarlas en el África subsahariana. Asimismo, estos modelos muestran limitaciones al replicar la variabilidad espacial observada, el comienzo y final de las temporadas lluviosas y ciertos indicadores de eventos extremos de precipitación. En conjunto, este análisis destaca la necesidad de mejorar los modelos climáticos para lograr proyecciones más fiables, que sirvan como base sólida para la formulación de políticas y la gestión adecuada de los recursos en los diversos contextos geográficos y económicos del continente africano.

Según OMM, (2025) advierte que, durante los próximos cinco años, las temperaturas globales se mantendrán en niveles récord o muy cercanos a ellos, intensificando los riesgos climáticos y sus efectos sobre las sociedades, las economías y el desarrollo sostenible. Las proyecciones indican una probabilidad del 80 % de que al menos uno de estos años supere a 2024 como el más cálido registrado, así como un 86 % de probabilidad de que se exceda temporalmente el umbral de 1,5 °C respecto al nivel preindustrial (1850–1900). Asimismo, existe un 70 % de probabilidad de que la temperatura media global del periodo 2025–2029 supere los 1,5 °C, aunque el calentamiento sostenido a largo plazo aún se mantiene por debajo de dicho umbral.

Según Ohba (2021), las variaciones futuras en las precipitaciones tienen consecuencias sociales significativas, ya que se espera que influyan profundamente en la gestión del riesgo de desastres y en el entorno natural, tanto a nivel global como

regional. Los efectos del cambio climático sobre las precipitaciones pueden clasificarse en transformaciones en su forma y en su cantidad o frecuencia. Los cambios en la forma son más probables en zonas de latitudes medias y altas, donde las temperaturas superficiales cercanas a los 0 °C afectan la transición entre lluvia y nieve. Además, se estima que la intensidad de los eventos extremos de precipitación aumentará en gran parte del planeta como resultado del cambio climático. Aunque persisten grandes incertidumbres y limitaciones en la resolución de las proyecciones a escala regional, los modelos climáticos pueden ser herramientas útiles para explorar posibles estrategias de adaptación. Para entender mejor estas futuras alteraciones en las precipitaciones, se propone utilizar la clasificación de patrones climáticos mediante mapas autoorganizados, una técnica que permite analizar de manera más clara los efectos del calentamiento global según diferentes tipos de eventos meteorológicos. Este método también ayuda a diferenciar los impactos del cambio climático según componentes dinámicos y termodinámicos, mejorando así la comprensión general de su consecuencia.

Según Urbina y Takahashi (2023), el Perú posee un territorio diverso que abarca la Amazonía, la cordillera de los Andes y una costa árida. Este territorio se ve afectado por una variedad de fenómenos meteorológicos y climáticos, entre los cuales destacan El Niño y fenómenos subestacionales tropicales, como la oscilación Madden-Julian en la atmósfera y las ondas Kelvin ecuatoriales oceánicas, que operan en escalas temporales de varias semanas. La implantación de servicios climáticos puede reportar importantes beneficios sociales y económicos, que son posibles gracias a la complejidad del clima. Los servicios de modelización hidrológica y agrometeorológica son especialmente importantes, ya que necesitan previsiones con un plazo de antelación de semanas. Esta garantía es esencial para maximizar el potencial de la oportunidad.

De acuerdo con los lineamientos de los ODS, el cambio climático representa una amenaza global que afectará, de una u otra forma, a todas las personas del planeta. Su avance, impulsado principalmente por actividades humanas y por el aumento continuo de los gases de efecto invernadero, está ocurriendo a un ritmo mucho más rápido de lo previsto. Esto incrementa la probabilidad de enfrentar eventos extremos, como sequías, inundaciones, olas de calor y el aumento del nivel del mar, poniendo en riesgo la seguridad, el desarrollo y la estabilidad de numerosos países. De no tomarse medidas contundentes, los progresos alcanzados en las últimas décadas podrían revertirse y podrían desencadenarse procesos masivos de migración y conflictos sociales. Para limitar el calentamiento global a 1,5 °C sobre los niveles preindustriales, las emisiones

ya deberían estar disminuyendo, pero las tendencias actuales muestran que este objetivo aún está lejos de cumplirse. Por ello, se requieren acciones urgentes y transformadoras que incluyan a sectores completos de la economía y permitan avanzar hacia sociedades resilientes y con emisiones netas cero. El tiempo para actuar es cada vez más reducido, y las decisiones que se tomen hoy serán determinantes para asegurar un futuro sostenible para las próximas generaciones. En el contexto de Perú, la predicción y gestión de las precipitaciones son aspectos fundamentales que pueden estar relacionados con varios ODS, dependiendo de cómo se gestionen y de los impactos que generen. Algunos de los ODS que están directamente relacionados incluyen: Agua limpia y saneamiento, Acción por el clima, Ciudades y comunidades sostenibles, Hambre cero y Vida de ecosistemas terrestres (ONU, 2018).

Según SENAMHI (2024), el organismo público que menciona está vinculado al Ministerio del Ambiente y su misión es ofrecer información y conocimiento sobre meteorología, hidrología y clima de forma confiable, oportuna y accesible para el beneficio de la sociedad peruana. A través de su portal, se puede acceder a datos hidrometeorológicos abiertos a nivel nacional, los cuales se pueden descargar directamente desde su sitio web. Estos datos son esenciales para diversos estudios y aplicaciones en campos como la climatología, la agricultura y la gestión de recursos hídricos.

Para los años 2020 y 2030, no se prevén cambios significativos en la distribución geográfica de las precipitaciones, las cuales continúan estrechamente vinculadas a los patrones climáticos habituales. Para el año 2030, se proyecta una disminución de las lluvias anuales, principalmente en la región andina, con reducciones entre el 10 % y 20 %, y en la selva alta del norte y centro, con descensos de hasta un 10%. En contraste, se anticipan aumentos en la costa norte y la selva sur, con incrementos que oscilarían entre y el 10 % y 20 %.

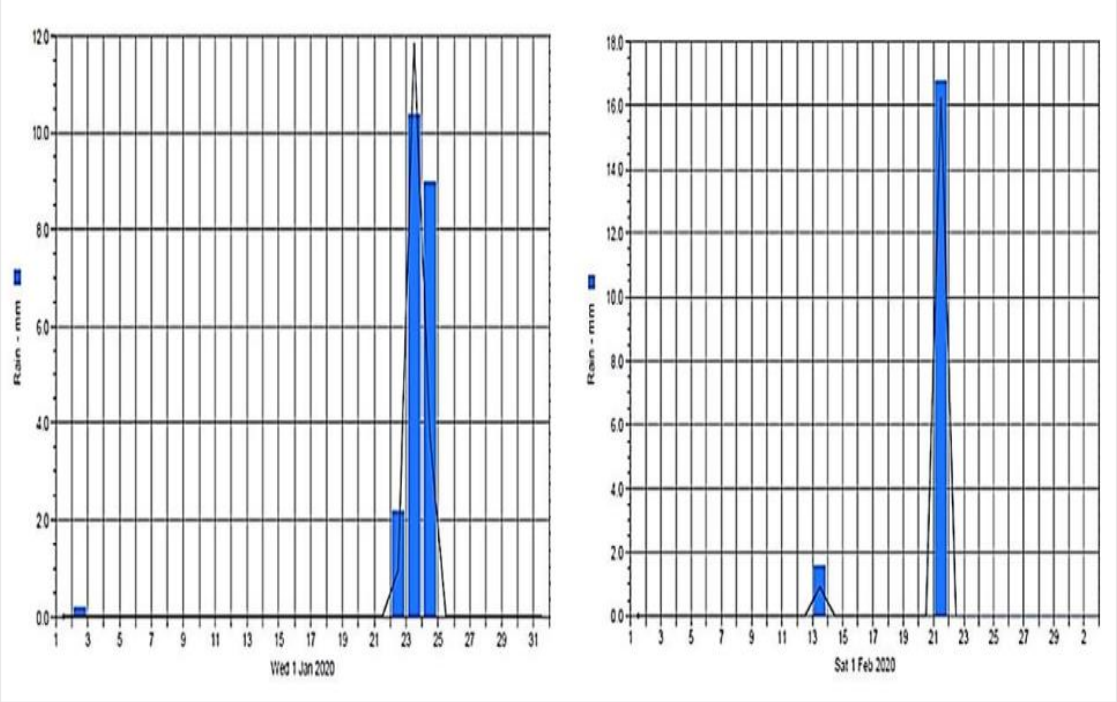
En cuanto a las estaciones, se esperan variaciones en el comportamiento de las lluvias. En verano, se registrarían déficits importantes en la mayor parte del país, mientras que en otoño las precipitaciones superarían los valores normales. Durante el invierno y la primavera, se presentarían tanto aumentos como disminuciones en la distribución espacial, con variaciones que van desde -30 % hasta +20 % respecto a los promedios.

Respecto a las precipitaciones máximas proyectadas para 2030, la tendencia general apunta a una disminución en la mayoría del territorio nacional, con aumentos localizados en algunas zonas específicas en comparación con los niveles actuales.

Pino y Chávarri, (2022) durante el verano de 2020, en la ciudad de Tacna ubicada en el sur del Perú, en la frontera con Chile, y correspondiente a la cabecera del desierto de Atacama, se presentaron dos importantes episodios de precipitaciones, según reportes del SENAMHI y los registros de la estación meteorológica automática situada en el campus de la Universidad Nacional Jorge Basadre Grohmann. Estos eventos alcanzaron intensidades máximas de 10,4 mm/h el 23 de enero y 16,8 mm/h el 21 de febrero. En la Figura 1 se puede observar cómo, la consecuencia, se produjeron flujos de tipo aluvional en las quebradas Caramolle y El Diablo, siendo este último escenario de un aluvión que causó la muerte de tres personas. El evento del 21 de febrero fue el más grave, provocando además considerables daños económicos en la ciudad, en la Figura 2.

Figura 1

Precipitación registrada el 23 de enero y 21 de febrero de 2020 en Tacna.



Nota. El grafico de Precipitaciones Registrada el 23 de enero y 21 de febrero del 2020 en la ciudad Tacna.

Figura 2

Diario Gestión, 22 febrero 2020, informa: huaicos en Tacna

Nota. El grafico presentadas de las Evidencias de Precipitaciones en Tacna.

El clima del valle de Tacna no es uniforme, ya que dos tercios de su territorio corresponden a la franja costera y un tercio se sitúa en las alturas de la cordillera. De acuerdo con 23 años de observaciones, desde 1931 hasta 1954, la temperatura media es de 16,54 ºC. Las temperaturas más bajas se registran en julio y agosto, mientras que las más altas se presentan en enero y febrero, oscilando entre 19,5 ºC y 22 ºC.

Durante los meses de invierno, las neblinas son frecuentes tanto en los valles como en las pampas del Alto de la Alianza. En cuanto a las lluvias, son muy escasas y varían gradualmente de año en año. Tanto las lluvias como las neblinas favorecen el desarrollo de la vegetación en las lomas (ENDES*, 2024*).

El centro poblado de Mirave ha sido afectado en varias ocasiones por inundaciones ocasionadas por la activación de quebradas. En el año 1927, se trataba

de una localidad muy pequeña, con unas pocas viviendas situadas en la parte baja del cono aluvial de la quebrada que lleva el mismo nombre. Se cuenta que este asentamiento desapareció debido a un aluvión generado por fuertes lluvias. A pesar de ello, sus habitantes regresaron al mismo lugar y, con el tiempo, la población se expandió a lo largo del cauce de la quebrada. Los años en los que la ciudad de Tacna, Mirave y otras localidades sufrieron el impacto de intensas lluvias no están necesariamente relacionados con un evento extraordinario del fenómeno El Niño. Más bien, estas precipitaciones parecen estar vinculadas a lluvias estacionales u otros fenómenos climáticos propios de la variabilidad del clima en la región sur (IGP, 2021).

En la presente investigación, se toma como punto de enfoque la estación meteorológica Jorge Basadre, ubicada en Tacna, la cual cumple un rol fundamental en el monitoreo de las condiciones climáticas locales y en la recopilación de datos meteorológicos. No obstante, se identifica una problemática específica relacionada con la calidad y continuidad de sus registros. Los datos disponibles presentan brechas importantes, evidenciándose información incompleta; por ejemplo, durante el año de la pandemia no se cuenta con registros diarios continuos. Esta situación constituye una limitación significativa para la realización de análisis que requieren información reciente, consistente y de alta calidad. En consecuencia, esta discontinuidad en los datos puede afectar negativamente la precisión y confiabilidad de los estudios, especialmente aquellos que dependen de información en tiempo real o de series temporales completas para la generación de resultados.

La falta de información precisamente pronosticada podría tener consecuencias dañinas; en febrero de 2020, fuertes lluvias y huaicos en Tacna dejaron al menos tres muertes, 250 viviendas afectadas y daños en diversas infraestructuras (RPP, 2020).

El objetivo de esta investigación es mejorar la precisión de las estimaciones de precipitación en la estación meteorológica Jorge Basadre a través de diversas técnicas de inteligencia cognitiva. A través de un modelo de machine learning alimentado con grandes conjuntos de datos históricos, proporcionaremos estimaciones precisas de la precipitación en distintos lapsos para el futuro de la lluvia: corto, mediano y largo plazo. Esta eficiencia adicional permitirá optimizar el uso de recursos hídricos en toda la región de Tacna.

En esta relación, la investigación se propone superar la superficialidad de los análisis univariantes con la caracterización precisa del fenómeno. La variable dependiente se define de este modo como Pronóstico de Precipitaciones (mm), cuyo comportamiento será estimado con relación a las condiciones meteorológicas locales.

Las variables predictoras (independientes) serán marcadas como Temperatura Máxima (°C), Temperatura Mínima (°C) y Humedad Relativa (%), eligiendo por su influencia termodinámica directa en la formación de lluvias. Reuniendo los análisis por estos factores, permitirá detectar los patrones climáticos que manejan la variabilidad en la estación Jorge Basadre, lo cual es información esencial para la gestión de riesgos. Por lo tanto, nuestra pregunta de investigación es:

1.2. Formulación del problema

1.2.1. Problema general

¿Cuál es el modelo predictivo de machine learning que ofrece mayor exactitud en el pronóstico de precipitaciones en la estación Jorge Basadre de la Región Tacna, utilizando como variables predictoras la Fecha, Temperatura Máxima, Temperatura Mínima, Humedad Relativa y Precipitación Histórica?

1.2.2. Problemas específicos

a. ¿Cuál es el conjunto de datos para el pronóstico de precipitaciones en la estación Jorge Basadre de la Región Tacna-Perú?

b. ¿Cómo entrenar modelos de machine learning para el pronóstico de precipitaciones en la estación Jorge Basadre de la Región Tacna - Perú?

c. ¿Cuál es la exactitud de los modelos de machine learning entrenados en el pronóstico de precipitaciones en la estación Jorge Basadre de la Región Tacna - Perú?

1.3. Justificación e importancia

1.3.1. Desde el punto económico

Tener un pronóstico confiable de las lluvias es clave para el desarrollo económico sostenible en regiones con climas complicados, como Tacna, Perú. Así que implementar un modelo de machine learning que pueda predecir las lluvias de la estación Jorge Basadre es una idea innovadora que podría mejorar la gestión de recursos hídricos, la agricultura y otras actividades económicas. Esta tecnología hace que la planificación agrícola sea más fácil, ya que permite a los agricultores prepararse y adaptarse mejor a las condiciones climáticas, además de reducir los riesgos que vienen con un clima adverso.

Por otra parte, al ofrecer a las autoridades datos precisos y a tiempo, el modelo también apoya la toma de decisiones en áreas como la logística y el turismo, lo que contribuye a un crecimiento económico más sólido y duradero. En el ensayo, se analiza cómo el modelo de machine learning puede beneficiar la economía de Tacna y se discuten las ventajas de esta tecnología en la gestión del agua, la prevención de desastres y la planificación estratégica en diversas industrias del sur de Perú.

1.3.2. Desde el punto social

Incrementar la precisión en las previsiones climáticas es una elección lógica porque afecta directamente al bienestar de nuestra propia gente. Mayor recopilación de datos climatológicos significa que la gente se prepara mejor para la dura tarea del tiempo extremo y, por lo tanto, los riesgos pueden reducirse. Cambiar la comprensión del clima implica un avance en previsiones, lo que significa que las personas y grupos de personas pueden tomar decisiones más seguras, o los agricultores pueden estar más seguros antes de plantar o cosechar.

No solo es que fortalece la seguridad alimentaria, también consolida la estabilidad económica local. Además, administrar el agua de manera más eficiente garantiza que el acceso al agua sea mucho más equitativo con todos los demás usos comunitarios.

1.3.3. Desde el punto tecnológico

El pronóstico del clima mejora mucho con el uso de la tecnología de machine learning. La mayor ventaja de un modelo es que puede pronosticar el clima en detalle y con la mayor anticipación. Por lo tanto, al analizar muchos datos en gran detalle, la tecnología cambia la forma en que vemos y predecimos el clima. Como resultado, las autoridades y la ciudadanía toman decisiones más informadas y precisas cuando se trata de elegir cómo actuar frente al clima todo el tiempo. Eso es crítico en un contexto de cambio climático, donde literalmente cada minuto es crítico.

1.3.4. Desde el punto ambiental

Este modelo representa una gran oportunidad para construir un futuro más seguro y sostenible en la región de Tacna. Al mejorar la capacidad de anticipar fenómenos climáticos, se pueden reducir significativamente los daños provocados por desastres naturales y fomentar prácticas más responsables con el medio ambiente. Tacna, como muchas otras regiones del Perú, enfrenta serios retos debido al cambio climático y a la

variabilidad de las lluvias. Por eso, una buena gestión del agua y una preparación adecuada frente a eventos extremos son fundamentales para proteger la seguridad alimentaria y el bienestar de las comunidades locales.

En este contexto, aplicar modelos de Machine Learning para predecir las precipitaciones en la estación Jorge Basadre no solo es una propuesta innovadora, sino también una alternativa prometedora que puede marcar una gran diferencia en la vida de las personas.

1.3.5. Desde el punto académico y científico

Esta investigación tiene un impacto enorme tanto en el mundo académico como en el científico, al abrir nuevas puertas en la conexión entre la ingeniería de sistemas y la meteorología. Desde una perspectiva académica, esto es un aporte realmente valioso, ya que analiza y compara diferentes modelos de aprendizaje automático como LSTM, GRU, ARIMA y Prophet para prever el clima. El enfoque está en la estación Jorge Basadre en Tacna, que es un área con un clima árido y muy variable, lo que hace que los resultados sean aún más significativos en un contexto real y complicado.

Estos modelos son capaces de procesar una gran cantidad de datos meteorológicos, tanto históricos como actuales, para identificar patrones y tendencias que ayudan a hacer pronósticos mucho más precisos. Al mejorar nuestra habilidad para anticipar el clima, podemos formular estrategias que disminuyan el impacto de desastres naturales, optimicen el uso del agua y fomenten prácticas agrícolas más sostenibles.

Además, el uso de estas tecnologías avanzadas también puede incentivar a las personas a ser más conscientes sobre la importancia de cuidar el medio ambiente y apostar por la sostenibilidad. Este enfoque basado en machine learning no solo busca que Tacna esté mejor preparada para enfrentar el clima, sino que también impulsa un futuro más seguro. y sostenible para las próximas generaciones.

1.4. Objetivos

1.4.1. Objetivo general

Determinar el modelo de machine learning de mayor exactitud para el pronóstico de precipitaciones en la estación Jorge Basadre de la Región Tacna – Perú, utilizando como variables predictoras la Fecha, Temperatura Máxima, Temperatura Mínima, Humedad Relativa y Precipitación Histórica.

1.4.2. Objetivos específicos

a. Generar un conjunto de datos para el pronóstico de precipitaciones en la estación Jorge Basadre de la Región Tacna - Perú.

b. Entrenar modelos de machine learning de precipitaciones en la estación Jorge Basadre de la Región Tacna - Perú.

c. Comparar la exactitud de los modelos de machine learning entrenados para el pronóstico de precipitaciones en la estación Jorge Basadre de la Región Tacna -Perú.

1.5. Hipótesis

1.5.1. Hipótesis general

El modelo de machine learning de mayor exactitud permite la predicción de pronóstico de precipitaciones en la estación Jorge Basadre de la Región Tacna – Perú, utilizando como variables predictoras la Fecha, Temperatura Máxima, Temperatura Mínima, Humedad Relativa y Precipitación Histórica.

1.5.2. Hipótesis específica

a. La generación de un conjunto de datos basado en los registros meteorológicos de la estación Jorge Basadre facilitará la creación de modelos más precisos para el pronóstico de precipitaciones en la región de Tacna - Perú.

b. El entrenamiento de modelos de machine learning con datos de la estación Jorge Basadre permitirá desarrollar predicciones precisas de precipitaciones en la región de Tacna - Perú.

c. La comparación de los modelos de machine learning entrenados con datos de la estación Jorge Basadre permitirá identificar el modelo más preciso para el pronóstico de precipitaciones en la región de Tacna - Perú.

CAPÍTULO II: MARCO TEÓRICO

2.1. Antecedentes de la investigación

2.1.1. Antecedentes internacionales

En Emiratos Árabes Unidos, Obaidalla (2024) en su investigación titulada “Predicción de precipitaciones mediante métodos de aprendizaje automático”, en esta investigación, se aplicaron la capacidad de anticipar con precisión las precipitaciones es fundamental tanto para la toma de decisiones económicas como para la implementación de medidas de prevención y seguridad. En este estudio, se analizó el comportamiento de tres modelos de predicción ampliamente utilizados: LSTM (Memoria a Largo Plazo), ARIMA (Modelo Autorregresivo Integrado de Media Móvil) y SARIMA (ARIMA con estacionalidad), con el objetivo de evaluar cuál ofrecía mejores resultados en la estimación de patrones de lluvia. Para ello, se llevó a cabo un análisis exhaustivo de los datos disponibles, teniendo en cuenta diversas escalas de tiempo y variaciones climáticas. Se partió de la idea de que el modelo LSTM, que se basa en redes neuronales profundas diseñadas para manejar secuencias de datos, tendría un rendimiento superior. Su habilidad para identificar relaciones complejas a largo plazo lo hacía especialmente prometedor para entender el comportamiento dinámico de las precipitaciones, que está influenciado por múltiples factores físicos y temporales. Gracias a su arquitectura avanzada, el LSTM pudo aprovechar grandes volúmenes de datos históricos sobre lluvias, lo que le permitió captar mejor las variaciones y patrones que otros modelos no logran distinguir con tanta precisión. Por otro lado, aunque ARIMA es eficaz para ciertas tareas de predicción, tiene limitaciones al no considerar de manera natural la estacionalidad, un aspecto crucial en el estudio de las precipitaciones. Esta debilidad afectó su rendimiento, especialmente en regiones donde el clima varía estacionalmente. En cambio, SARIMA incorpora componentes estacionales en la estructura de ARIMA, lo que lo convierte en una opción más adecuada para este tipo de análisis. Se esperaba que su rendimiento fuera mejor que el de ARIMA, aunque aún por debajo del logrado por LSTM, ya que no tiene la misma capacidad para captar relaciones no lineales complejas. Para entrenar y evaluar estos modelos, se utilizaron registros diarios de precipitaciones recopilados durante varias décadas. La evaluación del rendimiento se realizó mediante métricas comunes como el MAE (Error Absoluto Medio), el RMSE (Error Cuadrático Medio) y el MAPE (Error Porcentual Absoluto Medio), que permitieron medir con claridad la precisión y confiabilidad de cada enfoque. Los resultados confirmaron las expectativas: LSTM superó notablemente a los otros

modelos, demostrando ser una herramienta sólida y eficaz para el análisis de series temporales climáticas. ARIMA mostró un desempeño más limitado, especialmente en contextos donde la estacionalidad juega un papel importante.

En los Países Bajos Uzel (2023), en su investigación titulada “Análisis comparativo de LSTM, ARIMA y Prophet de Facebook para la previsión de tráfico: avances, desafíos y limitaciones”. Esta investigación internacional se enfocó en predecir el tráfico vehicular a corto plazo, analizando el rendimiento de tres modelos muy conocidos en el ámbito de las series temporales: LSTM (Long Short-Term Memory), ARIMA (Autoregressive Integrated Moving Average) y Prophet, que fue desarrollado por Facebook. El objetivo principal del estudio era identificar cuál de estos modelos ofrecía la mayor precisión en situaciones reales de movilidad urbana. Para lograrlo, se utilizó un conjunto de datos recopilado por 161 sensores de tráfico ubicados en diferentes lugares durante un periodo determinado. Los autores aplicaron una metodología rigurosa que incluyó el preprocesamiento de los datos, la configuración individual de cada modelo y su posterior evaluación mediante métricas estándar como el Error Cuadrático Medio (RMSE) y el Porcentaje de Error Absoluto Medio (MAPE), lo que permitió una comparación objetiva del rendimiento de cada enfoque. Los resultados mostraron que el modelo ARIMA superó tanto a LSTM como a Prophet, logrando un RMSE promedio de 5,5 y un MAPE de 25,77 %, lo que indica un nivel de precisión superior. En contraste, LSTM obtuvo un RMSE de 7,13 y un MAPE de 33,19 %, mientras que Prophet presentó un RMSE de 8,02 y un MAPE de 37,9 %. Este hallazgo es significativo, ya que desafía la creencia común de que los modelos de aprendizaje profundo, como LSTM, siempre son mejores que los modelos estadísticos tradicionales en tareas de predicción a corto plazo. El estudio demuestra que, en ciertos contextos y con datos específicos, modelos como ARIMA pueden ofrecer un rendimiento más sólido y confiable, lo cual es especialmente relevante para investigaciones aplicadas en áreas como el tráfico vehicular o el pronóstico del clima.

En Ecuador, López (2023) en su investigación titulada “Pronóstico de precipitación en el centro de Tungurahua aplicando aprendizaje estadístico con redes neuronales artificiales”, en este proyecto, se aplicaron redes neuronales artificiales para predecir la precipitación en Querochacha, provincia de Tungurahua. Se realizó un estudio sobre la aplicación de redes neuronales artificiales recurrentes para el pronóstico de precipitaciones en la estación meteorológica de Querochaca, ubicada en la provincia de Tungurahua. El análisis se basó en datos de lluvia correspondientes al período comprendido entre los años 2015 y 2021. El modelo propuesto está compuesto por una red neuronal recurrente de cinco capas, iniciando con una capa LSTM (Long

Short-Term Memory), la cual es especialmente adecuada para procesar información secuencial, como es el caso de las series temporales.

A esta primera capa le siguen cuatro capas completamente conectadas. La primera de ellas emplea la función de activación tanh y contiene 100 neuronas, mientras que las siguientes dos capas utilizan la función de activación ReLU, con 500 neuronas cada una. Estas tres capas intermedias incorporan un mecanismo de regularización mediante dropout con un valor de 0,02. Finalmente, la capa de salida está compuesta por una sola neurona con función de activación purelin, diseñada para realizar tareas de regresión lineal.

Durante el entrenamiento del modelo, se probaron distintas configuraciones, variando el número de neuronas en las capas intermedias (10, 50, 100, 500 y 1000 neuronas) y en la celda LSTM (10 y 100 unidades ocultas). La selección del mejor modelo se basó en el análisis del error cuadrático medio (RMSE) y en la comparación visual entre la gráfica de predicción y los valores reales. Se identificó que los hiperparámetros con mayor influencia en la calidad del pronóstico fueron el número máximo de épocas y la tasa de aprendizaje inicial. Los resultados más favorables se obtuvieron al emplear una cantidad reducida de unidades ocultas en la celda LSTM, alcanzando un error RMSE de 4,4535 y una curva de predicción altamente representativa del comportamiento real de la serie.

En México, Ponce y Guerrero (2023) en su investigación titulada “Pronóstico de temperatura en un invernadero usando algoritmos de aprendizaje supervisado”. Este estudio expone la predicción de la temperatura interna de un invernadero, considerando las condiciones climáticas tanto externas como internas en diferentes momentos. El propósito es realizar ajustes anticipados y desarrollar un sistema de control que contribuya a mantener un microclima óptimo mediante la aplicación de un algoritmo de aprendizaje automático. Se emplearon diversos métodos en un primer momento para comprender la complejidad de los datos, y se concluyó que no es necesario utilizar todos los parámetros dentro del invernadero para obtener una predicción aceptable.

En Colombia, Hernández (2022), en su investigación titulada “Modelo de pronóstico de demanda para productos del sector eléctrico”, se refiere a la propuesta formulada para mejorar los modelos de predicción de demanda actualmente empleados por la empresa Electrisol 1, específicamente en relación con los productos de alta rotación que representan más del 90 % de las ventas mensuales. Estos productos desempeñan un papel crucial en el proceso de compras del área de abastecimiento, ya que constituyen el primer paso en la previsión de inventarios. Actualmente, la empresa

utiliza en este proceso modelos clásicos de series temporales como Estacional simple, Aditivo de Holt-Winters y promedio móvil. Por este motivo, se llevó a cabo una evaluación de metodologías de Machine Learning y Deep Learning con el objetivo de mejorar la precisión en la predicción de cantidades vendidas.

Los resultados obtenidos revelan que los modelos de Gradient Boosting y Long Short Term Memory Neural Network presentan un rendimiento superior, utilizando la Raíz del Error Cuadrático Medio como métrica de comparación. En este sentido, los resultados promedio del RMSE fueron de 285,1 y 290,6, respectivamente. Por lo tanto, se sugiere implementar gradualmente estos modelos, comenzando con una muestra de control que permita medir el impacto en el nivel de servicio y la reducción de costos.

En Colombia, Naranjo (2021) en su investigación titulado “Pronóstico de la precipitación para la zona de influencia de la estación agroclimática Yariguies, utilizando técnicas de Machine Learning”, en este estudio se utilizaron datos de precipitación registrados en la estación agroclimática Yariguies, ubicada en Barrancabermeja, en el departamento de Santander, Colombia. La serie de tiempo analizada abarca desde el 1 de julio de 1967 hasta el 30 de septiembre de 2009. Se trabajó con un total de 19,266 registros, medidos en milímetros (mm). Para construir los modelos predictivos, el 80 % de los datos (15,412 registros) se destinaron al entrenamiento y el 20 % restante (3,854 registros) a la validación. Se probaron tres enfoques diferentes: el modelo estadístico Holt-Winters, árboles de decisión y una red neuronal secuencial basada en la arquitectura GRU. Las métricas utilizadas para evaluar el rendimiento de los modelos fueron MAE, MSE y RMSE, siendo la red neuronal GRU la que obtuvo los mejores resultados con valores de 0,05 mm, 0,01 mm y 0,1 mm, respectivamente. Sin embargo, este modelo presentó limitaciones a la hora de predecir lluvias fuertes (20-70 mm), intensas (70-150 mm) y torrenciales (más de 150 mm), ya que los errores en estas categorías fueron mayores de lo esperado. Por su parte, el modelo de árbol de decisión logró identificar correctamente este tipo de precipitaciones, pero mostró un bajo nivel de ajuste general. Esto podría explicarse porque los datos de precipitación presentan una naturaleza no lineal y compleja, con variaciones significativas en cantidad, frecuencia e intensidad dependiendo de la ubicación geográfica y del periodo analizado (día, mes, año), como señalan Mohini, Vipul y Harshadkumar. (2015). Además, se identificó un desequilibrio en los datos utilizados: aproximadamente el 75 % de los registros corresponden a precipitaciones inferiores a 5,3 mm y el 50 % a lluvias menores a 0,2 mm, lo que dificulta el aprendizaje de los modelos en eventos extremos.

En Bangladesh, Khan (2020), en su investigación titulada “Predicción de la temperatura y precipitación en Bangladesh utilizando redes neuronales recurrentes

LSTM", destacó la relevancia de contar con predicciones climáticas precisas, especialmente en contextos como el de Bangladesh, donde las variaciones en temperatura y lluvia inciden directamente en el desarrollo económico y en la aparición de enfermedades estacionales. Pese a la importancia de este tema, los autores identificaron una limitada aplicación de técnicas de inteligencia artificial, como las redes neuronales, en el análisis de patrones climáticos en dicho país. Por ello, su investigación implementó un modelo de tipo Long Short-Term Memory (LSTM) para predecir mensualmente la temperatura y la precipitación, utilizando una extensa base de datos climáticos que abarca 115 años, desde 1901 hasta 2015, conocida como Bangladesh Weather Dataset y disponible en la plataforma pública Kaggle. Este conjunto de datos incluye registros mensuales de temperatura y precipitaciones. Por ejemplo, en mayo de 1901 se registraron 27,88 °C y 267,21 mm de lluvia, mientras que en enero de 2014 se observaron valores de 17,10 °C y apenas 0,12 mm. El modelo fue entrenado utilizando el optimizador Adam con una tasa de aprendizaje de 0,001, y se configuró una red neuronal con 100 capas ocultas. Para la evaluación del rendimiento, se utilizó el Error Cuadrático Medio (MSE) como métrica de desempeño. Las pruebas se ejecutaron en un entorno computacional de alto rendimiento, haciendo uso de la librería PyTorch, una GPU NVIDIA RTX 2080Ti, y un procesador Intel Xeon E5-2620 acompañado de 16 GB de memoria RAM. Los resultados obtenidos fueron alentadores: se logró un error promedio de -0,38 °C en la predicción de la temperatura mensual, y de -17,64 mm en el caso de la precipitación, además de una baja desviación estándar en los errores. Esto evidenció la eficacia del modelo LSTM en la predicción de variables climáticas, abriendo paso a futuras investigaciones que podrían incorporar arquitecturas más avanzadas de aprendizaje profundo para mejorar aún más la precisión y eficiencia del análisis.

En Ecuador, Egas y Roque (2020), en su investigación titulada "Diseño de un modelo predictivo basado en técnicas de Machine Learning que permita determinar la temperatura usando los datos de una Miniestación Meteorológica en la ciudad de Guayaquil", este proyecto desarrolló un sistema para predecir la temperatura futura en la ciudad de Guayaquil. El sistema emplea una miniestación meteorológica para recoger datos sobre la temperatura, la humedad y la presión atmosférica. Estos datos se utilizan para entrenar un modelo predictivo que se basa en el aprendizaje automático, específicamente en la arquitectura de red neuronal recurrente (LSTM). Para evaluar el modelo, se compararon sus resultados con datos reales y con otros modelos predictivos similares. Los resultados mostraron que el modelo es preciso y puede ser utilizado para predecir la temperatura futura con confianza. El proyecto también incluye un sitio web donde se pueden visualizar los datos de la miniestación y las predicciones del modelo.

Este sitio web puede ser utilizado por ciudadanos, empresas y organizaciones para planificar sus actividades y tomar medidas de precaución ante variaciones de temperatura.

2.1.2. Antecedentes nacionales

En Ica, Quispe (2024) en su investigación titulada “Modelo de Predicción Climatológica con Inteligencia Artificial (AI) en la Región Ica, 2022”. Esta investigación se propuso modelar los parámetros meteorológicos utilizando técnicas de inteligencia artificial, con el objetivo de hacer una contribución significativa al análisis climático en el Centro de Investigación del Estudio de la Actividad Solar y sus Efectos Sobre la Tierra, que se encuentra en Ica, durante el periodo de 2019 a 2022. La metodología que se utilizó se basó en la estructura CRISP-DM, abarcando etapas como la preparación de los datos, el análisis de la temperatura, la implementación de modelos ARIMA y VAR para el pronóstico meteorológico, y la posterior evaluación de su rendimiento.

Los resultados indicaron que había algunos datos ausentes, y rellenarlos correctamente fue clave para mantener la coherencia temporal del conjunto. En términos del rendimiento de los modelos, ARIMA resultó ser más preciso que VAR al evaluarlo con varias métricas. En la discusión, se subrayó lo crucial que es manejar bien los datos incompletos y se propuso considerar modelos más avanzados en investigaciones futuras.

En Lima, Urbina y Takahashi (2023) en su investigación titulada “Desarrollo de un modelo de machine learning para el pronóstico meteorológico de precipitación a escala subestacional en Perú”. Este estudio expone los avances en el uso de Machine Learning (ML) para la predicción de precipitaciones en Perú a nivel subestacional, con un horizonte de hasta seis semanas. Se basa en el proyecto Subseasonal to Seasonal (S2S) para aplicar una técnica de downscaling, mejorando tanto la precisión como la resolución espacial de las predicciones, utilizando como referencia el producto PISCOp V2.1 del SENAMHI. Tras procesar los datos, se emplea un Análisis de Componentes Principales y se construyen modelos de regresión lineal múltiple para predecir estos componentes, usando los predictores proporcionados por el NCEP CFSv2 para cada día de pronóstico. Los resultados demuestran un buen rendimiento hasta el quinto día, sobre todo en la región amazónica, aunque la capacidad de predicción disminuye a partir de ese punto, dada la complejidad de los factores que inciden en las precipitaciones y la sencillez del modelo. Se propone incluir variables adicionales, como las oscilaciones

Madden-Julian (MJO), en versiones futuras del modelo para extender el horizonte de predicción.

En Chiclayo, Carreño (2023), en su investigación titulada “Modelo predictivo del proceso de ventas utilizando inteligencia de negocios y data analytics en la empresa Centro Textil De la Matta S.A.C.”, la empresa Centro Textil de la Matta S.A.C., ubicada en el Emporio Comercial Gamarra de Lima, Perú, se dedica a la producción y comercialización de telas deportivas. La empresa está en busca de mejorar su toma de decisiones y sus estrategias de negocio, pero se enfrenta a la dificultad de prever las ventas futuras y planificar la producción de manera efectiva. Para superar este reto, se creó un modelo de pronóstico de ventas que utiliza redes neuronales a través de técnicas de aprendizaje automático. Este modelo se fundamenta en inteligencia empresarial y análisis de datos, y fue entrenado con datos históricos de ventas de la empresa. Los resultados han demostrado que el modelo puede predecir las ventas futuras con gran precisión. Al implementar este modelo, la empresa podrá optimizar su toma de decisiones y sus estrategias comerciales. Por ejemplo, podrá planificar la producción de manera más eficiente, evitando tanto la escasez como el exceso de inventario.

En Piura, Nolasco (2023) en su investigación titulada “Aplicación de Machine Learning para Pronóstico de desplazamiento de lluvias usando imágenes del Radar de lluvias de UDEP”, un grupo de investigadores de la Universidad de Piura ha creado un sistema para prever el movimiento de tormentas, utilizando imágenes de un radar de lluvia. Este sistema emplea técnicas de aprendizaje automático para analizar las imágenes y anticipar tanto la ubicación como el desplazamiento de las tormentas. Para entrenar el modelo, los investigadores se basaron en datos del radar de lluvias de la Universidad de Piura y de otras instituciones. El modelo fue validado con datos nuevos, y los resultados demostraron que puede predecir las tormentas con gran precisión. Este sistema tiene el potencial de servir como una herramienta de alerta temprana para fenómenos que puedan impactar a Piura.

En Arequipa, Encina y Vargas (2023) en su investigación titulada “Técnicas de Machine Learning para la Predicción del caudal efluente de la Represa Condoroma”, este proyecto tiene como objetivo desarrollar un modelo predictivo de caudales efluentes en la represa Condoroma. Este modelo se utilizará para mejorar el conocimiento sobre la aplicación del aprendizaje automático para la predicción de caudales en represas. Además, el modelo se utilizará para contribuir a la gestión de riesgos de AUTODEMA y otras organizaciones relacionadas. El modelo podría ayudar a tomar decisiones informadas sobre la gestión, operación y mantenimiento de la represa. El modelo

también podría servir de guía para predicción de caudales en otras represas operadas por AUTODEMA. Además, podría servir como referencia para futuras investigaciones sobre la predicción de caudales en represas.

En Puno, Ponce (2022) en su investigación titulada “Modelo basado en Machine Learning para optimizar el Pronóstico de ventas de la empresa Ricos Pan, Año 2020 – 2021”. En los últimos años, el interés en Machine Learning ha aumentado, y esta investigación presenta un modelo diseñado para mejorar la predicción de ventas en la empresa Ricos Pan de la ciudad de Puno durante los años 2020 y 2021. Durante el preprocesamiento de datos, se abordaron registros faltantes debido al cierre temporal por la pandemia de Covid-19 y ventas atípicas en días festivos y fines de semana. Se aplicaron diversas técnicas de preprocesamiento, como diagramas de caja y bigotes, rangos intercuantiles y adición de atributos.

Se realizaron entrenamientos y pruebas con modelos como redes neuronales recurrentes (LSTM y GRU), redes neuronales convolucionales (CNN) y máquinas de vectores soporte (SVR), explorando diferentes épocas y transformaciones de datos, incluyendo logaritmos y diferencia simple. Los resultados indicaron que el modelo basado en redes neuronales convolucionales superó a otros modelos, y al compararse con el pronóstico manual de la empresa, mostró un rendimiento significativamente superior. La evaluación del desempeño se llevó a cabo utilizando el error porcentual absoluto medio.

En Lima, Ccoyccosi y Palomino (2021) en su investigación titulada “Propuesta de un modelo de machine learning para el pronóstico de la demanda de prendas de vestir en la Corporación Brusko S.A.C.” La investigación actual propone prever la demanda de los pantalones de hombre de la Corporación Brusko utilizando la metodología CRISP-DM y aplicando la técnica de regresión lineal. Se emplearon datos proporcionados por la empresa para el período de 2018 a 2021. se construyó el modelo de pronóstico y se calcularon los siguientes errores cuadráticos medios (RMSE): 23,78 para 2018, 13,22 para 2019, 47,12 para 2020 y 17,87 para 2021. Se llevó a cabo un análisis similar para todos los años, con un RMSE de 59,07. Los datos presentaron valores atípicos debido a la pandemia. Al eliminar estos valores atípicos, se volvió a ejecutar el modelo para todos los años, obteniendo un RMSE de 29,98, que resultó ser mejor en comparación con el modelo que incluía todos los datos sin valores atípicos. Al analizar los resultados, se optó por el modelo correspondiente al año 2019, ya que este año no presenta valores atípicos y tiene un RMSE adecuado.

2.2. Bases teóricas

2.2.1. Definición de precipitaciones

La precipitación se define como cualquier tipo de agua que proviene de la atmósfera y desciende hacia la superficie de la Tierra. Esto puede presentarse en forma líquida, como la llovizna y la lluvia, o en forma sólida, como la nieve y el granizo. Asimismo, abarca las lluvias que no son visibles, tales como el rocío y la escarcha blanca. Estos fenómenos son causados por cambios en la temperatura o la presión (Senamhi, 2023).

El pronóstico de precipitaciones se fundamenta en la recopilación y el análisis de datos meteorológicos, que incluyen factores como la temperatura, la humedad, la presión atmosférica y los patrones de viento. A través de modelos matemáticos y algoritmos de predicción, los meteorólogos son capaces de identificar tendencias y comportamientos en la atmósfera. Esta información les permite anticipar las condiciones climáticas futuras de manera más precisa.

2.2.2. Formación de la precipitación

La precipitación se origina a partir de la condensación del vapor de agua en la atmósfera. Cuando las gotas de agua se agrupan y alcanzan un tamaño crítico, vencen la resistencia del aire y caen por gravedad. Este proceso está influenciado por diversos factores como la humedad relativa, la temperatura, los vientos y la topografía (Pérez y Gómez, 2018).

2.2.3. Tipos de precipitaciones

Según la Organización Meteorológica Mundial (OMM, 2012) clasifica las precipitaciones según la forma en la que el agua se presenta al caer desde la atmósfera hacia la superficie terrestre. Esta clasificación distingue entre formas líquidas, sólidas o mixtas, y se describe de la siguiente manera:

a. Lluvia

La lluvia es la precipitación de gotas de agua líquida que caen libremente desde las nubes. Estas gotas, generalmente con un diámetro superior a 0,5 mm, pueden variar en tamaño.

Características:

- Se forma cuando las gotas de agua presentes en las nubes se agrupan, aumentan de tamaño y adquieren el peso suficiente para vencer la resistencia del aire, cayendo por acción de la gravedad.
- Puede presentarse de manera continua (lluvia estratiforme) o en forma de chubascos (lluvia convectiva).

Su intensidad se clasifica como:

- Débil: < 2,5 mm/h
- Moderada: 2,5 – 7,5 mm/h
- Fuerte: > 7,5 mm/h

Importancia:

Es la forma de precipitación más común y constituye una fuente esencial de agua dulce para ecosistemas, consumo humano y actividades agrícolas.

b. Llovizna

La llovizna es una precipitación compuesta por diminutas gotas de agua, con un diámetro inferior a 0,5 mm, que caen de manera uniforme y con muy baja intensidad.

Características:

- Se origina principalmente en nubes bajas del tipo estrato.
- Presenta una intensidad menor a 1 mm/h, lo que la hace percibirse más como humedad suspendida que como lluvia convencional.
- Puede reducir de forma considerable la visibilidad.

Importancia:

Aunque su aporte hídrico es limitado, influye en la seguridad vial y aérea, ya que puede favorecer la formación de niebla húmeda.

c. Nieve

La nieve es la precipitación que se presenta en forma de cristales de hielo o copos, generados por la sublimación del vapor de agua directamente a hielo, sin pasar por el estado líquido.

Características:

- Se forma en nubes con temperaturas inferiores a 0 °C.
- Los copos están compuestos por estructuras cristalinas que, debido a su baja densidad, descienden lentamente.
- Puede acumularse en el suelo, formando mantos nivosos.

Importancia:

Afecta el transporte, la agricultura y el equilibrio de los ecosistemas, especialmente en regiones de montaña y climas templados fríos.

d. Granizo

El granizo consiste en la caída de esferas o bloques de hielo, formados por el congelamiento de gotas de agua en el interior de nubes de gran desarrollo vertical, como los cumulonimbos.

Características:

- Se produce por fuertes corrientes ascendentes que mantienen las gotas congeladas en circulación, permitiendo que crezcan por capas.
- Su tamaño varía desde unos pocos milímetros hasta varios centímetros.
- La caída suele ser rápida y violenta.

Importancia:

Puede provocar daños significativos en cultivos, infraestructuras, vehículos y poner en riesgo la integridad física de las personas.

e. Aguanieve

La aguanieve es una precipitación mixta que combina lluvia y nieve, o gotas de lluvia parcialmente congeladas.

Características:

- Se presenta cuando la capa inferior de la atmósfera se encuentra cerca de 0 °C.
- Las partículas de nieve parcialmente derretidas pueden volver a congelarse o mezclarse con agua líquida durante su caída.
- Genera una sensación intensa de frío y humedad.

Importancia:

Es común en periodos de transición entre invierno y primavera, pudiendo generar superficies resbaladizas y condiciones de riesgo para la circulación.

2.2.4. Sistema meteorológico en el Perú

El sistema meteorológico en el Perú está conformado por un conjunto de instituciones, infraestructuras y procedimientos orientados a la observación, recolección, análisis y difusión de información climática a nivel nacional. Este sistema permite el monitoreo continuo de variables atmosféricas, facilitando la comprensión del comportamiento climático y la toma de decisiones en diversos sectores como la agricultura, gestión de riesgos y planificación territorial.

Dentro de este sistema, destaca el rol del organismo técnico encargado de centralizar y validar la información meteorológica, así como la red de estaciones que constituyen la fuente primaria de datos. La interacción entre estos componentes garantiza la disponibilidad de información confiable para su uso en estudios científicos y aplicaciones tecnológicas.

2.2.4.1. SENAMHI en el Perú

El Servicio Nacional de Meteorología e Hidrología del Perú (SENAMHI) es un organismo técnico especializado adscrito al Ministerio del Ambiente, responsable de generar, recopilar, procesar y difundir información meteorológica, hidrológica y climática en el territorio nacional.

SENAMHI cumple un rol fundamental en la vigilancia atmosférica mediante la implementación de estándares técnicos y metodológicos que aseguran la calidad y confiabilidad de los datos. Asimismo, desarrolla estudios, pronósticos y servicios climáticos que contribuyen a la prevención de desastres naturales y al desarrollo sostenible del país.

En el contexto de la presente investigación, SENAMHI constituye la fuente oficial de los datos meteorológicos utilizados, garantizando que la información empleada ha sido previamente validada bajo criterios técnicos establecidos.

2.2.4.2. Estaciones meteorológicas

Las estaciones meteorológicas son instalaciones diseñadas para medir y registrar variables atmosféricas como temperatura, humedad relativa, precipitación, presión atmosférica, velocidad y dirección del viento, entre otras. Estas estaciones pueden ser de tipo convencional o automática, dependiendo del nivel de tecnología y automatización en la recolección de datos.

En el Perú, las estaciones meteorológicas se encuentran distribuidas estratégicamente en diversas regiones, permitiendo obtener información representativa de las condiciones climáticas locales. La estación meteorológica Jorge Basadre, ubicada en la región Tacna, es una de las fuentes de información empleadas en esta investigación, proporcionando datos relevantes para el análisis y modelamiento de precipitaciones.

Estas estaciones constituyen el primer nivel de generación de datos, siendo fundamentales para el desarrollo de estudios climatológicos y modelos predictivos.

2.2.4.3. Relación entre SENAMHI y las estaciones meteorológicas

La relación entre SENAMHI y las estaciones meteorológicas se basa en un sistema integrado de monitoreo y gestión de datos. Las estaciones meteorológicas actúan como fuentes primarias de información, encargadas de registrar las variables atmosféricas en campo, mientras que SENAMHI cumple la función de supervisión, validación, procesamiento y difusión de dichos datos.

Este proceso implica la aplicación de controles de calidad, depuración de datos y estandarización de la información, asegurando su confiabilidad para fines científicos y operativos. De esta manera, SENAMHI garantiza que los datos provenientes de las estaciones meteorológicas cumplan con criterios técnicos que permitan su uso en investigaciones académicas.

En la presente investigación, esta relación es fundamental, ya que asegura que los datos utilizados para el entrenamiento y evaluación de los modelos de aprendizaje automático provienen de una fuente oficial, confiable y validada, lo cual fortalece la validez y rigor científico del estudio.

2.2.5. Definición de Machine Learning

El aprendizaje automático es una parte fascinante de la inteligencia artificial que permite a las computadoras aprender de los datos sin necesidad de ser programadas de manera

explícita. A diferencia de la programación tradicional, donde los programadores deben escribir instrucciones detalladas para cada tarea, el aprendizaje automático se basa en algoritmos que pueden identificar patrones en los datos (Bobadilla, 2021).

El aprendizaje automático es una rama fascinante de la inteligencia artificial (IA) y la informática que se centra en cómo utilizar datos y algoritmos para imitar la forma en que los humanos aprenden, mejorando su precisión de manera continua. Un modelo de machine learning es un tipo de software que las computadoras utilizan para tomar decisiones o hacer predicciones. Se clasifica en tres categorías principales: supervisado, no supervisado y por refuerzo. Lo interesante de estos modelos es que no necesitan ser programados de manera explícita para realizar tareas específicas; en cambio, se entrenan con grandes cantidades de datos, lo que les ayuda a mejorar su capacidad de generalización y les permite hacer predicciones precisas sobre datos que nunca han visto antes (MML, 2023).

2.2.6. Tipos de modelos de Machine Learning

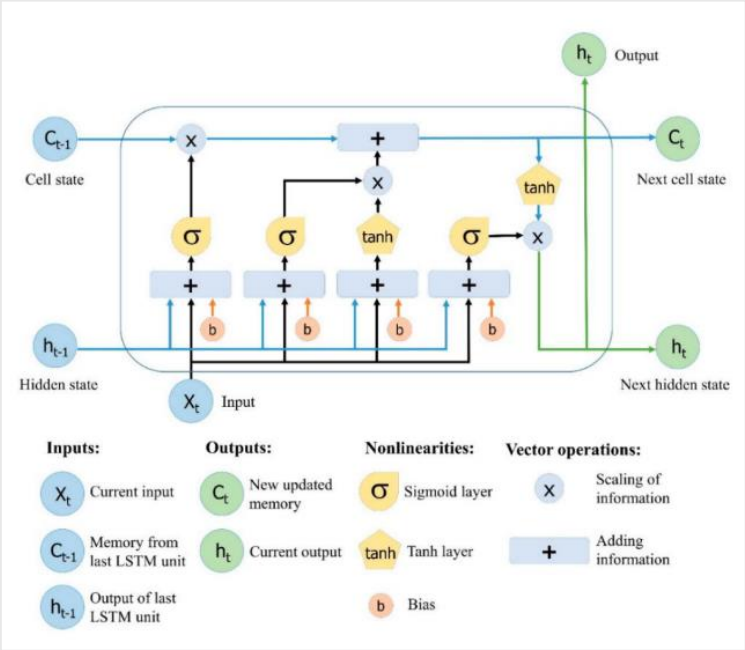
a. Modelo de LSTM (Long Short-Term Memory)

En varios ámbitos de la inteligencia artificial, como el procesamiento del lenguaje natural, el reconocimiento de voz, la visión por computadora y la predicción de series temporales, los datos suelen presentarse en forma de secuencias. Analizar este tipo de datos requiere modelos que puedan captar dependencias temporales complejas y retener información relevante a lo largo del tiempo. Las redes neuronales recurrentes (RNN) fueron una de las primeras soluciones a este desafío, ya que permiten procesar información secuencial mediante conexiones cíclicas. Sin embargo, en la práctica, las RNN tradicionales enfrentan dificultades para aprender dependencias a largo plazo debido a problemas como el desvanecimiento y el estallido del gradiente durante el entrenamiento (Bengio, 1994).

Para superar estas limitaciones, (Hochreiter y Schmidhuber, 1997) presentaron el modelo Long Short-Term Memory (LSTM). En la Figura 3 se observa una arquitectura recurrente mejorada que incluye una celda de memoria y un conjunto de compuertas (de entrada, olvido y salida). Estas compuertas funcionan como mecanismos de control que regulan el flujo de información dentro de la celda, permitiendo que el modelo aprenda qué datos conservar, cuáles descartar y cuándo utilizarlos para generar la salida.

Figura 3

Diagrama del modelo LSTM



Nota. El grafico de la estructura de una celda LSTM (Memoria a Largo y Corto Plazo).

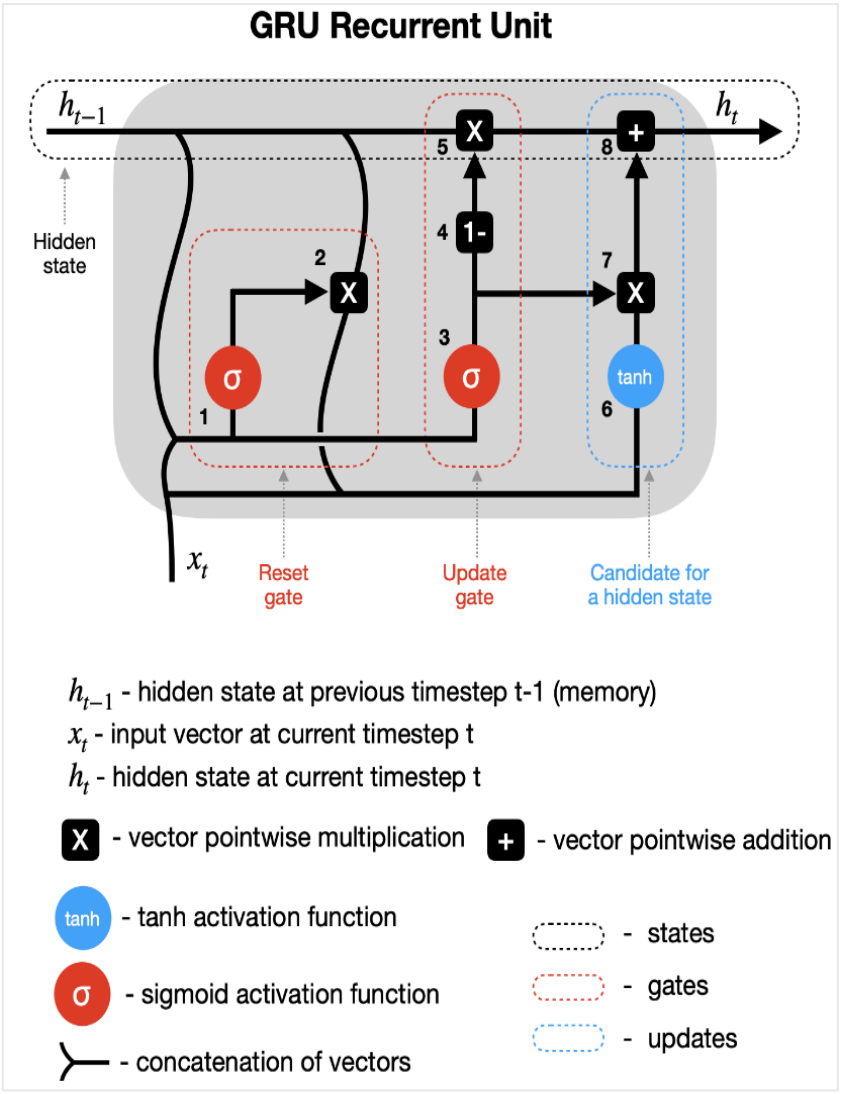
Gracias a su estructura, los LSTM han demostrado ser mucho más efectivos que las RNN tradicionales en tareas que requieren recordar patrones a largo plazo, como la traducción automática, la generación de texto y la predicción de señales temporales complejas.

b. Modelo de GRU (Gated Recurrent Unit)

El modelo GRU (Gated Recurrent Unit) es una arquitectura de red neuronal recurrente, En la Figura 4 se observa diseñada específicamente para el procesamiento y modelado de datos secuenciales o temporales, tales como texto, señales biomédicas, audio, video o series temporales financieras. Esta arquitectura fue introducida por (Cho , 2014) como una alternativa más eficiente y menos compleja en comparación con la Long Short-Term Memory (LSTM), con el objetivo de mitigar el problema del desvanecimiento del gradiente y mejorar la capacidad de aprendizaje en secuencias de largo plazo.

Figura 4

Diagrama del Modelo GRU



Nota. El grafico de la arquitectura interna de una unidad GRU (Gated Recurrent Unit).

A diferencia de las redes neuronales recurrentes (RNN) tradicionales, el modelo GRU introduce un mecanismo de compuertas adaptativas, que incluye la compuerta de actualización y la compuerta de reinicio. Estas compuertas gestionan de manera dinámica cómo fluye la información dentro de la red. Gracias a este mecanismo, se puede decidir qué información es importante conservar y cuál se puede descartar en cada momento, dependiendo del contexto actual y del anterior en la secuencia.(Chung , 2014). Esta estructura más compacta no solo permite un entrenamiento más ágil, sino que, en muchos casos, ofrece un rendimiento que es comparable o incluso superior al de las LSTM, especialmente en situaciones donde los recursos computacionales son limitados.

Varios estudios han respaldado la efectividad del modelo GRU en tareas de procesamiento del lenguaje natural (PLN), predicción de series temporales y análisis de sentimientos. Se ha destacado su habilidad para captar dependencias a largo plazo sin necesidad de arquitecturas complicadas (Zhou , 2020). Además, investigaciones recientes han demostrado que los modelos basados en GRU logran una buena generalización en contextos donde los datos son variables o ruidosos, lo que los convierte en una opción adecuada para aplicaciones en entornos del mundo real.

c. Modelo de ARIMA (AutoRegressive Integrated Moving Average)

El modelo ARIMA, que significa Promedio Móvil Integrado Autorregresivo, es una técnica de aprendizaje automático que se ha ganado su lugar como una herramienta confiable y muy utilizada para prever valores futuros basándose en datos históricos o en observaciones pasadas. Aunque ARIMA es conocido por ser un enfoque robusto y popular en el pronóstico de series temporales, gracias a su habilidad para detectar y anticipar patrones en los datos, sus aplicaciones se extienden a una variedad de sectores (Dilip, 2025).

Entre las diversas industrias que hacen uso frecuente de los modelos ARIMA, encontramos la predicción de la demanda energética, la gestión en el ámbito de la salud, la administración de cadenas de suministro, el marketing, el análisis del comportamiento del consumidor y las ciencias ambientales. Estos son solo algunos ejemplos de los múltiples usos que se pueden dar a los modelos ARIMA. En realidad, se trata de una técnica muy versátil que se puede adaptar a diferentes áreas y aplicaciones específicas de pronóstico dentro del análisis de series temporales.(Sen, 2016).

Sin embargo, en la Figura 5, el modelo ARIMA también tiene sus desventajas. Algunas de estas incluyen su sensibilidad a los parámetros, limitaciones en cómo maneja la estacionalidad, la incapacidad para captar relaciones no lineales y ciertos requisitos específicos de los datos. Estas desventajas pueden complicar su uso efectivo en entornos empresariales que necesitan pronósticos precisos.

Figura 5

Diagrama del modelo ARIMA

Nota. El grafico de la arquitectura interna del modelo ARIMA (AutoRegressive Integrated Moving Average. Elaboración propia.

d. Prophet (Modelo de Facebook)

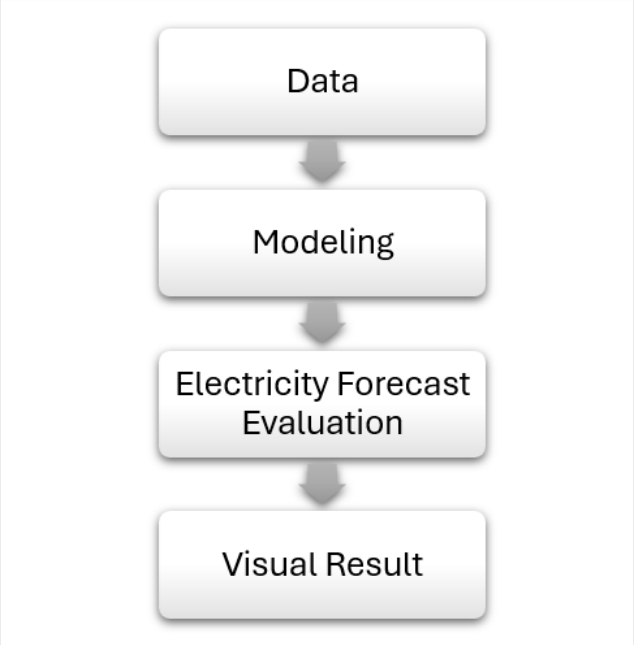
El modelo Prophet, creado por Facebook (ahora conocido como Meta), es una herramienta moderna para pronosticar series temporales. Su diseño busca lograr un balance entre precisión, flexibilidad y facilidad de uso, lo que permite modelar datos con tendencias no lineales, patrones estacionales complejos y eventos inusuales. Estos son aspectos que se encuentran en diversos campos, como el análisis económico, epidemiológico y climático (Taylor Letham, 2018).

Una de las características más destacadas de Prophet es su habilidad para identificar automáticamente los puntos de cambio en las tendencias, lo que permite una adaptación efectiva a series temporales que han sido influenciadas por intervenciones externas, cambios estructurales o comportamientos inusuales. Además, ha demostrado ser muy resistente ante datos faltantes, valores atípicos y estacionalidades irregulares, lo que lo convierte en una herramienta perfecta para trabajar con datos reales, que no siempre se comportan de manera ideal o limpia (Basak, 2022).

Prophet se ha implementado en lenguajes de programación muy utilizados como Python y R, lo que facilita su integración en flujos de trabajo analíticos y lo hace accesible para investigadores y profesionales con diferentes niveles de experiencia en modelado estadístico. En la Figura 6, su estructura parametrizable y su interpretación intuitiva han sido clave para su popularidad en el ámbito del análisis predictivo. (Pulzara y Losada, 2023).

Figura 6

Diagrama del modelo Prophet



Nota. El grafico del modelo Prophet, desarrollado por Facebook. Elaboración propia.

En el ámbito de la meteorología y el análisis ambiental, el modelo Prophet ha demostrado ser muy efectivo para predecir temperaturas, precipitaciones, humedad relativa y otros indicadores climáticos que muestran una clara estacionalidad. Estudios recientes han confirmado su eficacia en el modelado de series climáticas, resaltando su habilidad para captar patrones cíclicos y adaptarse a la variabilidad estacional de los datos. Esta flexibilidad hace que el modelo Prophet sea una herramienta clave para actividades como la planificación agrícola, la gestión de recursos naturales y la prevención de fenómenos climáticos extremos.

2.2.7. Dimensión de desempeño del algoritmo

a. Tiempo de respuesta

El tiempo de respuesta es una forma de medir cuán rápido un sistema puede reaccionar a una solicitud. Este tiempo se divide en dos partes: el tiempo de espera, que es el periodo que un usuario tiene que aguardar en la cola antes de recibir atención, y el tiempo de servicio, que es el tiempo que el sistema necesita para procesar la solicitud.

b. Inferencia de la IA

La inferencia en inteligencia artificial (IA) se refiere a la habilidad de los modelos previamente entrenados para identificar patrones y generar conclusiones basadas en datos nuevos y desconocidos.

c. Consumo de memoria

La memoria RAM constituye un elemento fundamental para el funcionamiento eficaz de cualquier sistema informático. La cantidad y el tipo de memoria RAM que tenga su sistema determinarán la velocidad a la que puede ejecutar programas y procesar datos (Barreto, 2023).

2.2.8. Indicadores de error

a. MAE (Error absoluto medio)

Willmott Matsuura (2005). El error absoluto medio (MAE) es una métrica para evaluar un error absoluto promedio entre dos datos, se formula de la siguiente forma:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \qquad (1)$$

- $\boldsymbol{n}$: Es el número total de observaciones o datos
- $\boldsymbol{y_i}$: Es el valor real o verdadero de la observación
- $\boldsymbol{\hat{y}_i}$: Es el valor predicho o estimado de la observación

b. MSE (Error cuadrático medio)

Willmott y Matsuura, (2005). El error cuadrático medio (MSE) es una métrica para evaluar un error cuadrático medio entre dos datos, que se expresa como:

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

- $\boldsymbol{n}$: Es el número total de observaciones o datos
- $\boldsymbol{y_i}$: Es el valor real o verdadero de la observación
- $\boldsymbol{\hat{y}_i}$: Es el valor predicho o estimado de la observación

c. MAPE (Error porcentual absoluto medio)

Willmott y Matsuura, (2005). El error porcentual absoluto medio (MAPE) es una métrica para evaluar un porcentaje de error absoluto. El MAPE es una representación porcentual del MAE, que se expresa como:

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \qquad (3)$$

- $\boldsymbol{n}$: Es el número total de observaciones o datos
- $\boldsymbol{A_t}$: Es el valor real o verdadero de la observación
- $\boldsymbol{F_t}$: Es el valor predicho o estimado de la observación

d. RMSE (Raíz del error cuadrático medio)

Willmott y Matsuura, (2005). El error cuadrático medio (RMSE) es una métrica utilizada para evaluar el error entre dos conjuntos de datos. El RMSE se calcula tomando la raíz cuadrada del MSE, de la siguiente manera:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \qquad (4)$$

- $\boldsymbol{n}$: Es el número total de observaciones o datos
- $\boldsymbol{y_i}$: Es el valor real o verdadero de la observación
- $\boldsymbol{\hat{y}_i}$: Es el valor predicho o estimado de la observación

2.3. Definición de términos

2.3.1. Precipitación

La precipitación es, básicamente, el agua que cae desde las nubes y llega hasta la superficie de la Tierra, ya sea en forma líquida o sólida. Puede manifestarse como lluvia, llovizna, nieve, aguanieve o granizo. Sin embargo, fenómenos como la virga (cuando la lluvia se evapora antes de tocar el suelo), la neblina o el rocío no se consideran precipitación, porque no provienen directamente de la caída del agua desde las nubes.

2.3.2. Intensidad de Precipitación

La intensidad de la precipitación es un índice que relaciona la cantidad de lluvia en función del tiempo. Su importancia radica no solo en la influencia que ejerce sobre el espacio geográfico, sino que también es considerada como un indicador de extremo climático, dado que las precipitaciones intensas tuvieron cambios significativos durante las últimas décadas (Blanco, 2020).

2.3.3. Frecuencia de Precipitación

Define la frecuencia como un parámetro fundamental en el análisis hidrológico, que permite estimar la recurrencia de lluvias extremas mediante distribuciones de probabilidad y momentos estadísticos (Campos, 2023).

2.3.4. Duración de Precipitación

Define la duración como el intervalo de tiempo, expresado en minutos u horas, durante el cual ocurre la precipitación máxima considerada en el análisis hidrológico (INAMHI, 2019).

2.3.5. Precipitación Acumulada

La precipitación acumulada es la profundidad total de agua líquida o equivalente líquido (incluyendo lluvia, nieve derretida, granizo, etc.) que cae sobre una superficie en un período determinado (Ahrens, 2009).

2.3.6. Serie temporal

Una serie temporal es simplemente un conjunto de datos de una misma variable que se van registrando en orden a lo largo del tiempo, ya sea día a día, mes a mes, año tras año u en otros periodos definidos. A este tipo de información es posible observar cómo

cambia la variable con el tiempo, detectar comportamientos repetitivos como tendencias, ciclos o estacionalidades, e incluso predecir cómo podría comportarse en el futuro a partir de lo que ha ocurrido en el pasado (Brillinger, 2001).

2.3.7. Machine Learning (Aprendizaje Automático)

El aprendizaje automático es una rama de la inteligencia artificial que tiene como propósito que los sistemas informáticos adquieran la capacidad de aprender a partir de la experiencia. En lugar de recibir instrucciones detalladas para cada tarea, se diseñan algoritmos capaces de procesar grandes volúmenes de datos, identificar patrones y optimizar su rendimiento de manera progresiva. El aprendizaje automático analiza la información disponible, construye un modelo a partir de ella y emplea dicho modelo para generar predicciones o resolver problemas, de forma semejante a cómo una persona mejora su desempeño a través de la práctica (Quintero, 2025).

2.3.8. Deep Learning (Aprendizaje Profundo)

El aprendizaje profundo es una de las áreas de la ciencia de datos de más rápido crecimiento. El aprendizaje profundo se refiere a una clase de algoritmos basados en redes neuronales artificiales optimizadas para trabajar con datos no estructurados como imágenes, voz, videos y texto. (Kotu y Deshpande, 2019).

2.3.9. Inteligencia Artificial (IA)

La inteligencia artificial (IA) se define como la capacidad de las máquinas de realizar tareas intelectuales habitualmente realizadas por humanos. Este término se utiliza como concepto general que engloba tanto el aprendizaje automático (AA) como el aprendizaje profundo (AP). Ambos conceptos pertenecen a un subcampo de la IA que se caracteriza por crear sistemas que son capaces de aprender, es decir, capaces de generar sus propias reglas sirviéndose únicamente a partir de los datos (Pérez, 2022).

2.3.10. Modelo pronóstico

Un modelo de pronóstico puede definirse como una herramienta analítica que, a partir de datos históricos y variables relevantes, permite estimar valores futuros de forma puntual o probabilística. Actualmente, estos modelos combinan tanto métodos estadísticos tradicionales como enfoques de aprendizaje automático, priorizando la

cuantificación de la incertidumbre, la consistencia jerárquica en diferentes niveles de agregación y la orientación hacia la toma de decisiones (Price, 2025).

2.3.11. Redes neuronales

Las redes neuronales son modelos de predicción, es decir, dados unos datos previos, son capaces de producir una predicción al enfrentar datos nuevos. Otros modelos de predicción más conocidos son la regresión lineal simple, la regresión múltiple o la regresión logística. Las redes neuronales obtienen mejores resultados que los anteriores ante problemas más complejos. En general, podemos dividir los modelos de predicción en modelos de clasificación y modelos de regresión. Los modelos de clasificación (Pérez, 2022).

2.3.12. Redes neuronales recurrentes (RNN)

Son un tipo de modelo de inteligencia artificial diseñado para trabajar con datos que llegan en forma de secuencia, como ocurre con las series de tiempo, el lenguaje, el audio o los videos. A diferencia de otros modelos, las RNN tienen la capacidad de “recordar” información de pasos anteriores, lo que les permite comprender mejor el contexto y las relaciones dentro de los datos. Estas memorias se utilizan en tareas muy cotidianas y útiles, como el análisis de videos, la creación automática de subtítulos, el procesamiento del lenguaje natural (como los asistentes virtuales), el reconocimiento de patrones musicales y la predicción de series temporales, lo que resulta clave en áreas como las finanzas, el pronóstico del clima o la salud (Mienye, 2024).

2.3.13. Redes neuronales convolucionales (CNN)

Las redes neuronales convolucionales (CNN o ConvNets) se utilizan principalmente en aplicaciones de visión artificial y clasificación de imágenes. Pueden detectar características y patrones dentro de imágenes y videos, lo que permite tareas como la detección de objetos, el reconocimiento de imágenes, el reconocimiento de patrones y el reconocimiento facial. Estas redes aprovechan principios del álgebra lineal, en particular la multiplicación de matrices, para identificar patrones dentro de una imagen (IBM, 2024).

2.3.14. Redes generativas antagónicas (GAN)

Las redes generativas antagónicas (GAN) son redes neuronales que se utilizan tanto dentro como fuera de la inteligencia artificial (IA) para crear nuevos datos que se

parezcan a los datos de entrenamiento originales. Pueden incluir imágenes que parezcan rostros humanos, pero son generadas, no tomadas de personas reales (IBM, 2024).

2.3.15. ARIMA (Autoregressive Integrated Moving Average)

El modelo autorregresivo integrado de medias móviles (ARIMA) es un método estadístico aplicado al análisis de series temporales, el cual se basa exclusivamente en los valores históricos de la variable y en los errores de estimación previos para realizar predicciones, sin necesidad de incorporar variables externas. Este enfoque, desarrollado y formalizado por Box y Jenkins en la década de 1970, se expresa como ARIMA (p, d, q), donde p corresponde al número de retardos autorregresivos, d al grado de diferenciación requerido para lograr la estacionariedad de la serie y q al número de términos de la media móvil (Spenassato, 2015).

2.3.16. Prophet

Prophet es un software de uso libre que permite hacer pronósticos de series de tiempo en R y Python. Funciona descomponiendo los datos en diferentes partes: la tendencia general, los patrones repetitivos como los que ocurren cada semana o cada año, los efectos de días festivos y, si se requiere, variables externas que influyen en el comportamiento de la serie. Su principal objetivo es generar predicciones fáciles de interpretar y confiables, incluso cuando los datos tienen vacíos, presentan cambios bruscos o incluyen valores atípicos. Además, ofrece la posibilidad de incorporar conocimiento del contexto y factores adicionales, lo que ayuda a mejorar la precisión de las proyecciones en horizontes como días, semanas o meses (Guo, 2021).

2.3.17. LSTM (Long Short-Term Memory)

Las redes neuronales de memoria a largo y corto plazo (LSTM) constituyen una variante avanzada de las redes neuronales recurrentes (RNN), cuyo propósito principal es resolver el problema del gradiente que se desvanece y posibilitar el aprendizaje de dependencias de largo alcance en secuencias. Su estructura se basa en una celda de memoria y en un conjunto de compuertas de entrada, de olvido y de salida que controlan el flujo de la información, determinando qué datos conservar, actualizar o descartar. Gracias a este diseño, las LSTM resultan especialmente útiles en el análisis de series temporales, el procesamiento del lenguaje natural y el tratamiento de señales biomédicas (Chambers, 2024).

2.3.18. GRU (Gated Recurrent Unit)

Las Gated Recurrent Unit (GRU) constituyen una variante de las redes neuronales recurrentes (RNN) que utilizan dos compuertas principales, la de actualización (update gate) y la de reinicio (reset gate), con el fin de controlar el paso de información a través de las secuencias. Estas compuertas permiten decidir qué elementos del estado previo deben mantenerse y cuáles conviene descartar, lo que facilita el aprendizaje de relaciones tanto de corto como de largo plazo en datos secuenciales, y poseen un diseño más sencillo y un menor número de parámetros, lo que se traduce en una mayor eficiencia computacional sin sacrificar la capacidad de modelar secuencias complejas. Debido a estas características, su aplicación es frecuente en áreas como el procesamiento de lenguaje natural, la predicción de series temporales y el análisis de patrones espacios temporales (Liu, 2024).

2.3.19. MAE (Error Absoluto Medio)

El Error Absoluto Medio (MAE) es una medida que calcula el promedio de las diferencias sin signo entre los valores observados y los estimados. Al expresarse en las mismas unidades que la variable analizada, permite conocer la distancia promedio entre las predicciones y los datos reales. Un valor cercano a cero indica que el modelo tiene una alta precisión (Springer, 2017).

2.3.20. MSE (Error Cuadrático Medio)

El Error Cuadrático Medio (MSE) es una métrica objetiva ampliamente empleada para medir el desempeño de los modelos, en particular cuando se trabaja con datos de distribución normal. Sin embargo, por sí solo no permite identificar en qué aspectos el modelo acierta o falla. Por ello, se plantea dividir el MSE en partes más comprensibles, asociadas a factores como la estacionalidad o la variabilidad (Hodson, 2021).

2.3.21. MAPE (Error Porcentual Absoluto Medio)

El Error Porcentual Absoluto Medio es una de las métricas más empleadas para medir la exactitud de modelos de predicción y pronóstico. A diferencia de otros indicadores de error, el MAPE refleja el promedio de la diferencia entre los valores reales y los estimados en forma de porcentaje, lo que lo convierte en una herramienta sencilla de interpretar y útil para comparar resultados en diferentes contextos y escalas. Gracias a esta cualidad, se aplica de manera

frecuente en áreas como la economía, la energía, la ingeniería y la inteligencia artificial, ya que facilita comprender qué tan distantes se encuentran, en promedio, las predicciones respecto a los valores observados. Un valor bajo de MAPE, cercano a cero, señala una alta precisión del modelo, mientras que valores mayores evidencian un rendimiento deficiente. Sin embargo, esta métrica presenta limitaciones, particularmente cuando los valores reales son muy pequeños, situación que puede generar porcentajes poco representativos (Coralogix, 2023).

2.3.22. RMSE (Raíz del Error Cuadrático Medio)

La raíz del error cuadrático medio (RMSE) es un indicador estadístico empleado para valorar la precisión de los modelos de regresión, ya que permite cuantificar la discrepancia entre los datos reales y las estimaciones generadas por el modelo. Su procedimiento de cálculo implica extraer la raíz cuadrada del promedio de los errores elevados al cuadrado, lo que proporciona una medida expresada en la misma unidad de la variable analizada. Esta característica hace que la RMSE represente el error típico de predicción, otorgando mayor peso a las desviaciones grandes. Gracias a su capacidad para reflejar con exactitud el rendimiento, se utiliza ampliamente en disciplinas como la meteorología, la ingeniería, la economía y el aprendizaje automático, con el fin de evaluar la exactitud y la confiabilidad de los resultados obtenidos (Mayer y Yang, 2024).

CAPÍTULO III: MARCO METODOLÓGICO

3.1. Diseño de la investigación

El diseño de esta investigación es experimental, según Hernández y Mendoza (2018). La investigación experimental se puede describir como aquella en la que no se manipulan intencionalmente las variables. En otras palabras, en este tipo de estudios no se alteran las variables independientes para observar su efecto sobre otras variables.

La investigación experimental no busca alterar intencionalmente las variables independientes. En lugar de eso, se centra en observar fenómenos que ya ocurrieron de forma natural, sin que el investigador intervenga directamente. Este tipo de estudio, conocido como investigación ex post facto, analiza hechos y variables que ya han tenido lugar, examinando cómo se relacionan en su entorno real, sin manipulaciones externas (Agudelo, 2008).

Por otro lado, las investigaciones experimentales se caracterizan por intervenir activamente en las variables. Esto implica:

- Modificar de manera intencional las variables independientes.
- Medir los efectos en las variables dependientes.
- Garantizar control y validez en el proceso.
- Comparar resultados entre dos o más grupos.

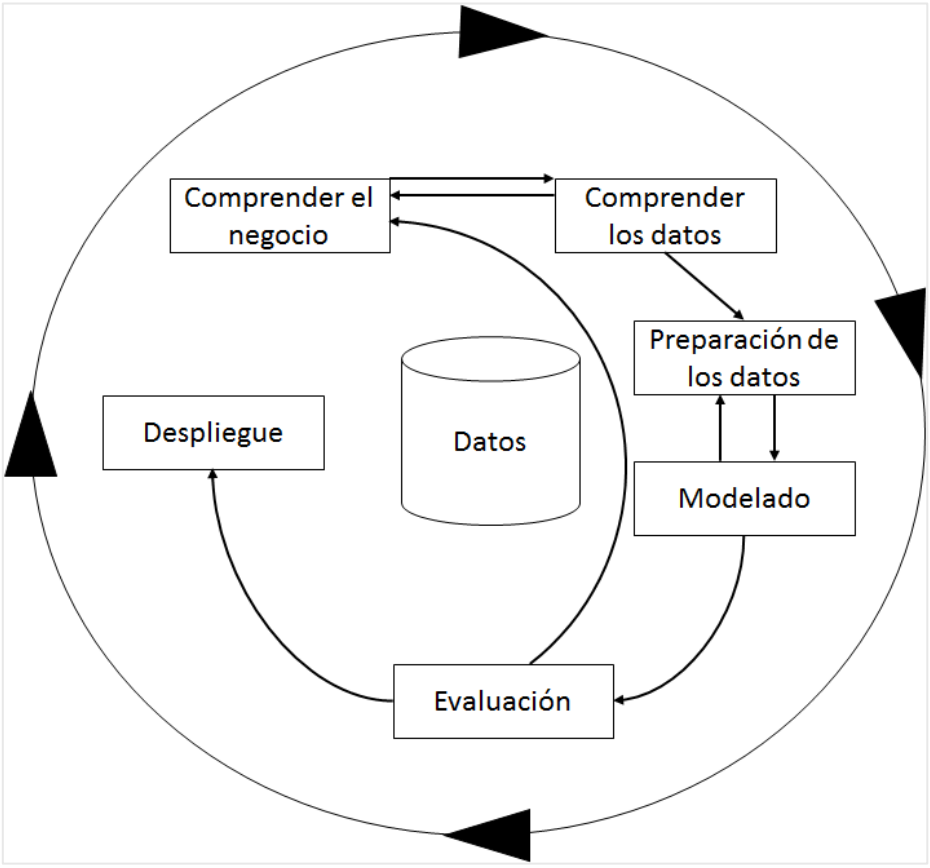
En estos casos, los participantes o elementos se asignan a los grupos de forma aleatoria o emparejada. El diseño de la investigación es como un mapa: una estrategia clara para recolectar la información necesaria, responder a las preguntas planteadas y alcanzar los objetivos definidos (Hernández Sampieri, 2018, p. 120).

En este estudio, por ejemplo, se trabajó con modelos de machine learning como variable independiente. Estos modelos fueron entrenados y luego se evaluó su desempeño midiendo la precisión de sus predicciones sobre precipitaciones. Para ello, se usó un conjunto de datos de entrenamiento con muestras asignadas al azar y un conjunto de validación para medir los resultados, asegurando la validez del experimento. Por estas características, el estudio se clasifica como una investigación experimental de tipo puro (Hernández Sampieri, 2018, p. 119). La matriz de consistencia que articula el problema, los objetivos, las hipótesis y las variables del estudio se presenta en el Anexo 1.

Para organizar el desarrollo del modelo de predicción, en la Figura 7 se siguió la metodología CRISP-DM (Cross Industry Standard Process for Data Mining), un enfoque ampliamente utilizado en proyectos de ciencia de datos y aprendizaje automático por su estructura clara y flexible. Esta metodología divide el proceso en seis etapas clave, que guían todo el ciclo, desde entender el problema hasta poner en práctica los resultados (Schröer, 2021). A continuación, se explica cómo se aplicó cada etapa en este proyecto:

Figura 7

Fases de la metodología CRISP-DM Aplicada en esta Investigación



Nota. El grafico de Proceso de Fases de la metodología CRISP-DM.

Comprensión del negocio: Antes de comenzar cualquier proyecto de minería de datos, lo más importante es comprender bien el negocio. En la Figura 8, esta etapa inicial es fundamental porque aquí se definen los objetivos y necesidades del proyecto desde una perspectiva empresarial. La idea es traducir esas metas en un plan técnico claro y bien estructurado.

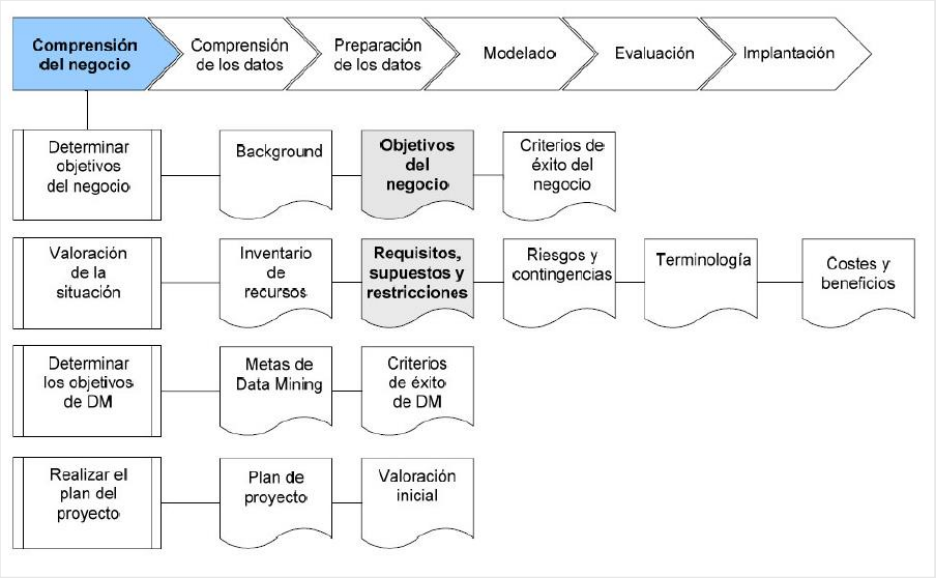
Si no logramos entender lo que realmente se quiere lograr, ningún modelo, por más sofisticado que sea, podrá ofrecer resultados útiles. Para que la minería de datos

sea efectiva, necesitamos tener una visión clara del problema que queremos resolver. Esto nos permitirá elegir los datos adecuados y darles el sentido correcto al analizarlos.

En esta fase, es esencial convertir ese conocimiento del negocio en un problema concreto de minería de datos y en un plan preliminar que nos acerque a los objetivos planteados. A continuación, se detallan las principales tareas que forman parte de este proceso inicial.

Figura 8

Fase comprensión del negocio de la metodología CRISP-DM



Nota. El grafico de Proceso de Primer Fase compresión del Negocio.

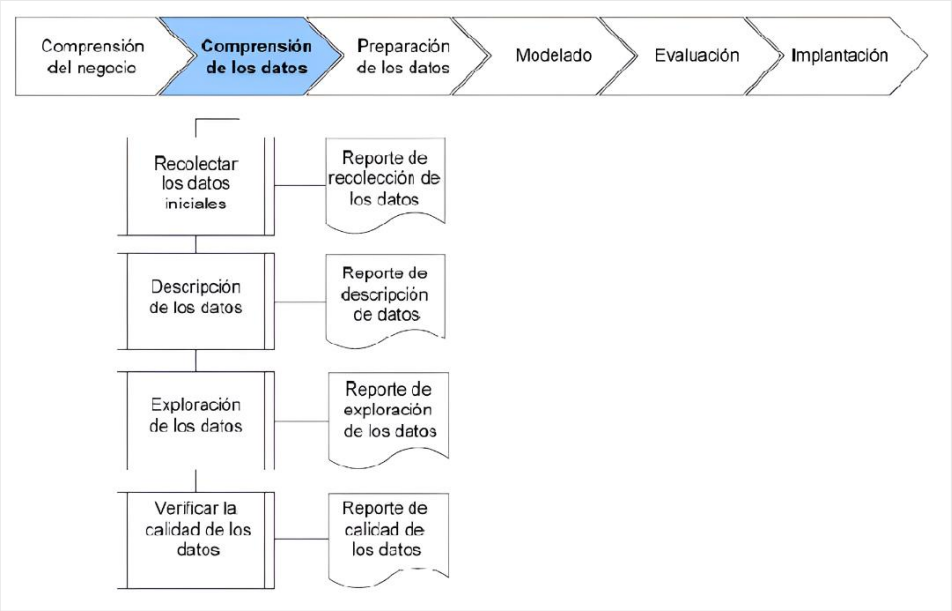
Comprensión de los datos: Una vez que entendemos el negocio, el siguiente paso es empezar a trabajar con los datos. En la Figura 9 en esta fase se enfoca en recolectar información por primera vez, lo que nos permite acercarnos al problema, conocerlo mejor, evaluar la calidad de los datos disponibles y descubrir patrones iniciales que nos ayuden a formular hipótesis.

Esta etapa junto con las dos que siguen suele ser una de las más demandantes en términos de tiempo y esfuerzo dentro de un proyecto de minería de datos. Si la organización ya cuenta con una base de datos corporativa, lo más recomendable es crear una nueva base específica para el proyecto. ¿Por qué? Porque durante el desarrollo se harán muchas consultas y posiblemente se modifiquen datos, lo que podría afectar el funcionamiento de la base original o generar conflictos.

A continuación, se detallan las tareas principales que forman parte de esta fase de exploración y entendimiento de los datos.

Figura 9

Fase c*omprensión de los datos de la metodología CRISP-DM*



Nota. El grafico de Proceso de Segundo Fase compresión de los datos.

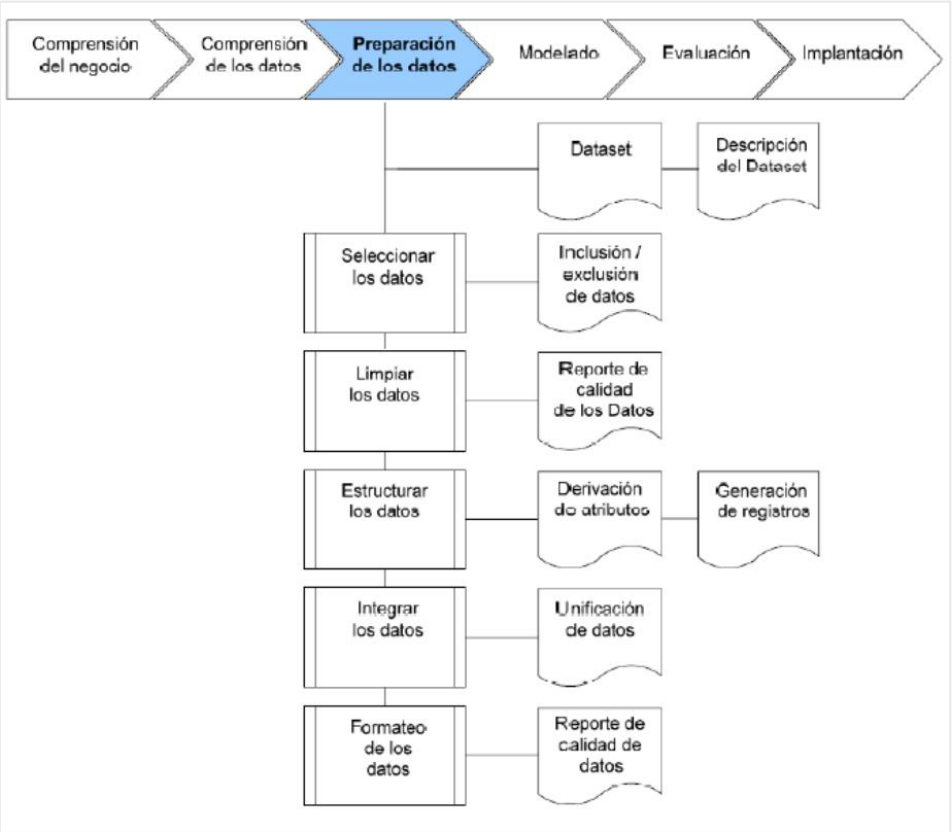
Preparación de los datos: Una vez que ya tenemos los datos iniciales en nuestras manos, el siguiente paso es prepararlos para que se adapten a las técnicas de minería de datos que vamos a aplicar. En la Figura 10 Esto puede incluir desde visualizar la información, hasta buscar relaciones entre variables o aplicar métodos más complejos para extraer valor.

La preparación de los datos implica varias tareas importantes: seleccionar qué datos se usarán para cada técnica, limpiar errores o inconsistencias, crear nuevas variables que puedan aportar valor, combinar datos de distintas fuentes y ajustar formatos para que todo funcione correctamente.

Esta etapa está muy conectada con la fase de modelado, ya que el tipo de modelo que elijamos influirá directamente en cómo debemos tratar los datos. Por eso, ambas fases suelen ir de la mano y retroalimentarse constantemente.

Figura 10

Fase preparación de los datos de la metodología CRISP-DM



Nota. El grafico de Proceso de Tercer Fase Preparación de los Datos.

Modelado: En esta etapa del proceso CRISP-DM, nos enfocamos en seleccionar las técnicas de modelado que mejor se ajusten al tipo de problema que queremos resolver con minería de datos. En la Figura 11, esta elección no se hace al azar: se basa en varios criterios importantes, como la naturaleza del problema, la disponibilidad y calidad de los datos, los requisitos específicos del proyecto, el tiempo que tenemos para desarrollar el modelo y el nivel de conocimiento que tenemos sobre cada técnica.

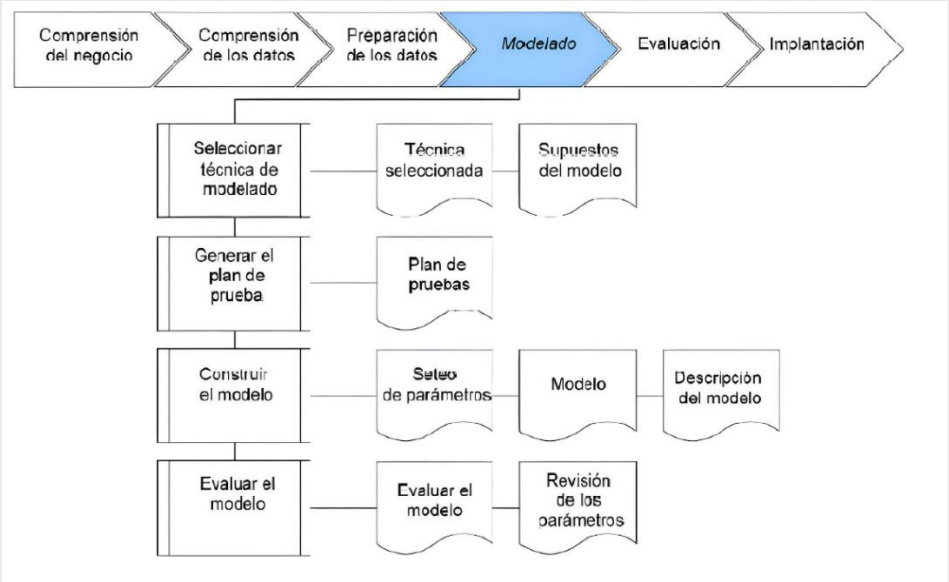
En el caso de este estudio, se exploraron y aplicaron distintos modelos de predicción para series temporales, como ARIMA, Prophet, LSTM y GRU. Cada uno tiene sus fortalezas: ARIMA y Prophet se basan en métodos estadísticos clásicos, mientras que LSTM y GRU son modelos de aprendizaje profundo capaces de identificar patrones complejos y no lineales en los datos.

Antes de construir los modelos, es fundamental definir cómo vamos a evaluar su rendimiento. Esto nos permite saber qué tan bien están funcionando y compararlos de forma justa. Una vez que tenemos claro el método de evaluación, pasamos a desarrollar

y probar los modelos. Los parámetros que usamos en esta fase dependen tanto de las características de los datos como del nivel de precisión que queremos alcanzar.

Figura 11

Fase modelado de los datos de la metodología CRISP-DM



Nota. El grafico de Proceso de Cuarta Fase Modelado.

Evaluación: En esta etapa del proceso CRISP-DM, el objetivo principal es evaluar el rendimiento del modelo para verificar si cumple con los criterios de éxito definidos al inicio del proyecto. En la Figura 12, Esta evaluación no solo se enfoca en los resultados obtenidos, sino también en la solidez del modelo frente a los datos utilizados.

Es importante tener presente que la precisión o fiabilidad del modelo está directamente relacionada con los datos empleados durante el análisis. Por lo tanto, sus resultados no siempre pueden generalizarse a otros contextos o conjuntos de datos distintos. Esta limitación nos obliga a ser cautelosos al interpretar los resultados y a considerar posibles sesgos o restricciones en los datos.

Además, esta fase implica una revisión crítica de todo el proceso. Los resultados pueden revelar la necesidad de ajustar parámetros, refinar el modelo o incluso regresar a etapas anteriores, como la preparación de datos o la selección del modelo. Este enfoque iterativo es clave para mejorar continuamente la calidad del análisis.

Para interpretar los resultados de manera efectiva, se pueden utilizar herramientas visuales (como gráficos de error o curvas de predicción) y técnicas analíticas que faciliten la comprensión del comportamiento del modelo.

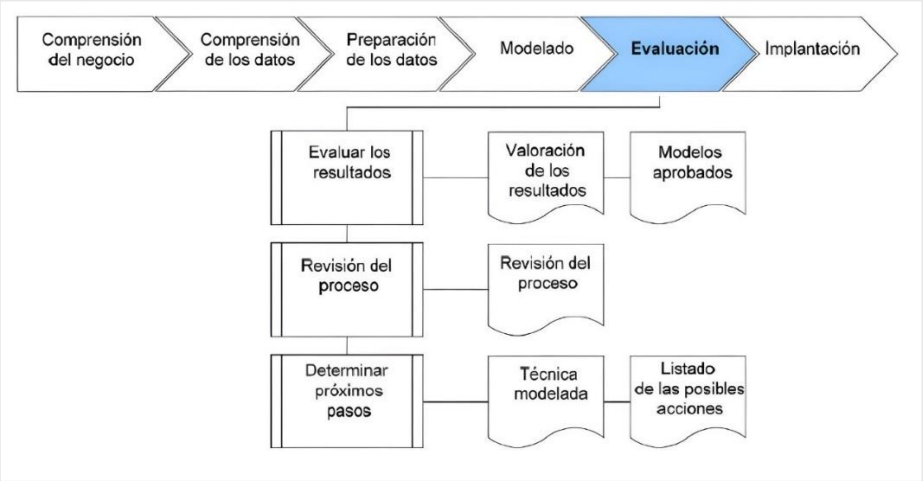
La efectividad del modelo se mide mediante métricas estadísticas específicas, que permiten cuantificar su rendimiento de forma objetiva y comparativa. Entre las más utilizadas se encuentran:

- MAE (Error Absoluto Medio): Mide el promedio de los errores absolutos entre las predicciones y los valores reales.
- MSE (Error Cuadrático Medio): Penaliza más los errores grandes, ya que eleva al cuadrado las diferencias.
- RMSE (Raíz del Error Cuadrático Medio): Facilita la interpretación del MSE al devolver el error en las mismas unidades que los datos originales.
- MAPE (Error Porcentual Absoluto Medio): Expresa el error como un porcentaje, lo que permite comparaciones más intuitivas.
- Precisión (Accuracy): Especialmente útil en modelos de clasificación, indica el porcentaje de predicciones correctas.

Si el modelo cumple con los estándares establecidos y demuestra un rendimiento sólido, se puede avanzar hacia su implementación práctica.

Figura 12

Fase evaluación de los datos de la metodología CRISP-DM



Nota. El grafico de Proceso de Quinta Fase Evaluación.

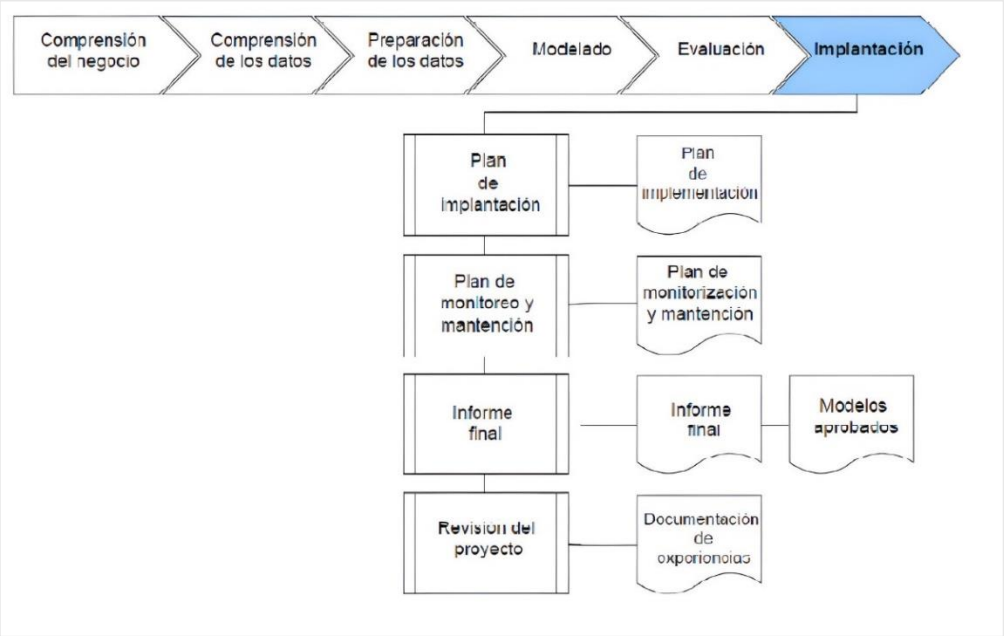
Despliegue: Una vez que el modelo ha sido desarrollado, probado y validado, llega el momento de ponerlo en práctica. En la Figura 13, en esta fase, el conocimiento generado se convierte en acciones concretas dentro del contexto del problema que estamos abordando. Esto puede reflejarse, por ejemplo, en recomendaciones que el analista propone basándose en los resultados del modelo, o en la integración del modelo dentro de un flujo de trabajo más amplio para analizar nuevos datos de forma continua.

Sin embargo, es importante entender que un proyecto de minería de datos no termina simplemente al implementar el modelo. Para que los resultados tengan un impacto real, es fundamental documentarlos de manera clara y accesible, de modo que cualquier persona interesada, ya sea dentro de la organización o en el área de aplicación, pueda comprenderlos y aprender de ellos.

Además, durante esta etapa también se debe garantizar el mantenimiento del modelo a lo largo del tiempo. Esto incluye monitorear su rendimiento, actualizarlo si cambian los datos o el entorno, y asegurarse de que siga siendo útil y relevante. Finalmente, compartir los hallazgos de forma efectiva puede aumentar su valor, permitiendo que otras áreas o equipos también se beneficien del conocimiento generado.

Figura 13

Fase despliegue de los datos de la metodología CRISP-DM



Nota. El grafico de Proceso de Sexta Fase Despliegue.

3.2. Acciones y actividades

Después de poner en marcha las variables y definir los instrumentos de análisis, comenzamos a organizar las acciones y actividades necesarias para llevar a cabo la investigación, asegurándonos de que se cumplieran los objetivos específicos que habíamos planteado. Estas acciones se organizaron de acuerdo con las fases de la metodología CRISP-DM, siguiendo un orden lógico y riguroso que garantiza la validez del proceso.

La validez de los modelos que implementamos se basa en un análisis exhaustivo de investigaciones anteriores, que se mencionan en los antecedentes de este estudio, así como en la revisión de la literatura científica más reciente. En la Figura 14, también tomamos en cuenta las recomendaciones de expertos y las prácticas estandarizadas en el ámbito de la predicción de series temporales, lo que respalda la fiabilidad de los resultados que esperamos obtener.

Las principales acciones realizadas incluyeron:

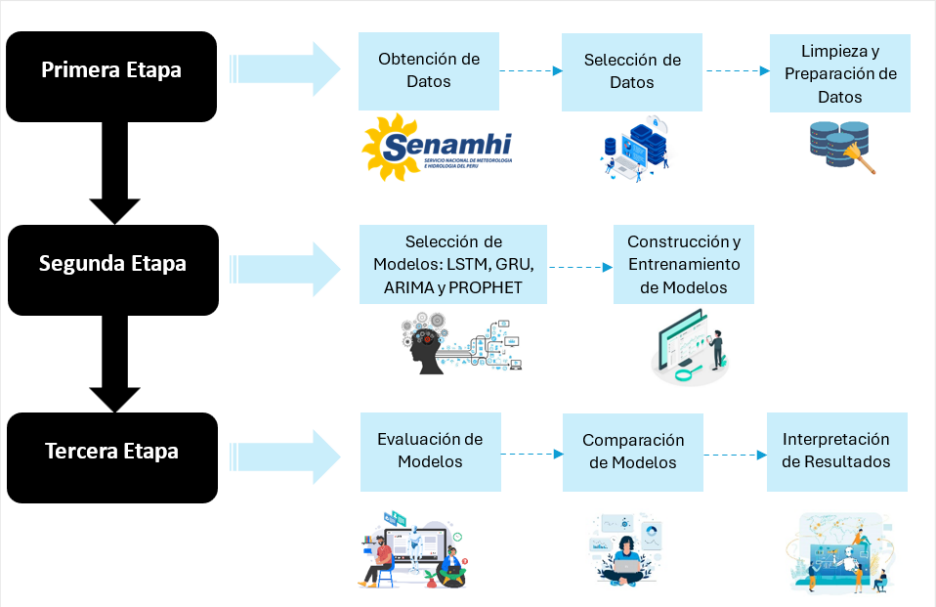
- *Recolección y limpieza:* Obtención de los datos meteorológicos históricos provenientes del SENAMHI y depuración de inconsistencias.
- *Análisis exploratorio de datos (EDA):* Identificación de tendencias, estacionalidades y valores atípicos mediante visualización estadística.
- Preparación de los datos: Aplicación de técnicas de normalización y escalamiento, así como la estructuración de ventanas de tiempo para alimentar los modelos secuenciales.
- *Fase de modelado y configuración:* Para la implementación de los algoritmos, se diseñaron arquitecturas específicas según la naturaleza del modelo. En el caso de los modelos de aprendizaje profundo (LSTM y GRU), se estableció una estructura secuencial para procesar tensores tridimensionales. El entrenamiento incluyó la optimización de hiperparámetros mediante búsqueda iterativa, ajustando el número de capas ocultas, la cantidad de neuronas por capa y la tasa de aprendizaje (learning rate). Se implementaron capas Dropout para mitigar el sobreajuste y se utilizó el algoritmo Adam como optimizador, empleando el Error Cuadrático Medio (MSE) como función de pérdida para guiar la convergencia. El flujograma completo que representa el proceso metodológico empleado se incluye en el Anexo 4.

Los modelos implementados de machine learning especializados en series temporales fueron los siguientes:

- *LSTM (Long Short-Term Memory):* Red neuronal recurrente con capacidad de capturar dependencias a corto y largo plazo, ideal para series temporales no lineales. Utiliza mecanismos de compuertas (entrada, olvido y salida) para controlar el flujo de información (Chen, 2025).
- *GRU (Gated Recurrent Unit):* Variante simplificada de LSTM, capaz de modelar dependencias a largo plazo mediante compuertas de actualización y reinicio, lo que reduce su complejidad computacional (Seifi, 2024).
- *ARIMA (AutoRegressive Integrated Moving Average):* Modelo estadístico clásico de series de tiempo, útil para capturar componentes autorregresivos, de integración y de media móvil en datos históricos (Meng, 2025).
- *Prophet:* Herramienta desarrollada por Facebook para modelar tendencias a largo plazo, estacionalidades múltiples y eventos especiales en series temporales, permitiendo generar pronósticos robustos y flexibles (Huang, 2024).
- El estudio sigue una metodología de cinco fases, avanzando de manera sistemática desde la conceptualización hasta la conclusión.
- *Fase de Validación y Evaluación:* Para garantizar la generalización de los resultados obtenidos, se aplicó una estrategia de división por tiempo (80% entrenamiento y 20% prueba). El modelo se prueba en 5 subconjuntos de datos diferentes por ciclo con validación cruzada usando k-fold (TimeSeriesSplit), para evaluar la estabilidad del modelo tras varios tipos diferentes de conjunto de datos y diferencias significativas al mismo tiempo. La exactitud de los modelos se evaluó empleando métricas estandarizadas ampliamente utilizadas en estudios de pronóstico. En primer lugar, el Error Absoluto Medio (MAE) permitió cuantificar la magnitud promedio del error cometido en las predicciones, ofreciendo una interpretación directa en las mismas unidades de la variable analizada. Asimismo, se utilizaron el Error Cuadrático Medio (MSE) y su raíz, el RMSE, los cuales penalizan de manera más severa los errores de gran magnitud, siendo especialmente útiles para identificar desviaciones significativas en el modelo. Finalmente, se incorporó el Error Porcentual Absoluto Medio (MAPE), que expresa el error de predicción en términos porcentuales y facilita la comparación relativa entre modelos o períodos de análisis. Estas métricas, en conjunto, proporcionan una evaluación integral del desempeño predictivo, permitiendo seleccionar el modelo que mejor balancea error promedio y sensibilidad a errores extremos.

Figura 14

Proceso de entrenamiento de datos para las predicciones



Nota. El grafico de Proceso de Entrenamiento de datos para las predicciones en 3 Etapas. Elaboración Propia.

3.3. Materiales e instrumentos

En el ámbito de la metodología de investigación para tesis, los autores de la obra Metodología de la investigación (Hernández-Sampieri Mendoza, 2018) explican que los instrumentos y técnicas de investigación son las herramientas que utilizamos para recolectar datos. Los instrumentos son esos elementos físicos que nos ayudan a obtener información, como un formulario, una entrevista, una prueba o una observación registrada. Por otro lado, las técnicas son los métodos o procedimientos que empleamos para conseguir esos datos, como la observación directa, las encuestas, las entrevistas o el análisis de documentos.

Además, es importante entender que la técnica de investigación se refiere a la forma específica de obtener la información necesaria para el estudio, mientras que el instrumento es cualquier herramienta o formato (ya sea en papel o digital) que utilizamos para adquirir, registrar o conservar datos(Arias, 2012,p.67-68).

Sin embargo, vale la pena mencionar que los autores se centran en métodos de investigación más tradicionales, como el método experimental o descriptivo. El aprendizaje automático es un enfoque relativamente nuevo y no se menciona de manera directa en el texto. En cuanto a la inteligencia artificial, podríamos argumentar que los

instrumentos incluyen el software y hardware que utiliza el investigador, mientras que las técnicas se refieren a los algoritmos y métodos informáticos que se aplican.

En este contexto, en la investigación actual se emplearon las siguientes técnicas e instrumentos:

Se llevó a cabo la recopilación de datos abiertos de los portales de SENAMHI. Luego, extraeremos la información obtenida, que ha demostrado ser una fuente valiosa para reunir datos clave, especialmente durante el período con la mayor cantidad de datos históricos, que va desde el 01 de enero de 2021 hasta el 31 de marzo de 2025. Estos datos, que se encuentran en las bases de datos de SENAMHI, son fundamentales para nuestro análisis y enriquecerán significativamente nuestra investigación. Utilizaremos fichas de recolección para obtener información relacionada con los pronósticos de precipitaciones. En la fase inicial de nuestra intervención, hemos considerado diversas características generales, incluyendo datos geográficos, variables meteorológicas específicas y detalles sobre los métodos utilizados.

*Técnica***:** Machine Learning (Aprendizaje de Máquina): En este trabajo, se utilizó un modelo de aprendizaje automático basado en la regresión lineal, elegido por su habilidad para captar las relaciones lineales entre las variables independientes y dependientes. Para su desarrollo, se empleó Scikit-learn, una biblioteca que proporciona herramientas avanzadas para crear y evaluar modelos de machine learning. La regresión lineal es un algoritmo que se puede interpretar fácilmente y es muy eficiente, ideal para situaciones donde las variables están relacionadas de manera aproximadamente lineal, lo que hace que su implementación y la validación de los resultados sean más sencillas. Este enfoque permitió analizar el comportamiento de las variables y hacer predicciones precisas con los datos disponibles.

Instrumento: Para el diseño y ejecución de los experimentos, se utilizó el siguiente hardware y software informático, proporcionado a través de una suscripción a Google Colab:

Se mencionan los siguientes componentes:

Procesador AMD Ryzen 5 5600GT con gráficos Radeon, funcionando a 3600 MHz, que cuenta con 6 núcleos y 12 hilos lógicos. Además, tiene 16 GB de memoria RAM instalada y gráficos integrados.

Procesador: INTEL CORE 5 con gráfico NVIDIA GeForce RTX 3050 210H, que cuenta con 12 núcleos, memoria RAM: 16 GB y disco sólido 500 GB.

Asimismo, se especifican las librerías de Python que se utilizarán para la limpieza de datos, entrenamiento de modelos y análisis del procesamiento.

Las técnicas e instrumentos mencionados son los mismos que se han utilizado en estudios similares analizados en los antecedentes de esta investigación.

Confiabilidad y consistencia de datos

Respecto a la validación de los instrumentos, considerando que la investigación utiliza registros históricos de una fuente oficial (SENAMHI), la confiabilidad no se sustenta en la validación de instrumentos tradicional (cuestionarios), sino en la verificación de la calidad y consistencia estadística de la información. Para garantizar que los datos sean representativos y fiables, se aplicó un procedimiento de depuración técnica basado en tres criterios fundamentales:

Continuidad de la serie: Se aseguró la integridad de la secuencia cronológica mediante técnicas matemáticas de estimación lineal, lo que permitió completar los vacíos de información sin alterar la tendencia histórica de los registros.

Depuración de datos (control de valores extremos): Se aplicaron medidas de dispersión y análisis de cuartiles para identificar y filtrar mediciones anómalas o erróneas que no corresponden al comportamiento físico real del clima, garantizando así la limpieza de la muestra.

Análisis de estabilidad (estacionariedad): Se verificaron las propiedades estadísticas de la serie mediante pruebas de hipótesis de raíz unitaria, confirmando que el comportamiento de los datos es adecuado para realizar proyecciones fiables a futuro.

3.4. Población y muestra de estudio

La población se define como el conjunto de todos los casos que cumplen con una serie de especificaciones (delimitación) (Hernández Sampieri, 2018, p. 120). En este estudio, consideramos a esta población como la unidad principal para analizar los datos, lo que nos permite abordar de manera precisa el objetivo de la investigación. Para definir la población del estudio, utilizaremos los datos abiertos que nos proporciona el Servicio Nacional de Meteorología e Hidrología (Senamhi). Esta información es detallada y accesible al público, lo que resulta fundamental para entender de manera precisa y actualizada el contexto ambiental que impacta a la población. La población del estudio se basa en la información del SENAMHI, correspondiente al período del 1 de enero de 2021 al 31 de marzo de 2025. Estos datos fueron empleados para entrenar el algoritmo

predictivo que hemos desarrollado. La Tabla 1: Es importante destacar que comenzamos trabajando con datos en bruto, es decir, sin ningún tipo de procesamiento previo. El conjunto de datos incluyó variables como la fecha, la temperatura máxima, la temperatura mínima, la humedad y la precipitación, sumando un total de 1658 registros.

Tabla 1

Datos de data set

N.º	Fecha	Temp. Máx	Temp. Mín	Humedad	Precipitación
1	01/01/2021	27,6	18,0	81,9	0,0
2	02/01/2021	28,0	18,2	79,1	0,0
3	03/01/2021	27,6	18,6	80,1	0,0
4	04/01/2021	28,0	19,2	76,7	0,0
5	05/01/2021	27,6	16,8	76,3	0,0
6	06/01/2021	28,4	16,6	76,1	0,0
7	07/01/2021	27,0	17,2	79,0	0,0
8	08/01/2021	27,4	17,4	79,9	0,0
9	09/01/2021	27,2	16,4	78,5	0,0
10	10/01/2021	26,6	16,8	80,8	0,0

Nota. Tabla de Registro de datos de SENAMHI de la estación Jorge Basadre – Tacna

Muestra

Se utilizaron datos abiertos que incluyen conjuntos de datos para investigar el pronóstico de precipitaciones significativas en la región de Tacna, específicamente en la estación Jorge Basadre.

La muestra estará compuesta por datos abiertos recopilados desde el 1 de enero de 2021 hasta el 31 de marzo de 2025, provenientes de los portales de SENAMHI. Estos datos se utilizarán para entrenar el algoritmo predictivo correspondiente. Es importante mencionar que estos datos se depurarán adecuadamente, asegurando su limpieza y calidad para su uso en la predicción.

Para este estudio, se utilizaron datos proporcionados por el SENAMHI, abarcando el periodo desde el 1 de enero de 2021 hasta el 31 de marzo de 2025 (La Tabla 2). Estos datos fueron la base para entrenar el modelo predictivo que se desarrolló. Es importante destacar que se trabajó únicamente con información depurada, es decir, sin errores ni valores nulos como “NaN” o “0”. Cada registro incluía

detalles sobre la fecha, la temperatura máxima, la temperatura mínima, la humedad y la precipitación. En total, se recopilaron 560 datos limpios, los cuales fueron esenciales para realizar un proceso de predicción más preciso y confiable.

Tabla 2

Datos de años o periodos

Nº	Año	Cantidad de Datos
1	2021	143
2	2022	144
3	2023	144
4	2024	120
5	2025	9
	Total, de datos (count)	560

Nota. Tabla de Datos proporcionados por el SENAMHI, correspondientes al periodo comprendido entre los años 2021/01/01 y 2025/03/31.

3.5. Operacionalización de variables

Tabla 3

Identificación y/o Caracterización de las Variables

Variable	Definición Conceptual	Dimensiones	Indicador	Escala	Técnicas o Métodos
Variable independiente Modelo de Machine Learning	Un modelo de Machine Learning es un software que las computadoras emplean para tomar decisiones o hacer predicciones. Se divide en tres tipos principales: supervisado, no supervisado y por refuerzo. Estos modelos no requieren ser programados explícitamente para realizar tareas específicas, sino que se entrenan con grandes volúmenes de datos para mejorar su capacidad de generalización, lo que les permite hacer predicciones precisas sobre datos no vistos previamente (MML, 2023.)	Desempeño del algoritmo	Tiempo de Respuesta Inferencia Consumo de Memoria	Razón	Técnica Observación Instrumento: Ficha de Observación

Tabla 4 *(Continuación)*

Variable	Definición Conceptual	Dimensiones	Indicador	Escala	Técnicas o Métodos
Variable dependiente Pronóstico de Precipitaciones	El pronóstico de precipitaciones se fundamenta en la recopilación y el análisis de datos meteorológicos, que incluyen factores como la temperatura, la humedad, la presión atmosférica y los patrones de viento. A través de modelos matemáticos y algoritmos de predicción, los meteorólogos son capaces de identificar tendencias y comportamientos en la atmósfera. Esta información les permite anticipar las condiciones climáticas futuras de manera más precisa.	Exactitud	MAE (Error absoluto medio) MSE (Error cuadrático medio) RMSE (Raíz del Error Cuadrático Medio) MAPE (Error absoluto relativo)	Razón	

Nota. Operacionalización de variables.

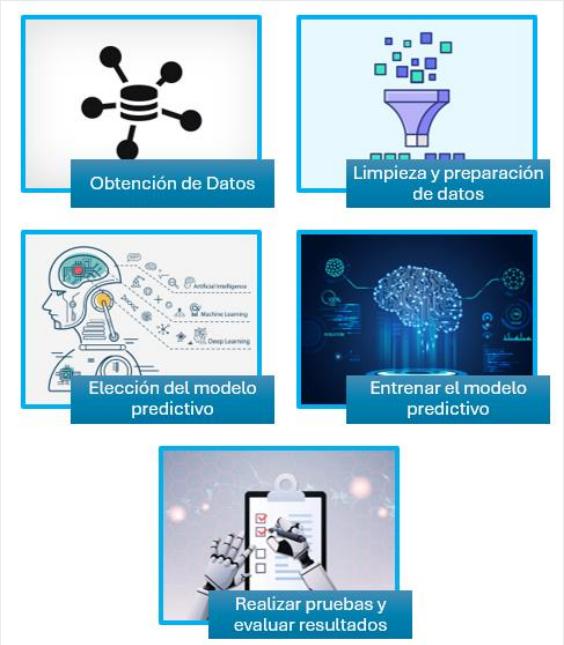
3.6. Técnicas de procesamiento y análisis estadístico

Técnica de Procesamiento

Para desarrollar un modelo de pronóstico de precipitaciones en la estación Jorge Basadre (Tacna) con datos de 2021/01-01 a 2025/03/31, se seguirá este flujo de trabajo en la Figura 15.

Figura 15

Técnicas de Procesamiento de Datos



Nota. El grafico de Procesamiento de datos de 5 fases. Elaboración Propia.

a. Obtención de datos: Reunir datos históricos de precipitaciones y variables meteorológicas relevantes.

b. Limpieza y preparación: Tratar valores faltantes o extremos y escalar las variables. Crear nuevas características si es necesario.

c. Selección del modelo: Evaluar modelos como LSTM, GRU, ARIMA y Prophet, considerando patrones estacionales y tendencias en los datos.

d. Entrenamiento: Dividir los datos en entrenamiento y prueba, y ajustar los modelos mediante optimización de hiperparámetros.

e. Pruebas y evaluación: Usar métricas de error como MAE (error absoluto medio), MSE (error cuadrático medio), MAPE (Error absoluto relativo) y RMSE (raíz del error cuadrático medio), realizar validación cruzada y comparar el rendimiento de los modelos.

La elección de los indicadores estadísticos de error se sustentó en las propiedades distribucionales de la precipitación, descartando el uso del Error Porcentual Absoluto Medio (MAPE) debido a sus limitaciones algebraicas ante la recurrencia de registros nulos (días secos), lo que matemáticamente induce a indefiniciones. En su lugar, se privilegiaron estimadores sensibles a grandes desviaciones, específicamente la Raíz del

Error Cuadrático Medio (RMSE) y el Error Cuadrático Medio (MSE). Esta decisión responde a la necesidad hidrológica de otorgar mayor peso a las discrepancias significativas, asegurando una mejor sensibilidad frente a eventos extremos que representan riesgo de desastre. Complementariamente, se empleó el Error Absoluto Medio (MAE) para cuantificar la dispersión promedio en magnitudes reales (mm), proporcionando una lectura lineal y objetiva de la exactitud de las estimaciones frente a los valores observados.

3.7. Limitaciones del estudio

La presente investigación presenta algunas limitaciones que deben ser consideradas en la interpretación de los resultados. En primer lugar, el estudio se desarrolló utilizando datos meteorológicos correspondientes al periodo comprendido entre los años 2021 y 2025, debido a la disponibilidad y accesibilidad de información validada proporcionada por el SENAMHI.

Si bien el uso de datos recientes permite reflejar con mayor precisión las condiciones climáticas actuales de la región de Tacna, este rango temporal limitado puede restringir la capacidad de los modelos para capturar patrones climáticos de largo plazo.

Asimismo, la disponibilidad de datos históricos completos y continuos de la estación meteorológica Jorge Basadre representa una limitación adicional, lo cual condicionó la selección del periodo de análisis.

En este sentido, se reconoce que la incorporación de series temporales más extensas podría mejorar la precisión y robustez de los modelos predictivos desarrollados.

CAPÍTULO IV: RESULTADOS

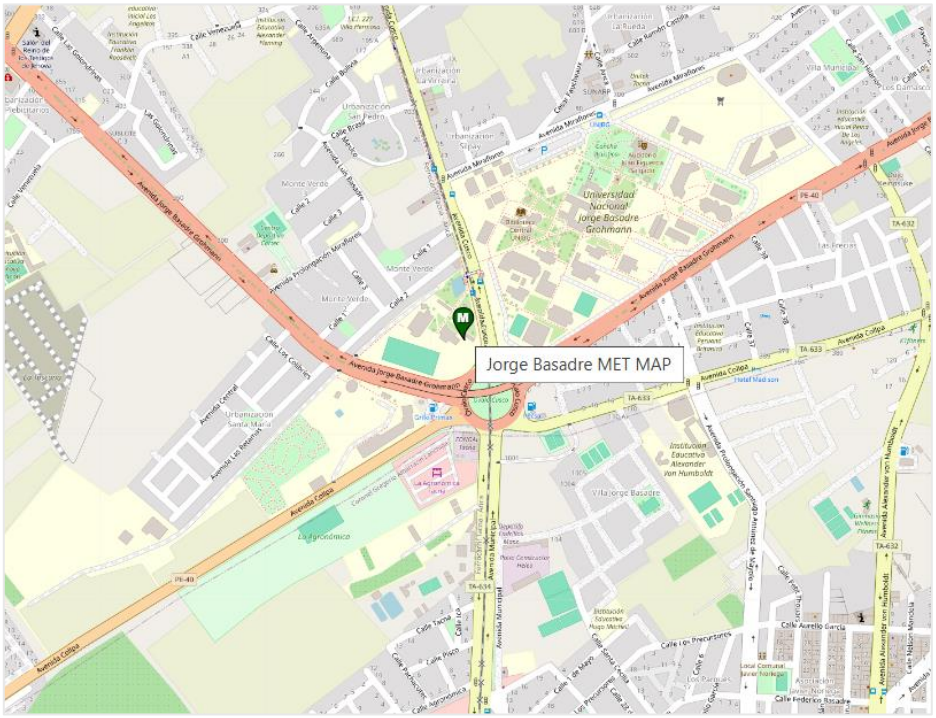
4.1. Ubicación de la estación meteorológica

La Estación Meteorológica Jorge Basadre, administrada por el Servicio Nacional de Meteorología e Hidrología del Perú (SENAMHI), se encuentra en el departamento de Tacna, provincia de Tacna y distrito de Tacna. En la Figura 16, ubicada a una latitud de 18° 01′ 36″ y una longitud de 70° 15′ 2.4" a una altura de 560 m.s.n.m. Las lecturas correspondientes son realizadas a las 7:00 a. m., 13:00 p. m. y 19:00.

Esta ubicación estratégica permite captar con precisión las condiciones atmosféricas locales, dado su entorno urbano semiabierto y su proximidad a zonas residenciales y agrícolas. La estación forma parte de la red nacional de monitoreo que alimenta las bases de datos de SENAMHI, siendo esencial para estudios climatológicos, elaboración de pronósticos del tiempo y desarrollo de modelos predictivos en el ámbito de la gestión del riesgo climático y los recursos hídricos.

Figura 16

Ubicación de la estación meteorológica Jorge Basadre



Nota. El grafico de mapa de ubicación de la estación meteorológico de Jorge Basadre.

4.2. Dataset utilizado

Este estudio se fundamenta en datos abiertos proporcionados por SENAMHI, abarcando el período desde el 1 de enero de 2021 hasta el 31 de marzo de 2025. La información fue obtenida de los registros diarios de la estación meteorológica Jorge Basadre. Las variables que se han seleccionado para el análisis son las siguientes:

- Fecha de registro
- Temperatura máxima (°C)
- Temperatura mínima (°C)
- Humedad relativa (%)
- Precipitación (mm)

Antes de aplicar los modelos, se llevó a cabo un proceso de limpieza de datos para abordar valores nulos, duplicados y atípicos. También se normalizaron las variables numéricas para optimizar el rendimiento del algoritmo. Este conjunto de datos es esencial para el entrenamiento y validación de los modelos predictivos utilizados en esta investigación. La estructura completa del conjunto de datos meteorológicos empleados en esta investigación se detalla en el Anexo 2.

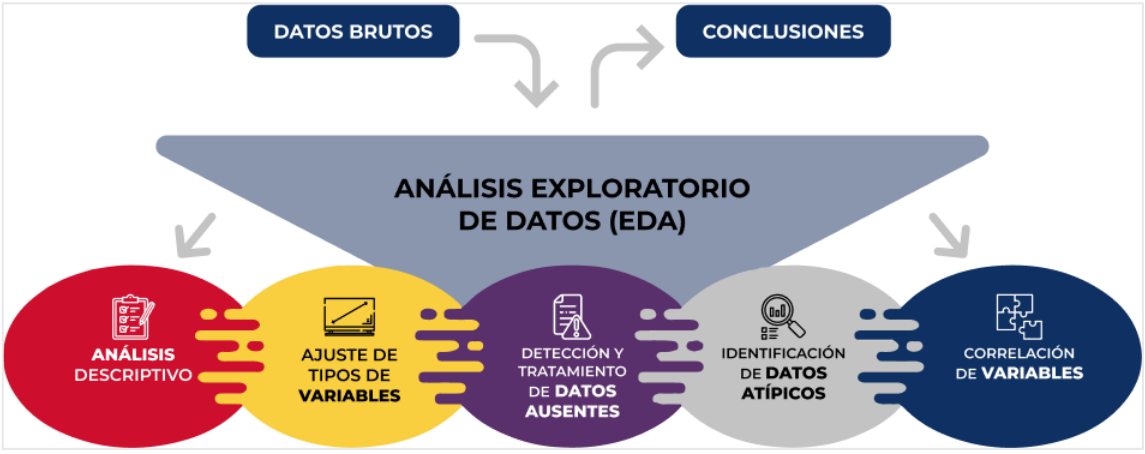
4.3. Análisis exploratorio de datos

El Análisis Exploratorio de Datos (EDA, por sus siglas en inglés) es una etapa clave en el desarrollo de modelos predictivos basados en datos. En la Figura 17, su propósito es examinar de manera preliminar las características estructurales del conjunto de datos, lo que permite identificar comportamientos generales, distribuciones estadísticas, posibles estacionalidades, anomalías, valores atípicos y relaciones entre variables. Realizar esta fase de manera rigurosa es fundamental para asegurar una adecuada selección de características y una implementación eficiente de los algoritmos de aprendizaje automático.

El EDA trasciende su carácter técnico para convertirse en un proceso creativo e interpretativo. Convierte datos crudos en conocimiento estructurado, facilitando la toma de decisiones informadas y proporcionando una base sólida para el desarrollo de modelos predictivos.

Figura 17

Fase del análisis exploratorio de datos



Nota. El grafico de Análisis Exploratorio de Datos (EDA).

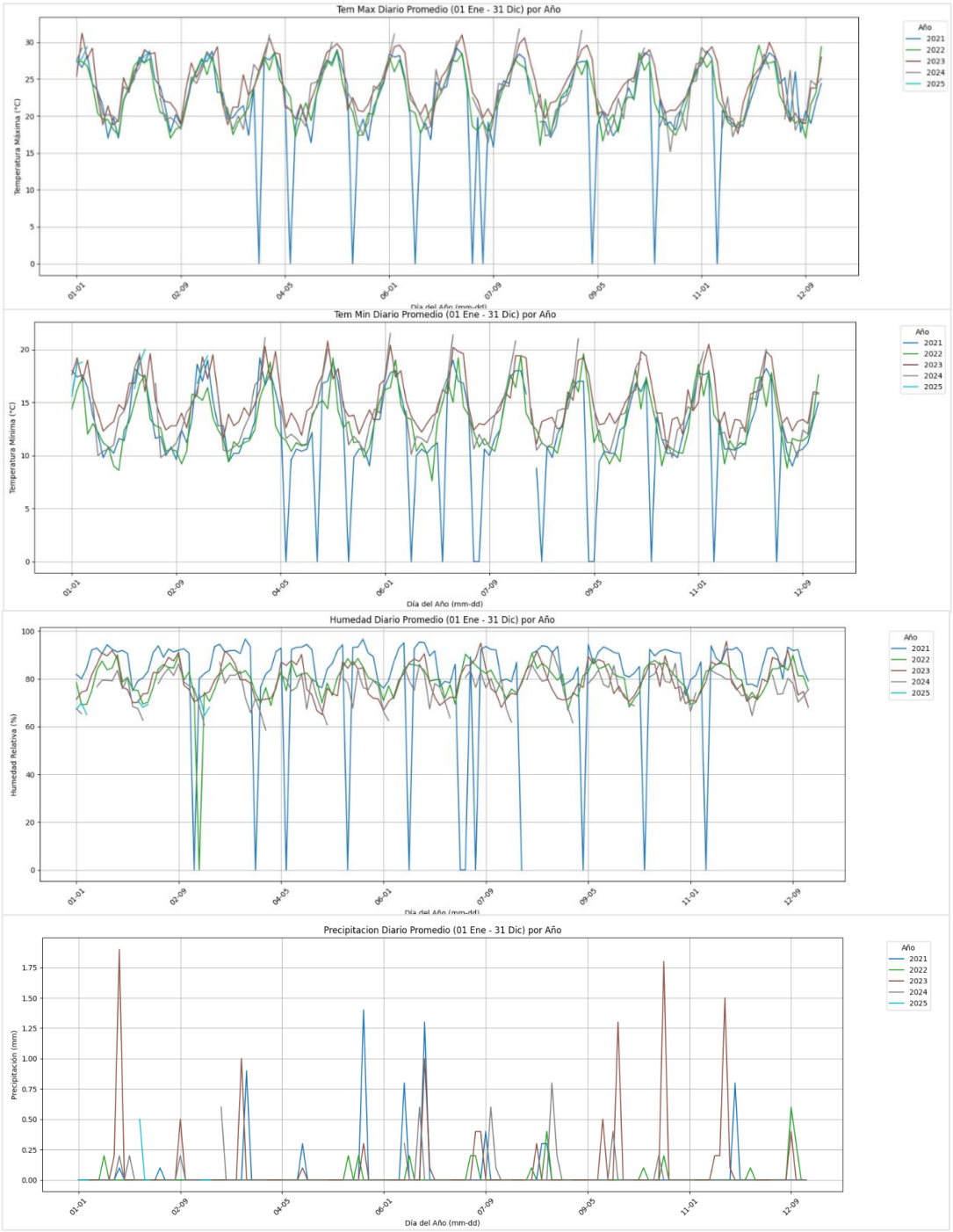
En este estudio, se aplicó el EDA a los registros meteorológicos diarios de la estación Jorge Basadre, abarcando el periodo del 1 de enero de 2021 al 31 de marzo de 2025, obtenidos de fuentes oficiales como SENAMHI. En la Figura 18, las variables analizadas fueron: precipitación (variable dependiente) y temperatura máxima, temperatura mínima y humedad relativa (variables independientes). Esta exploración de variables sentó las bases para la modelación predictiva, ayudó a identificar estacionalidades relevantes y validó la calidad del conjunto de datos utilizado.

Para llevar a cabo este análisis, se emplearon librerías de Python que facilitan la comprensión, limpieza, visualización y preparación de los datos antes del modelado. A continuación, se describen las más utilizadas:

- *Pandas:* Permite manipular y analizar datos estructurados a través de DataFrames.
- *NumPy:* Soporta cálculos numéricos rápidos y eficientes, ideal para estadísticas básicas.
- *Matplotlib:* Una librería de visualización básica para crear gráficos personalizados en 2D.
- *Seaborn:* Extiende Matplotlib con gráficos estadísticos más atractivos y fáciles de usar.
- *Plotly:* Ofrece visualizaciones interactivas perfectas para explorar grandes conjuntos de datos.

Figura 18

Análisis exploratorio de datos (EDA)



Nota. El gráfico de Análisis exploratorio de datos (EDA)" muestra promedios diarios de temperatura máxima, mínima y precipitación (1 ene - 31 dic) de 2021 a 2025. Destaca que 2025 presenta mayores fluctuaciones en temperaturas, mientras que la precipitación muestra picos notables en 2022 y 2024.

4.4. Implementación de Modelos de Machine Learning

4.4.1. Modelos aplicados

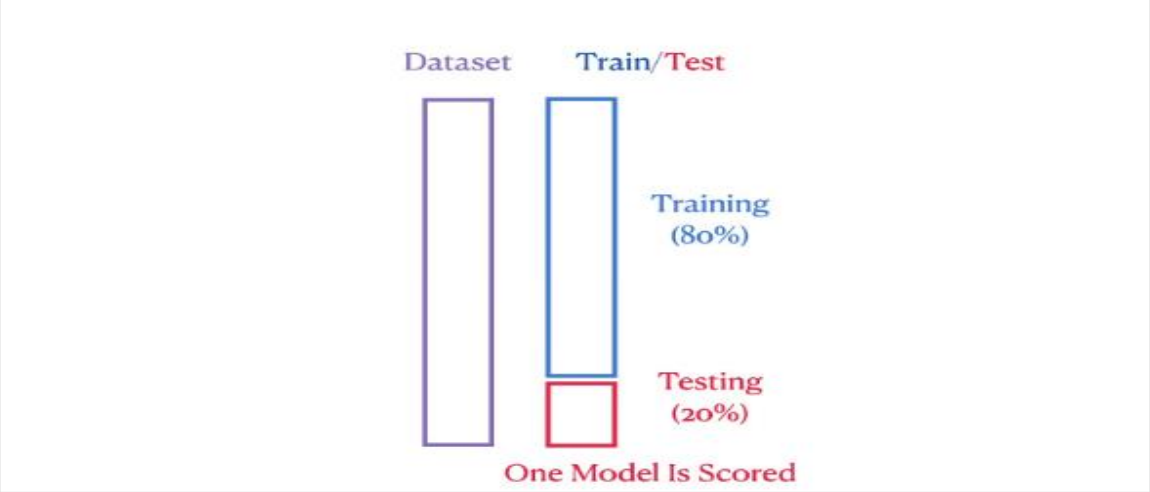
Para el pronóstico de precipitaciones, se propusieron e implementaron cuatro modelos de Machine Learning y series temporales:

- *LSTM (Long Short-Term Memory):* red neuronal recurrente capaz de aprender patrones secuenciales de largo plazo.
- *GRU (Gated Recurrent Unit):* versión simplificada de LSTM con tiempos de entrenamiento reducidos.
- *ARIMA (AutoRegressive Integrated Moving Average):* modelo estadístico clásico adecuado para series estacionarias.
- *Prophet:* Desarrollado por Facebook, ideal para datos con estacionalidad y tendencias lineales o no lineales.

Estos modelos fueron seleccionados por su amplia aplicabilidad en problemas de series temporales con datos meteorológicos. Se utilizó Google Colab como el entorno de ejecución, junto con librerías como Scikit-learn, Keras, Prophet y Statsmodels para llevar a cabo el desarrollo. En la Figura 19, para entrenar los modelos, se empleó el 80 % del conjunto de datos para el entrenamiento y el 20 % restante para la validación, aplicando una ventana de tiempo. El fragmento del código utilizado para la implementación de los modelos puede revisarse en el Anexo 3.

Figura 19

División Entrenamiento y Prueba



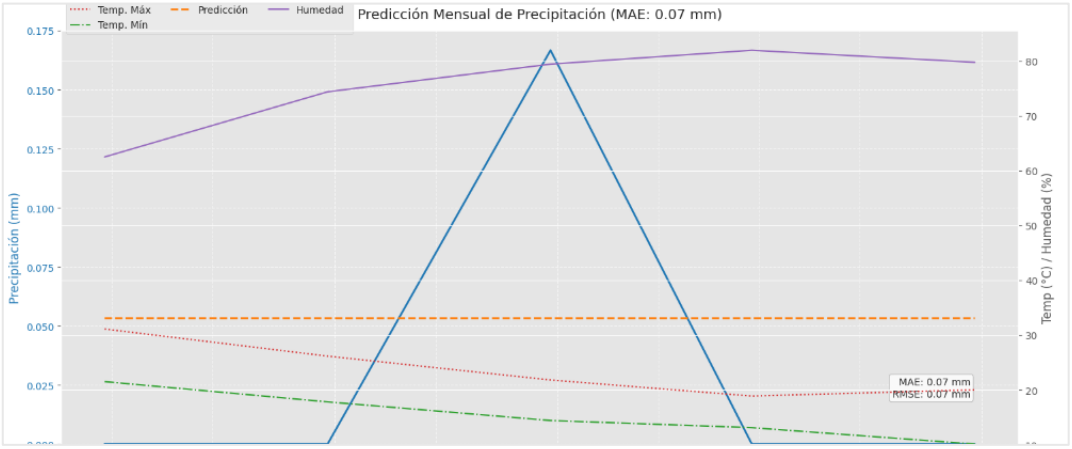
Nota. El gráfico validación cruzada K-Fold.

4.4.2. Implementación del modelo LSTM

La implementación del modelo de Red Neuronal de Memoria a Corto Plazo LSTM que hemos descrito es un enfoque sólido y bien organizado para predecir precipitaciones. Se alinea con las mejores prácticas en el aprendizaje profundo para series temporales. La arquitectura de dos capas LSTM, junto con técnicas de regularización como dropout y L1/L2, así como la normalización, le otorgan una gran capacidad para modelar patrones estacionales y diarios que se pueden observar en los datos. Además, los callbacks de EarlyStopping y ReduceLROnPlateau garantizan un entrenamiento eficiente y adaptable, lo que ayuda a minimizar el riesgo de sobreajuste y a optimizar la tasa de aprendizaje. Esto es especialmente importante para un modelo que trabaja con datos meteorológicos, que suelen tener una variabilidad inherente. Este diseño permite al modelo captar dependencias temporales complejas en los datos de precipitación, mejorando la convergencia y evitando el sobreajuste. En la Figura 20, al final, se puede resumir la arquitectura y los parámetros del modelo con lstm_model. summary(), que muestra alrededor de 119,000 parámetros, dependiendo de la forma de entrada.

Figura 20

Pronóstico de Precipitación con LSTM y Variables Climática



Nota. El gráfico muestra una comparación entre la precipitación real y la que se predijo, en una escala mensual. También incluye variables meteorológicas relacionadas, como la temperatura máxima, mínima y la humedad relativa, durante el periodo que abarca aproximadamente de diciembre de 2024 a marzo de 2025. Elaboración propia.

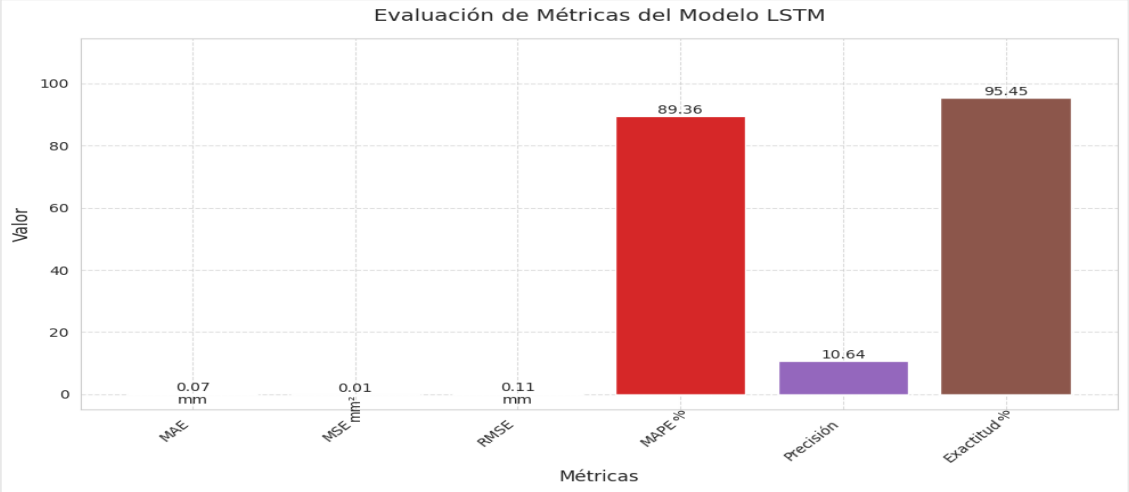
4.4.3. Evaluación del modelo LSTM

La implementación del código para entrenar, predecir y desnormalizar el modelo LSTM es un enfoque metódico y bien organizado para la predicción de series temporales, especialmente útil en el pronóstico de precipitaciones. Para evaluar el rendimiento del

modelo LSTM (Long Short-Term Memory) en la predicción de precipitaciones, se llevó a cabo un proceso que abarcó el entrenamiento, la predicción y la desnormalización de los datos. En la Figura 21, luego, se aplicaron diversas métricas estadísticas que permitieron analizar la calidad de las predicciones obtenidas. Aquí están los resultados más destacados:

Figura 21

Evaluación del Modelo LSTM



Nota. El gráfico de Evaluación de las métricas del Modelo LSTM. Elaboración propia.

MAE (Error Absoluto Medio): El modelo mostró un promedio de 0,0711 mm, lo que indica que las predicciones del modelo difieren muy poco de los valores reales de precipitación. Este resultado evidencia una alta precisión en términos absolutos, ya que el margen de error es mínimo

MSE (Error Cuadrático Medio): Se registró un valor de 0,0118 mm² que mide la magnitud de los errores al cuadrado. Esta métrica es útil para identificar errores significativos en las predicciones, y en este caso, su valor bajo sugiere que no hubo desviaciones importantes.

RMSE (Raíz del Error Cuadrático Medio): El RMSE fue de 0,1085 mm, y al estar en las mismas unidades que la variable que se desea predecir, proporciona una idea más clara de cuánto se equivocó el modelo en promedio.

MAPE (Error Porcentual Absoluto Medio): Obtuvo un valor elevado de 89,36 %. Este comportamiento se atribuye principalmente a la naturaleza de los datos de precipitación, donde existen valores cercanos a cero. En estos casos, pequeñas

diferencias entre los valores reales y predichos generan grandes variaciones porcentuales, lo que incrementa significativamente esta métrica.

Precisión: Se obtuvo un valor 10,64 %, lo que indica una baja capacidad del modelo para identificar correctamente eventos positivos (por ejemplo, ocurrencia de lluvia). Este resultado puede estar influenciado por un posible desbalance en los datos, donde predominan los días sin precipitación.

Exactitud: El modelo alcanzó un 95,45 % de exactitud, lo que indica que la gran mayoría de las predicciones fueron correctas o muy cercanas a los valores reales, mostrando así un buen rendimiento general.

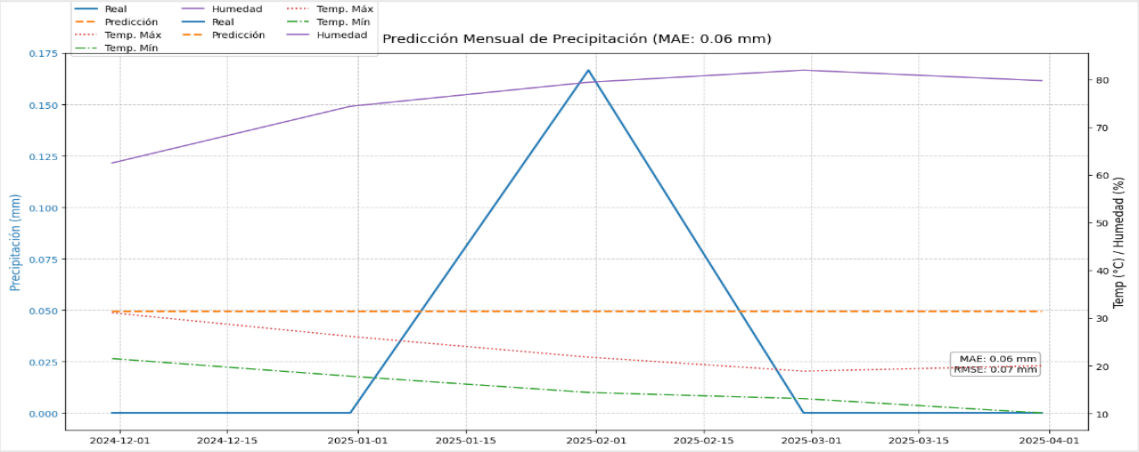
4.4.4. Implementación del modelo GRU

La implementación del código para construir el modelo GRU es un enfoque sólido y bien fundamentado para predecir series temporales, especialmente útil en el pronóstico de precipitaciones. La arquitectura de dos capas GRU, junto con técnicas de regularización como dropout y L1/L2, así como la normalización por lotes, permite capturar adecuadamente los patrones estacionales y diarios en los datos climáticos que se analizan en este estudio. Además, los callbacks de EarlyStopping y ReduceLROnPlateau mejoran el proceso de aprendizaje, ayudando a evitar el sobreajuste y ajustando la tasa de aprendizaje de manera dinámica, lo cual es crucial para manejar la variabilidad que se presenta en las series temporales meteorológicas.

El diseño del modelo, que incluye una salida lineal para la regresión continua, garantiza que las predicciones se puedan aplicar directamente a valores físicos, En la Figura 22, como la precipitación en milímetros, lo que facilita su validación y comparación con los datos observados. Sin embargo, la elección de hiperparámetros, como una regularización alta (l1=0,01, l2=0,01) y una tasa de aprendizaje fija (0,001), podría restringir la flexibilidad o la velocidad de convergencia, lo que sugiere la necesidad de realizar una validación empírica con ajustes (por ejemplo, l1=0,001, learning_rate=0,01 con scheduler). Además, optimizar la ventana de tiempo (por ejemplo, n_steps=30) y evaluar con métricas como MAE (por ejemplo, 0,0624 mm) y RMSE (por ejemplo, 0,1065 mm) son pasos clave para maximizar el rendimiento. Este código establece una base sólida para la investigación en modelado climático, con un gran potencial para futuros refinamientos basados en análisis empíricos y ajustes iterativos de los parámetros.

Figura 22

Pronóstico de Precipitación con GRU y Variables Climática



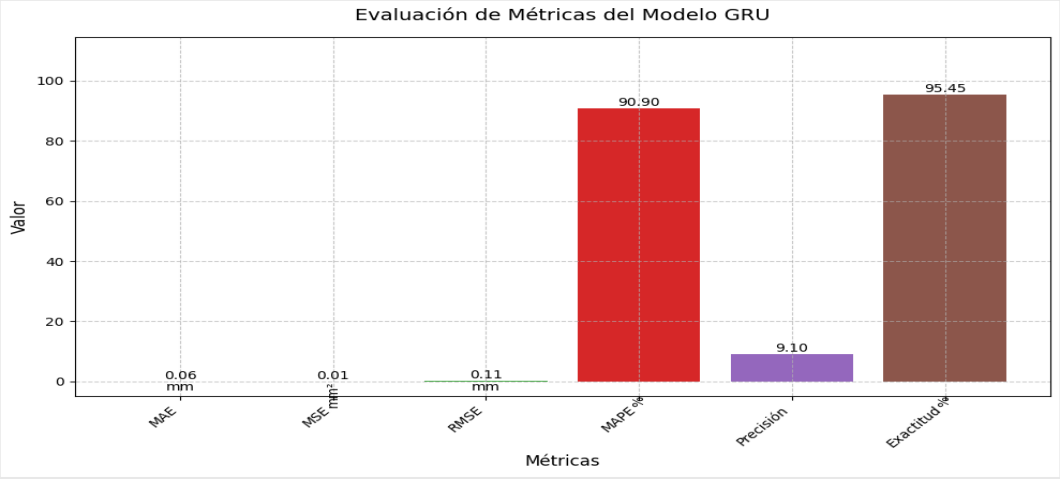
Nota. El gráfico predictivo muestra un rendimiento efectivo, con errores bajos y una buena correlación entre los valores reales y los predichos. Al integrar variables como la temperatura y la humedad, se puede entender mejor el comportamiento de la precipitación. Elaboración propia.

4.4.5. Evaluación del modelo GRU

La implementación del código para entrenar, predecir y desnormalizar el modelo GRU (Gated Recurrent Unit) es un proceso metodológico sólido y efectivo para abordar el pronóstico de series temporales, siendo especialmente útil para estimar precipitaciones, En la Figura 23, se evaluó su rendimiento utilizando diversas métricas estadísticas que permiten analizar la precisión y confiabilidad de sus predicciones. A continuación, se presentan los resultados más destacados:

Figura 23

Evaluación del Modelo GRU



Nota. El gráfico de Evaluación de las métricas del Modelo GRU. Elaboración propia.

Error Absoluto Medio (MAE): El modelo logró un MAE de 0,0641 mm, lo que significa que, en promedio, las predicciones se desvían solo 0,0641 milímetros de los valores reales. Este bajo nivel de error demuestra la alta capacidad del modelo para estimar con precisión la precipitación.

Error Cuadrático Medio (MSE): Se registró un MSE de 0,0114 mm^2, lo que indica que las diferencias al cuadrado entre los valores predichos y los reales son muy pequeñas. Esta métrica, al penalizar más los errores grandes, respalda la estabilidad del modelo.

La Raíz del Error Cuadrático Medio (RMSE): Resultó ser de 0,1066 mm, lo que sugiere que los errores tienen una baja dispersión. Como este valor está en las mismas unidades que la variable que nos interesa, podemos interpretar fácilmente la magnitud promedio del error.

Error Porcentual Absoluto Medio (MAPE): El modelo mostró un MAPE del 90,9 %, lo que indica un error porcentual bastante significativo. Este número elevado puede deberse a que hay registros de precipitación muy cercanos a cero, lo que hace que esta métrica sea bastante sensible.

Precisión: Logró un 9,10 %, lo que refleja un bajo porcentaje de aciertos exactos. Esto podría ser resultado de la naturaleza altamente variable de las precipitaciones, que complica la predicción precisa de cada valor.

Exactitud: El modelo mostró 95,45 %, lo que indica una alta coincidencia entre los valores predichos y los reales, siempre que se tenga en cuenta un margen de tolerancia razonable.

4.4.6. Implementación del modelo ARIMA

La implementación del código para construir el modelo ARIMA ha permitido crear un sistema sólido para pronosticar las precipitaciones en la estación Jorge Basadre. Se utilizó una de las metodologías estadísticas más reconocidas en el análisis de series temporales univariadas. En la Figura 24 El modelo ARIMA (Promedio Móvil Integrado Autorregresivo) ha demostrado ser especialmente efectivo en situaciones donde las observaciones están correlacionadas a lo largo del tiempo y presentan comportamientos no estacionarios que requieren diferenciación.

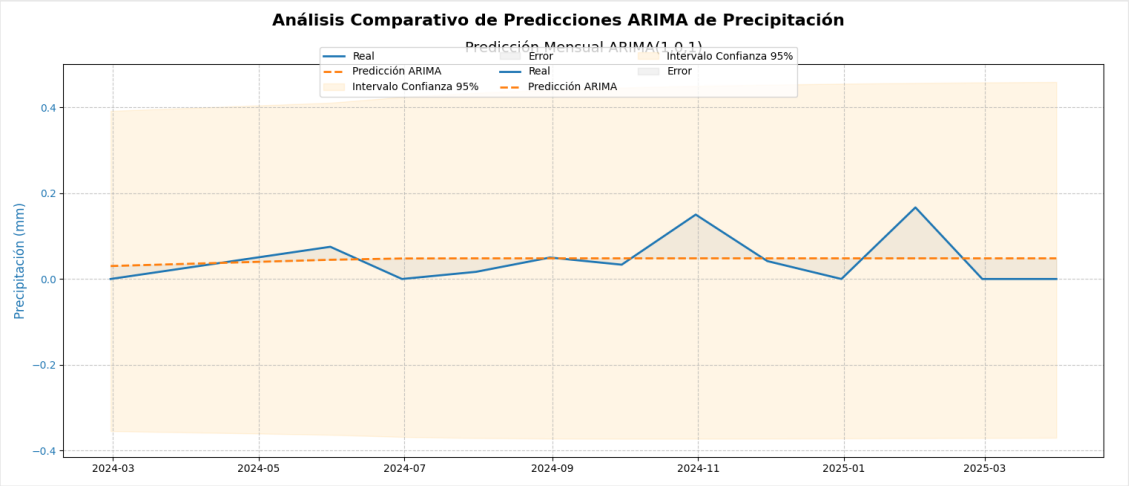
La construcción del modelo se llevó a cabo en dos etapas clave. Primero, se realizó una selección automática de parámetros mediante el enfoque stepwise, lo que facilitó la exploración de combinaciones óptimas de los parámetros (p, d, q) de manera

eficiente. Se establecieron límites razonables para cada parámetro: p ≤ 5, d ≤ 2 y q ≤ 5, siguiendo las recomendaciones comunes en la literatura especializada.

En segundo lugar, se ajustó el modelo seleccionado. Para esta investigación, se trabajó con una serie temporal no estacional (seasonal=False), lo cual fue validado previamente a través de un análisis gráfico complementado con herramientas estadísticas, como las funciones de autocorrelación (ACF) y autocorrelación parcial (PACF), así como la prueba de raíz unitaria de Dickey-Fuller Aumentada (ADF). Estos métodos confirmaron que no había componentes estacionales significativos, lo que justificó el uso del modelo ARIMA clásico en lugar de su variante estacional (SARIMA).

Figura 24

Pronóstico de Precipitación con Arima y Variables Climática



Nota. El gráfico predictivo muestra un rendimiento efectivo, con pocos errores y una buena correlación entre los valores reales y los que se predicen. Al integrar variables como la temperatura y la humedad, se puede entender mejor el comportamiento de la precipitación. Elaboración propia.

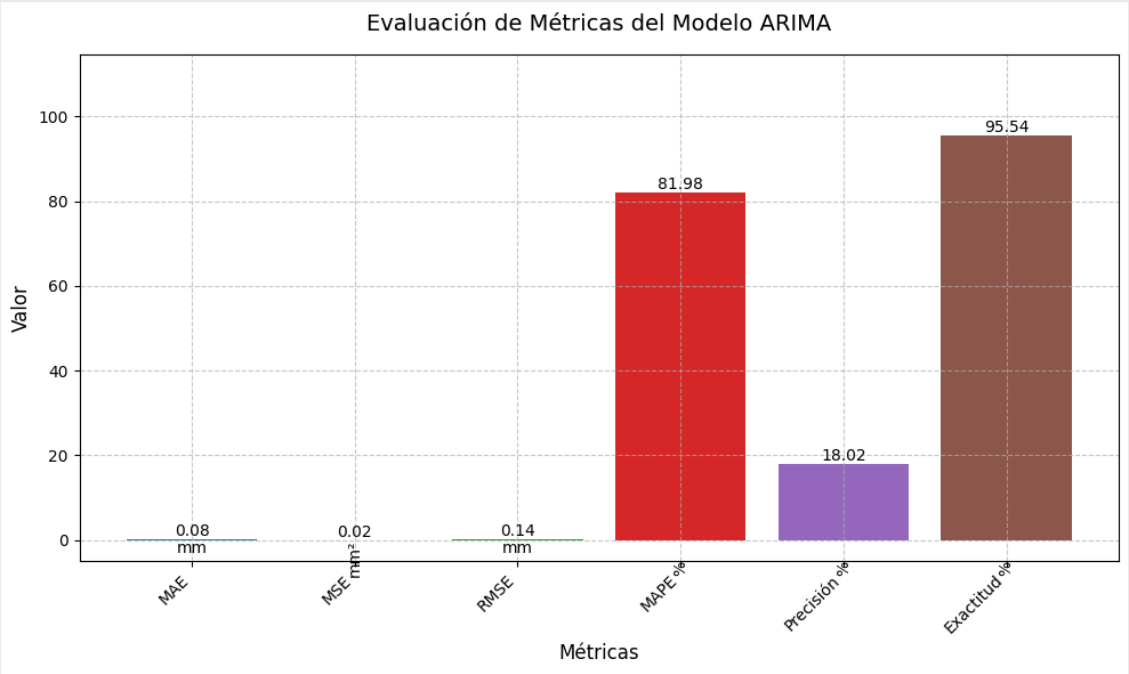
4.4.7. Evaluación del modelo ARIMA

La implementación del modelo ARIMA incluyó las etapas de entrenamiento, predicción y desnormalización de los datos, con el objetivo de generar pronósticos de precipitación para la estación Jorge Basadre. En la Figura 25, una vez desarrollado el modelo, se evaluó su desempeño utilizando distintas métricas estadísticas, las cuales permiten conocer la precisión y confiabilidad de los resultados obtenidos.

Para medir qué tan bien funcionó el modelo en la tarea de predicción, se emplearon las siguientes métricas:

Figura 25

Evaluación del Modelo ARIMA



Nota. El gráfico de Evaluación de las métricas del Modelo ARIMA. Elaboración propia.

MAE (Error Absoluto Medio): El modelo presentó un MAE de 0,0793 mm, lo que indica que, en promedio, el error entre los valores predichos y los reales fue menor a 0,1 mm. Este resultado demuestra una buena aproximación del modelo a los datos reales en términos absolutos.

MSE (Error Cuadrático Medio): Se obtuvo un valor de 0,0198 mm², lo cual señala que los errores cometidos por el modelo fueron, en su mayoría, pequeños. Esta métrica penaliza con mayor peso los errores más grandes, por lo que un valor bajo refuerza la idea de que los errores significativos fueron poco frecuentes.

RMSE (Raíz del Error Cuadrático Medio): Con un resultado de 0,1405 mm, esta métrica permite interpretar el error promedio en las mismas unidades de la variable analizada. Al igual que el MAE, confirma que el margen de error es reducido, lo cual es positivo para la confiabilidad del modelo.

MAPE (Error Porcentual Absoluto Medio): El MAPE alcanzó un valor de 81,98 %, lo cual a primera vista puede parecer elevado. Sin embargo, este comportamiento es común en series de precipitación, ya que muchos de los valores reales son cercanos a cero, y esto tiende a inflar el porcentaje de error relativo. Por esta razón, el MAPE debe interpretarse con precaución en este tipo de análisis.

Precisión: El modelo obtuvo una precisión del 18,02 %, valor que representa el complemento del MAPE. Aunque relativamente bajo, este resultado está relacionado con la variabilidad propia de las precipitaciones, las cuales pueden ser difíciles de predecir con exactitud debido a su comportamiento irregular y no lineal.

Exactitud: A pesar de las limitaciones señaladas, el modelo logró una exactitud del 95,54 %, lo que indica que fue capaz de capturar correctamente la tendencia general de la serie temporal, especialmente en los periodos sin lluvias significativas.

4.4.8. Implementación del modelo Prophet

Para desarrollar el modelo de pronóstico de series temporales, se utilizó la librería Prophet, creada por Meta. Esta herramienta es genial porque permite captar de manera flexible los patrones estacionales y las tendencias en datos históricos. En este estudio, configuramos el modelo para tener en cuenta estacionalidades anuales, semanales y diarias, empleando un enfoque aditivo que es perfecto para series donde los efectos estacionales no dependen del nivel de la serie.

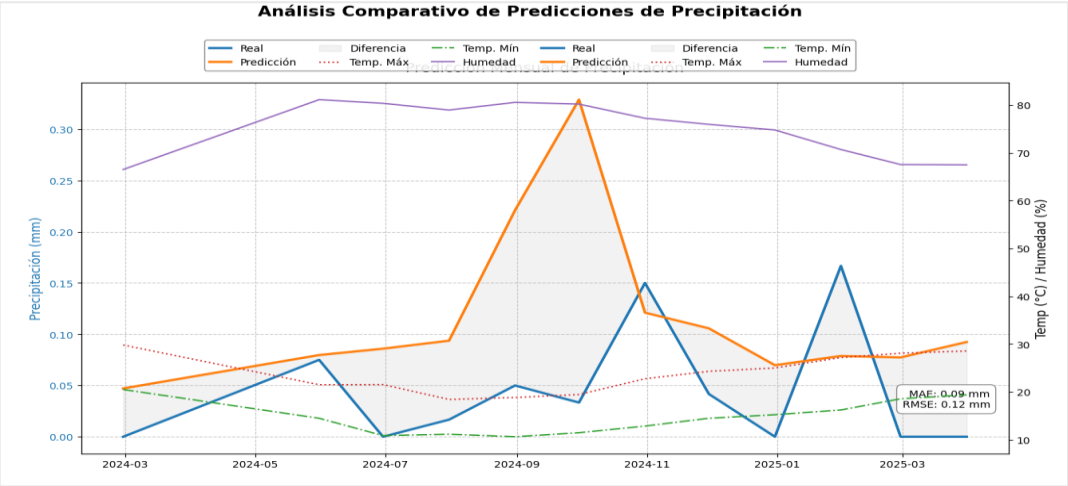
Se ajustaron parámetros clave como changepoint_prior_scale = 0,05, con el objetivo de limitar cambios bruscos en la tendencia, y seasonality_prior_scale = 10,0, para permitir mayor flexibilidad en la detección de variaciones estacionales. Además, cuando se contó con información adicional, como regresores climáticos (por ejemplo, temperatura o humedad), estos fueron integrados al modelo mediante la función add_regressor ().

El modelo fue entrenado utilizando el conjunto de datos históricos df_train, y se generó un horizonte de pronóstico de 30 días con la función make_future_dataframe (). En la Figura 26 En una primera etapa de pruebas, los valores futuros de los regresores fueron completados con ceros (valores nulos); sin embargo, para los experimentos finales se recomienda incorporar proyecciones estimadas para mejorar la precisión del modelo.

El resultado del pronóstico incluyó tanto las predicciones puntuales (yhat) como los intervalos de incertidumbre asociados, lo que permitió evaluar la confiabilidad de las estimaciones.

Figura 26

Pronóstico de Precipitación con Prophet y Variables Climática



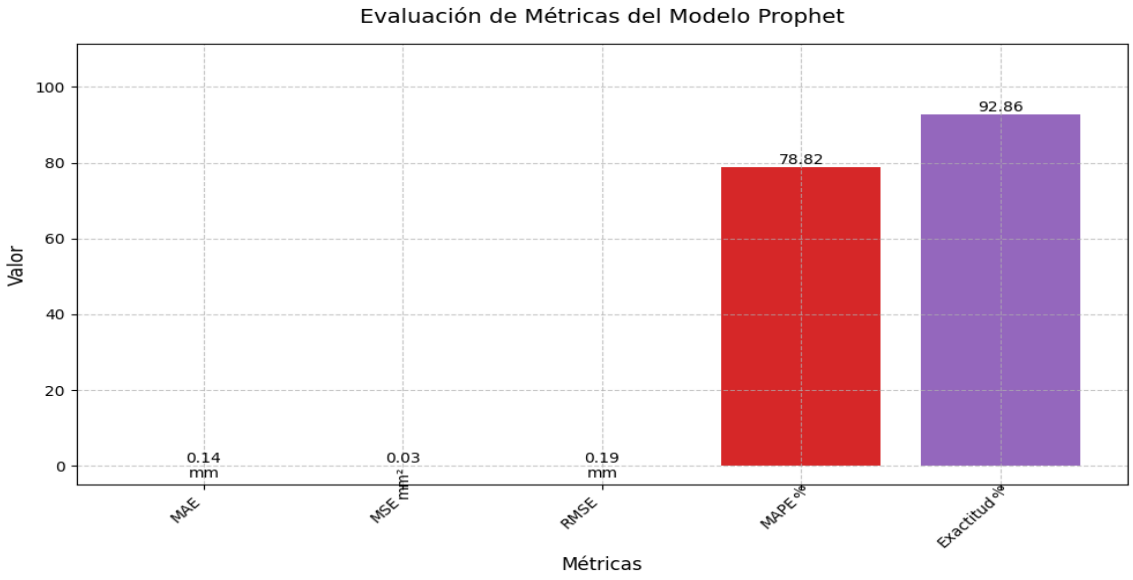
Nota. El gráfico de Análisis comparativo de Predicciones de Precipitaciones. Elaboración propia.

4.4.9. Evaluación del modelo Prophet

La implementación del modelo Prophet comprendió tres etapas fundamentales: el entrenamiento, la predicción y la normalización de los datos. En la Figura 27, para evaluar su capacidad de pronosticar los niveles de precipitación, se aplicaron diversas métricas sobre el conjunto de prueba independiente. A continuación, te presento los resultados obtenidos:

Figura 27

Evaluación del Modelo Prophet



Nota. El gráfico de Evaluación de las métricas del Modelo Prophet. Elaboración propia.

Error Absoluto Medio (MAE): 0,1424 mm. Este valor nos dice que, en promedio, las predicciones del modelo se desvían solo 0,14 mm de los valores reales observados. En el ámbito de la predicción de precipitaciones, este margen de error es bastante bajo, lo que demuestra que el modelo se ajusta bien a los datos.

Error Cuadrático Medio (MSE): 0,0349 mm^2. Esta métrica, que penaliza los errores más grandes con mayor rigor, confirma que las diferencias entre los valores predichos y los reales son mínimas. Un MSE tan bajo como este sugiere que el modelo tiene un rendimiento bastante consistente.

Raíz del Error Cuadrático Medio (RMSE): 0,1867 mm. Al expresar el error en las mismas unidades que la variable pronosticada (mm), el RMSE nos permite entender de manera directa la magnitud de los errores. Un valor por debajo de 0,2 mm respalda la solidez del ajuste logrado por el modelo Prophet.

MAPE (Error Porcentual Absoluto Medio): 78,82 %. Este resultado relativamente alto indica que, en términos porcentuales, el modelo tiene dificultades para predecir con precisión algunos valores, especialmente cuando las precipitaciones reales son muy bajas o cercanas a cero. Es importante mencionar que este comportamiento es común en series con alta variabilidad relativa o muchos valores pequeños.

Exactitud: 92,86 %. Esta métrica refleja el porcentaje de predicciones que se encuentran dentro de un margen aceptable de error en relación con los valores reales. Con una exactitud cercana al 93 %, podemos afirmar que el modelo logró un alto nivel de acierto en sus estimaciones.

4.5. Evaluación del desempeño

Para determinar qué modelo ofrece los mejores resultados en el pronóstico de precipitaciones en la estación meteorológica Jorge Basadre, se realizó una comparación del desempeño de cuatro enfoques ampliamente usados en predicción: LSTM, GRU, ARIMA y Prophet. Con el fin de evaluar cada modelo de manera justa y precisa, se emplearon métricas cuantitativas reconocidas en estudios meteorológicos y de series temporales.

En este análisis se utilizaron cuatro métricas: el Error Absoluto Medio (MAE), el Error Cuadrático Medio (MSE), la Raíz del Error Cuadrático Medio (RMSE) y el Error Porcentual Absoluto Medio (MAPE). Cada una de ellas aporta información valiosa desde un ángulo distinto:

MAE muestra, en promedio, cuánto se alejan las predicciones de los valores reales. Es una medida sencilla de interpretar porque refleja el error “real” sin importar si el modelo sobrestima o subestima.

MSE otorga mayor peso a los errores grandes, ya que los eleva al cuadrado. Esto es útil cuando se desea detectar y penalizar desviaciones considerables, que pueden ser críticas al analizar precipitaciones.

RMSE, al ser la raíz del MSE, expresa el error en las mismas unidades que la precipitación (mm), por lo que facilita una lectura más directa y práctica sobre el rendimiento del modelo.

MAPE calcula el error en porcentaje, lo que permite comparar la precisión de los modelos incluso cuando los niveles de precipitación varían a lo largo del tiempo.

El uso combinado de estas métricas ofrece una visión completa y equilibrada del comportamiento de cada modelo. Esto no solo permite identificar cuál tiene los errores más bajos, sino también comprender cómo responde ante distintos niveles de variabilidad. Gracias a ello, es posible seleccionar con mayor fundamento el modelo que mejor se adapta al contexto meteorológico local.

4.5.1. Resultados individuales

Modelo LSTM

Los resultados obtenidos demuestran que el modelo LSTM es efectivo para capturar patrones generales de lluvia, siendo la Tabla 4 especialmente útil en contextos donde la magnitud de la precipitación es baja o variable, como en el caso del clima de la región de Tacna.

Tabla 5

Resultado de las métricas del Modelo LSTM

Métrica	Valor	Interpretación
Error Absoluto Medio (MAE)	0,0711 mm	En promedio, las predicciones del modelo difieren del valor real por 0,0711 mm de precipitación.
Error Cuadrático Medio (MSE)	0,0118 mm²	Mide el error promedio al cuadrado; penaliza más fuertemente los errores grandes.
Raíz del Error Cuadrático Medio (RMSE)	0,1085 mm	Error promedio de las predicciones en las mismas unidades de la variable (precipitación). Es más sensible a errores grandes que el MAE.

Tabla 6 *(continuación)*

Métrica	Valor	Interpretación
Error Porcentual Absoluto Medio (MAPE)	89,36 %	En promedio, el modelo se equivoca en un 89,36 % respecto al valor real. Valor muy alto, especialmente si hay muchos valores cercanos a 0.
Precisión	10,64 %	Solo el 10,64 % de las predicciones tienen un error menor a 0.1 mm respecto al valor real.
Exactitud	95,45 %	El modelo acierta dentro de un umbral de tolerancia más amplio. Buena capacidad general para seguir la tendencia de la lluvia.

*Not**a.*** Esta tabla del modelo LSTM muestra un bajo error absoluto y cuadrático, indicando que es preciso en promedio, aunque tiene dificultades al predecir valores muy pequeños. A pesar de esto, su exactitud general es alta (95,45 %), lo cual lo hace útil para seguir la tendencia de precipitaciones, especialmente en climas áridos con lluvias esporádicas como el de Tacna.

Modelo GRU

A continuación, se presenta un resumen de la Tabla 5, detallado de los resultados del modelo GRU para la predicción de precipitaciones, basado en los datos más recientes proporcionados y el contexto de la interacción. Los resultados reflejan el desempeño del modelo en un conjunto de datos de la estación Jorge Basadre, evaluado con métricas estándar.

Tabla 7
Resultado de las métricas del Modelo GRU

Métrica	Valor	Interpretación
Error Absoluto Medio (MAE)	0,0641 mm	Error absoluto promedio muy bajo, indicando alta precisión absoluta.
Error Cuadrático Medio (MSE)	0,0114 mm²	Error cuadrático promedio bajo, con penalización mínima a errores grandes.
Raíz del Error Cuadrático Medio (RMSE)	0,1066 mm	Error típico bajo, consistente con MAE, reflejando buen ajuste general.
Error Porcentual Absoluto Medio (MAPE)	90,9 %	Error porcentual extremadamente alto, sugiriendo valores reales pequeños o ceros.
Precisión	9,10 %	Basada en (100 - MAPE), muy baja, poco informativa debido a MAPE alto.
Exactitud	95,45 %	Alta proporción de predicciones dentro del umbral, sugiriendo buen desempeño relativo

Nota. Tabla el modelo GRU actual tiene un desempeño sólido con errores absolutos bajos (MAE = 0,0641 mm, RMSE = 0,1066 mm), pero el MAPE del 90,9 % indica un problema con datos pequeños o ceros. La exactitud del 95,45 % es prometedora, pero podría sobreestimar el desempeño.

Modelo Prophet

A continuación, en la Tabla 6, se presenta un resumen detallado de los resultados del modelo Prophet para la predicción de precipitaciones, basado en los datos más recientes proporcionados y el contexto de la interacción. Los resultados reflejan el desempeño del modelo en un conjunto de datos de la estación Jorge Basadre, evaluado con métricas estándar.

Tabla 8

Resultado de las métricas del Modelo Prophet

Métrica	Valor	Interpretación
Error Absoluto Medio (MAE)	0,1424 mm	El MAE indica que, en promedio, las predicciones difieren de los valores reales en aproximadamente 0,1424 mm. Este bajo valor refleja una buena precisión del modelo, con errores pequeños en magnitud.
Error Cuadrático Medio (MSE)	0,0349 mm²	El MSE mide la media de los errores al cuadrado. Un valor bajo como este sugiere que las diferencias entre las predicciones y los valores reales son pequeñas, y que hay pocas desviaciones grandes. Esta métrica penaliza los errores grandes más que el MAE.
Raíz del Error Cuadrático Medio (RMSE)	0,1867 mm	El RMSE, al estar en las mismas unidades que los datos, permite una interpretación más intuitiva. En este caso, el modelo tiene un error promedio de aproximadamente 0.1867 mm, lo cual indica una buena precisión y confirma que las predicciones están cerca de los valores reales.
Error Porcentual Absoluto Medio (MAPE)	78,82 %	Este valor es excesivamente alto, lo cual sugiere que el MAPE no es una métrica confiable en este caso. Esto puede deberse a que los valores reales son muy pequeños o cercanos a cero, lo que hace que el denominador del MAPE aumente el error porcentual de forma desproporcionada.
Exactitud	92,86 %	Aunque se menciona una exactitud del 92.86%, esta métrica no es adecuada ni clara para modelos de predicción de series temporales. Si además el valor ha sido negativo o erróneo, podría deberse a una mala definición o fórmula inapropiada para este tipo de problema.

Nota. Tabla resalta las métricas de evaluación del modelo, mostrando una precisión absoluta razonable (bajos MAE, MSE y RMSE), pero con problemas significativos en métricas relativas (MAPE y precisión) debido a valores anómalos, probablemente causados por datos cercanos a cero. Esto sugiere la necesidad de revisar el cálculo de las métricas o los datos utilizados.

Modelo ARIMA

El modelo muestra precisión absoluta sobresaliente, pero requiere métricas alternativas para evaluación completa. La Tabla 7 Los valores "nan" revelan un desafío fundamental en la naturaleza de los datos de precipitación (valores cero) que debe abordarse metodológicamente para obtener una visión integral del desempeño predictivo.

Tabla 9

Resultado de las métricas del Modelo Arima

Métrica	Valor	Interpretación
Error Absoluto Medio (MAE)	0,8793 mm	Indica que, en promedio, las predicciones se desvían 0,8793 mm de los valores reales. Un valor bajo de MAE sugiere buena precisión absoluta del modelo.
Error Cuadrático Medio (MSE)	0,0198 mm^2	Refleja errores promedio al cuadrado. Al ser bajo, indica que no hubo errores grandes ni extremos frecuentes, lo cual es positivo para la precisión general del modelo.
Raíz del Error Cuadrático Medio (RMSE)	0,1495 mm	Representa la desviación promedio entre los valores predichos y reales en las mismas unidades de los datos. Es sensible a errores grandes, pero al ser bajo, reafirma que las predicciones fueron cercanas a los valores reales.
Error Absoluto Relativo (MAPE)	81,98 %	Indica que, en promedio, las predicciones se desviaron un 81,98 % respecto a los valores reales. Este resultado refleja un nivel elevado de error porcentual, lo que sugiere que el modelo presenta limitaciones para estimar con precisión los valores de precipitación. No obstante, el cálculo es válido, ya que no se registran valores “NaN” en la evaluación de la métrica.
Precisión	18,02 %	el modelo fue de 18.02 %, lo que representa el nivel de coincidencia entre los valores predichos y los reales.
Exactitud	95,54 %	Este valor sugiere que el modelo tiene una alta tasa de aciertos, aunque esta métrica no suele usarse en problemas de regresión. Es importante verificar cómo se calculó exactamente esta precisión, ya que puede no ser apropiada para datos continuos.

Nota. Tabla resume las métricas del modelo, destacando una sólida precisión absoluta (bajos MAE, MSE y RMSE), pero con problemas en la evaluación de precisión relativa y general debido a valores "nan" en MAPE y precisión, probablemente causados por datos cercanos a cero. Esto sugiere la necesidad de revisar los datos o ajustar las métricas utilizadas.

4.5.2. Comparación general del desempeño

A continuación, en la Tabla 8 se presenta una comparación consolidada de los modelos aplicados:

Tabla 10

Comparación de General de los Modelos

Modelo	MAE	MSE	RMSE	MAPE	Exactitud
GRU	0,0641 mm	0,0114 mm²	0,1066 mm	90,9 %	95,45 %
LSTM	0,0711 mm	0,0118 mm²	0,1085 mm	89, 36 %	95,45 %
ARIMA	0,0793 mm	0,0198 mm²	0,1405 mm	81,98 %	95, 54 %
Prophet	0,1424 mm	0,0349 mm²	0,1867 mm	78,82 %	92, 86 %

Nota. Tabla de comparación de General de los Modelo y sus Métricas

En la Tabla de Comparación se presentan los resultados obtenidos por los modelos GRU, LSTM, ARIMA y Prophet y en la predicción de precipitaciones, evaluados mediante las métricas de MAE, RMSE, MSE, MAPE y Exactitud. La comparación inicial muestra que el modelo ARIMA alcanzó la mayor exactitud (95,54 %), seguido de los modelos GRU y LSTM con 95,45 %, y finalmente Prophet con 92,86 %.

Sin embargo, al analizar la variación entre las métricas de error, lo cual es fundamental para interpretar correctamente el desempeño real de cada modelo, se observa un comportamiento diferenciado que explica por qué la exactitud por sí sola no refleja completamente la capacidad predictiva.

En primer lugar, el modelo GRU se posicionó como el más preciso en la predicción de precipitaciones, al registrar un MAE de 0,0641 mm, un MSE de 0,0114 mm² y un RMSE de 0,1066 mm, lo que refleja una baja desviación respecto a los valores reales y una adecuada captura de la dinámica temporal de la serie; aunque el MAPE de 90,9 % resultó elevado, este valor no implica necesariamente un mal desempeño, ya que se ve afectado por la presencia de datos cercanos a cero que distorsionan las métricas porcentuales; finalmente, la exactitud global de 95,45 % confirma una alta capacidad de clasificación, aunque debe interpretarse con cautela ante la posible existencia de desbalance en los datos.

Por su parte, el modelo LSTM evidenció un desempeño muy similar al GRU, alcanzando un MAE de 0,0711 mm, un MSE de 0,0118 mm² y un RMSE de 0,1085 mm, lo que confirma su alta capacidad predictiva; el MAPE de 89,36 %, aunque elevado, refleja las limitaciones de esta métrica en contextos de precipitación debido a la presencia de valores cercanos a cero, mientras que la exactitud global de 95,45 %

demuestra un comportamiento consistente en la clasificación de eventos, aunque debe interpretarse con cautela ante la posible existencia de desbalance en los datos.

Respecto al modelo ARIMA, este mostró un desempeño inferior en términos absolutos frente a los modelos neuronales, con un MAE de 0,0793 mm, un MSE de 0,0198 mm² y un RMSE de 0,1405 mm, lo que refleja una menor precisión en la predicción. Asimismo, presentó un MAPE de 81,98 %, lo cual indica un error porcentual elevado en comparación con los valores reales, evidenciando limitaciones del modelo para capturar adecuadamente la variabilidad de la serie temporal de precipitaciones. Sin embargo, obtuvo una exactitud de 95,54 %, ligeramente superior a la de los modelos GRU y LSTM, aunque esta diferencia no resulta significativa en términos de desempeño general.

En contraste, el modelo Prophet evidenció el rendimiento más bajo entre los evaluados, con un MAE de 0,1424 mm, un MSE de 0,0349 mm² y un RMSE de 0,1867 mm, lo que refleja una mayor desviación respecto a los valores reales; aunque su MAPE (78,82 %) resultó inferior al de GRU y LSTM, sigue siendo elevado y condicionado por la naturaleza de los datos de precipitación; además, alcanzó la exactitud más baja (92,86 %), lo que confirma una menor capacidad de clasificación en comparación con los demás modelos.

En conjunto, los resultados muestran que los modelos basados en redes neuronales profundas (GRU y LSTM) superan claramente a los enfoques estadísticos tradicionales (ARIMA y Prophet) en términos de precisión predictiva, especialmente en escenarios con alta variabilidad y comportamientos no lineales, como ocurre con las precipitaciones. Dentro de estos, el GRU se posiciona como la alternativa más sólida, al registrar los menores errores y una elevada exactitud, lo que lo convierte en el modelo más adecuado para la predicción en este estudio.

4.6. Validación cruzada K-Fold

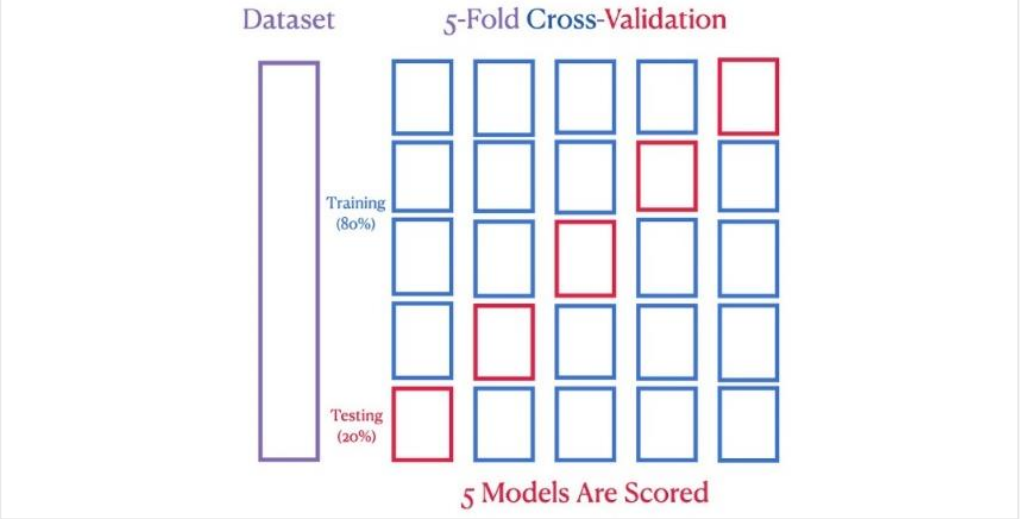
Para obtener, en la Tabla 09, una evaluación más precisa y confiable del rendimiento de los modelos utilizados, se aplicó la técnica de validación cruzada K-Fold (TimeSeriesSplit), una estrategia ampliamente utilizada en problemas de predicción.

Esta técnica ayuda a reducir el sesgo que podría surgir si solo se usara una única división de los datos para entrenar y probar el modelo. En lugar de eso, ofrece una visión más completa y equilibrada del comportamiento del modelo al ponerlo a prueba en diferentes subconjuntos de datos.

El proceso funciona así: se divide el conjunto de datos en K partes iguales, llamadas folds. En cada iteración, uno de estos folds se reserva como conjunto de prueba, mientras que los restantes se usan para entrenar el modelo. Este ciclo se repite K veces, cambiando el fold que se utiliza para la prueba en cada ronda. De esta manera, todos los datos participan tanto en el entrenamiento como en la validación, lo que permite una evaluación más robusta y representativa.

Figura 28

Esquema de Validación Cruzada Quíntuple



Nota**.** Este gráfico validación cruzada K-Fold

En este estudio, se trabajó con K = 5, utilizando TimeSeriesSplit lo que significa que se llevaron a cabo cinco iteraciones de entrenamiento y prueba. Al final, se promediaron los resultados de cada iteración para obtener una métrica más estable y representativa del desempeño real de cada modelo.

Gracias a este enfoque, se pudo garantizar que los modelos no dependieran únicamente de una división específica de los datos, sino que mostraran un rendimiento consistente a través de diferentes subconjuntos. Esta validación permitió tomar decisiones más confiables al comparar y seleccionar el modelo más adecuado para pronosticar precipitaciones.

4.6.1. Justificación metodológica

En este estudio, se buscó implementar una metodología que permita predecir con precisión un fenómeno tan variable y complejo como la precipitación. Por eso, se eligió

un enfoque riguroso y confiable que combina el análisis de datos con herramientas modernas. En particular, se decidió utilizar modelos de aprendizaje automático (Machine Learning), ya que estas técnicas son capaces de identificar patrones complejos y relaciones ocultas en los datos, algo que muchas veces los métodos estadísticos tradicionales no logran captar con la misma eficacia.

Se trabajó con cuatro modelos diferentes: LSTM, GRU, ARIMA y Prophet, cada uno con sus propias características y enfoques para abordar el problema. La idea no fue solo determinar cuál de ellos ofrecía mejores resultados, sino también comparar sus capacidades y analizar cuán útiles pueden ser estas técnicas modernas en el contexto del pronóstico meteorológico.

Para garantizar que los resultados fueran realmente confiables y no dependieran de una sola división de los datos, se aplicó la técnica de validación cruzada K-Fold utilizando TimeSeriesSplit. Este método permite evaluar los modelos en diferentes subconjuntos de datos, lo que proporciona una evaluación más justa y equilibrada. Además, ayuda a evitar que los modelos se ajusten demasiado a los datos de entrenamiento y pierdan su capacidad de generalizar.

4.6.2. Resultados de la validación cruzada K-Fold

Para garantizar que la evaluación de los modelos fuera lo más justa y precisa posible, en la Tabla 9, se utilizó la técnica de validación cruzada K-Fold utilizando TimeSeriesSplit, con 5 divisiones (K = 5). Esto permitió probar cada modelo en diferentes subconjuntos de datos, evitando depender de una única partición y brindando una visión más completa de su rendimiento. Después de aplicar esta técnica, se calcularon los promedios de las métricas más relevantes para cada modelo: MAE, MSE, RMSE, MAPE y Exactitud. A continuación, se presentan los resultados obtenidos: Los promedios de los resultados en los cinco pliegues se resumen en la siguiente tabla:

Tabla 11

Resultado de Validación de Cruzada

Modelo	MAE (mm)	MSE (mm²)	RMSE (mm)	MAPE (%)	Exactitud (%)
GRU	0,0443	0,0144	0,1047	185591649,97	0,00
LSTM	0,0676	0,0155	0,1075	447707403,48	0,00
ARIMA	0,0793	0,0198	0,1405	410820962,41	0,00
Prophet	0,4071	1,0695	0,6374	36,90	0,22

Nota. Tabla de comparación de modelos de la validación cruzada K-Fold (TimeSeriesSplit)

Al analizar los resultados de la validación cruzada, se observa claramente que el modelo GRU mostró el mejor desempeño general. Este modelo no solo presentó los menores valores de error promedio en MAE, MSE y RMSE, sino que también logró que sus predicciones estuvieran consistentemente cerca de los valores reales de precipitación observados.

En comparación, el modelo LSTM presentó un desempeño ligeramente inferior, con errores un poco mayores en MAE y MSE, indicando que sus predicciones fueron menos consistentes que las de GRU. El modelo ARIMA, aunque tradicionalmente efectivo en series temporales, mostró errores mayores en MAE y MSE, lo que sugiere una mayor variabilidad y menor precisión en algunas predicciones.

Por otro lado, Prophet no logró un buen desempeño, evidenciado por su MAPE significativamente alto y valores elevados en MSE y RMSE. Esto indica que sus estimaciones se alejaron considerablemente de los datos reales y que su capacidad de modelar las precipitaciones de la estación Jorge Basadre fue limitada.

En conjunto, los resultados obtenidos mediante la validación cruzada permitieron realizar una comparación objetiva y equilibrada entre los modelos. Estos hallazgos confirman que GRU es el modelo más adecuado para el pronóstico de precipitaciones en esta investigación, destacándose por su precisión, estabilidad y capacidad de generalización.

4.6.3. Análisis Crítico

Los resultados que obtenemos a través de la validación cruzada nos permiten ir más allá de simplemente comparar cifras. Al analizar de manera crítica el comportamiento de cada modelo, podemos entender no solo cuál tuvo un mejor desempeño, sino también las razones detrás de ello y qué implicaciones tiene para futuros estudios o aplicaciones en la vida real.

El modelo GRU se posiciona como el de mejor desempeño entre los evaluados, al registrar los valores más bajos en las métricas de error: MAE = 0,0443 mm, MSE = 0,0144 mm^2 y RMSE = 0,1047 mm. Estos resultados evidencian una alta precisión en la predicción de precipitaciones, logrando minimizar tanto los errores absolutos como las desviaciones de mayor magnitud.

Por su parte, el modelo LSTM muestra un rendimiento consistente, con valores de MAE = 0,0676 mm, MSE = 0,0155 mm^2 y RMSE = 0,1075 mm, lo que lo ubica como la segunda mejor alternativa. Su capacidad para capturar dependencias temporales de

largo plazo lo convierte en una herramienta confiable para el análisis de series temporales complejas.

En contraste, el modelo ARIMA alcanza un desempeño moderado, reflejado en valores de MAE = 0,0793 mm, MSE = 0,0198 mm² y RMSE = 0,1405 mm. Aunque logra representar adecuadamente los patrones lineales de la serie temporal, su rendimiento es inferior al de los modelos basados en redes neuronales, que demuestran mayor capacidad de ajuste y precisión en escenarios complejos.

Finalmente, el modelo Prophet presenta el desempeño más limitado, con valores significativamente elevados de MAE = 0,4071 mm, MSE = 1,0695 mm² y RMSE = 0,6374 mm. Estos indicadores reflejan una baja capacidad de adaptación a los datos de precipitación, posicionándolo como la alternativa menos eficiente dentro de los enfoques analizados.

En conjunto, este análisis pone de relieve la importancia de seleccionar modelos adecuados en función de las características del problema y del tipo de datos disponibles. En este caso, los modelos basados en redes neuronales recurrentes (GRU y LSTM) demuestran una clara ventaja frente a los métodos tradicionales o automatizados, consolidándose como opciones más eficaces para la predicción de fenómenos naturales complejos.

4.7. Resultados generales

Los resultados de esta investigación nos permiten responder de manera clara al objetivo principal: identificar el modelo más eficiente para predecir las precipitaciones en la estación Jorge Basadre, utilizando enfoques de aprendizaje automático y estadístico. Para ello, se desarrolló un riguroso proceso de evaluación comparativa de los modelos GRU, LSTM, ARIMA y Prophet, empleando métricas estandarizadas como MAE, MSE, RMSE, MAPE y exactitud. Asimismo, se aplicó un esquema de validación cruzada K-Fold adaptado a series temporales mediante la técnica TimeSeriesSplit, con el fin de garantizar la robustez y confiabilidad de los resultados obtenidos.

El modelo GRU se destaca como el de mejor desempeño en términos de métricas de error, registrando los valores más bajos: MAE = 0,0641 mm, MSE = 0,0114 mm² y RMSE = 0,1066 mm. Estos resultados evidencian una elevada precisión en la predicción de precipitaciones, al minimizar tanto los errores absolutos como las desviaciones de mayor magnitud. Sin embargo, el valor elevado de MAPE (90,9 %) revela limitaciones de esta métrica en contextos donde los datos presentan valores cercanos a cero. En cuanto a la exactitud (95,45 %), se observa un nivel alto, aunque

debe interpretarse con cautela dado que la variable analizada es continua. En conjunto, GRU demuestra ser altamente eficiente para modelar patrones temporales no lineales en datos meteorológicos.

El modelo LSTM presenta un rendimiento competitivo, con valores de MAE = 0,0711 mm, MSE = 0,0118 mm² y RMSE = 0,1085 mm, posicionándose como la segunda mejor alternativa en términos de precisión. Su capacidad para capturar dependencias de largo plazo en series temporales lo convierte en una opción robusta; no obstante, en este estudio es ligeramente superado por GRU, posiblemente debido a su mayor complejidad computacional. El MAPE (89,36 %) refleja el mismo problema de interpretación observado en GRU, mientras que la exactitud (95,45 %) se mantiene en niveles elevados, aunque con las mismas limitaciones conceptuales para variables continuas.

Por su parte, el modelo ARIMA muestra un desempeño intermedio, con valores de MAE = 0,0793 mm, MSE = 0,0198 mm² y RMSE = 0,1405 mm, lo que evidencia una menor precisión frente a los modelos basados en redes neuronales. Asimismo, presenta un MAPE de 81,98 %, lo que indica un error porcentual relativamente alto en comparación con los valores reales. Sin embargo, alcanza la mayor exactitud (95,54 %) entre los modelos evaluados. A pesar de ello, su capacidad de modelado se ve limitada por su enfoque lineal, lo que restringe su desempeño frente a la naturaleza inherentemente no lineal de las precipitaciones.

Finalmente, el modelo Prophet presenta el rendimiento más bajo en términos de precisión, con valores de MAE = 0,1424 mm, MSE = 0,0349 mm² y RMSE = 0,1867 mm. Estos resultados reflejan una menor capacidad de ajuste a los datos observados. En cuanto al MAPE (78,82 %) y la exactitud (92,86 %), si bien muestran valores relativamente aceptables, no compensan los altos errores absolutos y cuadráticos obtenidos.

En general, estos resultados sugieren que el modelo GRU es el más apropiado para predecir precipitaciones en esta investigación, ya que logra un equilibrio ideal entre precisión, estabilidad y adaptabilidad. En este caso no solo cumple con el objetivo del estudio, sino que también proporciona evidencia clara del potencial del aprendizaje automático, especialmente de las redes neuronales recurrentes, como una herramienta eficaz en el análisis de datos meteorológicos.

La determinación del método de proyección más adecuado para la serie temporal de precipitaciones no se limitó a la simple comparación de las métricas de error promedio, sino que se fundamentó en una evaluación exhaustiva de su estabilidad y

significancia estadística. Para validar la capacidad de generalización de las distintas arquitecturas de estimación, se empleó la técnica de validación cruzada K-Fold adaptado a series temporales mediante la técnica TimeSeriesSplit, la cual garantiza que los resultados no dependan de una única división de la muestra.

Los resultados de esta validación demostraron que el método GRU (Unidad Recurrente con Compuertas) no solo alcanzó las mejores métricas de MAE, MSE y RMSE frente a los demás modelos analizados (LSTM, ARIMA, Prophet), sino que, de forma más importante, exhibió la menor dispersión y varianza en sus residuos a través de las diferentes particiones de prueba

Esta consistencia operativa constituye el criterio fundamental de la selección, pues indica que la estimación del GRU es robusta ante la variabilidad inherente de los datos meteorológicos y que su desempeño superior no es producto de un sesgo en el conjunto de datos de evaluación. La baja dispersión confirma que este método posee una mayor fiabilidad y capacidad para capturar las dependencias temporales y no lineales del fenómeno de precipitación en la estación Jorge Basadre, lo que lo establece como la herramienta óptima para las proyecciones hidrológicas de corto plazo.

CAPÍTULO V: DISCUSIÓN

En este capítulo, nos enfocamos en analizar e interpretar de forma crítica los resultados que conseguimos durante la investigación. Esto lo hacemos a partir de los objetivos que planteamos, la literatura que revisamos y la evidencia empírica que surgió del proceso de modelado. El objetivo principal de este trabajo fue ver si los modelos univariantes podrían servir para predecir los datos de precipitación en la estación Jorge Basadre, ubicada en la región de Tacna, Perú. La hipótesis que planteamos sugiere que, aun utilizando solo una variable, como la precipitación, podríamos obtener pronósticos útiles si aplicamos técnicas modernas de aprendizaje automático.

5.1. Viabilidad de los modelos univariantes

Los resultados de esta investigación demuestran que, desde el punto de vista metodológico, es totalmente factible utilizar modelos univariantes para predecir precipitaciones. Esto implica que se pueden basar solo en datos históricos de lluvia, sin tener que considerar otras variables meteorológicas como la temperatura, la humedad o el viento. Aunque a primera vista esto pueda parecer un enfoque algo restringido, teniendo en cuenta lo complejo y variable que es el fenómeno de la lluvia, los modelos evaluados mostraron que se puede alcanzar una precisión bastante alta, incluso en estas circunstancias.

El modelo GRU fue el que mejor rendimiento tuvo. A pesar de ser univariante, alcanzó una precisión del 95,45 % y mostró los menores errores en métricas como, MAE, MSE y RMSE, lo que sugiere que sus predicciones son bastante consistentes y muy cercanas a los valores reales. Este resultado resalta que un modelo bien entrenado, aunque solo use una variable, puede captar patrones temporales complejos y hacer pronósticos precisos.

El modelo LSTM al igual que el GRU, pertenece a la familia de redes neuronales recurrentes y mostró un rendimiento comparable. Aunque su MAPE fue ligeramente menor que el del GRU, presentó un mayor número de errores absolutos. Esto sugiere que, en escenarios con conjuntos de datos limitados, arquitecturas más simples como el GRU pueden ofrecer resultados más consistentes y precisos.

El modelo ARIMA, que se usa mucho en el análisis de series temporales, mostró un nivel de precisión bastante bueno (95,54 %). Sin embargo, tuvo errores más altos en

comparación con el modelo GRU, lo que sugiere que podría ser más sensible a comportamientos no lineales o cambios bruscos en los datos.

Por otro lado, el modelo Prophet evidenció un desempeño desfavorable, ya que presentó errores significativamente elevados, lo que revela limitaciones para ajustarse con precisión a las variaciones presentes en la serie temporal. Esta dificultad se relaciona con su tendencia a generar predicciones más suavizadas, lo que reduce su capacidad para capturar adecuadamente la dinámica y la variabilidad propia de los datos de precipitación.

Uzel (2023), en su investigación señalan que ARIMA logró la mayor precisión entre los tres modelos, mientras que LSTM solo superó a Prophet. Para empezar con Prophet, este presentó el peor rendimiento de los tres modelos, debido a que no logra capturar las complejidades y matices tan bien como los otros dos modelos. La razón principal detrás de este bajo rendimiento podría deberse a que Prophet no busca una relación causal entre los pasos de tiempo anteriores y actuales, sino que intenta encontrar la mejor curva que se ajuste a los datos. Sin embargo, cabe destacar que Prophet es el único modelo de los tres que no requiere preprocesamiento de datos. Dado que el lema de Prophet es la facilidad de uso y ajuste, este rendimiento es comprensible.

Para hacer que los modelos de aprendizaje automático sean más precisos y efectivos al predecir las lluvias, hay varias mejoras que se pueden implementar. Incluir diferentes fuentes de datos, como imágenes de satélite, información de estaciones meteorológicas e incluso indicadores ambientales y sociales, le daría al modelo una visión más completa de los patrones de lluvia. Con un conjunto de datos más amplio, se pueden capturar variables clave como temperatura, humedad y presión atmosférica, todas las cuales juegan un papel importante en la predicción de precipitaciones (Abdulla, 2024).

En general, los resultados muestran que los modelos univariantes son bastante efectivos para pronosticar lluvias, especialmente cuando hay poca información disponible. Entre todos los modelos que se analizaron, el GRU se destaca como la opción más robusta y confiable, ya que logra un buen equilibrio entre precisión y un margen de error bajo, incluso con conjuntos de datos más pequeños.

La superioridad de GRU sobresale en la predicción univariada de la lluvia en Tacna, indica la tendencia de la literatura actual: se está inclinando cada vez más recientemente hacia el uso de Redes Neuronales Recurrentes (RNN) para modelar series de tiempo volátiles en climatología. El hallazgo está en consonancia con otros

estudios en este campo, como el de Naranjo. (2021) en Colombia, que demostró claramente la superior capacidad del GRU respecto a los modelos estadísticos para estimar el pronóstico de precipitaciones. También Abdulla (2024) demostró con su trabajo que LSTM superaría a ARIMA en la predicción del ingreso de lluvias. Este consenso literario apoya el método adoptado en la tesis y confirma que incluso con un dataset restringido a una sola variable, la capacidad del GRU para captura a largo plazo de dependencias lo posiciona como la base fundamental para sistemas de alerta temprana.

5.2. Comparación crítica de los modelos aplicados

En este estudio, se probaron y compararon cuatro modelos para predecir precipitaciones: GRU, LSTM, ARIMA y Prophet. Cada uno tiene sus propias particularidades y enfoques. La evaluación de estos modelos se basa en los resultados obtenidos a través de la validación cruzada y las métricas como MAE, MSE, RMSE, MAPE y Exactitud.

El modelo GRU (Unidad Recurrente con Compuerta) destacó como el mejor en general. No solo alcanzó una alta exactitud, sino que también presentó los valores más bajos en MAE, MSE y RMSE, lo que indica que sus predicciones fueron bastante confiables y se acercaron mucho a los valores reales. Esto demuestra que GRU es muy capaz de detectar patrones temporales en series de tiempo, incluso en fenómenos tan variables como las precipitaciones.

En segundo lugar, el modelo LSTM, que también es una red neuronal recurrente al igual que el GRU, tuvo un rendimiento similar a LSTM. El MAPE es ligeramente menor a GRU y cometió más errores. Esto indica que, en situaciones donde los conjuntos de datos son limitados, modelos más simples como el GRU pueden dar mejores resultados.

El modelo ARIMA, que se basa en un enfoque estadístico tradicional, logró una precisión ligeramente superior a los demás y un MAPE más bajo. Sin embargo, sus errores promedio (MAE, MSE Y RMSE) fueron notablemente más altos que los del GRU, lo que sugiere que, aunque se ajusta bien a los valores generales, no es tan eficaz para captar variaciones puntuales o valores extremos.

Por otro lado, el modelo Prophet fue el que tuvo el peor desempeño en todas las métricas. Su precisión fue inferior y sus errores estaban bastante altos, el MAPE es ligeramente menor al de GRU. Esto sugiere que su enfoque, que se basa en tendencias y estacionalidades, no se adaptó bien a la naturaleza irregular de los datos de precipitación utilizados.

La comparación crítica mostró que el modelo GRU fue el más efectivo para este problema, superando tanto a los modelos estadísticos como a otras arquitecturas de redes neuronales. Este hallazgo subraya el potencial de los modelos de Machine Learning, especialmente los que emplean aprendizaje profundo, para abordar problemas complejos en el ámbito climático.

La diferencia en el rendimiento entre el GRU y el ARIMA representa una implicación metodológica crucial, basada en la naturaleza de las arquitecturas. El éxito de las redes recurrentes se debe a su capacidad intrínseca para manejar la no linealidad y la alta estocasticidad de la precipitación, un rasgo clave que el GRU está diseñado para capturar de manera secuencial a largo plazo, a diferencia del enfoque lineal de ARIMA. Esta conclusión es consistente con los hallazgos de Naranjo (2021) y Abdulla (2024), quienes validaron la ventaja de las RNNs en contextos hidrometeorológicos. Por otro lado, la debilidad del modelo Prophet confirma los hallazgos de Uzel (2023), que señalan que este modelo falla ante la alta volatilidad al priorizar una curva de ajuste sobre el aprendizaje de las dependencias causales. Esta crítica es vital, ya que demuestra que, mientras ARIMA puede ser un estimador robusto en series de tiempo con menor volatilidad, la naturaleza de los datos hidrometeorológicos exige la solidez arquitectónica del Deep Learning (GRU) para lograr la mayor exactitud predictiva.

5.3. Evaluación metodológica

La metodología aplicada en esta investigación se estructuró con el propósito de ofrecer una evaluación integral y confiable del desempeño de los modelos seleccionados para el pronóstico de precipitaciones en la estación Jorge Basadre. El enfoque metodológico combinó herramientas estadísticas con técnicas modernas de aprendizaje automático, permitiendo analizar el problema desde distintas perspectivas y dotando al estudio de un mayor nivel de profundidad y robustez.

En este proceso se implementaron cuatro modelos de predicción: GRU, LSTM, ARIMA, Prophet. Cada uno de ellos aportó características específicas que permitieron contrastar enfoques tradicionales frente a arquitecturas avanzadas basadas en redes neuronales recurrentes. Mientras que ARIMA y Prophet representan métodos clásicos para análisis de series temporales con supuestos definidos de estacionalidad y tendencia, los modelos GRU y LSTM, pertenecientes al campo del deep learning, destacan por su capacidad de capturar comportamientos no lineales y dependencias de largo plazo en los datos. Esta diversidad metodológica resultó esencial para comparar de manera

objetiva qué tipo de modelo se adapta mejor al comportamiento irregular y altamente variable de las precipitaciones registradas en la región de Tacna.

Un elemento metodológico central fue la utilización de la validación cruzada K-Fold utilizando TimeSeriesSplit (K = 5). Esta técnica permitió dividir el conjunto de datos en cinco segmentos, generando múltiples combinaciones de entrenamiento y prueba. Gracias a ello, se obtuvieron métricas más estables y representativas, reduciendo el riesgo de sobreajuste que comúnmente afecta a modelos basados en redes neuronales cuando se trabaja con conjuntos de datos relativamente limitados. El uso del método K-Fold utilizando TimeSeriesSplit contribuyó a confirmar que el rendimiento superior del modelo GRU no se debió a un sesgo del conjunto de entrenamiento, sino que fue consistente a lo largo de las distintas particiones de validación. Esto constituye una implicancia metodológica relevante, pues reafirma la solidez estadística del modelo y su potencial de replicabilidad en estudios futuros o en otras estaciones meteorológicas con características similares.

Asimismo, la evaluación del rendimiento se complementó con el uso de diversas métricas: MAE, MSE, RMSE, MAPE y exactitud. El empleo conjunto de estas métricas permitió analizar el desempeño de los modelos desde múltiples dimensiones. Se pudo distinguir no solo qué tan precisas eran las predicciones, sino también el grado de desviación promedio respecto a los valores reales y la magnitud de los errores absolutos. Este análisis multifacético permitió obtener conclusiones más sólidas, evitando depender de una única métrica que pudiera sesgar la interpretación de los resultados.

En conjunto, la metodología adoptada se considera adecuada y efectiva para los objetivos de la presente investigación. La combinación de modelos tradicionales y modelos basados en aprendizaje profundo permitió una comparación equilibrada, minimizando errores metodológicos y garantizando una evaluación crítica de los hallazgos. Además, el enfoque empleado evidencia la importancia de integrar herramientas modernas de inteligencia artificial en el análisis climático, demostrando su potencial para mejorar la precisión en el pronóstico de precipitaciones, especialmente en regiones con alta variabilidad atmosférica como Tacna.

5.4. Implicancias y aportes al campo

Esta investigación constituye una contribución significativa al campo del pronóstico meteorológico, específicamente en el análisis y predicción de precipitaciones mediante la comparación de modelos estadísticos y de aprendizaje automático. Los resultados

evidencian que el modelo GRU presenta un desempeño superior frente a LSTM, ARIMA, y Prophet, demostrando la capacidad de las redes neuronales recurrentes para manejar series temporales altamente variables, no lineales y con patrones de estacionalidad irregular, como ocurre con las precipitaciones en la región de Tacna. Este hallazgo abre nuevas posibilidades para la aplicación de arquitecturas de deep learning en escenarios donde los métodos tradicionales muestran limitaciones.

Desde el enfoque metodológico, uno de los principales aportes de este estudio es la incorporación de la validación cruzada K-Fold utilizando TimeSeriesSplit, que permitió obtener métricas más robustas, estables y representativas del comportamiento real de los modelos. Asimismo, el uso de métricas como MSE, MAE, RMSE, MAPE y Exactitud, brindó una evaluación integral del desempeño, permitiendo una comparación equilibrada entre los distintos enfoques y fortaleciendo la rigurosidad técnica del análisis. Este marco metodológico constituye una referencia valiosa para investigaciones posteriores que busquen evaluar modelos predictivos en contextos de datos climáticos limitados o con alta variabilidad.

A nivel local, este trabajo tiene repercusiones directas, ya que se desarrolló a partir de datos de la estación meteorológica Jorge Basadre en Tacna. El estudio impulsa la adopción de herramientas avanzadas basadas en inteligencia artificial para enfrentar problemáticas regionales críticas, como la gestión de riesgos climáticos, la planificación agrícola y la administración del recurso hídrico. La demostración de la eficacia del modelo GRU ofrece a instituciones como la Dirección de Gestión del Riesgo de Desastres y el SENAMHI un modelo replicable y escalable que puede incorporarse en sistemas de alerta temprana y plataformas de monitoreo climático.

De este modo, el aporte práctico de esta tesis trasciende el ámbito académico, pues proporciona una base científica para la toma de decisiones estratégicas en la región de Tacna. La incorporación de predicciones más precisas contribuye al fortalecimiento de la planificación territorial, la anticipación de eventos meteorológicos extremos y la alineación con lineamientos de la Política Nacional del Ambiente y la Gestión del Riesgo de Desastres. En conjunto, este trabajo demuestra cómo un modelo de inteligencia artificial puede convertirse en una herramienta de política pública proactiva y orientada a la prevención.

Finalmente, las implicancias de esta investigación abren líneas claras para futuros estudios, tales como la integración de nuevas variables meteorológicas (presión atmosférica, velocidad del viento, temperatura del mar), la ampliación del horizonte histórico de datos, el uso de arquitecturas híbridas basadas en aprendizaje profundo y

la evaluación del modelo en otras estaciones del sur del Perú. Estas mejoras permitirían perfeccionar el desempeño del modelo y consolidar su aplicación en escenarios climáticos complejos.

CONCLUSIONES

Se ha logrado con éxito el objetivo principal de la investigación: identificar el modelo de machine learning más preciso para predecir las precipitaciones en la estación Jorge Basadre, ubicada en la región de Tacna, Perú. Los resultados obtenidos permiten aceptar la hipótesis general que postulaba que el modelo de machine learning de mayor exactitud permite la predicción de pronóstico de precipitaciones en la estación Jorge Basadre de la Región Tacna-Perú, en comparación con los métodos estadísticos clásicos.

Primero, la creación de un conjunto de datos estructurado, basado en los registros históricos de precipitaciones de la estación meteorológica Jorge Basadre, es fundamental para desarrollar modelos predictivos. Este conjunto de datos proporciona una base sólida y confiable, lo que facilita tanto el entrenamiento como la validación de algoritmos de machine learning, mejorando así la precisión y eficiencia en la predicción de eventos meteorológicos en Tacna, Perú. Por lo tanto, se declara que la primera hipótesis específica ha sido aceptada, validando que la etapa de depuración y estructuración del registro cumplió con los estándares de confiabilidad y continuidad cronológica requeridos para la correcta aplicación del procedimiento metodológico de estimación.

Segundo, el entrenamiento de modelos de machine learning con los datos específicos de la estación Jorge Basadre ha probado ser una estrategia efectiva para generar predicciones sobre precipitaciones. Al incluir características del contexto climático local, el modelo se ajusta mejor a las condiciones reales, aumentando la confiabilidad de los resultados obtenidos para la región de Tacna, Perú. En consecuencia, se acepta la segunda hipótesis específica, la cual postulaba que el enfoque de estimación avanzada lograría generar proyecciones ajustadas y confiables al incorporar las variables predictoras del contexto climático, lo cual valida la totalidad del enfoque metodológico implementado.

Tercero, la comparación sistemática de varios modelos de machine learning (LSTM, GRU, ARIMA y Prophet), entrenados con los datos de la estación Jorge Basadre, permitió identificar diferencias significativas en su rendimiento a través de los

indicadores de error y el análisis de estabilidad, lo que confirma la aceptación de la tercera hipótesis específica. En particular, el modelo GRU mostró una mayor estabilidad y consistencia en las métricas de evaluación, destacándose como el algoritmo con la mejor capacidad predictiva entre los analizados. Esta comparación ayuda a seleccionar el modelo más adecuado para su futura implementación en sistemas de alerta temprana o en la planificación hidrometeorológica de la región.

RECOMENDACIONES

Las líneas de investigación futuras derivadas de este estudio podrían enfocarse en:

El SENAMHI (Servicio Nacional de Meteorología e Hidrología) debe garantizar la disponibilidad y el acceso a una mayor cantidad de datos diarios de precipitación, abarcando periodos de al menos cinco años. Universidad y centros de investigación (CITE) deben aplicar el modelo desarrollado en la presente tesis utilizando datos de otras estaciones meteorológicas del país y sus diferentes regiones del Perú.

El SENAMHI (Servicio Nacional de Meteorología e Hidrología), en su rol de gestor primario de datos climáticos en el Perú, debe liderar y garantizar la recopilación de registros históricos adicionales y la inclusión de datos de estaciones meteorológicas cercanas. Esta ampliación permitirá mejorar la robustez del modelo y facilitar su generalización a otras zonas climáticas de la región sur del Perú.

Investigadores, ingenieros de sistemas y la comunidad académica peruana (específicamente en las áreas de Ingeniería y Meteorología), en estrecha coordinación con el SENAMHI para el acceso a la data histórica, deben priorizar la incorporación de variables climáticas complementarias como temperatura, humedad relativa, velocidad del viento y presión atmosférica, a fin de enriquecer el conjunto de características de entrada del modelo y aumentar su capacidad de aprendizaje sobre las condiciones que influyen en la ocurrencia de precipitaciones.

Los investigadores del estudio y los ingenieros de Sistemas o Ciencia de Datos encargados del mantenimiento y mejora del modelo deberán optimizar los modelos seleccionados mediante técnicas avanzadas de ajuste de hiperparámetros, tales como Grid Search, Random Search o algoritmos bayesianos, con el fin de mejorar el rendimiento de predicción y reducir los errores métricos.

La Comunidad Académica Peruana (a nivel de posgrado y proyectos de investigación), en colaboración con los Centros de Innovación Tecnológica (CITE) y el equipo técnico de SENAMHI, debe explorar rigurosamente la viabilidad de explorar modelos híbridos y arquitecturas más complejas, como combinaciones entre modelos estadísticos (ARIMA)

y redes neuronales (LSTM y GRU), que podrían capturar tanto componentes lineales como no lineales de las series temporales meteorológicas.

El Gobierno Regional de Tacna, a través de la Dirección de Gestión del Riesgo de Desastres y la Gerencia de Recursos Hídricos, en estrecha colaboración con el SENAMHI, debe garantizar la implementación del modelo seleccionado (GRU) en una plataforma operativa con capacidad de visualización en tiempo real, para brindar apoyo a la toma de decisiones en áreas críticas como la gestión de recursos hídricos, agricultura, defensa civil y planificación territorial en la región de Tacna.

Los Equipos Técnicos de Machine Learning y Monitoreo del SENAMHI, en colaboración con el Personal de Mantenimiento de Software de los Gobiernos Regionales, deben realizar validaciones periódicas del modelo con nuevos datos, asegurando su actualización y adaptabilidad ante cambios en los patrones climáticos. Esto permitirá mantener un nivel de precisión adecuado en las predicciones a lo largo del tiempo.

El Consejo Nacional de Ciencia, Tecnología e Innovación Tecnológica (CONCYTEC), el Ministerio del Ambiente (MINAM) y las Gerencias Regionales de Desarrollo Económico y Social deben fomentar la integración de herramientas de Inteligencia Artificial en el monitoreo climático regional, promoviendo alianzas entre instituciones meteorológicas, universidades y gobiernos locales, para fortalecer los sistemas de alerta temprana frente a eventos hidrometeorológicos extremos.

Se recomienda utilizar series de datos meteorológicos de al menos 10 años, ya que un mayor volumen de información permite capturar patrones climáticos de largo plazo, mejorando la precisión y capacidad de generalización de los modelos de aprendizaje automático

REFERENCIAS BIBLIOGRÁFICAS

Alfaro, D., Chaparro, D., Lozano, J. y Palma, J. (2025). *Análisis de rendimiento de modelos gratuitos de Machine Learning (aprendizaje automático) utilizados en infraestructura de la nube en la predicción de la diabetes.* [Tesis de grado - epecialización, Universidad EAN]. Repositorio Institucional Universidad EAN. http://hdl.handle.net/10882/14477

Arias, F. (2012). *El Proyecto de Investigación Introducción a la metodología científica*. https://abacoenred.org/wp-content/uploads/2019/02/El-proyecto-de-investigación-F.G.-Arias-2012-pdf-1.pdf

Basak, A., Rahman, A., Das, J., Hosono, T., & Kisi, O. (2022). *Drought forecasting using the Prophet model in a semi-arid climate region of western India*. https://doi.org/10.6084/M9.FIGSHARE.19886629.V2

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166. https://doi.org/10.1109/72.279181

Blanco, P. (2020). La intensidad de las precipitaciones y el cambio climático: tendencias y variabilidades interanuales registradas en algunas localidades del Nordeste Argentino (período 1971-2019). *Geográfica Digital, 17*(34), 47–64. https://doi.org/10.30972/geo.17344481

Brillinger, D. (2001). *Time Series Data Analysis and Theory*. https://es.scribd.com/document/551986865/David-r-Brillinger-Time-Series-Data-Analysis-and-Theory-2001-Compress

Campos, D. (2023). Temas recientes del análisis de frecuencias hidrológico. *En Instituto Mexicano de Tecnología del Agua eBooks.* https://doi.org/10.24850/b-imta-2023-12

Carreño, S. (2023). *Modelo predictivo del proceso de ventas utilizando inteligencia de negocios y data analitics en la empresa centro textil De la Matta S.A.C.* 1–215. [Tesis de pregrado, Universidad Señor de Sipán]. Repositorio Institucional Universidad Señor de Sipán. https://hdl.handle.net/20.500.12802/10636

Ccoyccosi R., Huanay L., Huayllasco E., Loayza V., y Mayorga K. (2021). *Propuesta de un modelo de machine learning para el pronóstico de la demanda de prendas de vestir en la Corporación Brusko S.A.C.* [Tesis de Licenciatura, Universidad ESAN. Facultad de Ingeniería]. Repositorio Institucional Universidad ESAN. https://hdl.handle.net/20.500.12640/2931

Chambers, J., Cook, M., Burkitt, A., & Grayden, D. (2024). Using Long Short-Term Memory (LSTM) recurrent neural networks to classify unprocessed EEG for seizure prediction. *Frontiers in Neuroscience*, *18*, 1472747. https://doi.org/10.3389/fnins.2024.1472747

Chen, Y., Liu, X., Rao, M., Qin, Y., Wang, Z., & Ji, Y. (2025). Explicit speed-integrated LSTM network for non-stationary gearbox vibration representation and fault detection under varying speed conditions. *Reliability Engineering & System Safety*, *254*, 110596. https://doi.org/10.1016/J.RESS.2024.110596

Cho, K., Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. *Proceedings of SSST 2014 -*

8th Workshop on Syntax, Semantics and Structure in Statistical Translation, 103–111. https://doi.org/10.3115/V1/W14-4012

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. https://arxiv.org/pdf/1412.3555

Coralogix. (2025, 3 junio). A Comprehensive Guide to Mean Absolute Percentage Error (MAPE) - Coralogix. Obtenido de https://coralogix.com/ai-blog/a-comprehensive-guide-to-mean-absolute-percentage-error-mape/

Egas, K., y Roque, T. (2020). *Diseño de un modelo predictivo basado en técnicas de machine learning que permita determinar la temperatura usando los datos de una miniestación meteorológica de la ciudad de Guayaquil*. [Tesis de pregrado, Universidad de Guayaquil]. Repositorio Institucional Universidad de Guayaquil. http://repositorio.ug.edu.ec/handle/redug/48892

Encina, A., Pacheco, M., y Vargas, V. (2023). *Técnicas de Machine Learning para la predicción del caudal efluente de la represa Condoroma*. [Tesis de pregrado,Universidad ESAN]. Repositorio Institucional Universidad ESAN. https://hdl.handle.net/20.500.12640/3377

Gamez, M. (2015, septiembre 17). Objetivos y metas de desarrollo sostenible. Desarrollo Sostenible. Obtenido de https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible

Guichard, D., Aguilar, M., Muciño, J. J., y Pérez, C. (2020). Obtención de factores de ajuste por duración y curvas i-d-tr , a partir de precipitaciones históricas en la cuenca de Chicoasén, perteneciente a la rh 30, en Chiapas. https://ingenieria.unach.mx/images/Articulos_revista/revistapakbal_41_pag26-30.pdf

Guo, L., Fang, W., Zhao, Q., & Wang, X. (2021). The hybrid PROPHET-SVR approach for forecasting product time series demand with seasonality. *Computers and Industrial Engineering*, *161*. https://doi.org/10.1016/j.cie.2021.107598

Hernández, I. y Torres, L. (2022). *Modelo de pronóstico de demanda para productos del sector eléctrico*. [Tesis pregrado, Universidad de los Andes]. Repositorio Institucional Universidad de los Andes. https://hdl.handle.net/1992/58747

Hernández, R., y Mendoza, C. (2018). Metodología de la investigación. *Las rutas cuantitativa, cualitativa y mixta.* https://virtual.cuautitlan.unam.mx/rudics/?p=2612

Heskes, T. (2017). Expectation Propagation. En Encyclopedia of Machine Learning and Data Mining (pp. 482–487). Springer Science+Business Media. https://doi.org/10.1007/978-1-4899-7687-1_953

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

Hodson, T., Over, T., & Foks, S. (2021). Mean Squared Error, Deconstructed. *Journal of Advances in Modeling Earth Systems*, 13. e2021MS002681. https://doi.org/10.1029/2021MS002681

Holdsworth, J., y Scapicchio, M. (2025, febrero 21). ¿Qué es el deep learning? Ibm.com. Obtenido de https://www.ibm.com/es-es/think/topics/deep-learning

Huang, C., Zhao, T., Huang, D., Cen, B., Zhou, Q., & Chen, W. (2024). Artificial intelligence-based power market price prediction in smart renewable energy systems: Combining prophet and transformer models. *Heliyon*, *10*(20), e38227. https://doi.org/10.1016/j.heliyon.2024.e38227

IGP. (2021, diciembre 1). Análisis y evaluación histórica de lluvias extremas en la región Tacna. *Instituto Geofísico del Perú.* https://repositorio.igp.gob.pe/items/d8ea3f3e-ab7e-4835-b9bc-9dd8eb8c98a5

Khan, M., Siddique, M., Sakib, S., Aziz, A., Tasawar, I., & Hossain, Z. (2020). Prediction of Temperature and Rainfall in Bangladesh using Long Short Term Memory Recurrent Neural Networks. *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Istanbul, Turkey, 2020, pp. 1-6*. https://doi.org/10.1109/ISMSIT50672.2020.9254585

Kotu, V., & Deshpande, B. (2019). Deep Learning. En *Data Science* (pp. 307–342). Elsevier. https://doi.org/10.1016/b978-0-12-814761-0.00010-1

Liu, X., Hou, B., Xie, Z., Feng, N., & Dong, X. (2024). Integrating gated recurrent unit in graph neural network to improve infectious disease prediction: an attempt. *Frontiers in Public Health*, *12*, 1397260. https://doi.org/10.3389/fpubh.2024.1397260

López, J. (2023). *Pronóstico de precipitación en el centro de Tungurahua aplicando aprendizaje estadístico con redes neuronales artificiales*. [Tesis de posgrado, Universidad Técnica de Ambato]. Repositorio Institucional Universidad Técnica de Ambato. https://repositorio.uta.edu.ec/handle/123456789/37468

Marín, D., y Pineda, I. (2019). *Modelo predictivo Machine Learning aplicado a análisis de datos Hidrometeorológicos para un SAT en Represas.* [Tesis de pregrado, Universidad Tecnológica del Peru]. Repositorio Institucional Universidad Tecnológica del Peru. https://hdl.handle.net/20.500.12867/3300

Mayer, M., & Yang, D. (2024). Potential root mean square error skill score. *Journal of Renewable and Sustainable Energy*, *16*(1). https://doi.org/10.1063/5.0187044

Meng, Z., Qin, X., Liu, J., Li, J., & Fan, F. (2025). A denoising algorithm based on ARIMA and competitive K-SVD for the diagnosis of rolling bearing faults. *Applied Acoustics*, *228*, 110309. https://doi.org/10.1016/j.apacoust.2024.110309

Mienye, I., Swart, T., & Obaido, G. (2024). Recurrent neural networks: A comprehensive review of architectures, variants, and applications. In *Information (Switzerland)* (Vol. 15, Issue 9). Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/info15090517

Naranjo, F., & González, J. (2009). *Pronostico de la precipitación para la zona de influencia de la estación agroclimática Yariguies, utilizando técnicas de Machine Learning*. [Tesis de posgrado, Fundación Universitaria Los Libertadores]. Repositorio Institucional Fundación Universitaria Los Libertadores http://hdl.handle.net/11371/4913

Nolasco, P. (2023). *Aplicación de Machine Learning para pronóstico de desplazamiento de lluvias usando imágenes del radar de lluvias de UDEP.* [Tesis de pregrado, Universidad de Piura]. Repositorio Institucional Universidad de Piura. https://hdl.handle.net/11042/6007

Obaidalla, I. (2024). Rainfall Prediction Using Machine Learning Methods. [Graduate Thesis Approval, Rochester Institute of Technology]. Institutional Repository Rochester Institute of Technology. https://repository.rit.edu/theses/12028/

Ohba, M. (2021). Precipitation under climate change. En *Elsevier eBooks* (pp. 21-51). https://doi.org/10.1016/b978-0-12-822699-5.00002-1

OMM. (2025, mayo 27). Las predicciones climáticas mundiales indican temperaturas en niveles sin precedentes o cercanas a ellos durante los próximos cinco años.

Obtenido de https://wmo.int/es/news/media-centre/las-predicciones-climaticas-mundiales-indican-temperaturas-en-niveles-sin-precedentes-o-cercanas

ONU. (2021, septiembre 22). ODS Objetivos de Desarrollo Sostenible. Pacto Mundial ONU. Obtenido de https://www.pactomundial.org/que-puedes-hacer-tu/ods/

Ortega, C. (2023, noviembre 7). Modelos de machine learning: Qué son, tipos y aplicaciones. QuestionPro. https://www.questionpro.com/blog/es/modelos-de-machine-learning/

Pérez, A., Menéndez, P., Sanz, P., Iglesias, Lloret. L., y Rodríguez, D. (2022). Inteligencia artificial en Radiología: introducción a los conceptos más importantes. *Radiología, 64*(3), 228-236. https://doi.org/10.1016/j.rx.2022.03.003

Pino, E., y Chávarri, E. (2022). Evidencias de cambio climático en la región hiperárida de la costa sur de Perú, cabecera del desierto de Atacama. *Tecnología y Ciencias del Agua, 13(1)*, 333–376. https://doi.org/10.24850/j-tyca-2022-01-08

Ponce, G. (2022). *Modelo basado en Machine Learning para optimizar el pronóstico de ventas de la empresa Ricos Pan, año 2020 - 2021.* [Tesis de pregrado, Universidad Nacional del Altiplano - Puno]. Repositorio Institucional Universidad UNAP. https://repositorio.unap.edu.pe/handle/20.500.14082/19132

Ponce, R., (2023). *Pronóstico de temperatura en un invernadero usando algoritmos de aprendizaje supervisado*. [Tesis de Postgrado, Instituto Tecnológico y de Estudios Superiores de Occidente]. Repositorio Institucional ITESO. https://repositorio.ausjal.org/handle/20.500.12032/122534

Price, I., Sanchez-Gonzalez, A., Alet, F., Andersson, T., El-Kadi, A., Masters, D., Ewalds, T., Stott, J., Mohamed, S., Battaglia, P., Lam, R., & Willson, M. (2025). Probabilistic weather forecasting with machine learning. *Nature*, *637*(8044), 84–90. https://doi.org/10.1038/s41586-024-08252-9

Pulzara, C., y Losada, J. (2023). Análisis de series temporales en estaciones meteorológicas para la predicción de la precipitación en la ciudad de Manizales, Colombia. *Revista de Climatología*, *23*, 58–70. https://doi.org/10.59427/RCLI/2023/V23.58-70

Quispe, L. (2024). *Modelo de predicción climatológica con inteligencia artificial (AI) en la región Ica, 2022*. [Tesis de posgrado, Universidad Nacional San Luis Gonzaga]. Repositorio Institucional Universidad Nacional San Luis Gonzaga. https://hdl.handle.net/20.500.13028/5629

Reddy, B., Naik, J., Kumar, V., Kumar, S., Haritha, G., & Reddy, R. (2025). A Methodological Review on Time Series Forecasting by using ARIMA. *Advances In Engineering Research/Advances In Engineering Research,* 709-719. https://doi.org/10.2991/978-94-6463-662-8_55

RPP. (2020, 23 febrero). Huaicos en Tacna han dejado tres muertos y 250 viviendas afectadas. RPP Noticias. Obtenido de https://rpp.pe/peru/tacna/tacna-huaicos-han-dejado-cuatro-muertos-y-250-viviendas-afectadas-noticia-1247251

Seifi, A., Pourebrahim, S., Ehteram, M., & Shabanian, H. (2024). A robust multi-model framework for groundwater level prediction: The BFSA-MVMD-GRU-RVM model. *Results in Engineering*, *24*, 103250. https://doi.org/10.1016/j.rineng.2024.103250

Sen, P., Roy, M., & Pal, P. (2016). Application of ARIMA for forecasting energy consumption and GHG emission: A case study of an Indian pig iron manufacturing organization. *Energy*, *116*, 1031–1038.

https://doi.org/10.1016/j.rineng.2024.103250

SENAMHI. (2024, noviembre 8). Perspectivas climáticas, periodo febrero - abril 2024. Informe Técnico N°01-2024/SENAMHI-DMA-SPC. Obtenido de https://hdl.handle.net/20.500.12542/3105

Spenassato, D., Trierweiller, A., Bornia, A., & Frazzon, L. (2015). Dow jones sustainability index: Use of forecasting models to assist decision making. *Revista ESPACIOS.* (No 11). https://www.revistaespacios.com/a15v36n11/15361121.html

Taguela, T., Akinsanola, A., Bobde, V., Raji, I., Adeyeri, O., & Adebiyi, A. (2025). Precipitation distribution over Africa: observations and modeling. *Aerosols and Precipitation Over Africa: Progress, Challenges, and Prospects*, 121–146. https://doi.org/10.1016/B978-0-44-314050-1.00009-8

Taylor, S., & Letham, B. (2018). Forecasting at Scale. *American Statistician*, *72*(1), 37–45. https://doi.org/10.1080/00031305.2017.1380080

Urbina, B., y Takahashi, K. (2023). *Desarrollo de un modelo de Machine Learning para el pronóstico meteorológico de precipitación a escala subestacional en Perú* [Instituto Geofísico del Perú]. http://hdl.handle.net/20.500.12816/5567

Uzel, Z. (2023). *Comparative Analysis of LSTM, ARIMA, and Facebook's Prophet for Traffic Forecasting: Advancements, Challenges, and Limitations*. https://repository.tudelft.nl/record/uuid:29fcfb96-3e96-4b11-93ec-217ba69ea412

Willmott, C., & Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research*, *30*(1), 79–82. https://doi.org/10.3354/CR030079

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2020). Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *35th AAAI Conference on Artificial Intelligence, AAAI 2021*, *12B*, 11106–11115. https://doi.org/10.1609/aaai.v35i12.17325

ANEXOS

Anexo 1. Matriz de consistencia

TÍTULO: MODELO DE MACHINE LEARNING PARA EL PRONÓSTICO DE PRECIPITACIONES EN LA ESTACIÓN JORGE BASADRE DE LA REGIÓN TACNA

Problema	Objetivos	Hipótesis	Variables	Indicador	Metodología
Problema general ¿Cuál es el modelo predictivo de machine learning que ofrece mayor exactitud en el pronóstico de precipitaciones en la estación Jorge Basadre de la Región Tacna, utilizando como variables predictoras la Fecha, Temperatura Máxima, Temperatura Mínima, Humedad Relativa y Precipitación Histórica?	**Objetivo general** Determinar el modelo de machine learning de mayor exactitud para el pronóstico de precipitaciones en la estación Jorge Basadre de la Región Tacna – Perú, utilizando como variables predictoras la Fecha, Temperatura Máxima, Temperatura Mínima, Humedad Relativa y Precipitación Histórica.	**Hipótesis general** El modelo de machine learning de mayor exactitud permite la predicción de pronóstico de precipitaciones en la estación Jorge Basadre de la Región Tacna – Perú, utilizando como variables predictoras la Fecha, Temperatura Máxima, Temperatura Mínima, Humedad Relativa y Precipitación Histórica.	**Variables independientes** **Modelo de Machine Learning**	Tiempo de Respuesta Inferencia Consumo de Memoria	
Problemas específicos a. ¿Cuál es el conjunto de datos para el pronóstico de precipitaciones en la estación Jorge Basadre de la Región Tacna-Perú? b. ¿Cómo entrenar modelos de machine learning para el pronóstico de precipitaciones en la estación Jorge Basadre de la Región Tacna - Perú? c. ¿Cuál es la exactitud de los modelos de machine learning entrenados en el pronóstico de precipitaciones en la estación Jorge Basadre de la Región Tacna - Perú?	**Objetivos específicos** a. Generar un conjunto de datos para el pronóstico de precipitaciones en la estación Jorge Basadre de la Región Tacna - Perú. b. Entrenar modelos de machine learning de precipitaciones en la estación Jorge Basadre de la Región Tacna - Perú. c. Comparar la exactitud de los modelos de machine learning entrenados para el pronóstico de precipitaciones en la estación Jorge Basadre de la Región Tacna- Perú.	**Hipótesis especifica** a. La generación de un conjunto de datos basado en los registros meteorológicos de la estación Jorge Basadre facilitará la creación de modelos más precisos para el pronóstico de precipitaciones en la región de Tacna - Perú. b. El entrenamiento de modelos de machine learning con datos de la estación Jorge Basadre permitirá desarrollar predicciones precisas de precipitaciones en la región de Tacna - Perú. c. La comparación de los modelos de machine learning entrenados con datos de la estación Jorge Basadre permitirá identificar el modelo más preciso para el pronóstico de precipitaciones en la región de Tacna - Perú.	**Variable dependiente** **Pronóstico de Precipitaciones**	MAE (Error Absoluto Medio) MSE (Error Cuadrático Medio) RMSE (Raíz del Error Cuadrático Medio) MAPE (Error Porcentual Absoluto Medio)	**Tipo de Investigación** Aplicada **Enfoque de Investigación** Cuantitativo **Diseño de investigación** Experimental **Nivel de Investigación** Descriptivo y Comparativo.

Anexo 2. Estructura del conjunto de datos meteorológicos utilizados

Tabla 12

Variables del Conjunto de Datos Meteorológicos Diarios

Nombre de la variable	**Descripción**	**Tipo de dato**	**Unidad de medida**	**Ejemplo**
Fecha	Fecha de la medición meteorológica diaria	Fecha	—	2023-01-01 – 2025-03-31
temp_max	Temperatura máxima registrada durante el día	Numérico	°C	28,4
temp_min	Temperatura mínima registrada durante el día	Numérico	°C	16,2
Humedad	Humedad relativa promedio durante el día	Numérico	%	75,0
Precipitación	Precipitación total acumulada durante el día	Numérico	mm	3,5

Nota. Descripción de las Variables del Conjunto de Datos Meteorológicos Diarios

Anexo 3. Código fuente empleado para la implementación del modelo

En el presente anexo se incluye el código fuente utilizado para la implementación de los modelos predictivos desarrollados en el marco de esta investigación. Dichos algoritmos fueron aplicados al conjunto de datos meteorológicos pertenecientes a la estación Jorge Basadre, ubicada en la región de Tacna, los cuales fueron previamente procesados y depurados con el fin de garantizar la calidad y consistencia de la información.

Durante el desarrollo del estudio, se implementaron los siguientes modelos de predicción:

Modelo ARIMA (Modelo Autorregresivo Integrado de Media Móvil): Modelo estadístico clásico para el análisis de series temporales lineales, que incorpora componentes autorregresivos, de diferenciación e integración, y de medias móviles, permitiendo capturar patrones estacionales y de tendencia.

Modelo LSTM (Memoria a Largo Corto Plazo): Este tipo de red neuronal recurrente (RNN) es capaz de captar relaciones a largo plazo en secuencias de datos temporales. Esto es especialmente valioso en la meteorología, donde las variaciones son bastante complejas.

Modelo GRU (Unidad Recurrente con Compuertas): Esta es una versión de las redes LSTM que cuenta con una estructura más sencilla y eficiente, pero aun así ofrece un rendimiento competitivo en las tareas de pronóstico de series temporales.

Modelo Prophet: Este modelo aditivo fue creado por Meta (anteriormente Facebook) y está diseñado para prever series temporales que incluyen tendencias y estacionalidades no lineales. Es fácil de configurar y se adapta a diferentes situaciones.

También se realizó una limpieza y preparación exhaustiva del conjunto de datos, que abarcó varios pasos:

- Eliminación y manejo de valores nulos o inconsistentes.
- Conversión de las marcas de tiempo al formato estándar de fecha y hora (DateTime).
- Detección de anomalías mediante métodos estadísticos y visuales, como boxplots, para reducir el impacto de los valores atípicos en el entrenamiento de los modelos.
- Los siguientes fragmentos de código son evidencia del desarrollo técnico de este estudio y pueden ser útiles para futuras investigaciones o réplicas experimentales.

Limpieza de Datos

1. Instalar e importar Librerías

```
# Paso 1: Configurar el entorno
# Instala solo si no está instalado
!pip install pandas matplotlib Seaborn
# Paso 2: Importar las librerías necesarias
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import drive
```

2. Montar Google Drive

```
# 📌 Paso 2: Montar Google Drive
drive.mount('/content/drive')
```

3. Cargar Datos

```
# Paso 3: Cargar el dataset
file_path = '/content/drive/MyDrive/proyecto/Limpieza de
Datos/datasenamhi2025.csv'
data = pd.read_csv(file_path)
```

4. Limpieza básica de datos

```
# Paso 4: Limpieza básica
data.columns = data.columns.str.strip()  # Eliminar espacios en los nombres de
columnas
data['fecha'] = pd.to_datetime(data['fecha'], dayfirst=True,
errors='coerce')  # Convertir a fecha
```

5. Seleccionar datos

```
# Paso 5: Crear un rango de fechas completo (desde 2021-01-01 hasta 2025-03-31)
rango_fechas = pd.date_range(start='2021-01-01', end='2025-03-31')
df_fechas = pd.DataFrame({'fecha': rango_fechas})
```

6. Visualizar datos

```
# Paso 6: Margen del rango completo con los datos originales
data_completo = df_fechas.merge(data, on='fecha', how='left')
```

7. Visualizar fechas con valores faltantes

```
# Paso 7: Mostrar fechas con valores faltantes
columnas_criticas = ['tem max', 'tem min', 'humedad', 'precipitacion']
fechas_con_na =
data_completo[data_completo[columnas_criticas].isnull().any(axis=1)]
print("Fechas con valores faltantes en columnas críticas:")
display(fechas_con_na[['fecha'] + columnas_criticas])
```

Figura 29

Registro con valores nulos

	fecha	tem max	tem min	humedad	precipitacion
12	2021-01-13	NaN	NaN	NaN	NaN
13	2021-01-14	NaN	NaN	NaN	NaN
14	2021-01-15	NaN	NaN	NaN	NaN
15	2021-01-16	NaN	NaN	NaN	NaN
16	2021-01-17	NaN	NaN	NaN	NaN
...	...	...	...	...	...
1546	2025-03-27	NaN	NaN	NaN	NaN
1547	2025-03-28	NaN	NaN	NaN	NaN
1548	2025-03-29	NaN	NaN	NaN	NaN
1549	2025-03-30	NaN	NaN	NaN	NaN
1550	2025-03-31	NaN	NaN	NaN	NaN

991 rows × 5 columns

Nota. Se visualiza todos los registros con valores nulos. Elaboración Propia.

```
# Guardar tabla de fechas con NaN
nan_output_path = '/content/drive/MyDrive/proyecto/Limpieza de
Datos/fechas_con_nan.csv'
fechas_con_na.to_csv(nan_output_path, index=False)
```

8. Eliminar filas con valores nulos

```
# Paso 8: Eliminar filas con valores nulos en las columnas críticas
data_clean = data_completo.dropna(subset=columnas_criticas)
```

9. Mostrar conteo de filas antes y después

```
# Paso 09: Mostrar conteo de filas antes y después
print("Cantidad de filas originales:", len(data))
print("Cantidad de filas con fechas completas:", len(data_completo))
print("Cantidad de filas después de limpieza:", len(data_clean))
```

10. Guardar dataset limpio

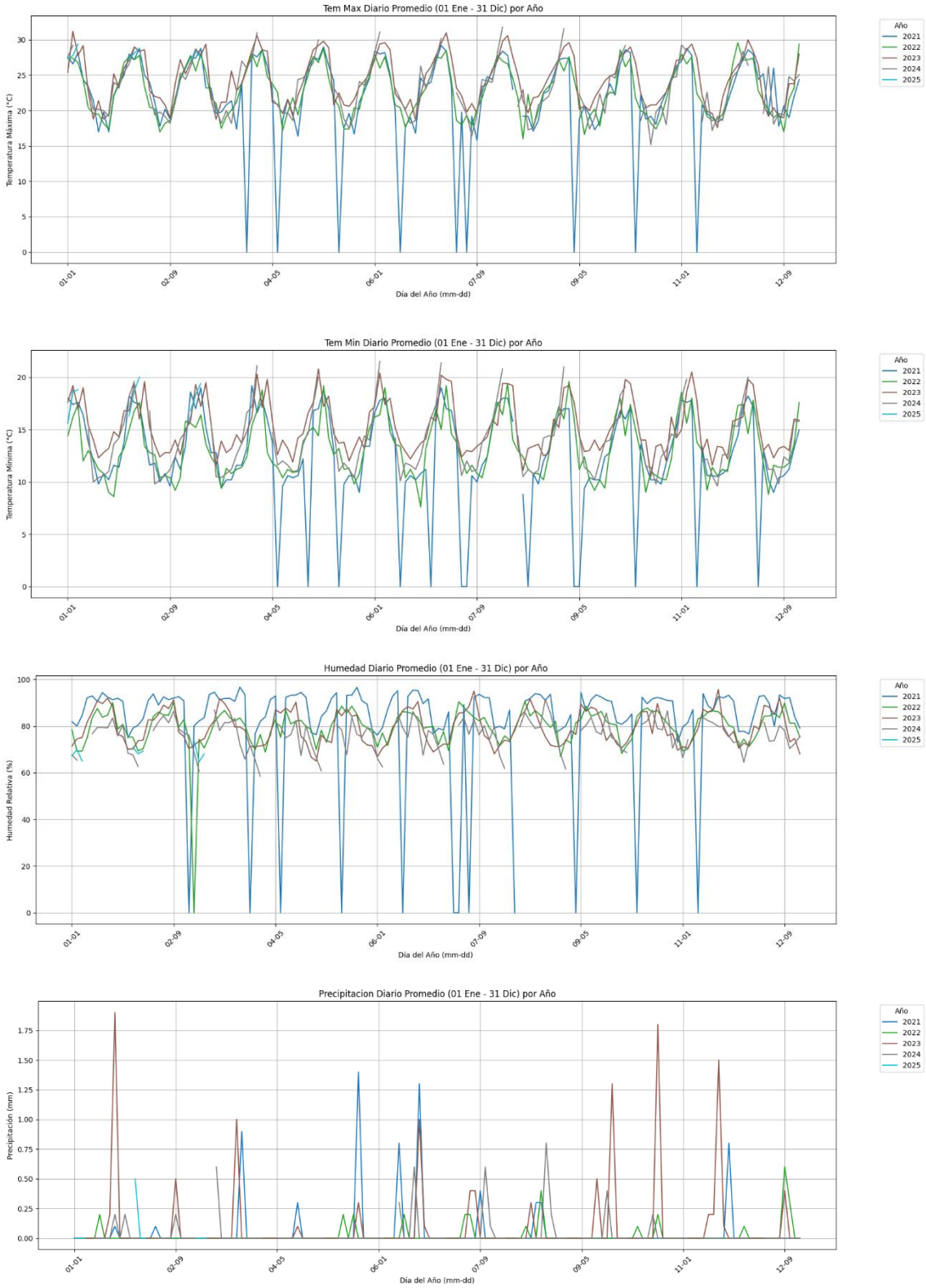
```
# Paso 10: Guardar dataset limpio
output_path = '/content/drive/MyDrive/proyecto/Limpieza de
Datos/datasetlimpia.csv'
data_clean.to_csv(output_path, index=False)
```

11. Análisis Exploratorio de Datos

```
# Paso 11: Análisis Exploratorio de Datos (EDA)
# Crear columnas auxiliares
data_clean['año'] = data_clean['fecha'].dt.year
data_clean['dia_mes'] = data_clean['fecha'].dt.strftime('%m-%d')  # Para eje X

# Función para graficar una variable por día del año
def graficar_variable(variable, ylabel):
    pivot = data_clean.pivot_table(index='dia_mes', columns='año',
values=variable, aggfunc='mean')

    plt.figure(figsize=(18, 6))
    pivot.plot(ax=plt.gca(), cmap='tab10')
    plt.title(f'{variable.title()} Diario Promedio (01 Ene - 31 Dic) por Año')
    plt.xlabel('Día del Año (mm-dd)')
    plt.ylabel(ylabel)
    plt.xticks(rotation=45)
    plt.grid(True)
    plt.legend(title='Año', bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.tight_layout()
    plt.show()

# Graficar variables importantes
graficar_variable('tem max', 'Temperatura Máxima (°C)')
graficar_variable('tem min', 'Temperatura Mínima (°C)')
graficar_variable('humedad', 'Humedad Relativa (%)')
graficar_variable('precipitacion', 'Precipitación (mm)')
```

Figura 30

Gráfico de variables meteorológicas



Nota. Representación gráfica de las variables meteorológicas: temperatura máxima, temperatura mínima, humedad y precipitación. Elaboración Propia.

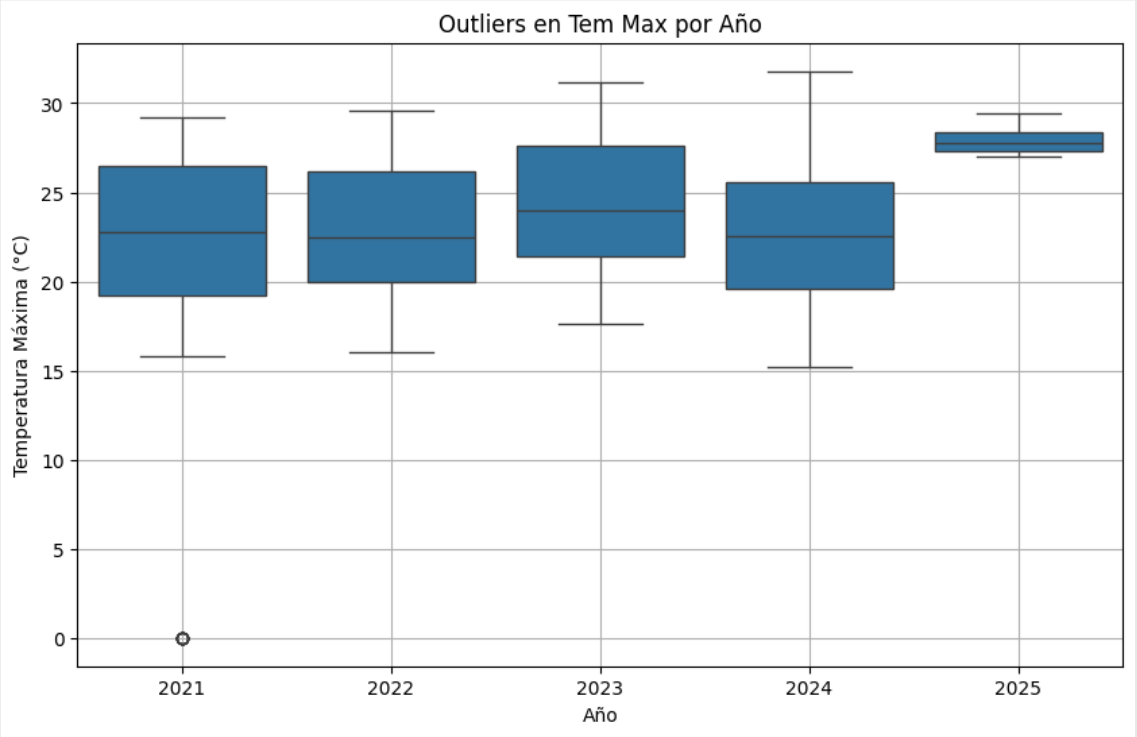
12. Visualización de outliers o atípicos con boxplots

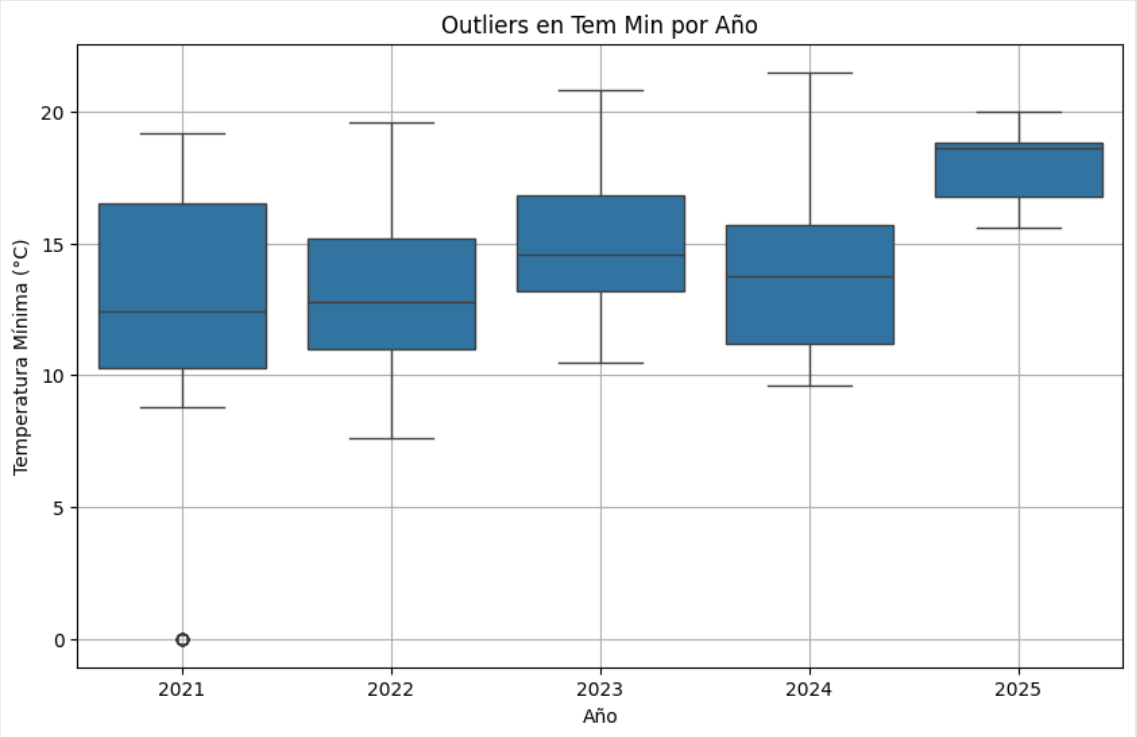
```
# Paso 12: Visualización de outliers o atípicos con boxplots

def graficar_boxplot(variable, ylabel):
    plt.figure(figsize=(10, 6))
```

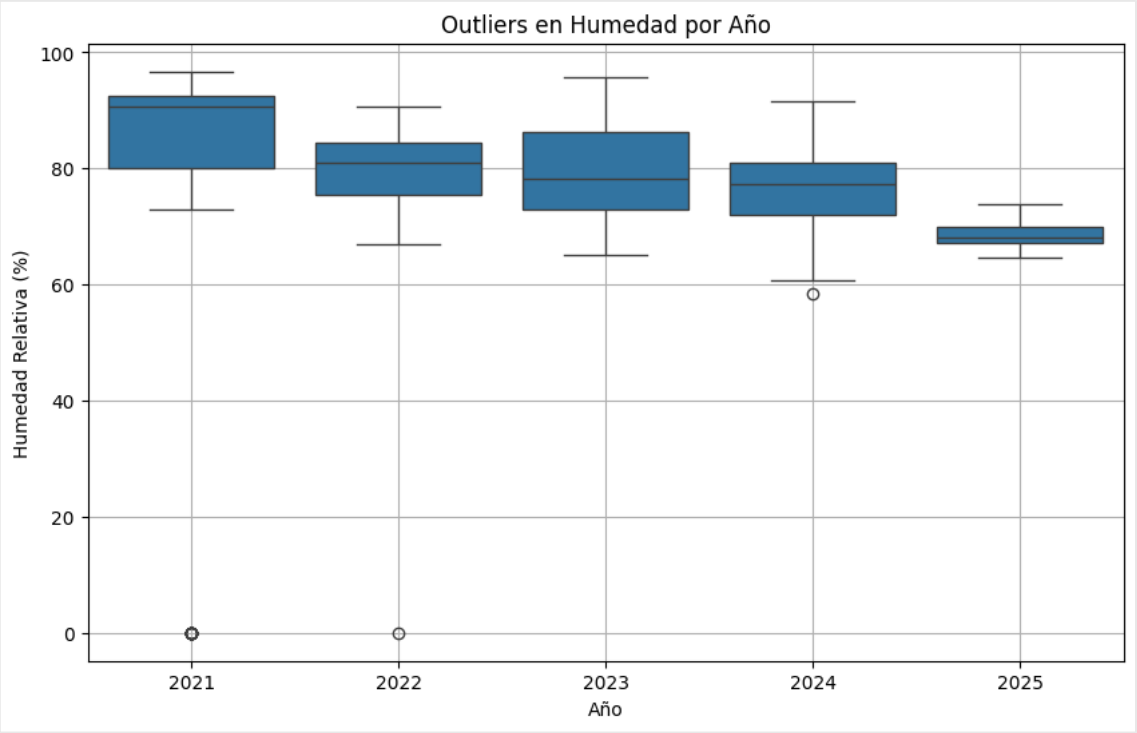
```
    sns.boxplot(x=data_clean['año'], y=data_clean[variable])
    plt.title(f'Outliers en {variable.title()} por Año')
    plt.xlabel('Año')
    plt.ylabel(ylabel)
    plt.grid(True)
    plt.show()

# Graficar boxplots para variables críticas
graficar_boxplot('tem max', 'Temperatura Máxima (°C)')
graficar_boxplot('tem min', 'Temperatura Mínima (°C)')
graficar_boxplot('humedad', 'Humedad Relativa (%)')
graficar_boxplot('precipitacion', 'Precipitación (mm)')
```

Figura 31

Gráfico de outliers o atípicos con boxplots







Nota. Representación gráfica de outliers o atípicos con boxplots. Elaboración Propia.

Test de Datos Limpios

1. Montar Google Drive para acceder al archivo

```
# ------------------------------------------
# 1. Montar Google Drive para acceder al archivo
# ------------------------------------------
from google.colab import drive

# Monta tu Google Drive (solo debes autorizar una vez)
drive.mount('/content/drive')
```

2. Importar librerías necesarias para análisis

```
# ------------------------------------------
# 2. Importar librerías necesarias para análisis
# ------------------------------------------
import pandas as pd              # Para manipulación de datos
import numpy as np               # Para cálculos numéricos
import seaborn as sns            # Para visualizaciones estadísticas
import matplotlib.pyplot as plt  # Para gráficos generales

# Configuraciones generales
sns.set(style="whitegrid")
```

3. Cargar el archivo CSV desde Google Drive

```
# ------------------------------------------
# 3. Cargar el archivo CSV desde Google Drive
# ------------------------------------------
# Cambia esta ruta según la ubicación exacta de tu archivo en Drive
ruta_archivo = '/content/drive/MyDrive/proyecto/Limpieza de
Datos/datasetlimpia.csv'

# Leer el archivo CSV en un DataFrame
df = pd.read_csv(ruta_archivo)

# Mostrar las primeras filas como vista previa
df.head()
```

Figura 32

Visualización de datos limpios importados

	fecha	tem max	tem min	humedad	precipitacion
0	2021-01-01	27.6	18.0	81.9	0.0
1	2021-01-02	26.6	17.4	80.0	0.0
2	2021-01-03	28.2	17.6	84.5	0.0
3	2021-01-04	24.4	16.4	92.1	0.0
4	2021-01-05	23.4	13.8	92.9	0.0

Nota. Se presentan los primeros registros del conjunto de datos depurados. Elaboración Propia.

4. Función para testear limpieza del dataset

```python
# ------------------------------------------
# 4. Función para testear limpieza del dataset
# ------------------------------------------
def test_limpieza_dataset(df):
    print("=== TEST DE LIMPIEZA DE DATOS ===\n")

    # ------------------------------------
    # A. Información general del DataFrame
    # ------------------------------------
    print("→ Información general del dataset:")
    print(df.info())  # Tipos de datos y cantidad de no nulos
    print("\n→ Primeras 5 filas del dataset:")
    print(df.head())  # Muestra rápida del contenido

    # ------------------------------------
    # B. Verificación de valores nulos
    # ------------------------------------
    print("\n→ Recuento de valores nulos por columna:")
    print(df.isnull().sum())  # Conteo de valores faltantes

    # ------------------------------------
    # C. Verificación de duplicados
    # ------------------------------------
    duplicados = df.duplicated().sum()
    print(f"\n→ Filas duplicadas en el dataset: {duplicados}")

    # ------------------------------------
    # D. Revisión de tipos de datos
    # ------------------------------------
    print("\n→ Tipos de datos por columna:")
    print(df.dtypes)

    # ------------------------------------
    # E. Revisión de valores únicos por columna
    # ------------------------------------
    print("\n→ Cantidad de valores únicos por columna:")
    for col in df.columns:
        print(f"{col}: {df[col].nunique()} únicos")

    # ------------------------------------
    # F. Detección de outliers con el método IQR
    # ------------------------------------
    print("\n→ Detección de outliers (para variables numéricas):")
    num_cols = df.select_dtypes(include=[np.number]).columns
    for col in num_cols:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        outliers = df[(df[col] < Q1 - 1.5 * IQR) | (df[col] > Q3 + 1.5 * IQR)]
        print(f"{col}: {len(outliers)} valores atípicos detectados")

    # ------------------------------------
    # G. Validación de columnas de fecha
    # ------------------------------------
    print("\n→ Verificación de columnas con fechas válidas:")
    for col in df.columns:
        try:
            pd.to_datetime(df[col])
            print(f"{col}: ✅ Fechas válidas detectadas")
```

```
        except (ValueError, TypeError):
            pass  # No se pudo convertir, se ignora

    print("\n=== FIN DEL TEST DE LIMPIEZA ===")
```

5. Ejecutar el test con tu DataFrame

```
# ------------------------------------------
# 5. Ejecutar el test con tu DataFrame
# ------------------------------------------
test_limpieza_dataset(df)
```

Figura 33

Test de limpieza de datos

```
=== TEST DE LIMPIEZA DE DATOS ===

→ Información general del dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 560 entries, 0 to 559
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   fecha          560 non-null    object
 1   tem max        560 non-null    float64
 2   tem min        560 non-null    float64
 3   humedad        560 non-null    float64
 4   precipitacion  560 non-null    float64
dtypes: float64(4), object(1)
memory usage: 22.0+ KB
None

→ Primeras 5 filas del dataset:
        fecha  tem max  tem min  humedad  precipitacion
0  2021-01-01     27.6     18.0     81.9            0.0
1  2021-01-02     26.6     17.4     80.0            0.0
2  2021-01-03     28.2     17.6     84.5            0.0
3  2021-01-04     24.4     16.4     92.1            0.0
4  2021-01-05     23.4     13.8     92.9            0.0

→ Recuento de valores nulos por columna:
fecha            0
tem max          0
tem min          0
humedad          0
precipitacion    0
dtype: int64
```

```
→ Filas duplicadas en el dataset: 0

→ Tipos de datos por columna:
fecha             object
tem max          float64
tem min          float64
humedad          float64
precipitacion    float64
dtype: object

→ Cantidad de valores únicos por columna:
fecha: 560 únicos
tem max: 118 únicos
tem min: 104 únicos
humedad: 257 únicos
precipitacion: 15 únicos

→ Detección de outliers (para variables numéricas):
tem max: 9 valores atípicos detectados
tem min: 13 valores atípicos detectados
humedad: 13 valores atípicos detectados
precipitacion: 56 valores atípicos detectados

→ Verificación de columnas con fechas válidas:
fecha: ✅ Fechas válidas detectadas
tem max: ✅ Fechas válidas detectadas
tem min: ✅ Fechas válidas detectadas
humedad: ✅ Fechas válidas detectadas
precipitacion: ✅ Fechas válidas detectadas
```

Nota. La imagen presenta un resumen del proceso de exploración y validación inicial del conjunto de datos (EDA). Elaboración Propia.

Implementación del Modelo LSTM

1. Instalar e importar Librerías

```
# 📌 Paso 1: Instalar librerías necesarias
!pip install numpy pandas scikit-learn tensorflow folium

import numpy as np
import pandas as pd
import folium
import time
import math
import seaborn as sns
sns.set_style("whitegrid")
import tensorflow as tf
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from matplotlib.gridspec import GridSpec
from IPython.display import display, IFrame
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import LSTM, Dense, Dropout, Input,
BatchNormalization
from tensorflow.keras.regularizers import l1_l2, l1, l2 # Import l1_l2
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau #
Asegúrate de importar EarlyStopping y ReduceLROnPlateau
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from google.colab import drive
```

2. Montar Google Drive

```
# 📌 Paso 2: Montar Google Drive
drive.mount('/content/drive')
```

3. Cargar Datos y seleccionar variables

```
# 📌 Paso 3.1: Cargar datos
dataset_path = "/content/drive/MyDrive/TESIS MODELOS DE MACHINE LEARNING
V4/Data/datasetlimpia.csv"  # Ajusta la ruta si es necesario
data = pd.read_csv(dataset_path, parse_dates=['fecha'], dayfirst=True)
data = data.sort_values('fecha')
print("Primeros 5 registros del dataset:")
print(data.head())

# 📌 Paso 3.2: Seleccionar y preparar variables relevantes
data = data[['fecha', 'precipitacion', 'tem max', 'tem min',
'humedad']].dropna()
data.columns = ['fecha', 'precipitacion', 'temp_max', 'temp_min',
'humedad']  # Renombrar columnas
data.set_index('fecha', inplace=True)

# 📌 Paso 3.3: Validar rango de fechas hasta el 31 de Marzo de 2025
data = data[data.index <= '2025-03-31']
```

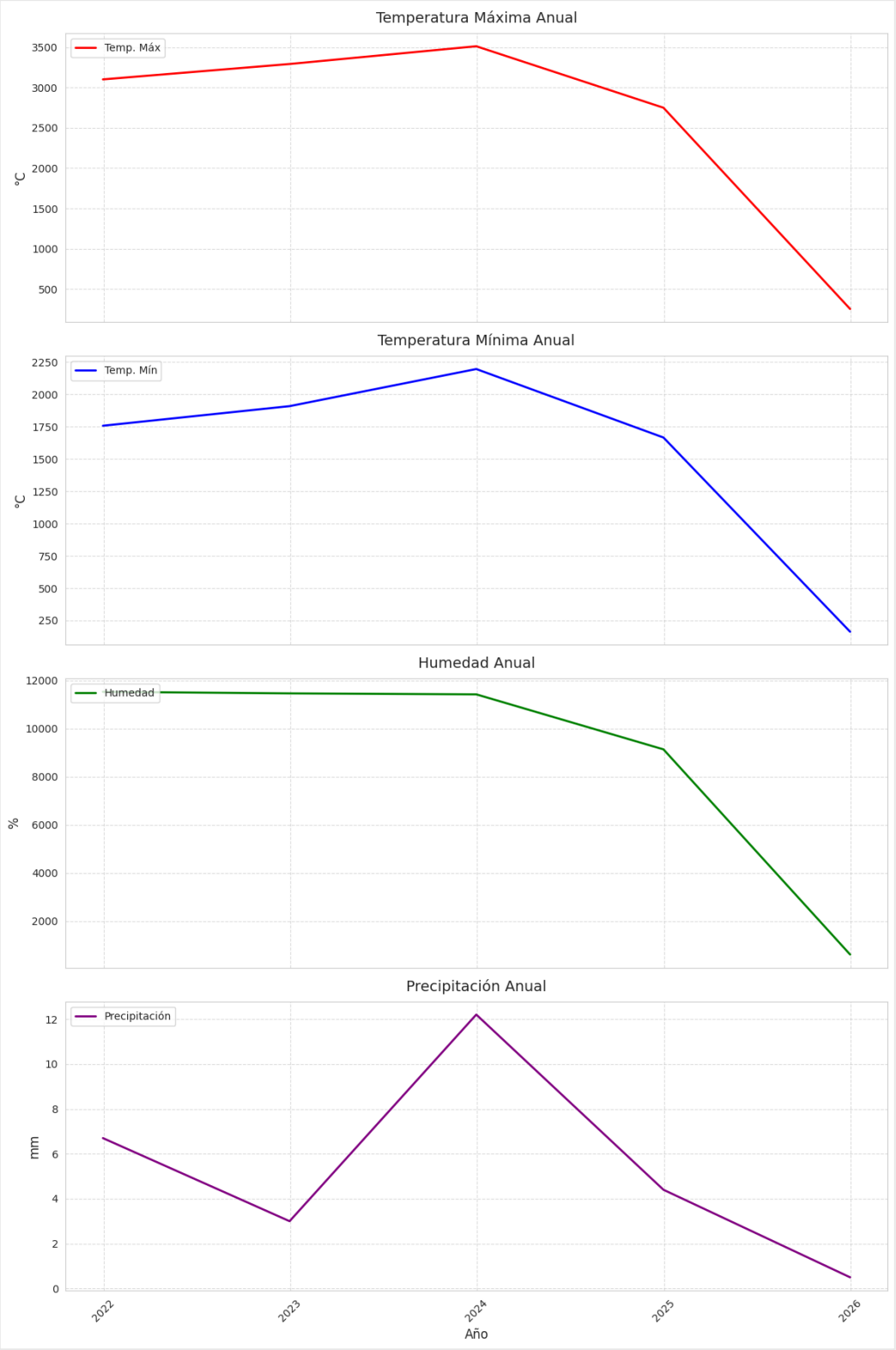
4. Visualización de Datos Reales

```
# 📌 Paso 4.1: Visualizar los datos por mes
data_mensual = data.resample('ME').sum()

fig, axes = plt.subplots(4, 1, figsize=(10, 15), sharex=True)
axes[0].plot(data_mensual.index, data_mensual['temp_max'], color='red',
label='Temp. Máx')
axes[0].set_title('Temperatura Máxima Mensual')
axes[0].set_ylabel('°C')
axes[1].plot(data_mensual.index, data_mensual['temp_min'], color='blue',
label='Temp. Mín')
axes[1].set_title('Temperatura Mínima Mensual')
axes[1].set_ylabel('°C')
axes[2].plot(data_mensual.index, data_mensual['humedad'], color='green',
label='Humedad')
axes[2].set_title('Humedad Mensual')
axes[2].set_ylabel('%')
axes[3].plot(data_mensual.index, data_mensual['precipitacion'],
color='purple', label='Precipitación')
axes[3].set_title('Precipitación Mensual')
axes[3].set_ylabel('mm')
for ax in axes:
    ax.legend()
    ax.grid(True, linestyle='--', alpha=0.7)
plt.xticks(rotation=45)
```

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

# 📌 Paso 4.2: Visualizar los datos Anual
data_anual = data.resample('YE').sum()

# Crear figura con subgráficos
fig, axes = plt.subplots(4, 1, figsize=(12, 18), sharex=True)

# Gráfico de Temperatura Máxima Anual
axes[0].plot(data_anual.index, data_anual['temp_max'], color='red',
label='Temp. Máx', linewidth=2)
axes[0].set_title('Temperatura Máxima Anual', fontsize=14, pad=10)
axes[0].set_ylabel('°C', fontsize=12)
axes[0].legend(loc='upper left', fontsize=10)
axes[0].grid(True, linestyle='--', alpha=0.7)

# Gráfico de Temperatura Mínima Anual
axes[1].plot(data_anual.index, data_anual['temp_min'], color='blue',
label='Temp. Mín', linewidth=2)
axes[1].set_title('Temperatura Mínima Anual', fontsize=14, pad=10)
axes[1].set_ylabel('°C', fontsize=12)
axes[1].legend(loc='upper left', fontsize=10)
axes[1].grid(True, linestyle='--', alpha=0.7)

# Gráfico de Humedad Anual
axes[2].plot(data_anual.index, data_anual['humedad'], color='green',
label='Humedad', linewidth=2)
axes[2].set_title('Humedad Anual', fontsize=14, pad=10)
axes[2].set_ylabel('%', fontsize=12)
axes[2].legend(loc='upper left', fontsize=10)
axes[2].grid(True, linestyle='--', alpha=0.7)

# Gráfico de Precipitación Anual
axes[3].plot(data_anual.index, data_anual['precipitacion'], color='purple',
label='Precipitación', linewidth=2)
axes[3].set_title('Precipitación Anual', fontsize=14, pad=10)
axes[3].set_ylabel('mm', fontsize=12)
axes[3].legend(loc='upper left', fontsize=10)
axes[3].grid(True, linestyle='--', alpha=0.7)

# Ajustar formato de fechas y etiquetas
plt.xlabel('Año', fontsize=12)
plt.xticks(rotation=45, fontsize=10)
axes[3].xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
axes[3].xaxis.set_major_locator(mdates.YearLocator())

# Ajustar diseño y mostrar
plt.tight_layout()
fig.suptitle('Visualización Anual de Variables Meteorológicas',
             fontsize=16, fontweight='bold', y=1.02)
plt.show()
```

Figura 34

Gráfico anual de variables meteorológicas para el modelo LSTM



Nota. La imagen presenta cuatro gráficos que muestran la evolución anual de la Temperatura Máxima, la Temperatura Mínima, la Humedad y la Precipitación desde el año 2022 hasta el 2026. Elaboración Propia.

5. División de Datos por fecha en entrenamiento y prueba

```
# 📌 Paso 5: División en conjunto de entrenamiento y prueba
fecha_inicio = '2021-01-01'
```

```
fecha_fin = '2025-03-31'
data.index = pd.to_datetime(data.index)
data_filtered = data.loc[fecha_inicio:fecha_fin].copy()

# División 80% entrenamiento, 20% prueba
split_index = int(len(data_filtered) * 0.8)
data_train = data_filtered.iloc[:split_index].copy()
data_test = data_filtered.iloc[split_index:].copy()

features = ['precipitacion', 'temp_max', 'temp_min', 'humedad']
```

6. Normalizar Datos

```
# 📌 Paso 6: Normalización de los datos
scaler = MinMaxScaler()
scaler.fit(data_train[features])
train_scaled = scaler.transform(data_train[features])
test_scaled = scaler.transform(data_test[features])
```

7. Crear secuencias y optimizar ventana de tiempo

```
# 📌 Paso 7.1: Definir función para crear secuencias de datos con fechas
def create_sequences(data, dates, n_steps=30, target_col=0):
    """Crea secuencias para modelos recurrentes."""
    if len(data) <= n_steps:
        raise ValueError("El tamaño de los datos es insuficiente para la
ventana temporal.")
    X, y, y_dates = [], [], []
    for i in range(n_steps, len(data)):
        X.append(data[i - n_steps:i, :])
        y.append(data[i, target_col])
        y_dates.append(dates[i])
    return np.array(X), np.array(y), np.array(y_dates)

# 📌 Paso 7.2: Optimizar ventana de tiempo
n_steps_options = [7, 15, 30, 60, 90]
best_n_steps = 30
best_mae = float('inf')

for n_steps in n_steps_options:
    X_train, y_train, y_dates_train = create_sequences(train_scaled,
data_train.index, n_steps)
    X_test, y_test, y_dates_test = create_sequences(test_scaled,
data_test.index, n_steps)
    X_train = X_train.reshape((X_train.shape[0], X_train.shape[1],
X_train.shape[2]))
    X_test = X_test.reshape((X_test.shape[0], X_test.shape[1],
X_test.shape[2]))
    mae = np.mean(np.abs(y_test - np.mean(y_test)))  # Placeholder
    if mae < best_mae:
        best_mae = mae
        best_n_steps = n_steps
    print(f"n_steps={n_steps}, X_train.shape={X_train.shape}, MAE
(simulado)={mae:.4f}")

n_steps = best_n_steps
X_train, y_train, y_dates_train = create_sequences(train_scaled,
data_train.index, n_steps)
X_test, y_test, y_dates_test = create_sequences(test_scaled, data_test.index,
n_steps)
```

```
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1],
X_train.shape[2]))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], X_test.shape[2]))
print(f"Mejor ventana de tiempo: {n_steps}")
```

8. Construcción del modelo LSTM

```
# 📌 Paso 8.1: Construcción del modelo LSTM
def build_lstm_model(input_shape):
    model = Sequential([
        Input(shape=input_shape),
        LSTM(128, return_sequences=True, activation='tanh',
recurrent_dropout=0.2,
             kernel_regularizer=l1_l2(l1=0.01, l2=0.01)),
        BatchNormalization(),
        Dropout(0.3),
        LSTM(64, return_sequences=False, activation='tanh',
recurrent_dropout=0.2,
             kernel_regularizer=l1_l2(l1=0.005, l2=0.005)),
        BatchNormalization(),
        Dropout(0.3),
        Dense(32, activation='relu'),
        Dropout(0.2),
        Dense(1, activation='linear')
    ])
    model.compile(optimizer=Adam(learning_rate=0.001, clipvalue=0.5),
loss='mse', metrics=['mae'])
    return model

input_shape = (X_train.shape[1], X_train.shape[2])
lstm_model = build_lstm_model(input_shape)
lstm_model.summary()

# 📌 Paso 8.2: Callbacks
callbacks = [
    EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True),
    ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5, verbose=1)
]
```

9. Entrenamiento y Predicción

```
# 📌 Paso 9.1: Entrenamiento del modelo
np.random.seed(42)
tf.random.set_seed(42)
start_time = time.time()
history = lstm_model.fit(X_train, y_train, validation_data=(X_test, y_test),
epochs=100,
                         batch_size=32, callbacks=callbacks, verbose=1)
end_time = time.time()
print(f"Tiempo total de entrenamiento: {end_time - start_time:.2f} segundos")

# 📌 Paso 9.2: Predicción
start_time = time.time()
y_pred_scaled = lstm_model.predict(X_test, verbose=0)
end_time = time.time()
print(f"Tiempo de predicción: {end_time - start_time:.2f} segundos")

# 📌 Paso 9.3: Desnormalización
```

```
y_pred = scaler.inverse_transform(np.hstack([y_pred_scaled,
np.zeros((len(y_pred_scaled), len(features)-1))]))[:, 0]
y_test_rescaled = scaler.inverse_transform(np.hstack([y_test.reshape(-1, 1),
np.zeros((len(y_test), len(features)-1))]))[:, 0]
```

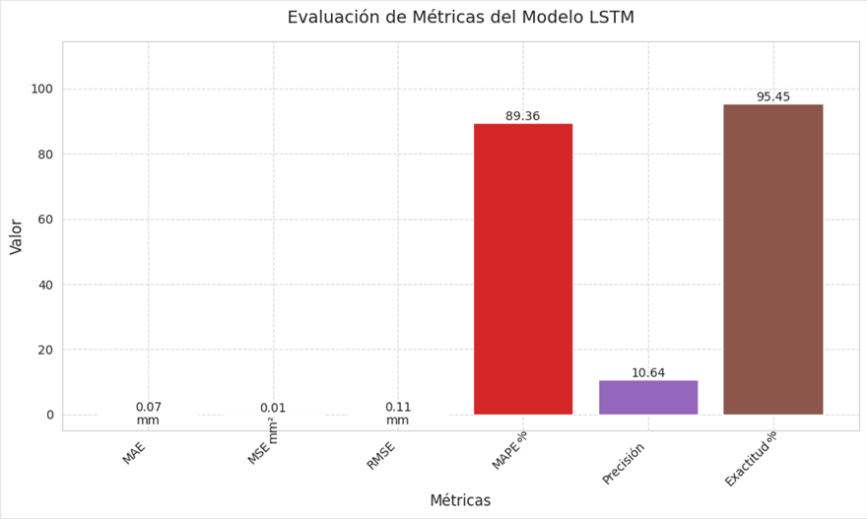
10. Calcular métricas

```
# 📌 Paso 10.1: Calcular métricas
def calcular_metricas(y_real, y_pred, threshold=0.5):
    """Calcula e imprime métricas de evaluación para el modelo LSTM.

    Parámetros:
    - y_real (array-like): Valores reales de salida.
    - y_pred (array-like): Valores predichos por el modelo.
    - threshold (float, optional): Umbral relativo para calcular exactitud
(default: 0.1, o 10%).

    Retorna:
    - metrics (dict): Diccionario con métricas de desempeño redondeadas.
    """
    y_real, y_pred = np.array(y_real).flatten(), np.array(y_pred).flatten()
    if len(y_real) != len(y_pred):
        raise ValueError("Longitudes de y_real y y_pred deben coincidir.")

    mae = mean_absolute_error(y_real, y_pred)
    mse = mean_squared_error(y_real, y_pred)
    rmse = np.sqrt(mse)
    non_zero_mask = y_real != 0
    mape = np.mean(np.abs((y_real[non_zero_mask] - y_pred[non_zero_mask]) /
y_real[non_zero_mask])) * 100 if np.any(non_zero_mask) else np.nan
    precision = 100 - mape if not np.isnan(mape) else np.nan

    # Calcular exactitud (porcentaje de predicciones dentro del umbral)
    max_value = np.max(y_real) if np.max(y_real) > 0 else 1.0  # Evitar
división por cero
    accuracy = np.mean(np.abs(y_real - y_pred) <= threshold * max_value) * 100
if max_value > 0 else np.nan

    metrics = {
        'MAE': round(mae, 4), 'MSE': round(mse, 4), 'RMSE': round(rmse, 4),
        'MAPE': round(mape, 2) if not np.isnan(mape) else np.nan,
        'Precisión': round(precision, 2) if not np.isnan(precision) else
np.nan,
        'Exactitud': round(accuracy, 2) if not np.isnan(accuracy) else np.nan
    }

    print("\n📊 EVALUACIÓN DEL MODELO LSTM")
    print("="*50)
    for metric, value in metrics.items():
        unit = 'mm' if 'MAE' in metric or 'RMSE' in metric else 'mm²' if 'MSE'
in metric else '%' if 'MAPE' in metric or 'Precisión' in metric or 'Exactitud'
in metric else ''
        print(f"{metric:15} {value:>10} {unit}")
    print("="*50)
    return metrics
metricas = calcular_metricas(y_test_rescaled, y_pred)

# 📌 Paso 10.2: Generar gráfico de métricas
metrics_names = list(metricas.keys())
```

```
metrics_values = list(metricas.values())

# Configurar el gráfico
plt.figure(figsize=(10, 6))
bars = plt.bar(metrics_names, metrics_values, color=['#1f77b4', '#ff7f0e',
'#2ca02c', '#d62728', '#9467bd', '#8c564b'])
plt.title('Evaluación de Métricas del Modelo LSTM', fontsize=14, pad=15)
plt.xlabel('Métricas', fontsize=12)
plt.ylabel('Valor', fontsize=12)

# Añadir etiquetas de valor en las barras
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2., height,
             f'{height:.2f}' if not np.isnan(height) else 'N/A',
             ha='center', va='bottom' if height > 0 else 'top')

# Añadir unidades como anotaciones
units = ['mm', 'mm²', 'mm', '%', '', '%']
for i, unit in enumerate(units):
    plt.text(i, -0.1 if i in [0, 2] else -0.05 if i == 1 else -5 if i in [3,
5] else -0.2,
             unit, ha='center', va='top', rotation=0 if i in [0, 2] else 90)

# Ajustar límites del eje y
plt.ylim(min(-5, min(metrics_values) * 1.2), max(metrics_values) * 1.2)

# Añadir rejilla y estilo
plt.grid(True, linestyle='--', alpha=0.7)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()

# Mostrar gráfico
plt.show()
```

Figura 35

Evaluación de métricas del modelo LSTM



Nota. El grafico de Evaluación de las métricas del Modelo LSTM. Elaboración propia.

11. Visualización general de variables meteorológicas

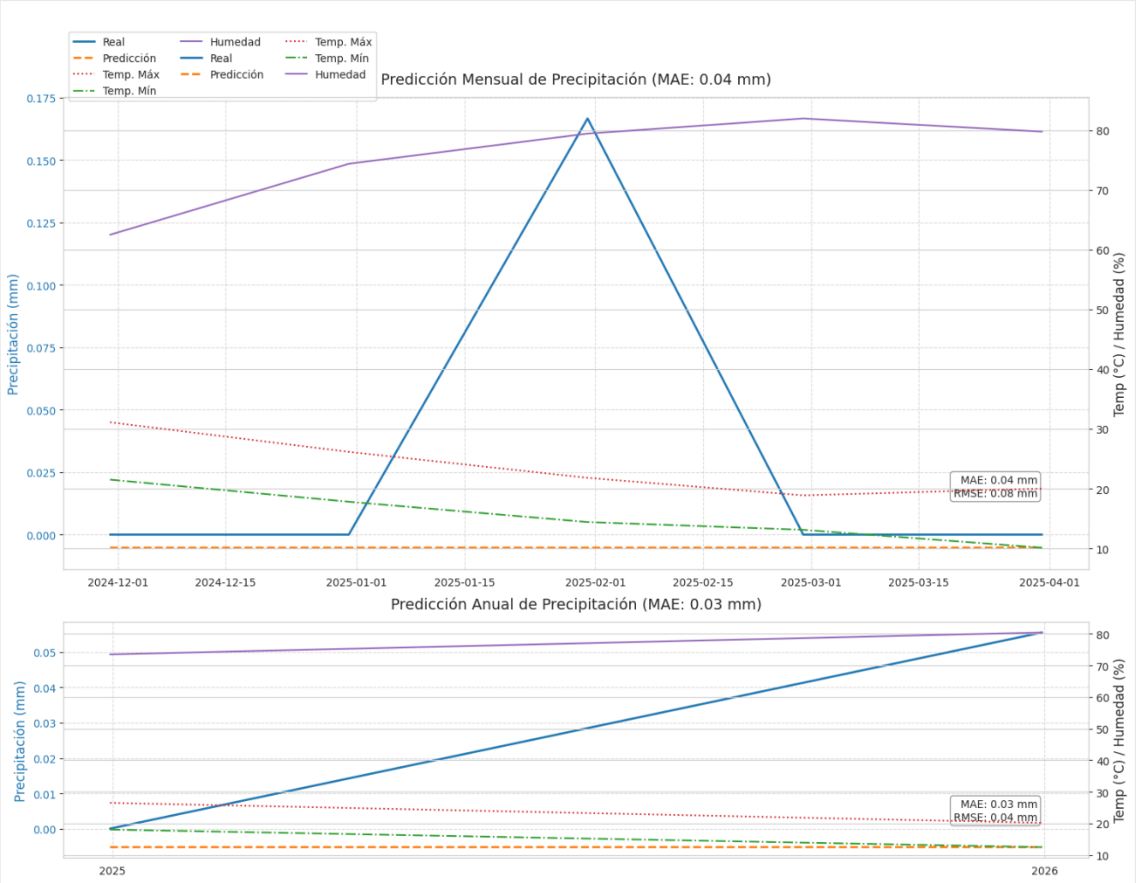
```
# 📌 Paso 11.1: Visualización general de variables meteorológicas
# Asegurar consistencia en la longitud de los arrays
n_samples = min(len(y_pred), len(y_test_rescaled), len(y_dates_test),
len(data_test))

# Crear DataFrames con resampleo para diferentes escalas temporales
plot_data = {
    'Real': y_test_rescaled[:n_samples],
    'Predicción': y_pred[:n_samples],
    'Temp Max': data_test['temp_max'].values[:n_samples],
    'Temp Min': data_test['temp_min'].values[:n_samples],
    'Humedad': data_test['humedad'].values[:n_samples]
}

data_monthly = pd.DataFrame(plot_data,
index=y_dates_test[:n_samples]).resample('ME').mean()
data_annual = pd.DataFrame(plot_data,
index=y_dates_test[:n_samples]).resample('YE').mean()

# Calcular métricas de exactitud (solo MAE y RMSE)
def calcular_metricas(reales, pred):
    return {
        'MAE': mean_absolute_error(reales, pred),
        'RMSE': math.sqrt(mean_squared_error(reales, pred))
    }

metrics_monthly = calcular_metricas(data_monthly['Real'],
data_monthly['Predicción'])
metrics_annual = calcular_metricas(data_annual['Real'],
data_annual['Predicción'])



# 2. Configuración de la figura
fig, (ax_monthly, ax_annual) = plt.subplots(
    2, 1,
    figsize=(15, 12),
```

```
    sharex=False,
    gridspec_kw={'height_ratios': [2, 1]}
)

# Estilo consistente para las gráficas
plot_config = {
    'Real': {'color': '#1f77b4', 'linewidth': 2, 'label': 'Real'},
    'Predicción': {'color': '#ff7f0e', 'linestyle': '--', 'linewidth': 2,
'label': 'Predicción'},
    'Temp Max': {'color': '#d62728', 'linestyle': ':', 'linewidth': 1.5,
'label': 'Temp. Máx'},
    'Temp Min': {'color': '#2ca02c', 'linestyle': '-.', 'linewidth': 1.5,
'label': 'Temp. Mín'},
    'Humedad': {'color': '#9467bd', 'linestyle': '-', 'linewidth': 1.5,
'label': 'Humedad'}
}

# 3. Gráfico Mensual con métricas
# Precipitación (eje principal)
ax_monthly.plot(data_monthly.index, data_monthly['Real'],
**plot_config['Real'])
ax_monthly.plot(data_monthly.index, data_monthly['Predicción'],
**plot_config['Predicción'])
ax_monthly.set_ylabel("Precipitación (mm)",
color=plot_config['Real']['color'], fontsize=12)
ax_monthly.tick_params(axis='y', labelcolor=plot_config['Real']['color'])

# Variables climáticas (eje secundario)
ax_monthly_clim = ax_monthly.twinx()
for var in ['Temp Max', 'Temp Min', 'Humedad']:
    ax_monthly_clim.plot(data_monthly.index, data_monthly[var],
**plot_config[var])
ax_monthly_clim.set_ylabel("Temp (°C) / Humedad (%)", fontsize=12)

# Añadir métricas como texto
metrics_text_monthly = (
    f"MAE: {metrics_monthly['MAE']:.2f} mm\n"
    f"RMSE: {metrics_monthly['RMSE']:.2f} mm"
)
ax_monthly.text(0.95, 0.15, metrics_text_monthly,
                transform=ax_monthly.transAxes,
                ha='right', va='bottom',
                bbox=dict(facecolor='white', alpha=0.8, edgecolor='gray',
boxstyle='round'),
                fontsize=10)

ax_monthly.grid(True, linestyle='--', alpha=0.7)
ax_monthly.set_title('Predicción Mensual de Precipitación (MAE: {:.2f}
mm)'.format(
    metrics_monthly['MAE']), fontsize=14, pad=12)

# 4. Gráfico Anual con métricas
# Precipitación (eje principal)
ax_annual.plot(data_annual.index, data_annual['Real'], **plot_config['Real'])
ax_annual.plot(data_annual.index, data_annual['Predicción'],
**plot_config['Predicción'])
ax_annual.set_ylabel("Precipitación (mm)", color=plot_config['Real']['color'],
fontsize=12)
ax_annual.tick_params(axis='y', labelcolor=plot_config['Real']['color'])
```

```
# Variables climáticas (eje secundario)
ax_annual_clim = ax_annual.twinx()
for var in ['Temp Max', 'Temp Min', 'Humedad']:
    ax_annual_clim.plot(data_annual.index, data_annual[var],
**plot_config[var])
ax_annual_clim.set_ylabel("Temp (°C) / Humedad (%)", fontsize=12)

# Añadir métricas como texto
metrics_text_annual = (
    f"MAE: {metrics_annual['MAE']:.2f} mm\n"
    f"RMSE: {metrics_annual['RMSE']:.2f} mm"
)
ax_annual.text(0.95, 0.15, metrics_text_annual,
               transform=ax_annual.transAxes,
               ha='right', va='bottom',
               bbox=dict(facecolor='white', alpha=0.8, edgecolor='gray',
boxstyle='round'),
               fontsize=10)

# Configuración de ejes temporales
ax_annual.xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
ax_annual.xaxis.set_major_locator(mdates.YearLocator())
ax_annual.grid(True, linestyle='--', alpha=0.7)
ax_annual.set_title('Predicción Anual de Precipitación (MAE: {:.2f}
mm)'.format(
    metrics_annual['MAE']), fontsize=14, pad=12)

# 5. Elementos comunes y formato final
# Leyenda unificada
lines = []
labels = []
for ax in [ax_monthly, ax_monthly_clim, ax_annual, ax_annual_clim]:
    l, lab = ax.get_legend_handles_labels()
    lines.extend(l)
    labels.extend(lab)

ax_monthly.legend(lines, labels,
                  loc='upper left',
                  fontsize=10,
                  frameon=True,
                  bbox_to_anchor=(0, 1.15),
                  ncol=3)

# Título general
fig.suptitle('Análisis Comparativo de Predicciones de Precipitación con
Métricas de Error',
             fontsize=16,
             fontweight='bold',
             y=1.05)

# Ajustes finales
plt.xticks(rotation=45, fontsize=10)
plt.xlabel('Fecha', fontsize=12)
plt.tight_layout()
plt.subplots_adjust(top=0.85)
plt.show()
```

Figura 36

Predicciones de precipitación con métricas de error del modelo LSTM



Nota. El gráfico muestra un análisis comparativo de predicciones de precipitación con métricas de error del modelo LSTM. Elaboración propia.

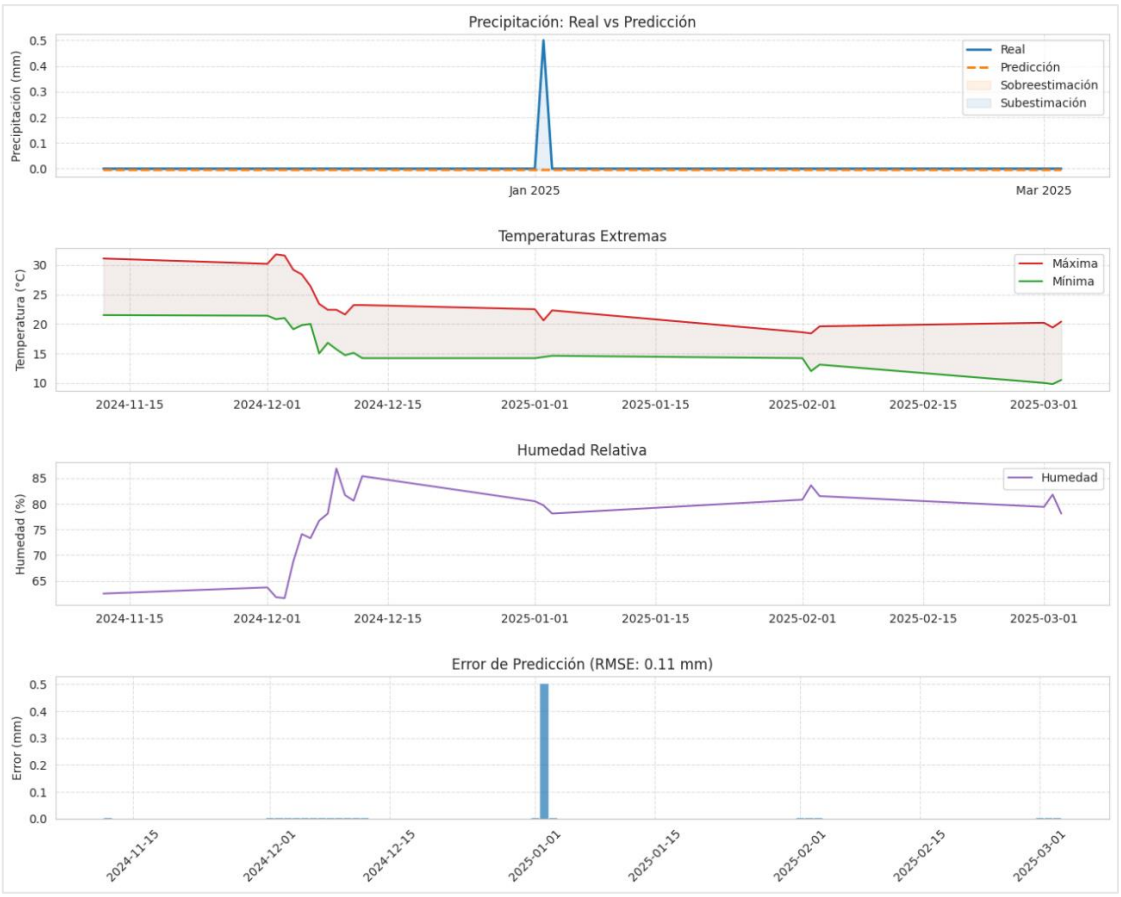
12. Visualización detallada de variables meteorológicas

```
# 📌 Paso 12: Visualización general de variables meteorológicas
fig = plt.figure(figsize=(16, 12))
gs = GridSpec(4, 1, figure=fig, hspace=0.5)
ax1 = fig.add_subplot(gs[0])
ax1.plot(y_dates_test[:n_samples], y_test_rescaled[:n_samples], label='Real',
color='#1f77b4', linewidth=2)
ax1.plot(y_dates_test[:n_samples], y_pred[:n_samples], label='Predicción',
color='#ff7f0e', linestyle='--', linewidth=2)
ax1.fill_between(y_dates_test[:n_samples], y_test_rescaled[:n_samples],
y_pred[:n_samples], where=(y_pred[:n_samples] > y_test_rescaled[:n_samples]),
                 color='#ff7f0e', alpha=0.1, label='Sobreestimación')
ax1.fill_between(y_dates_test[:n_samples], y_test_rescaled[:n_samples],
y_pred[:n_samples], where=(y_pred[:n_samples] <= y_test_rescaled[:n_samples]),
                 color='#1f77b4', alpha=0.1, label='Subestimación')
ax1.set_title('Precipitación: Real vs Predicción')
ax1.set_ylabel('Precipitación (mm)')
ax1.legend()
ax1.xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
ax1.xaxis.set_major_locator(mdates.MonthLocator(interval=2))
ax1.grid(True, linestyle='--', alpha=0.6)
```

```
ax2 = fig.add_subplot(gs[1])
ax2.plot(y_dates_test[:n_samples], data_test['temp_max'].values[:n_samples],
label='Máxima', color='#d62728')
ax2.plot(y_dates_test[:n_samples], data_test['temp_min'].values[:n_samples],
label='Mínima', color='#2ca02c')
ax2.fill_between(y_dates_test[:n_samples],
data_test['temp_min'].values[:n_samples],
data_test['temp_max'].values[:n_samples],
                 color='#8c564b', alpha=0.1)
ax2.set_title('Temperaturas Extremas')
ax2.set_ylabel('Temperatura (°C)')
ax2.legend()
ax2.grid(True, linestyle='--', alpha=0.6)

ax3 = fig.add_subplot(gs[2])
ax3.plot(y_dates_test[:n_samples], data_test['humedad'].values[:n_samples],
label='Humedad', color='#9467bd')
ax3.set_title('Humedad Relativa')
ax3.set_ylabel('Humedad (%)')
ax3.legend()
ax3.grid(True, linestyle='--', alpha=0.6)

ax4 = fig.add_subplot(gs[3])
error = y_test_rescaled[:n_samples] - y_pred[:n_samples]
ax4.bar(y_dates_test[:n_samples], error, width=1, color=np.where(error > 0,
'#1f77b4', '#ff7f0e'), alpha=0.7)
ax4.axhline(0, color='black', linewidth=0.8)
ax4.set_title(f'Error de Predicción (RMSE: {np.sqrt(np.mean(error**2)):.2f}
mm)')
ax4.set_ylabel('Error (mm)')
ax4.grid(True, linestyle='--', alpha=0.6)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Figura 37

Visualización detallada de variables meteorológicas del modelo LSTM



Nota. El gráfico muestra una visualización detallada de variables meteorológicas del modelo LSTM. Elaboración propia

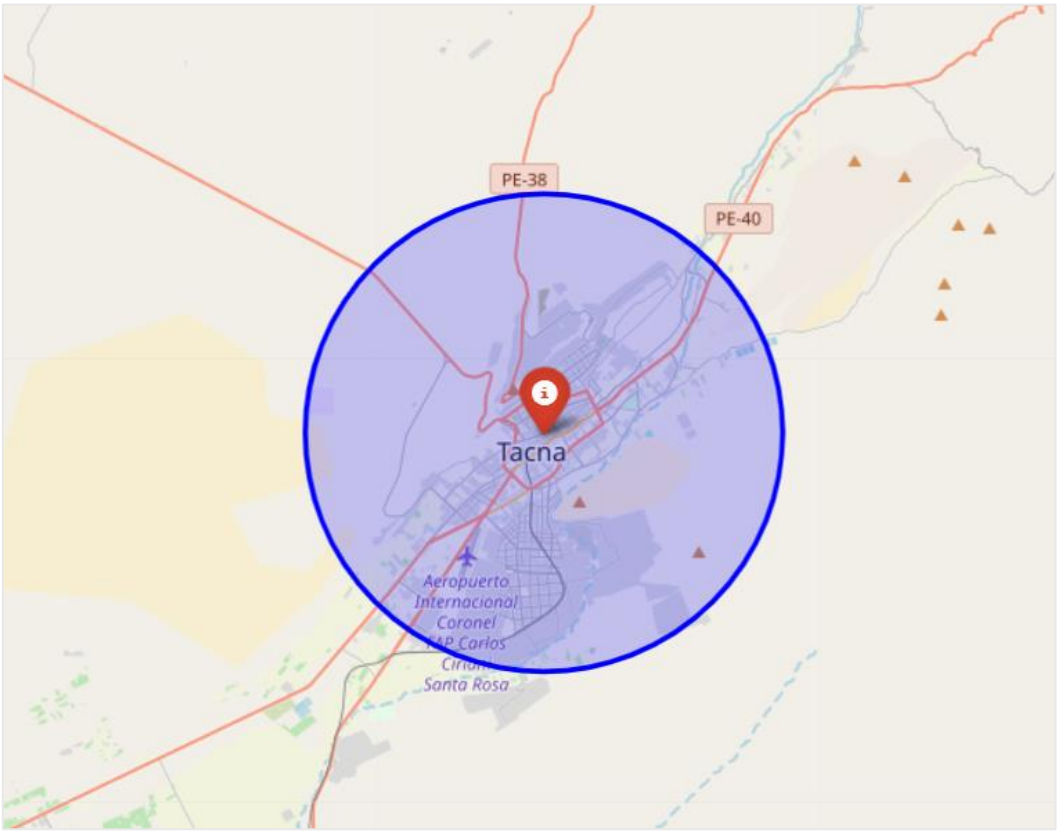
13. Mapa de la Estación Meteorológica Jorge Basadre

```
# 📌 Paso 13: Agregar mapa con la ubicación de la estación meteorológica Jorge Basadre
# Coordenadas aproximadas de la estación Jorge Basadre, Tacna.
latitude = -18.0066
longitude = -70.2463

# Crear un mapa centrado en Tacna
mapa = folium.Map(location=[-18.0066, -70.2463], zoom_start=10)

# Agregar marcador de la estación
folium.Marker(
    location=[latitude, longitude],
    popup='Estación Meteorológica Jorge Basadre',
    icon=folium.Icon(color='red')
).add_to(mapa)

# Resaltar la ciudad de Tacna con un círculo
folium.Circle(
    location=[-18.0066, -70.2463],
    radius=10000,
    color='blue',
    fill=True,
    fill_color='blue',
    fill_opacity=0.2,
```

```
    popup='Ciudad de Tacna'
).add_to(mapa)

# Mostrar el mapa
Mapa
```

Figura 38

Mapa de la estación meteorológica Jorge Basadre



Nota. El gráfico muestra la ubicación de Estación Meteorológica Jorge Basadre. Elaboración propia

Implementación del Modelo GRU

1. Instalar e importar Librerías

```
# 📌 Paso 1.1: Instalar librerías
!pip install -q numpy pandas matplotlib scikit-learn tensorflow folium

# 📌 Paso 1.2: Importar librerías
import numpy as np
import pandas as pd
import folium
import math
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from matplotlib.gridspec import GridSpec
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, GRU, Dense, Dropout,
BatchNormalization
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.regularizers import l1_l2
import tensorflow as tf
from google.colab import drive
import time
```

2. Montar Google Drive

```
# 📌 Paso 2: Montar Google Drive
drive.mount('/content/drive')
```

3. Cargar Datos y seleccionar variables

```
# 📌 Paso 3.1: Cargar datos
dataset_path = "/content/drive/MyDrive/TESIS MODELOS DE MACHINE LEARNING
V4/Data/datasetlimpia.csv" # Ajusta la ruta si es necesario
data = pd.read_csv(dataset_path, parse_dates=['fecha'], dayfirst=True)
data = data.sort_values('fecha')
print("Primeros 5 registros del dataset:")
print(data.head())

# 📌 Paso 3.2: Seleccionar y preparar variables relevantes
data = data[['fecha', 'precipitacion', 'tem max', 'tem min',
'humedad']].dropna()
data.columns = ['fecha', 'precipitacion', 'temp_max', 'temp_min',
'humedad']  # Renombrar columnas
data.set_index('fecha', inplace=True)

# 📌 Paso 3.3: Validar rango de fechas hasta el 31 de Marzo de 2025
data = data[data.index <= '2025-03-31']
```

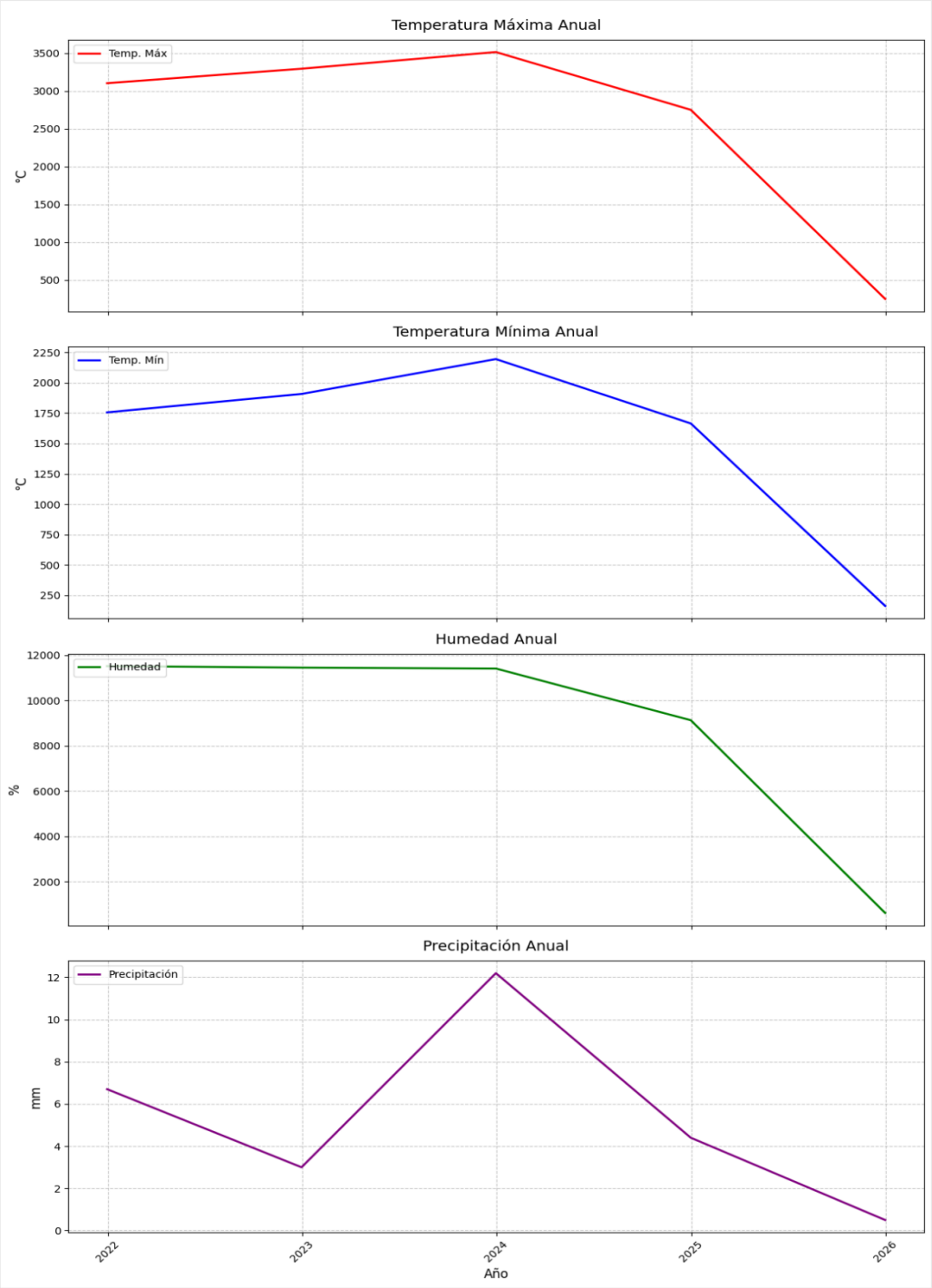
4. Visualización de Datos Reales

```
# 📌 Paso 4.1: Visualizar los datos por mes
data_mensual = data.resample('ME').sum()

fig, axes = plt.subplots(4, 1, figsize=(10, 15), sharex=True)
axes[0].plot(data_mensual.index, data_mensual['temp_max'], color='red',
label='Temp. Máx')
axes[0].set_title('Temperatura Máxima Mensual')
axes[0].set_ylabel('°C')
axes[1].plot(data_mensual.index, data_mensual['temp_min'], color='blue',
label='Temp. Mín')
axes[1].set_title('Temperatura Mínima Mensual')
axes[1].set_ylabel('°C')
axes[2].plot(data_mensual.index, data_mensual['humedad'], color='green',
label='Humedad')
axes[2].set_title('Humedad Mensual')
axes[2].set_ylabel('%')
axes[3].plot(data_mensual.index, data_mensual['precipitacion'],
color='purple', label='Precipitación')
axes[3].set_title('Precipitación Mensual')
axes[3].set_ylabel('mm')
for ax in axes:
    ax.legend()
    ax.grid(True, linestyle='--', alpha=0.7)
```

```
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# 📌 Paso: Visualizar los datos Anual
data_anual = data.resample('YE').sum()

# Crear figura con subgráficos
fig, axes = plt.subplots(4, 1, figsize=(12, 18), sharex=True)

# Gráfico de Temperatura Máxima Anual
axes[0].plot(data_anual.index, data_anual['temp_max'], color='red',
label='Temp. Máx', linewidth=2)
axes[0].set_title('Temperatura Máxima Anual', fontsize=14, pad=10)
axes[0].set_ylabel('°C', fontsize=12)
axes[0].legend(loc='upper left', fontsize=10)
axes[0].grid(True, linestyle='--', alpha=0.7)

# Gráfico de Temperatura Mínima Anual
axes[1].plot(data_anual.index, data_anual['temp_min'], color='blue',
label='Temp. Mín', linewidth=2)
axes[1].set_title('Temperatura Mínima Anual', fontsize=14, pad=10)
axes[1].set_ylabel('°C', fontsize=12)
axes[1].legend(loc='upper left', fontsize=10)
axes[1].grid(True, linestyle='--', alpha=0.7)

# Gráfico de Humedad Anual
axes[2].plot(data_anual.index, data_anual['humedad'], color='green',
label='Humedad', linewidth=2)
axes[2].set_title('Humedad Anual', fontsize=14, pad=10)
axes[2].set_ylabel('%', fontsize=12)
axes[2].legend(loc='upper left', fontsize=10)
axes[2].grid(True, linestyle='--', alpha=0.7)

# Gráfico de Precipitación Anual
axes[3].plot(data_anual.index, data_anual['precipitacion'], color='purple',
label='Precipitación', linewidth=2)
axes[3].set_title('Precipitación Anual', fontsize=14, pad=10)
axes[3].set_ylabel('mm', fontsize=12)
axes[3].legend(loc='upper left', fontsize=10)
axes[3].grid(True, linestyle='--', alpha=0.7)

# Ajustar formato de fechas y etiquetas
plt.xlabel('Año', fontsize=12)
plt.xticks(rotation=45, fontsize=10)
axes[3].xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
axes[3].xaxis.set_major_locator(mdates.YearLocator())

# Ajustar diseño y mostrar
plt.tight_layout()
fig.suptitle('Visualización Anual de Variables Meteorológicas',
             fontsize=16, fontweight='bold', y=1.02)
plt.show()
```

Figura 39

Gráfico anual de variables meteorológicas para el modelo GRU



Nota. La imagen presenta cuatro gráficos que muestran la evolución anual de la Temperatura Máxima, la Temperatura Mínima, la Humedad y la Precipitación desde el año 2022 hasta el 2026. Elaboración Propia.

5. División de Datos por fecha en entrenamiento y prueba

```
# 📌 Paso 5: División en conjunto de entrenamiento y prueba
fecha_inicio = '2021-01-01'
fecha_fin = '2025-03-31'
data.index = pd.to_datetime(data.index)
data_filtered = data.loc[fecha_inicio:fecha_fin].copy()

# División 80% entrenamiento, 20% prueba
```

```
split_index = int(len(data_filtered) * 0.8)
data_train = data_filtered.iloc[:split_index].copy()
data_test = data_filtered.iloc[split_index:].copy()

features = ['precipitacion', 'temp_max', 'temp_min', 'humedad']
```

6. Normalizar Datos

```
# 📌 Paso 6: Normalización de datos
scaler = MinMaxScaler()
scaler.fit(data_train[features])
train_scaled = scaler.transform(data_train[features])
test_scaled = scaler.transform(data_test[features])
```

7. Crear secuencias y optimizar ventana de tiempo

```
# 📌 Paso 7.1: Crear secuencias
def create_sequences(data, dates, n_steps=30, target_col=0):
    """Crea secuencias para modelos recurrentes."""
    if len(data) <= n_steps:
        raise ValueError("El tamaño de los datos es insuficiente para la
ventana temporal.")
    X, y, y_dates = [], [], []
    for i in range(n_steps, len(data)):
        X.append(data[i - n_steps:i, :])
        y.append(data[i, target_col])
        y_dates.append(dates[i])
    return np.array(X), np.array(y), np.array(y_dates)

# 📌 Paso 7.2. Optimizar ventana de tiempo
n_steps_options = [7, 15, 30, 60, 90]
best_n_steps = 30
best_mae = float('inf')
for n_steps in n_steps_options:
    X_train, y_train, y_dates_train = create_sequences(train_scaled,
data_train.index, n_steps)
    X_test, y_test, y_dates_test = create_sequences(test_scaled,
data_test.index, n_steps)
    X_train = X_train.reshape((X_train.shape[0], X_train.shape[1],
X_train.shape[2]))
    X_test = X_test.reshape((X_test.shape[0], X_test.shape[1],
X_test.shape[2]))
    mae = np.mean(np.abs(y_test - np.mean(y_test)))  # Placeholder
    if mae < best_mae:
        best_mae = mae
        best_n_steps = n_steps
    print(f"n_steps={n_steps}, X_train.shape={X_train.shape}, MAE
(simulado)={mae:.4f}")

n_steps = best_n_steps
X_train, y_train, y_dates_train = create_sequences(train_scaled,
data_train.index, n_steps)
X_test, y_test, y_dates_test = create_sequences(test_scaled, data_test.index,
n_steps)
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1],
X_train.shape[2]))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], X_test.shape[2]))
print(f"Mejor ventana de tiempo: {n_steps}")
```

8. Construcción del modelo GRU

```
# 📌 Paso 8.1: Construir Modelo GRU
def build_gru_model(input_shape):
    model = Sequential([
        Input(shape=input_shape),
        GRU(128, return_sequences=True, activation='tanh',
recurrent_dropout=0.2,
            kernel_regularizer=l1_l2(l1=0.01, l2=0.01)),
        BatchNormalization(),
        Dropout(0.3),
        GRU(64, return_sequences=False, activation='tanh',
recurrent_dropout=0.2,
            kernel_regularizer=l1_l2(l1=0.005, l2=0.005)),
        BatchNormalization(),
        Dropout(0.3),
        Dense(32, activation='relu'),
        Dropout(0.2),
        Dense(1, activation='linear')
    ])
    model.compile(optimizer=Adam(learning_rate=0.001, clipvalue=0.5),
loss='mse', metrics=['mae'])
    return model

input_shape = (X_train.shape[1], X_train.shape[2])
gru_model = build_gru_model(input_shape)
gru_model.summary()

# 📌 Paso 8.2: Callbacks
callbacks = [
    EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True),
    ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5, verbose=1)
]
```

9. Entrenamiento y Predicción

```
# 📌 Paso 9.1: Entrenamiento
np.random.seed(42)
tf.random.set_seed(42)
start_time = time.time()
history = gru_model.fit(X_train, y_train, validation_data=(X_test, y_test),
epochs=100,
                        batch_size=32, callbacks=callbacks, verbose=1)
end_time = time.time()
print(f"Tiempo total de entrenamiento: {end_time - start_time:.2f} segundos")

# 📌 Paso 9.2: Predicción
start_time = time.time()
y_pred_scaled = gru_model.predict(X_test, verbose=0)
end_time = time.time()
print(f"Tiempo de predicción: {end_time - start_time:.2f} segundos")

# 📌 Paso 9.2: Desnormalización
y_pred = scaler.inverse_transform(np.hstack([y_pred_scaled,
np.zeros((len(y_pred_scaled), len(features)-1))]))[:, 0]
y_test_rescaled = scaler.inverse_transform(np.hstack([y_test.reshape(-1, 1),
np.zeros((len(y_test), len(features)-1))]))[:, 0]
```

10. Calcular métricas

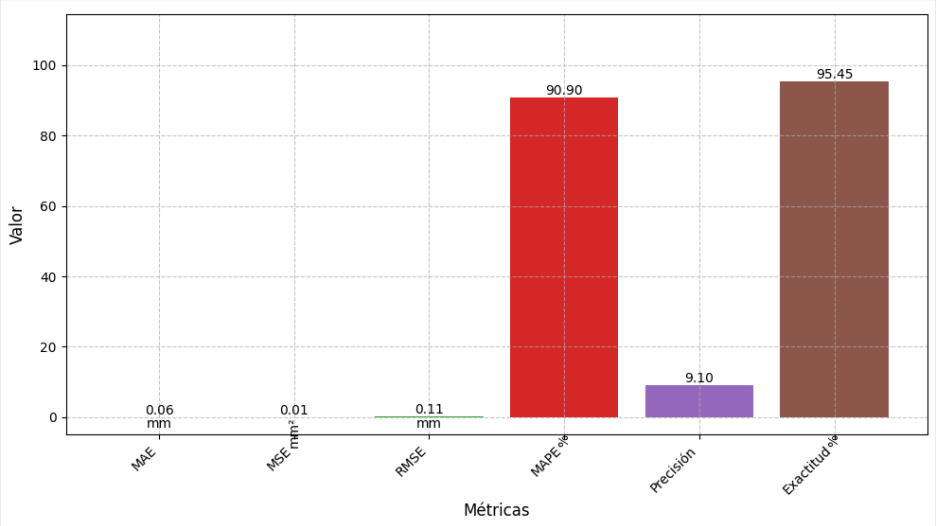
```
# 📌 Paso 10.1: Calcular métricas
```

```
def calcular_metricas(y_real, y_pred, threshold=0.5):
    """Calcula e imprime métricas de evaluación para el modelo GRU.
    Parámetros:
    - y_real (array-like): Valores reales de salida.
    - y_pred (array-like): Valores predichos por el modelo.
    - threshold (float, optional): Umbral relativo para calcular exactitud
(default: 0.1, o 10%).
    Retorna:
    - metrics (dict): Diccionario con métricas de desempeño redondeadas.
    """
    y_real, y_pred = np.array(y_real).flatten(), np.array(y_pred).flatten()
    if len(y_real) != len(y_pred):
        raise ValueError("Longitudes de y_real y y_pred deben coincidir.")
    mae = mean_absolute_error(y_real, y_pred)
    mse = mean_squared_error(y_real, y_pred)
    rmse = np.sqrt(mse)
    non_zero_mask = y_real != 0
    mape = np.mean(np.abs((y_real[non_zero_mask] - y_pred[non_zero_mask]) /
y_real[non_zero_mask])) * 100 if np.any(non_zero_mask) else np.nan
    precision = 100 - mape if not np.isnan(mape) else np.nan

    # Calcular exactitud (porcentaje de predicciones dentro del umbral)
    max_value = np.max(y_real) if np.max(y_real) > 0 else 1.0  # Evitar
división por cero
    accuracy = np.mean(np.abs(y_real - y_pred) <= threshold * max_value) * 100
if max_value > 0 else np.nan
    metrics = {
        'MAE': round(mae, 4), 'MSE': round(mse, 4), 'RMSE': round(rmse, 4),
        'MAPE': round(mape, 2) if not np.isnan(mape) else np.nan,
        'Precisión': round(precision, 2) if not np.isnan(precision) else
np.nan,
        'Exactitud': round(accuracy, 2) if not np.isnan(accuracy) else np.nan
    }
    print("\n📊 EVALUACIÓN DEL MODELO GRU")
    print("="*50)
    for metric, value in metrics.items():
        unit = 'mm' if 'MAE' in metric or 'RMSE' in metric else 'mm²' if 'MSE'
in metric else '%' if 'MAPE' in metric or 'Precisión' in metric or 'Exactitud'
in metric else ''
        print(f"{metric:15} {value:>10} {unit}")
    print("="*50)
    return metrics

metricas = calcular_metricas(y_test_rescaled, y_pred)
# 📌 Paso 10.2: Generar gráfico de métricas
metrics_names = list(metricas.keys())
metrics_values = list(metricas.values())

# Configurar el gráfico
plt.figure(figsize=(10, 6))
bars = plt.bar(metrics_names, metrics_values, color=['#1f77b4', '#ff7f0e',
'#2ca02c', '#d62728', '#9467bd', '#8c564b'])
plt.title('Evaluación de Métricas del Modelo GRU', fontsize=14, pad=15)
plt.xlabel('Métricas', fontsize=12)
plt.ylabel('Valor', fontsize=12)

# Añadir etiquetas de valor en las barras
for bar in bars:
```

```
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2., height,
             f'{height:.2f}' if not np.isnan(height) else 'N/A',
             ha='center', va='bottom' if height > 0 else 'top')

# Añadir unidades como anotaciones
units = ['mm', 'mm²', 'mm', '%', '', '%']
for i, unit in enumerate(units):
    plt.text(i, -0.1 if i in [0, 2] else -0.05 if i == 1 else -5 if i in [3,
5] else -0.2,
             unit, ha='center', va='top', rotation=0 if i in [0, 2] else 90)

# Ajustar límites del eje y
plt.ylim(min(-5, min(metrics_values) * 1.2), max(metrics_values) * 1.2)

# Añadir rejilla y estilo
plt.grid(True, linestyle='--', alpha=0.7)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()

# Mostrar gráfico
plt.show()
```

Figura 40

Evaluación de métricas del modelo GRU



Nota. El grafico de Evaluación de las métricas del Modelo GRU. Elaboración propia.

11. Visualización general de variables meteorológicas

```
# 📌 Paso 11.1: Visualización general de variables meteorológicas
# Asegurar consistencia en la longitud de los arrays
n_samples = min(len(y_pred), len(y_test_rescaled), len(y_dates_test),
len(data_test))

# Crear DataFrames con resampleo para diferentes escalas temporales
plot_data = {
    'Real': y_test_rescaled[:n_samples],
    'Predicción': y_pred[:n_samples],
    'Temp Max': data_test['temp_max'].values[:n_samples],
    'Temp Min': data_test['temp_min'].values[:n_samples],
```

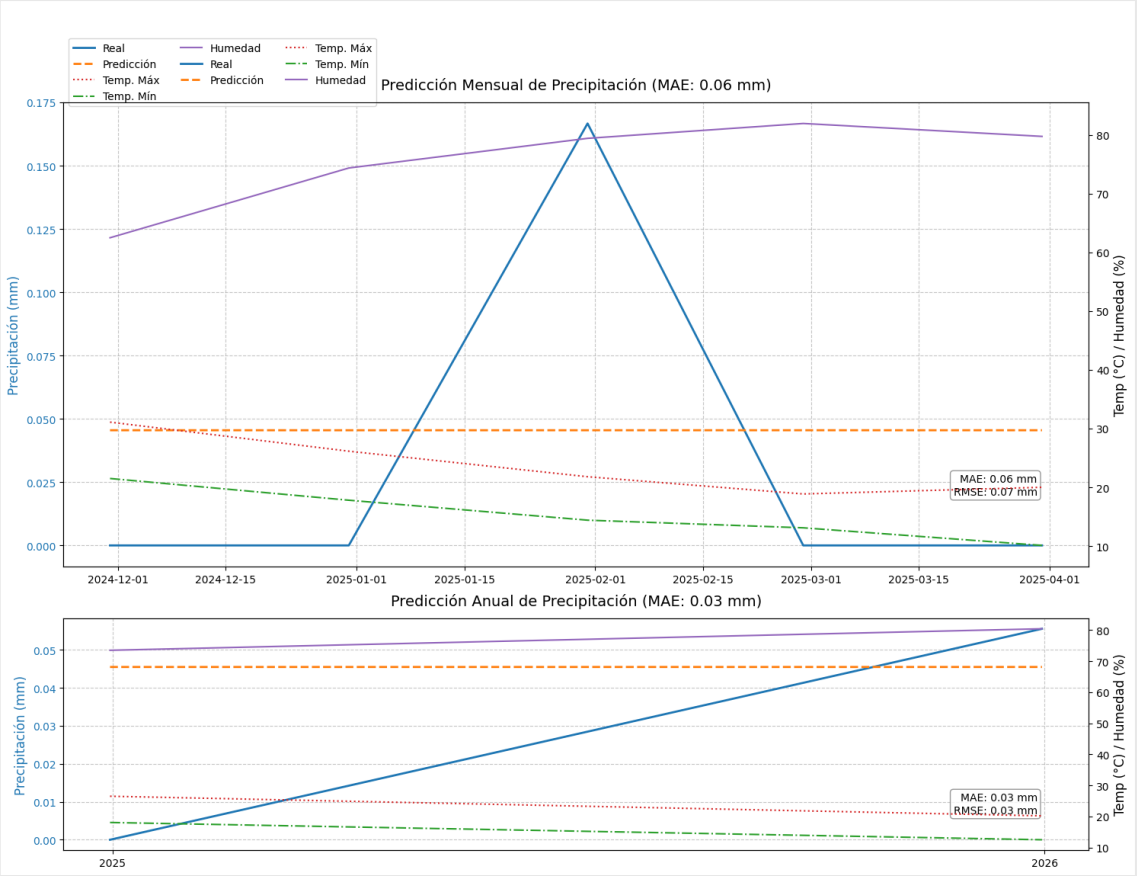
```
    'Humedad': data_test['humedad'].values[:n_samples]
}
data_monthly = pd.DataFrame(plot_data,
index=y_dates_test[:n_samples]).resample('ME').mean()
data_annual = pd.DataFrame(plot_data,
index=y_dates_test[:n_samples]).resample('YE').mean()

# Calcular métricas de exactitud (solo MAE y RMSE)
def calcular_metricas(reales, pred):
    return {
        'MAE': mean_absolute_error(reales, pred),
        'RMSE': math.sqrt(mean_squared_error(reales, pred))
    }
metrics_monthly = calcular_metricas(data_monthly['Real'],
data_monthly['Predicción'])
metrics_annual = calcular_metricas(data_annual['Real'],
data_annual['Predicción'])

# 2. Configuración de la figura
fig, (ax_monthly, ax_annual) = plt.subplots(
    2, 1,
    figsize=(15, 12),
    sharex=False,
    gridspec_kw={'height_ratios': [2, 1]}
)

# Estilo consistente para las gráficas
plot_config = {
    'Real': {'color': '#1f77b4', 'linewidth': 2, 'label': 'Real'},
    'Predicción': {'color': '#ff7f0e', 'linestyle': '--', 'linewidth': 2,
'label': 'Predicción'},
    'Temp Max': {'color': '#d62728', 'linestyle': ':', 'linewidth': 1.5,
'label': 'Temp. Máx'},
    'Temp Min': {'color': '#2ca02c', 'linestyle': '-.', 'linewidth': 1.5,
'label': 'Temp. Mín'},
    'Humedad': {'color': '#9467bd', 'linestyle': '-', 'linewidth': 1.5,
'label': 'Humedad'}
}

# 3. Gráfico Mensual con métricas
# Precipitación (eje principal)
ax_monthly.plot(data_monthly.index, data_monthly['Real'],
**plot_config['Real'])
ax_monthly.plot(data_monthly.index, data_monthly['Predicción'],
**plot_config['Predicción'])
ax_monthly.set_ylabel("Precipitación (mm)",
color=plot_config['Real']['color'], fontsize=12)
ax_monthly.tick_params(axis='y', labelcolor=plot_config['Real']['color'])

# Variables climáticas (eje secundario)
ax_monthly_clim = ax_monthly.twinx()
for var in ['Temp Max', 'Temp Min', 'Humedad']:
    ax_monthly_clim.plot(data_monthly.index, data_monthly[var],
**plot_config[var])
ax_monthly_clim.set_ylabel("Temp (°C) / Humedad (%)", fontsize=12)

# Añadir métricas como texto
metrics_text_monthly = (
    f"MAE: {metrics_monthly['MAE']:.2f} mm\n"
```

```
    f"RMSE: {metrics_monthly['RMSE']:.2f} mm"
)
ax_monthly.text(0.95, 0.15, metrics_text_monthly,
                transform=ax_monthly.transAxes,
                ha='right', va='bottom',
                bbox=dict(facecolor='white', alpha=0.8, edgecolor='gray',
boxstyle='round'),
                fontsize=10)
ax_monthly.grid(True, linestyle='--', alpha=0.7)
ax_monthly.set_title('Predicción Mensual de Precipitación (MAE: {:.2f}
mm)'.format(
    metrics_monthly['MAE']), fontsize=14, pad=12)

# 4. Gráfico Anual con métricas
# Precipitación (eje principal)
ax_annual.plot(data_annual.index, data_annual['Real'], **plot_config['Real'])
ax_annual.plot(data_annual.index, data_annual['Predicción'],
**plot_config['Predicción'])
ax_annual.set_ylabel("Precipitación (mm)", color=plot_config['Real']['color'],
fontsize=12)
ax_annual.tick_params(axis='y', labelcolor=plot_config['Real']['color'])

# Variables climáticas (eje secundario)
ax_annual_clim = ax_annual.twinx()
for var in ['Temp Max', 'Temp Min', 'Humedad']:
    ax_annual_clim.plot(data_annual.index, data_annual[var],
**plot_config[var])
ax_annual_clim.set_ylabel("Temp (°C) / Humedad (%)", fontsize=12)

# Añadir métricas como texto
metrics_text_annual = (
    f"MAE: {metrics_annual['MAE']:.2f} mm\n"
    f"RMSE: {metrics_annual['RMSE']:.2f} mm"
)
ax_annual.text(0.95, 0.15, metrics_text_annual,
               transform=ax_annual.transAxes,
               ha='right', va='bottom',
               bbox=dict(facecolor='white', alpha=0.8, edgecolor='gray',
boxstyle='round'),
               fontsize=10)

# Configuración de ejes temporales
ax_annual.xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
ax_annual.xaxis.set_major_locator(mdates.YearLocator())
ax_annual.grid(True, linestyle='--', alpha=0.7)
ax_annual.set_title('Predicción Anual de Precipitación (MAE: {:.2f}
mm)'.format(
    metrics_annual['MAE']), fontsize=14, pad=12)

# 5. Elementos comunes y formato final
# Leyenda unificada
lines = []
labels = []
for ax in [ax_monthly, ax_monthly_clim, ax_annual, ax_annual_clim]:
    l, lab = ax.get_legend_handles_labels()
    lines.extend(l)
    labels.extend(lab)
ax_monthly.legend(lines, labels,
                  loc='upper left',
```

```
                fontsize=10,
                frameon=True,
                bbox_to_anchor=(0, 1.15),
                ncol=3)

# Título general
fig.suptitle('Análisis Comparativo de Predicciones de Precipitación con
Métricas de Error',
             fontsize=16,
             fontweight='bold',
             y=1.05)

# Ajustes finales
plt.xticks(rotation=45, fontsize=10)
plt.xlabel('Fecha', fontsize=12)
plt.tight_layout()
plt.subplots_adjust(top=0.85)
plt.show()
```

Figura 41

Predicciones de precipitación con métricas de error del modelo GRU



Nota. El gráfico muestra un análisis comparativo de predicciones de precipitación con métricas de error del modelo GRU. Elaboración propia.

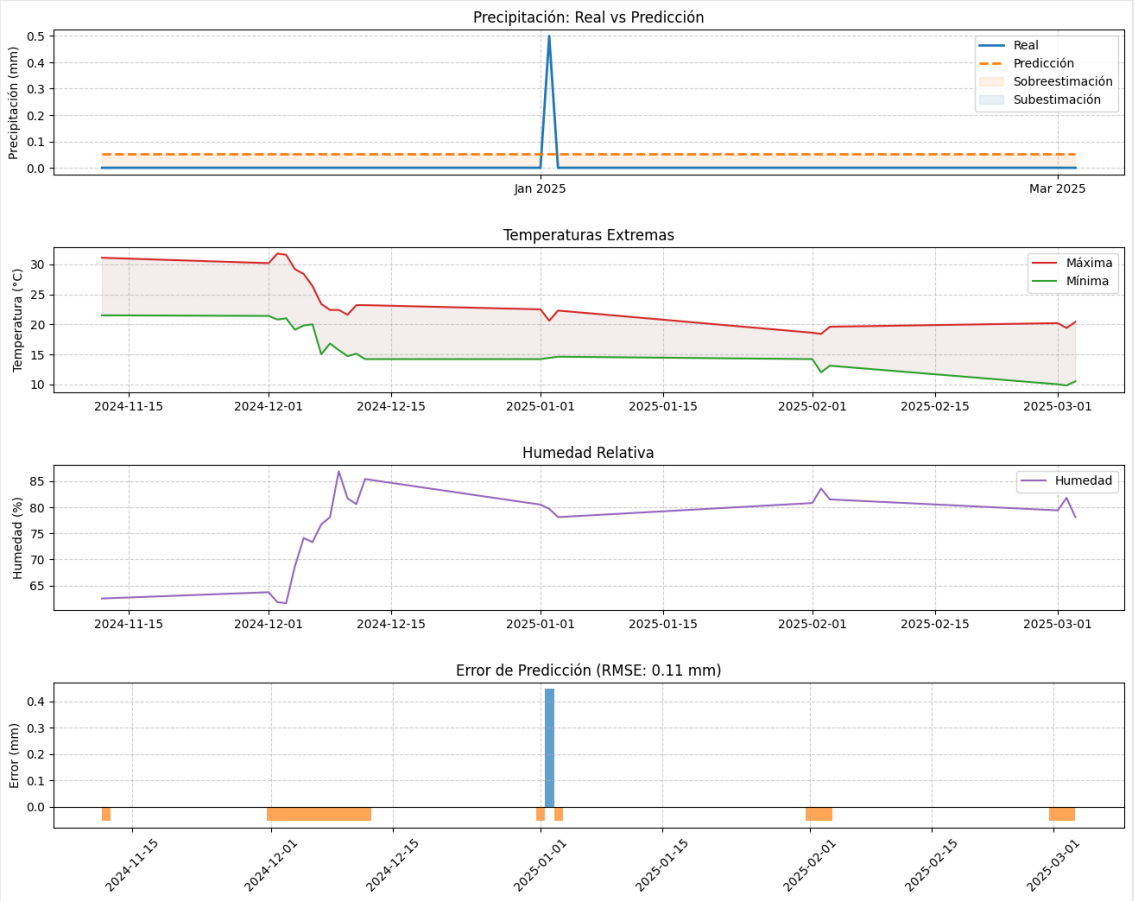
12. Visualización detallada de variables meteorológicas

```
# 📌 Paso 12: Visualización general de variables meteorológicas
fig = plt.figure(figsize=(16, 12))
gs = GridSpec(4, 1, figure=fig, hspace=0.5)
ax1 = fig.add_subplot(gs[0])
ax1.plot(y_dates_test[:n], y_test_rescaled[:n], label='Real', color='#1f77b4',
linewidth=2)
ax1.plot(y_dates_test[:n], y_pred[:n], label='Predicción', color='#ff7f0e',
linestyle='--', linewidth=2)
ax1.fill_between(y_dates_test[:n], y_test_rescaled[:n], y_pred[:n],
where=(y_pred[:n] > y_test_rescaled[:n]),
                 color='#ff7f0e', alpha=0.1, label='Sobreestimación')
ax1.fill_between(y_dates_test[:n], y_test_rescaled[:n], y_pred[:n],
where=(y_pred[:n] <= y_test_rescaled[:n]),
                 color='#1f77b4', alpha=0.1, label='Subestimación')
ax1.set_title('Precipitación: Real vs Predicción')
ax1.set_ylabel('Precipitación (mm)')
ax1.legend()
ax1.xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
ax1.xaxis.set_major_locator(mdates.MonthLocator(interval=2))
ax1.grid(True, linestyle='--', alpha=0.6)

ax2 = fig.add_subplot(gs[1])
ax2.plot(y_dates_test[:n], data_test['temp_max'].values[:n], label='Máxima',
color='#d62728')
```

```
ax2.plot(y_dates_test[:n], data_test['temp_min'].values[:n], label='Mínima',
color='#2ca02c')
ax2.fill_between(y_dates_test[:n], data_test['temp_min'].values[:n],
data_test['temp_max'].values[:n],
                 color='#8c564b', alpha=0.1)
ax2.set_title('Temperaturas Extremas')
ax2.set_ylabel('Temperatura (°C)')
ax2.legend()
ax2.grid(True, linestyle='--', alpha=0.6)

ax3 = fig.add_subplot(gs[2])
ax3.plot(y_dates_test[:n], data_test['humedad'].values[:n], label='Humedad',
color='#9467bd')
ax3.set_title('Humedad Relativa')
ax3.set_ylabel('Humedad (%)')
ax3.legend()
ax3.grid(True, linestyle='--', alpha=0.6)

ax4 = fig.add_subplot(gs[3])
error = y_test_rescaled[:n] - y_pred[:n]
ax4.bar(y_dates_test[:n], error, width=1, color=np.where(error > 0, '#1f77b4',
'#ff7f0e'), alpha=0.7)
ax4.axhline(0, color='black', linewidth=0.8)
ax4.set_title(f'Error de Predicción (RMSE: {np.sqrt(np.mean(error**2)):.2f}
mm)')
ax4.set_ylabel('Error (mm)')
ax4.grid(True, linestyle='--', alpha=0.6)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Figura 42

Visualización detallada de variables meteorológicas del modelo GRU



Nota. El gráfico muestra una visualización detallada de variables meteorológicas del modelo GRU. Elaboración propia.

Implementación del Modelo ARIMA

1. Instalar e importar Librerías

```
# 📌 Paso 1: Instalación limpia y controlada de librerías específicas
!pip install numpy==1.26.4 pandas==2.2.2 pmdarima --upgrade --force-reinstall

# 📌 1.2 Importar librerías
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error
from pmdarima.arima import auto_arima
from statsmodels.tsa.arima.model import ARIMA
from math import sqrt
import folium
from google.colab import drive
import matplotlib.dates as mdates
from matplotlib.gridspec import GridSpec
```

2. Montar Google Drive

```
# 📌 Paso 2: Montar Google Drive
drive.mount('/content/drive')
```

3. Cargar Datos y seleccionar variables

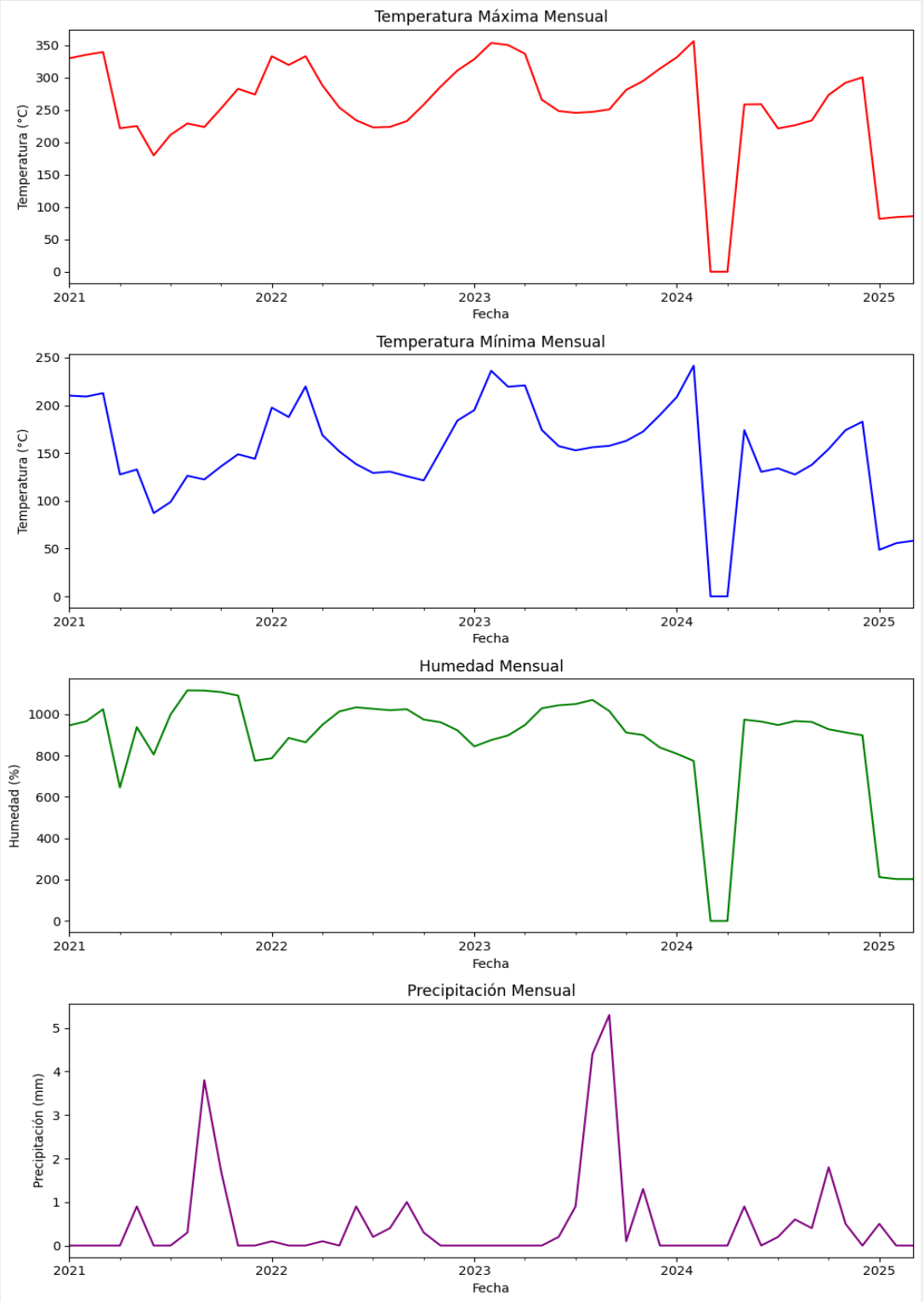
```
# Paso 3: Cargar el dataset
dataset_path = "/content/drive/MyDrive/TESIS MODELOS DE MACHINE LEARNING
V4/Data/datasetlimpia.csv"  # Ajusta la ruta si es necesario
data = pd.read_csv(dataset_path, parse_dates=['fecha'], dayfirst=True)
data = data.sort_values('fecha')
data = data.set_index('fecha') # Set the 'fecha' column as the index
print("Primeros 5 registros del dataset:")
print(data.head())
```

4. Visualización de Datos Reales

```
# Paso 4: Visualizar los datos
# Resampleamos a una frecuencia mensual para un análisis más claro
data_mensual = data.resample('ME').sum()

# Gráfico de las diferentes variables
fig, axes = plt.subplots(4, 1, figsize=(10, 15))

data_mensual['tem max'].plot(ax=axes[0], title='Temperatura Máxima Mensual',
xlabel='Fecha', ylabel='Temperatura (°C)', color='red')
data_mensual['tem min'].plot(ax=axes[1], title='Temperatura Mínima Mensual',
xlabel='Fecha', ylabel='Temperatura (°C)', color='blue')
data_mensual['humedad'].plot(ax=axes[2], title='Humedad Mensual',
xlabel='Fecha', ylabel='Humedad (%)', color='green')
data_mensual['precipitacion'].plot(ax=axes[3], title='Precipitación Mensual',
xlabel='Fecha', ylabel='Precipitación (mm)', color='purple')

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

Figura 43

Gráfico anual de variables meteorológicas para el modelo ARIMA



Nota. La imagen presenta cuatro gráficos que muestran la evolución anual de la Temperatura Máxima, la Temperatura Mínima, la Humedad y la Precipitación desde el año 2021 hasta el 2025. Elaboración Propia.

5. División de Datos por fecha en entrenamiento y prueba

```
# 📌 Paso 5: Dividir los datos en conjuntos de entrenamiento y prueba
train_size = int(len(data) * 0.8)  # 80% para entrenamiento
```

```
train, test = data[:train_size], data[train_size:]
```

6. Normalizar Datos

```
# 📌 Paso 6: Normalización de datos
scaler = MinMaxScaler()
train_scaled = scaler.fit_transform(train[['precipitacion']])  # Reemplaza
'value' con tu columna
test_scaled = scaler.transform(test[['precipitacion']])
train_scaled_series = train_scaled.flatten()
test_scaled_series = test_scaled.flatten()
```

7. Construcción del modelo ARIMA

```
# 📌 Paso 7: Construir modelo ARIMA
auto_model = auto_arima(
    train_scaled_series,
    seasonal=False,  # Cambia a True si hay estacionalidad
    stepwise=True,
    trace=False,
    max_p=5, max_d=2, max_q=5,
    suppress_warnings=True
)
order = auto_model.order
model = ARIMA(train_scaled_series, order=order)
model_fit = model.fit()
```

8. Entrenamiento y Prediccion

```
# Entrenar modelo ARIMA
model = ARIMA(train_scaled_series, order=order)
model_fit = model.fit()


# Predecir en el conjunto de prueba
forecast_scaled = model_fit.forecast(steps=len(test_scaled_series))
forecast_scaled_2d = forecast_scaled.reshape(-1, 1)
forecast = scaler.inverse_transform(forecast_scaled_2d).flatten()
```

9. Calcular métricas

```
# 📌 9 Calcular métricas
def calcular_metricas(y_real, y_pred, threshold=0.5):
    """Calcula e imprime métricas de evaluación para el modelo ARIMA.

    Parámetros:
    - y_real (array-like): Valores reales de salida.
    - y_pred (array-like): Valores predichos por el modelo.
    - threshold (float, optional): Umbral relativo para calcular exactitud
(default: 0.1, o 10%).

    Retorna:
    - metrics (dict): Diccionario con métricas de desempeño redondeadas.
    """
    y_real, y_pred = np.array(y_real).flatten(), np.array(y_pred).flatten()
    if len(y_real) != len(y_pred):
        raise ValueError("Longitudes de y_real y y_pred deben coincidir.")

    mae = mean_absolute_error(y_real, y_pred)
    mse = mean_squared_error(y_real, y_pred)
```

```
    rmse = np.sqrt(mse)
    non_zero_mask = y_real != 0
    mape = np.mean(np.abs((y_real[non_zero_mask] - y_pred[non_zero_mask]) /
y_real[non_zero_mask])) * 100 if np.any(non_zero_mask) else np.nan
    precision = 100 - mape if not np.isnan(mape) else np.nan

    # Calcular exactitud (porcentaje de predicciones dentro del umbral)
    max_value = np.max(np.abs(y_real)) if np.max(np.abs(y_real)) > 0 else
1.0  # Evitar división por cero
    accuracy = np.mean(np.abs(y_real - y_pred) <= threshold * max_value) * 100
if max_value > 0 else np.nan

    metrics = {
        'MAE': round(mae, 4), 'MSE': round(mse, 4), 'RMSE': round(rmse, 4),
        'MAPE': round(mape, 2) if not np.isnan(mape) else np.nan,
        'Precisión': round(precision, 2) if not np.isnan(precision) else
np.nan,
        'Exactitud': round(accuracy, 2) if not np.isnan(accuracy) else np.nan
    }

    print("\n📊 EVALUACIÓN DEL MODELO ARIMA")
    print("="*50)
    for metric, value in metrics.items():
        unit = 'mm' if 'MAE' in metric or 'RMSE' in metric else 'mm²' if 'MSE'
in metric else '%' if 'MAPE' in metric or 'Precisión' in metric or 'Exactitud'
in metric else ''
        print(f"{metric:15} {value:>10} {unit}")
    print("="*50)
    return metrics

# Calcular métricas
metricas = calcular_metricas(test['precipitacion'], forecast)

# 📌 9.1 Generar gráfico de métricas
metrics_names = list(metricas.keys())
metrics_values = list(metricas.values())

# Configurar el gráfico
plt.figure(figsize=(10, 6))
bars = plt.bar(metrics_names, metrics_values, color=['#1f77b4', '#ff7f0e',
'#2ca02c', '#d62728', '#9467bd', '#8c564b'])
plt.title('Evaluación de Métricas del Modelo ARIMA', fontsize=14, pad=15)
plt.xlabel('Métricas', fontsize=12)
plt.ylabel('Valor', fontsize=12)

# Añadir etiquetas de valor en las barras
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2., height,
             f'{height:.2f}' if not np.isnan(height) else 'N/A',
             ha='center', va='bottom' if height > 0 else 'top')

# Añadir unidades como anotaciones
units = ['mm', 'mm²', 'mm', '%', '%', '%']
for i, unit in enumerate(units):
    plt.text(i, -0.1 if i in [0, 2] else -0.05 if i == 1 else -5 if i in [3,
4, 5] else -0.2,
             unit, ha='center', va='top', rotation=0 if i in [0, 2] else 90)
```

```
# Ajustar límites del eje y
plt.ylim(min(-5, min([v for v in metrics_values if not np.isnan(v)] + [0]) *
1.2),
         max([v for v in metrics_values if not np.isnan(v)] + [0]) * 1.2)

# Añadir rejilla y estilo
plt.grid(True, linestyle='--', alpha=0.7)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()

# Mostrar gráfico
plt.show()

# 📌 9.2 Visualizar datos y predicciones
plt.figure(figsize=(12, 6))
plt.plot(train.index, train['precipitacion'], label='Entrenamiento
(original)')
plt.plot(test.index, test['precipitacion'], label='Prueba (original)',
color='blue')
plt.plot(test.index, forecast, label='Predicciones (desnormalizadas)',
color='red', linestyle='--')
plt.title(f'Predicciones del Modelo ARIMA (Orden {order})')
plt.xlabel('Fecha')
plt.ylabel('Valor')
plt.legend()
plt.grid(True)
plt.show()

# 📌 9.3 Imprimir resumen del modelo
print(model_fit.summary())
```

10. Visualización general de variables meteorológicas

```
# 📌 Paso 10: Visualización optimizada para modelo ARIMA
import matplotlib.dates as mdates
# Asegurar consistencia en la longitud de los arrays
n_samples = min(len(forecast), len(test['precipitacion']), len(test.index))

# Crear DataFrames con resampleo para diferentes escalas temporales
plot_data = {
    'Real': test['precipitacion'].values[:n_samples],
    'Predicción': forecast[:n_samples],
}

# Use the original test index for plotting
test_index_subset = test.index[:n_samples]

data_monthly = pd.DataFrame(plot_data,
index=test_index_subset).resample('ME').mean()
data_annual = pd.DataFrame(plot_data,
index=test_index_subset).resample('YE').mean()

# Handle NaNs in the resampled data
data_monthly = data_monthly.dropna()
data_annual = data_annual.dropna()

# Calculate confidence intervals from the forecast result
forecast_result = model_fit.get_forecast(steps=len(test))
```

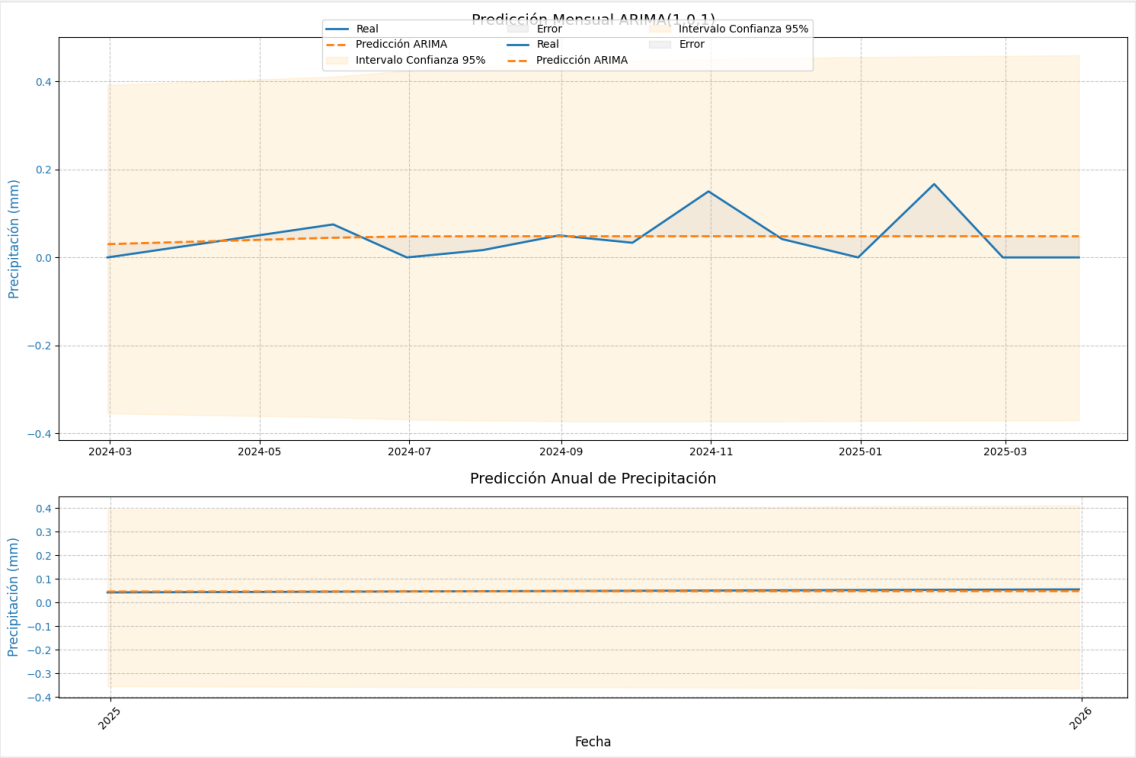
```
forecast_ci = scaler.inverse_transform(forecast_result.conf_int())
conf_int_upper = forecast_ci[:, 1]
conf_int_lower = forecast_ci[:, 0]

# Calcular métricas de exactitud
def calcular_metricas(reales, pred):
    return {
        'MAE': mean_absolute_error(reales, pred),
        'MSE': mean_squared_error(reales, pred),  # Nueva métrica agregada
        'RMSE': sqrt(mean_squared_error(reales, pred)),
        'MAPE': np.mean(np.abs((reales - pred) / reales)) * 100 if
np.all(reales != 0) else np.nan
    }

# Calculate metrics only if there is data after dropping NaNs
metrics_monthly = calcular_metricas(data_monthly['Real'],
data_monthly['Predicción']) if not data_monthly.empty else {'MAE': np.nan,
'MSE': np.nan, 'RMSE': np.nan, 'MAPE': np.nan}
metrics_annual = calcular_metricas(data_annual['Real'],
data_annual['Predicción']) if not data_annual.empty else {'MAE': np.nan,
'MSE': np.nan, 'RMSE': np.nan, 'MAPE': np.nan}

# Suposición: Parámetros ARIMA y AIC (deben provenir del modelo ajustado)
arima_params = model_fit.model.order
arima_aic = model_fit.aic

# Configuración de la figura
fig = plt.figure(figsize=(15, 10))
gs = fig.add_gridspec(2, 1, height_ratios=[2, 1])  # Adjusted to remove the
right column

ax_monthly = fig.add_subplot(gs[0, 0])
ax_annual = fig.add_subplot(gs[1, 0])

# Estilo consistente para las gráficas
plot_config = {
    'Real': {'color': '#1f77b4', 'linewidth': 2, 'label': 'Real'},
    'Predicción': {'color': '#ff7f0e', 'linestyle': '--', 'linewidth': 2,
'label': 'Predicción ARIMA'},
}

# Gráfico Mensual
if not data_monthly.empty:
    ax_monthly.plot(data_monthly.index, data_monthly['Real'],
**plot_config['Real'])
    ax_monthly.plot(data_monthly.index, data_monthly['Predicción'],
**plot_config['Predicción'])
    ax_monthly.fill_between(data_monthly.index,
                            data_monthly['Predicción'] -
(data_monthly['Predicción'] - conf_int_lower[:len(data_monthly)]),
                            data_monthly['Predicción'] +
(conf_int_upper[:len(data_monthly)] - data_monthly['Predicción']),
                            color='orange', alpha=0.1, label='Intervalo
Confianza 95%')
    ax_monthly.fill_between(data_monthly.index,
                            data_monthly['Real'],
                            data_monthly['Predicción'],
                            color='gray', alpha=0.1, label='Error')
```

```
    ax_monthly.set_ylabel("Precipitación (mm)",
color=plot_config['Real']['color'], fontsize=12)
    ax_monthly.tick_params(axis='y', labelcolor=plot_config['Real']['color'])
    ax_monthly.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
    ax_monthly.xaxis.set_major_locator(mdates.MonthLocator(interval=2))
    ax_monthly.grid(True, linestyle='--', alpha=0.7)
    ax_monthly.set_title(f'Predicción Mensual
ARIMA({arima_params[0]},{arima_params[1]},{arima_params[2]})', fontsize=14,
pad=12)
else:
    ax_monthly.set_title('No monthly data available after dropping NaNs')
    ax_monthly.axis('off')

# Gráfico Anual
if not data_annual.empty:
    ax_annual.plot(data_annual.index, data_annual['Real'],
**plot_config['Real'])
    ax_annual.plot(data_annual.index, data_annual['Predicción'],
**plot_config['Predicción'])
    ax_annual.fill_between(data_annual.index,
                           data_annual['Predicción'] -
(data_annual['Predicción'] - conf_int_lower[:len(data_annual)]),
                           data_annual['Predicción'] +
(conf_int_upper[:len(data_annual)] - data_annual['Predicción']),
                           color='orange', alpha=0.1, label='Intervalo
Confianza 95%')
    ax_annual.fill_between(data_annual.index,
                           data_annual['Real'],
                           data_annual['Predicción'],
                           color='gray', alpha=0.1, label='Error')
    ax_annual.set_ylabel("Precipitación (mm)",
color=plot_config['Real']['color'], fontsize=12)
    ax_annual.tick_params(axis='y', labelcolor=plot_config['Real']['color'])
    ax_annual.xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
    ax_annual.xaxis.set_major_locator(mdates.YearLocator())
    ax_annual.grid(True, linestyle='--', alpha=0.7)
    ax_annual.set_title('Predicción Anual de Precipitación', fontsize=14,
pad=12)
else:
    ax_annual.set_title('No annual data available after dropping NaNs')
    ax_annual.axis('off')

# Leyenda unificada
lines = []
labels = []
for ax in [ax_monthly, ax_annual]:
    if not ax.get_title().startswith('No'):
        l, lab = ax.get_legend_handles_labels()
        lines.extend(l)
        labels.extend(lab)

if lines:
    fig.legend(lines, labels,
               loc='upper center',
               fontsize=10,
               frameon=True,
               ncol=3,
               bbox_to_anchor=(0.5, 0.98))
```

```
# Título general
fig.suptitle('Análisis Comparativo de Predicciones ARIMA de Precipitación',
             fontsize=16,
             fontweight='bold',
             y=1.02)

# Ajustes finales
plt.xticks(rotation=45, fontsize=10)
plt.xlabel('Fecha', fontsize=12)
plt.tight_layout()
plt.subplots_adjust(top=0.95)
plt.show()
```

Figura 44

Predicciones de precipitación con métricas de error del modelo ARIMA



Nota. El gráfico muestra un análisis comparativo de predicciones de precipitación con métricas de error del modelo ARIMA. Elaboración propia.

11. Visualización detallada de variables meteorológicas

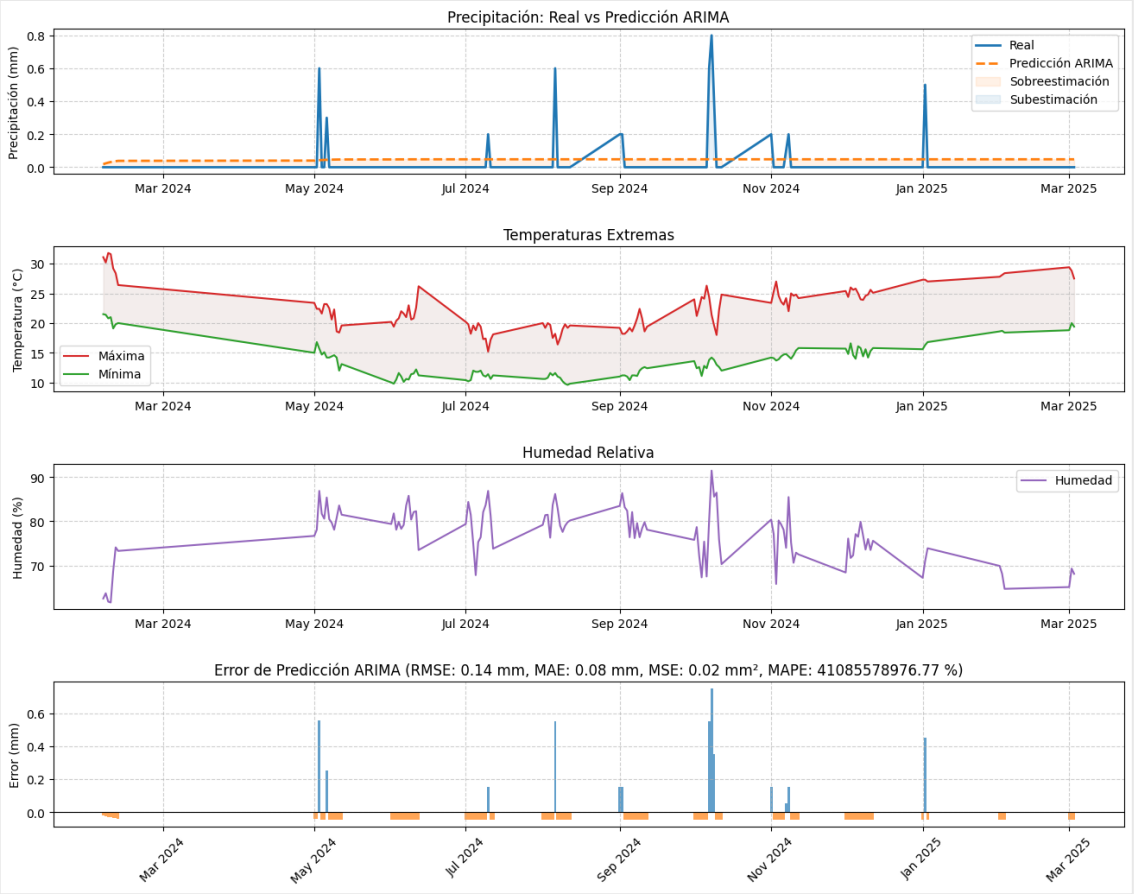
```
# Ensure forecast length matches test length for plotting
n = min(len(forecast), len(test))

# Create figure for visualization
fig = plt.figure(figsize=(16, 12))
gs = GridSpec(4, 1, figure=fig, hspace=0.5)

# Subplot 1: Precipitation - Real vs Prediction ARIMA
ax1 = fig.add_subplot(gs[0])
ax1.plot(test.index[:n], test['precipitacion'][:n], label='Real',
color='#1f77b4', linewidth=2)
```

```
ax1.plot(test.index[:n], forecast[:n], label='Predicción ARIMA',
color='#ff7f0e', linestyle='--', linewidth=2)
ax1.fill_between(test.index[:n], test['precipitacion'][:n], forecast[:n],
                 where=(forecast[:n] > test['precipitacion'][:n]),
color='#ff7f0e', alpha=0.1,
                 label='Sobreestimación')
ax1.fill_between(test.index[:n], test['precipitacion'][:n], forecast[:n],
                 where=(forecast[:n] <= test['precipitacion'][:n]),
color='#1f77b4', alpha=0.1,
                 label='Subestimación')
ax1.set_title('Precipitación: Real vs Predicción ARIMA')
ax1.set_ylabel('Precipitación (mm)')
ax1.legend()
ax1.xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
ax1.xaxis.set_major_locator(mdates.MonthLocator(interval=2))
ax1.grid(True, linestyle='--', alpha=0.6)

# Subplot 2: Extreme Temperatures
ax2 = fig.add_subplot(gs[1], sharex=ax1)
ax2.plot(test.index[:n], test['tem max'][:n], label='Máxima', color='#d62728')
ax2.plot(test.index[:n], test['tem min'][:n], label='Mínima', color='#2ca02c')
ax2.fill_between(test.index[:n], test['tem min'][:n], test['tem max'][:n],
                 color='#8c564b', alpha=0.1)
ax2.set_title('Temperaturas Extremas')
ax2.set_ylabel('Temperatura (°C)')
ax2.legend()
ax2.grid(True, linestyle='--', alpha=0.6)

# Subplot 3: Relative Humidity
ax3 = fig.add_subplot(gs[2], sharex=ax1)
ax3.plot(test.index[:n], test['humedad'][:n], label='Humedad',
color='#9467bd')
ax3.set_title('Humedad Relativa')
ax3.set_ylabel('Humedad (%)')
ax3.legend()
ax3.grid(True, linestyle='--', alpha=0.6)

# Subplot 4: Error de Predicción ARIMA con métricas adicionales
ax4 = fig.add_subplot(gs[3], sharex=ax1)
error = test['precipitacion'][:n] - forecast[:n]
# Calculate error metrics
rmse = np.sqrt(np.mean(error**2))  # Root Mean Squared Error
mae = np.mean(np.abs(error))  # Mean Absolute Error
mse = np.mean(error**2)  # Mean Squared Error
# MAPE: Avoid division by zero by replacing values close to zero in
test['precipitacion']
epsilon = 1e-10  # Small threshold to avoid division by zero
mape = np.mean(np.abs(error / np.where(np.abs(test['precipitacion'][:n]) <
epsilon, epsilon, test['precipitacion'][:n]))) * 100
# Plot error bars
ax4.bar(test.index[:n], error, width=1, color=np.where(error > 0, '#1f77b4',
'#ff7f0e'), alpha=0.7)
ax4.axhline(0, color='black', linewidth=0.8)
ax4.set_title(f'Error de Predicción ARIMA (RMSE: {rmse:.2f} mm, MAE: {mae:.2f}
mm, MSE: {mse:.2f} mm², MAPE: {mape:.2f} %)')
ax4.set_ylabel('Error (mm)')
ax4.grid(True, linestyle='--', alpha=0.6)

# Rotate x-axis labels and adjust layout
```

```
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Figura 45

Visualización detallada de variables meteorológicas del modelo ARIMA



Nota. El gráfico muestra una visualización detallada de variables meteorológicas del modelo ARIMA. Elaboración propia

Implementación del Modelo PROPHET

1. Instalar e importar Librerías

```
# Instalar Prophet
!pip install prophet
# Importar librerías
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from prophet import Prophet
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.preprocessing import MinMaxScaler
import math
```

```
import folium
```

2. Montar Google Drive

```
# 📌 Paso 2: Montar Google Drive
drive.mount('/content/drive')
```

3. Cargar Datos y seleccionar variables

```
# Cargar el archivo desde la ruta local
# 📌 Paso 3.1: Cargar datos
dataset_path = "/content/drive/MyDrive/TESIS MODELOS DE MACHINE LEARNING
V4/Data/datasetlimpia.csv"  # Ajusta la ruta si es necesario
data = pd.read_csv(dataset_path, parse_dates=['fecha'], dayfirst=True)
data = data.sort_values('fecha')
print("Primeros 5 registros del dataset:")
print(data.head())

# 📌 Paso 3.2: Seleccionar y preparar variables relevantes
data = data[['fecha', 'precipitacion', 'tem max', 'tem min',
'humedad']].dropna()
data.columns = ['fecha', 'precipitacion', 'temp_max', 'temp_min',
'humedad']  # Renombrar columnas
data.set_index('fecha', inplace=True)

# 📌 Paso 3.3: Validar rango de fechas hasta el 31 de Marzo de 2025
data = data[data.index <= '2025-03-31']
```

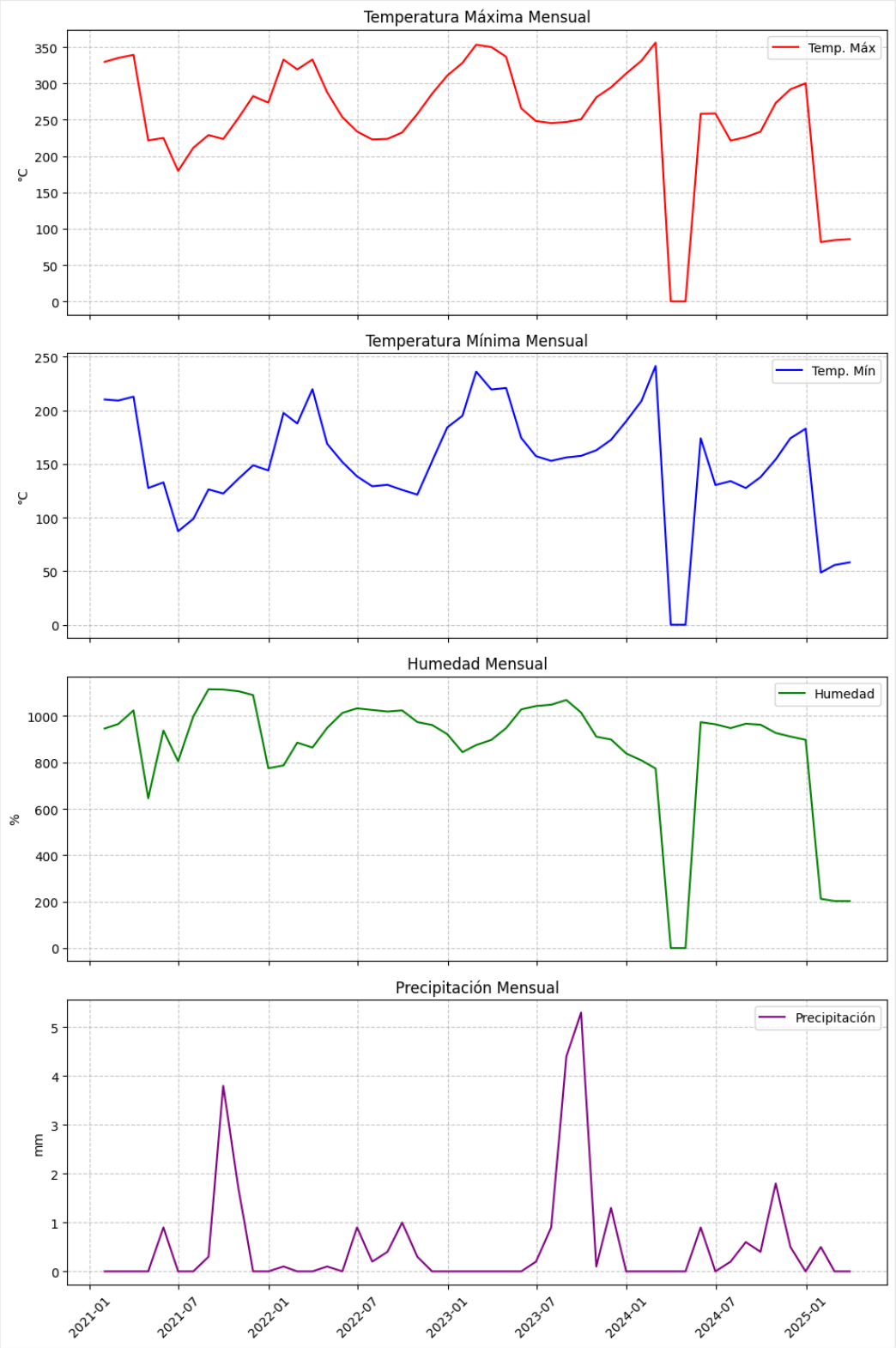
4. Visualización de Datos Reales

```
# 📌 Paso 4.1: Visualizar los datos por mes
data_mensual = data.resample('ME').sum()
fig, axes = plt.subplots(4, 1, figsize=(10, 15), sharex=True)
axes[0].plot(data_mensual.index, data_mensual['temp_max'], color='red',
label='Temp. Máx')
axes[0].set_title('Temperatura Máxima Mensual')
axes[0].set_ylabel('°C')
axes[1].plot(data_mensual.index, data_mensual['temp_min'], color='blue',
label='Temp. Mín')
axes[1].set_title('Temperatura Mínima Mensual')
axes[1].set_ylabel('°C')
axes[2].plot(data_mensual.index, data_mensual['humedad'], color='green',
label='Humedad')
axes[2].set_title('Humedad Mensual')
axes[2].set_ylabel('%')
axes[3].plot(data_mensual.index, data_mensual['precipitacion'],
color='purple', label='Precipitación')
axes[3].set_title('Precipitación Mensual')
axes[3].set_ylabel('mm')
for ax in axes:
    ax.legend()
    ax.grid(True, linestyle='--', alpha=0.7)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# 📌 Paso: Visualizar los datos Anual
data_anual = data.resample('YE').sum()
# Crear figura con subgráficos
fig, axes = plt.subplots(4, 1, figsize=(12, 18), sharex=True)
```

```
# Gráfico de Temperatura Máxima Anual
axes[0].plot(data_anual.index, data_anual['temp_max'], color='red',
label='Temp. Máx', linewidth=2)
axes[0].set_title('Temperatura Máxima Anual', fontsize=14, pad=10)
axes[0].set_ylabel('°C', fontsize=12)
axes[0].legend(loc='upper left', fontsize=10)
axes[0].grid(True, linestyle='--', alpha=0.7)

# Gráfico de Temperatura Mínima Anual
axes[1].plot(data_anual.index, data_anual['temp_min'], color='blue',
label='Temp. Mín', linewidth=2)
axes[1].set_title('Temperatura Mínima Anual', fontsize=14, pad=10)
axes[1].set_ylabel('°C', fontsize=12)
axes[1].legend(loc='upper left', fontsize=10)
axes[1].grid(True, linestyle='--', alpha=0.7)

# Gráfico de Humedad Anual
axes[2].plot(data_anual.index, data_anual['humedad'], color='green',
label='Humedad', linewidth=2)
axes[2].set_title('Humedad Anual', fontsize=14, pad=10)
axes[2].set_ylabel('%', fontsize=12)
axes[2].legend(loc='upper left', fontsize=10)
axes[2].grid(True, linestyle='--', alpha=0.7)

# Gráfico de Precipitación Anual
axes[3].plot(data_anual.index, data_anual['precipitacion'], color='purple',
label='Precipitación', linewidth=2)
axes[3].set_title('Precipitación Anual', fontsize=14, pad=10)
axes[3].set_ylabel('mm', fontsize=12)
axes[3].legend(loc='upper left', fontsize=10)
axes[3].grid(True, linestyle='--', alpha=0.7)

# Ajustar formato de fechas y etiquetas
plt.xlabel('Año', fontsize=12)
plt.xticks(rotation=45, fontsize=10)
axes[3].xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
axes[3].xaxis.set_major_locator(mdates.YearLocator())
# Ajustar diseño y mostrar
plt.tight_layout()
fig.suptitle('Visualización Anual de Variables Meteorológicas',
             fontsize=16, fontweight='bold', y=1.02)
plt.show()
```

Figura 46

Gráfico anual de variables meteorológicas para el modelo PROPHET



Nota. La imagen presenta cuatro gráficos que muestran la evolución anual de la Temperatura Máxima, la Temperatura Mínima, la Humedad y la Precipitación desde el año 2021 hasta el 2025. Elaboración Propia.

5. División de Datos por fecha en entrenamiento y prueba

```
# 📌 Paso 5: División en conjunto de entrenamiento y prueba
fecha_inicio = '2021-01-01'
```

```
fecha_fin = '2025-03-31'
data.index = pd.to_datetime(data.index)
data_filtered = data.loc[fecha_inicio:fecha_fin].copy()

# División 80% entrenamiento, 20% prueba
split_index = int(len(data_filtered) * 0.8)
data_train = data_filtered.iloc[:split_index].copy()
data_test = data_filtered.iloc[split_index:].copy()

features = ['precipitacion', 'temp_max', 'temp_min', 'humedad']
```

6. Normalizar Datos

```
# 📌 Paso 6: Normalización de datos
scaler = MinMaxScaler()
scaler.fit(data_train[features])
train_scaled = scaler.transform(data_train[features])
test_scaled = scaler.transform(data_test[features])
```

7. Crear secuencias y optimizar ventana de tiempo

```
# 📌 Paso 7.1: Crear secuencias
def create_sequences(data, dates, n_steps=30, target_col=0):
    """Crea secuencias para modelos recurrentes."""
    if len(data) <= n_steps:
        raise ValueError("El tamaño de los datos es insuficiente para la
ventana temporal.")
    X, y, y_dates = [], [], []
    for i in range(n_steps, len(data)):
        X.append(data[i - n_steps:i, :])
        y.append(data[i, target_col])
        y_dates.append(dates[i])
    return np.array(X), np.array(y), np.array(y_dates)

# 📌 Paso 7.2. Optimizar ventana de tiempo
n_steps_options = [7, 15, 30, 60, 90]
best_n_steps = 30
best_mae = float('inf')

for n_steps in n_steps_options:
    X_train, y_train, y_dates_train = create_sequences(train_scaled,
data_train.index, n_steps)
    X_test, y_test, y_dates_test = create_sequences(test_scaled,
data_test.index, n_steps)
    X_train = X_train.reshape((X_train.shape[0], X_train.shape[1],
X_train.shape[2]))
    X_test = X_test.reshape((X_test.shape[0], X_test.shape[1],
X_test.shape[2]))
    mae = np.mean(np.abs(y_test - np.mean(y_test)))  # Placeholder
    if mae < best_mae:
        best_mae = mae
        best_n_steps = n_steps
    print(f"n_steps={n_steps}, X_train.shape={X_train.shape}, MAE
(simulado)={mae:.4f}")

n_steps = best_n_steps
X_train, y_train, y_dates_train = create_sequences(train_scaled,
data_train.index, n_steps)
X_test, y_test, y_dates_test = create_sequences(test_scaled, data_test.index,
n_steps)
```

```
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1],
X_train.shape[2]))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], X_test.shape[2]))
print(f"Mejor ventana de tiempo: {n_steps}")
```

8. Preparar datos para el modelo PROPHET

```
# 📌 Paso 1: Preparar datos para Prophet
# Suponiendo que X_train es un arreglo 3D (muestras, pasos de tiempo,
características) y y_train es el objetivo
# Convertir al formato requerido por Prophet: DataFrame con 'ds' (fechas) y
'y' (valores)
def prepare_prophet_data(X_train, y_train, start_date='2021-01-21',
end_date='2025-03-31'):
    # Generar un rango de fechas desde el 21/01/2021 hasta el 31/03/2025 con
frecuencia diaria
    date_range = pd.date_range(start=start_date, end=end_date, freq='D')

    # Aplanar o seleccionar la variable objetivo (suponiendo que y_train es 1D
o puede ser remodelado)
    y_flat = y_train.flatten() if len(y_train.shape) > 1 else y_train

    # Asegurarse de que el rango de fechas coincida con la longitud de y_flat
    if len(date_range) < len(y_flat):
        raise ValueError(f"El rango de fechas ({len(date_range)} días) es
menor que la longitud de y_train ({len(y_flat)})")
    dates = date_range[:len(y_flat)]  # Truncar para que coincida con la
longitud de y_train

    # Crear DataFrame para Prophet
    df = pd.DataFrame({
        'ds': dates,
        'y': y_flat
    })

    # Si hay variables exógenas (características en X_train), agregarlas como
regresores
    # Suponiendo que X_train tiene forma (muestras, pasos de tiempo,
características)
    if X_train.shape[2] > 1:  # Verificar si hay características adicionales
        for i in range(1, X_train.shape[2]):  # Empezar desde 1 para omitir la
serie principal
            df[f'regressor_{i}'] = X_train[:, :, i].flatten()[:len(y_flat)]

    return df

# Preparar los datos de entrenamiento
df_train = prepare_prophet_data(X_train, y_train, start_date='2021-01-21',
end_date='2025-03-31')
```

9. Construcción del modelo PROPHET

```
# 📌 Paso 9: Construir el modelo Prophet
def build_prophet_model():
    model = Prophet(
        yearly_seasonality=True,      # Incluir estacionalidad anual
        weekly_seasonality=True,      # Incluir estacionalidad semanal
        daily_seasonality=True,       # Incluir estacionalidad diaria
        seasonality_mode='additive', # Puede ser 'additive' (aditiva) o
'multiplicative' (multiplicativa)
```

```
        changepoint_prior_scale=0.05,    # Controla la flexibilidad de la
tendencia
        seasonality_prior_scale=10.0     # Controla la flexibilidad de la
estacionalidad
    )
    # Agregar regresores exógenos si están presentes
    if 'regressor_1' in df_train.columns:
        for col in df_train.columns:
            if col.startswith('regressor_'):
                model.add_regressor(col)
    return model

# Inicializar el modelo
prophet_model = build_prophet_model()

# 📌 Paso 9.1: Ajustar (entrenar) el modelo Prophet
# Prophet no usa callbacks tradicionales, pero se puede limitar el número de
iteraciones o monitorear el rendimiento
prophet_model.fit(df_train)

# 📌 Paso 9.2: Preparar el DataFrame futuro para la predicción
# Crear fechas futuras para pronóstico (extiende más allá de la fecha final
para predecir)
future = prophet_model.make_future_dataframe(periods=30, freq='D')

# Si se usan regresores exógenos, extenderlos para las predicciones futuras
# Nota: Se necesitan valores futuros de los regresores; aquí se asumen ceros o
el último valor conocido
if 'regressor_1' in df_train.columns:
    for col in df_train.columns:
        if col.startswith('regressor_'):
            future[col] = np.zeros(len(future))  # Marcador de posición;
reemplazar con valores reales
# 📌 Paso 9.3: Realizar predicciones
forecast = prophet_model.predict(future)

# 📌 Paso 9.4: Evaluar el modelo (similar a las métricas del GRU: MSE, MAE)
# Suponiendo que existen y_test y X_test para evaluación
df_test = prepare_prophet_data(X_test, y_test, start_date='2021-01-01',
end_date='2025-03-31')
y_pred = prophet_model.predict(df_test)[['yhat']].values.flatten()
y_true = df_test['y'].values

mse = mean_squared_error(y_true, y_pred)
mae = mean_absolute_error(y_true, y_pred)
print(f"Modelo Prophet - MSE: {mse:.4f}, MAE: {mae:.4f}")

# 📌 Paso 9.5: Visualizar resultados (Opcional)
# Prophet proporciona funciones de visualización integradas
prophet_model.plot(forecast)
prophet_model.plot_components(forecast)
```

10. Calcular métricas

```
# 📌 10 Calcular métricas
def calcular_metricas(y_real, y_pred, threshold=0.5):
    """Calcula e imprime métricas de evaluación para el modelo Prophet.

    Parámetros:
```

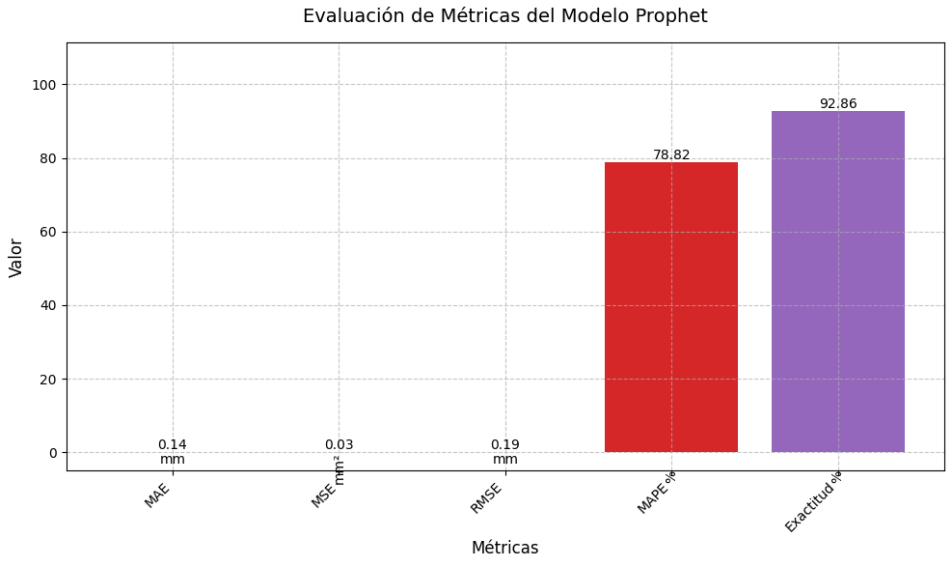
```
    - y_real (array-like): Valores reales de salida.
    - y_pred (array-like): Valores predichos por el modelo.
    - threshold (float, optional): Umbral relativo para calcular exactitud
(default: 0.5, o 50%).

    Retorna:
    - metrics (dict): Diccionario con métricas de desempeño redondeadas.
    """
    y_real, y_pred = np.array(y_real).flatten(), np.array(y_pred).flatten()
    if len(y_real) != len(y_pred):
        raise ValueError("Longitudes de y_real y y_pred deben coincidir.")

    mae = mean_absolute_error(y_real, y_pred)
    mse = mean_squared_error(y_real, y_pred)
    rmse = np.sqrt(mse)

    # Calculate SMAPE (Symmetric Mean Absolute Percentage Error)
    # This is a more robust metric when actual values can be zero
    mask = y_real != 0
    mape = np.mean(np.abs((y_real[mask] - y_pred[mask]) / y_real[mask])) * 100


    # Calcular exactitud (porcentaje de predicciones dentro del umbral)
    max_value = np.max(np.abs(y_real)) if np.max(np.abs(y_real)) > 0 else
1.0  # Evitar división por cero
    accuracy = np.mean(np.abs(y_real - y_pred) <= threshold * max_value) * 100
if max_value > 0 else np.nan

    metrics = {
        'MAE': round(mae, 4), 'MSE': round(mse, 4), 'RMSE': round(rmse, 4),
        'MAPE': round(mape, 2),
        'Exactitud': round(accuracy, 2) if not np.isnan(accuracy) else np.nan
    }

    print("\n📊 EVALUACIÓN DEL MODELO PROPHET")
    print("="*50)
    for metric, value in metrics.items():
        unit = 'mm' if 'MAE' in metric or 'RMSE' in metric else 'mm²' if 'MSE'
in metric else '%' if 'SMAPE' in metric or 'Exactitud' in metric else ''
        print(f"{metric:15} {value:>10} {unit}")
    print("="*50)
    return metrics

# Suponiendo que 'data_train' y 'data_test' son DataFrames con columnas
'fecha' y 'precipitacion'
# Preparar datos para Prophet
train_prophet = data_train.reset_index().rename(columns={'fecha': 'ds',
'precipitacion': 'y'})
test_prophet = data_test.reset_index().rename(columns={'fecha': 'ds',
'precipitacion': 'y'})

# 📌 10.1 Entrenar el modelo Prophet
model = Prophet(yearly_seasonality=True, weekly_seasonality=True,
daily_seasonality=True)
model.fit(train_prophet)

# 📌 10.2 Generar predicciones
# Create future dataframe directly from the test set dates
```

```
future = pd.DataFrame({'ds': test_prophet['ds']})

forecast = model.predict(future)

# Filter the forecast to match the test set dates
forecast_test = forecast['yhat'].values


# 📌 10.3 Calcular métricas
metricas = calcular_metricas(test_prophet['y'], forecast_test)

# 📌 10.4 Generar gráfico de métricas
metrics_names = list(metricas.keys())
metrics_values = list(metricas.values())

plt.figure(figsize=(10, 6))
# Update bar plot colors to match the new number of metrics
colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd']
bars = plt.bar(metrics_names, metrics_values,
color=colors[:len(metrics_names)])
plt.title('Evaluación de Métricas del Modelo Prophet', fontsize=14, pad=15)
plt.xlabel('Métricas', fontsize=12)
plt.ylabel('Valor', fontsize=12)

# Añadir etiquetas de valor en las barras
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2., height,
             f'{height:.2f}' if not np.isnan(height) else 'N/A',
             ha='center', va='bottom' if height > 0 else 'top')

# Añadir unidades como anotaciones
# Update units list to match the new metrics
units = ['mm', 'mm²', 'mm', '%', '%']
for i, unit in enumerate(units):
     plt.text(i, -0.1 if i in [0, 2] else -0.05 if i == 1 else -5 if i in [3,
4] else -0.2,
             unit, ha='center', va='top', rotation=0 if i in [0, 2] else 90)


# Ajustar límites del eje y
plt.ylim(min(-5, min([v for v in metrics_values if not np.isnan(v)] + [0]) *
1.2),
         max([v for v in metrics_values if not np.isnan(v)] + [0]) * 1.2)

# Añadir rejilla y estilo
plt.grid(True, linestyle='--', alpha=0.7)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()

# Mostrar gráfico
plt.show()

# 📌 10.5 Visualizar datos y predicciones
plt.figure(figsize=(12, 6))
plt.plot(train_prophet['ds'], train_prophet['y'], label='Entrenamiento
(original)')
```

```
plt.plot(test_prophet['ds'], test_prophet['y'], label='Prueba (original)',
color='blue')
plt.plot(test_prophet['ds'], forecast_test, label='Predicciones (Prophet)',
color='red', linestyle='--')
plt.title('Predicciones del Modelo Prophet')
plt.xlabel('Fecha')
plt.ylabel('Precipitación (mm)')
plt.legend()
plt.grid(True)
plt.show()

# 📌 10.6 Imprimir componentes del modelo Prophet
fig = model.plot_components(forecast)
plt.show()
```

Figura 47

Evaluación de métricas del modelo PROPHET



Nota. El grafico de Evaluación de las métricas del Modelo PROPHET. Elaboración propia.

11. Visualización general de variables meteorológicas

```
# 📌 Paso 11: Visualización general de variables meteorológicas
# Ensure test_prophet has a DatetimeIndex
if not isinstance(test_prophet.index, pd.DatetimeIndex):
    if 'ds' in test_prophet.columns:
        test_prophet.set_index('ds', inplace=True)
    else:
        print("Error: 'ds' column not found in test_prophet. Cannot set
DatetimeIndex.")

# Ensure consistency in the length of the arrays
n_samples = min(len(forecast_test), len(test_prophet['y']),
len(test_prophet.index), len(data_test))

# Create DataFrames with resampleo for different temporal scales
# Resample the original dataframes first and then select the columns
data_monthly_real = test_prophet.resample('ME').mean()['y']
```

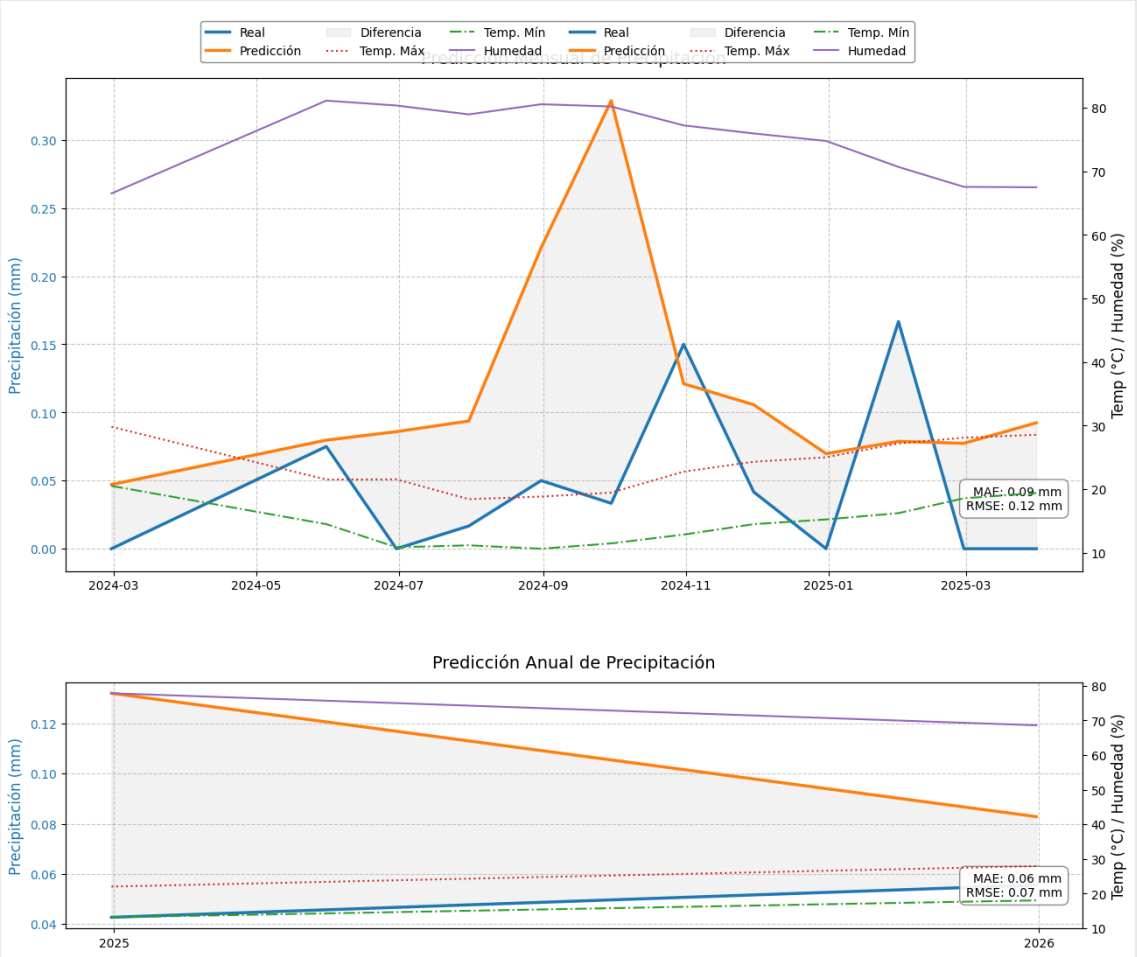
```
data_monthly_pred = pd.Series(forecast_test[:n_samples],
index=test_prophet.index[:n_samples]).resample('ME').mean()
data_monthly_temp_max = data_test.resample('ME').mean()['temp_max']
data_monthly_temp_min = data_test.resample('ME').mean()['temp_min']
data_monthly_humedad = data_test.resample('ME').mean()['humedad']

data_monthly = pd.DataFrame({
    'Real': data_monthly_real,
    'Predicción': data_monthly_pred,
    'Temp Max': data_monthly_temp_max,
    'Temp Min': data_monthly_temp_min,
    'Humedad': data_monthly_humedad
}).dropna() # Remove rows with NaN values

data_annual_real = test_prophet.resample('YE').mean()['y']
data_annual_pred = pd.Series(forecast_test[:n_samples],
index=test_prophet.index[:n_samples]).resample('YE').mean()
data_annual_temp_max = data_test.resample('YE').mean()['temp_max']
data_annual_temp_min = data_test.resample('YE').mean()['temp_min']
data_annual_humedad = data_test.resample('YE').mean()['humedad']

data_annual = pd.DataFrame({
    'Real': data_annual_real,
    'Predicción': data_annual_pred,
    'Temp Max': data_annual_temp_max,
    'Temp Min': data_annual_temp_min,
    'Humedad': data_annual_humedad
}).dropna() # Remove rows with NaN values


# Calculate metrics of exactitud (only MAE and RMSE)
def calcular_metricas(reales, pred):
    return {
        'MAE': mean_absolute_error(reales, pred),
        'RMSE': math.sqrt(mean_squared_error(reales, pred))
    }

metrics_monthly = calcular_metricas(data_monthly['Real'],
data_monthly['Predicción'])
metrics_annual = calcular_metricas(data_annual['Real'],
data_annual['Predicción'])

# 2. Configuración de la figura
fig, (ax_monthly, ax_annual) = plt.subplots(
    2, 1,
    figsize=(15, 12),
    sharex=False,
    gridspec_kw={'height_ratios': [2, 1], 'hspace': 0.3}
)

# Estilo consistente para las gráficas
plot_config = {
    'Real': {'color': '#1f77b4', 'linewidth': 2.5, 'label': 'Real'},
    'Predicción': {'color': '#ff7f0e', 'linewidth': 2.5, 'label':
'Predicción'},
    'Temp Max': {'color': '#d62728', 'linestyle': ':', 'linewidth': 1.5,
'label': 'Temp. Máx'},
    'Temp Min': {'color': '#2ca02c', 'linestyle': '-.', 'linewidth': 1.5,
'label': 'Temp. Mín'},
```

```
    'Humedad': {'color': '#9467bd', 'linestyle': '-', 'linewidth': 1.5,
'label': 'Humedad'}
}

# 3. Gráfico Mensual con métricas
# Precipitación (eje principal)
ax_monthly.plot(data_monthly.index, data_monthly['Real'],
**plot_config['Real'])
ax_monthly.plot(data_monthly.index, data_monthly['Predicción'],
**plot_config['Predicción'])
# Añadir área sombreada entre real y predicción
ax_monthly.fill_between(data_monthly.index,
                        data_monthly['Real'],
                        data_monthly['Predicción'],
                        color='gray', alpha=0.1, label='Diferencia')
ax_monthly.set_ylabel("Precipitación (mm)",
color=plot_config['Real']['color'], fontsize=12)
ax_monthly.tick_params(axis='y', labelcolor=plot_config['Real']['color'])
ax_monthly.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
ax_monthly.xaxis.set_major_locator(mdates.MonthLocator(interval=2))

# Variables climáticas (eje secundario)
ax_monthly_clim = ax_monthly.twinx()
for var in ['Temp Max', 'Temp Min', 'Humedad']:
    ax_monthly_clim.plot(data_monthly.index, data_monthly[var],
**plot_config[var])
ax_monthly_clim.set_ylabel("Temp (°C) / Humedad (%)", fontsize=12)

# Añadir métricas como texto
metrics_text_monthly = (
    f"MAE: {metrics_monthly['MAE']:.2f} mm\n"
    f"RMSE: {metrics_monthly['RMSE']:.2f} mm"
)
ax_monthly.text(0.98, 0.12, metrics_text_monthly,
                transform=ax_monthly.transAxes,
                ha='right', va='bottom',
                bbox=dict(facecolor='white', alpha=0.9, edgecolor='gray',
boxstyle='round,pad=0.5'),
                fontsize=10)

ax_monthly.grid(True, linestyle='--', alpha=0.7)
ax_monthly.set_title('Predicción Mensual de Precipitación', fontsize=14,
pad=12)

# 4. Gráfico Anual con métricas
# Precipitación (eje principal)
ax_annual.plot(data_annual.index, data_annual['Real'], **plot_config['Real'])
ax_annual.plot(data_annual.index, data_annual['Predicción'],
**plot_config['Predicción'])
# Añadir área sombreada entre real y predicción
ax_annual.fill_between(data_annual.index,
                       data_annual['Real'],
                       data_annual['Predicción'],
                       color='gray', alpha=0.1, label='Diferencia')
ax_annual.set_ylabel("Precipitación (mm)", color=plot_config['Real']['color'],
fontsize=12)
ax_annual.tick_params(axis='y', labelcolor=plot_config['Real']['color'])

# Variables climáticas (eje secundario)
```

```
ax_annual_clim = ax_annual.twinx()
for var in ['Temp Max', 'Temp Min', 'Humedad']:
    ax_annual_clim.plot(data_annual.index, data_annual[var],
**plot_config[var])
ax_annual_clim.set_ylabel("Temp (°C) / Humedad (%)", fontsize=12)

# Añadir métricas como texto
metrics_text_annual = (
    f"MAE: {metrics_annual['MAE']:.2f} mm\n"
    f"RMSE: {metrics_annual['RMSE']:.2f} mm"
)
ax_annual.text(0.98, 0.12, metrics_text_annual,
               transform=ax_annual.transAxes,
               ha='right', va='bottom',
               bbox=dict(facecolor='white', alpha=0.9, edgecolor='gray',
boxstyle='round,pad=0.5'),
               fontsize=10)

# Configuración de ejes temporales
ax_annual.xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
ax_annual.xaxis.set_major_locator(mdates.YearLocator())
ax_annual.grid(True, linestyle='--', alpha=0.7)
ax_annual.set_title('Predicción Anual de Precipitación', fontsize=14, pad=12)

# 5. Elementos comunes y formato final
# Leyenda unificada
lines = []
labels = []
for ax in [ax_monthly, ax_monthly_clim, ax_annual, ax_annual_clim]:
    l, lab = ax.get_legend_handles_labels()
    lines.extend(l)
    labels.extend(lab)

fig.legend(lines, labels,
           loc='upper center',
           fontsize=10,
           frameon=True,
           bbox_to_anchor=(0.5, 0.98),
           ncol=6,
           edgecolor='gray',
           facecolor='white',
           framealpha=0.9)

# Título general
fig.suptitle('Análisis Comparativo de Predicciones de Precipitación',
             fontsize=16,
             fontweight='bold',
             y=1.02)

# Ajustes finales
plt.xlabel('Fecha', fontsize=12)
plt.tight_layout()
plt.subplots_adjust(top=0.92)
plt.show()
```

Figura 48

Predicciones de precipitación con métricas de error del modelo PROPHET



Nota. El gráfico muestra un análisis comparativo de predicciones de precipitación con métricas de error del modelo PROPHET. Elaboración propia.

12. Visualización detallada de variables meteorológicas

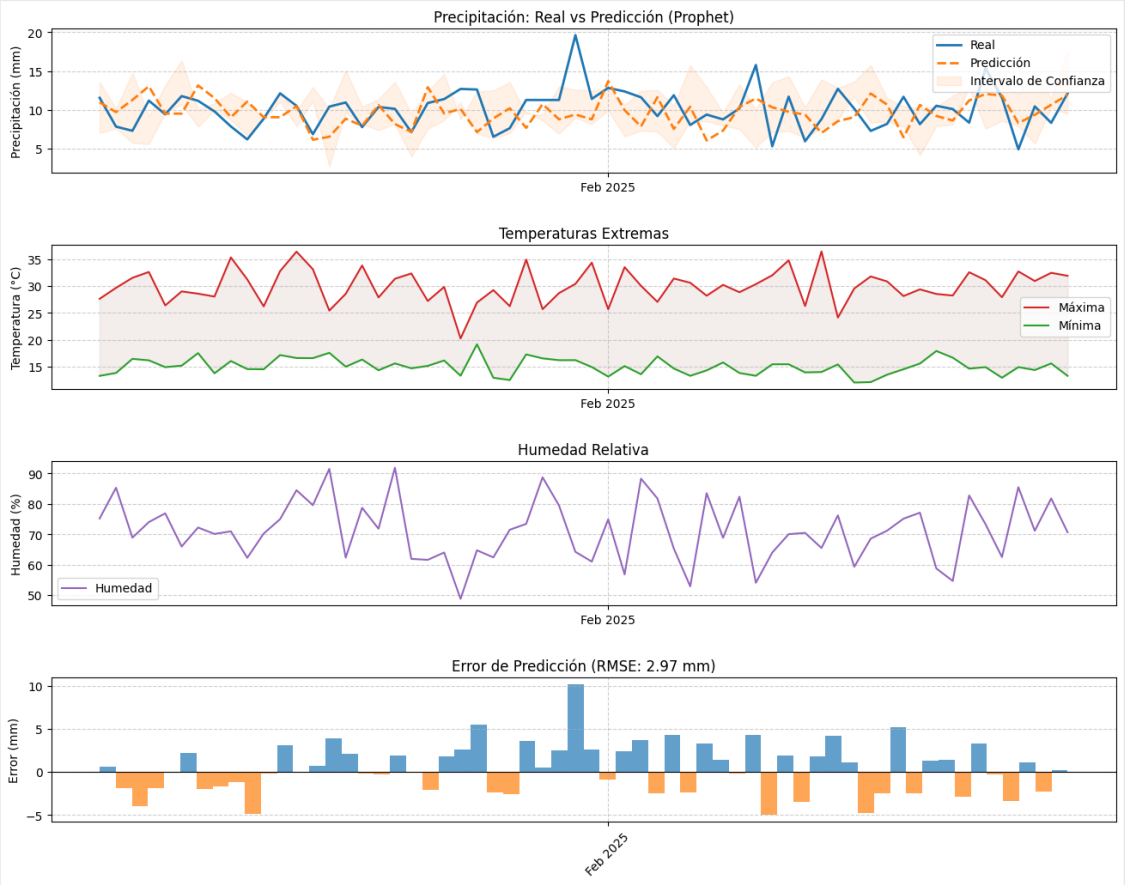
```
# Paso 12 Visualización general de variables meteorológicas

def plot_prophet_weather_variables(prophet_df, y_test, data_test, n):
    """
    Visualiza predicciones de Prophet junto con variables meteorológicas.

    Args:
        prophet_df: DataFrame con columnas 'ds', 'yhat', 'yhat_lower',
'yhat_upper' de Prophet.
        y_test: Valores reales de precipitación (Serie o array).
        data_test: DataFrame con columnas 'temp_max', 'temp_min', 'humedad'.
        n: Número de puntos a graficar.
    """
    # Asegurar que las fechas estén en formato datetime
    prophet_df['ds'] = pd.to_datetime(prophet_df['ds'])

    # Configuración de la figura
    fig = plt.figure(figsize=(16, 12))
    gs = GridSpec(4, 1, figure=fig, hspace=0.5)

    # Paleta de colores consistente
    colors = {
        'real': '#1f77b4',
```

```
        'pred': '#ff7f0e',
        'temp_max': '#d62728',
        'temp_min': '#2ca02c',
        'humedad': '#9467bd',
        'error_pos': '#1f77b4',
        'error_neg': '#ff7f0e',
        'confidence': '#ff7f0e'
    }

    # Gráfico 1: Precipitación Real vs Predicción con intervalos de confianza
    ax1 = fig.add_subplot(gs[0])
    ax1.plot(prophet_df['ds'][:n], y_test[:n], label='Real',
color=colors['real'], linewidth=2)
    ax1.plot(prophet_df['ds'][:n], prophet_df['yhat'][:n], label='Predicción',
color=colors['pred'], linestyle='--', linewidth=2)
    ax1.fill_between(prophet_df['ds'][:n], prophet_df['yhat_lower'][:n],
prophet_df['yhat_upper'][:n],
                     color=colors['confidence'], alpha=0.1, label='Intervalo
de Confianza')
    ax1.set_title('Precipitación: Real vs Predicción (Prophet)')
    ax1.set_ylabel('Precipitación (mm)')
    ax1.legend()
    ax1.xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
    ax1.xaxis.set_major_locator(mdates.MonthLocator(interval=2))
    ax1.grid(True, linestyle='--', alpha=0.6)

    # Gráfico 2: Temperaturas Extremas
    ax2 = fig.add_subplot(gs[1])
    ax2.plot(prophet_df['ds'][:n], data_test['temp_max'].values[:n],
label='Máxima', color=colors['temp_max'])
    ax2.plot(prophet_df['ds'][:n], data_test['temp_min'].values[:n],
label='Mínima', color=colors['temp_min'])
    ax2.fill_between(prophet_df['ds'][:n], data_test['temp_min'].values[:n],
data_test['temp_max'].values[:n],
                     color='#8c564b', alpha=0.1)
    ax2.set_title('Temperaturas Extremas')
    ax2.set_ylabel('Temperatura (°C)')
    ax2.legend()
    ax2.xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
    ax2.xaxis.set_major_locator(mdates.MonthLocator(interval=2))
    ax2.grid(True, linestyle='--', alpha=0.6)

    # Gráfico 3: Humedad Relativa
    ax3 = fig.add_subplot(gs[2])
    ax3.plot(prophet_df['ds'][:n], data_test['humedad'].values[:n],
label='Humedad', color=colors['humedad'])
    ax3.set_title('Humedad Relativa')
    ax3.set_ylabel('Humedad (%)')
    ax3.legend()
    ax3.xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
    ax3.xaxis.set_major_locator(mdates.MonthLocator(interval=2))
    ax3.grid(True, linestyle='--', alpha=0.6)

    # Gráfico 4: Error de Predicción
    ax4 = fig.add_subplot(gs[3])
    error = y_test[:n] - prophet_df['yhat'][:n]
    ax4.bar(prophet_df['ds'][:n], error, width=1, color=np.where(error > 0,
colors['error_pos'], colors['error_neg']), alpha=0.7)
    ax4.axhline(0, color='black', linewidth=0.8)
```

```
    ax4.set_title(f'Error de Predicción (RMSE:
{np.sqrt(np.mean(error**2)):.2f} mm)')
    ax4.set_ylabel('Error (mm)')
    ax4.xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
    ax4.xaxis.set_major_locator(mdates.MonthLocator(interval=2))
    ax4.grid(True, linestyle='--', alpha=0.6)

    # Ajustar el diseño y rotar etiquetas del eje x
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

# Generar datos de ejemplo
np.random.seed(42)  # Para reproducibilidad
n = 60  # Número de puntos a graficar (2 meses con datos diarios)
dates = pd.date_range(start='2025-01-01', periods=n, freq='D')
prophet_df = pd.DataFrame({
    'ds': dates,
    'yhat': 10 + np.random.normal(0, 2, n),  # Predicciones simuladas
    'yhat_lower': 8 + np.random.normal(0, 2, n),  # Intervalo inferior
    'yhat_upper': 12 + np.random.normal(0, 2, n),  # Intervalo superior
})
y_test = 10 + np.random.normal(0, 2.5, n)  # Valores reales de precipitación
data_test = pd.DataFrame({
    'temp_max': 30 + np.random.normal(0, 3, n),  # Temperaturas máximas
    'temp_min': 15 + np.random.normal(0, 2, n),  # Temperaturas mínimas
    'humedad': 70 + np.random.normal(0, 10, n)   # Humedad relativa
})

# Llamar a la función para generar las visualizaciones
plot_prophet_weather_variables(prophet_df, y_test, data_test, n)
```

Figura 49

Visualización detallada de variables meteorológicas del modelo PROPHET



Nota. El gráfico muestra una visualización detallada de variables meteorológicas del modelo PROPHET. Elaboración propia.

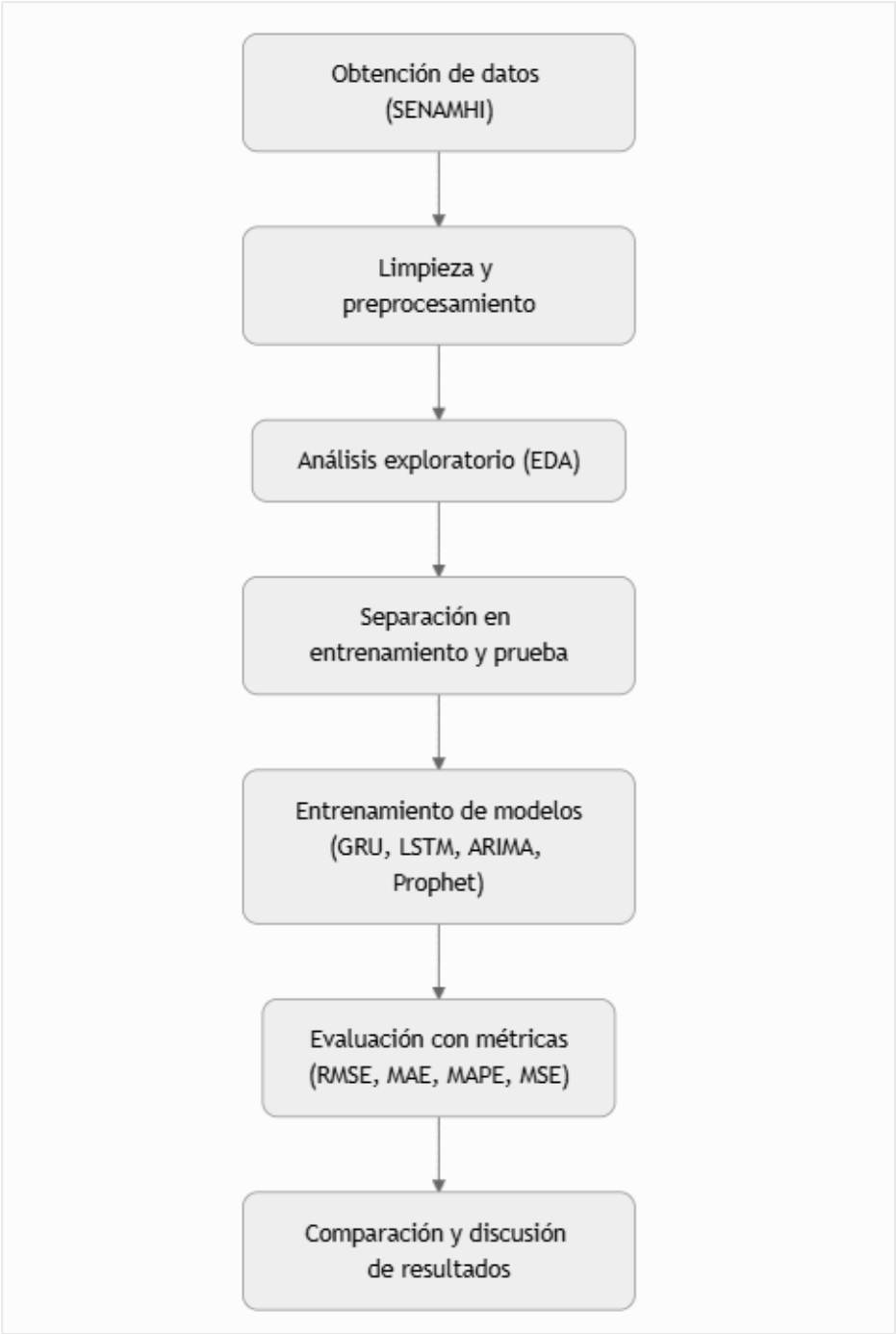
Nota: Todo el código fuente y los archivos necesarios para ejecutar los modelos están organizados y compartidos en una carpeta de Google Drive que se puede acceder:

https://drive.google.com/drive/folders/1IKOKmFDaOKhXcrL-I8nXUVNqpNDNN9Oc?usp=sharing

Anexo 4. Flujograma de la Metodología

Figura 50

Flujograma de la Metodología



Nota. El grafico del proceso del desarrollo de investigación. Elaboración Propia.